Deep learning methods for the analysis of biological electron microscopy images

M.Sc. Kavitha Shaga Devan
From Ipoh, Malaysia

A thesis submitted to attain the degree of Dr. rer. nat. of the Faculty of Engineering, Computer Sciences and Psychology at
Ulm University

2023

Dean of the Faculty of Engineering, Computer Sciences and Psychology: Prof. Dr. Anke Huckauf

Reviewers:
Prof. Dr. Paul Walther
Prof. Dr. Thomas Müller-Reichert
Prof. Dr. Heiko Neumann

Date of Defense: 26th June 2023

Abstract

The analysis of biological structures in electron microscopy (EM) images is an important task that can lead to a deeper understanding of cellular processes and disease development, as it is widely used when researching and diagnosing various diseases such as viral infections and genetic disorders. However, large-scale manual analysis and quantification of EM images is time-consuming, subjective, and error-prone.

Deep learning (DL) techniques have the potential to automate image analysis workflows for objective quantification of biological structures in EM images. Despite this, deep learning has not been fully explored in biological EM, as it often requires large amounts of labelled ground truth datasets. The availability of such data is rather scarce in the biological field, which in turn limits its applications. To overcome this challenge, this thesis focuses on developing DL methods that can classify, synthesize, detect, and segment biological structures in EM images in an effective manner using only small amounts of labelled ground truth data.

The methods developed are intended to provide biologists with an automated quantification and analytical workflow for routine use. To this end, we leveraged transfer learning techniques to detect human cytomegalovirus (HCMV) particles in transmission electron microscopy (TEM) images. Two different transfer learning techniques were investigated to assess their effectiveness and suitability for this task. Our study provided the proof of principle that transfer learning can be applied to the development of DL models that allow for the automatic detection of particles in EM images.

This work was then extended to detect and classify HCMV secondary capsid envelopment stages in TEM images. Since lack of large high-quality labeled ground truth datasets hampers model performance, we introduced a technique to generate synthetic TEM images and self-labelling data as an augmentation method. We could show, that the addition of synthetic data greatly improves the DL models' learning and generalization capability and in turn, improves particle detection. This

work speeds up the development of DL-based classification and detection applications, as it reduces the demands on data labeling.

Finally, we developed a weighted average ensemble method to effectively segment various types of biological structures in both TEM and scanning electron microscopy (SEM) images using very small labelled ground truth datasets. Our approach significantly outperformed the standard single model approach, yielding performance almost similar to expert-labeled data. To make our model interpretable, we added an explainable component in our ensemble model, which visually communicates the model's prediction.

By making these three contributions, we have demonstrated throughout this thesis that it is feasible to automatically detect, synthesize, classify, and segment biological structures in EM images in an effective manner with a reduced number of expert labels. We hope this will help researchers and practitioners optimize routine analytical workflows for biological EM quantification.

Dedication

To my parents, to my sisters, to my husband, to my son, to my beloved pet, to my extended family members, to my friends, and to all those who made this achievement possible. My infinite gratitude to all of you.

We are what our thoughts have made us; so, take care about what you think. Words are secondary, thoughts live. They travel far.

- Swami Vivekananda

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Dr. Paul Walther, for his continuous support, advice and motivation during my Ph.D. study. His guidance and knowledge greatly helped me during the course of my research work. I could not have imagined having a better supervisor and mentor than him. I would like to thank Prof. Dr. Hans A. Kestler for his encouragement, insightful comments, and help with research ideas formulations that enabled me to widen my research perspective. My deepest appreciation to Dr. Clarissa Read for educating me on biological electron microscopy as well as the virology field. She has provided valuable ideas and contributions to my research work. I would like to extend my thanks to Prof. Dr. Timo Ropinski for providing help with the research idea formulations as well as giving constructive feedback. Without their precious support, it would not be possible to conduct this research.

My sincere gratitude to the Baden-Württemberg Stiftung gGmbH for funding my research through the ABEM project. Thank you to Jun. Prof. Dr. Jens von Einem for providing interesting feedback on my work from a virology perspective. I would like to thank my collaborators from the Institute of Media Informatics, Dr. Pedro Hermosilla, and Hannah Kniesel, with whom I work with in the ABEM project. Their insights and ideas have been invaluable to my work. I am grateful my fellow colleagues, Tim Bergner, Dr. Ulrich Rupp, Dr. Mohammed Abdellatif, Andrea Bauer and Carmen Litschel for always providing encouragement and a helping hand in the Central Facility of Electron Microscopy. Also, thank you to Renate Kunz and Reinhard Weih for their technical assistance in the department.

I want to thank previous supervisors and work colleagues in Malaysia, New Zealand and Sweden for encouraging and guiding me throughout my research journey. I thank all my friends who have always been there for me, both personally and professionally. Last but not least my deepest appreciation to my family. Thank you to my husband who has always been supportive in all ways possible. My parents, sisters, and brother in-laws who have always encouraged me to reach for the sky. Finally, thank you to my son, who brings meaning and happiness to my life.

Table of Contents

List of Figures

List of Abbreviations

Artificial intelligence	AI
Charge-couple device	CCD
Convolutional neural network	CNN
Deep learning	DL
Deep neural network	DNN
Electron microscopy	EM
Generative adversarial network	GAN
Gradient-weighted class activation mapping	Grad-CAM
Human cytomegalovirus	HCMV
Machine learning	ML
Recurrent neural network	RNN
Rectified linear activation function	ReLU
Scanning electron microscopy	SEM
Transmission electron microscopy	TEM
Single image super resolution	SISR
Vision transformer	ViT

List of Scientific Publications

The following peer-reviewed journal publications are a part of this thesis.

[SD-1] **Devan KS**, Walther P, von Einem J, Ropinski T, Kestler HA, Read C. Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. Histochem Cell Biol. 2019 Feb. 151(2):101-114. https://doi.org/10.1007/s00418-018-1759-5. Epub 2018 Nov 28. PMID: 30488339

[SD-2] **Shaga Devan K**, Walther P, von Einem J, Ropinski T, A Kestler H, Read C. Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network. Cell Microbiol. 2021 Feb. 23(2): e13280. https://doi.org/10.1111/cmi.13280. Epub 2020 Nov 16. PMID: 33073426

[SD-3] **Shaga Devan K**, Kestler H.A, Read C, Walther P. Weighted average ensemble-based semantic segmentation in biological electron microscopy images. Histochem Cell Biol. 2022. 158:447–462. https://doi.org/10.1007/s00418-022-02148-3

The following peer-reviewed publication is related but not included in this thesis.

[SD-4] Kniesel H, Ropinski T, Bergner T, **Shaga Devan K**, Read C, Walther P, Ritschel T, Hermosilla P. Clean Implicit 3D Structure from Noisy 2D STEM Images. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. pp. 20730-20740. https://doi.org/10.1109/CVPR52688.2022.02010

1. Introduction

1.1 Motivation and challenges

The analysis of biological ultrastructure is an important step towards understanding various pathogens and diseases. Electron microscopy (EM) has evolved into a powerful tool for cell biology (Koster and Klumperman 2003; McIntosh 2001; Graham and Orenstein 2007; Callaway 2020; Assaiya et al. 2021; Flannigan et al. 2010; Fernandez-Leiro, Scheres 2016). It enables scientists to view structures in very high resolution, yielding important insights into cellular processes and disease development (Pelletier et al. 2006; Franken et al. 2020; Hazelton and Gelderblom 2003). EM has been continuously used for the research and diagnosis of various diseases such as viral infections and genetic disorders (Golding et al. 2016; Richert-Pöggeler et al. 2019; Möller et al. 2020, Goldsmith and Miller 2009).

In the past decade, EM has seen an unprecedented increase in the number and size of datasets acquired. However, visual analysis and quantification of organelle morphology in a large volume of images present a substantial challenge (von Chamier et al. 2021; Radulović et al. 2022). Traditionally, the biological quantification process involves the manual analysis of EM images by experts. This analysis is time-consuming, expensive, subjective, and prone to error (Treder et al. 2022). Most often, it is not feasible to perform large-scale manual visual assessments of EM samples and this, in turn, limits their applications in the biological field. To combat this challenge, new algorithmic methods that are less reliant on manual human intervention and expertise are highly desired (Treder et al. 2022). Therefore, there is currently a demand for the design and development of artificial intelligence (AI) based automated image analysis workflows for the objective quantification of electron microscopy images.

Machine learning (ML) is a subfield of artificial intelligence that has gained widespread recognition as an advantageous technique for many computer automated tasks. In recent times, machine learning algorithms have been successfully used for a variety of tasks such as

classification, regression, clustering, classification, translation, dimensionality reduction, detection, recommendation, synthesis, and many others.

Deep learning (DL) is a specialized branch of machine learning that has led to significant progress in the field of computer vision. The main strength of deep learning lies in its ability to extract complex high-level features from raw input data without the need for human intervention (LeCun et al. 2015). This differs from traditional machine learning approaches, which require carefully designed handcrafted features according to current domain-specific knowledge. The incorporation of feature engineering into the learning process has rapidly increased the popularity of DL. The availability of large amounts of data, as well as the continuous increase in computing power, has propelled it to achieve unprecedented performance in a multitude of fields (Chai et al. 2021). Deep learning has outperformed conventional image processing methods for tasks such as image classification, segmentation, object detection, image restoration, and many others (Ede 2021). These achievements have led to an increased interest in DL-based image analysis workflows for tasks that present significant challenges when performed manually.

Recently, DL has demonstrated incredible success in the analysis of microscopy images in the biomedical domain (Ching et al. 2018; Wainberg et al. 2018; Goecks et al. 2020; Berrar and Dubitzky 2021; Greener et al. 2022). Despite its widespread use in light microscopic applications, it has not fully explored in the field of biological electron microscopy. Some of the main challenges that hamper the use of DL for EM images are a lack of labelled ground truth data, low contrast, noise, inhomogeneity, and appearance variability. To overcome the challenges associated with EM-based image analysis, this thesis focuses on leveraging DL methods for the morphological analysis of biological ultrastructure in EM images. In particular, we have focused on detection and classification of HCMV viral particles, synthetic generation of EM images, and the semantic segmentation of organelles with a focus on explainable deep learning networks.

1.2 Objectives

The main aim of this work is to leverage DL techniques for the automated analysis and quantification of biological EM images. While DL has shown astounding success in many

applications, it requires a large amount of labelled data for learning. Manually generating large amounts of labelled ground truth data is time-consuming, subjective, and expensive. Therefore, there is a scarcity of good quality ground truth data for efficient model learning in biological EM. To this end, the focus of this work is to overcome this limitation by developing deep learning methods that are able to classify, synthesize, detect, and segment biological ultrastructure in EM images in an effective manner using only small labelled ground truth datasets. The methods developed are intended to provide biologists with an automated quantification and analytical workflow for routine use.

The following objectives briefly highlight the main steps involved in achieving this aim.

1) Development of transfer learning techniques for the automated detection of human cytomegalovirus (HCMV) particles in transmission electron microscope (TEM) images (Devan et al. 2019) [SD-1].
2) Development of a data augmentation technique for the generation of synthetic TEM images which only requires a low amount of training data using a generative adversarial network (GAN) and the subsequent automated labelling of ground truth data for the detection of multi-class HCMV secondary envelopment stages (Shaga Devan et al. 2021) [SD-2].
3) Development of an ensemble-based semantic segmentation method for automated multi-class segmentation of chromosomes, mitochondria, cytoplasm, and nuclei in both TEM and scanning electron microscope (SEM) images (Shaga Devan et al. 2022) [SD-3].

1.3 Thesis organization

This is a cumulative thesis with the following structure. Chapter 1 introduces the motivations and challenges associated with the application of DL methods for biological EM along with the intended objectives of this work. Chapter 2 provides a brief overview of EM, such as imaging instruments, working principles, advantages and limitations, and sample preparation techniques. Chapter 3 briefly elucidates the concept of artificial intelligence. It mainly focuses on key concepts and terminologies of machine and deep learning, which are used throughout this thesis. The chapter also highlights the various applications of DL in the field of EM and their associated

challenges. Chapter 4 summarizes the overall contribution of this thesis, which is composed of three individual publications in peer-reviewed journals (Devan et al. 2019; Shaga Devan et al. 2021, 2022) [SD-1], [SD-2] and [SD-3]. Chapter 5 contain the reprint of Devan et al. (2019) [SD-1] which describes the detection of herpesvirus capsids in transmission electron microscopy images using transfer learning techniques. This technique in then further expanded in Chapter 6 with the improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network as shown by the reprint of Shaga Devan et al. 2021 [SD-2]. Chapter 7 contain the reprint of Shaga Devan et al. (2022) [SD-3] which describes the weighted average ensemble-based approach for the semantic segmentation of biological structures in electron microscopy images. Individual contributions from the main author, as well as a short synopsis of the work, are given for each publication in Chapter 5, 6 and 7. Finally, Chapter 8 concludes this thesis and provides a future outlook to extend the currently proposed methods for the automation of EM-based biomedical applications in a broader context.

1.4 Publications included in this thesis

This thesis is a cumulation of peer-reviewed journal publications that are listed here in order of their appearance in the thesis.

[SD-1] **Devan KS**, Walther P, von Einem J, Ropinski T, Kestler HA, Read C. Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. Histochem Cell Biol. 2019 Feb;151(2):101-114. https://doi.org/10.1007/s00418-018-1759-5. Epub 2018 Nov 28. PMID: 30488339

[SD-2] **Shaga Devan K**, Walther P, von Einem J, Ropinski T, A Kestler H, Read C. Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network. Cell Microbiol. 2021 Feb;23(2):e13280. https://doi.org/10.1111/cmi.13280. Epub 2020 Nov 16. PMID: 33073426

[SD-3] **Shaga Devan K**, Kestler HA, Read C, Walther P. Weighted average ensemble-based semantic segmentation in biological electron microscopy images. Histochem Cell Biol. 2022. https://doi.org/10.1007/s00418-022-02148-3. PMID: 35988009

2. Electron Microscopy

2.1 Brief history of electron microscopy

The term *microscope* is composed of the Greek words *mikros* and *skopeo*, which mean *small* and *look at*, respectively. In 1931, the first electron microscope prototype was built by Ernst Ruska and Max Knoll at the Technische Universität Berlin, Germany. This prototype was able to produce a magnification of 400 and was the first device of its kind that was able to demonstrate the principles of electron microscopy (EM). The first commercial electron microscope was released to the public in 1938 and since then, it has been an integral part of scientific development (Haguenau et al. 2003).

Due to the much shorter wavelength of electrons compared to photons, the resolution of an electron microscope far exceeds that of a light microscope. Therefore, the electron microscope plays a powerful role in the characterization of a wide variety of biological and non-biological specimens. Its ability to render images at high spatial resolution makes it a very valuable tool for a wide variety of applications. In the biomedical field, EM is commonly used to investigate the detailed structure of tissues, cells, organelles, viral particles, and macromolecular complexes.

2.2 Working principles of electron microscopy

There are two main types of electron microscopes; which are the transmission electron microscope (TEM) and the scanning electron microscope (SEM). The combination of TEM and SEM into a single instrument gave rise to the scanning transmission electron microscope (STEM). TEM is capable of achieving atomic-scale resolution, while SEM magnifies surface features as small as a few nanometers (ThermoFisher Scientific 2022). Therefore, electron microscopes are able to produce detailed images that reveal complex and delicate structures of specimens.

A TEM (Figure 1) is used to view ultrathin samples through which a high voltage beam of electrons can pass through, forming a projection image that relays information about the structure of the sample. Electrons emitted from the electron gun pass through a condenser lens before interacting

with the sample, close to the objective lens. The image of the sample is then magnified by a series of lenses and is recorded when it hits the fluorescent screen or a light-sensitive sensor such as a charge-coupled device (CCD) camera. The image detected by the CCD is displayed in real-time on a monitor or computer (Williams and Carter 1996; Winey et al. 2014).

A SEM projects and scans a focused beam of electrons over a sample surface to generate an image. When the scanned beam hits the sample, backscattered and secondary electrons are emitted from the sample surface due to the excitation from the primary electron beam. These signals that are highly localized to the actual impact point of the primary beam contain information about the sample surface such as its topography and composition. The signals are acquired by detectors and are used to form an image, pixel, by pixel and line by line (Akhtar et al. 2018). Focused ion beam scanning electron microscope (FIB-SEM) is an evolution of SEM, in which a sample block is repeatedly abraded with a focused ion beam (FIB) and the newly exposed surface is imaged with an SEM (Figure 2).This technology produces three dimensional datasets (Kizilyaprak et al. 2014; AZO Life Sciences 2020).

The work presented in this thesis was performed using datasets comprising of TEM and FIB-SEM images.

2.3. Sample preparation techniques

Sample preparation directly impacts the quality of the EM characterization and is therefore the most crucial step in biological EM. The standard protocol is to fix the biological samples with chemical crosslinkers such as glutaraldehyde, dehydrate it, and embed it in a polymer that is very stable in the electron beam of the microscope. Because this leads to structural changes due to osmotic effects, cryofixation has been established as an alternative. Today, cryo-electron microscopy has become the gold standard for sample preparation. It is a method to image frozen and still hydrated biological samples in a transmission electron microscope at cryogenic temperatures, typically below -150°C using stages cooled with liquid nitrogen. This approach prevents dehydration and artifacts related to dehydration that are typical for conventional specimen preparation. The samples are, however, very beam sensitive, which makes observation with the electron microscope difficult (Milne et al. 2013; Cabra and Samsó 2015). The samples used in this thesis have all been prepared by what is called a hybrid method: high pressure freezing, freeze substitution and plastic embedding. This approach combines some advantages of cryo-

transmission electron microscopy (good structural preservation) and conventional preparation with plastic embedding (which makes the sample very resistant to the electron beam) (Walther and Ziegler, 2002).

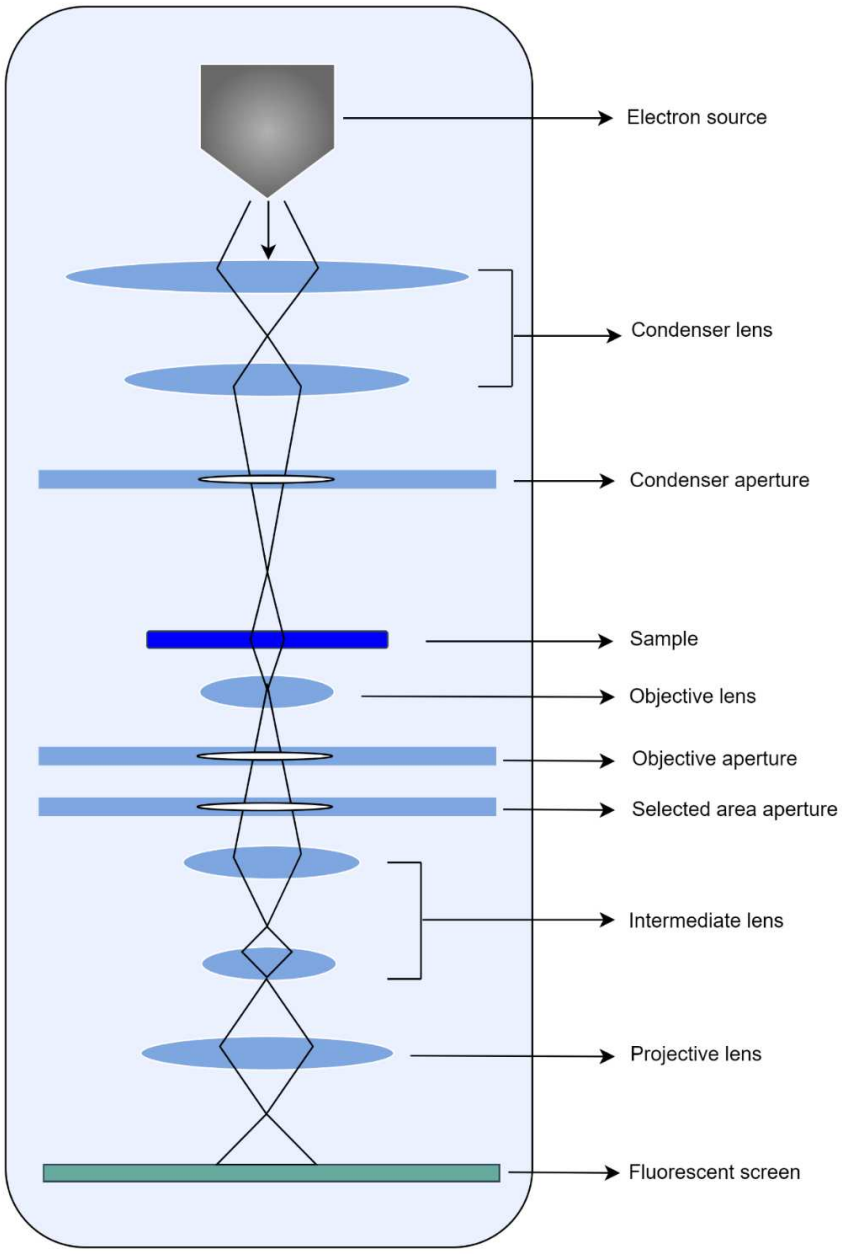


Figure 1. Schematic diagram of a transmission electron microscope (TEM) (Aryal 2022).

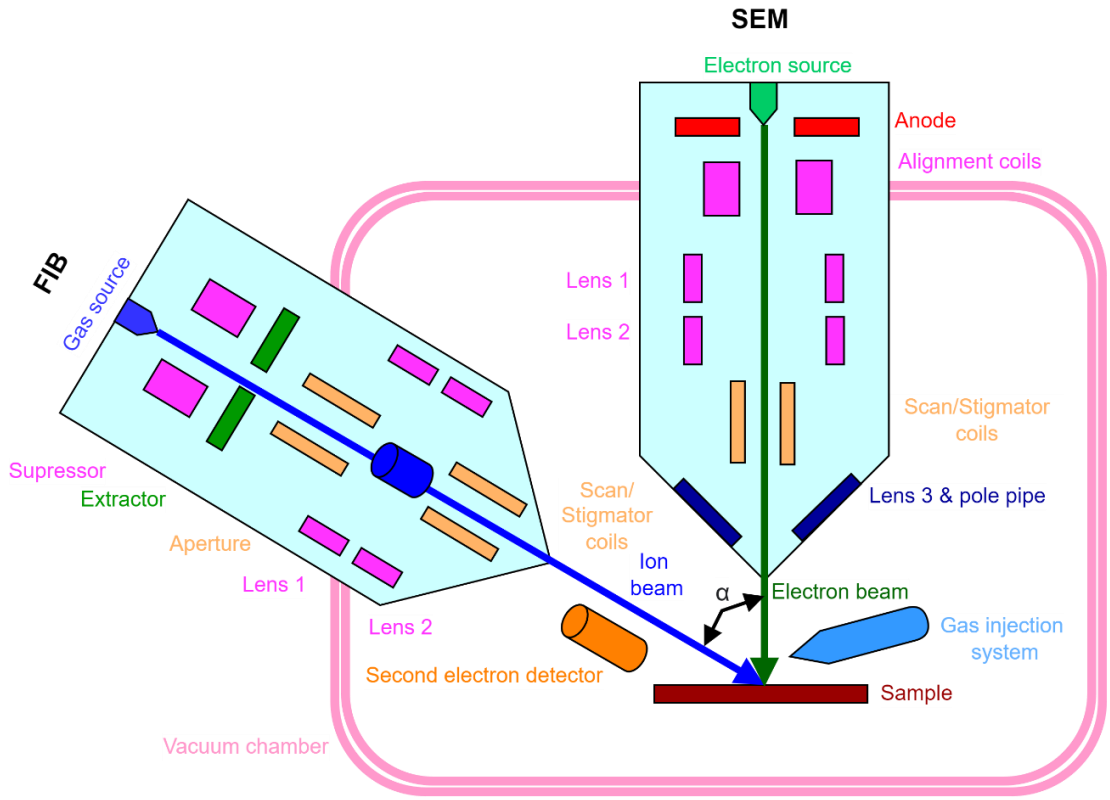


Figure 2. Schematic diagram of a focused ion beam scanning electron microscope (FIB-SEM). The angle (α) between the electron beam and ion beam is normally in the range of 52 and 55° (Yougui Liao 2007).

The crucial first step of this protocol is to freeze the sample in a physiologically defined state. However, this is complex, since water tends to form crystals during freezing that heavily destroy the biological structure. An approach to overcome this limitation is high-pressure freezing. By applying a pressure of 2000 bar to the small sample at the very moment of freezing, the formation of ice crystals is suppressed, since the pressure prevents expansion of the water that occurs when hexagonal ice is formed, since the density of ice is less than the density of liquid water. High-pressure freezing is currently considered the best method for fixation of cells and tissue and has therefore been applied for all samples of the three publications of this thesis (Walther, Schmid and Höhn 2013; Villinger et al. 2014; Villinger et al. 2015).

The frozen samples are then processed by freeze substitution; they are immersed into acetone at a temperature of -90°C and slowly warmed up. During this process, ice is slowly "substituted" by acetone, which is liquid at these cold temperatures. The sample is stabilized by fixatives, e. g. osmium tetroxide and uranyl acetate, which have been added to the acetone. Once the sample has reached room temperature, the acetone is stepwise replaced by the resin Epon, which is then polymerized. The result is a very stable solid-state sample. For TEM, the sample is then cut into very thin slices with a diamond knife for imaging in the transmission mode (Walther, Schmid and Höhn 2013; Villinger et al. 2014; Villinger et al. 2015).

Samples for SEM imaging do not need to be ultra-thin as the beam of electrons does not pass through the sample, but instead, the beam is scanned across the sample for image formation (Akhtar et al. 2018).

3. Deep Learning

3.1 Artificial intelligence

Artificial intelligence (AI) is a broad term for technologies that enable machines to mimic human intelligence (Figure 3). AI uses computer systems to perform complex tasks in a way that is similar to the human problem-solving approach. A major subcategory of AI is machine learning (ML). It refers to a wide variety of methods and algorithms that allow computer systems to make decisions using a given set of data, without being explicitly programmed. This warrants the computer systems to learn on its own, gradually improving performance (Kersting 2018; Janiesch C, Zschech P and Heinrich 2018).

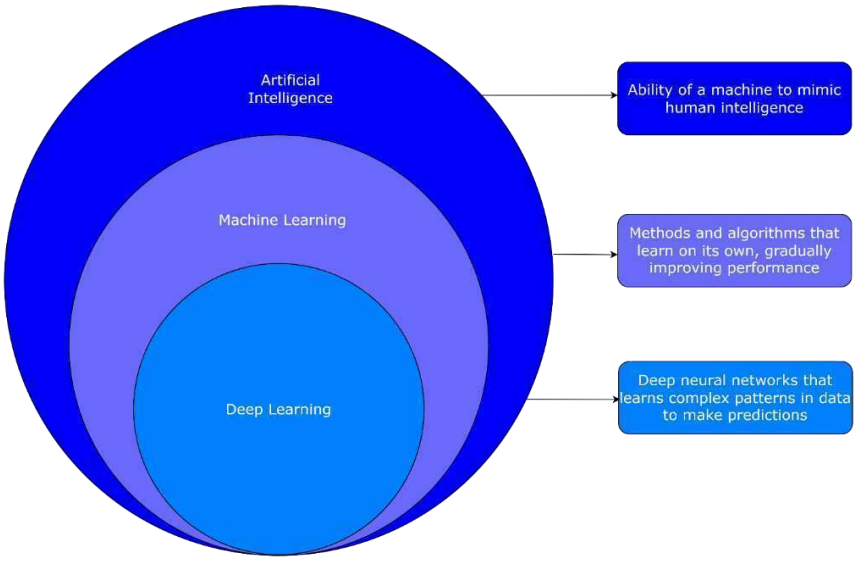


Figure 3. Artificial intelligence is an umbrella term for technologies that are able to mimic intelligent human behavior. Machine learning is a subfield of artificial intelligence and deep learning is a subcategory of machine learning.

There are two main approaches in ML which are; supervised and unsupervised learning. The main difference between these two approaches is defined by the use of training data. Supervised learning algorithms require labelled ground truth data for model training and subsequent prediction. On the

other hand, unsupervised learning algorithms work with unlabelled training datasets. It is able to discover patterns in the data without the need for external intervention.

3.2 Principles of deep learning

Deep learning (DL) is a major subfield of ML. DL models use large artificial neural networks to learn complex patterns in data and make predictions independent of human input (LeCun et al. 2015). It automatically extracts features from the input during the model learning process, eliminating the need for engineering features. In contrast, many classical ML algorithms are highly dependent on human intervention for the feature engineering process. DL is used for both supervised and unsupervised learning. It is primarily credited as the catalyst for accelerating progress in fields such as computer vision, natural language processing, and speech recognition (Sarker 2021).

Artificial neural networks, also known simply as neural networks, are the backbone of deep learning algorithms. Their structure is loosely inspired by the human brain, emulating the way that biological neurons signal with one another. Similar to the interconnection of neurons in the human brain, neural networks also have neurons that are interconnected with each other in various layers of the networks (Figure 4) (Mishra and Srivastava 2014). These neurons are called nodes, and neural networks are composed of multiple layers of nodes. An individual node may be connected to several nodes in the layer before it, with which it receives data, and several nodes in the layer behind it, to which it sends data. Neural networks consist of three types of layers, which are the input layer through which data enters the network, one or more hidden layers where computations are performed, and an output layer which produces the result.

Each of the individual nodes consist of input data, weights, a bias (or threshold), and an output. When an input layer has been determined, corresponding weights are assigned. These weights aid in determining the significance of any given variable, with larger weights having a greater contribution to the output. Next, each input is multiplied by its corresponding weight and added together. The output is then sent through an activation function.

The final output of the node is the activation function of a weighted sum of the node's input as denoted by the equation below,

$$Y = f\left(b + \sum_{i=1}^{n} x_i\, w_i\right)$$

where Y is the output of the node, f is the function, b is the bias, i is the index, x is the input and w is the weight.

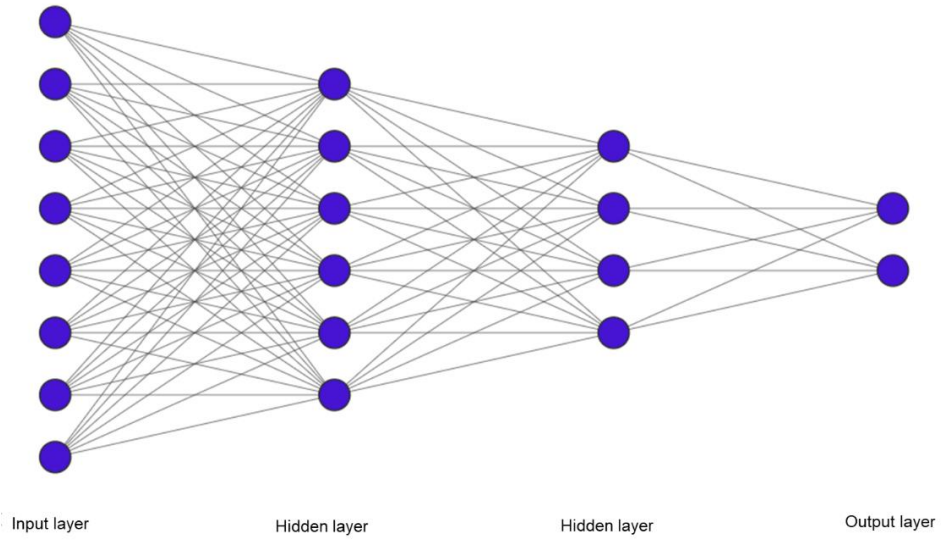


Figure 4. Architecture of a neural network. The neural network architecture is made of individual units called neurons which organization mimics the biological behavior of the brain.

If that output exceeds a certain threshold, the node is activated and the data is forwarded to the next layer of the network. As a result, the input of one node becomes the output of the following node. This neural network is defined as a feedforward network since data is passed from one layer to the next layer throughout the process (IBM, 2020).

Recently, many types of deep learning architectures have been developed due to the continually increasing interest in this field. These architectures include convolutional neural networks (CNN) (LeCun et al. 1998), recursive neural networks (RCN) (Chinea 2009), and generative adversarial networks (GAN) (Goodfellow et al. 2014) and Vision Transformers (Dosovitskiy et al. 2020). However, major advancements in computer vision have been achieved primarily with one

particular DL architecture, the CNN. They are similar to the artificial neural networks described above. The major difference is that CNN contains a three-dimensional arrangement of neurons instead of the standard two-dimensional array.

Common CNN architectures generally comprise three main types of layers; convolutional, pooling, and fully connected (O'Shea and Nash 2015) (Figure 5). The convolutional layers are the core building blocks of the network that gives it its name. In these layers, a convolution process is applied on the receptive field of an input image using a filter to produce a feature map. The layer performs the computation by moving the filter across the receptive fields of an image, checking if a feature is present. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. The calculated value is then stored in the output feature map. The filter is then shifted according to its stride, and the process is repeated until the filter has swept across the entire image. The resulting output from the convolution operation is a two-dimensional array called a feature map.

Each value in the feature map is then typically passed through a rectified linear activation function (ReLU) to include the nonlinearity of the input dataset. The pooling layers are typically applied after the convolutional layer and perform dimensionality reduction to reduce the number of parameters in the input. Similar to the convolutional layer, the pooling operation also moves across the entire input, but it does not contain any weights. The filter applies an aggregation function within the receptive field, in order to obtain the output array. There are two main types of pooling which are the max and average pooling where the maximum and average values are taken respectively, to populate the output array (IBM, 2020). The pooling layer helps to reduce complexity, limit overfitting and improve the efficiency of the CNN (Gholamalinezhad and Khosravi, 2020).In the final fully connected layer, each node is connected directly to a node in the previous layer. Fully connected layers are normally found at the end of a CNN architecture and are generally used to perform classification tasks based on the features extracted through the previous layers and their corresponding filters. It leverages activation functions such as softmax or sigmoid to obtain its predictions (IBM, 2020).

Deep neural networks, in the form of CNN networks with three or more layers, generally perform better than shallow networks. CNN architectures are capable of processing image data, and are designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns (Yamashita et al. 2018). Earlier layers focus on low-level features such as colors and edges. As the image data progresses through the CNN layers, it is able to recognize high-level complex features of the object until it is able to fully identify the object of interest (LeCun et al. 2015).

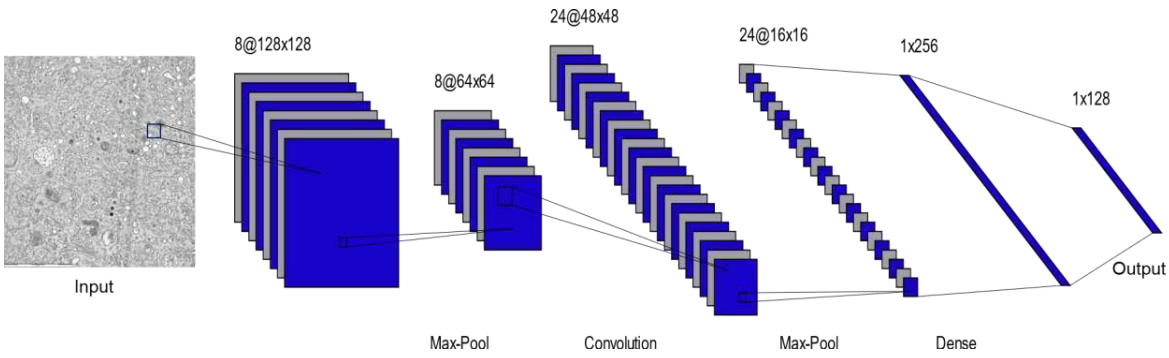


Figure 5. The basic architecture of a LeNet-5 convolutional neural network. Max-pool indicates the pooling layer and dense indicates the fully connected layer (LeCun et al. 1998).

3.3 Applications of deep learning in electron microscopy

In recent times, DL methods have been applied to electron microscopy (EM) image analysis with impressive results (Treder et al. 2022; von Chamier et al. 2021; Ede 2021). It has been shown to significantly outperform classical image processing methods in terms of accuracy, specificity, and sensitivity. DL has been successfully applied for image denoising, restoration, synthetic image generation, object detection, and semantic segmentation tasks (Ede 2021).

EM images inherently suffer from noise. In recent times, denoising algorithms have been applied to these images in order to improve the image signal-to-noise ratio. These algorithms were able to learn from different types of input data such as paired low- and high-quality images (Buchholz et al. 2019; Vasudevan and Jesse 2019; Giannatou et al. 2019; Mohan et al. 2022), unpaired low- and

high-quality images (Wang et al. 2020), low-quality paired images (Lehtinen et al. 2018; Li et al. 2022), and single noisy images for noise removal (Krull et al. 2019; Kniesel et al. 2022).

Automated detection of structures in EM images will be able to improve and optimize the analytical workflow of researchers and practitioners. Single particle picking is a widely used application of DL in EM, with the development of tools such as DeepEM (Zhu et al. 2017), DeepPicker (Wang et al. 2016), FastParticlePicker (Nguyen et al. 2021), DRPNet (Nguyen et al. 2021), PickerOptimizer (Li et al. 2021) and DeepCryoPicker (Al-Azzawi et al. 2020). Apart from that, DL was also utilized for the automated detection of a variety of structures such as immunogold particles (Jerez et al. 2021), viral particles (Ito et al. 2018; Devan et al. 2019; Shaga Devan et al. 2021; Andriasyan et al. 2021; Matuszewski and Sintorn 2021) and nanoparticles (Güven and Oktay 2018; Faraz et al. 2022).

DL requires a large amount of labelled training data for learning. However, manual labelling is extremely time consuming, expensive, requires expert knowledge, and prone to error (Northcutt et al. 2021). Recently, a few researchers have attempted generating synthetic EM images (Kharin 2020; Shaga Devan et al. 2021; Cid-Mejás et al. 2021) and automated labeling of ground truth (Weber et al. 2018; Groschner et al. 2021). In terms of image restoration, efforts have been made to super-resolve noisy and blurry low-resolution EM images using single image super-resolution (SISR) (Suveer et al. 2019; de Haan et al. 2019), point scanning (Fang et al.2021) and deep residual attention networks (Wang et al. 2021).

Segmentation of EM images is a common task in the biological field in order to perform quantification and analysis of structures. Deep learning-based semantic segmentation methods have been recently applied to EM images with great success. Semantic segmentation is the task of assigning a class label to every pixel in an image (Mo et al. 2022; Sehar and Naseem 2022). Applications of semantic segmentation in EM include the segmentation of neurons (Urakubo et al. 2019; Khadangi et al. 2020), nuclei (Shaga Devan et al. 2021; Spiers et al. 2021), mitochondria (Xiao et al. 2018; Khadangi et al. 2020, 2021; Fischer et al. 2020; Franco-Barranco et al. 2021), extracellular vesicles (Gomez-de-Mariscal et al. 2021) and a variety of other structures (Couedic et al. 2020; Abdollahzadeh et al. 2021; George et al. 2021; Akers et al. 2021; Jacobs et al. 2022).

3.4 Deep learning challenges in biological electron microscopy

While DL has demonstrated unparalleled success in many areas, several significant issues plague its application to biological EM data. One common challenge faced is the inherent lack of large-scale labelled ground truth data for model training (Sapoval et al. 2022). The success of DL is highly dependent on the availability of a large amount of training data, commonly ranging from thousands to millions of training examples (Sarker, 2021). Datasets of this magnitude are rather scarce in EM. Apart from that, there are often severe discrepancies between training data distribution and real test-set distribution causing generalization issues.

Training complex DL models usually requires extensive computational resources to achieve state-of-the-art performance. However, these high-level resources are often not available in EM around the globe, making DL undesirable in various applications. Beyond data and computational complexity, recent demands for the explainability and interpretability are a bottleneck that hampers progress. DL models are generally “black-box”, which means their functions are too complicated to comprehend (Rudin 2019). Therefore, they are not easily explainable and interpretable to users. This is a critical issue in the biomedical domain because the question of why a DL model made a certain prediction needs to be addressed in order to evaluate its trustworthiness.

Our work specifically focuses on addressing the lack of large-scale ground truth training data in biological EM. This work is intended to contribute to the existing body of work by introducing methods that can utilize a small amount of labelled ground truth training data for improving model learning capabilities, training efficiencies, generalization, and interpretability of biological EM deep learning models.

4. Summary of Contributions

This thesis focuses on the application of deep learning (DL) methods to biological electron microscopy (EM). The proposed methods in our publications have been developed for the analysis and subsequent quantification of EM images, with a focus on automated analytical workflows. This work is intended to assist researchers and practitioners in the quantification of various biological ultrastructures. We have focused on the development of DL-based models that are able to work with small labelled ground truth training datasets as well as with limited computational resources.

This thesis contributes to the existing body of knowledge in the following ways: First, it provides a fast and efficient method for the detection of human cytomegalovirus (HCMV) capsids in transmission electron microscopy (TEM) images by using transfer learning techniques [SD-1]. This enables the deep learning model to learn with a small amount of training data by leveraging transfer learning where the existent knowledge of networks pretrained with common every-day images was harnessed to be used for the detection of capsids. Second, it focuses on the generation of synthetic EM images as a means of data augmentation for increasing dataset size and diversity. It also introduces a method for the automatic labeling of the secondary envelopment stages of HCMV to serve as ground truth data. These two techniques are incorporated together in a DL framework for the automated detection of HCMV secondary capsid envelopment stages [SD-2]. The third contribution focuses on the semantic segmentation of various biological organelles in both TEM and SEM images using a weighted average-based ensemble model [SD-3].

The peer-reviewed journal publications that form the main contribution of this thesis are denoted as [SD-1], [SD-2] and [SD-3] and are presented as individual chapters (Chapter 5-7). In the following, we give a brief introduction to each publication as well as a brief summary of its content and contributions.

4.1 Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning [SD-1] (Chapter 5)

The *Human betaherpesvirus 5*, also known as human cytomegalovirus (HCMV), is a member of the viral family known as *Herpesviridae* or *herpesviruses*. Depending on the socioeconomic status of a country, HCMV infects between 50 and 100% of the adult population (Cannon et al. 2010). A major characteristic of HCMV is the ability to establish life-long latent infection. Generally, primary infection and subsequent reactivation rarely causes any health problems for healthy individuals. However, it can cause a life-threatening disease in immunocompromised individuals and severe sequelae in newborns as a result of infection during pregnancy (Forte et al. 2020; Jackson et al. 2021). Currently, there are no vaccines available for HCMV and antiviral drugs used, showed poor effectiveness (Andrei et al. 2009; Mercorelli et al. 2011). Therefore, a better understanding of the life cycle could lead to the development of antiviral agents for controlling virus transmissions.

Quantitative analysis of HCMV capsids in TEM images has proven to be an essential step towards this process. Manual analysis is the de facto standard for these analyses, which is generally time-consuming, labor intensive, prone to error, and requires expert knowledge. Therefore, automatic detection of HCMV nucleocapsids in TEM images could help biologists to perform large-scale quantitative studies of virion morphogenesis in an objective and time-efficient manner.

In this work, we used DL techniques to develop an automated pipeline for the detection of HCMV nucleocapsids in TEM images. However, DL requires a large amount of labelled training for effective learning. The availability of such data is often scarce in biological electron microscopy as it is expensive and time-consuming to generate. In order to overcome this limitation, we leveraged transfer learning techniques to detect nucleocapsids in TEM images utilizing only a small labelled dataset for model training.

The process of transfer learning involves utilizing convolutional neural network models that have been trained to perform a specific task, and adapting them to a related but different task by transferring knowledge (Weiss et al. 2016). In our study, we applied this approach to detect HCMV

nucleocapsids in TEM images by using CNN models that were pre-trained on the ImageNet dataset. This dataset consists of over 14 million images belonging to more than 21000 object classes (Deng et al. 2009). Specifically, we utilized pre-trained models including VGG16, ResNet50, and InceptionV3.

VGG16 is a deep neural network that has 16 layers (Simonyan & Zisserman. 2015). It is a large network, with 138 million parameters, and was able to achieve an impressive accuracy of 92.7% in ImageNet. In fact, it was also one of the top performing models in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 competition (Russakovsky et al. 2015). VGG16 is similar to AlexNet in architecture (Krizhevsky et al. 2012), but the authors behind it found that by increasing the depth of the network to 16-19 weight layers and using small convolution filters, they were able to significantly improve its performance compared to previous configurations. The VGG16 network is made up of 13 convolutional layers and 3 fully connected layers.

The concept of using deeper networks to improve accuracy by incorporating more layers was first introduced by the VGG authors. However, this approach can cause the vanishing gradient problem, which occurs when the model's weight cannot change due to an extremely low learning rate. To address this issue, the Residual Network architecture was developed by He et al. (2016). ResNet is short for Residual Network, and ResNet50 is a convolutional neural network with 50 layers, including 48 convolutional layers, one MaxPool layer, and one average pool layer, with 23 million parameters. ResNet is constructed by stacking residual blocks, which allows for the addition of more convolutional layers to a CNN without encountering the vanishing gradient problem by using skip connections. Skip connections bypass certain convolutional layers, transforming a conventional network into a residual network, substantially reducing training time while improving accuracy. ResNet is typically less complicated than VGG and has fewer filters.

InceptionV3 is a CNN network which has 25 million parameters (Szegedy et al. 2016). Unlike previous architectures, Inception network uses wider networks that contain multiple inception modules. These modules extract features from different levels by performing 1x1, 3x3, and 5x5 convolutions within the same module. The filters' output is stacked along channel dimensions and

passed to the subsequent layer of the network. In terms of the number of parameters and computational resources, InceptionV3 has proven to be efficient.

We explored two transfer learning techniques: feature extractor and fine-tuning. In the feature extractor method, pre-trained CNNs were utilized with their original weights, but the classifier portion was removed. The remaining part of the CNNs was utilized as a fixed feature extractor for the training dataset. A new classifier with a fully connected (FC) layer with 1024 nodes and ReLU activation, as well as a second FC layer with a single node output activated by sigmoid function, was added to the end of each CNN. The features extracted by this method represent the last activation maps before the FC layers. The new classifiers were trained on the transmission electron microscopy training dataset for new classifications, with all the convolutional layers of the pre-trained CNN models frozen (Shin et al. 2016). The models resulting from this technique are called VGG16-FE, InceptionV3-FE, and ResNet50-FE.

To address the limited size of the training dataset (only 190 images) and the low similarity to the ImageNet dataset, an alternative approach was used; the fine-tuning technique. This involved removing the classifier portion of the pre-trained CNN models and replacing it with a custom classifier, as outlined in the feature extractor approach. Additionally, some of the higher-level convolutional layers of the CNN were unfrozen and re-trained using the transmission electron microscopy dataset, while the remaining convolutional layers were left frozen (Tajbakhsh et al. 2016). Approximately 25% of the convolutional layers were re-trained, with the remaining 75% remaining frozen. For the VGG16 model, 3 out of 16 convolutional layers were re-trained, while for the ResNet50 model, 12 out of 50 layers were re-trained. Finally, for the InceptionV3 model, 9 out of 48 layers were re-trained. These models were subsequently referred to as VGG16-FT, ResNet50-FT, and InceptionV3-FT respectively.

It was observed that the fine-tuning technique outperformed the feature extractor technique for all the models, with ResNet50-FT emerging as the best performing model with InceptionV3-FT as the second best. In order to further investigate the effect of the fine-tuning approach in order to determine whether a better performance could be accomplished, we utilized a deep fine-tuning method were a higher number of convolutional layers than the ones mentioned above were

unfrozen and retrained. For this, we selected the InceptionV3-FT model and the convolutional layers were increasingly unfrozen and re-trained in a block wise manner. The reason InceptionV3 was selected for deep fine-tuning despite ResNet50 exhibiting the best performance is because generally the smaller the number of learning parameters in a model, the better is its performance on a small dataset as overfitting is prevented. Therefore, generally ResNet50-FT would have given a good performance for deep fine-tuning as it had the lowest number of learning parameters (23 million parameters) compared to InceptionV3 (24 million parameters). However, InceptionV3-FT performed the second best and the Inception-FE model performed the best out of all in the feature extractor method. Therefore, we were curious about its performance when deep fine-tuned and therefore this model was selected for further fine-tuning. It was observed that, as a higher number of convolutional layers were re-trained, the performance of the InceptionV3 model improved. However, this only worked up to a certain number of convolutional layers re-training. At a certain point, we observed that the results became worse as a higher number of convolutional layers were unfrozen. This indicates that there is a trade-off between the number of layers re-trained and the performance of the model.

Our study provides proof-of-principle that transfer learning can be applied for the development of DL models that allow automatic detection of particles in EM images. We have also successfully shown that using transfer learning techniques, it becomes possible to obtain good performance with only a small dataset of labelled ground truth (190 images). With this we have developed the basis for future applications of transfer learning approaches for not only automatic detection but also for the categorization of viral capsids from TEM images.

4.2 Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network [SD-2] (Chapter 6)

The virion of HCMV consists of a dsDNA filled *capsid*, a *tegument* layer, and a lipid bilayer as a *viral envelope*. The secondary envelopment process refers to the completion of virion morphogenesis in a virally regulated process, in which the capsids acquire their envelope. Currently, TEM is the gold standard for imaging secondary envelopment process and it has been proven crucial for HCMV morphogenesis studies (Read et al. 2019).

There are three stages of the secondary envelopment process: (a) capsids without contact to a bending membrane (naked capsids), (b) capsids in the process of being enwrapped by a membrane (budding capsids), and (c) capsids that are enclosed by the membrane envelope (enveloped capsids) (Read et al. 2019). Quantitative analysis of the capsids and the three envelopment stages require a large number of images to be manually analyzed which is rather tedious and time-consuming. Therefore, the analysis of the HCMV secondary envelopment stages could greatly benefit from automation. In [SD-1], we presented the first step towards automatic quantification of the three stages of secondary envelopment in TEM images by showing that automatic detection of virus capsids, irrespective of their envelopment stage, is possible even with a small ground truth training dataset when a transfer learning approach is utilized. We now further extend this approach for the automated detection and classification of HCMV capsids into the three secondary envelopment stages.

In this work, we introduced a new data augmentation method that is able to increase training dataset size, improve data diversity, and balance data classes. As high-quality labelled ground truth training data is scarce in the biological EM field, synthetic training data will be beneficial for model learning. A CNN based generative network was used to generate synthetic TEM images as a form of data augmentation to increase the number of ground truth training data for each capsid class.

In 2014, Goodfellow et al. introduced Generative Adversarial Networks (GANs). GANs are a type of generative modelling approach that can learn patterns in input training data and create new synthetic data that appear realistic. The GAN is made up of two main parts: the generator and discriminator. The generator is a neural network that produces fake data, while the discriminator is another neural network that tries to differentiate between real and fake input data. The generator's aim is to deceive the discriminator and learn from more data to create synthetic outputs that look more realistic.

The architecture of GAN is adversarial in nature, where the generator and discriminator work against each other with opposing objectives. The generator strives to produce data that looks

realistic, while the discriminator aims to effectively distinguish between real and fake data. This creates a competitive environment where both entities improve their abilities over time. The construction of GANs follows a minimax game approach, where the generator aims to minimize the loss function value (V) and the discriminator aims to maximize it.

$$min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log 1 - D(G(Z)))]$$

The generator's output when it receives input noise z, the discriminator's probability that the original data x is real, and the discriminator's probability that a synthetic sample $G(z)$ of data is real are denoted by $G(z)$, $D(x)$, and $D(G(z))$, respectively. The mean likelihood over all original data and synthetic data is represented by E_x and E_z, respectively. The discriminator (D) focuses on maximizing its likelihood to achieve the correct prediction (real or synthetic) during the classification of the input data, while the generator (G) focuses on generating synthetic data that is able to fool the discriminator, according to the loss function used during the training process.

For synthetic data generation in this study, a particular type of GAN known as SinGAN (Shaham et al. 2019) was utilized. This generative model can capture the internal statistics of an image and is unconditional. The training samples are patches of a single image, and the SinGAN network captures the statistics of the image structures at different scales, including the global properties of the images, fine details, and textural information. To accomplish this, the SinGAN consists of a hierarchy of patch-based GANs, with each responsible for capturing the statistical distribution in the patch at different scales. The GANs have small receptive fields, preventing them from memorizing the entire image, and each generator produces a realistic image with respect to the patch distribution in the corresponding image using the adversarial approach.

During the training of SinGAN, noise is injected at every level, starting from the coarsest to the finest scale. Each generator and discriminator have the same receptive field, which allows them to capture finer details as the generative process progresses. Once the GAN is trained, it is fixed and training moves to the next level to generate realistic-looking synthetic images. The training loss is a combination of adversarial and reconstruction loss. The adversarial loss penalizes for differences between the patch distribution of synthetic and generated images, while the reconstruction loss enables the existence of a specific set of noise maps that can create the original image.

Using an unconditional generative model, we were able to learn from only a single image to produce multiple synthetic images. In this work, a single image was used to generate 50 synthetic images. A total of 10 original images were used to generate 500 synthetic electron microscopy images. A self-labeling technique was also used in conjunction with the TEM synthetic images for fast and automated labelled ground truth generation for HCMV capsid envelopment stage detection.

To train the CNN based Faster R-CNN detector (Ren et al. 2017) for detecting the HCMV capsid envelopment stages, synthetic images were generated and combined with the original training images. The Faster R-CNN was trained using real ground truth data (350 images) and synthetic images (500 images). It acts as an object detector, identifying the three envelopment stages. The Faster R-CNN consists of a region proposal network (RPN), a CNN backbone, an ROI pooling layer, and fully connected layers with two branches for classification and bounding box regression networks. The RPN generates region proposals, which reduces computational time and improves feature representation by allowing the region proposal stage to share layers with the subsequent detection stages.

Two experiments were conducted where varying ratio of original image and synthetic images were fed into the detector as training data in order to understand the effect of synthetic data augmentation on model performance. In the first experiment, synthetic images were added in an incremental manner to original ground truth dataset for model training. In the second experiment, the reverse was performed, where original ground truth data were incrementally added to the synthetic image dataset for model training. Synthetic images were only utilized for training purposes while original images were used for both validation and testing. A holdout set of test images were used for testing the performance of the detector. The average precision (AP) was used as the evaluation metric and indicates the average AP for all the three classes.

These experiments were conducted to investigate the effect of synthetic images on the HCMV capsid detection model's prediction. In the first experiment, it was observed that training the model with Faster R-CNN model with only original 315 ground truth images gave a reasonable but a non-satisfactory performance. However, the performance of the model increased linearly as synthetic

images were incrementally added to the training dataset. In the second experiment, training the model with only 500 synthetic images resulted in extremely poor performance. However, when original ground truth images were incrementally added to the synthetic training dataset, the performance of the model also increased in a linear fashion.

Based on these results, we can summarize that in situations where there are only small limited training datasets, data augmentation using synthetic image generation could be highly beneficial. Using a small purely original ground truth dataset, a reasonable performance was possible, however this was not satisfactory. The addition of synthetic images acted as a booster in order to improve the performance of the model. This is because, the Faster R-CNN was able to obtain additional low-level feature information from the added synthetic images thereby boosting model performance. We found that original ground truth data is highly important for model performance. Model trained with only synthetic images gave extremely poor performance. This indicates that the model was not able to learn useful discriminative features from them. The model was able to obtain high level feature information from original images and additional supplementary low-level feature information from the synthetic images for improved performance. Therefore, it can be deduced that original ground truth data constitute the fundament of the detection model and a high ratio of original ground truth images is the most crucial element for model performance.

However, the addition of synthetic data can be effectively used as a data augmentation method to improve the model's learning and generalization capabilities and in turn, increases its detection capability in situation with limited labeled training dataset. In a nutshell, we believe that our work speeds up the development of deep learning-based classification and detection applications as it removes the need for large data labeling efforts.

4.3 Weighted average ensemble-based semantic segmentation in biological electron microscopy images [SD-3] (Chapter 7)

Advancements in the field of EM have enabled the acquisition of large volumes of data. Biologists have routinely used segmentation methods to quantify the morphological parameters of organelles and cell structures in electron micrographs. However, manual segmentation of large EM image

datasets is time-consuming and prone to errors and bias. While there are many segmentation tools available, a significant amount of user interaction is required for corrections and quality control. Therefore, there is a demand for the development of accurate and efficient automated segmentation tools for the quantification of these images.

We have previously utilized DL methods for the classification and detection of HCMV capsids and their secondary envelopment stages [SD-1] and [SD-2]. In this work, we extend the application of deep learning in biological EM toward the development of semantic segmentation models for biological ultrastructures. Inherent to DL, most of the currently available semantic segmentation methods require a large amount of labelled training data in order to perform and generalize well. However, manually labeling large amounts of data with pixel-wise accuracy is not feasible.

To overcome this limitation, we have exploited the concept of ensemble-based deep learning to improve segmentation performance. Our proposed method utilizes a CNN-based weighted averaging ensemble approach that can learn complex discriminant features from a small labelled training dataset to segment biological structures in both TEM and SEM images. We were able to perform multiclass semantic segmentation of biological structures such as cytoplasm, nuclei, mitochondria, and chromosomes in the images with a very small amount of training images.

To achieve multiclass segmentation of EM images, we used a weighted average ensemble model consisting of U-Nets (Ronneberger et al. 2015). Each U-Net was trained with a different pretrained network, and their predictions were combined through a weighted average technique. Our ensemble model comprised of five individual base-learners, all constructed using the popular U-Net architecture widely used for semantic segmentation applications. This architecture has an encoder network that includes convolution blocks followed by maxpool downsampling, designed to encode the input image into feature representations at different levels of the U-Net. The decoder network projects discriminative features learnt by the encoder onto the pixel space to obtain classification. The decoder comprises of concatenation and upsampling followed by convolution operations.

For this project, we replaced the standard encoder with pre-trained CNN networks that act as pixel classifiers. These networks were trained on ImageNet (Deng et al. 2009) and included ResNet34 (He et al. 2016), InceptionV3 (Szegedy et al. 2015), VGG19 (Simonyan and Zisserman 2015),

SeResNet34 (Hu et al. 2019), and EfficientNet-B4 (Tan and Le 2020), resulting in five base-learners. Each of these pre-trained networks has its own unique architecture, number of parameters, and model complexity. In order to address the vanishing gradient problem, we utilized skip connections in the U-net architecture to allow gradients to flow freely through the model. These skip connections were also used to concatenate the encoder's features with the same scale features of the decoder, compensating for any loss of spatial information during the downsampling process of the encoder. We made sure to use the appropriate number of layers in both the encoder and decoder to regain the input images' resolutions. To combine the varied architectures of the pre-trained models with the decoder block of the U-Net, we used the Segmentations Models library (Iakubovskii 2019) with Tensorfow 2.2.0. Due to the differences in their architectures and model complexity, the five base-learner models have varying learning and generalization capabilities on the training data.

Seven different electron microscopy image datasets were used to train five base-learners, and their performance based on the Jaccard Index and F1 score was carefully monitored. The three top-performing models for each dataset were then chosen to be combined into a weighted average ensemble model. To create the ensemble model, a grid search was conducted to find the optimal weight combination of the base-learners in order to achieve the best possible accuracy in terms of evaluation metrics. Individual base-learner performance was used to select the weights, with larger weights indicating better-performing models. The optimal weight combination for the ensemble model was determined to be one that outperformed any individual base-learner and an ensemble that used equal weights (Liashchynskyi and Liashchynskyi 2019). The weighted sum was calculated by multiplying the optimized weight for each learner with the corresponding base-learner's prediction and summing it for all learners. The final classifier output was determined by averaging the weighted sum to obtain the weighted average prediction.

Our weighted average ensemble significantly outperformed the standard single convolutional neural network model approach yielding performance almost similar to expert labelled ground truth. Our approach was able to give better predictive performance since it produces a lower error rate and reduces variance. By combining multiple diverse learners, more information can be captured of the underlying structure which can significantly improve the accuracy of the prediction. In order to make our model interpretable, gradient-weighted class activation

mapping (Grad-CAM) visually demonstrates how the ensemble model achieved the segmentation prediction obtained in this work. Grad-CAM was able to show regions in an image that is important for the prediction of the various biological segmentation classes.

While multiple machine learning based-ensemble approaches have been proposed in literature (Khadangi et al. 2021; Haberl et al. 2018; Ghosh et al. 2021; Baskaran et al. 2022; Yin et al. 2022), our work is one of the very first ones that have used this approach in biological electron microscopy focusing specifically on biological structures and multiple electron microscopy imaging modalities. In this work, we have combined multiple CNN networks with varying architectures, model complexities, learning parameters as well as learning and generalization capabilities. We have analyzed the effect of these various combination of ensemble models on a variety of different biological structures for both transmission and scanning electron microscopy images. We believe that this could help biologists in determining the best models for various multi-class classifications and segmentation tasks for a variety of biological structures. Taken together, this work leveraged ensemble-based learning for the semantic segmentation of EM images using a very small labelled training dataset. Therefore, we believe that our approach can lead to the rapid development of deep learning-based tools for biological EM.

5. Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning [SD-1]

This chapter has been published as a peer-reviewed journal paper.

Devan KS, Walther P, von Einem J, Ropinski T, Kestler HA, Read C. Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. Histochem Cell Biol. 2019 Feb;151(2):101-114. https://doi.org/10.1007/s00418-018-1759-5. Epub 2018 Nov 28. PMID: 30488339.

Synopsis: This work develops an end-to-end deep learning network to automatically detect herpesvirus capsids in transmission electron microscopy images using transfer learning.

Contributions of thesis author: Algorithm design and implementation, computational experiments and manuscript writing.

Supplementary materials of the publication can be accessed at the above mentioned DOI link.

Histochemistry and Cell Biology (2019) 151:101–114
https://doi.org/10.1007/s00418-018-1759-5

ORIGINAL PAPER

Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning

K. Shaga Devan[1] · P. Walther[1] · J. von Einem[2] · T. Ropinski[3] · H. A. Kestler[4] · C. Read[1,2]

Accepted: 24 November 2018 / Published online: 28 November 2018

Abstract

The detailed analysis of secondary envelopment of the *Human betaherpesvirus 5*/human cytomegalovirus (HCMV) from transmission electron microscopy (TEM) images is an important step towards understanding the mechanisms underlying the formation of infectious virions. As a step towards a software-based quantification of different stages of HCMV virion morphogenesis in TEM, we developed a transfer learning approach based on convolutional neural networks (CNNs) that automatically detects HCMV nucleocapsids in TEM images. In contrast to existing image analysis techniques that require time-consuming manual definition of structural features, our method automatically learns discriminative features from raw images without the need for extensive pre-processing. For this a constantly growing TEM image database of HCMV infected cells was available which is unique regarding image quality and size in the terms of virological EM. From the two investigated types of transfer learning approaches, namely feature extraction and fine-tuning, the latter enabled us to successfully detect HCMV nucleocapsids in TEM images. Our detection method has outperformed some of the existing image analysis methods based on discriminative textural indicators and radial density profiles for virus detection in TEM images. In summary, we could show that the method of transfer learning can be used for an automated detection of viral capsids in TEM images with high specificity using standard computers. This method is highly adaptable and in future could be easily extended to automatically detect and classify virions of other viruses and even distinguish different virion maturation stages.

Keywords Human cytomegalovirus · *Human betaherpesvirus 5* · Secondary envelopment · Transfer learning · Artificial intelligence · Transmission electron microscopy · Automated image analysis

Introduction

The *Human betaherpesvirus 5*, also known as human cytomegalovirus (HCMV), is an important human pathogen that is distributed worldwide. Depending of the socioeconomic status of a country, 50–100% of the adult population are latent carriers of the virus. After primary infection, HCMV establishes a latent infection from which it can reactivate. While infection of healthy individuals is mostly asymptomatic, HCMV can cause life-threatening disease in immunosuppressed individuals and severe sequelae in newborns as result of infection during pregnancy. Currently, there is no licensed HCMV vaccine available and the currently used antiviral drugs show long-term toxicity, poor oral bioavailability and can be overcome by the development of drug resistances (Andrei et al. 2009; Mercorelli et al. 2011). Therefore, a better understanding of the HCMV life cycle is an important step towards development of new antiviral agents that can control virus transmission.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00418-018-1759-5) contains supplementary material, which is available to authorized users.

✉ P. Walther
paul.walther@uni-ulm.de

[1] Central Facility for Electron Microscopy, Ulm University, Ulm, Germany

[2] Institute of Virology, Ulm University Medical Center, Ulm, Germany

[3] Institute of Media Informatics, Ulm University, Ulm, Germany

[4] Institute of Medical Systems Biology, Ulm University, Ulm, Germany

All members of the herpesvirus family are characterized by a linear double-stranded DNA genome packaged within an icosahedral capsid, a tegument layer and a host cell derived viral envelope (Mocarski et al. 2007). Packaging of viral DNA into preformed capsids occurs in the nucleus, resulting in a nucleocapsid. The nucleocapsid needs to be translocated into the cytoplasm where it acquires its tegument layer and envelope in a process called secondary envelopment, resulting in the generation of infectious virions. HCMV secondary envelopment only occurs in a virally induced perinuclear region in the cytoplasm, the cytoplasmic viral assembly complex (cVAC) (Schauflinger et al. 2013). There partially tegumented capsids establish contact to cellular vesicles and start budding into those. The vesicle membrane then wraps around the tegumented capsid, resulting in the complete enveloped virion. The complete virion inside the vesicle is then transported towards the plasma membrane where it is released from the cell.

Quantitative analysis of secondary envelopment from transmission electron microscopy (TEM) images of the viral assembly complex has been proven essential to determine the role of viral proteins for this process (Cappadona et al. 2015; Dietz et al. 2018; Schauflinger et al. 2011). Quantification of different stages of secondary envelopment can reveal defects in this process. It could be shown recently that the HCMV tegument protein pUL71 is required for efficient secondary envelopment because in its absence the majority of nucleocapsids at the cVAC have not finished the process (Schauflinger et al. 2011). While these quantitative analyses are very informative, they are generally labor intensive, time consuming and prone to subjective decisions and thus require highly skilled personal. Therefore, an automatic detection of nucleocapsids in TEM images would be a milestone towards a better applicability of quantitative studies of virion morphogenesis.

There are only few reports about the automatic detection of virus particles in TEM images. Detection of viral capsids in those studies has been performed mostly using computer-aided detection techniques, which generally make use of feature extraction of color, shape and texture information in combination with machine learning classifiers. Methods such as radial density profile (Kylberg et al. 2012; Sintorn et al. 2004), linear deformation analysis (Ryner et al. 2006) and discriminative textural indicators were used to detect the viral particles in TEM images (Proença et al. 2013).

A major problem with viral capsids, however, is that they change their appearance during the maturation process. In addition, their appearance in TEM images varies considerably depending on their spatial projection, the sample preparation protocol and the settings used for image acquisition. Thus, the development of computer-aided detection methods for accurate capsid detection requires a set of well-defined hand-crafted features, such as texture, contrast, size, gray level values and others. Proper characterization of these features is difficult as TEM images are often corrupted by noise, blurring and illumination issues. Apart from that, finding a robust set of features that ideally represents all the characteristics of the nucleocapsids is a challenging task that requires a high-level of specific knowledge.

Deep learning has the potential to overcome these limitations. It is a subfield of the machine learning and artificial intelligence field. Convolutional neural network (CNN) is a type of deep learning method that eliminates the need to determine a set of features manually as it enables the automatic acquisition of features from the objects of interest. It has produced state-of-the-art results in the field of computer vision and image classification (LeCun et al. 2015). The effectiveness of CNNs in the field of electron microscopy (EM) has already been demonstrated by a few studies.

CNNs have been used to detect feline calicivirus particles in TEM images (Ito et al. 2018) as well as secondary structure elements of proteins from 3D cryo-EM images (Li et al. 2016). Automated segmentation methods based on CNNs were used to segment mitochondrial membranes and neural structures in TEM images (Roels et al. 2017). Zhu et al. (2017) developed an automated particle extraction method from raw cryo-EM micrographs in the absence of a template. The efficiency of CNNs, however, relies heavily on the availability of a large amount of training data ranging from thousands to millions of images (Bengio 2012; LeCun et al. 2015). A limited amount of training data leads to overfitting where the model learns the details but also the noise of the training data to the extent that it inhibits the performance of the model on a new set of data. A small training dataset may also lead to the model not being able to generalize the features learnt well, resulting in poor performance when the model makes predictions on unseen data (Bengio 2012).

In the field of EM, the availability of training data is, however, typically quite limited because it requires time-consuming sample preparation, image acquisition, manual labelling and therefore specific expertise. One approach to overcome these issues of CNNs for automated image classification is the concept of transfer learning. Transfer learning is a technique in which neural networks are used that has proven to do well on a specific classification task (base task). What is learned from that task is then adapted to a separate but potentially related task (target task) (Yosinski et al. 2014). In our work, the base task is the classification of common natural world images, such as images of animals, vehicles or people and the target task is the detection of HCMV nucleocapsids in TEM images. This strategy utilizes the fact that many different datasets of images share important characteristics and features which can be transferred between them. This allows building of powerful image classification systems using a small training dataset.

In recent years, the transfer learning method has been widely used to detect and classify breast cancer (Ribli et al. 2018), skin cancer (Esteva et al. 2017), lung diseases (Shin et al. 2016), retinal lesion (Lam et al. 2018), colonic polyps (Ribeiro et al. 2016), brain lesion (Ghafoorian et al. 2017) and coronary artery tissue (Abdolmanafi et al. 2017). Concerning the field of EM, only a few studies have investigated the applicability of transfer learning for classification purposes. Sousa et al. (2015) used a transfer learning framework based on stacked denoising autoencoder for the recognition of immunogold particles in TEM images. Transfer learning was also used for the automatic categorization and labelling of images obtained by scanning electron microscopy (Modarres et al. 2017).

In this work, we investigated the applicability of different transfer learning methods for the automated detection of HCMV nucleocapsids in TEM images. We tested how well low-level features from a non-electron microscopy domain learned by the CNNs can be adapted to detect virus capsids in TEM images. This was done by two transfer learning approaches which were using pre-trained CNN models as a feature extractor and then fine-tuning them. We initially shallow fine-tuned the CNN models and the best performing model was layer-wise deep-tuned to find out the amount of tuning necessary to obtain a good capsid detection accuracy. To validate the effectiveness of the transfer learning approach, we also built and trained a CNN without any feature transfer and compared its performance to our transfer learning models. As a result, we show that state-of-the-art pre-trained CNN models could be used without creating and training a deep CNN from scratch. This enables us to use the computational resources available in a standard EM laboratory environment. With this we have developed the basis for future applications of transfer learning approaches not only for the automatic detection but also the categorization of virus capsids from TEM images.

Materials and methods

TEM specimen preparation

TEM samples were prepared as previously described (Schauflinger et al. 2013; Villinger et al. 2014, 2015; Dietz et al. 2018). Briefly, human foreskin fibroblasts (HFFs) were maintained in minimal essential medium (Gibco, ThermoFisher Scientific Inc., Waltham, MA, USA) supplemented with 10% fetal calf serum (Biochrom AG, Berlin, Germany), 2 mM L-glutamine (PAA Laboratories GmbH, Pasching/Linz, Austria), 1% penicillin–streptomycin (PAA Laboratories) and 1% non-essential amino acids (Biochrom AG). HFFs were seeded on carbon-coated sapphire discs (Engineering Office M. Wohlwend GmbH, Sennwald, Switzerland). After attachment of the cells for 24 h, HFFs were infected with different recombinant versions of the HCMV bacterial artificial chromosome (BAC) clone TB40-BAC4 (Sinzger et al. 2008) with a multiplicity of infection of 1 and incubated for 3 or 5 days at 37 °C and 5% CO_2. Infected cells on sapphire disks were prepared for TEM by high-pressure freezing, freeze substitution and Epon embedding as described. Ultrathin sections were cut parallel to the sapphire disk and mounted on formvar and carbon-coated single slot grids. The sections were obtained from nine Epon blocks from six independent experiments.

Data acquisition and processing

Ultrathin sections were analyzed with a Jeol 1400 TEM (Jeol, Tokyo, Japan) at an accelerating voltage of 120 kV and the images were digitally recorded with a Veleta CCD camera (Olympus, Münster, Germany). The resolution of the TEM images ranges from a pixel size of 2.78 nm to a pixel size of 0.33 nm. The TEM images were 8-bit grayscale TIFF with a dimension of 2048×2048 pixels. A total of 190 images were used for training and validation of the network and 21 images were set aside as an independent test set.

Naturally, EM images contain a large variety of different structures (Fig. 1) of which an algorithm would have to recognize the structure of interest. An issue with EM data very often is that the same structure appears differently in different images. Therefore, we aimed to approach this with a convolutional neural network, where the algorithm self-learns characteristics of nucleocapsids from a training dataset. This approach is superior to manual definition of nucleocapsid features. To generate the training dataset, we manually annotated nucleocapsids in our training TEM images.

First, 1623 HCMV nucleocapsids were manually labelled in 190 TEM images by annotating the capsid center. The annotated nucleocapsids comprised all the three categories of the secondary envelopment process (naked, budding and enveloped nucleocapsids). Based on the images with the annotated capsid centers, capsid patches with a size of 128×128 pixels were extracted from the TEM images. The patches enclosed the entire capsid. To account for a wide variety of non-capsid regions which include structures such as endosomes and other cellular compartments also non-capsid patches were extracted from the TEM images. The non-capsid patches were randomly extracted from the non-annotated regions in the TEM images. These both types of capsid patches make up the two class labels in our work.

In result, the training data contained 1623 capsid patches (Fig. 2, left) and 1623 non-capsid patches (Fig. 2, right). An equal number of capsid and non-capsids patches were used to avoid bias towards the majority class during training. We further increased the size of the training dataset using data augmentation methods (Krizhevsky et al. 2012).

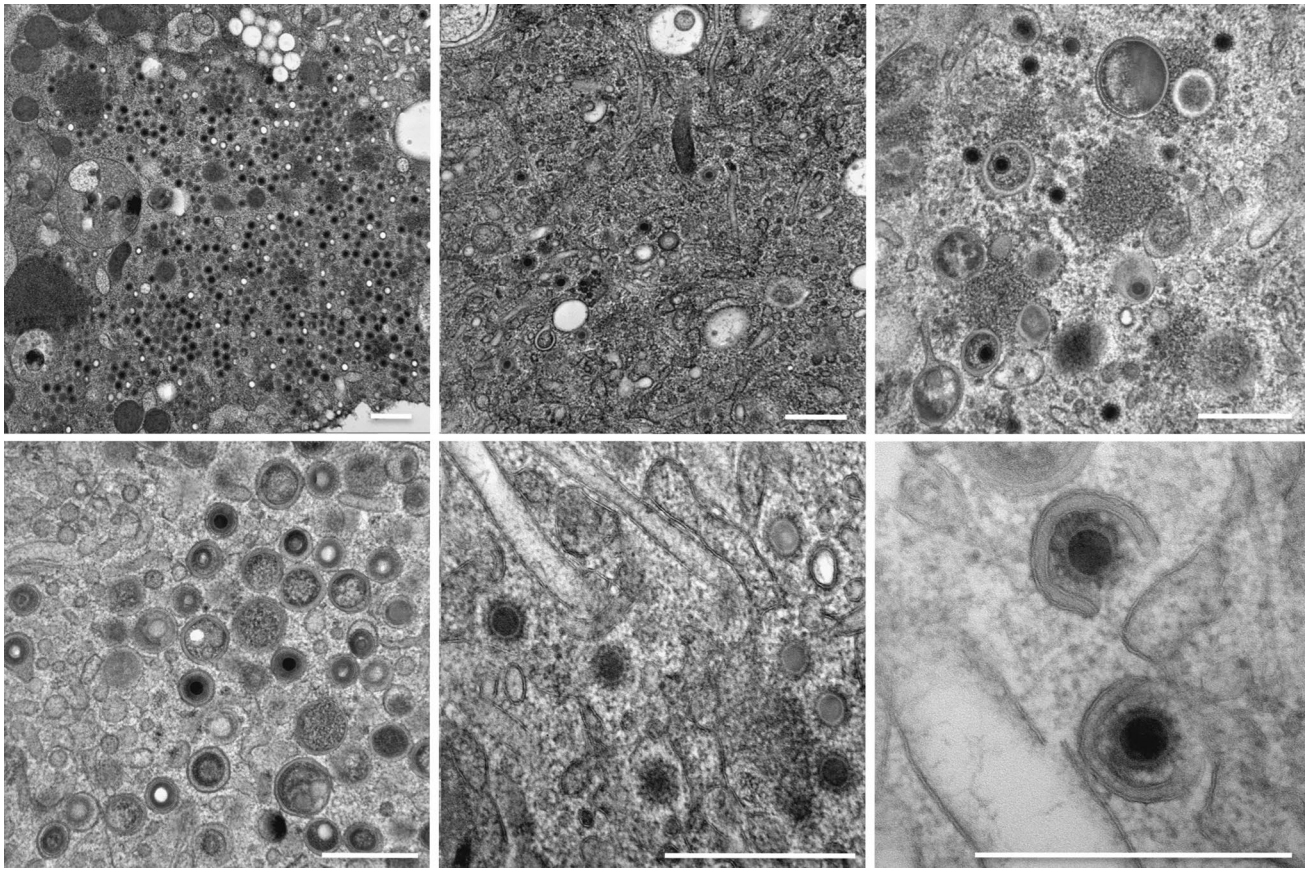


Fig. 1 Examples for TEM images of HCMV-infected cells from our dataset illustrating the variety of capsid and non-capsid structures. Also note the variations in HCMV nucleocapsid appearance, contrast, magnification levels and noise that would make it difficult to define a good set of discriminative features for classification (bars 500 nm)

The augmentations performed on the patches were rotation (0°–180° in increment of 20°), shearing (factor 0.2), vertical and horizontal flipping, vertical and horizontal shifting (factor 0.2), scaling (factor 0.2) and contrast limited adaptive histogram equalization. This resulted in a total of 55,182 training patches.

CNN trained from scratch

We first created a CNN (CNN-TFS) that we trained from scratch (TFS) without transfer learning. The CNN-TFS consisted of four layers and was based on the work of LeCun et al. (1998). The network architecture is depicted in Fig. 3. The network configurations for our CNN-TFS such as image resolution, network depth, filters' size for each individual layer, and the batch size were determined experimentally using our training data to maximize the detection rate and to allow use of the available technical resources in an EM lab. Optimized results were achieved by setting the filter size for all the convolutional layers to 3×3 and the maximum pooling size to 2×2. The patches were resized to 52×52 pixel images and used as the input of the first convolutional layer with 32 filters and a rectified linear units (ReLU) activation (Krizhevsky et al. 2012). This was followed by a max-pooling layer. The second convolutional layer took the output of the first layer and filtered it with 32 filters. The third convolutional layer took the output of the previous layer and filtered it with 64 filters and the final convolutional layer filtered the previous layer output with 64 filters. The output from the convolutional layers was then flattened and fed into the fully connected (FC) layers. The first FC layer had 64 output nodes and is activated by a ReLU. The second FC layer which was also the final output layer had a single node which then actually performed the classification into capsid or non-capsid using sigmoid activation.

According to Chollet (2017) and Yosinski et al. (2014), the filter activations retain most of the information present in the input image at the earlier convolutional layer, such as edge, color, texture or gradient features. Later layers extract higher level features that are very class-specific, meaning that as the image progresses through the later layers, the activations became continuously more abstract and less visually interpretable (Fig. 4). The layers started to encode higher level concepts such as the membrane layers and capsids. The

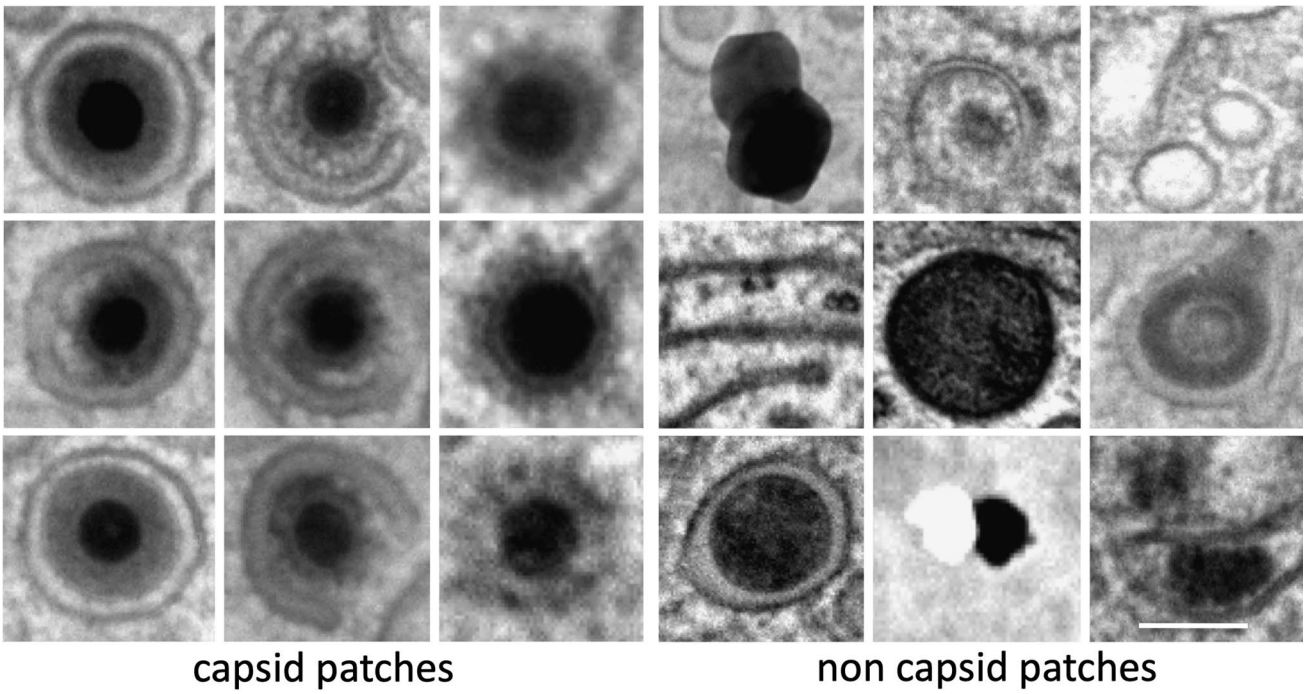


Fig. 2 Typical capsid patches used as training data. (Left) Capsid patches including nucleocapsids belonging to the three maturation stages enveloped, budding and naked. (Right) Non-capsid patches with a wide variety in appearance (bar 100 nm)

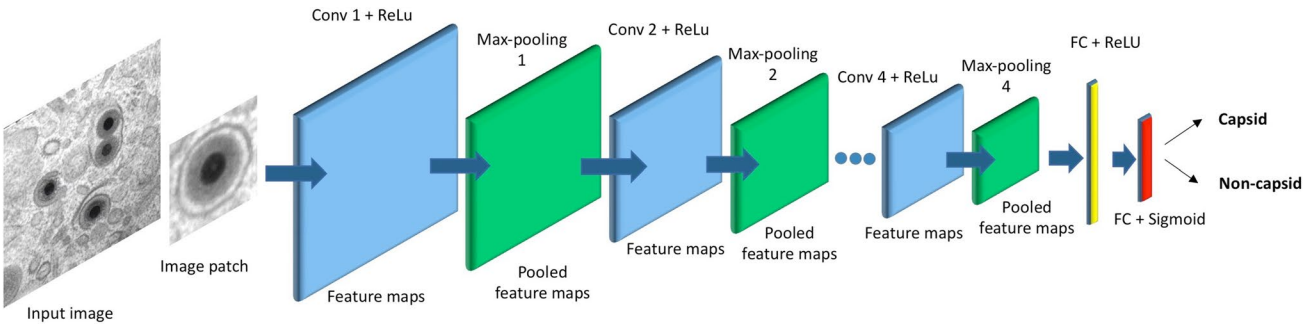


Fig. 3 Schematic overview of the CNN-TFS model. *Conv* convolutional layer, *FC* fully connected, *ReLU* rectified linear units activation

later layers contain increasingly less information about the visual properties of the input image but increasingly more information related to the class of the image.

Transfer learning

We have used three popular pre-trained deep CNNs, which are VGG16 (Simonyan and Zisserman 2015), InceptionV3 (Szegedy et al. 2016) and Residual Neural Networks 50 (ResNet50) (He et al. 2016). These CNN models have been pre-trained on the ImageNet, which is a dataset that contains over 15 million labelled high-resolution images belonging to roughly over 22,000 categories (Deng et al. 2009). VGG16, InceptionV3 and ResNet50 have achieved state-of-the-art performance on the ImageNet challenge in 2014 and 2015 (Russakovsky et al. 2015). We then transferred the features learned by these pre-trained networks from the ImageNet database to the task of capsid detection. In recent years, researches have shown that using pre-trained CNN models and applying it to other domains have yielded better performing models compared to training a network from scratch (Shin et al. 2016).

In this work, we used two main transfer learning approaches. We first used the pre-trained CNNs as a fixed feature extractor and then additionally fine-tuned the pre-trained CNNs in a layer-wise manner (Tajbakhsh et al. 2016). To determine whether these pre-trained models were good candidates for our transfer learning approach,

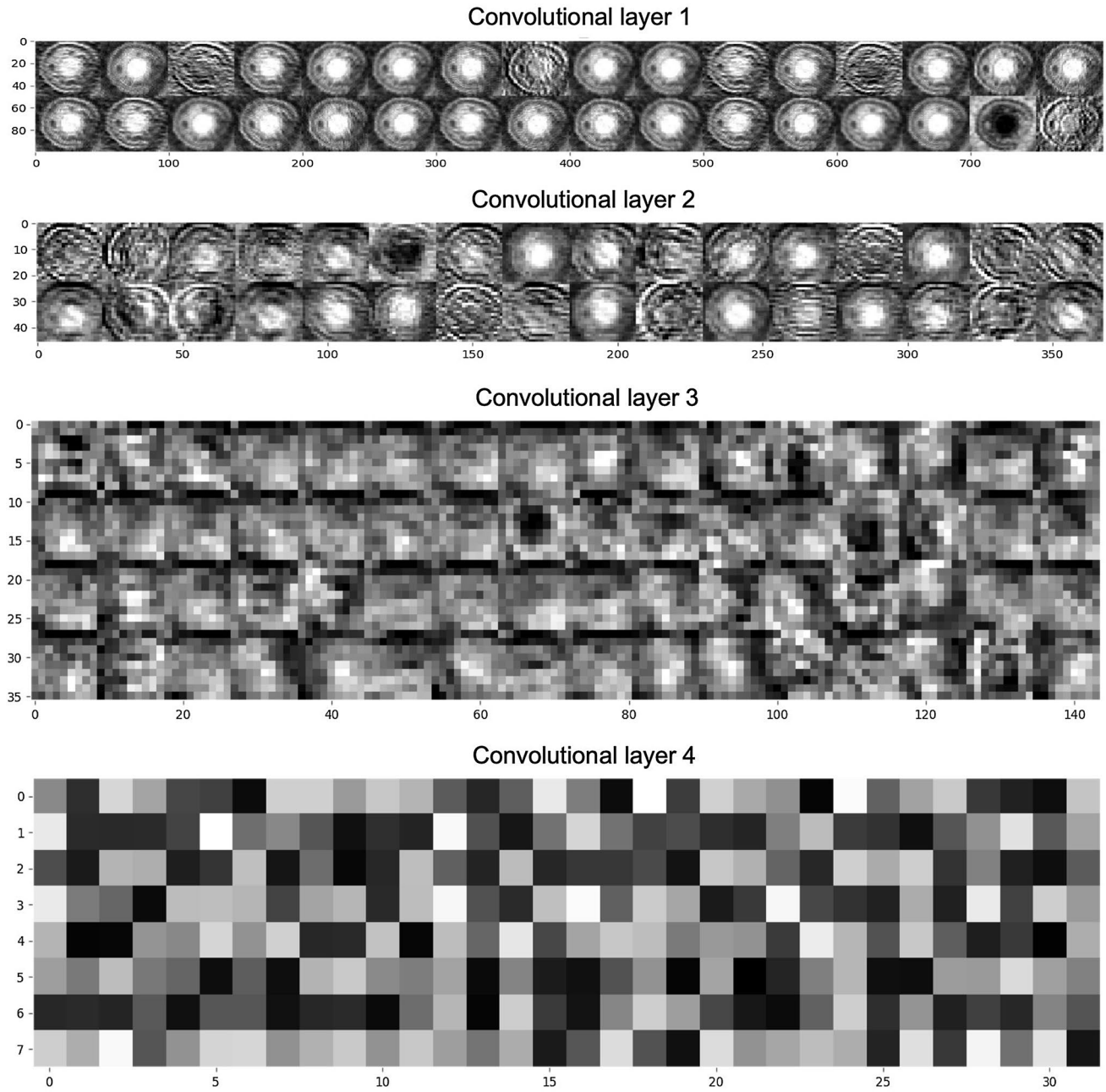


Fig. 4 Visualizations of the feature maps of the CNN-TFS model. These feature maps show what the filter learned at every convolutional layer

we initially performed a *t*-distributed stochastic neighbor embedding (*t*-SNE) on the features extracted from the pre-trained CNNs (Fig. 5) (Van Der Maaten and Hinton 2008). t-SNE is a dimensionality reduction technique that is used to visualize high dimensional data to two or three dimensions.

CNN as a feature extractor

The CNNs pre-trained on ImageNet were taken with their weights, and then the last FC layer was removed. The remaining portion of the CNNs was used as a fixed feature extractor for our own dataset. For each CNN model, we added a FC layer with a global average pooling layer with 1024 nodes and with ReLU activation. We then added a second FC layer with a single node output activated by sigmoid function. The features extracted by this method are called 'bottleneck features'. These bottleneck features represent the last activation maps before the FC layers in the pre-trained model. With the layers of the pre-trained model frozen, we then trained our FC network on those extracted

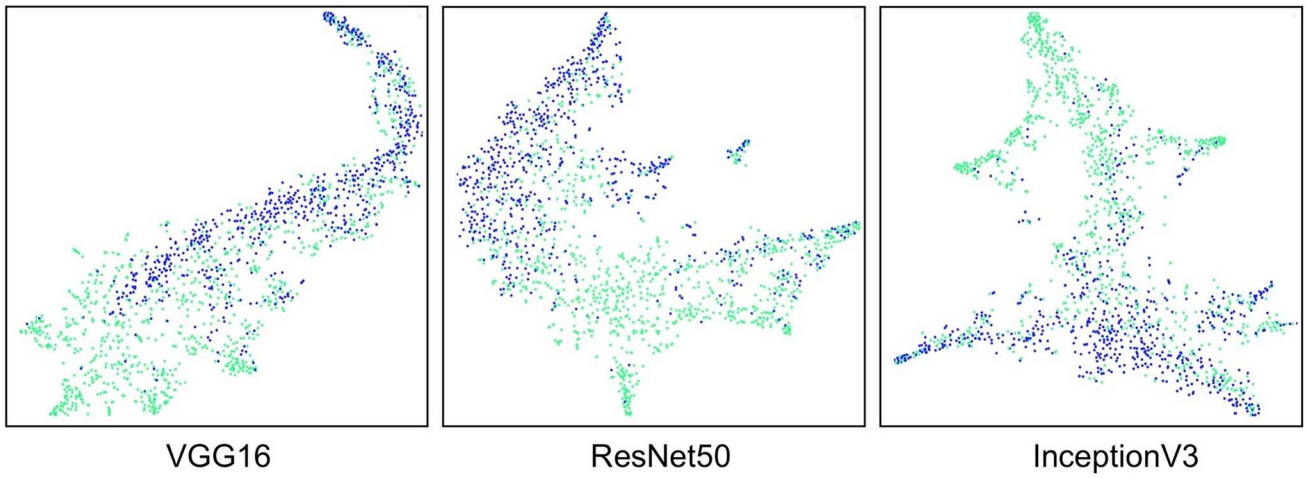


Fig. 5 t-SNE embedding of the features from the pre-trained models VGG16, ResNet50 and InceptionV3 into capsid (blue) and non-capsid (green) classes

bottleneck features with our training data to classify our dataset (Shin et al. 2016). The reason for this was that when the layers of a CNN are frozen and initialized from a pre-trained model, there is no need to back-propagate through them during training and they behave as fixed features without changing on the new task (Yosinski et al. 2014). The resulting models were then used as feature extractors and are from hereby referred to as VGG16-FE, InceptionV3-FE and ResNet50-FE.

Fine-tuning the CNN

In our work, the size of the dataset is limited as well as the data similarity to ImageNet is very low. Therefore, another approach was to fine-tune the weights of the networks. There is no specific method to fine-tune a network. However, the standard practice is to start from the last layer and then incrementally include more layers in the update process until the desired performance is reached.

We initially shallow tuned all the three pre-trained models. We did this by first removing the last FC layer and adding our own FC network as described in our CNN as a feature extractor approach. We then re-trained some of the higher level convolutional layers of the network and froze the lower-level layers (Tajbakhsh et al. 2016). For this, we froze the initial k number layers of the pre-trained models and trained the remaining n–k layers again. The top layers were then customized to our TEM dataset. The small size of the dataset was compensated by keeping the initial layers pre-trained (which have been previously trained on the ImageNet) and the weights for those layers were frozen.

The convolutional layers were re-trained using a block-wise manner. For VGG16-FT, we re-trained only the last convolutional block of the VGG16 architecture as well as the FC layers. This model is referred to as VGG16-FT. For both the InceptionV3 and ResNet50 models, we re-trained the final two convolutional blocks as well as the FC layers and these models are referred to as InceptionV3-FT and ResNet50-FT, respectively. The InceptionV3-FT model was then used for deep fine-tuning to determine whether better performance can be accomplished. The deep fine-tuning was performed by increasing the number of convolutional layers that were used for re-training. The deep fine-tuned models are referred to as InceptionV3-FT-2 (2 convolutional blocks retrained), InceptionV3-FT-3 (3 convolutional blocks retrained), InceptionV3-FT-4 (4 convolutional blocks retrained) and InceptionV3-FT-5 (5 convolutional blocks retrained).

Hyperparameter settings

Adam optimization algorithm (Kingma and Ba 2015) was used to update the model parameters such as weights and bias values of the CNNs used in this work. The initial learning rate was set to 0.001. The stride was set as 1. We used a mini batch size of 16. To prevent our models from over-fitting, a dropout of 0.5 was used for all the models in this work to avoid that a layer sees twice the exact same pattern, thereby acting in a similar manner to data augmentation (Bengio 2012). All the models were trained for 100 epochs. The training was performed on a NVIDIA GeForce GTX 1060 3 GB GPU machine with Keras software on a Tensorflow backend. To avoid bias in training, we performed a five times fivefold stratified cross-validation (CV) to evaluate the models.

Capsid detection

We used a sliding window to detect the HCMV nucleocapsids in the TEM images. It is a rectangular region of fixed width and height that "slides" across an image in a left to right and top to bottom manner. For each of these windows, the ResNet50-FT model was applied to the window region to detect the nucleocapsids. When a window with a capsid is detected, the centroid coordinate of the window is saved. The obtained centroid coordinates are then used to plot the location of the nucleocapsids in the 21 test TEM images. We combine the sliding window with image pyramids, which is a multi-scale representation of an image so that the CNN can detect the nucleocapsids at various scales and locations in the TEM images (Adelson et al. 1984). Two major parameters for the sliding window are the window size and step size. The window size indicates the size of the rectangular region of the window and the step size indicates how many pixels are skipped in both the x and y direction. In our work, the window size was set as 52×52 pixels and the step size was set as 20 pixels.

Validation and evaluation of CNN performance

The performance of the feature extractor models and the fine-tuned models was evaluated in comparison to the CNN-TFS. For this, we evaluated the CNNs using four widely used performance measures: accuracy, recall, precision and F1 score. These measures were computed from the true positive (TP), false positive (FP), true negative (TN) and false negative (FN) results obtained. True positives are correctly identified nucleocapsids, false positives are structures, which were detected as nucleocapsids but are not nucleocapsids and false negatives are nucleocapsids that were not detected as such.

The formulas for the performance measures are given below:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{4}$$

We also applied two existing image analysis techniques, the radial density profile (RDP) method (Kylberg et al. 2012; Sintorn et al. 2004) and texture indicator (TI) method (Proença et al. 2013) to our TEM image dataset and compared the results obtained to our work to validate the effectiveness of our transfer learning methods.

Results and discussion

The model CNN-TFS which was generated without any transfer learning gave the poorest performance for all performance measures (Table 1). This is not surprising as the high feature variability in TEM images requires deeper architectures to obtain a full representation of features from the EM images. Training such a deep network from scratch is not feasible because it requires a large amount of annotated data as well as expensive computational and memory resources. Thus, this approach had its practical limitations. Therefore, we wanted to investigate the application of transfer learning techniques to our work. This is in line with the study by Yosinski et al. (2014) that stated that transferring features from a distance task may actually give a better performance than just using random features. The pre-trained models, InceptionV3, VGG16 and ResNet50 were used as our base for the transfer learning approach.

Before we conducted feature extraction from the pre-trained models InceptionV3, VGG16 and ResNet50, we tested how high their potential for nucleocapsid detection is. For that we applied the t-SNE test. A good transfer learning model will have an embedding that will be visually separated better. As can be seen in Fig. 5, the blue capsid class cluster is well separated from the green non-capsid class cluster for all the three models. In none of the models, however, there is

Table 1 Performance measure of the models

	Accuracy	Recall	Precision	F1 score
CNN-TFS	0.8242 ± 0.0018	0.8142 ± 0.0040	0.8014 ± 0.01260	0.8077 ± 0.060
VGG16-FE	0.8304 ± 0.0022	0.8333 ± 0.0054	0.8169 ± 0.0085	0.8250 ± 0.0066
ResNet50-FE	0.8381 ± 0.0025	0.8026 ± 0.0038	0.8197 ± 0.0006	0.8833 ± 0.0010
InceptionV3-FE	0.8936 ± 0.0034	0.8901 ± 0.0068	0.8942 ± 0.0036	0.8921 ± 0.0047
VGG16-FT	0.9300 ± 0.0019	0.9218 ± 0.0011	0.9287 ± 0.0040	0.9252 ± 0.0017
ResNet50-FT	**0.9544 ± 0.0046**	**0.9539 ± 0.0007**	**0.9505 ± 0.0022**	**0.9522 ± 0.0011**
InceptionV3-FT	0.9474 ± 0.0035	0.9412 ± 0.0121	0.9387 ± 0.0032	0.9399 ± 0.0037

The mean and standard deviation for all the measures are listed. The best performing model is highlighted in bold

perfect separation. Visually, InceptionV3 has the most efficient separation between classes, followed byResNet50 and VGG16. This indicates that all the three pre-trained state-of-the-art models are good candidates for transfer learning.

The selection of the transfer learning approach (feature extraction and fine-tuning) is dependent on the size of the dataset and its similarity to the original large dataset which the pre-trained model was trained upon (Tajbakhsh et al. 2016). Since it was not known which transfer learning approach and pre-trained CNN is most suitable for nucleocapsid detection, we first used the three pre-trained models VGG16, ResNet50 and InceptionV3 as fixed feature extractors and then fine-tuned them with our TEM dataset. Finally, we compared the nucleocapsid detection performance of all models with the CNN-TFS with the four widely used performance measures accuracy, recall, precision and F1 score (Table 1).

The fine-tuned models VGG16-FT, ResNet50-FT and InceptionV3-FT gave better results for all performance metrics compared to the feature extractor models VGG16-FE, InceptionV3-FE and ResNet50-FE (Table 1). From this, we can conclude that an efficient transfer of features could not be achieved by just re-training the FC layers alone, some of the convolutional layers need to be re-trained as well to obtain better performance. The reason for that is the very low similarity between the ImageNet database which the models were pre-trained with and our TEM image dataset (Yosinski et al. 2014). However, the transfer learning approach was still able to learn features from the ImageNet database which could be applied for nucleocapsid detection. For example, features learned from images of the musical instrument gong and nematodes (roundworms) were useful for our model to learn the appearance of the nucleocapsids and their membrane layer. The validation accuracy of VGG16-FE, VGG16-FT, InceptionV3-FT and ResNet50-FT stabilizes at an earlier point compared to the other models for 100 epochs trained (Fig. 6).

ResNet50-FT yielded the best performance among all the models. InceptionV3-FT also performed very well and the differences between the performances between them were minimal. Both, InceptionV3-FT and ResNet50-FT had low numbers of false negatives and false positives. In the context of HCMV nucleocapsid detection, achieving a low number of false negatives is highly desirable. False positives have to be tolerated as a comprehensive detection of nucleocapsids is important to fully analyze TEM images.

The number of trainable parameters for each and every model differs based on the architecture of the models. Table 2 shows the number of parameters trained for all the models in our work. The higher the number of trained parameters, the higher is the amount of fine-tuning performed on the models to adapt them to our TEM image dataset. However, as the structure of the convolutional layers

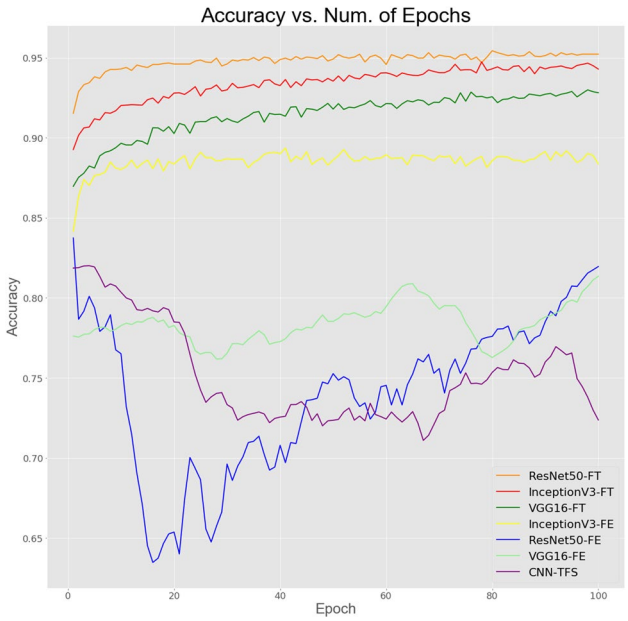


Fig. 6 Graph showing the validation accuracy of all the models for 100 epochs

and FC layers in the CNN-TFS, VGG16, InceptionV3 and ResNet50 differs greatly, the number of trained parameters is relative to the total number of parameters in each model.

The higher the number of parameters re-trained in a model, the better was the performance of the model (Tables 1, 2). The training time for the networks is dependent on the complexity of the network as well as the type of computer and graphical processing unit (GPU) being used. Complex networks such as InceptionV3 and ResNet50 generally have larger training time compared to VGG16. Additionally, the training time also increases with the amount of layers re-trained, the dataset size and the extent of data augmentation performed.

The best performing network was the ResNet50-FT. This network has a different architecture compared to the standard CNN, VGG16 and InceptionV3 networks. Thus, setting up and training this model took a longer time compared to the other models. Setting up the neural network of ResNet50-FE, consumed about 4 human working hours, and 50,412 s computer training time. Setting up the neural network ResNet50-FT required about 6 human working hours and the computer training time was 56,173 s since a higher number of layers needed to be re-trained compared to the ResNet50-FE. The computer training time for the other networks is specified in Table 2. Computer training of the networks generated the models that were used for the detection of capsids in the test images which took about 64 s per image with the ResNet50-FT model. The detection time is heavily influenced by the step size of the sliding window process. The step size for the sliding window in this work was 20 pixels. A smaller step

Table 2 The total number of parameters, re-trained parameters, frozen parameter (un-trained) and training time in seconds for all the models is listed

Model	Re-trained parameters	Frozen parameters	Total parameters	Training time (s)
CNN-TFS	69,793	0	69,793	1890
VGG16-FE	2,097,665	14,714,688	16,812,353	9912
InceptionV3-FE	33,554,945	21,802,784	55,357,729	46,823
ResNet50-FE	524,801	23,587,712	24,112,513	50,412
VGG16-FT	9,177,089	7,635,264	16,812,353	12,480
InceptionV3-FT	34,361,823	13,734,106	55,357,729	54,418
ResNet50-FT	21,010,689	3,101,824	24,112,513	56,173

size would increase the detection time per image, whereas a larger step size would decrease the detection time per image. Setting up of the neural networks and training is a one-time process. Once the model is generated, it could be deployed as a stand-alone software for the detection of any amount of images with about 64 s detection time per image.

We additionally performed deep fine-tuning with a higher number of parameter re-training to investigate its impact on the network performance. We selected InceptionV3-FT for deep fine-tuning as its performance is almost similar to ResNet50 but its architecture is more easily interpretable compared to ResNet50. As the architecture of InceptionV3 is structured in convolutional blocks, deep fine-tuning is performed by block-wise re-training. Figure 7 shows the validation accuracy of the various deep fine-tuned InceptionV3-FT models for 100 epochs.

We observed that the validation accuracy improved when a higher number of convolutional blocks were re-trained. There seems to be a trade-off between the number of parameters re-trained and the amount of data available for training. Thus, while the accuracy of the models initially increased up to re-training four convolutional blocks, the performance started to degrade when five convolutional blocks were re-trained. Re-training larger amounts of parameters with a small dataset could lead to degradation of performance. This is because, as the FC layers on top are randomly initialized, very large weight updates would propagate through the network and effectively destroy the features previously learned. It was observed that the improvement in accuracy for deeper fine-tuning was rather minimal but the training time increased.

To validate the effectiveness of our transfer learning method, we compared the performance of InceptionV3-FT and ResNet50-FT to radial density profile (RDP) method (Kylberg et al. 2012; Sintorn et al. 2004) and texture indicator (TI) method (Proença et al. 2013) on our dataset.

It can be seen that our method gave superior results for all performance measures compared to the RDP and TI method (Table 3). In contrast, CNN-TFS, VGG16-FE, ResNet50-FE performed worse than the RDP and TI method (Tables 1, 3). This proves that the fine-tuning approach of transfer learning is able to yield generally a better performance compared to existing image analysis techniques for viral capsid detection in TEM images.

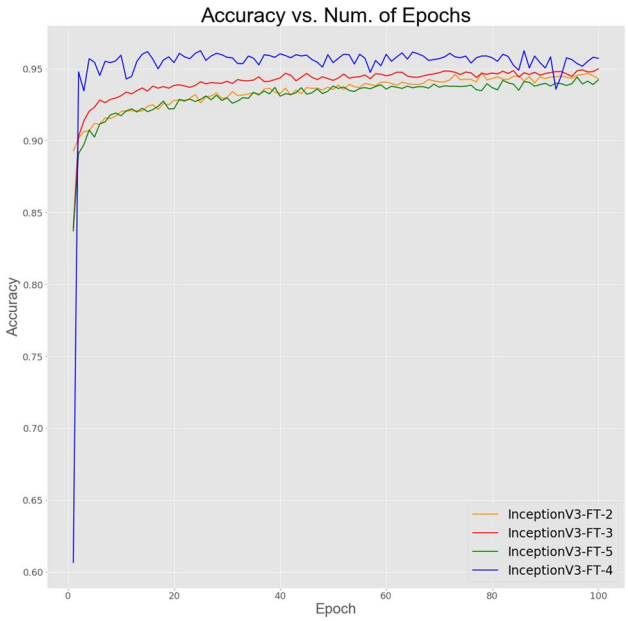


Fig. 7 Graph of validation accuracy for 100 epochs trained with InceptionV3 model. InceptionV3-FT-2 indicates two convolution blocks re-trained, InceptionV3-FT-3 indicates there convolution blocks re-trained, InceptionV3-FT-4 indicates four convolution blocks re-trained and InceptionV3-FT-5 indicates five convolution blocks re-trained

As Resnet50-FT was the best performing model, we applied it for nucleocapsid detection in our test image dataset (Fig. 8). The test dataset was an independent dataset that was not used for training and validation. The model was successful in detecting nucleocapsids. There were a total of 366 nucleocapsids in the test images. The model detected 328 true positives, 38 false negatives and 35 false positives. However, the false positives and false negatives resulted from black dots that looked similar to nucleocapsids, other imaging and preparation artifacts as well as nucleocapsids with a faint core and the low number of training data available (Fig. 9). The sliding window method is not the state-of-the-art technique for object detection. Thus, while the ResNet50-FT had a high

Table 3 Comparison for the performance measures of existing image analysis and the here developed transfer learning methods

Method	Accuracy	Recall	Precision	F1-score
TI	0.8657 ± 0.0013	0.8642 ± 0.0033	0.8656 ± 0.0010	0.8641 ± 0.0022
RDP	0.8328 ± 0.0090	0.8335 ± 0.0011	0.8382 ± 0.020	0.8324 ± 0.0015
ResNet50-FT	**0.9544 ± 0.0046**	**0.9539 ± 0.0007**	**0.9505 ± 0.0022**	**0.9522 ± 0.0011**
InceptionV3-FT	0.9474 ± 0.0035	0.9412 ± 0.0121	0.9387 ± 0.0032	0.9399 ± 0.0037

The values of the finally selected ResNet50-FT are shown in bold

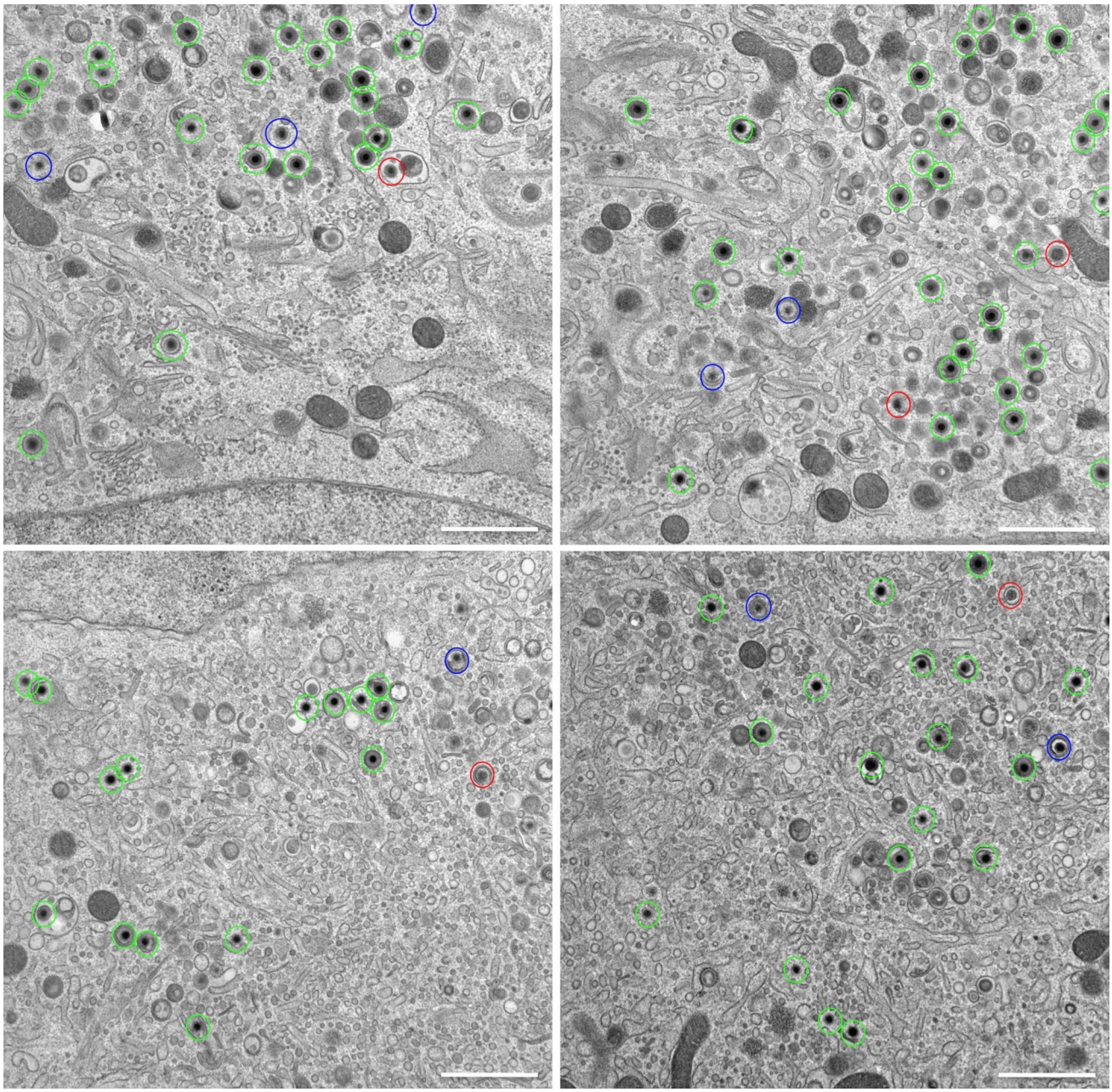


Fig. 8 Examples of test images showing nucleocapsids detected using the Resnet50-FT model. The green circles indicate true positives, the red indicate false positives and blue indicate false negatives (bars 1 µm)

accuracy, its detection using the sliding window method is slightly lower. Other state-of-the-art methods such as Faster RCNN or YOLO may yield better detection rate compared to the sliding window. However, as the main aim of this work is to investigate the applicability of transfer learning for nucleocapsids, the scope of this work was focused towards the transfer learning techniques such as feature extraction and fine-tuning. The investigation of advanced object detection methods will be the scope of future work.

Our study provides the proof-of-principle that transfer learning can be applied for the generation of CNNs that allow automatic detection of objects in EM images. In this study, the CNNs were trained to specifically detect HCMV nucleocapsids in TEM images of high-pressure frozen, freeze substituted and Epon embedded samples. The object automatically detected by a CNN is determined by the type of training data regarding sample preparation and imaging technique and obviously by the biological structure of interest annotated for training. Thus, it is possible to extend the presented approach to a variety of other structures (e.g., mitochondria or synaptic vesicles) originating from different EM sample preparation protocols and imaging modalities (e.g., scanning electron microscopy; SEM). For every individual task the CNN needs to be optimized and trained individually. However, the performance of the networks is dependent on the structural complexity and diversity of the objects being classified as well as the resolution and quality of the images used for training. VGG16, InceptionV3 and ResNet50 were the networks used within the scope of this work. There are other transfer learning networks that are also available for use. While ResNet50-FT was the best performer for this work, for a different application, another network such as InceptionV3 or VGG16 might be more suitable.

Nevertheless, this work provides the basis for the generation of an automated pipeline for structure detection in EM images, not only for quantification but possibly also for data representation purposes (segmentation). This will cut down time for manual interaction with the data tremendously. Furthermore, this automated pipeline allows to analyze EM data in an efficient and, more importantly, reproducible way. This is even valuable for analysis of relatively small TEM datasets as used for routine phenotyping of HCMV mutants where for one mutant virus about 60 images of a total of 15 cells are subjected to quantitative analysis to gain significant results (Cappadona et al. 2015). In line with this, comparison of results between different researchers and labs would become possible, given that they work with the same EM preparation and imaging techniques.

It is furthermore worth mentioning that the current development of EM techniques aims at the automatic acquisition of large 3D datasets with a size of up to several terabytes (reviewed in Peddie and Collinson 2014). The most applied techniques for this purpose are automated serial sectioning and block face SEM (e.g., Villinger et al. 2015). Our approach can be further developed to be applied to large 3D datasets produced by these methods.

Conclusion and future work

While deep learning methods have exhibited better classification and detection capability compared to traditional image processing methods, they generally require a large amount of training data with manual annotation to perform well. We have shown that using transfer learning techniques it is possible to obtain good results with a small training dataset. In this work, we applied transfer learning for the detection of HCMV nucleocapsids in TEM images. We

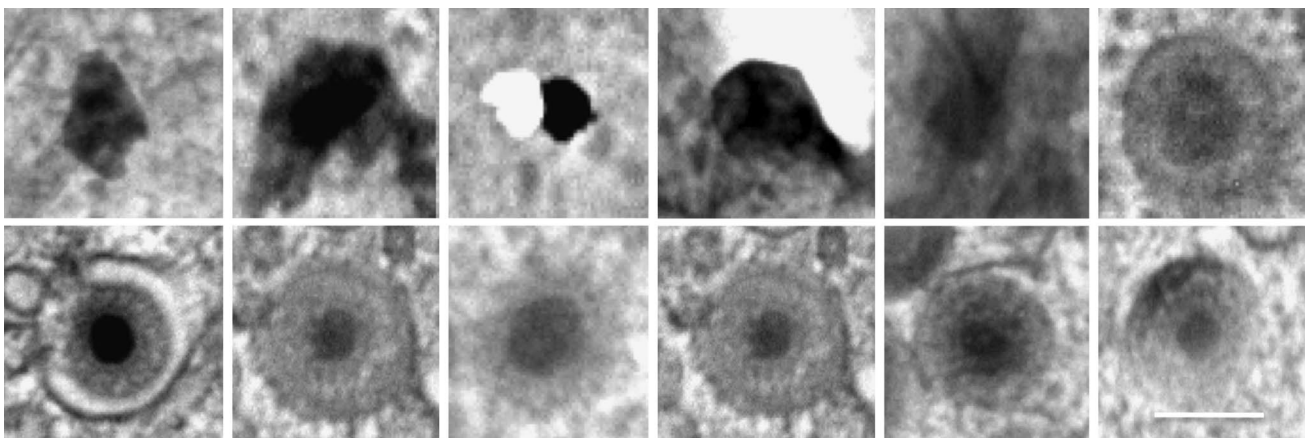


Fig. 9 Some of the false positives (top row) and false negatives (bottom row) detected by the ResNet50-FT model in our test images (bar 100 nm)

compared the performance of a CNN that was trained from scratch with pre-trained CNN models as well as existing image analysis methods. It was observed that pre-trained CNN models performed significantly better than the CNN trained from scratch showing that features learned from the ImageNet database could be successfully transferred to the detection of viral capsids although the TEM images were extremely dissimilar to the ImageNet images. Furthermore, fine-tuning of the pre-trained models gave significantly better performances compared to the feature extractor models indicating that re-training both the convolutional and the FC layers, is more beneficial than just re-training the FC layers. The fine-tuned models also performed better than the existing image analysis methods. Performing the classification of different capsid maturation stages with state-of-the-art detection methods would be the next step with this approach.

Acknowledgements This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Projektnummer 316249678-SFB 1279.

References

Abdolmanafi A, Duong L, Dahdah N, Cheriet F (2017) Deep feature learning for automatic tissue classification of coronary artery using optical coherence tomography. Biomed Opt Express 8:1203–1220. https://doi.org/10.1364/BOE.8.001203

Adelson EH, Anderson CH, Bergen JR, Burt PJ, Ogden JM (1984) Pyramid methods in image processing. RCA Eng 29(6):33–41

Andrei G, De Clercq E, Snoeck R (2009) Drug targets in cytomegalovirus infection. Infect Disord Drug Targets 9:201–222. https://doi.org/10.2174/187152609787847758

Bengio Y (2012) Practical recommendations for gradient-based training of deep architectures. In: In Montavon G, B.Orr G, Müller KR (eds) Neural networks: tricks of the trade. Lecture notes in computer science, vol 7700. Springer, Berlin, pp 437–478. https://doi.org/10.1007/978-3-642-35289-8_26

Cappadona I, Villinger C, Schutzius G, Mertens T, von Einem J (2015) Human cytomegalovirus pUL47 modulates tegumentation and capsid accumulation at the viral assembly complex. J Virol 89:7314–7328. https://doi.org/10.1128/JVI.00603-15

Chollet FC (2017) Deep learning with python, 1st edn. Manning Publications, New York, pp 160–166

Deng J, Dong W, Socher R, Li LJ, Li K, Li FF (2009) ImageNet: a large-scale hierarchical image database. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), Miami, pp 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dietz AN, Villinger C, Becker S, Frick M, von Einem J (2018) A tyrosine-based trafficking motif of the tegument protein pUL71 is crucial for human cytomegalovirus secondary envelopment. J Virol 92:e00907–e00917. https://doi.org/10.1128/JVI.00907-17

Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S (2017) Dermatologist-level classification of skin cancer with deep neural networks. Nature 542:115–118. https://doi.org/10.1038/nature21056

Ghafoorian M, Mehrtash A, Kapur T, Karssemeijer N, Marchiori E, Pesteie M, Guttmann CRG, Leeuw FE, Tempany CM, van Ginneken B, Fedorov A, Abolmaesumi P, Platel B, Wells WM III (2017) Transfer learning for domain adaptation in MRI: application in brain lesion segmentation. In: Descoteaux M, Maier-Hein L, Franz A (eds) Proceedings of the 20th medical image computing and computer-assisted intervention (MICCAI) international conference, Part 3, Quebec, pp 516–524. https://doi.org/10.1007/978-3-319-66179-7

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, pp 770–778. https://doi.org/10.1109/CVPR.2016.90

Ito E, Sato T, Sano D, Utagawa E, Kato T (2018) Virus particle detection by convolutional neural network in transmission electron microscopy images. Food Environ Virol 10:201–208. https://doi.org/10.1007/s12560-018-9335-7

Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: Proceedings of the international conference on learning representations, San Diego, pp 1–15. https://arxiv.org/abs/1412.6980v8

Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks, vol 1. In: Proceedings of the 25th international conference on neural information processing systems, Lake Tahoe, pp 1097–1105. https://doi.org/10.1145/3065386

Kylberg G, Uppström M, Hedlund KO, Borgefors G, Sintorn IM (2012) Segmentation of virus particle candidates in transmission electron microscopy images. J Microsc 245:140–147. https://doi.org/10.1111/j.1365-2818.2011.03556.x

Lam C, Yu C, Huang L, Rubin D (2018) Retinal lesion detection with deep learning using image patches. Invest Ophthalmol Vis Sci 59:590–596. https://doi.org/10.1167/iovs.17-22721

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognitions. Proc IEEE 86:2278–2324. https://doi.org/10.1109/5.726791

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444. https://doi.org/10.1038/nature14539

Li R, Si D, Zeng T, Ji S, He J (2016) Deep convolutional neural networks for detecting secondary structures in protein density maps from cryo-electron microscopy. In: Proceedings of IEEE international conference on bioinformatics and biomedicine, Shenzhen, pp 41–46. https://doi.org/10.1109/BIBM.2016.7822490

Mercorelli B, Lembo D, Palù G, Loregian A (2011) Early inhibitors of human cytomegalovirus: state-of-art and therapeutic perspectives. Pharmacol Ther 131:309–329. https://doi.org/10.1016/J.PHARMTHERA.2011.04.007

Mocarski E, Shenk T, Pass RF (2007) Cytomegoloviruses. In: Knipe DM, Howley PM, Griffin DE, Lamb RA, Martin MA, Roizman B, Straus SE (eds) Field virology, 5th edn. Lippincott Williams & Wilkins, Philadelphia, pp 2701–2771

Modarres MH, Aversa R, Cozzini S, Ciancio R, Leto A, Brandino GP (2017) Neural network for nanoscience scanning electron microscope image recognition. Sci Rep. https://doi.org/10.1038/s41598-017-13565-z

Peddie CJ, Collinson LM (2014) Exploring the third dimension: volume electron microscopy comes of age. Micron 61:9–19. https://doi.org/10.1016/j.micron.2014.01.009

Proença MC, Nunes JFM, de Matos APA (2013) Texture indicators for segmentation of polyomavirus particles in transmission electron microscopy images. Microsc Microanal 19:1170–1182. https://doi.org/10.1017/S1431927613001736

Ribeiro E, Uhl A, Wimmer G, Häfner M (2016) Exploring deep learning and transfer learning for colonic polyp classification. Comput Math Methods Med 2016:1–16. https://doi.org/10.1155/2016/6584725

Ribli D, Horváth A, Unger Z, Pollner P, Csabai I (2018) Detecting and classifying lesions in mammograms with deep learning. Sci Rep 8:4165. https://doi.org/10.1038/s41598-018-22437-z

Roels J, De Vylder J, Aelterman J, Saeys Y, Philips W (2017) Convolutional neural network pruning to accelerate membrane segmentation in electron microscopy. In: Proceedings of the 14th

international symposium on IEEE biomedical imaging, Melbourne, pp 633–637. https://doi.org/10.1109/ISBI.2017.7950600

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis 115:211–252. https://doi.org/10.1007/s11263-015-0816-y

Ryner M, Strömberg JO, Söderberg-Nauclér C, Homman-Loudiyi M (2006) Identification and classification of human cytomegalovirus capsids in textured electron micrographs using deformed template matching. J Virol 3:57. https://doi.org/10.1186/1743-422X-3-57

Schauflinger M, Fischer D, Schreiber A, Chevillotte M, Walther P, Mertens T, von Einem J (2011) The tegument protein UL71 of human cytomegalovirus is involved in late envelopment and affects multivesicular bodies. J Virol 85:3821–3832. https://doi.org/10.1128/JVI.01540-10

Schauflinger M, Villinger C, Mertens T, Walther P, von Einem J (2013) Analysis of human cytomegalovirus secondary envelopment by advanced electron microscopy. Cell Microbiol 15:305–314. https://doi.org/10.1111/cmi.12077

Shin HC, Roth HR, Gao M, Lu L, Xu Z, Nogues I, Yao J, Mollura D, Summers RM (2016) Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE Trans Med Imaging 35:1285–1298. https://doi.org/10.1109/tmi.2016.2528162

Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. https://arxiv.org/abs/1409.1556

Sintorn IM, Homman-Loudiyi M, Söderberg-Nauclér C, Borgefors G (2004) A refined circular template matching method for classification of human cytomegalovirus capsids in TEM images. Comput Methods Programs Biomed 76:95–102. https://doi.org/10.1016/j.cmpb.2004.03.006

Sinzger C, Digel M, Jahn G (2008) Cytomegalovirus cell tropism. In: Compans RW, Malissen B, Aktories K, Rappuoli R, Galan JE, Ahmed R, Palme K, Casadevall A, Garcia-Sastre A, Iwasaki A, Akira S (eds) Curr Top Microbiol Immunol 325:63–83. https://doi.org/10.1007/978-3-540-77349-8_4

Sousa RG, Esteves T, Rocha S, Figueiredo F, De Sá JM, Alexandre LA, Santos JM, Silva LM (2015) Transfer learning for the recognition of immunogold particles in TEM imaging. In: Rojas I, Joya G, Catala A (eds) Advances in computational intelligence. Lecture notes in computer science, vol 9094. Springer, Cham, pp 374–384. https://doi.org/10.1007/978-3-319-19258-1_32

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the Inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, pp 2818–2826. https://doi.org/10.1109/cvpr.2016.308

Tajbakhsh N, Shin JY, Gurudu SR, Hurst RT, Kendall CB, Gotway MB, Liang J (2016) Convolutional neural networks for medical image analysis: full training or Fine tuning? IEEE Trans Med Imaging 35:1299–1312. https://doi.org/10.1109/TMI.2016.2535302

Van Der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res Vol 9:2579–2605

Villinger C, Schauflinger M, Gregorius H, Kranz C, Höhn K, Nafeey S, Walther P (2014) Three-dimensional imaging of adherent cells using FIB/SEM and STEM. Methods Mol Biol 1117:617–638. https://doi.org/10.1007/978-1-62703-776-1_27

Villinger C, Neusser G, Kranz C, Walther P, Mertens T (2015) 3D analysis of HCMV induced-nuclear membrane structures by FIB/SEM tomography: insight into an unprecedented membrane morphology. Viruses 7:5686–5704. https://doi.org/10.3390/v7112900

Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) Advances in neural information processing systems, vol 27. NIPS 2014, Montreal, pp 3320–3328

Zhu Y, Ouyang Q, Mao Y (2017) A deep convolutional neural network approach to single-particle recognition in cryo-electron microscopy. BMC Bioinform 18:348. https://doi.org/10.1186/s12859-017-1757-y

6. Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network [SD-2]

This chapter has been published as a peer-reviewed journal paper.

Shaga Devan K, Walther P, von Einem J, Ropinski T, Kestler HA, Read C. Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network. Cell Microbiol. 2020 Feb;23(2):e13280. https://doi.org/10.1111/cmi.13280. Epub 2020 Nov 16. PMID: 33073426.

Synopsis: This work develops an end-to-end deep learning network for the automatic detection of herpesvirus secondary envelopment stages in transmission electron microscopy images. A data augmentation method of generating synthetic labelled training images using a generative adversarial network was proposed.

Contributions of thesis author: Algorithm design and implementation, computational experiments and manuscript writing.

Supplementary materials of the publication can be accessed at the above mentioned DOI link.

Received: 3 July 2020 | Revised: 1 October 2020 | Accepted: 14 October 2020

DOI: 10.1111/cmi.13280

RESEARCH ARTICLE

WILEY

Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network

Kavitha Shaga Devan[1] | Paul Walther[1] | Jens von Einem[2] | Timo Ropinski[3] | Hans A. Kestler[4] | Clarissa Read[1,2]

[1]Central Facility for Electron Microscopy, Ulm University, Ulm, Germany

[2]Institute of Virology, Ulm University Medical Center, Ulm, Germany

[3]Institute of Media Informatics, Ulm University, Ulm, Germany

[4]Medical Systems Biology, Ulm University, Ulm, Germany

Correspondence
Kavitha Shaga Devan, Central Facility for Electron Microscopy, Ulm University, Albert-Einstein-Allee 11, 89081 Ulm, Germany.
Email: kavitha.shaga-devan@uni-ulm.de

Funding information
Baden-Württemberg Stiftung, Grant/Award Number: ABEM; Deutsche Forschungsgemeinschaft, Grant/Award Number: 316249678 - SFB 1279

Abstract

Detailed analysis of secondary envelopment of the herpesvirus human cytomegalovirus (HCMV) by transmission electron microscopy (TEM) is crucial for understanding the formation of infectious virions. Here, we present a convolutional neural network (CNN) that automatically recognises cytoplasmic capsids and distinguishes between three HCMV capsid envelopment stages in TEM images. 315 TEM images containing 2,610 expert-labelled capsids of the three classes were available for CNN training. To overcome the limitation of small training datasets and thus poor CNN performance, we used a deep learning method, the generative adversarial network (GAN), to automatically increase our labelled training dataset with 500 synthetic images and thus to 9,192 labelled capsids. The synthetic TEM images were added to the ground truth dataset to train the Faster R-CNN deep learning-based object detector. Training with 315 ground truth images yielded an average precision (AP) of 53.81% for detection, whereas the addition of 500 synthetic training images increased the AP to 76.48%. This shows that generation and additional use of synthetic labelled images for detector training is an inexpensive way to improve detector performance. This work combines the gold standard of secondary envelopment research with state-of-the-art deep learning technology to speed up automatic image analysis even when large labelled training datasets are not available.

KEYWORDS

automatic object detection, convolutional neural network, deep learning, generative adversarial network, HCMV, transmission electron microscopy

1 | INTRODUCTION

The virion of the human cytomegalovirus (HCMV), also known as the human betaherpesvirus 5, consist of a dsDNA filled *capsid*, a *tegument* layer and a lipid bilayer as *viral envelope* (Figure 1a). This lipid bilayer is derived from host cell membranes and is acquired by the virus via budding into cytoplasmic vesicles. This process is referred to as secondary envelopment (Mettenleiter, 2004). Several viral proteins are involved in this process (Dietz, Villinger, Becker, Frick, & von Einem, 2018; Mach et al., 2007; Smith, Kosuri, & Kerry, 2014) and there is growing evidence that these viral proteins as well as cellular proteins and host cell membranes form an intricate interaction network that drives the secondary envelopment process (Kuan et al., 2016; Phillips, Cygnar, Thomas, & Bresnahan, 2012; Seo & Britt, 2007).

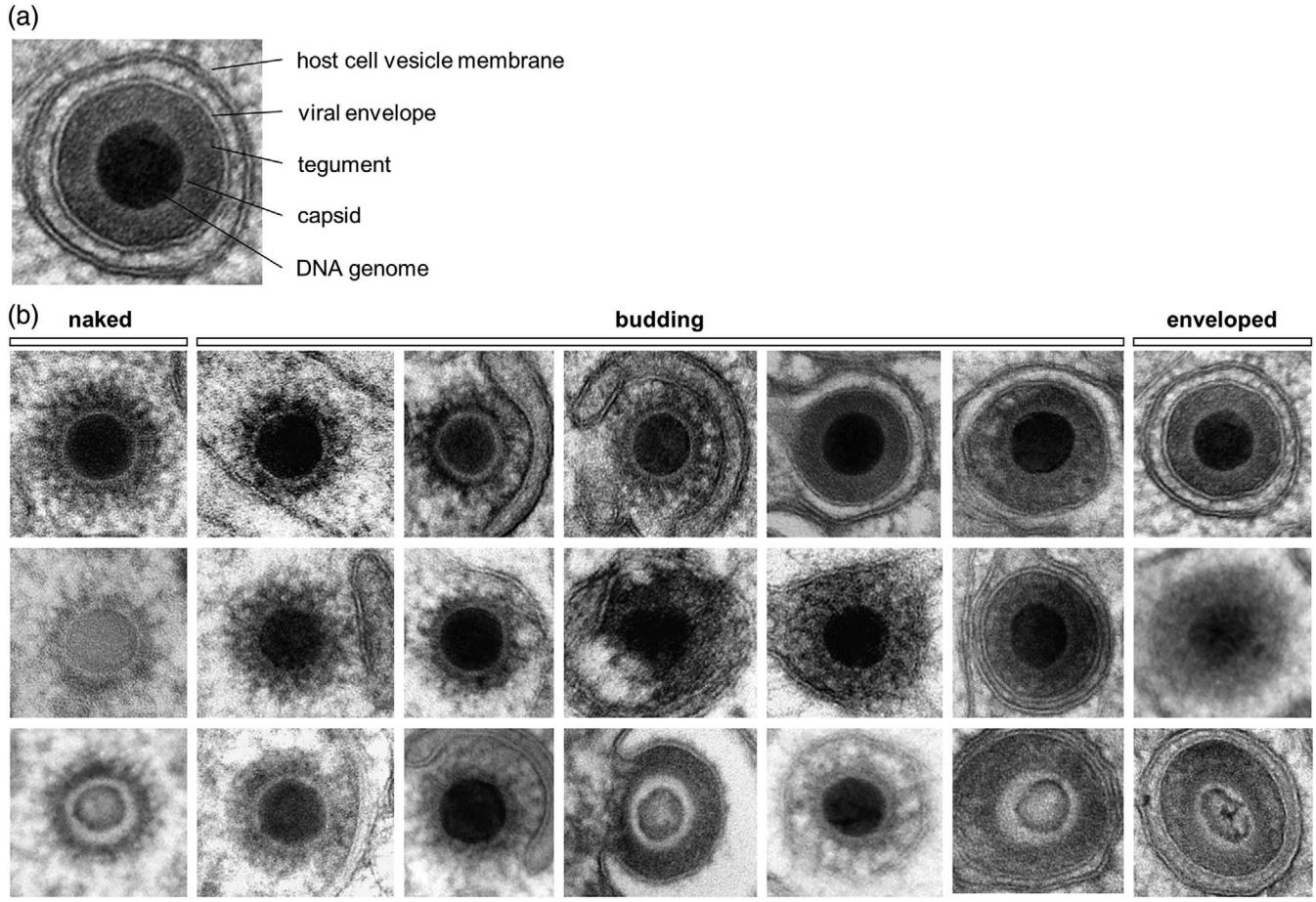


FIGURE 1 TEM images of HCMV particles. (a) HCMV virion in the cytoplasm after budding into a host cell vesicle. (b) HCMV capsid envelopment stages naked, budding and enveloped. Note the intra-class variations and inter-class similarities of the depicted capsids

Electron microscopy (EM) is the only method that allows direct visualisation of secondary envelopment processes and has been proven crucial for understanding HCMV morphogenesis. From these studies three major stages of secondary envelopment can be distinguished (Figure 1b): (a) capsids without contact to a bending membrane (naked capsids), (b) capsids in the process of being enwrapped by a membrane (budding capsids) and (c) capsids that are enclosed by the membrane envelope (enveloped capsids) (Read, Schauflinger, Nikolaenko, Walther, & von Einem, 2019; Schauflinger, Villinger, Mertens, Walther, & von Einem, 2013). Commonly, transmission electron microscopy (TEM) studies of embedded cells in combination with mutational analyses are performed to discover the factors involved in regulating HCMV secondary envelopment (Chevillotte et al., 2010; Kestler et al., 2013; Mach et al., 2007; Phillips & Bresnahan, 2012; Read et al., 2019; Schauflinger et al., 2013; Seo & Britt, 2007). To assess whether a mutation affects this process, qualitative TEM characterizations must be supported by quantitative analyses of capsids and their differentiation of the different envelopment stages, including naked, budding and enveloped capsids (Read et al., 2020; Schauflinger et al., 2013). As in all biological systems, high numbers of cells and capsids need to be involved in these analyses. This is time-consuming and laborious and would greatly benefit from automation.

Automatic object detection has revolutionised biomedical image analysis over the past decade (Wainberg, Merico, Delong, & Frey, 2018). The advantages of automatic object detection go beyond the mere analysis of large datasets in a short time. The data can be analysed independently of the availability and knowledge of a human expert with automated detection, thus avoiding inter- and intra-observer variations. It therefore allows data analysis in a consistent manner at any place and time and even over long periods during which human experts may lose concentration. Nowadays, deep learning techniques, particularly convolutional neural networks (CNNs), are the most common way to perform automatic object detection. They have already achieved remarkable success in many computer vision related tasks, such as image classification, object detection and semantic segmentation (LeCun, Bengio, & Hinton, 2015; Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015; Tang et al., 2019). The biggest advantage of CNNs is that they automatically extract and learn high-level features from the input data, thereby eliminating the need for complicated feature extraction that requires a high level of domain expertise (Tajbakhsh et al., 2016). In other words, the network does not need information about the characteristics and appearances of the objects, instead, the network is fed with training data containing images with labelled objects of interest and the network learns to distinguish them from other objects and from the image background.

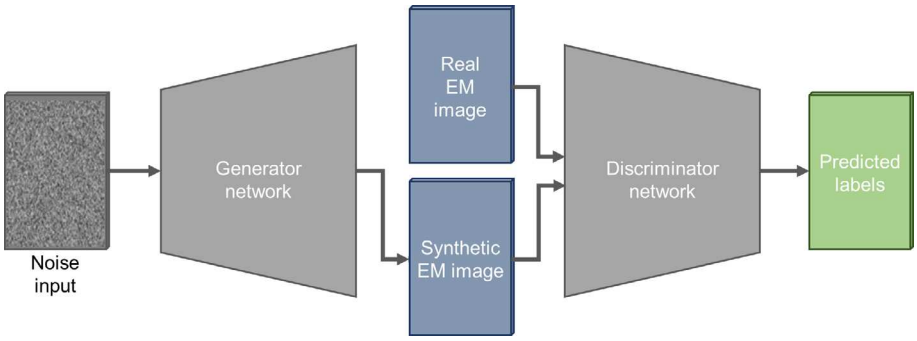


FIGURE 2 A generic GAN model architecture consists of a generator network, which generates images using a noise input, and a discriminator network, which discriminates between the real and generated images. The discriminator uses both the real and generated images to generate realistic synthetic images

The major requirement for a successful object detection through CNNs is the availability of large labelled training datasets. The larger the training dataset, the better is the ability of the CNN to learn. Therefore, CNNs do not perform well when trained with a limited amount of training data (Tajbakhsh et al., 2016). The limited size of labelled training data might be the reason that automatic object detection based on deep learning approaches are not yet routinely applied in the field of biomedical EM of embedded samples. First, access to large labelled EM datasets is often limited because the preparation of biological samples for EM and the collection of high-quality EM images is time consuming. Second, reliable labelling of EM data requires specialised skills, experience and time. Third, the scientific questions are often highly specialised and vary depending on the type of structure of interest, sample preparation technique and imaging modality. Another reason for the hesitation of biologists to develop deep learning-based object detection tools might be that collaborations with computer scientists are often not at hand and the opinion prevails that establishing such collaborative projects might take longer than performing the quantification by hand. However, the demand for automatic data analysis tools is steadily increasing especially since the development towards automatic image acquisition tools has started.

In a recent study we presented the first step towards automatic quantification of the three stages of secondary envelopment in TEM images of embedded samples. We showed that automatic detection of virus capsids, irrespective of their envelopment stage (single-class detection), is possible by deep learning based methods even with a small TEM dataset when a transfer learning approach is used (Devan et al., 2019). However, the use of this transfer learning approach in the same manner is not suitable for classification of capsids into the three capsid envelopment stages (multi-class detection) due to high intra-class variabilities, inter-class similarities (Figure 1b), imbalance of naked, budding and enveloped capsids in the training dataset and the training dataset's insufficient size. A common technique to overcome the limitation of small training datasets is to adopt classical data augmentation techniques, such as geometric and intensity transformations of original images (Shorten & Khoshgoftaar, 2019). However, those augmented images intrinsically resemble the original ones, leading to limited learning abilities of the CNN that would not be sufficient to detect the three capsid classes (Shorten & Khoshgoftaar, 2019). Therefore, we applied image synthesis techniques to generate realistic new images that are similar to the real image distribution but with varied object configurations.

In this context, the generative adversarial network (GAN), a type of generative models that has excellently performed in general computer vision tasks (Goodfellow et al., 2014), was used to generate new synthetic images. Generative modelling is an unsupervised machine learning task that involves automatic learning of patterns from input data in a way that the model can automatically generate plausible images resembling the input images. A GAN usually includes a generator and a discriminator network, which are two independent neural networks (Figure 2). The generator network takes a random noise vector as input and generates synthetic images, which are then fed to the discriminator network. The discriminator uses the real and generated images to determine whether the input image is a real image or a generated one. Every time the discriminator notices a difference between the real and generated images, the generator slightly adjusts its parameters to reduce these differences. If the synthetic images cannot be distinguished anymore from the real image by the discriminator, it is given out as new synthetic image.

GAN and its variations were recently proposed for generating high quality realistic looking natural images (Arjovsky, Chintala, & Bottou, 2017; Karras, Aila, Laine, & Lehtinen, 2018; Radford, Metz, & Chintala, 2016; Shaham et al., 2019). Recently, GANs have demonstrated promising results with biological data such as X-ray image synthesis (Galbusera et al., 2018), MRI to CT image translation (Nie et al., 2017), brain MRI image (Dar et al., 2018; Kazuhiro et al., 2018; Sanchez & Vilaplana, 2018), retinal image synthesis (Costa et al., 2017; Yu et al., 2019; Zhao, Li, Maurer-Stroh, & Cheng, 2018), skin lesion image synthesis (Bissoto, Perez, Valle, & Avila, 2018) and synthesis of fluorescence microscopy images of cells (Baniukiewicz, Lutton, Collier, & Bretschneider, 2019; Osokin, Chessel, Salas, & Vaggi, 2017). In the field of EM, Han, Murphy, and Ramanan (2018) produced realistic synthetic EM images in which the positions of cell membranes and mitochondria have been automatically labelled using GAN. Apart from synthetic image generation, GAN has been already used in the EM field for denoising (Su, Zhang, Schawinski, Zhang, & Cianfrocco, 2018) and resolution enhancement (de Haan, Ballard, Rivenson, Wu, & Ozcan, 2019).

In this study, we focused on the development of automatic detection of three different secondary envelopment stages of cytoplasmic HCMV capsids in TEM images. The limited availability of a large labelled dataset prompted us to explore the generation of synthetic labelled TEM images by a GAN. We applied GAN to generate synthetic TEM images containing the three labelled capsid classes that resemble the original image. We then used a deep learning-based

object detection model, the Faster R-CNN, for capsid detection. The augmentation of the CNN training dataset by these synthetic images substantially improved the detection performance. Taken together, we developed an end-to-end CNN approach for automatic detection of three different secondary envelopment stages of HCMV capsids in TEM images of embedded cells.

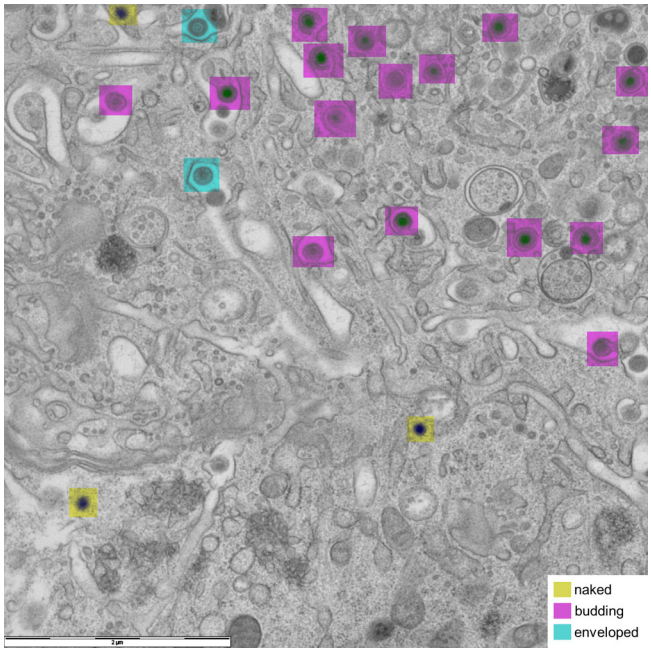


FIGURE 3 One of ten labeled ground truth TEM images showing the three classes of HCMV capsids that was used as input for GAN for generation of synthetic images; labels annotated by human expert in LabelImg were substituted by color-coded bounding boxes corresponding to the three capsid classes

2 | RESULTS

2.1 | Generation of synthetic labelled TEM images

Data labelling to generate ground truth data for training of CNN is usually the most time-consuming and tedious part of the deep learning process. We thought to overcome this hurdle by creating synthetic data that is already labelled. The synthetic TEM images should contain HCMV capsids from the three secondary envelopment stages. By using the SinGAN technique we generated 500 labelled synthetic TEM images with a vast variety in appearance from only 10 single ground truth labelled images (Example in Figure 3). Generation of these images took 10 hr on a Nvidia GeForce GTX 1060 3GB GPU. The 10 ground truth images contained expert-labelled capsids of the three capsid classes naked, budding and enveloped. The synthetically generated images shared some characteristics of the ground truth images, such as a realistic TEM image appearance or the presence of similar structures, but contained minor differences in each of the generated training images, making them unique (Figure S2).

After extraction of the bounding box coordinates from the labelled images, these coordinates and the unlabeled synthetic images were used as training images for the Faster R-CNN detector. An example of an ground truth unlabeled TEM image with one of the 50 corresponding labelled synthetic images and the same synthetic image after removal of the labels is shown in Figure 4. The generation

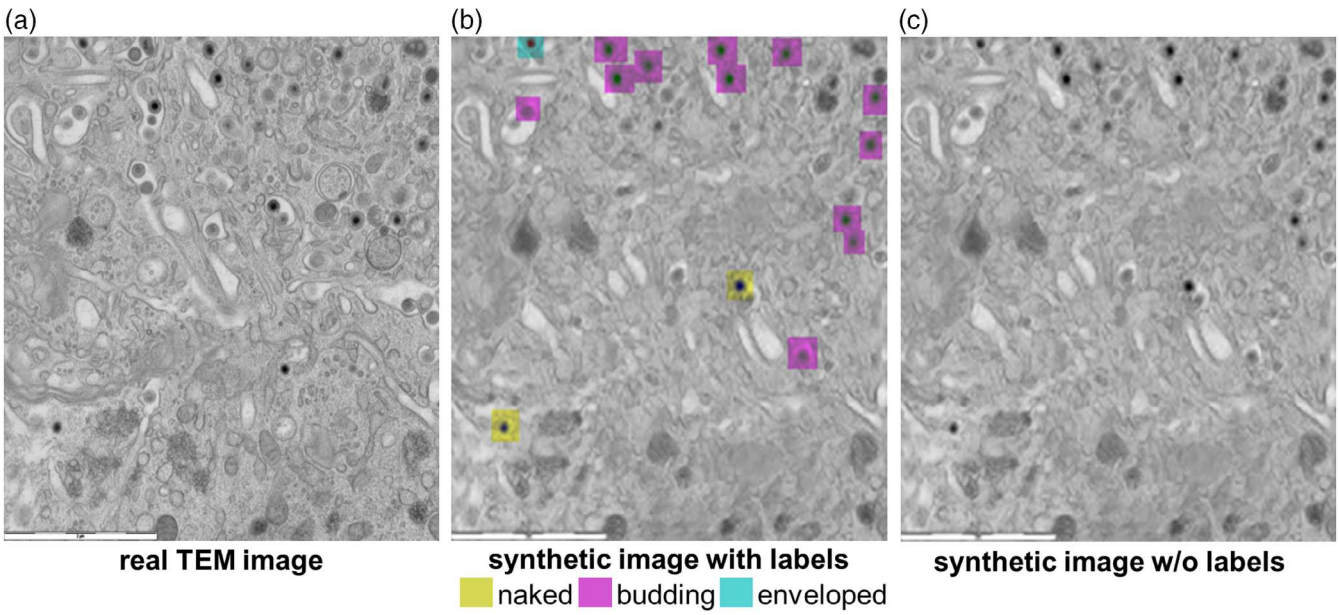


FIGURE 4 Example of a labeled image synthetically generated from a ground truth TEM image. (A) This images is shown without labels for better visibility of structures. (B) Labeled synthetic image generated by GAN. (C) Synthetic image with the labels removed is used as training image for the Faster R-CNN detector together with the bounding box coordinates of the labels

TABLE 1 Details of the dataset for training the Faster R-CNN detector including ground truth and synthetic images

	Ground truth images	Synthetic images	Total images
Number of images	315	500	815
Number of capsids	2,610	6,582	9,192
Naked	767	3,264	4,031
Budding	1,508	2,136	3,644
Enveloped	335	1,182	1,517

TABLE 2 Numbers of true positive, false negative and false positive capsids detected in the two example images in Figure 5 and Figure S4 for differently trained models

Validation image in figure	Model	Number of		
		True positives	False negatives	False positives
6	Gt	18	14	0
	Syn	1	20	2
	Gt+Syn	30	6	1
S4	Gt	13	4	1
	Syn	6	25	1
	Gt+Syn	22	1	1

of 500 synthetic images by SinGAN increased the number of capsids for Faster R-CNN detector training from a total of 2,610 (naked: 767, budding: 1,508, enveloped: 335) to 9,192 capsids (352.2%; naked: 4,031, budding: 3,644, enveloped: 1,517, Table 1).

2.2 | Evaluation of automatic HCMV capsid class detection

To evaluate the impact of synthetic images generated by SinGAN on the performance of our secondary envelopment stage detector, we added the 500 synthetic images to the training dataset to train the Faster R-CNN detector. The training time for the complete dataset of both ground truth and synthetic images was 10 hr on a Nvidia GeForce GTX 1060 3GB GPU. Then, the trained model was used for the detection of capsids in the 35 validation images. The time of detection of a single validation image was 15 seconds. Thus, analysis of all validation images was finished after approximately 9 min. Figure S3 shows example validation images comparing expert detection and automatic detection after training with both truth and synthetic images. In addition, the 35 validation images were analysed with capsid detection models that were trained with only ground truth and only synthetic images, respectively.

Figures 5 and S4 show two representative validation images after detection by an expert and automatic detection by our Faster R-CNN model after training with only synthetic (Syn), only ground truth (Gt) and ground truth and synthetic (Gt+Syn) images. The Syn-trained model showed a poor detection performance as most capsids in the validation images were not detected (Figures 5b and S4b, Table 2), indicating that the Faster R-CNN model is not able to learn well when only synthetic images are supplied.

The Gt-trained model detected more capsids in the validation images, however, detection resulted in substantial numbers of incorrect and missed detections as well as lower confidence scores for individual capsids (Figures 5c and S4c, Table 2). The model was able to correctly identify naked capsids but showed weaknesses in identifying budding and enveloped capsids.

The Gt+Syn-trained model showed superior performance compared to the other two models (Figures 5d and S4d). The Gt+Syn-trained model detected most of the HCMV capsid envelopment classes similarly to the expert-labelled ground truth images (Figures 5a, S3 and S4a), resulting in a high number of true positive detections as well as high confidence scores (Table 2). Budding capsids contributed to the high amounts of false negatives and false positives while naked capsids showed the lowest number of false negatives and false positives. This might be explainable by the intra-class variability which is higher for budding capsids compared to naked and enveloped capsids (Figure 1b). In a few cases, automatic detection failed to detect individual capsids, especially when there were many capsids located close to each other, when the capsid core appeared in another grey value than black, as it is the case for capsids that are not filled with DNA, or when the capsid structures were low on contrast. However, the Gt+Syn-trained model performed well even on images with preparation artefacts (Figure 5a, white areas) and detected capsids that were not labelled by the expert. Most interestingly, the model succeeded in discriminating nuclear and cytoplasmic capsids although they have a similar appearance (Figure S4). The confidence scores for all capsid classes were similar (naked: 52–99%; budding: 50–99%; enveloped: 55–99%) indicating that the model was not biased towards any specific capsid class (Figures 5, S3 and S4). In conclusion, the performance of object detection improved by augmentation of the training dataset by synthetic images.

To further evaluate how the performance of the object detection model increases by adding synthetic images to the training data, we conducted two experiments. In the first experiment we trained the detector model with the available 315 original images as ground truth

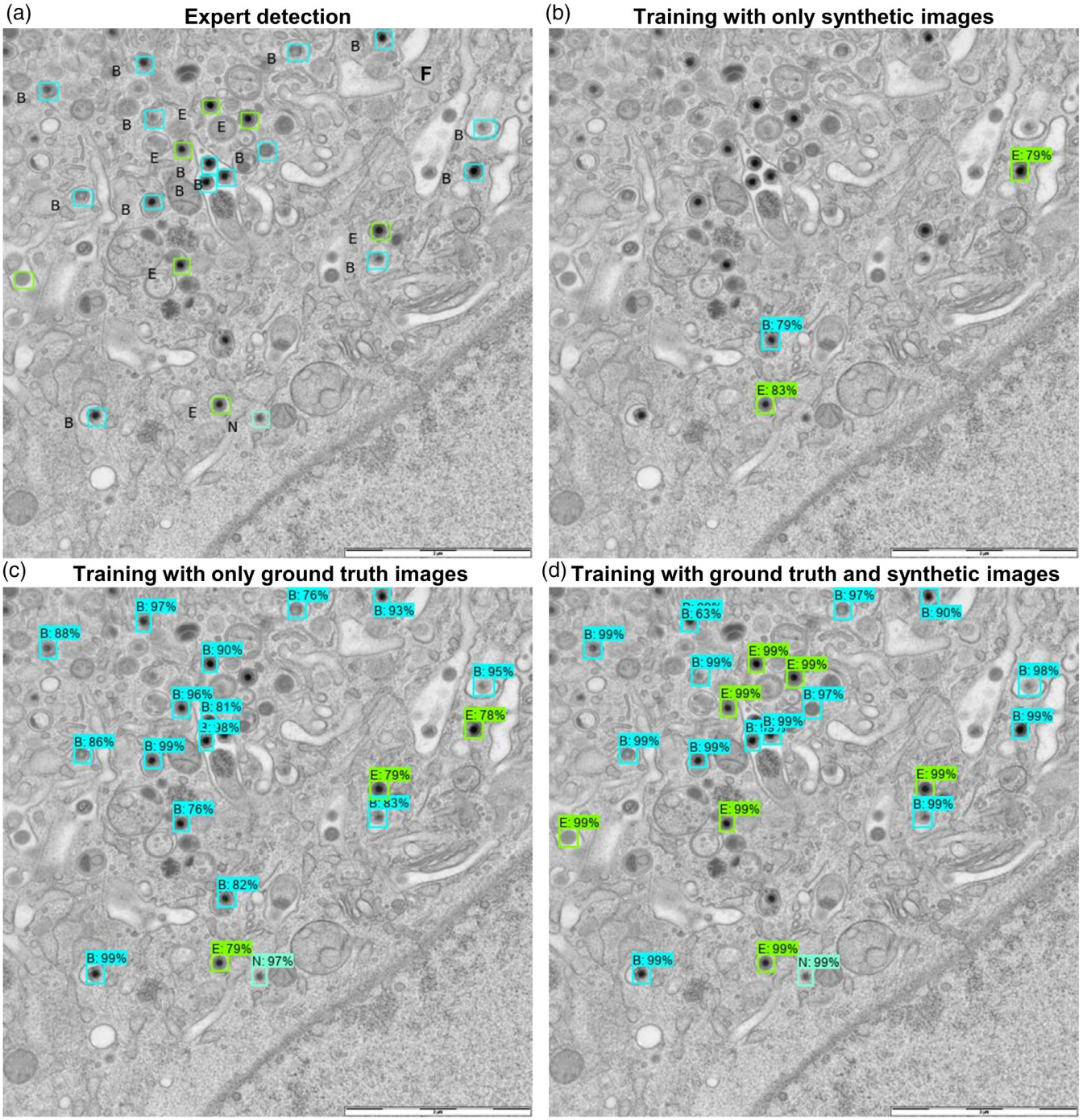


FIGURE 5 Validation image after (A) manual and (BD) automatic capsid detection by Faster R-CNN model trained with different training datasets for comparison of detection efficiencies. Confidence scores are shown for each bounding box. Detection by (B) model trained with only synthetic images, (C) model trained with only ground truth images and (D) model trained with both, ground truth and synthetic images. N: naked, B: budding, E: enveloped

images and called this the baseline model. Then, we gradually added 100 randomly selected synthetic images to the training dataset and trained the detector model again. Figure 6a shows the performance ($AP_{0.50}$ and $AP_{0.75}$) of the HCMV capsid class detection. The baseline model achieved 53.81% for $AP_{0.50}$ and 43.53% for $AP_{0.75}$. As expected, addition of increasing numbers of synthetic images gradually improved the performance of the HCMV capsid detector. For the complete dataset comprising of 315 ground truth images and 500 synthetic images, we managed to achieve 76.48% for $AP_{0.50}$ and 70.01% for $AP_{0.75}$, respectively. Adding more synthetic images achieved higher detection rates.

In the second experiment we kept the number of synthetic images constant at 500. Our baseline model was now the model that was only trained with 500 synthetic images. We then incrementally

added ground truth images to the training dataset in steps of 35 images until 315. The baseline model yielded very low detection rates for virus capsids with 6.10% for $AP_{0.50}$ and 2.30% for $AP_{0.75}$ (Figure 6b). By contrast, adding 35 ground truth images increased the $AP_{0.50}$ and $AP_{0.75}$ to 55.57 and 14.17%, respectively. While $AP_{0.50}$ increased rapidly with just the addition of 35 ground truth images, $AP_{0.75}$ increased to a reasonable value (above 50%) only after the addition of 175 ground truth images to the training dataset. This experiment shows that the addition of increasing numbers of ground truth images to the training dataset of 500 synthetic images gradually improves the HCMV capsid detector performance.

2.3 | DISCUSSION

The field of biological EM often requires quantification to obtain meaningful results. This is usually done manually and therefore time consuming. Thus, quantification tasks in biological EM analyses would greatly benefit from automation. The first step towards this goal is the development of a model for automatic object detection in biological EM images. The state-of-the-art approach for this is machine learning. However, application of machine learning techniques is often stalled by the requirement of large labelled training datasets, which are usually not easily available in such specialised fields such as biological EM. In this study, we demonstrate that the generation of synthetic labelled data by using GAN technology and their addition to detector network training datasets can greatly improve performance of detection. We show the potential of this approach to train a CNN that automatically detects three classes of cytoplasmic HCMV capsids in TEM images of high-pressure frozen, freeze substituted and plastic embedded specimens.

We used the SinGAN architecture to generate labelled synthetic images, instead of classical image synthesis GANs, such as Conditional GAN (Mirza & Osindero, 2014) and Deep Convolutional GAN (DCGAN) (Radford et al., 2016). The incentive for this was that the SinGAN architecture is able to generate multiple synthetic images from a single original image while classical image synthesis GANs require large number of training data for image generation. Importantly, SinGAN generates synthetic images that maintain the global structure and the fine textures of the individual original images, while creating significant structural variability. In addition to this, the high degree of natural diversity among the 10 original ground truth TEM images that were used for synthetic image generation increased the variability to the training dataset. Thus, our SinGAN model learns from highly variable ground truth images and generates synthetic images that are representative of the diverse nature of the ground truth dataset. This is especially important for our approach of secondary envelopment stage classification because the different stages are defined by complex morphological differences (Read et al., 2019; Schauflinger et al., 2013). Preliminary training of a detector model performed before synthetic data was available within this project indicated that extensive geometric data augmentation techniques did not help to improve detection performance (data not shown). Those augmented images intrinsically resemble the original ones, leading to limited learning abilities of the CNN. Thus, extensive geometric data augmentation techniques were not used for training of the detector model. Our detector model was able to learn well with just GAN-based augmentation. This enhanced the model's generalisation

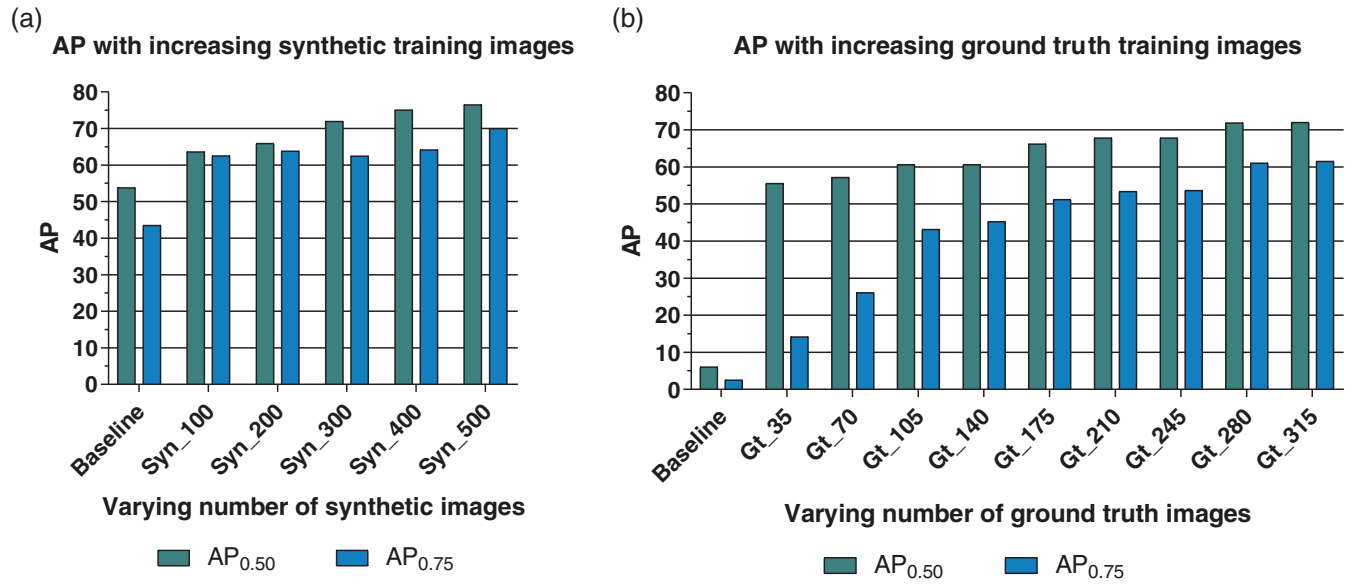


FIGURE 6 Impact of ground truth and synthetic images on performance of the Faster R-CNN network detector. (A) Experiment 1: Model was trained with 315 ground truth (Gt) images (baseline model) or with 315 ground truth images plus the indicated numbers of additional synthetic images (Syn) generated by SinGAN. (B) Experiment 2: Model was trained with 500 synthetic images (baseline model) or with 500 synthetic images plus indicated numbers of ground truth images. For each training outcome, AP0.50 and AP0.75 values are given

capability significantly, forming an important base for future development of the model for other EM related tasks, such as capsid detection in chemically fixed samples, detection of other viruses or cell organelles. Another field of application are viruses imaged by other EM techniques, especially 3D approaches (Risco et al., 2014; Romero-Brey & Bartenschlager, 2015), such as scanning electron microscopy (SEM) from block face (Kantor, Grant, Balaraman, White, & Franz, 2018) or array (Kataoka et al., 2019) tomography, or scanning transmission electron microscopy (STEM) tomography data (Read et al., 2019). These tomography approaches are mostly automated and produce large datasets in a short time. Analysis of this data by automatic object detection is a highly intuitive application of the here described method.

By using the here described approach we obtained reasonable capsid detection results with just a small dataset of ground truth labelled TEM images (Figure 6a). Most capsids are correctly detected and in some cases the model even outperforms the human expert by detecting capsids that were missed in the ground truth expert labelling. We were also pleased to realise that nuclear capsids in the images were not detected by the model, which means that when the model is used in the next development phase for automatic quantifications, the presence of nuclear capsids in TEM images will not influence the results of cytoplasmic capsids. We do not suspect that the model distinguishes nuclear from cytoplasmic capsids based on the different background texture that is found between nucleoplasm and cytoplasm, but rather that the model is able to detect the fine differences between the capsid appearance itself. The fact that also capsids in images with sample preparation artefacts (e.g., knife marks, wrinkles or poor resin infiltration causing holes, see Figure S3) were detected unequivocally demonstrates the robustness of the approach. However, detection performance could further be optimised in the future. For instance, our model was not successful in detecting every capsid for example, in dense clusters or that showed a less electron dense structure than the capsids shown in Figure 1. One reason for this could be that capsids are small (bounding box area of 24 pixel^2) and many deep learning-based object detection models do not work well on very small objects, for example, objects of less than 32 pixel^2 (Tong, Wu, & Zhou, 2020). CNNs generally have several convolutional layers followed by a pooling layer. Most object detection networks use these layers, in which the resolution of input images is reduced as it progresses through the layers. Accordingly, small object features that are extracted at the very first layers may be lost in the middle of the network and not used for detection. Despite this limitation, our model has been able to give very good performance in detecting capsids. The automatic detection of small objects could either be optimised when future images are recorded in slightly higher magnifications or by modifying the network.

When we evaluated the impact of synthetic images for training, we observed that the addition of the synthetic images to the ground truth dataset resulted in a major improvement in the Faster R-CNN model's performance (Figures 5, 6a and S4), that resembled the performance of the human expert. The more synthetic images where added, the more $AP_{0.50}$ and $AP_{0.75}$ values increased. This suggests that the performance of the model is not yet saturated and may further increase with the addition of more than 500 synthetic images. This is underlined by the basic principle of deep learning technology stating that the larger the training dataset used for CNN model training, the better will be the model performance (Shorten & Khoshgoftaar, 2019). In this study, we used only 500 synthetic images to show that a limited number of synthetic images can already greatly improve the model's performance. Here we show that our model is able to learn how to classify HCMV capsids from synthetic images and that it generalised well on our validation images. Our model was able to return similar confidence score ranges for all the three capsid classes. This indicates that the model does not exhibit any bias towards a specific capsid class.

We furthermore observed that the model performed very poorly with just synthetic images (Syn) as training dataset (Figure 6b). However, the performance of the model increased already with just the addition of a small number of ground truth images. A good value of $AP_{0.75}$ (above 50%) was only obtained after the addition of a larger number of ground truth data compared to $AP_{0.50}$. This is because, after the addition of 35 ground truth training images, the capsids with an IoU overlap of 50% could be detected and classified, however, since $AP_{0.75}$ required an overlap of 75%, a larger number of ground truth images is needed to train the model to detect and classify capsids at a higher cut-off value.

In the Gt+Syn-trained model, information learned from both synthetic and ground truth capsids contributed to its performance. These were 9,192 capsids in 815 images (Table 1). Thus, we can predict that similarly good detection could be reached with only ground truth capsids (without generation and addition of synthetic capsids) if a similar number of capsids were used for training (approximately 9,000 capsids with ideally 3,000 for each capsid class). For this, several hundred additional TEM images need to be recorded and manually labelled. As the availability of labelled images is scarce in most biological fields, addition of synthetic images to the training dataset considerably speeds up the development of deep learning-based capsid detection model.

Taken together, in this work we explored the possibility of generating synthetic labelled TEM images using GAN. We were able to obtain good quality realistic labelled images using just 10 ground truth images. We demonstrated that these synthetically generated images can be used to enlarge a training dataset and therefore improve the learning and generalisation capability of the deep learning model. While there are a few techniques published for synthetic image generation (Mirza & Osindero, 2014; Radford et al., 2016; Shaham et al., 2019), to our knowledge, generation of already labelled synthetic images is a novel approach in the biological EM field. Generation of synthetic images will be highly useful in applications with limited data availability, such as studies that involve specialised imaging data. Our method could be further adopted for the detection of various other structures in the field of biological image analysis, such as nuclei and mitochondria, and for other types of microscopy, including 3D electron microscopy.

2.4 | EXPERIMENTAL PROCEDURES

2.4.1 | EM specimen preparation and image acquisition

EM samples were prepared as previously described by Dietz et al. (2018). Briefly, human foreskin fibroblasts (HFFs) were used before passage 23 and maintained in minimal essential medium (Gibco, ThermoFisher Scientific Inc., Waltham, MA, USA) supplemented with 10% fetal calf serum (Biochrom AG, Berlin, Germany), 2 mM L-glutamine (PAA Laboratories GmbH, Pasching/Linz, Austria), 1% penicillin–streptomycin (PAA Laboratories) and 1% non-essential amino acids (Biochrom AG). For TEM experiments, HFFs were seeded on carbon coated sapphire discs (Engineering Office M. Wohlwend GmbH, Sennwald, Switzerland). After attachment of the cells for 24 hr HFFs were infected with different recombinant versions of the HCMV bacterial artificial chromosome (BAC) clone TB40-BAC4 (Sinzger, Digel, & Jahn, 2008) with a multiplicity of infection of 1 and incubated for three or 5 days at 37°C and 5% CO_2 (Schauflinger et al., 2013; Villinger, Neusser, Kranz, Walther, & Mertens, 2015). Infected cells on sapphire disks were prepared for TEM by high pressure freezing and freeze substitution as described in Villinger et al. (2014). Ultrathin sections (75–80 nm) were cut parallel to the sapphire disk and mounted on formvar and carbon coated single slot grids. Sections were analysed with a Jeol 1400 TEM (Jeol, Tokyo, Japan) at an accelerating voltage of 120 kV, and the images were digitally recorded with a Veleta camera (Olympus, Münster, Germany). The magnification of the TEM images ranged from 25,000× with a pixel size of 2.78 nm to 120,000× with a pixel size of 0.57 nm. A total of 350 TEM images were acquired. The TEM images were 8-bit grayscale TIFF with a dimension of 2048 × 2048 pixels.

2.4.2 | Ground truth dataset preparation

In order to obtain the ground truth dataset for generation of synthetic images and detector training, all 350 TEM images were manually annotated with ground truth bounding boxes for the three HCMV capsid envelopment stages by a biologist via the LabelImg tool (Tzutalin, 2015)

TABLE 3 Number of labelled capsids in the ground truth dataset (350 images)

Dataset	Number of images	Naked	Budding	Enveloped
Training	315	767	1,508	335
Validation	35	118	187	54

(Figure 1b). These images were randomly divided into a training and validation dataset with a ratio of 9:1. Our training dataset contains a diverse set of EM images, which varied in terms of magnification, contrast, brightness and noise levels to account for the variability that is typical for biological EM. A total of 2,969 cytoplasmic HCMV capsids were labelled in 350 images. The number of capsids for each envelopment stage in our dataset is summarised in Table 3.

2.4.3 | Training methodology

Generation of synthetic images and training and validation of automatic capsid detection was done by using a Nvidia GeForce GTX 1060 3GB GPU with Tensorflow 1.14.0 and Python 3.6.7. Figure 7 shows the pipeline for the automatic detection of HCMV envelopment stages in TEM images.

2.4.4 | Generation of synthetic labelled TEM images

We used only 10 randomly selected ground truth (original) images for synthetic image generation by GAN. These 10 expert-labelled TEM images showed large variations in appearance and contained the three classes of HCMV capsids. All images were converted from grayscale to RGB and the three expert-labelled capsid classes in the image were labelled with colour-coded bounding boxes before the images were used as input for the GAN. The GAN network generated synthetic images with capsids, which were automatically colour-coded according to the capsid class they belonged to.

For the generation of synthetic TEM images, we used the GAN variant called SinGAN with the published settings (Shaham et al., 2019). This technique uses an unconditional generative model that has the capability to learn from a single image. This is an important factor for most EM applications as large ground truth datasets are often not available. SinGAN contains a pyramid of fully convolutional GANs, which are responsible for learning the patch distribution at different scales of the image ranging from coarse to fine. This task differs from a classical GAN setting as in SinGAN the training samples are patches of a single image, rather than whole image samples from a database.

The images were first resized to 512 x 512 pixels using bicubic downsampling for training. Each SinGAN training session for one ground truth image produced 50 synthetic variations of this TEM image. Thus, a total of 500 synthetic images were produced from 10 ground truth images. The synthetic images had a size of

FIGURE 7 Pipeline for the automatic detection of HCMV envelopment stages. Step 1 and 3 are performed manually while steps 2, 4 and 5 are automatically performed by deep learning models

256 × 256 pixels. The capsids in the synthetic TEM images were automatically labelled with the bounding boxes according to the colour-coded capsid classes of the ground truth images. Coordinates of the bounding boxes were extracted from the synthetic images and converted to PASCAL visual object classes format as it is the common format used by many deep learning-based object detectors (Everingham et al., 2015). After extraction of the coordinates, the coloured boxes were removed from the synthetic images and the images were converted back to grayscale. The grayscale synthetic images and bounding box coordinates were used as part of the training data.

2.4.5 | Detection of HCMV capsid envelopment stages in TEM images

The Faster R-CNN (Ren, He, Girshick, & Sun, 2017) architecture was used for the automated detection of the HCMV capsid envelopment stages in the validation images. Faster R-CNN uses a fully convolutional data driven approach for generating region proposals by using a region proposal network (RPN). The RPN generates bounding boxes of possible objects in the images. The Faster R-CNN detector classifies and refines these bounding boxes around those proposals. The ResNet-101 was used as feature extractor for our model.

Before training Faster R-CNN with the respective ground-truth data, we did not perform any preprocessing on the training images so that the system is able to also learn from noisy, low contrast images. Data augmentation by horizontal flip was performed to increase the diversity of the training dataset. The input images had a size of 512 × 512 pixels. We used the momentum optimizer with a value of 0.90. The initial learning rate was set to 0.0003. The mini batch size was set to 1. The feature stride was set to 16. We fine-tuned the models with pre-trained weights. The non-maximum suppression method was used for detection. We used the transfer learning mechanism where we first initialized CNN frameworks with pre-trained models based on ImageNet dataset and then fine-tuned all layers of the network using our own training dataset (Deng et al., 2009; Devan et al., 2019). The model was trained for 50,000 steps.

2.4.6 | Evaluation metrics

The trained Faster R-CNN model for HCMV envelopment stage detection was tested on a validation dataset consisting of 35 randomly chosen ground truth TEM images. The model evaluation was performed using the MS COCO evaluation metric, which is the average precision (AP). This is the gold standard evaluation metric for deep learning-based object detection (Everingham et al., 2015; Everingham, Van Gool, Williams, Winn, & Zisserman, 2010; Hoeim, Chodpathumwan, & Dai, 2012; Lin et al., 2014; Russakovsky et al., 2015).

For object detection (capsid detection) two tasks need to be performed: object classification and object localization. Object classification means that the model tests if a capsid is present in an image and allocates it to its corresponding object class (naked, budding or enveloped). Object localization predicts the coordinates of the capsid and draws a bounding box around it. The accuracy of localization is evaluated by Intersection over Union (IoU), which measures the degree of overlap between the coordinates of the predicted bounding box and the ground truth bounding box (Rezatofighi et al., 2019, Figure S1). The IoU is defined by Equation (1), where S_P is the predicted bounding box and S_G represents the ground truth bounding box.

$$IoU = \frac{S_P \cap S_G}{S_P \cup S_G}. \quad (1)$$

For each capsid detection, an IoU score is computed. The object detector predicts bounding boxes, each associated with a confidence score. The confidence score is the probability that a capsid class appears in that bounding box (Everingham et al., 2015; Ren et al., 2017), that is, the probability that the capsid in this particular bounding box can be classified as an enveloped, budding or naked capsid. Accordingly, we set an IoU threshold to turn this probability into a classification. This means detections with a confidence score above the threshold are considered true positives, while the ones below the threshold are considered false positives. In our experiments, two IoU thresholds of 50% and 75% are set, which represent a bounding box overlap of 50% and 75%, respectively. True positives are correctly detected capsids, false positives are non-capsid structures which were detected as capsids and false negatives are undetected capsids. In our example with a threshold of 50%, this means that a capsid with a bounding box for enveloped capsid and a confidence score of 45% is not classified as enveloped whereas a capsid with a confidence score of 80% is classified as enveloped (true positive). AP stands for average precision. It is a metric that evaluates the performance of object detectors by combining precision and recall.

Precision (P), also referred to as positive predictive value, is the probability of the predicted bounding boxes matching the ground truth bounding boxes and describes the ratio of correctly detected capsids to the total detected capsids. Precision scores range from 0 to 1. A precision close to 1 means that almost all detected capsids were detected correctly, that is, corresponding to the classes described in the ground truth data. Recall (R) is the true positive rate and measures the probability of ground truth capsids being correctly detected. Recall ranges from 0 to 1. A recall score close to 1 means that almost all ground truth capsids were detected correctly. The mathematical definitions of precision and recall are shown in Equations (2) and (3).

$$\text{Precision}\,(\text{P}) = \text{TP}/\text{TP} + \text{FP} \quad (2)$$

$$\text{Recall}\,(\text{R}) = \text{TP}/\text{TP} + \text{FN}. \quad (3)$$

The AP value summarises the precision-recall curve by averaging precision across recall values of the capsids from 0 to 1 as given by its mathematical definition in Equation (4).

$$AP = \int_0^1 P(R)dr. \quad (4)$$

This means that for assessing the $AP_{0.50}$ value only capsids with an IoU and a confidence score of 50% and above are considered, whereas $AP_{0.75}$ is computed with only capsids with an IoU and confidence score of 75% or above. Thus, $AP_{0.75}$ values are a more conservative measure than $AP_{0.50}$ values.

ACKNOWLEDGEMENTS

We would like to thank Renate Kunz, Reinhard Weih, Nadia Wenske and Gesa Freimann of the Central Facility for Electron Microscopy for their support. This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Projektnummer 316249678 – SFB 1279 and the Baden-Württemberg Stiftung, Project ABEM.

CONFLICT OF INTEREST

The authors have no conflicts of interest to declare.

AUTHOR CONTRIBUTIONS

Kavitha Shaga Devan and Timo Ropinski: Conceived the idea presented in this work. **Kavitha Shaga Devan:** Designed the algorithm, data analysis and performed all the experiments. **Timo Ropinski:** Contributed to the design of the experiments. **Hans A. Kestler:** Contributed towards the validation of the methods used in this work. **Paul Walther, Clarisssa Read and Jens von Einem:** Provided the TEM images used in this work. **Clarissa Read:** Performed the manual labelling for all ground truth images in this work as well as the verification and analysis of the results. **Paul Walther and Hans A. Kestler:** Provided supervision and guidance in developing this research work. **Kavitha Shaga Devan and Clarissa Read:** Wrote the manuscript with support from Jens von Einem. All authors discussed the results and contributed to the manuscript.

DATA AVAILABILITY STATEMENT

The Python source codes with Tensorflow used in this work can be accessed at https://github.com/Kavit212/HCMV_Capsid_Detection and a selection of ground truth and synthetic images are provided for training. The full datasets are available upon request. Contact details are available on the GitHub page.

ORCID

Kavitha Shaga Devan https://orcid.org/0000-0003-0745-9167
Clarissa Read https://orcid.org/0000-0002-4632-2684

REFERENCES

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. *ArXiv: 1701.07875 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1701.07875

Baniukiewicz, P., Lutton, E. J., Collier, S., & Bretschneider, T. (2019). Generative adversarial networks for augmenting training data of microscopic cell images. *Frontiers in Computer Science*. https://doi.org/10.3389/fcomp.2019.00010

Bissoto, A., Perez, F., Valle, E., & Avila, S. (2018). Skin lesion synthesis with generative adversarial networks. *ArXiv:1902.03253 [Cs]*, (Vol.11041, pp. 294–302). Retrieved from https://doi.org/10.1007/978-3-030-01201-4_32

Chevillotte, M., von Einem, J., Meier, B. M., Lin, F.-M., Kestler, H. A., & Mertens, T. (2010). A new tool linking human cytomegalovirus drug resistance mutations to resistance phenotypes. *Antiviral Research*, *85* (2), 318–327. https://doi.org/10.1016/j.antiviral.2009.10.004

Costa, P., Galdran, A., Meyer, M. I., Abràmoff, M. D., Niemeijer, M., Mendonça, A. M., & Campilho, A. (2017). Towards adversarial retinal image synthesis. *ArXiv:1701.08974 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1701.08974

Dar, S. U. H., Yurt, M., Karacan, L., Erdem, A., Erdem, E., & Çukur, T. (2018). Image synthesis in multi-contrast MRI with conditional generative adversarial networks. *ArXiv:1802.01221 [Cs]*. Retrieved from http://arxiv.org/abs/1802.01221

de Haan, K., Ballard, Z. S., Rivenson, Y., Wu, Y., & Ozcan, A. (2019). Resolution enhancement in scanning electron microscopy using deep learning. *Scientific Reports*, *9*(1), 12050. https://doi.org/10.1038/s41598-019-48444-2

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). *ImageNet: A large-scale hierarchical image database*. Paper presented at 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 248–255). Retrieved from https://doi.org/10.1109/CVPR.2009.5206848

Devan, K. S., Walther, P., von Einem, J., Ropinski, T., Kestler, H. A., & Read, C. (2019). Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. *Histochemistry and Cell Biology*, *151*(2), 101–114. https://doi.org/10.1007/s00418-018-1759-5

Dietz, A. N., Villinger, C., Becker, S., Frick, M., & von Einem, J. (2018). A tyrosine-based trafficking motif of the tegument protein pUL71 is crucial for human cytomegalovirus secondary envelopment. *Journal of Virology*, *92*(1), e00907–e00917. https://doi.org/10.1128/JVI.00907-17

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015). The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, *88*(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, *111*(1), 98–136. https://doi.org/10.1007/s11263-014-0733-5

Galbusera, F., Niemeyer, F., Seyfried, M., Bassani, T., Casaroli, G., Kienle, A., & Wilke, H.-J. (2018). Exploring the potential of generative adversarial networks for synthesizing radiological images of the spine to be used in in silico trials. *Frontiers in Bioengineering and Biotechnology*, *6*, 53. https://doi.org/10.3389/fbioe.2018.00053

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., …Bengio, Y. (2014). Generative adversarial networks. *ArXiv: 1406.2661 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1406.2661

Han, L., Murphy, R. F., & Ramanan, D. (2018, March). *Learning generative models of tissue organization with supervised GANs*. Paper presented at 2018 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 682–690). Retrieved from https://doi.org/10.1109/WACV.2018.00080

Hoeim, D., Chodpathumwan, Y., & Dai, Q. (2012). Diagnosing errors in object detectors. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, & C. Schmid (Eds.), *Computer vision – ECCV 2012. ECCV 2012* (Vol. 7574). Berlin, Germany: Lecture notes in computer science, Springer. Retrieved from https://doi.org/10.1007/978-3-642-33712-3_25

Kantor, A. M., Grant, D. G., Balaraman, V., White, T. A., & Franz, A. W. E. (2018). Ultrastructural analysis of Chikungunya virus dissemination from the Midgut of the yellow fever mosquito, *Aedes aegypti*. *Viruses*, *10*(10), 571. https://doi.org/10.3390/v10100571

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. *ArXiv:1710.10196 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1710.10196

Kataoka, M., Ishida, K., Ogasawara, K., Nozaki, T., Satoh, Y. I., Sata, T., … Nakajima, N. (2019). Serial section array scanning electron microscopy analysis of cells from lung autopsy specimens following fatal A/H1N1 2009 pandemic influenza virus infection. *Journal of Virology*, *93*(19), e00644. https://doi.org/10.1128/JVI.00644-19

Kazuhiro, K., Werner, R. A., Toriumi, F., Javadi, M. S., Pomper, M. G., Solnes, L. B., … Rowe, S. P. (2018). Generative adversarial networks for the creation of realistic artificial brain magnetic resonance images. *Tomography (Ann Arbor, Mich.)*, *4*(4), 159–163. https://doi.org/10.18383/j.tom.2018.00042

Kestler, H. A, Core Unit Bioinformatics, Institute of Medical Systems Biology, & Institute of Virology (2013). MRA-Mutation Resistance Analyzer Software Tool. *University of Ulm*, https://www.informatik.uni-ulm.de/ni/mitarbeiter/HKestler/mra/app/index.php?plugin=form.

Kuan, M. I., O'Dowd, J. M., Chughtai, K., Hayman, I., Brown, C. J., & Fortunato, E. A. (2016). Human Cytomegalovirus nuclear egress and secondary envelopment are negatively affected in the absence of cellular p53. *Virology*, *497*, 279–293. https://doi.org/10.1016/j.virol.2016.07.021

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – ECCV 2014* (pp. 740–755). Basel, Switzerland: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-10602-1_48

Mach, M., Osinski, K., Kropff, B., Schloetzer-Schrehardt, U., Krzyzaniak, M., & Britt, W. (2007). The Carboxy-terminal domain of glycoprotein N of human cytomegalovirus is required for Virion morphogenesis. *Journal of Virology*, *81*(10), 5212–5224. https://doi.org/10.1128/JVI.01463-06

Mettenleiter, T. C. (2004). Budding events in herpesvirus morphogenesis. *Virus Research*, *106*(2), 167–180. https://doi.org/10.1016/j.virusres.2004.08.013

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *ArXiv:1411.1784 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1411.1784

Nie, D., Trullo, R., Lian, J., Petitjean, C., Ruan, S., Wang, Q., & Shen, D. (2017). Medical image synthesis with context-aware generative adversarial networks. In M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins, & S. Duchesne (Eds.), *Medical image computing and computer assisted intervention – MICCAI 2017* (pp. 417–425). Basel, Switzerland: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-66179-7_48

Osokin, A., Chessel, A., Salas, R. E. C., & Vaggi, F. (2017). GANs for biological image synthesis. *ArXiv:1708.04692 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1708.04692

Phillips, S. L., & Bresnahan, W. A. (2012). The human cytomegalovirus (HCMV) tegument protein UL94 is essential for secondary envelopment of HCMV Virions. *Journal of Virology*, *86*(5), 2523–2532. https://doi.org/10.1128/JVI.06548-11

Phillips, S. L., Cygnar, D., Thomas, A., & Bresnahan, W. A. (2012). Interaction between the human cytomegalovirus tegument proteins UL94 and UL99 is essential for virus replication. *Journal of Virology*, *86*(18), 9995–10005. https://doi.org/10.1128/JVI.01078-12

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *ArXiv:1511.06434 [Cs]*. Retrieved from http://arxiv.org/abs/1511.06434

Read, C., Schauflinger, M., Nikolaenko, D., Walther, P., & von Einem, J. (2019). Regulation of human cytomegalovirus secondary envelopment by a C-terminal tetralysine motif in pUL71. *Journal of Virology*, *93*(13). https://doi.org/10.1128/JVI.02244-18

Read, C., Walther, P., & von Einem, J., et al. (2020). Quantitative electron microscopy to study HCMV morphogenesis. In A. D. Yurochko (Ed.), *Human cytomegaloviruses*, Methods in Molecular Biology 2244, 2, Berlin, Germany: Springer. Retrieved from https://doi.org/10.1007/978-1-0716-1111-1_14

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. *ArXiv:1902.09630 [Cs]*. Retrieved from http://arxiv.org/abs/1902.09630

Risco, C., de Castro, I. F., Sanz-Sánchez, L., Narayan, K., Grandinetti, G., & Subramaniam, S. (2014). Three-dimensional imaging of viral infections. *Annual Review of Virology*, *1*, 453–473. https://doi.org/10.1146/annurev-virology-031413-085351

Romero-Brey, I., & Bartenschlager, R. (2015). Viral infection at high magnification: 3D electron microscopy methods to analyze the architecture of infected cells. *Viruses*, *7*, 6316–6345. https://doi.org/10.3390/v7122940

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *ArXiv:1409.0575 [Cs]*. Retrieved from https://arxiv.org/abs/1409.0575

Sanchez, I., & Vilaplana, V. (2018). Brain MRI super-resolution using 3D generative adversarial networks. *ArXiv:1812.11440 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1812.11440

Schauflinger, M., Villinger, C., Mertens, T., Walther, P., & von Einem, J. (2013). Analysis of human cytomegalovirus secondary envelopment by advanced electron microscopy. *Cellular Microbiology*, *15*(2), 305–314. https://doi.org/10.1111/cmi.12077

Seo, J.-Y., & Britt, W. J. (2007). Cytoplasmic envelopment of human cytomegalovirus requires the postlocalization function of tegument protein pp28 within the assembly compartment. *Journal of Virology*, *81*(12), 6536–6547. https://doi.org/10.1128/JVI.02852-06

Shaham, T. R., Dekel, T., & Michaeli, T. (2019). SinGAN:Learning a Generative Model from a Single Natural Image. *IEEE International Conference on Computer Vision, Seoul, Korea, October 27-November 2 2019*, arXiv:1905.01164v2.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Sinzger, C., Digel, M., & Jahn, G. (2008). Cytomegalovirus cell tropism. *Current Topics in Microbiology and Immunology*, *325*, 63–83. https://doi.org/10.1007/978-3-540-77349-8_4

Smith, R. M., Kosuri, S., & Kerry, J. A. (2014). Role of human cytomegalovirus tegument proteins in virion assembly. *Viruses*, *6*(2), 582–605. https://doi.org/10.3390/v6020582

Su, M., Zhang, H., Schawinski, K., Zhang, C., & Cianfrocco, M. A. (2018). Generative adversarial networks as a tool to recover structural information from cryo-electron microscopy data. *BioRxiv*. Retrieved from https://doi.org/10.1101/256792

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the inception architecture for computer vision. *ArXiv:1512.00567 [Cs]*. Retrieved from http://arxiv.org/abs/1512.00567

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., & Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, *35*(5), 1299–1312 10.1109/TMI.2016.2535302

Tang, Z., Chuang, K. V., DeCarli, C., Jin, L.-W., Beckett, L., Keiser, M. J., & Dugger, B. N. (2019). Interpretable classification of Alzheimer's disease pathologies with a convolutional neural network pipeline. *Nature*

Communications, *10*(1), 2173. https://doi.org/10.1038/s41467-019-10212-1

Tong, K., Wu, Y., & Zhou, F. (2020). Recent advances in small object detection based on deep learning: A review. *Image and Vision Computing*, *97*, 103910. https://doi.org/10.1016/j.imavis.2020.103910

Tzutalin. (2015). *LabelImg*. Retrieved from https://github.com/tzutalin/labelImg

Villinger, C., Neusser, G., Kranz, C., Walther, P., & Mertens, T. (2015). 3D analysis of HCMV induced-nuclear membrane structures by FIB/SEM tomography: Insight into an unprecedented membrane morphology. *Viruses*, *7*(11), 5686–5704. https://doi.org/10.3390/v7112900

Villinger, C., Schauflinger, M., Gregorius, H., Kranz, C., Höhn, K., Nafeey, S., & Walther, P. (2014). Three-dimensional imaging of adherent cells using FIB/SEM and STEM. *Methods in Molecular Biology (Clifton, N.J.)*, *1117*, 617–638. https://doi.org/10.1007/978-1-62703-776-1_27

Wainberg, M., Merico, D., Delong, A., & Frey, B. J. (2018). Deep learning in biomedicine. *Nature Biotechnology*, *36*(9), 829–838. https://doi.org/10.1038/nbt.4233

Yu, Z., Xiang, Q., Meng, J., Kou, C., Ren, Q., & Lu, Y. (2019). Retinal image synthesis from multiple-landmarks input with generative adversarial networks. *BioMedical Engineering OnLine*, *18*(1), 62. https://doi.org/10.1186/s12938-019-0682-x

Zhao, H., Li, H., Maurer-Stroh, S., & Cheng, L. (2018). Synthesizing retinal and neuronal images with generative adversarial nets. *Medical Image Analysis*, *49*, 14–26. https://doi.org/10.1016/j.media.2018.07.001

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Shaga Devan K, Walther P, von Einem J, Ropinski T, A. Kestler H, Read C. Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network. *Cellular Microbiology*. 2021;23:e13280. https://doi.org/10.1111/cmi.13280

7. Weighted average ensemble-based semantic segmentation in biological electron microscopy images [SD-3]

This chapter has been published as a peer-reviewed journal paper.

Shaga Devan K, Kestler HA, Read C, Walther P. Weighted average ensemble-based semantic segmentation in biological electron microscopy images. Histochem Cell Biol. 2022. 158:447–462. https://doi.org/10.1007/s00418-022-02148-3. PMID: 35988009

Synopsis: This work develops an end-to-end deep learning network for the automated semantic segmentation of biological structures in both transmission and scanning electron microcopy images using a weighted average-ensemble approach.

Contributions of thesis author: Algorithm design and implementation, computational experiments and manuscript writing.

Supplementary materials of the publication can be accessed at the above mentioned DOI link.

Histochemistry and Cell Biology
https://doi.org/10.1007/s00418-022-02148-3

ORIGINAL PAPER

Weighted average ensemble-based semantic segmentation in biological electron microscopy images

Kavitha Shaga Devan[1] · **Hans A. Kestler**[2] · **Clarissa Read**[1,3] · **Paul Walther**[1]

Accepted: 8 August 2022

Abstract

Semantic segmentation of electron microscopy images using deep learning methods is a valuable tool for the detailed analysis of organelles and cell structures. However, these methods require a large amount of labeled ground truth data that is often unavailable. To address this limitation, we present a weighted average ensemble model that can automatically segment biological structures in electron microscopy images when trained with only a small dataset. Thus, we exploit the fact that a combination of diverse base-learners is able to outperform one single segmentation model. Our experiments with seven different biological electron microscopy datasets demonstrate quantitative and qualitative improvements. We show that the Grad-CAM method can be used to interpret and verify the prediction of our model. Compared with a standard U-Net, the performance of our method is superior for all tested datasets. Furthermore, our model leverages a limited number of labeled training data to segment the electron microscopy images and therefore has a high potential for automated biological applications.

Keywords Artificial intelligence · Deep learning · Automated image analysis · Electron microscopy · Semantic segmentation · Ensemble-based machine learning

Introduction

Comprehensive analysis of organelles, cell structures, and viral particles leads to understanding of various pathological processes, enabling discoveries, and insights into disease mechanisms. Electron microscopy (EM) has been proven to be a valuable method for biologists to analyze these biological structures in high resolution. In recent years, advancements in this field have enabled the acquisition of large volumes of three-dimensional EM data (Villinger et al. 2014; Kubota et al. 2018; Zheng et al. 2018; Maniates-Selvin et al. 2020; Read et al. 2021). To quantify these large volumes of EM data, biologists have routinely used segmentation tools to obtain critical information about the morphological parameters of the organelles, cell structures, and viral particles. However, manual segmentation of large EM datasets requires expert input, and is tedious as well as labor-intensive. To address this issue, attempts have been made to develop semi-automated segmentation methods using machine learning approaches such as ilastik (Sommer et al. 2011) and DeepMIB (Belevich et al. 2016).

While these approaches successfully improved the rate of segmentation, a considerable amount of manual interaction is required for corrections and quality control. This is because the inherently low contrast in EM images and the variations in appearances in biological structures and sample preparation artifacts could lead to segmentation inaccuracies. Therefore, the lack of fully automatic methods for EM segmentation impedes the routine use for quantitative analysis of biological structures, and there is a demand for the development of accurate and efficient tools for the automatic quantification of EM images.

Recently, deep learning (DL) methods including the above-mentioned approaches have shown incredible success in a wide variety of biomedical image analysis applications (Webb 2018; Ching et al. 2018; Tang et al. 2019; Tian et al. 2021). Convolution neural networks (CNNs) are a class of

✉ Kavitha Shaga Devan
kavitha.shaga-devan@uni-ulm.de

[1] Central Facility of Electron Microscopy, Ulm University, Albert Einstein-Allee 11, 89081 Ulm, Germany

[2] Medical Systems Biology, Ulm University, Albert Einstein-Alee 11, 89081 Ulm, Germany

[3] Institute of Virology, Ulm University Medical Center, Albert Einstein-Allee 11, 89081 Ulm, Germany

Published online: 20 August 2022

deep neural networks, which are most commonly applied to analyze visual images. The main advantage of the CNN approach is that it utilizes raw images and expert-labeled data instead of hand-crafted feature vectors that require a high level of domain expertise. As a result, it can automatically learn high-level discriminant features for visual pattern recognition tasks from the input data (Tajbakhsh et al. 2016). The success of CNNs has prompted the development of several deep architectures for the semantic segmentation of images, whereby each pixel is classified into a specific class. CNN-based semantic segmentation has been proven to outperform traditional segmentation methods (Mahony et al. 2020).

In this context, U-Net (Ronneberger et al. 2015) has been widely used by the biomedical image analysis community and is regarded as one of the most successful architectures for semantic segmentation. Heinrich et al. (2021) developed a U-Net-based 3D segmentation model that is able to segment 35 different cellular organelle classes in focused ion beam scanning electron microscopy (FIB-SEM) images. Modified versions of U-Net have also been successfully used to segment various biological structures such as cells, small extracellular vesicles, and mitochondria in EM images (Casser et al. 2020; Fischer et al. 2020).

Monchot et al. (2021) utilized the Mask R-CNN architecture to segment titanium dioxide particles in the form of agglomerates in scanning electron microscopy (SEM) images. SegNet, a Bayesian-based architecture, was introduced by Koobragade and Agarwal (2018) to carry out multi-class segmentation in serial section transmission electron microscopy (ssTEM) images. George et al. (2021) developed CASSPER, a DL tool for the automated segmentation of protein particles in cryogenic transmission electron microscopy (cryo-TEM) images. The rise in the application of deep CNNs for EM images prompted Kharabag (2021) to compare the performance of four DL architectures for the semantic segmentation of HeLa cells in serial block-face scanning electron microscopy (SBFSEM) images. On the other hand, Horwath et al. (2020) studied a variety of CNN architectures to define the most important features of DL models for the efficient segmentation of structures in TEM images.

Most of the currently available semantic segmentation methods require a large number of representative ground truth data, in order to be successful and generalize well on unseen images. Generalization is the ability of the model to perform well on unseen data and is an essential characteristic of a successful DL model. Therefore, a large number of images have to be manually labeled to provide sufficient ground truth for algorithm learning. However, labeling large amounts of data in the biological field requires expert knowledge, is extremely time-consuming, and also expensive. Moreover, the task is prone to inter- and intra-user bias. It therefore could lead to poor quality of ground truth images, especially when many images need to be manually labeled. The low availability of large labeled training data might be the reason that automatic analysis of EM images based on deep learning approaches are not yet routinely applied in the field of biomedical EM.

To mitigate this issue, researchers have been studying various techniques for segmenting EM images with a small labeled dataset and still obtaining performance that is on par with human experts. Roels and Saeys (2019) proposed a method for cost-efficient segmentation of electron microscopy images using active learning by smartly selecting the samples that require labeling. Generative adversarial networks were used by Shaga Devan et al. (2021) for the generation of synthetic EM images as a data augmentation method which improved the detection and localization of herpesvirus human cytomegalovirus (HCMV). The authors of EM-Net developed a scalable deep neural network ensemble for rapid learning from ground-truth data for binary image segmentation (Khadangi et al. 2021). The model was trained and tested with two binary datasets. For the ensemble process, multiple models were created with various batch normalizations modifications and combined for final prediction.

Our work was inspired by CDeep3M (Haberl et al. 2018), which is a ready-to-use large-scale image segmentation tool employing cloud-based deep convolutional neural networks for the segmentations of biological structures from both electron and light microscopy modalities. This tool was developed using an ensemble structure integrating models trained with one, three, and five consecutive image frames and utilizes the concept of transfer learning. For performing segmentations, users are provided with a series of pre-trained networks for structures such as mitochondria, synapses, membranes, and vesicles that were originally trained on electron tomography and serial block-face scanning electron microscopy (SBEM) images.

While the availability of a plug-and-play segmentation tool such as CDeep3M is indeed revolutionary and extremely helpful to the biological EM community, it may not be able to segment a large variety of biological structures due to the intra-class variations and inter-class similarities in appearances that is inherent in many viral particles, cells and organelles. EM images appearances also often vary due to image acquisition, noise, and sample preparations artifacts that could lead to segmentation inaccuracies which will then still require pre- and post-processing user intervention.

Therefore, encouraged by this tool, we aim to develop a semantic segmentation model for the multi-class segmentation of biological structures in EM images using user customized limited labeled ground truth training datasets curated by biologists. In this work, we explore the segmentation of biological structures such as cytoplasm,

nucleus, mitochondria, and chromosomes in both TEM and SEM images. Our goal is to provide the biologists with a segmentation model that can be easily adapted to their own EM image datasets as well as biological structures of choice. These enables biologists to have flexibility and control in the usage of deep learning for their customized segmentation needs without the need for large unaffordable labeling efforts. To this end, we have explored the concept of DL ensembles for improving segmentation quality. We investigated the feasibility of using off-the-shelf ImageNet (Russakovsky et al. 2015) pre-trained networks for our ensemble model which differed from the method used by Haberl (2018).

While there is no standard size requirement for DL training datasets, generally, the larger the training dataset, the better is the ability of the CNN to learn. Therefore, CNNs do not perform well when trained with a small amount of training data (Tajbakhsh et al. 2016). To overcome this limitation, our proposed method utilizes a CNN-based weighted averaging ensemble approach that can learn complex discriminant features from a small dataset to segment biological structures in both TEM and SEM images. The amount of training images in all of the datasets used in this work range from 66 to 258. The primary motivation for using a weighted average ensemble approach is that it has better predictive performance compared than a single model since it produces a lower error rate and reduces variance (Mustafa et al. 2020). However, a single model will not capture the entire underlying structure of the data to achieve optimal predictions. Therefore, by combining multiple base-learners, more information can be captured of the data's underlying structure and can significantly improve prediction accuracy (Shahhosseini et al. 2021).

We refer to our proposed model as WAE-Net (weighted average ensemble network). It is built from an ensemble of U-Nets, each of which is trained with a different pre-trained network and, their predictions are combined in a weighted average manner for the multi-class segmentation of EM images. To provide a deeper understanding of the segmentation results given by the ensemble model, we further applied a visual CNN interpretation approach called Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju et al. 2017) to identify critical regions in the images for prediction. We compared our proposed approach against the standard U-Net for quantitative and qualitative performance. Our ensemble method enabled us to obtain better segmentation results than the U-Net. Taken together, we have developed an approach to perform end-to-end multi-class segmentation of EM images using small datasets allowing us to leverage the benefits of current DL developments without needing unaffordable extensive labeling efforts.

Materials and methods

Image dataset

We evaluated our proposed method using seven biological EM image datasets encompassing two types of EM imaging modalities; TEM and SEM for both training and testing. Datasets 1–3 are biological cells prepared by high-pressure freezing, freeze substitution, and plastic embedding. The EM datasets were obtained with two different approaches. The first approach is to mount the embedded cells in a scanning EM additionally equipped with a focused ion beam (FIB-SEM). A small portion of the embedded cell is removed with the focused ion beam and the newly produced surface is imaged with the scanning EM. This process is repeated several hundred times and the images can then be reconstructed to a three-dimensional model (Villinger et al. 2012). We have used this approach to obtain dataset 1. The second approach is to section the plastic embedded cell using an ultra-microtome equipped with a diamond knife and then taking images of each section with a TEM, which was used to obtain datasets 2 and 3 (Villinger et al. 2014). Datasets 1–3 are comprised of two-dimensional images obtained from three-dimensional samples; a biological cell embedded in a plastic that is then sectioned either with a diamond knife or an ion beam. Datasets 4–7 were publicly available two-dimensional image datasets from Morath et al. (2013). All the images used in this work, were processed as two-dimensional images. The details of each dataset are listed in Table 1.

These seven datasets contain diverse images, which differ in biological structures, imaging modality, pixel resolution, contrast, brightness, and noise levels to account for the variability typical for biological EM. All seven datasets, containing whole-slice images, were pixel-wise manually labeled by biologists and serve as the ground truth for network training and testing. These datasets were randomly divided into a training and hold-out test set with a ratio of 8:2. In order to avoid bias, a 5×5-fold cross validation was performed during model training. The hold-out test set was only used for testing. Table 2 shows the details of the distribution of images for training and testing and the number of segmentation classes for all the datasets. In this work, the background, which comprises all structures that are not of interest in the EM images, is also considered a segmentation class.

Experimental methods pipeline

A representative diagram illustrating the pipeline of our proposed weighted average ensemble model (WAE-Net)

Table 1 Summary of details about the datasets used in this work

Dataset	Specimen	Imaging modality	Pixel resolution (nm)	Structures for segmentation
Dataset 1	Human pancreatic carcinoid cell line	FIB-SEM	9	Cytoplasm, nucleus
Dataset 2	BON cell during interphase	Serial section TEM	9	Cytoplasm, chromosomes
Dataset 3	BON cell during mitosis	Serial section TEM	26.3	Cytoplasm, nucleus, mitochondria
Dataset 4	Human T-cell line Jurkat	TEM	6.41	Cytoplasm, nucleus
Dataset 5	Primary human T-cell blood	TEM	2.33	Cytoplasm, nucleus
Dataset 6	Murine B-cell line J558L	TEM	15.26	Cytoplasm, nucleus
Dataset 7	Phytohemagglutinin/IL-2 expanded human T cells	TEM	15.74	Cytoplasm, nucleus

Table 2 Number of segmentation classes, total images, training images and test images for all datasets

Dataset	Classes for segmentation	Total images	Training images	Test images
Dataset 1	3	323	258	65
Dataset 2	3	167	133	34
Dataset 3	4	117	66	17
Dataset 4	3	135	108	27
Dataset 5	3	122	97	25
Dataset 6	3	115	92	23
Dataset 7	3	108	86	22

for the semantic segmentation of biological structures in EM images is shown in Fig. 1. All the tasks in the proposed model pipeline were done using a Tesla T4 GPU with Tensorflow 2.2.0, Keras 2.3.1 and Python 3.7.11.

Base-learner construction and training

We first trained five individual base-learners as ensemble members to construct the weighted average ensemble. The base-learners were created using the U-Net architecture. Figure 2 shows the typical architecture of a U-Net. Its architecture contains an encoder network that is followed by a decoder network (Ronneberger et al. 2015). The bottommost layer at the base of the U-shaped network is the bottleneck section. The encoder is a classification network consisting of alternating convolution layers with rectified linear unit (ReLU) and batch normalizations, followed by maximum pooling-based downsampling operations which increases the number of feature maps per layer. The input images are encoded into feature representations at various levels.

The bottleneck layer provides mediation between the encoder and decoder layer. The decoder semantically projects the discriminative features learned by the encoder onto the pixel space to obtain a dense classification. It consists of upsampling of the feature maps followed by convolution operations. Upsampling is performed to restore the condensed feature maps to the original size of the input image. This is achieved by expanding the feature dimensions.

In the base-learners of our WAE-Net, short and long skip connections were used within the pre-trained encoders as well as between the encoders and decoders. Detailed diagrams indicating the convolutional layers and skip connections of our proposed model are given as Electronic Supplementary Material. For ResNet34, SeResNet34, and EfficientNetB4, short skip connections were used within the encoders to pass information from the initial layers to deeper layers via matrix additions. Specifically, activations and convolutional layers were short skip connected to the layers performing arithmetic operations (Electronic Supplementary Material). These skip connections provide an uninterrupted gradient flow from the first layer to the last layer, thereby mitigating the effects of the vanishing gradient problem, which is a common occurrence in deep architectures such as the encoders mentioned above. InceptionV3 and VGG19 architectures did not use short skip connections within their encoders.

Long skip connections were utilized in ResNet34, InceptionV3, VGG19, SeResNet34, and EfficientNetB4 pre-trained base-learners (Fig. 2). They were used to concatenate the feature maps from the encoder's layer to the same scale feature maps of the decoder. Specifically, the activation layers in the encoder were connected to the corresponding concatenate layers in the decoder (Electronic Supplementary Material). This was done in order to compensate for the loss of spatial information in the encoder during the downsampling process. Taken together, the use of skip training in our WAE-Net helped to stabilize training and convergence. Simply stated, the encoder part encodes the semantics and contextual information of the input images while the decoder part uses this encoded information for the generation of segmentation maps.

To construct the individual base-learners, we replaced the encoder part of the U-Net with a pre-trained CNN network. The decoder part was maintained as depicted in Fig. 2.

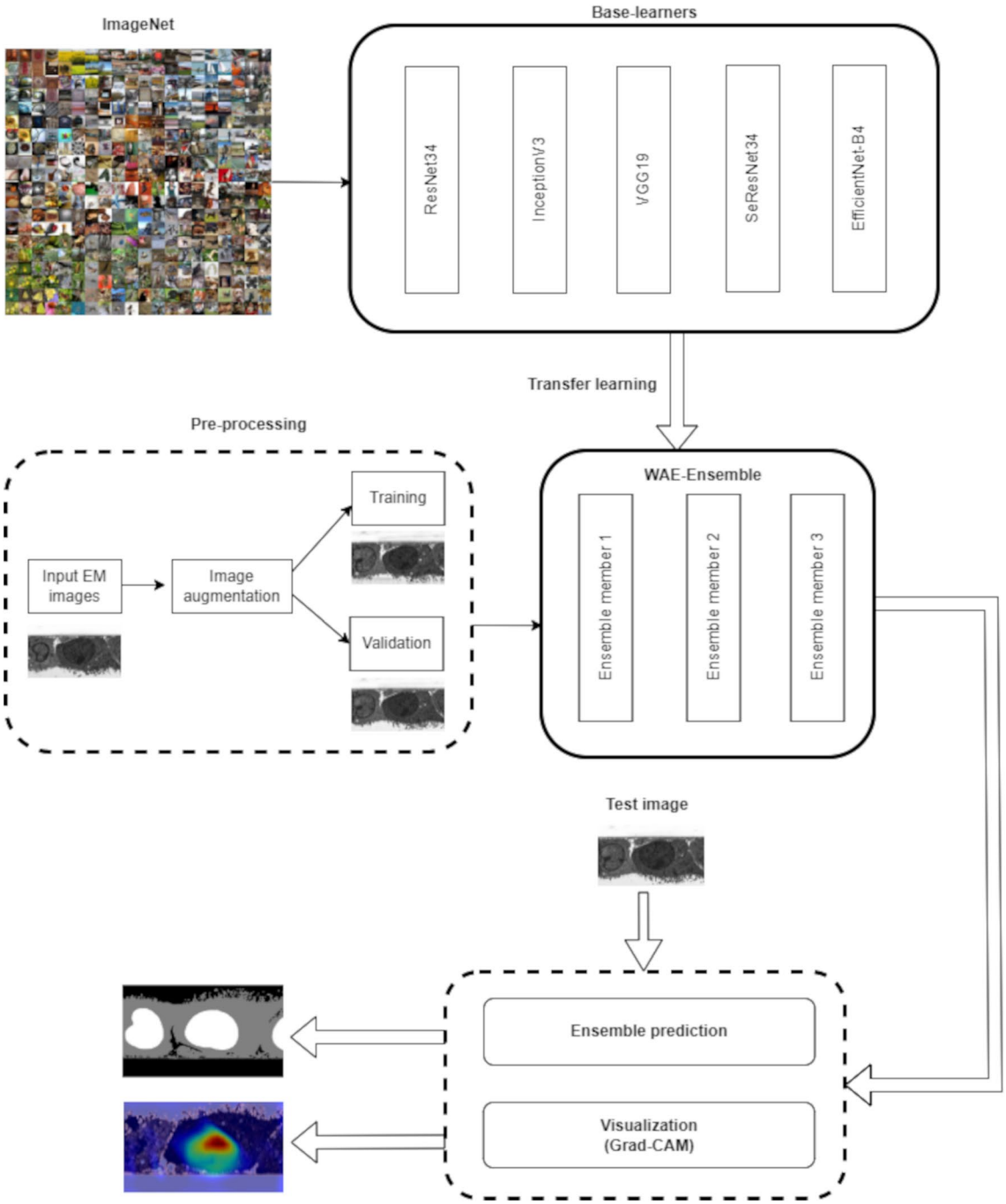


Fig. 1 Schematic architecture of the proposed ensemble-based semantic segmentation of biological electron microscopy images. The individual base-learners were trained with the ImageNet dataset. Then, the top-three best-performing learners were combined in an ensemble. The ensemble model was further trained with the electron microscopy datasets and tested. The Grad-CAM was used to further verify the correctness of the predictions

We built the encoder by removing the fully connected layers of the pre-trained networks and replacing them with a single convolutional layer of 1024 feature channels that serve as the bottleneck part of the base-learner, separating the encoder from the decoder. The output of the transposed convolution layers is then concatenated with the output of the corresponding part of the decoder. The resultant feature maps are treated by convolution operations to keep the number of channels the same to preserve symmetry of the network.

Pre-trained networks were used as encoders in order to leverage transfer learning for our segmentation goals. Devan

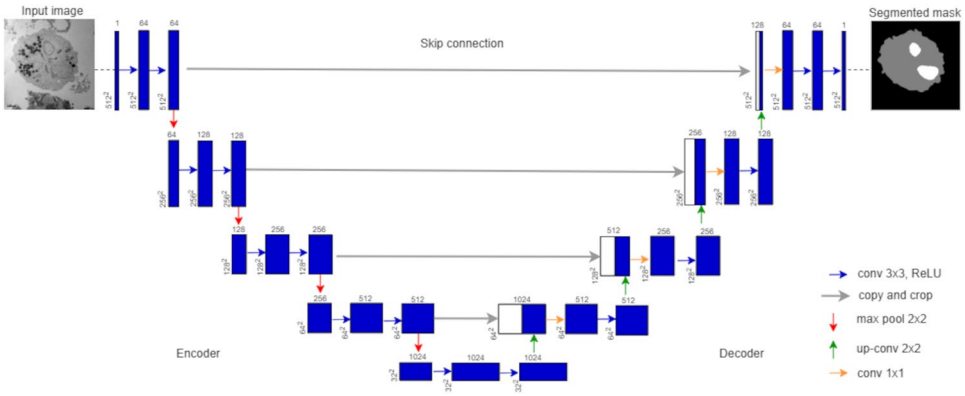


Fig. 2 A standard U-Net architecture. The network consists of an encoder and decoder path. Each blue box corresponds to a multi-channel feature map. The number of channels for each feature map is denoted at the top of the box. The image dimension is provided at the lower-left edge of the box. White boxes represent copied feature maps. The arrows denote the various convolutional neural-network operations (Ronneberger et al. 2015)

et al. (2019) have shown that transfer learning is a highly effective performance booster for EM images when working with a small labeled dataset. It is the process of taking a pre-trained neural network and adapting the neural network to a different dataset by transferring its learned features. Therefore, in this work, the encoder was initialized with pre-trained weights from ImageNet database (Russakovsky et al. 2015) and trained on our own datasets.

The rationale for using pre-trained networks is that the imported network will already have sufficient knowledge in the broader aspects of images such as edge, texture, and shape information that could also be useful for EM image segmentation (Kolesnikov et al. 2020; Dhillon and Haque 2020). While the U-Net can also be trained from scratch on the ImageNet database, however, training the huge dataset, which contains over 14 million images, will require extremely high computational resources, which is a major limitation in many electron microscopy laboratories around the globe. Therefore, using pre-trained networks will be an optimized cost-effective solution. We selected five state-of-the-art pre-trained networks, and for each dataset, we replaced the encoder of the U-Net with each of the five networks, yielding five base-learners.

For an ensemble to outperform any of its members, the base-learners must be accurate and diverse enough to capture the structure of the data effectively. The diversity in the base learners is where the strength of the ensemble lies (Zhou 2009). Furthermore, different pre-trained networks have different properties and learning schemes that contribute towards the performance of the base-learners in the ensemble. Therefore, proper selection of pre-trained networks is imperative to the success of our ensemble model.

While many pre-trained networks are available, we selected these five based on three requirements. First, is that the networks have consistently given state-of-the-art performance. Secondly, the networks must have different architectural designs, and finally, the networks should only have a small number of parameters. While the first requirement is self-explanatory, the second requirement is crucial for creating diversity in our ensemble as different base learners have different properties and learning paradigms. A diverse ensemble can maximize the learning ability of the model while minimizing variance and bias. The final requirement regarding the network's parameters warrants deeper analysis.

Current CNN-based state-of-the-art networks have billions of learning parameters that provide highly competitive results when trained with large datasets. On the other hand, these large networks perform poorly when trained with small datasets. It has been observed that training large complex networks with small datasets often leads to overfitting (Ying 2019). Overfitting happens when noise or random fluctuations in the training data is picked up and learned as features

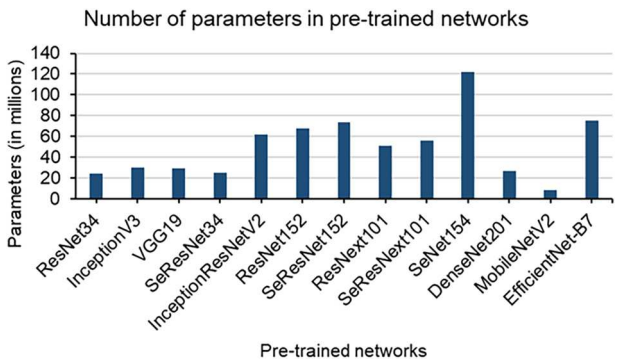


Fig. 3 Graph depicting the total number of learning parameters in publicly available state-of-the-art pre-trained convolutional neural networks. The total number of parameters encompass both trainable and non-trainable learning parameters

by the network to the extent that it negatively impacts its performance and subsequently affects segmentation quality. Figure 3 shows a selection of commonly used state-of-the-art pre-trained networks and their corresponding number of parameters.

Based on Fig. 3, we have selected networks that yield good performance, are diverse in their learning approach, and have a total number of parameters of less than 30 million. A threshold of 30 million parameters was chosen because based on preliminary testing (data not shown), a network with a larger number of parameters overfit the datasets while a network with lower number of parameters did not learn effectively from the training data. Therefore, the pre-trained networks used as encoders in this work are ResNet34 (He et al. 2016), InceptionV3 (Szegedy et al. 2015), VGG19 (Simonyan and Zisserman 2015), SeResNet34 (Hu et al. 2019), and EfficientNet-B4 (Tan and Le 2020). Table 3 shows the details of the selected pre-trained networks. For each dataset, the above-mentioned five pre-trained networks were individually used as encoders in the U-Net and trained. This results in five individual base-learners for each dataset.

Hyperparameter optimization

Hyperparameters are all the parameters that are set in a CNN model before starting the training process in order to configure the model to our dataset. Hyperparameter optimization has a major impact on the performance of the model because it directly influences the training process. Efficient hyperparameter selection can avoid overfitting, improve results, and form a generalized model. Data augmentation was performed on the original image datasets to expand the size of the training dataset (Shorten and Khoshgoftaar 2019). In order to optimize computational resources while increasing dataset size, we have applied five geometric image augmentation methods that preserve the semantic information on the training images using the Albumentations library (Buslaev et al. 2020). The augmentations applied were vertical flip, random rotate, horizontal flip, transpose, and grid distortion. These augmentations were selected on the basis of the most useful transformations for the structures of interest in this work which are the cytoplasm, nucleus, chromosomes, and mitochondria.

Table 3 Number of parameters in the selected pre-trained networks

Model	Trainable parameters	Non-trainable parameters	Total parameters
ResNet34	24,439,094	17,350	24,456,444
InceptionV3	36,416	29,896,979	29,933,395
VGG19	4032	29,058,227	26,062,259
SeResNet34	17,350	24,600,290	24,617,640
EfficientNet-B4	25,735,307	25,608,123	25,735,307

For the training of our WAE-Net model, we used the Adam optimizer with an initial learning rate of 0.0001. We reduced the learning rate by a factor of 4 when the validation loss has stopped decreasing for ten epochs. The mini-batch size was set to 1. Next, weight balancing was performed so that the model is not biased towards a specific segmentation class.

Loss functions play a critical part in effective CNN learning. In order to reduce the validation loss, a combination of focal and dice loss function was used for training (Jadon 2020). Focal loss addresses class imbalance by down-weighting the contribution of easy training examples and therefore enabling the model to focus more on learning hard examples (Wang et al. 2020). This is crucial to our work, as the segmentation classes in this work, such as chromosomes, mitochondria, cytoplasm, and nucleus have unbalanced representation in the image, thereby increasing the risk of the training being dominated by the most prevalent class. The mathematical definition of focal loss is shown in Eq. (1),

$$L_{fl} = -\alpha(1-p)^{\gamma}\log(1-p) \tag{1}$$

where, p is the model's estimated probability, α, γ are two hyperparameters, α is used to adjust the distribution of the easy samples, and $(1\text{-}p)^{\gamma}$ is the dynamic scaling factor, which is used to adjust the distribution of hard samples.

The dice loss was introduced by Milletari et al. (2016) and is derived from the Sørensen–Dice coefficient. It is commonly used for bio-medical image segmentation tasks. Equation (2) shows the mathematical definition of dice coefficient, in which p_i and g_i represent pairs of corresponding pixel values of prediction and ground truth, respectively.

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \tag{2}$$

Weighted average ensemble training

Our weighted average ensemble, WAE-Net, is constructed by combining the predictive base-learners, where the contribution of each base-learner to the final prediction is weighted by the performance of the individual learner. A representative diagram of the proposed WAE-Net is shown in Fig. 4.

We experimented with our ensemble method to segment the EM images into various biological classes such as cytoplasm, nucleus, mitochondria, and chromosomes. For each dataset, its top-three best-performing base-learners were combined together for the ensemble. We avoided

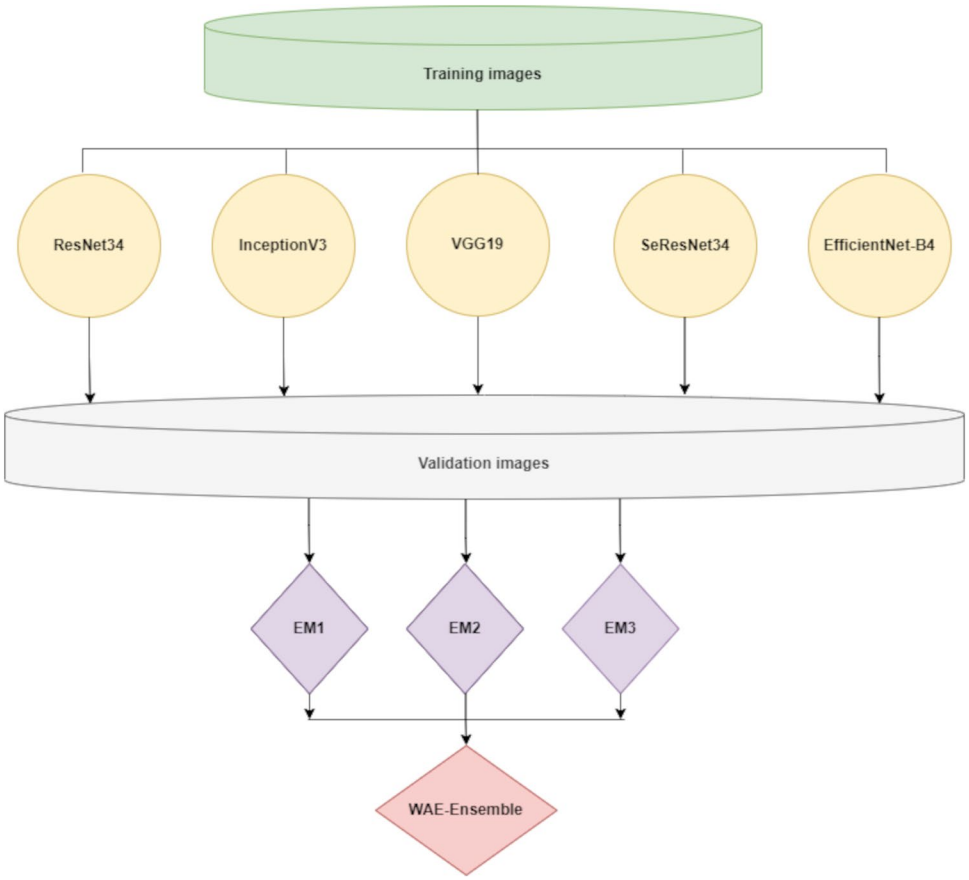


Fig. 4 Graphical representation of the proposed WAE-Net. ResNet34, InceptionV3, VGG19, SeResNet34, and EfficientNet-B4 are the pre-trained networks that serve as the base-learners. EM1, EM2, and EM3 represent the predictions of the top-three best-performing base-learners that make up the ensemble. WAE-Net denotes the final prediction for our proposed ensemble

including all base-learners because not all learners performed well. The inclusion of weak-performing base-learners in the ensemble wastes computational resources and time.

The challenging aspect of using a weighted average ensemble is how to choose the relative weighting for each ensemble member. We used the grid search method to search for the appropriate weights between 0 and 1 for each ensemble member based on the training data, where the learners with better performance receive a higher weight (Liashchynskyi and Liashchynskyi 2019). The best optimized weights are those that result in performance that is better than any contributing individual base-learner and an ensemble that uses equal weights. These weights are then multiplied by the prediction made by the individual base-learners and the weighted average is then used for the final prediction. The prediction of the ensemble was then tested on the test images. We use the original U-Net as a benchmark comparison for evaluating the performance of our WAE-Net. The same hyperparameters, losses, augmentations, and data split with cross-validation as described above were applied for both the U-Net and WAE-Net for the training process.

Explainable segmentation using Grad-CAM

While CNN-based architectures yield impressive performance for image segmentation, their black-box nature makes it challenging to understand why it has given a specific prediction (Lin et al. 2019). Therefore, an interpretative and transparent model will be highly valuable to understand how the CNN makes its decisions, thereby allowing biologists to perform more informative and efficient data analysis.

In our effort to develop such models, we employ the Grad-CAM technique (Selvaraju et al. 2017) to visually show the activation regions of a particular biological class in EM images for model prediction. Grad-CAM exploits the spatial information that is preserved through convolutional layers of the CNN in order to understand which parts of an input image were important for the prediction. Grad-CAM specifically measures the gradients of features maps in the final convolution layer on a CNN model for an image to identify the critical regions that are class-discriminating saliency maps. The class-discriminative saliency map, L^c for the target biological class, c in an image is defined as follows,

$$L_{i,j}^{c} = \mathrm{Re}LU\left(\sum_{k} w_{k}^{c} A_{i,j}^{k}\right) \tag{3}$$

where, $A^{k}_{i,j}$ denotes the activation map for the k-th filter at a spatial location (i,j) and ReLU captures the positive features of the target class. The target class weights of the k-th filter are computed as given by Eq. (4).

$$w_{k}^{c} = \frac{1}{Z}\sum_{i}\sum_{j}\frac{\partial Y^{c}}{\partial A_{i,j}^{k}} \tag{4}$$

The field-of-view (FOV) of a network is the size of the region in the input space that produces the feature. It is a measure of association of an output feature to the input region (Luo et al. 2017). The FOV plays an important role for Grad-CAM class-discriminating saliency maps because it gives an indication of where we are obtaining our results from as data flows through the layers of the network. The advantages of FOV in recognizing visual patterns lie in the fact that the units or neurons within a layer are directly tasked with learning visual features from a small region of the input data. It is therefore imperative to have a convolutional model with a FOV that covers the entire relevant input image region. The Grad-Cam takes an input image and predicts the output class-discriminating saliency map. However, if the network does not have the capacity to consider all the relevant pixels when performing the predictions, the resultant activation map will not be complete. Taking this into account, all the pre-trained networks used in this work have FOVs that cover the entire input image to yield correct information.

In our work, the Grad-CAM visualization is incorporated at the end edge of the proposed ensemble model after the training process, as displayed in Fig. 1. The GradCAM is applied to the convolutional layers at the U-Net bottleneck, which is at the end of the encoder before upsampling process. The class discriminatory saliency map for a specific biological class is then calculated and used to create an activation heat-map. The heat-map is then superimposed onto the given input image. The final Grad-CAM image enables us to verify that the proposed model obtained information of the biological structures from the correct regions in the images. It also enables us to identify the various regions in the EM image that are important for the prediction of each of the base-learners in the WAE-ensemble.

Evaluation metrics

A successful segmentation by a CNN model is one that maximizes the overlap between the predicted and ground truth regions in an image. In order to evaluate the performance of the proposed method, we considered two commonly used evaluation metrics for semantic segmentation tasks, which are the Jaccard index and the F1 score. The Jaccard index (JI), which is also referred to as the Intersection over Union (IoU) metric, (Ronneberger et al. 2015; Cetina et al. 2018) quantifies the percentage of overlap between the ground truth mask and predicted output mask. The Jaccard index is defined by Eq. (5) below,

$$Jaccard\ index = \frac{GT \cap PO}{GT \cup PO} \tag{5}$$

where, GT and PO denote the pixels in the ground truth mask and predicted output respectively.

The F1 score (Gadosey et al. 2020) conveys the balance between the precision and the recall of the model and is defined by Eq. (6).

$$F1 - score = \frac{2|GT \cap PO|}{|GT| + |PO|} \tag{6}$$

Both of these metrics range from 0 to 1, with 0 signifying no overlap and 1 signifying a perfectly overlapping segmentation.

Results

Quantitative results

For each of the datasets, the five base-learners were trained and the top-three best-performing learners were then selected and combined together in order to construct the weighted average ensemble. The best-performing learners and their corresponding ensemble weighting for each dataset are shown in Table 4, whereby the sum of the weights for each dataset is equal to 1. Base-learners with zero weighting were not included in the ensemble. Based on our experiments (data not shown), grid search on different subsets of

Table 4 Top-three best-performing base-learners for each dataset and their corresponding ensemble weighting obtained from Grid-CV search

Dataset	Top-three best-performing base-learners	Weighting
Dataset 1	ResNet34, InceptionV3, VGG19	0.34, 0.33, 0.33
Dataset 2	ResNet34, InceptionV3, EfficientNet-B4	0.50, 0.50, 0.00
Dataset 3	ResNet34, SeResNet34, EfficientNet-B4	1.00, 0.00, 0.00
Dataset 4	ResNet34, VGG19, EfficientNet-B4	0.50, 0.00, 0.50
Dataset 5	ResNet34, InceptionV3, EfficientNet-B4	0.40, 0.40, 0.20
Dataset 6	ResNet34, InceptionV3, EfficientNet-B4	0.40, 0.20, 0.40
Dataset 7	ResNet34, InceptionV3, VGG19	0.40, 0.20, 0.40

Table 5 Segmentation performance comparison between WAE-Net and U-Net

	U-Net		WAE-Net	
	Jaccard index	F1 score	Jaccard index	F1 score
Dataset 1	0.8552	0.8980	**0.9865**	**0.9923**
Dataset 2	0.7645	0.6942	**0.8448**	**0.8842**
Dataset 3	0.5571	0.5381	**0.7243**	**0.6915**
Dataset 4	0.7686	0.7866	**0.9692**	**0.9685**
Dataset 5	0.8753	0.7994	**0.9725**	**0.9784**
Dataset 6	0.8456	0.7924	**0.9644**	**0.9817**
Dataset 7	0.7099	0.8676	**0.8992**	**0.9452**

The Jaccard index and F1 score were averaged over all test images in each dataset

Table 6 Segmentation performance comparison between WAE-Net and U-Net for each segmentation class

Dataset	Segmentation class	U-Net	WAE-Net
Dataset 1	Background	0.9296	**0.9567**
	Cytoplasm	0.8663	**0.9882**
	Nucleus	0.7742	**0.9918**
Dataset 2	Background	0.9157	**0.9321**
	Cytoplasm	0.8936	**0.9143**
	Chromosomes	0.4867	**0.6826**
Dataset 3	Background	0.9547	**0.9762**
	Cytoplasm	0.6858	**0.8416**
	Nucleus	0.5783	**0.8052**
	Mitochondria	0.0000	**0.2143**
Dataset 4	Background	0.9344	**0.9897**
	Cytoplasm	0.6838	**0.9464**
	Nucleus	0.6728	**0.9676**
Dataset 5	Background	0.9646	**0.9912**
	Cytoplasm	0.8166	**0.9452**
	Nucleus	0.8372	**0.9645**
Dataset 6	Background	0.9769	**0.9941**
	Cytoplasm	0.8154	**0.9567**
	Nucleus	0.7461	**0.9569**
Dataset 7	Background	0.9562	**0.9765**
	Cytoplasm	0.5838	**0.8191**
	Nucleus	0.5712	**0.8994**

The performances are evaluated in terms of Jaccard index

the training data based on the 5×5-fold cross validation lead to the same weighting for the top-three performing models for each dataset.

ResNet34 has emerged as the best-performing base-learner for all the seven EM datasets. This was followed by InceptionV3 and EfficientNet-B4, respectively. SeResNet34 was the most poorly performing base-learner, as it only emerged once as a top-three best performer. In terms of ensemble weighting, the combination of two or more base-learners resulted in the best performance for most of the datasets. However, dataset 3 did not benefit from an ensemble model as only the contribution of ResNet34 was useful for its final prediction (Table 4).

After the construction of the WAE-Net using the best-performing base-learners, we evaluated its segmentation performance and compared it with the U-Net (Table 5). The results were quantitatively evaluated by using the Jaccard index and F1 score metric. WAE-Net yielded a mean Jaccard index (JI) of 0.9087 and mean F1 score of 0.9203 for all the seven datasets while U-Net only yielded a mean JI of 0.7709 and mean F1 score of 0.7680.

In order to further understand the segmentation results obtained by the proposed model, we evaluated the performance of both WAE-Net and U-Net on the individual segmentation classes (Table 6). Both U-Net and WAE-Net were able to segment the background class similar to ground truth data with a mean JI of above 0.90 for all datasets. Our model performed very well in segmenting the cytoplasm with a mean JI of 0.9160 while U-Net's performance deteriorated with a mean JI value of 0.7636. When segmenting the nucleus, our ensemble model yielded good segmentation results with a mean JI of 0.9309 while the U-Net had only managed to achieve a mean JI of 0.6966. The chromosomes were able to be segmented by WAE-Net with a JI of 0.6826 compared to U-Net which performed poorly with a JI of only 0.4867. Both models performed very badly for the segmentation of mitochondria in dataset 3. The U-Net was not able to segment any mitochondria at all, while our model was able to segment only some of the mitochondria pixels with a JI of 0.2143. The reason for the low segmentation rates for both chromosomes and mitochondria might be because both of these structures are quite small compared to the other structures in the EM images and therefore our model did not have sufficient pixel coverage in the training images for learning. Furthermore, in dataset 3, there were very few training images but a higher number of segmentation classes which explains the low segmentation rate for chromosomes. Generally, WAE-Net was able to consistently obtain superior performance compared to the U-Net for all datasets as well as for all individual segmentation classes (Tables 5 and 6, Figs. 5, 6, 7 and S1). Dataset 1 had the best performance while the worst was from dataset 3.

We also performed a small experiment to investigate the performance of smaller base-learner U-Nets compared to their relatively larger counterparts that were used in our WAE-Net. The size was defined by the number of learning parameters and convolutional layers and the performance was evaluated using Jaccard index (JI).

We compared the performance of base-learners using ResNet18 and ResNet34, VGG16 with VGG19, and

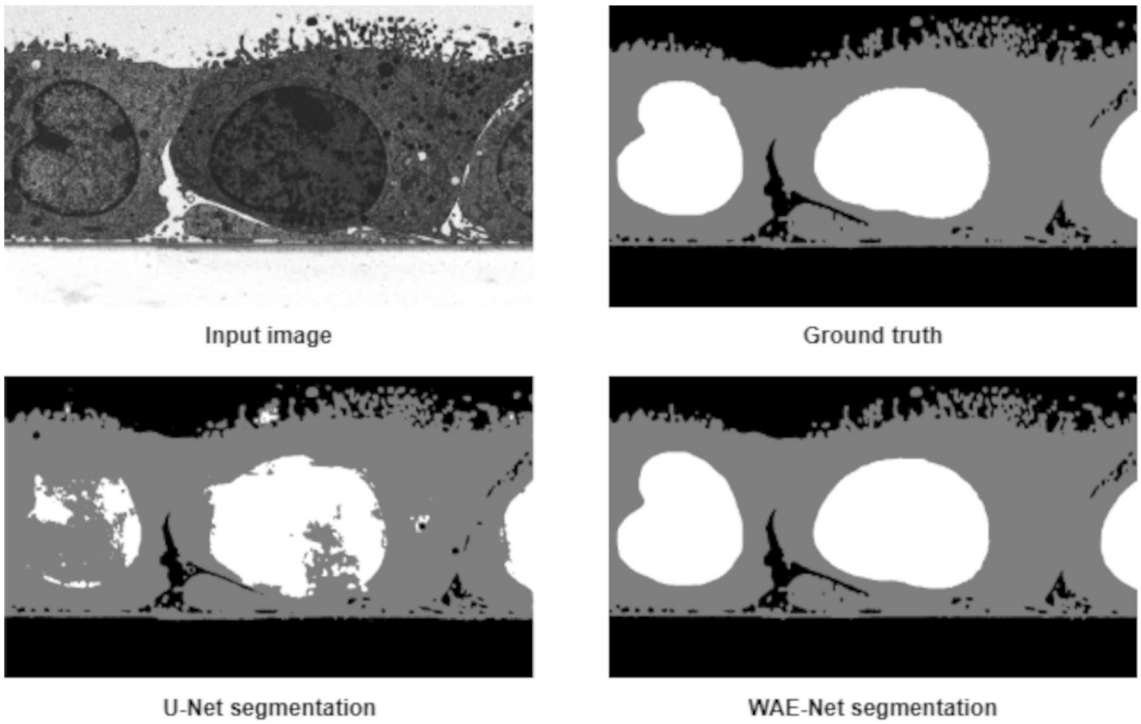


Fig. 5 Quantitative evaluation of the segmentation results produced by WAE-Net and U-Net over a single image in dataset 1. The ground truth and predicted output images contains three segmented structures which are the background (black), cytoplasm (grey), and nucleus (white)

EfficientNetB0 with EfficientNetB4. Both ResNet and VGG were trained on a subset of data from dataset 1 while EfficientNet was trained on a subset of data from dataset 4. We observed that ResNet18, which has 18 convolutional layers and 14,340,860 parameters, gave a Jaccard index (JI) of 0.8045 while ResNet34 with 34 layers and 24,456,444 parameters gave a JI of 0.8288. VGG16 that has 16 layers and 23,752,563 parameters yielded a JI of 0.8147 while VGG19 with 19 layers and 29,062,259 parameters yielded a JI of 0.8197. We compared EfficientNetB0, which has 10,115,791 parameters with EfficientNetB4, which has 25,735,307 parameters and observed that the former had a JI of 0.7622 while the latter had a JI of 0.8206. This indicates that the smaller networks performed almost as well as their relatively larger counterpart for the structures of interest in this work, with the larger counterparts giving only slightly improved performances. Surprisingly, VGG16 and VGG19 gave almost similar performance.

Qualitative results

Figures 5, 6, 7, and S1 show representative test images after segmentation, comparing input image, ground truth, WAE-Net prediction, and U-Net prediction. An expert biologist verified that the WAE-Net was able to segment the biological structures in all test EM images more accurately compared to U-Net. Both the models were able to segment the background very well, however, the U-Net was not able to segment the boundary of both cytoplasm and nucleus accurately and was only able to detect the chromosomes partially. In comparison, our ensemble model was able to segment the cytoplasm and nucleus in a highly accurate manner and was able to detect most of the chromosomes. We observed that our model detected some of the mitochondria present in dataset 2 (Fig. 6) and classified them as chromosomes. This could be attributed to the fact that the chromosomes and mitochondria shared certain similarities in terms of appearance and there were very small amounts of mitochondria present in dataset 2 for our model to learn to differentiate between these two structures.

The worst performance was obtained from dataset 3 (Fig. 7), where both models were unsuccessful in segmenting the mitochondria. Our model was not able to segment the mitochondria in the images in a manner similar to the ground truth. Most of the mitochondria were not detected. However, the ensemble model still exhibited better performance, as the U-Net was not able to detect any mitochondria at all and was incorrect in segmenting the cytoplasm and nucleus. Overall, based on the segmented images obtained, we are able to visually verify that our model was able to give segmentation results that are nearer to the ground truth compared to the U-Net.

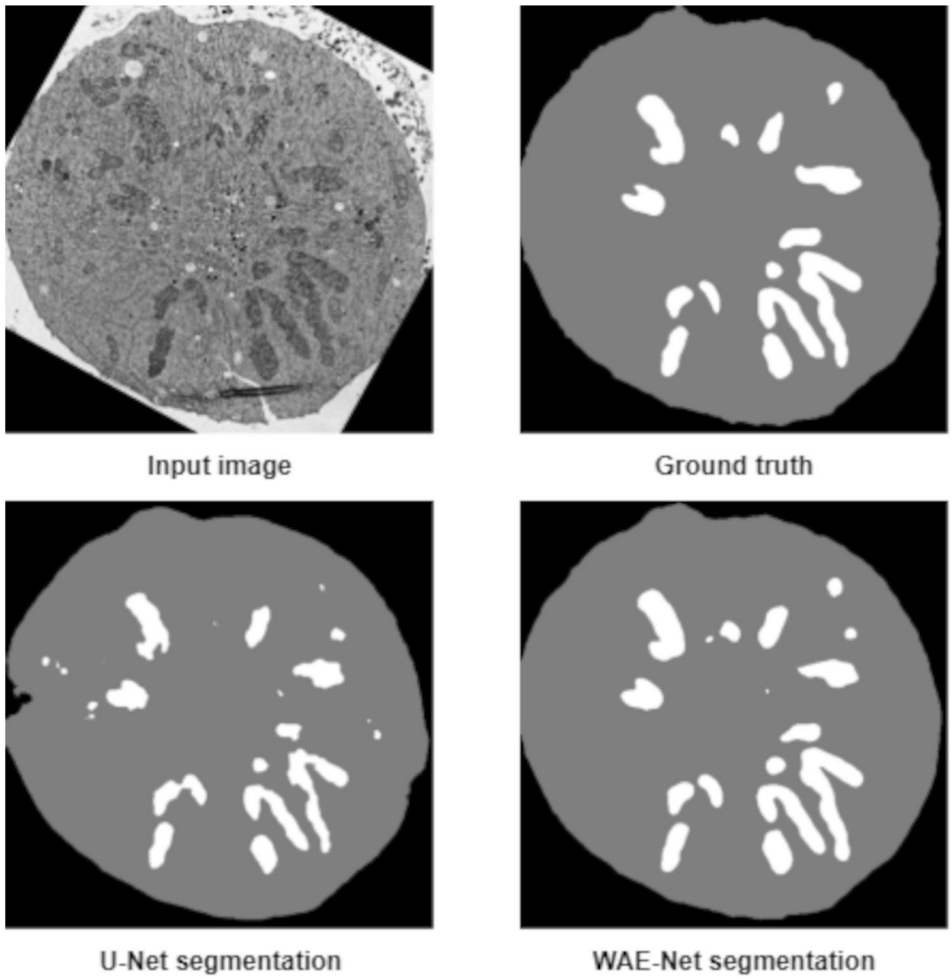


Fig. 6 Quantitative evaluation of the segmentation results produced by WAE-Net and U-Net over a single image in dataset 2. The ground truth and predicted output images contains three segmented structures which are the background (black), cytoplasm (grey), and chromosomes (white)

Visualization with Grad-CAM

We applied the Grad-CAM method to visually depict the pertinent areas in the test images where WAE-Net emphasizes the segmentation decision for a given biological class. We found that the convolutional layers of the U-Net bottleneck, which is at the end of the encoder before upsampling, are more informative than layers close to the end of the U-Net decoder. Figure 8 and S2 show representative images of the Grad-CAM visualization using the top-three base-learners for each segmentation class. The Grad-CAM visualization locates the relevant areas in the EM image that is important for the segmentation of a specific biological structure for each of the base-learners. The red regions highlight the most important discriminative regions and the blue regions the least important as depicted by the color bar in Fig. 8.

Importantly, we were able to learn the behavior of the individual base-learners that compose the WAE-Net. It was observed that not all base-learners in the WAE-Net were triggered by the same region for prediction. Different base-learners look at different areas in an image for the segmentation of the same biological structure. Figure 8 shows that both ResNet34 and InceptionV3 were triggered by the correct regions to identify the background of the EM image. However, VGG19 was triggered by regions that were outside of the background area, specifically the nucleus for background segmentation. For the cytoplasm, all the three base-learners were triggered by the correct region. GradCAM visualizations indicate that different areas within the cytoplasm was important for different base-learners. In example, the cytoplasm area that was very relevant for InceptionV3 prediction was however less relevant for ResNet34 and VGG19. Both InceptionV3 and VGG19 were strongly triggered by the main nucleus in the image for nucleus segmentation. In contrary, ResNet34 was not triggered by the main nucleus, but rather by a small nucleus region at the right of the image. Further visualizations in S2 provide insight into the behavior of the individual base-learners. We observed in both Fig. 8 and S2 that areas that were not part of the object of interest were also triggered as relevant by the model for prediction.

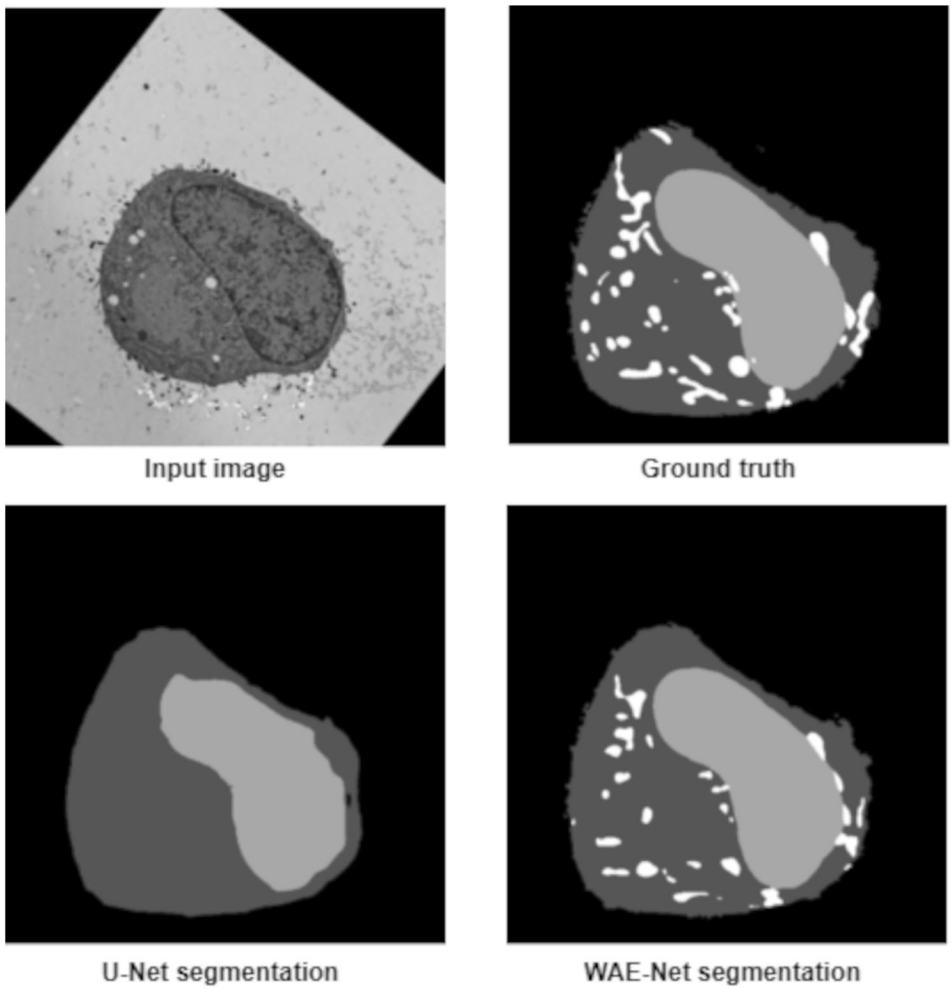


Fig. 7 Quantitative evaluation of the segmentation results produced by WAE-Net and U-Net over a single image in dataset 3. The ground truth and predicted output images contains four segmented structures which are the background (black), cytoplasm (dark grey), nucleus (light grey) and mitochondria (white)

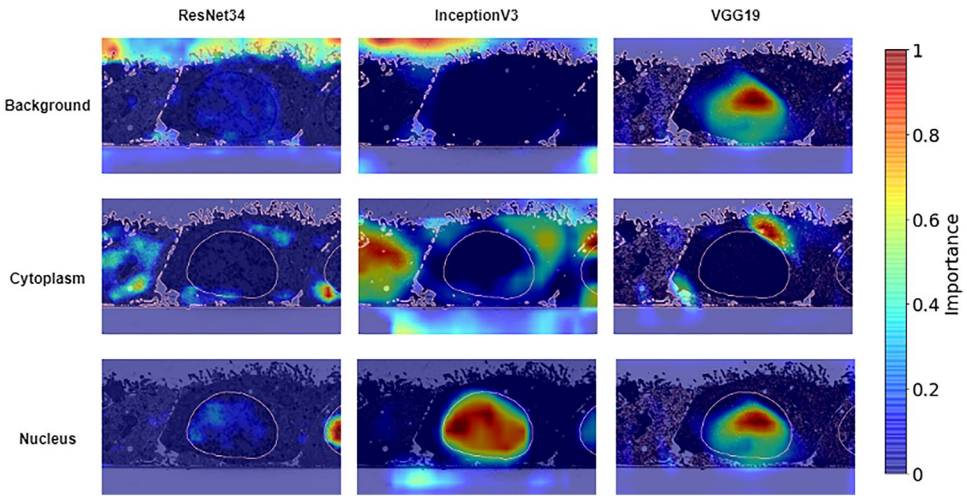


Fig. 8 Grad-CAM visualization on the input image of Fig. 5 using the WAE-Net. These visualizations were obtained from the bottleneck layer of the base-learners of our network. The columns indicate the respective base-learners while the rows indicate the target segmentation class localized by Grad-CAM. The red regions highlight the most important discriminative regions, while the blue regions the least important. The pink outline indicates that the region of interest is contained within this area

Discussion

The key to new insights and discoveries about cell structures and organelles goes hand in hand with new technological developments that enable acquiring and analyzing relevant, high-quality information from EM images. However, the implementation of state-of-the-art DL approaches for biological analysis is often hindered by the requirement of large labeled training datasets, which are usually scarce in specialized fields such as biological EM. In order to provide a novel contribution towards eliminating this limitation, we have presented a robust approach for the fully automatic semantic segmentation of biological structures in EM images using limited training dataset. This study demonstrates that a weighted average ensemble of CNN models can significantly improve the segmentation rate compared to a single model trained with a small labeled dataset. We show the potential of this approach to segment both TEM and SEM.

We first constructed the individual base-learners using a transfer learning approach. This approach increased the segmentation ability of the learners significantly even though our images differ greatly from the ImageNet database. The base-learners were able to extract feature representations from ImageNet and successfully applied them to the EM images. Then, we combined the best-performing top-three learners for each dataset in a weighted average manner for the final segmentation.

We observed that the best base-learners differs for each dataset. This indicates that no one ultimate combination of base-learners gave the best results for all the datasets. This is because the datasets differ from one another in terms of image properties as well as biological appearances. Therefore, one base-learner might be suited to learn the semantic information of specific biological structure while another is better at a different one. Thereby, combining these base-learners in an ensemble model enables us to harness this learning power.

We have demonstrated that an ensemble model is imperative for a good segmentation. Most of the datasets gave the best results with a combination of two or more base-learners. Datasets 1, 2, and 4 had equal weighting among its ensemble members while, the rest of the datasets had unequal weighting (Table 4). This proves that not all base-learners contribute equally towards the performance of an ensemble model and weighting their contribution has a powerful effect on the final prediction. Only dataset 3 did not benefit from the ensemble process as only one base-learner significantly contributed to the final prediction. This is because this dataset contained a higher number of segmentation classes compared to the other datasets but was trained with comparatively low amount of training images (Table 2). Therefore, most of the base-learners were not able learn sufficient discriminative features for segmentation, rendering the ensemble model inefficient for this dataset.

The WAE-Net's results were compared with the original U-Net from quantitative and qualitative perspectives. Our ensemble model outperformed the U-Net for all the datasets significantly (Tables 5 and 6 and Figs. 5, 6, 7 and S1). This suggests that an ensemble of models can outperform one single model when trained with only a small number of ground truth images. In addition, we investigated the performance of both the models on the individual segmentation classes (Table 6). Large differences between the two models were observed in their abilities to detect cytoplasm, nucleus, chromosomes, and mitochondria. Overall, the WAE-Net was able to segment the EM images in a manner similar to the ground truth for most of the datasets despite the low availability of training images. Our proposed model was able to generalize well on the hold-out test dataset (Figs. 5, 6, 7, and S1). When we look at the training time of the WAE-Net, it takes approximately between 4 and 8 h on a single GPU, depending on the pre-trained network parameters. Because it takes several months for a biologist to segment a large dataset manually, the total cost of training time is far less.

From the small study conducted to observe the performance of smaller pre-trained U-Net base-learners compared to their relatively larger counterparts, we learned that smaller networks than those utilized in this work are able to give good predictive performance. However, further study needs to be conducted with a larger variety of biological structures and electron microscopy modalities to understand the feasibility of small pre-trained U-Nets for semantic segmentation.

The Grad-CAM has demonstrated to be a powerful tool to visually understand how the ensemble model achieved the segmentation prediction obtained in this work (Figs. 8 and S2). Its inclusion enabled us to understand the behavior of the individual base-learners for the various segmentation classes. We observed that different regions were found relevant by different base-learners for the segmentation of the same structure. Since the WAE-Net model is a combination of learners, strongly relevant regions found by all learners were combined for the final prediction, thereby improving the model's overall performance.

While Grad-CAM visualizations are class-discriminative and are able to localize relevant image regions well, they lack the ability to show fine-grained pixel-wise details. Despite this, we believe that the Grad-CAM based visualizations is a first step towards the incorporation of transparency and interpretability in deep learning for biological EM applications. As deep learning is a black-box approach in the computer science domain, the inclusion of these visualizations may increase biologists trust in the model's prediction. It may also be useful for future applications where model selection is crucial.

Concludingly, this work explored the possibility of semantically segmenting EM images using a very small training dataset. Our model has shown the ability to segment four types of biological structures in both TEM and SEM images. More importantly, the WAE-Net segmentation predictions provide a close estimate of the ground truth data. The standard U-Net might be able to reach and even surpass the performance of WAE-Net given the availability of large amount of labeled ground truth data. However, since the availability of large amounts of labeled data is often scarce in the biological EM field, we believe that our approach can lead to the rapid development of deep learning applications in this area.

Since our method reduces the labeling burden on biologists, it could be further adopted for the segmentation of various other structures in the field of biological image analysis, as well as for other types of microscopy modalities. Apart from that, our approach can also be used with many other pre-trained networks not explored in this work. Therefore, WAE-Net can be easily customized to suit the individual segmentation needs of the biologists. As there is still room for improvement in terms of segmentation performance for very small datasets, further pursuit in the direction of this study would be worthwhile. Finally, we believe that the integration of this tool into the morphological analysis of images could help biologists to have a deeper understanding of disease mechanisms as well as help expand the impact of artificial intelligence in the biological domain.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00418-022-02148-3.

Acknowledgements We would like to thank Renate Kunz and Reinhard Weih of the Central Facility for Electron Microscopy for their technical assistance. Timo Ropinski, Tim Bergner, Pedro Hermosilla Casajus and Hannah Kniesel are gratefully acknowledged for their support and expertise. We thank Gerhard Wanner for providing some of the images used in this paper. This work was funded by the Baden-Württemberg Stiftung gGmbH through the ABEM project.

Author contribution SDK and WP: conceived the idea presented in this work. SDK: designed the algorithm, data analysis and performed all the experiments. KHA: contributed towards the validation of the proposed methods. WP, and RC: provided the EM images for Dataset 1–3. RC: performed the manual labeling for the ground truth images of Dataset 1–3. WP and KHA: provided supervision and guidance in developing this research work. SDK: wrote the manuscript with support from WP: All authors reviewed the results and approved the final version of the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. Project funding by Baden-Württemberg Stiftung gGmbH through ABEM.

Data availability The source codes, pre-trained weights and images used in this work can be accessed at https://data.mendeley.com/datasets/9rdmnn2x4x/1.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Belevich I, Joensuu M, Kumar D et al (2016) Microscopy image browser: a platform for segmentation and analysis of multidimensional datasets. PLOS Biol 14:e1002340. https://doi.org/10.1371/journal.pbio.1002340

Buslaev A, Parinov A, Khvedchenya E et al (2020) Albumentations: fast and flexible image augmentations. Information 11:125. https://doi.org/10.3390/info11020125

Casser V, Kang K, Pfister H, Haehn D (2020) Fast Mitochondria Detection for Connectomics. InMedical Imaging with Deep Learning. 111-120. PMLR

Cetina K, Buenaposada JM, Baumela L (2018) Multi-class segmentation of neuronal structures in electron microscopy images. BMC Bioinformatics 19:298. https://doi.org/10.1186/s12859-018-2305-0

Ching T, Himmelstein DS, Beaulieu-Jones BK et al (2018) Opportunities and obstacles for deep learning in biology and medicine. J R Soc Interface 15:20170387. https://doi.org/10.1098/rsif.2017.0387

Devan KS, Walther P, von Einem J et al (2019) Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. Histochem Cell Biol 151:101–114. https://doi.org/10.1007/s00418-018-1759-5

Dhillon H, Haque A (2020) Towards network traffic monitoring using deep transfer learning. IEEE 19th Int Conf Trust Secur Priv Comput Commun Trust. https://doi.org/10.1109/TrustCom50675.2020.00144

Fischer CA, Besora-Casals L, Rolland SG et al (2020) MitoSegNet easy-to-use deep learning segmentation for analyzing mitochondrial morphology. iScience. https://doi.org/10.1016/j.isci.2020.101601

Gadosey PK, Li Y, Agyekum EA et al (2020) SD-UNet: stripping down U-Net for segmentation of biomedical images on platforms with low computational budgets. Diagnostics 10:110. https://doi.org/10.3390/diagnostics10020110

George B, Assaiya A, Roy RJ et al (2021) CASSPER is a semantic segmentation-based particle picking algorithm for single-particle cryo-electron microscopy. Commun Biol 4:1–12. https://doi.org/10.1038/s42003-021-01721-1

Haberl MG, Churas C, Tindall L et al (2018) CDeep3M-Plug-and-Play cloud-based deep learning for image segmentation. Nat Methods 15:677–680. https://doi.org/10.1038/s41592-018-0106-z

He K, Zhang X, Ren S, Sun J (2016) Deep Residual Learning for Image Recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp 770–778

Heinrich L, Bennett D, Ackerman D et al (2021) Whole-cell organelle segmentation in volume electron microscopy. Nature 599:141–146. https://doi.org/10.1038/s41586-021-03977-3

Horwath JP, Zakharov DN, Mégret R, Stach EA (2020) Understanding important features of deep learning models for segmentation of high-resolution transmission electron microscopy images. Npj Comput Mater 6:1–9. https://doi.org/10.1038/s41524-020-00363-x

Hu J, Shen L, Sun G (2018) Squeeze-and-Excitation Networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7132–7141. https://doi.org/10.1109/CVPR.2018.00745

Jadon S (2020) A survey of loss functions for semantic segmentation. IEEE Conf Comput Intell Bioinforma Comput Biol CIBCB 2020:1–7. https://doi.org/10.1109/CIBCB48159.2020.9277638

Khadangi A, Boudier T, Rajagopal V (2021) EM-stellar: benchmarking deep learning for electron microscopy image segmentation. Bioinformatics 37:97–106. https://doi.org/10.1093/bioinformatics/btaa1094

Kharabag (2021) Semantic segmentation of HeLa cells: An objective comparison between one traditional algorithm and four deep-learning architectures. https://journals.plos.org/plosone/article?id=https://doi.org/10.1371/journal.pone.0230605

Khobragade N, Agarwal C (2018) Multi-Class Segmentation of Neuronal Electron Microscopy Images Using Deep Learning. InMedical Imaging 10574:105742W. https://doi.org/10.1117/12.2293940

Kolesnikov A, Beyer L, Zhai X et al (2020) Big Transfer (BiT): General Visual Representation Learning. Springer, Cham

Kubota Y, Sohn J, Kawaguchi Y (2018) Large volume electron microscopy and neural microcircuit analysis. Front Neural Circuits 12:98. https://doi.org/10.3389/fncir.2018.00098

Liashchynskyi P, Liashchynskyi P (2019) Grid Search. A Big Comparison for NAS, Random Search, Genetic Algorithm

Lin ZQ, Shafiee MJ, Bochkarev S, et al (2019) Do Explanations Reflect Decisions? A Machine-centric Strategy to Quantify the Performance of Explainability Algorithms. ArXiv191007387 Cs

Luo W, Li Y, Urtasun R, Zemel R (2016) Understanding the effective receptive field in deep convolutional neural networks. In: 29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona. ArXiv170104128 Cs

Mahony NO, Campbell S, Carvalho A et al (2020) Deep learning vs. traditional computer vision. InConference Proceedings CVC. https://doi.org/10.1007/978-3-030-17795-9

Maniates-Selvin JT, Hildebrand DGC, Graham BJ et al (2020) Reconstruction of motor control circuits in adult Drosophila using automated transmission electron microscopy. Cell 9:4390

Milletari F, Navab N, Ahmadi S-A (2016) V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In: Fourth International Conference on 3D Vision (3DV), 565–571. https://doi.org/10.1109/3DV.2016.79

Monchot P, Coquelin L, Guerroudj K et al (2021) Deep learning based instance segmentation of titanium dioxide particles in the form of agglomerates in scanning electron microscopy. Nanomater Basel Switz 11:968. https://doi.org/10.3390/nano11040968

Morath V (2013) Semi-automatic determination of cell surface areas used in systems biology. Front Biosci E5:533–545. https://doi.org/10.2741/E635

Mustafa B, Riquelme C, Puigcerver J, et al (2020) Deep Ensembles for Low-Data Transfer Learning. ArXiv201006866 Cs Stat

Read C, Walther P, von Einem J (2021) Quantitative electron microscopy to study HCMV morphogenesis. Methods Mol Biol Clifton NJ 2244:265–289. https://doi.org/10.1007/978-1-0716-1111-1_14

Roels J, Saeys Y (2019) Cost-efficient segmentation of electron microscopy images using active learning. ArXiv191105548 Cs

Ronneberger O, Fischer P, Brox T (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv150504597 Cs

Russakovsky O, Deng J, Su H, et al (2015) ImageNet Large Scale Visual Recognition Challenge. ArXiv14090575 Cs

Selvaraju RR, Cogswell M, Das A, et al (2017) Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. pp 618–626

Shaga Devan K, Walther P, von Einem J et al (2021) Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network. Cell Microbiol 23:e13280. https://doi.org/10.1111/cmi.13280

Shahhosseini M, Hu G, Pham H (2021) Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. Mach Learn Appl. 7:100251

Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. J Big Data 6:60. https://doi.org/10.1186/s40537-019-0197-0

Simonyan K, Zisserman A (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv14091556 Cs

Sommer C, Straehle C, Köthe U, Hamprecht FA (2011) ilastik: Interactive learning and segmentation toolkit. In: 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro. pp 230–233

Szegedy C, Liu W, Jia Y, et al (2015) Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp 1–9

Tajbakhsh N, Shin JY, Gurudu SR et al (2016) Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans Med Imaging 35:1299–1312. https://doi.org/10.1109/TMI.2016.2535302

Tan M, Le QV (2020) EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ArXiv190511946 Cs Stat

Tang B, Pan Z, Yin K, Khateeb A (2019) Recent advances of deep learning in bioinformatics and computational biology. Front Genet 10:214. https://doi.org/10.3389/fgene.2019.00214

Tian L, Hunt B, Bell MAL et al (2021) Deep learning in biomedical optics. Lasers Surg Med 53:748–775. https://doi.org/10.1002/lsm.23414

Villinger C, Gregorius H, Kranz C et al (2012) FIB/SEM tomography with TEM-like resolution for 3D imaging of high-pressure frozen cells. Histochem Cell Biol 138:549–556. https://doi.org/10.1007/s00418-012-1020-6

Villinger C, Schauflinger M, Gregorius H et al (2014) Three-dimensional imaging of adherent cells using FIB/SEM and STEM. Methods Mol Biol Clifton NJ 1117:617–638. https://doi.org/10.1007/978-1-62703-776-1_27

Wang Z, Chen J, Hoi SCH (2020) Deep Learning for Image Super-resolution: A Survey. ArXiv190206068 Cs

Webb S (2018) Deep learning for biology. Nature 554:555–557. https://doi.org/10.1038/d41586-018-02174-z

Ying X (2019) An overview of overfitting and its solutions. J Phys Conf Ser 1168:022022. https://doi.org/10.1088/1742-6596/1168/2/022022

Zheng Z, Lauritzen JS, Perlman E et al (2018) A complete electron microscopy volume of the brain of adult drosophila melanogaster. Cell 174:730-743.e22. https://doi.org/10.1016/j.cell.2018.06.019

Zhou Z-H (2009) Ensemble Learning. In: Li SZ, Jain A (eds) Encyclopedia of Biometrics. Springer, US, Boston, MA, pp 270–273

8. Conclusion and Future Outlook

Novel technologies, significant instrumentation improvements, and advances in specimen preparation methods have enabled rapid developments in EM field (Assaiya et al. 2021; Flannigan et al. 2010; Fernandez-Leiro, Scheres 2016). This has given rise to the production of considerably larger volumes of electron microscopy data in recent years (Yury et al. 2019; Yilai et al 2020). However, analysis and quantification of these data are still being predominantly performed in a manual manner which creates a massive workload for scientist and practitioners. Recently, DL-based techniques have demonstrated success in the automated analysis of EM images in a wide variety of computer vision tasks such as image classification, detection, segmentation, generation, and restoration (Ede 2021). In this work, we focused on the development of DL-based methods for the objective analysis and quantification of biological structures in EM images. DL methods generally require a large amount of labelled ground truth data for efficient model learning, which in turn requires expensive and time-consuming labeling efforts. To alleviate this issue, in this thesis, we proposed DL techniques that enable efficient model learning with just small labelled EM training datasets. A brief conclusion of the work undertaken in this thesis, as well as future research outlook is presented as follows.

8.1 Conclusion

In [SD-1] (Chapter 5), we proposed a method for the automatic detection of human cytomegalovirus nucleocapsids in TEM images. We utilized the concept of transfer learning to leverage models pre-trained on real-world common images. The models were adapted to work on our capsid detection task by employing knowledge transfer of features. Two types of transfer learning techniques were investigated to determine the feasibility of this method for our data. Our experimental results show that transfer learning enabled efficient model performance with only a small dataset of labelled ground truth data. This technique was then developed as a standalone application for HCMV viral particle detection in the Central Facility of Electron Microscopy, Ulm University, to be used by in-house biologists and virologists.

In [SD-2] (Chapter 6), we extended this approach for the automated detection of three types of human cytomegalovirus secondary envelopment stages in TEM images. A synthetic TEM image generation method was proposed for data augmentation as a means of increasing training data size and diversity. A generative adversarial network (GAN) was used to achieve this. As manual labeling of ground truth data is a rather time-consuming and tedious task, automated self-labeling was incorporated with synthetic image generation for fast and efficient ground truth generation. A DL-based detector was then trained with both real ground truth data and synthetic images for the envelopment stages detection. We showed that the addition of synthetic images to the original ground truth dataset resulted in a significant increase in model performance. This reduces the need for unaffordable data labeling efforts.

Moreover, in [SD-3] (Chapter 7), we dealt with the challenging task of segmenting a variety of biological structures in both TEM and SEM images. Our weighted average ensemble model combines predictions from multiple pre-trained DL networks for the multi-class segmentation of EM images. Using this method, we were able to segment the cytoplasm, chromosomes, mitochondria, and nuclei using only small labelled ground truth datasets and generalize well on unseen images. The proposed method significantly outperformed the standard single model approach yielding performance almost similar to expert labeling. In order to increase model transparency and interpretability, we incorporated the gradient-weighted class activation mapping (Grad-CAM) technique into this work. This technique is able to visually indicate regions in an image that were important for the prediction of a certain segmentation class, thereby increasing user trust.

8.2 Future outlook

8.2.1 Semi-supervised and unsupervised learning

DL methods are generally data-expensive and require huge manual labeling efforts. In this thesis, we have introduced techniques to reduce these limitations [SD-1], [SD-2] and [SD-3] (Devan et al. 2019; Shaga Devan et al. 2021, 2022). Unsupervised and semi-supervised learning is another potential solution for this issue as they require very little to no labelled ground truth data. However,

their application in biological EM is rather limited. Despite this, we believe that unsupervised and semi-supervised learning is a fruitful approach that requires further work.

8.2.2 Explainability, interpretability, and transparency

Most machine learning models, such as deep neural networks are black boxes, meaning that it is extremely difficult for humans to comprehend how variables are combined to make the final predictions (Rudin 2019). However, recent events have revealed that a lack of transparency, interpretability, and explainability in deep learning models can lead to severe consequences for its users. Therefore, there has been a rising interest in efforts to improve the interpretability, explainability, and transparency of machine learning models (Doshi-Velez and Kim 2017; Lipton 2017; Chakraborty et al. 2017; Meng et al. 2022; Li et al. 2022).

Explainability indicates the degree to which a model can provide clarity for its predictions, while transparency is the degree to which a model reveals information about its inner workings (Belle and Papantonis, 2021). Model interpretability is a combination of both explainability and transparency and is the degree to which a human can understand the cause of a prediction (Miller 2019). In the biological field, interpretable models are highly desirable to build trust between the user and the model and to enable researchers and practitioners to make a better judgment of the trustworthiness of the model based on their expert knowledge (Sheu 2020). Apart from that, model interpretation helps to improve human understanding of data and algorithm interaction, thereby giving rise to the development of better models. Interpretability is highly critical for researchers and developers to explain their models and understands the value and accuracy of their findings and subsequently perform better model validation. This also enables the development of fair and ethical decision-making models (Leilani et al. 2018).

In order to contribute to a more ethical and fairer AI movement, we have incorporated an interpretable decision-making component into our segmentation model in [SD-3] (Shaga Devan et al. 2022). This enables scientists and practitioners to have a better understanding of DL model's prediction and evaluate the trustworthiness of the model. As fairness and ethics are becoming an

integral part of machine learning model development, further research in this area will be highly beneficial to science and society.

8.2.3 Synthetic data generation

Scarcity of EM training images is a major deterrent to the development of DL applications in biology. Although large volumes of images are produced due to advances in the image acquisition process, collecting and curating these datasets is difficult due to data copyright, privacy and ownership issues. Therefore, the generation of synthetic EM images could help reduce this problem. Generative models have typically been used to generate synthetic images with astonishing results (Karras, Laine and Aila 2019; Brock, Donahue, Simonyan 2018; Ramesh et al. 2021). We have demonstrated the effectiveness of this method for the automatic detection of viral particles in [SD-2] (Shaga Devan et al. 2021). Synthetic data generation warrants further research for the rapid application of DL based automated computer vision tasks in biological EM.

8.2.4 Vision transformer-based methods

Recently, vision transformers have emerged as a powerful competitor to convolutional neural networks for computer vision tasks. They are based on the working nature of transformer structures used in the field of natural language processing (Dosovitskiy et al. 2020). Generally, the transformer learns by measuring relationship between input token pairs. In computer vision, patches of images are taken as the token and the relationship is learned by attention in the network (Dosovitskiy et al. 2020). In biology, vision transformers have yet to outperform DL techniques. Despite this, vision transformers have shown huge potential in the medical field (He et al. 2022) and many research works are currently being carried out to realize their full potential in biology.

9. Bibliography

Abdollahzadeh A, Belevich I, Jokitalo E et al (2021) DeepACSON automated segmentation of white matter in 3D electron microscopy. Commun Biol 4:1–14. https://doi.org/10.1038/s42003-021-01699-w

Akers S, Kautz E, Trevino-Gavito A et al (2021) Rapid and flexible segmentation of electron microscopy data using few-shot machine learning. Npj Comput Mater 7:1–9. https://doi.org/10.1038/s41524-021-00652-z

Akhtar K, Khan SA, Khan SB, Asiri AM (2018) Scanning Electron Microscopy: Principle and applications in nanomaterials characterization. In: Sharma SK (ed) Handbook of Materials Characterization. Springer International Publishing, Cham, pp 113–145

Al-Azzawi A, Ouadou A, Max H et al (2020) DeepCryoPicker: fully automated deep neural network for single protein particle picking in cryo-EM. BMC Bioinformatics 21:509. https://doi.org/10.1186/s12859-020-03809-7

Andrei G, Clercq ED, Snoeck R (2009) Drug targets in Cytomegalovirus infection. Infect Disord - Drug Targets. 9(2):201–222. PMID: 19275707. https://doi.org/10.2174/187152609787847758

Andriasyan V, Yakimovich A, Petkidis A et al (2021) Microscopy deep learning predicts virus infections and reveals mechanics of lytic-infected cells. iScience 24:102543. https://doi.org/10.1016/j.isci.2021.102543

Aryal S (2022) Electron Microscope- definition, principle, types, uses, labelled diagram. In: Microbe Notes. https://microbenotes.com/electron-microscope-principle-types-components-applications-advantages-limitations/. Accessed 29 Sep 2022

Assaiya A, Burada AP, Dhingra S, Kumar J (2021) An overview of the recent advances in cryo-electron microscopy for life sciences. Emerg Top Life Sci. 14;5(1):151-168. https://doi.org/10.1042/ETLS20200295. PMID: 33760078

AZO Life Sciences (2020) What is FIB-SEM? News-Medicalnet. https://www.azolifesciences.com/article/What-is-FIB-SEM.aspx. Accessed on 24 June 2022

Baskaran A, Lin Y, Wen J, Chan MKY (2022) Ensemble of Pre-Trained Neural Networks for Segmentation and Quality Detection of Transmission Electron Microscopy Images. Workshop on

Machine Learning for Materials Science in conjunction with 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. https://doi.org/10.48550/arXiv.2209.01908

Belle V, Papantonis I (2021) Principles and practice of explainable machine learning. Front. Big Data. Sec. Data Mining and Management. https://doi.org/10.3389/fdata.2021.688969

Berrar D, Dubitzky W (2021) Deep learning in bioinformatics and biomedicine. Brief Bioinform. 22:1513–1514. https://doi.org/10.1093/bib/bbab087

Brock A, Donahue J, Simonyan K (2019) Large scale GAN training for high fidelity natural image synthesis. In: Proceedings of the International Conference on Learning Representations (ICLR) 2019. arXiv:1809.11096v2 [cs.LG]. https://doi.org/10.48550/arXiv.1809.11096

Buchholz T-O, Krull A, Shahidi R, et al (2019) Content-aware image restoration for electron microscopy. Methods Cell Biol 152:277–289. https://doi.org/10.1016/bs.mcb.2019.05.001

Cabra V, Samsó M (2015) Do's and don'ts of cryo-electron microscopy: a primer on sample preparation and high quality data collection for macromolecular 3D reconstruction. J Vis Exp JoVE 52311. https://doi.org/10.3791/52311

Callaway E (2020) Revolutionary cryo-EM is taking over structural biology. Nature News. https://www.nature.com/articles/d41586-020-00341-9. Accessed on 2 November 2022

Cannon MJ, Schmid DS, Hyde TB (2010) Review of cytomegalovirus seroprevalence and demographic characteristics associated with infection. Rev Med Virol. 20(4):202-13. https://doi.org/10.1002/rmv.655. PMID: 20564615

Chai J, Zeng H, Li A, Ngai EWT (2021) Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Mach Learn Appl. 6:100134. https://doi.org/10.1016/j.mlwa.2021.100134

Chakraborty S, Tomsett R, Raghavendra R et al (2017) Interpretability of deep learning models: A survey of results. In: 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). pp 1–6. https://doi.org/10.1109/UIC-ATC.2017.8397411

Chinea A (2009) Understanding the principles of recursive neural networks: A Generative Approach to Tackle Model Complexity. In: Alippi C, Polycarpou M, Panayiotou C, Ellinas G (eds) Artificial Neural Networks – ICANN 2009. Springer. Berlin, Heidelberg. pp 952–963

Ching T, Himmelstein DS, Beaulieu-Jones BK et al (2018) Opportunities and obstacles for deep learning in biology and medicine. J R Soc Interface. 15:20170387. https://doi.org/10.1098/rsif.2017.0387

Cid-Mejías A, Alonso-Calvo R, Gavilán H et al (2021) A deep learning approach using synthetic images for segmenting and estimating 3D orientation of nanoparticles in EM images. Comput Methods Programs Biomed. 202:105958. https://doi.org/10.1016/j.cmpb.2021.105958

Couedic TL, Caillon R, Rossant F et al (2020) Deep-learning based segmentation of challenging myelin sheaths. In: 2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA). pp 1–6. https://doi.org/10.1109/IPTA50016.2020.9286715

de Haan K, Ballard ZS, Rivenson Y et al (2019) Resolution enhancement in scanning electron microscopy using deep learning. Sci Rep. 9:12050. https://doi.org/10.1038/s41598-019-48444-2

Deng J, Dong W, Socher R, Li L.-J, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. 2009 IEEE conference on computer vision and pattern recognition. Pp. 248–255.

Devan KS, Walther P, von Einem J et al (2019) Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. Histochem Cell Biol. 151:101–114. https://doi.org/10.1007/s00418-018-1759-5

Doshi-Velez F, Kim B (2017) Towards a rigorous science of interpretable machine learning. arXiv:1702.08608v2 [stat.ML]. https://doi.org/10.48550/arXiv.1702.08608

Dosovitskiy A, Beyer L, Kolesnikov A et al (2020) An Image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929v2 [cs.CV]. https://doi.org/10.48550/arXiv.2010.11929

Ede JM (2021) Deep learning in electron microscopy. Mach. Learn. Sci. Technol. (2)1. https://doi.org/10.48550/arXiv.2009.08328

Fang L, Monroe F, Novak SW et al (2021) Deep learning-based point-scanning super-resolution imaging. Nat Methods. **18:** 406–416. https://doi.org/10.1038/s41592-021-01080-z

Faraz K, Grenier T, Ducottet C et al (2022) Deep learning detection of nanoparticles and multiple object tracking of their dynamic evolution during in situ ETEM studies. Sci Rep. 12:2484. https://doi.org/10.1038/s41598-022-06308-2

Fernandez-Leiro R, Scheres S (2016) Unravelling biological macromolecules with cryo-electron microscopy. Nature. 537:339–346. https://doi.org/10.1038/nature19948

Fischer CA, Besora-Casals L, Rolland SG, et al (2020) MitoSegNet: Easy-to-use Deep Learning Segmentation for Analyzing Mitochondrial Morphology. iScience. 23:101601. https://doi.org/10.1016/j.isci.2020.101601

Flannigan DJ, Barwick B, Zewail AH (2010) Biological imaging with 4D ultrafast electron microscopy. Proc Natl Acad Sci U S A. 107(22):9933-7. https://doi.org/10.1073/pnas.1005653107. Epub 2010 May 17. PMID: 20479261; PMCID: PMC2890498

Forte E, Zhang Z, Thorp EB et al (2020) Cytomegalovirus latency and reactivation: An intricate interplay with the host immune response. Front Cell Infect Microbiol. 10:130. https://doi.org/10.3389/fcimb.2020.00130

Franco-Barranco D, Muñoz-Barrutia A, Arganda-Carreras I (2021) Stable deep neural network qrchitectures for mitochondria segmentation on electron microscopy volumes. Neuroinformatics. https://doi.org/10.1007/s12021-021-09556-1

Franken LE, Grünewald K, Boekema EJ et al (2020) A technical introduction to transmission electron microscopy for soft-matter: Imaging, possibilities, choices, and technical developments. Small. 16:1906198. https://doi.org/10.1002/smll.201906198

George B, Assaiya A, Roy RJ, et al (2021) CASSPER is a semantic segmentation-based particle picking algorithm for single-particle cryo-electron microscopy. Commun Biol. 4:1–12. https://doi.org/10.1038/s42003-021-01721-1

Gholamalinezhad H, Khosravi H (2020) Pooling methods in deep neural networks, a Review. arXiv:2009.07485v1 [cs.CV]. https://doi.org/10.48550/arXiv.2009.07485

Ghosh A, Sumpter BG, Dyck O et al (2021) Ensemble learning-iterative training machine learning for uncertainty quantification and automated experiment in atom-resolved microscopy. npj Comput Mater 7:100. https://doi.org/10.1038/s41524-021-00569-7

Giannatou E, Papavieros G, Constantoudis V et al (2019) Deep learning denoising of SEM images towards noise-reduced LER measurements. Microelectron Eng. 216:111051. https://doi.org/10.1016/j.mee.2019.111051

Gilpin LH, Bau D, Yuan BZ et al(2018) Explaining explanations: An overview of interpretability of machine learning. arXiv:1806.00069v3 [cs.AI]. https://doi.org/10.48550/arXiv.1806.00069

Goecks J, Jalili V, Heiser LM et al (2020) How machine learning will transform biomedicine. Cell. 181:92–101. https://doi.org/10.1016/j.cell.2020.03.022

Golding CG, Lamboo LL, Beniac DR et al (2016) The scanning electron microscope in microbiology and diagnosis of infectious disease. Sci Rep. 6:26516. https://doi.org/10.1038/srep26516

Goldsmith CS, Miller SE (200) Modern uses of electron microscopy for detection of viruses. Clin Microbiol Rev. 22(4):552-63. https://doi.org/10.1128/CMR.00027-09. PMID: 19822888; PMCID: PMC2772359

Gomez-de-Mariscal E, Garcıa-Lopez-de-Haro C, Donati L et al (2021) DeepImageJ: A user-friendly plugin to run deep learning models in ImageJ. Nat Methods. 18:1192–1195. https://doi.org/10.1038/s41592-021-01262-9

Goodfellow IJ, Pouget-Abadie J, Mirza M, et al (2014) Generative adversarial networks. arXiv:1406.2661v1 [stat.ML]. https://doi.org/10.48550/arXiv.1406.2661

Graham L, Orenstein J (2022) Processing tissue and cells for transmission electron microscopy in diagnostic pathology and research. Nat Protoc. 2:2439–2450. https://doi.org/10.1038/nprot.2007.304

Greener JG, Kandathil SM, Moffat L et al (2022) A guide to machine learning for biologists. Nat Rev Mol Cell Biol. 23:40–55. https://doi.org/10.1038/s41580-021-00407-0

Groschner K, Ben-Moshe A, Pattinson A et al (2021) Automated data Labeling and label cleaning for nanoparticle classification in electron microscopy. Microsc Microanal. 27:2526–2528. https://doi.org/10.1017/S1431927621009004

Güven G, Oktay AB (2018) Nanoparticle detection from TEM images with deep learning. In: 2018 26th Signal Processing and Communications Applications Conference (SIU). pp 1–4 https://doi.org/10.1109/SIU.2018.8404468

Haberl MG, Churas C, Tindall L et al (2018) CDeep3M-Plug-and-Play cloud-based deep learning for image segmentation. Nat Methods 15:677–680. https://doi.org/10.1038/s41592-018-0106-z

Haguenau F, Hawkes PW, Hutchison JL et al (2003) Key events in the history of electron microscopy. Microsc Microanal. 9:96–138. https://doi.org/10.1017/S1431927603030113

Hazelton PR, Gelderblom HR (2003) Electron microscopy for rapid diagnosis of infectious agents in emergent situations. Emerg Infect Dis. 9(3):294-303. https://doi.org/10.3201/eid0903.020327. PMID: 12643823; PMCID: PMC2958539

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, pp 770–778. https://doi.org/10.1109/CVPR.2016.90

Hu J, Shen L, Sun G (2018) Squeeze-and-Excitation Networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7132–7141. https://doi.org/10.1109/CVPR. 2018.00745

Iakubovskii (2019) Segmentation Models. Github, GitHub Repository. https://github.com/qubvel/segmentation_models

IBM. Convolutional Neural Networks (2020) IBM Cloud Learn Hub. https://www.ibm.com/cloud/learn/convolutional-neural-networks. Accessed on 19 May 2022

IBM. Neural Networks (2020) IBM Cloud Learn Hub. https://www.ibm.com/cloud/learn/neural-networks. Accessed on 8 June 2022

Ito E, Sato T, Sano D et al (2018) Virus particle detection by convolutional neural network in transmission electron microscopy images. Food Environ Virol. 10:201–208. https://doi.org/10.1007/s12560-018-9335-7

Jackson SE, Chen KC, Groves IJ et al (2021) Latent Cytomegalovirus-driven recruitment of activated CD4+ T cells promotes virus reactivation. Front Immunol. 12:657945. https://doi.org/10.3389/fimmu.2021.657945

Jacobs R, Shen M, Liu Y et al (2022) Performance and limitations of deep learning semantic segmentation of multiple defects in transmission electron micrographs. Cell Rep Phys Sci. 3:100876. https://doi.org/10.1016/j.xcrp.2022.100876

Jerez D, Stuart E, Schmitt K et al (2021) A deep learning approach to identifying immunogold particles in electron microscopy images. Sci Rep 11:7771. https://doi.org/10.1038/s41598-021-87015-2

Janiesch C, Zschech P, Heinrich K (2021) Machine learning and deep learning. Electron Markets 31, 685–695. https://doi.org/10.1007/s12525-021-00475-2

He K, Gan C, Li Z et al (2022) Transformers in Medical Image Analysis: A Review. Intelligent Medicine. ISSN 2667-1026. https://doi.org/10.1016/j.imed.2022.07.002

Karras T, Laine S, Aila T (2019) A style-based generator architecture for generative adversarial networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019. pp. 4396-4405. https://doi.org/10.1109/CVPR.2019.00453

Kersting K (2018) Machine Learning and Artificial Intelligence: Two fellow travellers on the quest for intelligent behaviour in machines. Sec. Machine Learning and Artificial Intelligence. https://doi.org/10.3389/fdata.2018.00006

Khadangi A, Boudier T, Rajagopal V (2020) EM-net: Deep learning for electron microscopy image segmentation. bioRxiv 2020.02.03.933127. https://doi.org/10.1101/2020.02.03.933127

Khadangi A, Boudier T, Rajagopal V (2021) EM-stellar: benchmarking deep learning for electron microscopy image segmentation. Bioinformatics. 37:97–106. https://doi.org/10.1093/bioinformatics/btaa1094

Kharin AYu (2020) Deep learning for scanning electron microscopy: Synthetic data for the nanoparticles detection. Ultramicroscopy. 219:113125. https://doi.org/10.1016/j.ultramic.2020.113125

Kizilyaprak C, Daraspe J, Humbel B (2014) Focused ion beam scanning electron microscopy in biology. J Microsc. 254:109–114. https://doi.org/10.1111/jmi.12127

Kniesel H, Ropinski T, Bergner T et al (2022) Clean Implicit 3D Structure from noisy 2D STEM images. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp 20730–20740. https://doi.org/10.1109/CVPR52688.2022.02010

Koster AJ, Klumperman J (2003) Electron microscopy in cell biology: integrating structure and function. Nat Rev Mol Cell Biol. Suppl:SS6-10. PMID: 14587520

Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet Classification with deep convolutional neural networks. In: Communications of the ACM. 60 Issue 6:84–90. https://doi.org/10.1145/3065386

Krull A, Buchholz T-O, Jug F (2019) Noise2Void - learning denoising from single noisy images. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, USA. pp.2124-2132. https://doi.org/10.1109/CVPR.2019.00223

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature. 521:436–444. https://doi.org/10.1038/nature14539

LeCun Y, Bottou L, Bengio Y et al (1998) Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, 86(11):2278-2324. https://doi.org/10.1109/5.726791

Lehtinen J, Munkberg J, Hasselgren J et al (2018) Noise2Noise: Learning image restoration without clean data. arXiv:1803.04189v3 [cs.CV]. https://doi.org/10.48550/arXiv.1803.0418

Li H, Chen G, Gao S, et al (2021) PickerOptimizer: A deep learning-based particle optimizer for cryo-electron microscopy particle-picking algorithms. In: Wei, Y., Li, M., Skums, P., Cai, Z. (eds) Bioinformatics Research and Applications. ISBRA 2021. Lecture Notes in Computer Science, 13064. Springer, Cham. https://doi.org/10.1007/978-3-030-91415-8_46

Li H, Zhang H, Wan X et al (2022) Noise-Transfer2Clean: Denoising cryo-EM images based on noise modeling and transfer. Bioinformatics. 38(7):2022–2029. https://doi.org/10.1093/bioinformatics/btac052

Li X, Xiong H, Li X et al (2022) Interpretable deep learning: interpretation, interpretability, trustworthiness, and beyond. Knowl Inf Syst. https://doi.org/10.1007/s10115-022-01756-8

Li Y, Cash JN, Tesmer JJG et al (2020) High-throughput Cryo-EM enabled by user-free preprocessing routines. Structure. 28(7):858-869.e3. ISSN 0969-2126. https://doi.org/10.1016/j.str.2020.03.008

Liashchynskyi P, Liashchynskyi P (2019) Grid Search. A Big Comparison for NAS, Random Search, Genetic Algorithm

Lipton ZC (2017) The mythos of model interpretability. arXiv:1606.03490v3 [cs.LG]. https://doi.org/10.48550/arXiv.1606.03490

Matuszewski DJ, Sintorn I-M (2021) TEM virus images: Benchmark dataset and deep learning classification. Comput Methods Programs Biomed. 209:106318. https://doi.org/10.1016/j.cmpb.2021.106318

McIntosh JR (2001) Electron microscopy of cells: A new beginning for a new century. J Cell Biol. 153 (6): F25–F32. https://doi.org/10.1083/jcb.153.6.F25

Meng C, Trinh L, Xu N et al (2022) Interpretability and fairness evaluation of deep learning models on MIMIC-IV dataset. Sci Rep. 12:7166. https://doi.org/10.1038/s41598-022-11012-2

Mercorelli B, Lembo D, Palù G, Loregian A (2011) Early inhibitors of human cytomegalovirus: state-of-art and therapeutic perspectives. Pharmacol Ther. 131:309–329. https://doi.org/10.1016/j.pharmthera.2011.04.007

Miller T (2019) Explanation in artificial intelligence: Insights from the social sciences. Artif Intell 267:1–38. https://doi.org/10.1016/j.artint.2018.07.007
Milne JLS, Borgnia MJ, Bartesaghi A et al (2013) Cryo-electron microscopy: A primer for the non-microscopist. FEBS J 280:28–45. https://doi.org/10.1111/febs.12078

Mishra M, Srivastava M (2014) A view of artificial neural network. In: 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014). pp 1–3. https://doi.org/10.1109/ICAETR.2014.7012785

Mo Y, Wu Y, Yang X et al (2022) Review the state-of-the-art technologies of semantic segmentation based on deep learning. Neurocomputing. 493:626-646. ISSN 0925-2312. https://doi.org/10.1016/j.neucom.2022.01.005

Mohan S, Manzorro R, Vincent JL et al (2022) Deep denoising for scientific discovery: A case study in electron microscopy. IEEE Trans Comput Imaging. 1–1. https://doi.org/10.1109/TCI.2022.3176536

Möller L, Holland G, Laue M (2020) Diagnostic electron microscopy of viruses with low-voltage electron microscopes. J Histochem Cytochem Off J Histochem Soc. 68:389–402. https://doi.org/10.1369/0022155420929438

Nguyen NP, Ersoy I, Gotberg J et al (2021) DRPnet: automated particle picking in cryo-electron micrographs using deep regression. BMC Bioinformatics. 22:55. https://doi.org/10.1186/s12859-020-03948-x

Northcutt CG, Athalye A, Mueller J (2021) Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. arXiv:2103.14749v4 [stat.ML]. https://doi.org/10.48550/arXiv.2103.14749

O'Shea K, Nash R (2015) An introduction to convolutional neural networks. arXiv:1511.08458v2. https://doi.org/10.48550/arXiv.1511.08458

Pelletier L, O'Toole E, Schwager A et al (2006) Centriole assembly in Caenorhabditis elegans. Nature. 444:619–623. https://doi.org/10.1038/nature05318

Radulović S, Sunkara S, Rachel R et al (2022). Three-dimensional SEM, TEM, and STEM for analysis of large-scale biological systems. Histochem Cell Biol. 158:203–211. https://doi.org/10.1007/s00418-022-02117-w

Ramesh A, Pavlov M, Goh G et al (2021). Zero-Shot Text-to-Image Generation. arXiv:2102.12092v2 [cs.CV]. https://doi.org/10.48550/arXiv.2102.12092

Read C, Schauflinger M, Nikolaenko D et al (2019) Regulation of human cytomegalovirus secondary envelopment by a C-Terminal tetralysine motif in pUL71. J Virol. 93:e02244-18. https://doi.org/10.1128/JVI.02244-18

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards realtime object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Richert-Pöggeler KR, Franzke K, Hipp K, Kleespies RG (2019) Electron microscopy methods for virus diagnosis and high resolution analysis of viruses. Front Microbiol. 9:3255. https://doi.org/10.3389/fmicb.2018.03255

Ronneberger O, Fischer P, Brox T (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv150504597 Cs

Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell. 1:206–215. https://doi.org/10.1038/s42256-019-0048-x

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet Large Scale Visual Recognition Challenge. IJCV 2015.

Sapoval N, Aghazadeh A, Nute MG et al (2022) Current progress and open challenges for applying deep learning across the biosciences. Nat Commun. 13:1728. https://doi.org/10.1038/s41467-022-29268-7

Sarker IH (2021) Deep Learning: A comprehensive overview on techniques, taxonomy, applications and research directions. SN Comput Sci. 2:420. https://doi.org/10.1007/s42979-021-00815-1

Sehar, U, Naseem, ML (2022) How deep learning is empowering semantic segmentation. Multimed Tools Appl. 81:30519–30544. https://doi.org/10.1007/s11042-022-12821-3

Shaga Devan K, Kestler HA, Read C, Walther P (2022) Weighted average ensemble-based semantic segmentation in biological electron microscopy images. Histochem Cell Biol. https://doi.org/10.1007/s00418-022-02148-3

Shaga Devan K, Walther P, von Einem J, et al (2021) Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network. Cell Microbiol. 23:e13280. https://doi.org/10.1111/cmi.13280

Shaham, T. R., Dekel, T., & Michaeli, T. (2019). SinGAN:Learning a Generative Model from a Single Natural Image. IEEE International Conference on Computer Vision, Seoul, Korea, October 27-November 2 2019, arXiv: 1905.01164v2.

Sheu Y (2020) Illuminating the black box: Interpreting deep deural network models for psychiatric research. Front Psychiatry. 11:551299. https://doi.org/10.3389/fpsyt.2020.551299

Shin HC, Roth HR, Gao M, Lu L, Xu Z, Nogues I, Yao J, Mollura D, Summers RM (2016) Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE Trans Med Imaging. 35:1285–1298. https://doi.org/10.1109/tmi.2016.2528162

Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. https://arxiv.org/abs/1409.1556

Spiers H, Songhurst H, Nightingale L et al (2021) Deep learning for automatic segmentation of the nuclear envelope in electron microscopy data, trained with volunteer segmentations. Traffic Cph Den. 22:240–253. https://doi.org/10.1111/tra.12789

Suveer A, Gupta A, Kylberg G, Sintorn IM (2019). Super-Resolution reconstruction of transmission electron microscopy images using deep learning. In: IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019). pp. 548-551. https://doi.org/10.1109/ISBI.2019.8759153

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the Inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, pp 2818–2826. https://doi.org/10.1109/ cvpr.2016.308

Tajbakhsh N, Shin JY, Gurudu SR, Hurst RT, Kendall CB, Gotway MB, Liang J (2016) Convolutional neural networks for medical image analysis: full training or Fine tuning? IEEE Trans Med Imaging. 35:1299–1312. https://doi.org/10.1109/TMI.2016.2535302

Tan M, Le QV (2020) EfcientNet: Rethinking Model Scaling for Convolutional Neural Networks. ArXiv190511946 Cs Stat

ThermoFisher Sceintific (2022) Electron Microscopy. https://www.thermofisher.com/de/de/home/materials-science/learning-center/applications/sem-tem-difference.html. Accessed on: 14 May 2022

Treder KP, Huang C, Kim JS et al. (2022) Applications of deep learning in electron microscopy. Microscopy. (71)1: i100–i115. https://doi.org/10.1093/jmicro/dfab043

Urakubo H, Bullmann T, Kubota Y et al (2019) UNI-EM: An environment for deep neural network-based automated segmentation of neuronal electron microscopic images. Sci Rep. 9: 19413. https://doi.org/10.1038/s41598-019-55431-0

Vasudevan RK, Jesse S (2019) Deep learning as a tool for image denoising and drift correction. Microsc Microanal. 25:190–191. https://doi.org/10.1017/S1431927619001685

Villinger C et al (2014) Three-dimensional imaging of adherent cells using FIB/SEM and STEM. In: Kuo, J. (eds) Electron Microscopy. Methods in Molecular Biology.1117. Humana Press, Totowa, NJ. https://doi.org/10.1007/978-1-62703-776-1_27

Villinger C, Neusser G, Kranz C et al (2015) 3D analysis of HCMV induced-nuclear membrane structures by FIB/SEM tomography: Insight into an unprecedented membrane morphology. Viruses. 7(11):5686-704. https://doi.org/10.3390/v7112900. PMID: 26556360; PMCID: PMC4664973

von Chamier L, Laine RF, Jukkala J et al (2021) Democratising deep learning for microscopy with ZeroCostDL4Mic. Nat Commun. 12:2276. https://doi.org/10.1038/s41467-021-22518-0

Wainberg M, Merico D, Delong A, Frey BJ (2018) Deep learning in biomedicine. Nat Biotechnol. 36:829–838. https://doi.org/10.1038/nbt.4233

Walther P and Ziegler A (2002) Freeze substitution of high-pressure frozen samples: the visibility of biological membranes is improved when the substitution medium contains water. Journal of Microscopy. 208: 3-10. https://doi.org/10.1046/j.1365-2818.2002.01064.x

Walther P, Schmid E, Höhn K (2013) High-pressure freezing for scanning transmission electron tomography analysis of cellular organelles. Methods Mol Biol. 931:525-35. https://doi.org/10.1007/978-1-62703-056-4_28. PMID: 23027022

Wang F, Gong H, Liu G et al (2016) DeepPicker: A deep learning approach for fully automated particle picking in cryo-EM. J Struct Biol. 195:325–336. https://doi.org/10.1016/j.jsb.2016.07.006

Wang F, Henninen TR, Keller D, Erni R (2020) Noise2Atom: unsupervised denoising for scanning transmission electron microscopy images. Appl Microsc. 50:23. https://doi.org/10.1186/s42649-020-00041-8

Wang J, Lan C, Wang C, Gao Z (2021) Deep learning super-resolution electron microscopy based on deep residual attention network. Int J Imaging Syst Technol. 31:2158–2169. https://doi.org/10.1002/ima.22588

Weber GH, Ophus C, Ramakrishnan L (2018) Automated labeling of electron microscopy images using deep learning. In: 2018 IEEE/ACM Machine Learning in HPC Environments (MLHPC). pp 26–36. https://doi.org/10.1109/MLHPC.2018.8638633

Weiss K, Khoshgoftaar TM, Wang D (2016). A survey of transfer learning. J Big Data. 3:9. https://doi.org/10.1186/s40537-016-0043-6

Williams DB, Carter CB (1996) The transmission electron microscope. In: Williams DB, Carter CB (eds) Transmission Electron Microscopy: A Textbook for Materials Science. Springer US, Boston, MA, pp 3–17

Winey M, Meehl JB, O'Toole ET et al(2014) Conventional transmission electron microscopy. Mol Biol Cell. 25:319–323. https://doi.org/10.1091/mbc.E12-12-0863

Xiao C, Chen X, Li W et al (2018) Automatic mitochondria segmentation for EM data Using a 3D supervised convolutional network. Front Neuroanat. 12. https://doi.org/10.3389/fnana.2018.00092

Yamashita R, Nishio M, Do RKG, Togashi K (2018) Convolutional neural networks: an overview and application in radiology. Insights Imaging. 9:611–629. https://doi.org/10.1007/s13244-018-0639-9

Yin B, Hu Q, Zhu Y, Zhao C, Zhou K (2022) Paw-Net: Stacking ensemble deep learning for segmenting scanning electron microscopy images of fine-grained shale samples. Computers & Geosciences 168. https://doi.org/10.1016/j.cageo.2022.105218

Yury SB, Cohen N, Gabrielli N et al (2019) High-throughput ultrastructure screening using electron microscopy and fluorescent barcoding. J Cell Biol. 218 (8): 2797–2811. https://doi.org/10.1083/jcb.201812081

Yougui Liao (2007) Practical Electron Microscopy and Database, Northwestern University, USA. https://www.globalsino.com/

Zhu Y, Ouyang Q, Mao Y (2017) A deep convolutional neural network approach to single-particle recognition in cryo-electron microscopy. BMC Bioinformatics. 18:348. https://doi.org/10.1186/s12859-017-1757-y

10. Supplementary Materials

Supplementary Materials: Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning [SD-1] (Chapter 5)

Devan KS, Walther P, von Einem J, Ropinski T, Kestler HA, Read C. Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. Histochem Cell Biol. 2019 Feb;151(2):101-114. https://doi.org/10.1007/s00418-018-1759-5. Epub 2018 Nov 28. PMID: 30488339.

Supplementary Material

Coding for the methods described in our work

Model: CNN-TFS

```
import datetime
import keras
import os
import numpy as np
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import ZeroPadding2D, AveragePooling2D
from keras.layers import Dense, Dropout, Activation, Flatten,
BatchNormalization, Input
from keras.layers import GlobalAveragePooling2D
from keras.applications.inception_v3 import InceptionV3
from keras.models import Model
from keras.utils.data_utils import get_file
from keras import optimizers
from keras import backend as K
import matplotlib.pyplot as plt
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.metrics import categorical_crossentropy
from sklearn.metrics import confusion_matrix, classification_report
import itertools
from sklearn.metrics import roc_curve, auc

plt.style.use('ggplot')


batch_size = 16
numEpochs = 100
img_width, img_height = 52, 52
model_architecture = 'convnet_4'

data_dir = 'data/training-patches_latest'
nb_train_samples = 13008
nb_validation_samples = 3248
train_data_dir = data_dir + '/train'
validation_data_dir =  data_dir + '/validation'

# timestamp as string used to name output files of the current run
timeStamp = datetime.datetime.now().strftime("%y%m%d-%H%M")

if K.image_data_format() == 'channels_first':
    input_shape = (3, img_width, img_height)
else:
```

```
    input_shape = (img_width, img_height, 3)

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))

model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])



# serialize model to YAML - yaml file is used for object detection
model_yaml = model.to_yaml()
if not os.path.exists('models/'):
    os.makedirs('models/')
with open('models/' + model_architecture + '_tt' + timeStamp + '.yaml',
'w') as yaml_file:
    yaml_file.write(model_yaml)


# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
    rescale = 1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
```

```
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary', shuffle=False)



history = model.fit_generator(
          train_generator,
          steps_per_epoch=nb_train_samples // batch_size,
          epochs=numEpochs,
          validation_data=validation_generator,
          validation_steps=nb_validation_samples // batch_size)

model.save_weights('models/' + model_architecture + '_tt' + timeStamp +
'.h5')
```

Model : VGG16-FE

```
from keras.applications import VGG16

import datetime
import keras
import os
import numpy as np
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import ZeroPadding2D, AveragePooling2D
from keras.layers import Dense, Dropout, Activation, Flatten,
BatchNormalization, Input
from keras.layers import GlobalAveragePooling2D
from keras.applications.inception_v3 import InceptionV3
from keras.models import Model
from keras.utils.data_utils import get_file
from keras import optimizers
from keras import backend as K
import matplotlib.pyplot as plt
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.metrics import categorical_crossentropy
from sklearn.metrics import confusion_matrix, classification_report
from keras.applications.vgg16 import VGG16
from keras.applications.resnet50 import ResNet50
```

```
import itertools
from sklearn.metrics import roc_curve, auc
from keras import models
from keras import layers


numEpochs = 100
batch_size = 16
img_width, img_height = 150, 150
model_architecture = 'vgg-fe'


# timestamp as string used to name output files of the current run
timeStamp = datetime.datetime.now().strftime("%y%m%d-%H%M")


data_dir = 'data/training-patches_latest'
nb_train_samples = 13008
nb_validation_samples = 3248
train_data_dir = data_dir + '/train'
validation_data_dir =  data_dir + '/validation'


conv_base = VGG16(weights='imagenet',include_top=False,input_shape=(150,
150, 3))
#conv_base.summary()


model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))

#model.summary()

#print('This is the number of trainable weights '      'before freezing
the conv base:', len(model.trainable_weights))

conv_base.trainable = False

#print('This is the number of trainable weights '       'after freezing
the conv base:', len(model.trainable_weights))

train_datagen = ImageDataGenerator(
    rescale = 1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255)
```

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary', shuffle=False)
```

```
model.compile(loss='binary_crossentropy',
                  optimizer=optimizers.RMSprop(lr=2e-5),
                  metrics=['accuracy'])

# serialize model to YAML - yaml file is used for object detection
model_yaml = model.to_yaml()
if not os.path.exists('models/'):
    os.makedirs('models/')
with open('models/' + model_architecture + '_tt' + timeStamp + '.yaml',
'w') as yaml_file:
    yaml_file.write(model_yaml)


history = model.fit_generator(
        train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=numEpochs,
        validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size)


model.save_weights('models/' + model_architecture + '_tt' + timeStamp +
'.h5')
```

Model: InceptionV3-FE

```
import datetime
import keras
import os
import numpy as np
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
```

```
from keras.layers import ZeroPadding2D, AveragePooling2D
from keras.layers import Dense, Dropout, Activation, Flatten,
BatchNormalization, Input
from keras.layers import GlobalAveragePooling2D
from keras.applications.inception_v3 import InceptionV3
from keras.applications.resnet50 import ResNet50
from keras.applications.vgg16 import VGG16
from keras.models import Model
from keras.utils.data_utils import get_file
from keras import optimizers
from keras import backend as K
import matplotlib.pyplot as plt
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.metrics import categorical_crossentropy
from sklearn.metrics import confusion_matrix, classification_report
import itertools
from sklearn.metrics import roc_curve, auc
from keras import models
from keras import layers


numEpochs = 100
batch_size = 16
img_width, img_height = 299, 299
model_architecture = 'inception-fe'


# timestamp as string used to name output files of the current run
timeStamp = datetime.datetime.now().strftime("%y%m%d-%H%M")


data_dir = 'data/training-patches_latest'
nb_train_samples = 13008
nb_validation_samples = 3248#647#1302
train_data_dir = data_dir + '/train'
validation_data_dir =  data_dir + '/validation'

#conv_base.summary()
conv_base = InceptionV3(weights='imagenet', include_top=False,
input_shape=(299, 299, 3))


model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))

#model.summary()


#print('This is the number of trainable weights before freezing the conv
base:', len(model.trainable_weights))
```

```
conv_base.trainable = False

#print('This is the number of trainable weights after freezing the conv
base:', len(model.trainable_weights))

train_datagen = ImageDataGenerator(
    rescale = 1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary', shuffle=False)


model.compile(loss='binary_crossentropy',
                  optimizer=optimizers.RMSprop(lr=2e-5),
                  metrics=['accuracy'])

# serialize model to YAML - yaml file is used for object detection
model_yaml = model.to_yaml()
if not os.path.exists('models/'):
    os.makedirs('models/')
with open('models/' + model_architecture + '_tt' + timeStamp + '.yaml',
'w') as yaml_file:
    yaml_file.write(model_yaml)


history = model.fit_generator(
        train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=numEpochs,
        validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size)


model.save_weights('models/' + model_architecture + '_tt' + timeStamp +
'.h5')
```

Model: ResNet50-FE

```
import datetime
import keras
import os
import sys
import numpy as np
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import ZeroPadding2D, AveragePooling2D
from keras.layers import Dense, Dropout, Activation, Flatten,
BatchNormalization, Input
from keras.layers import GlobalAveragePooling2D
from keras.applications.inception_v3 import InceptionV3
from keras.models import Model
import matplotlib.pyplot as plt
from keras.utils.data_utils import get_file
from keras import optimizers
from keras import backend as K
from keras.applications.resnet50 import ResNet50
from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.metrics import confusion_matrix, classification_report
from keras.applications.inception_v3 import InceptionV3
from keras.callbacks import ModelCheckpoint
from keras.callbacks import TensorBoard
import os.path

from sklearn.metrics import roc_curve, auc
plt.style.use('ggplot')



data_dir = 'data/training-patches_latest'
train_data_dir = data_dir + '/train'
validation_data_dir =  data_dir + '/validation'

model_architecture = 'Resnet-FE'

numEpochs = 100
img_width, img_height = 224, 224  # change based on the shape/structure of
your images
batch_size = 16  # try 4, 8, 16, 32, 64, 128, 256 dependent on CPU/GPU
memory capacity (powers of 2 values).
nb_train_samples = 13008
nb_validation_samples = 3248
top_model_wt_path = "cl_model.h5"

top_layers_checkpoint_path = 'cp.feature_resnet-fe.hdf5'
```

```
#fine_tuned_checkpoint_path = 'cp.fine_tuned.best_inception_dt_165.hdf5'


# timestamp as string used to name output files of the current run
timeStamp = datetime.datetime.now().strftime("%y%m%d-%H%M")


input_tensor = Input(shape=(224, 224, 3))  # this assumes
K.image_data_format() == 'channels_last'

    # create the base pre-trained model
base_model =
ResNet50(input_tensor=input_tensor,weights='imagenet',include_top=False)



x = base_model.output
x = GlobalAveragePooling2D(data_format='channels_last')(x)
x = Dense(1, activation='sigmoid')(x)

model = Model(base_model.input, x)

for layer in base_model.layers:
    layer.trainable=False


# serialize model to YAML
model_yaml = model.to_yaml()
if not os.path.exists('models/'):
    os.makedirs('models/')
with open('models/' + model_architecture + '_tt' + timeStamp + '.yaml',
'w') as yaml_file:
    yaml_file.write(model_yaml)

model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])



train_datagen = ImageDataGenerator(
rescale = 1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator()

train_generator = train_datagen.flow_from_directory(
      train_data_dir,
      target_size=(img_width, img_height),
      batch_size=batch_size,
      class_mode='binary', shuffle=True)

validation_generator = test_datagen.flow_from_directory(
```

```
      validation_data_dir,
      target_size=(img_width, img_height),batch_size=batch_size,
      class_mode='binary', shuffle=True )

mc_top = ModelCheckpoint(top_layers_checkpoint_path, monitor='val_acc',
verbose=0, save_best_only=True)

print("start history model")

history = model.fit_generator(
      train_generator,
      steps_per_epoch=nb_train_samples // batch_size,
      epochs=numEpochs,
      validation_data=validation_generator,
      validation_steps=nb_validation_samples // batch_size,
callbacks=[mc_top])


model.save_weights('models/' + model_architecture + '_tt' + timeStamp +
'.h5')
```

Model: VGG16-FT

```
import datetime
import keras
import os
import numpy as np
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import ZeroPadding2D, AveragePooling2D
from keras.layers import Dense, Dropout, Activation, Flatten,
BatchNormalization, Input
from keras.layers import GlobalAveragePooling2D
from keras.applications.inception_v3 import InceptionV3
from keras.models import Model
from keras.utils.data_utils import get_file
from keras import optimizers
from keras import backend as K
import matplotlib.pyplot as plt
from keras.callbacks import ModelCheckpoint, EarlyStopping


#model_architecture = 'vgg16-ft' # vgg16 requires the number of samples to
be a multiple of batch_size

numEpochs = 100
batch_size = 16


data_dir = 'data/training-patches_merged'
nb_train_samples = 13008
nb_validation_samples = 3248

train_data_dir = data_dir + '/train'
validation_data_dir =  data_dir + '/validation'

# timestamp as string used to name output files of the current run
timeStamp = datetime.datetime.now().strftime("%y%m%d-%H%M")


def createModelVGG16(img_width, img_height):

    def save_bottleneck_features():
        datagen = ImageDataGenerator(rescale=1 / 255)

        # build the vgg16 model
        model = applications.VGG16(include_top=False, weights='imagenet')

        generator = datagen.flow_from_directory(train_data_dir,
target_size=(img_width, img_height), shuffle=False, class_mode=None,
```

```
                                                batch_size=batch_size)  #
class_mode=None means our data will only yield
        # batches of data, no labels, shuffle=False means our data will be
in order

        # generates predication for a generator. Steps: total no of
batches. Returns a numpy array of predictions
        bottleneck_features_train =
model.predict_generator(generator=generator, steps=nb_train_samples //
batch_size)
        # saves an array to a binary file
        np.save(file='models/vgg16_tt' + timeStamp +
'_bottleneck_features_train.npy', arr=bottleneck_features_train)

        generator = datagen.flow_from_directory(validation_data_dir,
target_size=(img_width, img_height), batch_size=batch_size,
                                                class_mode=None,
shuffle=False)
        bottleneck_features_validation =
model.predict_generator(generator, nb_validation_samples // batch_size)
        np.save(file='models/vgg16_tt' + timeStamp +
'_bottleneck_features_validate.npy', arr=bottleneck_features_validation)

    def train_top_model():
        train_data = np.load(file='models/vgg16_tt' + timeStamp +
'_bottleneck_features_train.npy')
        train_labels = np.array([0] * (nb_train_samples // 2) + [1] *
(nb_train_samples // 2)) #np.empty_like(train_data)#

        validation_data = np.load(file='models/vgg16_tt' + timeStamp +
'_bottleneck_features_validate.npy')
        validation_labels = np.array([0] * (nb_validation_samples // 2) +
[1] * (nb_validation_samples // 2))#= np.empty_like(validation_data)#

        model = Sequential()
        model.add(Flatten(input_shape=train_data.shape[1:]))  # don't need
to tell batch size in input shape
        model.add(Dense(256, activation='relu'))
        model.add(Dropout(0.5))
        model.add(Dense(1, activation='sigmoid'))

        model.compile(optimizer='rmsprop',
                      loss='binary_crossentropy', metrics=['accuracy'])

        model.fit(train_data, train_labels,
                  epochs=numEpochs,
                  batch_size=batch_size,
                  validation_data=(validation_data, validation_labels))
        model.save_weights('models/vgg16_tt' + timeStamp +
'_prefinetuning.h5')


    save_bottleneck_features()
    train_top_model()
```

```
    # build the VGG16 network
    base_model = applications.VGG16(weights='imagenet', include_top=False,
input_shape=(150,150,3))

    # build a classifier model to put on top of the convolutional model
    top_model = Sequential()
    top_model.add(Flatten(input_shape=base_model.output_shape[1:]))
    top_model.add(Dense(256, activation='relu'))
    top_model.add(Dropout(0.5))
    top_model.add(Dense(1, activation='sigmoid'))

    # note that it is necessary to start with a fully-trained
    # classifier, including the top classifier,
    # in order to successfully do fine-tuning
    top_model.load_weights('models/vgg16_tt' + timeStamp +
'_prefinetuning.h5')

    # add the model on top of the convolutional base
    # model.add(top_model)
    model = Model(inputs=base_model.input,
outputs=top_model(base_model.output))

    # set the first 25 layers (up to the last conv block)
    # to non-trainable (weights will not be updated)
    for layer in model.layers[:15]:
        layer.trainable = False

        # compile the model with a SGD/momentum optimizer
    # and a very slow learning rate.
    model.compile(loss='binary_crossentropy',
                  optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
                  metrics=['accuracy'])
    return model


# set parameters and build architecture for desired model
if model_architecture == 'vgg16-ft':
    img_width, img_height = 150, 150
    model = createModelVGG16(img_width, img_height)

# serialize model to YAML
model_yaml = model.to_yaml()
if not os.path.exists('models/'):
    os.makedirs('models/')
with open('models/' + model_architecture + '_tt' + timeStamp + '.yaml',
'w') as yaml_file:
    yaml_file.write(model_yaml)


# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
    rescale = 1.0 / 255,
    shear_range=0.2,
```

```
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=numEpochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size)

model.save_weights('models/' + model_architecture + '_tt' + timeStamp +
'.h5')
```

Model: InceptionV3 - FT

```
import datetime
import keras
import os
import sys
import numpy as np
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import ZeroPadding2D, AveragePooling2D
from keras.layers import Dense, Dropout, Activation, Flatten,
BatchNormalization, Input
from keras.layers import GlobalAveragePooling2D
from keras.applications.inception_v3 import InceptionV3
from keras.models import Model
import matplotlib.pyplot as plt
from keras.utils.data_utils import get_file
from keras import optimizers
```

```
from keras import backend as K
from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.metrics import confusion_matrix, classification_report
from keras.applications.inception_v3 import InceptionV3
from keras.callbacks import ModelCheckpoint
from keras.callbacks import TensorBoard
import os.path

from sklearn.metrics import roc_curve, auc
plt.style.use('ggplot')



data_dir = 'data/training-patches_latest'
train_data_dir = data_dir + '/train'
validation_data_dir =  data_dir + '/validation'

numEpochs = 100
img_width, img_height = 299, 299  # change based on the shape/structure of
your images
batch_size = 16  # try 4, 8, 16, 32, 64, 128, 256 dependent on CPU/GPU
memory capacity (powers of 2 values).
nb_train_samples = 13008
nb_validation_samples = 3248
top_model_wt_path = "cl_model.h5"


top_layers_checkpoint_path = 'cp.top.best.hdf5'
fine_tuned_checkpoint_path = 'cp.fine_tuned.best.hdf5'




timeStamp = datetime.datetime.now().strftime("%y%m%d-%H%M")
model_architecture = 'InceptionV3-FT'




# create the base pre-trained model
base_model = InceptionV3(weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(1024, activation='relu')(x)
# and a logistic layer -- let's say we have 200 classes
predictions = Dense(1, activation='sigmoid')(x)



# this is the model we will train
```

```
model = Model(inputs=base_model.input, outputs=predictions)



# first: train only the top layers (which were randomly initialized)
# i.e. freeze all convolutional InceptionV3 layers
for layer in base_model.layers:
    layer.trainable = False


# serialize model to YAML
model_yaml = model.to_yaml()
if not os.path.exists('models/'):
    os.makedirs('models/')
with open('models/' + model_architecture + '_tt' + timeStamp + '.yaml',
'w') as yaml_file:
    yaml_file.write(model_yaml)

# compile the model (should be done *after* setting layers to non-
trainable)
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])

 # this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
rescale = 1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)


test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
            train_data_dir,
            target_size=(img_width, img_height),
            batch_size=batch_size,
            class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
            validation_data_dir,
            target_size=(img_width, img_height),
            batch_size=batch_size,
            class_mode='binary', shuffle=False)

mc_top = ModelCheckpoint(top_layers_checkpoint_path, monitor='val_acc',
verbose=0, save_best_only=True)

#Save the TensorBoard logs.
#tb = TensorBoard(log_dir='./logs', histogram_freq=1, write_graph=True,
write_images=True)

model.fit_generator(
```

```
          train_generator,
          steps_per_epoch=nb_train_samples // batch_size,
          epochs=numEpochs,
          validation_data=validation_generator,
          validation_steps=nb_validation_samples // batch_size,
callbacks=[mc_top])


# let's visualize layer names and layer indices to see how many layers
# we should freeze:
#for i, layer in enumerate(base_model.layers):
   #print(i, layer.name)


mc_fit = ModelCheckpoint(fine_tuned_checkpoint_path, monitor='val_acc',
verbose=0, save_best_only=True)


# we chose to train the top 2 inception blocks, i.e. we will freeze
# the first 249 layers and unfreeze the rest: try 229, 197, 165
for layer in model.layers[:249]:
   layer.trainable = False
for layer in model.layers[249:]:
   layer.trainable = True

# we need to recompile the model for these modifications to take effect
# we use SGD with a low learning rate
from keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9),
loss='binary_crossentropy', metrics=['accuracy'])


history = model.fit_generator(
          train_generator,
          steps_per_epoch=nb_train_samples // batch_size,
          epochs=numEpochs,
          validation_data=validation_generator,
          validation_steps=nb_validation_samples // batch_size,
callbacks=[mc_fit])

model.save_weights('models/' + model_architecture + '_tt' + timeStamp +
'.h5')
```

Model: ResNet50-FT

```
"""

import datetime
import keras
```

```
import os
import numpy as np
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import ZeroPadding2D, AveragePooling2D
from keras.layers import Dense, Dropout, Activation, Flatten,
BatchNormalization, Input
from keras.layers import GlobalAveragePooling2D
from keras.applications.inception_v3 import InceptionV3
from keras.models import Model
from keras.utils.data_utils import get_file
from keras import optimizers
from keras import backend as K
import matplotlib.pyplot as plt
from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import roc_curve, auc

plt.style.use('ggplot')


model_architecture = 'Resnet-FT' #
numEpochs = 100
batch_size = 16
img_width, img_height = 224, 224


data_dir = 'data/training-patches_latest'
nb_train_samples = 13008
nb_validation_samples = 3248
train_data_dir = data_dir + '/train'
validation_data_dir =  data_dir + '/validation'

# timestamp as string used to name output files of the current run
timeStamp = datetime.datetime.now().strftime("%y%m%d-%H%M")



def createModelResnetFT(img_width, img_height):

    def save_bottleneck_features():
        datagen = ImageDataGenerator(rescale=1 / 255)

        # build the model
        model = applications.ResNet50(include_top=False,
weights='imagenet',input_shape=(224, 224, 3))

        generator = datagen.flow_from_directory(train_data_dir,
target_size=(img_width, img_height), shuffle=False, class_mode=None,
                                                batch_size=batch_size)  #
class_mode=None means our data will only yield
```

```
        # batches of data, no labels, shuffle=False means our data will be
in order

        # generates predication for a generator. Steps: total no of
batches. Returns a numpy array of predictions
        bottleneck_features_train =
model.predict_generator(generator=generator, steps=nb_train_samples //
batch_size)
        # saves an array to a binary file
        np.save(file='models/resnet-ft_tt' + timeStamp +
'_bottleneck_features_train.npy', arr=bottleneck_features_train)

        generator = datagen.flow_from_directory(validation_data_dir,
target_size=(img_width, img_height), batch_size=batch_size,
                                                class_mode=None,
shuffle=False)
        bottleneck_features_validation =
model.predict_generator(generator, nb_validation_samples // batch_size)
        np.save(file='models/resnet-ft_tt' + timeStamp +
'_bottleneck_features_validate.npy', arr=bottleneck_features_validation)

    def train_top_model():
        train_data = np.load(file='models/resnet-ft_tt' + timeStamp +
'_bottleneck_features_train.npy')
        train_labels = np.array([0] * (nb_train_samples // 2) + [1] *
(nb_train_samples // 2)) #np.empty_like(train_data)#

        validation_data = np.load(file='models/resnet-ft_tt' + timeStamp +
'_bottleneck_features_validate.npy')
        validation_labels = np.array([0] * (nb_validation_samples // 2) +
[1] * (nb_validation_samples // 2))#= np.empty_like(validation_data)#

        model = Sequential()
        model.add(Flatten(input_shape=train_data.shape[1:]))  # don't need
to tell batch size in input shape
        model.add(Dense(256, activation='relu'))
        model.add(Dropout(0.5))
        model.add(Dense(1, activation='sigmoid'))

        model.compile(optimizer='rmsprop',
                      loss='binary_crossentropy', metrics=['accuracy'])

        model.fit(train_data, train_labels,
                  epochs=numEpochs,
                  batch_size=batch_size,
                  validation_data=(validation_data, validation_labels))
        model.save_weights('models/resnet-ft_tt' + timeStamp +
'_prefinetuning.h5')


    save_bottleneck_features()
    train_top_model()

    # build the network
```

```
    base_model = applications.ResNet50(weights='imagenet',
include_top=False, input_shape=(224,224,3))

    # build a classifier model to put on top of the convolutional model
    top_model = Sequential()
    top_model.add(Flatten(input_shape=base_model.output_shape[1:]))
    top_model.add(Dense(256, activation='relu'))
    top_model.add(Dropout(0.5))
    top_model.add(Dense(1, activation='sigmoid'))

    # note that it is necessary to start with a fully-trained
    # classifier, including the top classifier,
    # in order to successfully do fine-tuning
    top_model.load_weights('models/resnet-ft_tt' + timeStamp +
'_prefinetuning.h5')

    # add the model on top of the convolutional base
    # model.add(top_model)
    model = Model(inputs=base_model.input,
outputs=top_model(base_model.output))

    model.trainable = True

    set_trainable = False
    for layer in model.layers:
        if layer.name == 'res4a_branch2a':
            set_trainable = True
        if set_trainable:
            layer.trainable = True
       else:
           layer.trainable = False




    # set the first 25 layers (up to the last conv block)
    # to non-trainable (weights will not be updated)


    #for layer in model.layers[:249]:
        #layer.trainable = False

        # compile the model with a SGD/momentum optimizer
    # and a very slow learning rate.
    model.compile(loss='binary_crossentropy',
                  optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
                  metrics=['accuracy'])
    return model



model = createModelResnetFT(img_width, img_height)

# serialize model to YAML
```

```
model_yaml = model.to_yaml()
if not os.path.exists('models/'):
    os.makedirs('models/')
with open('models/' + model_architecture + '_tt' + timeStamp + '.yaml',
'w') as yaml_file:
    yaml_file.write(model_yaml)


# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
    rescale = 1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary', shuffle=False)


history = model.fit_generator(
        train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=numEpochs,
        validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size)

model.save_weights('models/' + model_architecture + '_tt' + timeStamp +
'.h5')
```

Sliding window detection

```
import datetime
import csv
import glob
```

```
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
import sys
import time
from keras.models import model_from_yaml
from PIL import Image, ImageDraw
from scipy import ndimage
from scipy.ndimage.interpolation import zoom
from skimage import measure
from skimage import morphology
from skimage.transform import resize
from skimage.feature import peak_local_max
from skimage.filters import threshold_otsu
from skimage.morphology import watershed

# timestamp as string used to name output files of the current run
timeStamp = datetime.datetime.now().strftime("%y%m%d-%H%M")

def drawMarkerFilled(arr2, posX, posY, size, increment):
    for x in range(size):
        for y in range(size):
            if (posX+x < arr2.shape[0] and posY+y < arr2.shape[1]):
                if (arr2[posX+x,posY+y] < 255-increment):
                    arr2[posX+x,posY+y] += increment


# classify image patch contained in area
# do not call directly, called from classifyImage
def classifyPatch(model, patch, patchScale):
    patch = zoom(patch, (patchScale[0], patchScale[1], 1.0), order=1)
    patch = np.expand_dims(patch, axis=0)
    #plt.imshow(patch)
    #Image.resize(size, resample=0)

    #print(patch.shape)
    #print(str(patchScale[0]) + ";")
    #print(patchScale[1])
    #print(type(patch))
    #patch = resize(patch, (52, 52))
    #patch = np.kron(patch, np.ones((52,52)))
    preds = model.predict(patch, 32, 0)
    return preds[0,0]


# classify entire image by splitting into patches
# do not call directly, called from classifyDirectory
def classifyImage(inputDirectory, resultDirectory, inputImage, myModel,
patchWidth, patchHeight, model_input_width, model_input_height, offset,
csvWriter):
    inputArray = imageio.imread(inputDirectory + '/' + inputImage)
    # if image is gray scale only, convert to RGB
    if (inputArray.ndim == 2):
```

```
        rgbArray = np.zeros((inputArray.shape[0], inputArray.shape[1], 3),
'uint8')
        rgbArray[..., 0] = inputArray[...]
        rgbArray[..., 1] = inputArray[...]
        rgbArray[..., 2] = inputArray[...]
        inputArray = rgbArray

    # scale image wrt magnification level
    # 20, 25 (ref), 30, 40, 50, 60, 80, 100, 120, 150, 200, 250, 300
    resizeFraction = 1.0
    if (inputImage.count('300k') > 0):
        resizeFraction = 25.0/300.0
    elif (inputImage.count('250k') > 0):
        resizeFraction = 25.0/250.0
    elif (inputImage.count('200k') > 0):
        resizeFraction = 25.0/200.0
    elif (inputImage.count('150k') > 0):
        resizeFraction = 25.0/150.0
    elif (inputImage.count('120k') > 0):
        resizeFraction = 25.0/120.0
    elif (inputImage.count('100k') > 0):
        resizeFraction = 25.0/100.0
    elif (inputImage.count('80k') > 0):
        resizeFraction = 25.0/80.0
    elif (inputImage.count('60k') > 0):
        resizeFraction = 25.0/60.0
    elif (inputImage.count('50k') > 0):
        resizeFraction = 25.0/50.0
    elif (inputImage.count('40k') > 0):
        resizeFraction = 25.0/40.0
    elif (inputImage.count('30k') > 0):
        resizeFraction = 25.0/30.0
    elif (inputImage.count('25k') > 0):
        resizeFraction = 25.0/25.0
    elif (inputImage.count('20k') > 0):
        resizeFraction = 25.0/20.0
    else:
        print('WARNING: ' + inputImage + ' has unknown magnification
level!')
    if (resizeFraction != 1.0):
        zoomedArray = zoom(inputArray, (resizeFraction, resizeFraction,
1.0), order=1)
    else:
        zoomedArray = inputArray
    #plt.imshow(zoomedArray)

    # initialize markerArray and mark object location in there
    markerArray = np.empty((zoomedArray.shape[0], zoomedArray.shape[1]))
    markerArray.fill(0)
    patchScale = (model_input_width/patchWidth,
model_input_height/patchHeight)
    for i in range(markerArray.shape[0]):
        #if (i % 100 == 0):
            #sys.stdout.write('.')
```

```
            #sys.stdout.flush()
        for j in range(markerArray.shape[1]):
            if (i%offset == 0 and j%offset == 0):
                if (i+patchWidth < markerArray.shape[0] and j+patchHeight
< markerArray.shape[1]):
                    patch = zoomedArray[i:i+patchWidth, j:j+patchHeight]
                    probability = classifyPatch(myModel, patch,
patchScale)
                    if (probability > 0):
                        sys.stdout.write(str(probability)  + '\n')
                    sys.stdout.flush()

                    if (probability == 1):
                        drawMarkerFilled(markerArray, i, j, 52, 20)


    #PARAMETERS: OFFSET + GAUSS + PEAKMAXIMADIST (=OFFSET?)
    # GAUSSIAN DEPENDENT ON ZOOM?
    #postprocess labels stored in markerArray
    # apply Gaussian convolution
    markerImage = Image.fromarray(markerArray)
    markerArray = ndimage.gaussian_filter(markerImage, 9)
    plt.imshow(markerArray)

    if markerArray.min() != markerArray.max():
        # apply global Otsu threshold
        val = threshold_otsu(markerArray)
        maskArray = markerArray < val

        # apply watershed transform
        maskArray = np.invert(maskArray)
        distanceArray = ndimage.distance_transform_edt(maskArray)
        #plt.imshow(distanceArray)
        localMaxi = peak_local_max(distanceArray, indices=False,
footprint=np.ones((offset, offset)), labels=maskArray)
        markers = morphology.label(localMaxi)
        labelArray = watershed(-distanceArray, markers, mask=maskArray)
        #plt.imshow(labelArray)

        # analyze region properties, annotate images, and write
coordinates to csv file
        markerRadius = 50*round(1.0/resizeFraction);
        properties = measure.regionprops(labelArray)
        sys.stdout.write(' (' + str(len(properties)) + ' objects
detected)\n')
        sys.stdout.flush()
        resultImage = Image.fromarray(inputArray)#.convert('RGB')
        draw = ImageDraw.Draw(resultImage)
        for property in properties:
            centroidZoomed = property.centroid
            centroid = (centroidZoomed[0] * round(1.0/resizeFraction),
centroidZoomed[1] * round(1.0/resizeFraction))
            csvWriter.writerow([inputImage, str(centroid[0]),
str(centroid[1])])
```

```
            # draw circle
            draw.ellipse((centroid[1]-markerRadius+2, centroid[0]-
markerRadius+2, centroid[1]+markerRadius+2, centroid[0]+markerRadius+2),
outline=128)
            draw.ellipse((centroid[1]-markerRadius+1, centroid[0]-
markerRadius+1, centroid[1]+markerRadius+3, centroid[0]+markerRadius+3),
outline=128)
            draw.ellipse((centroid[1]-markerRadius, centroid[0]-
markerRadius, centroid[1]+markerRadius, centroid[0]+markerRadius),
outline=255)
            draw.ellipse((centroid[1]-markerRadius-1, centroid[0]-
markerRadius-1, centroid[1]+markerRadius+1, centroid[0]+markerRadius+1),
outline=255)
        del draw

        #resultImage.show()
        resultImage.save(resultDirectory + '/' + inputImage[:-4]+'_m.png')
    else:
        sys.stdout.write(' (0 objects detected)\n')
        sys.stdout.flush()


# classify all images in the provided directory
def classifyDirectory(inputDirectory, modelFileName, patchWidth,
patchHeight, model_input_width, model_input_height, offset):
    # load model architecture and weights
    yamlFile = open(modelFileName + '.yaml', 'r')
    yamlModel = yamlFile.read()
    yamlFile.close()
    myModel = model_from_yaml(yamlModel)
    myModel.load_weights(modelFileName + ".h5")
    myModel.compile(loss='binary_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
    print("Pre-trained model sucessfully loaded.")

    scriptDirectory = os.getcwd()
    os.chdir(inputDirectory)
    fileList = glob.glob('*.png')
    os.chdir("../")
    modelName = modelFileName[modelFileName.rfind('/')+1:]
    resultDirectory = 'automatic-annotations_' + modelName + '_ct' +
timeStamp
    if not os.path.exists(resultDirectory):
        os.mkdir(resultDirectory)
    os.chdir(resultDirectory)
    resultDirectory = os.getcwd()
    with open('centroid-list.csv', 'w', newline='') as csvfile:
        csvWriter = csv.writer(csvfile, delimiter=';', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
        curImage = 1;
        start = time.clock()
        os.chdir(scriptDirectory)
        print(os.getcwd())
        for file in fileList:
```

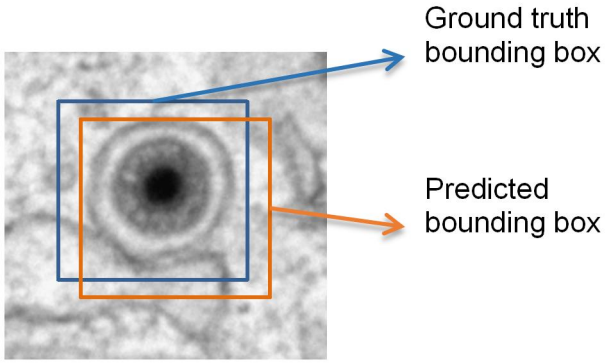
```
            sys.stdout.write('Processing image ' + str(curImage) + ' from
' + str(len(fileList)) + ' (' + file + ')')
            sys.stdout.flush()
            classifyImage(inputDirectory, resultDirectory, file, myModel,
patchWidth, patchHeight, model_input_width, model_input_height, offset,
csvWriter)
            curImage = curImage + 1
        duration = time.clock() - start;
        print('Finished in ' + str(round(duration)) + ' seconds (' +
str(round(duration/len(fileList))) + ' per image)')


#classifyDirectory('data/annotated-images_1708/original-images_png',
'models/convnet_tt180114-1805', 52, 52, 52, 52, 10)
#classifyDirectory('data/annotated-images_1708/original-images_png',
'models/resnet_tt180114-1831', 52, 52, 224, 224, 10)
#classifyDirectory('data/annotated-images_1708/original-images_png',
'models/vgg16_tt180114-2023', 52, 52, 150, 150, 10)
#classifyDirectory('data/annotated-images_1708/original-images_png',
'models/inceptionv3_tt180114-1939', 52, 52, 299, 299, 10)


classifyDirectory('data/test_images', 'models/vgg16-ft_tt180723-1020', 52,
52, 150, 150, 20)

#classifyDirectory('data/test_images', 'models/classification/resnet-
FT_tt180316-1257', 52, 52, 224, 224, 20)
##classifyDirectory('data/annotated-images_1705/original-images_png',
'models/resnet_tt180114-1831', 52, 52, 224, 224, 10)
#classifyDirectory('data/annotated-images_1705/original-images_png',
'models/vgg16_tt180114-2023', 52, 52, 150, 150, 10)
#classifyDirectory('data/annotated-images_1705/original-images_png',
'models/inceptionv3_tt180114-1939', 52, 52, 299, 299, 10)
```

Supplementary Materials: Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network [SD-2] (Chapter 6)

Shaga Devan K, Walther P, von Einem J, Ropinski T, Kestler HA, Read C. Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network. Cell Microbiol. 2020 Feb;23(2):e13280. https://doi.org/10.1111/cmi.13280. Epub 2020 Nov 16. PMID: 33073426.

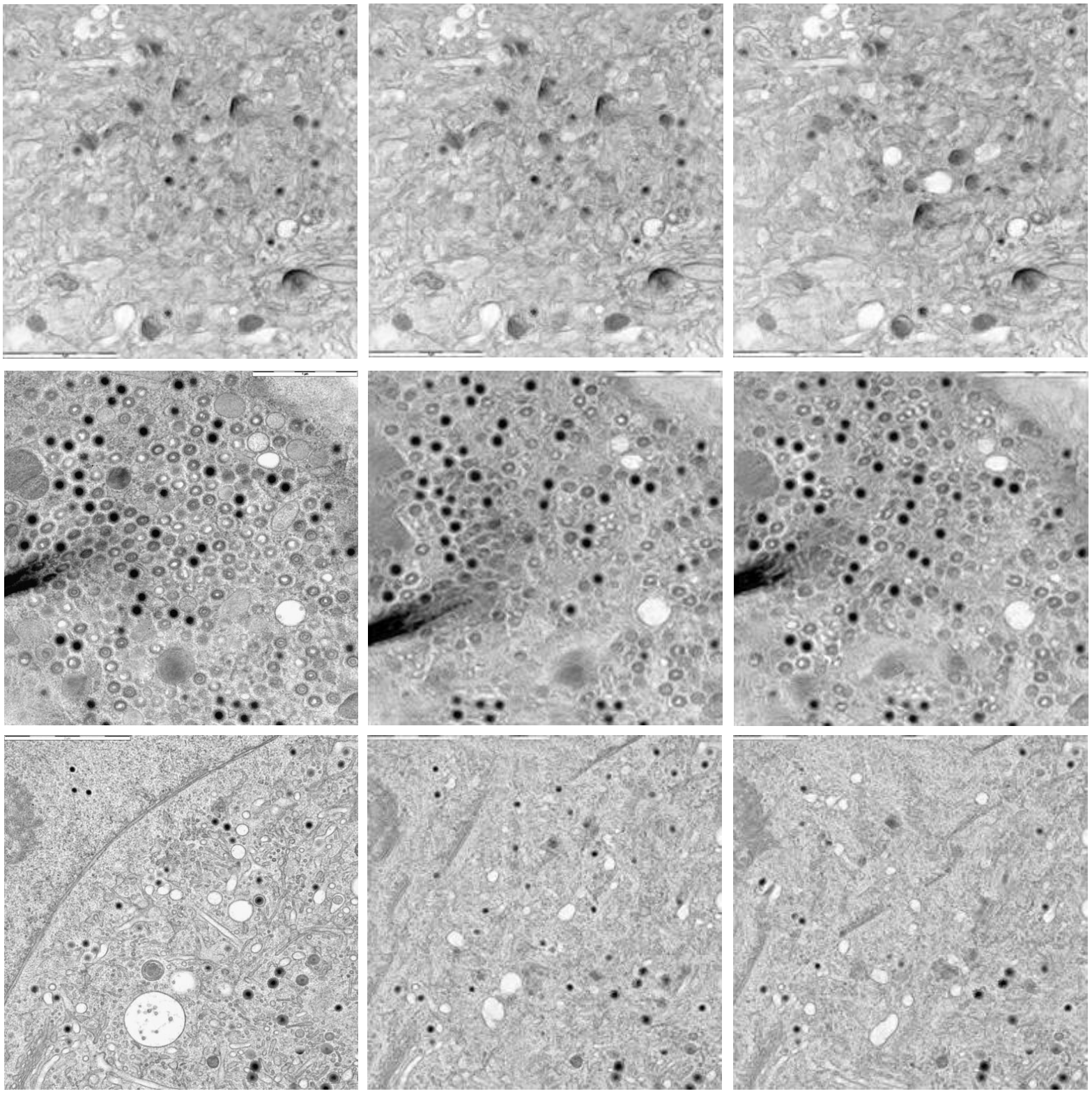
Supplementary Figure S1

This figure shows a capsid with both the ground truth bounding box (blue) and the automatic predicted bounding box (orange). The overlap of these two boxes represent the Interval over Union (IoU) overlap.

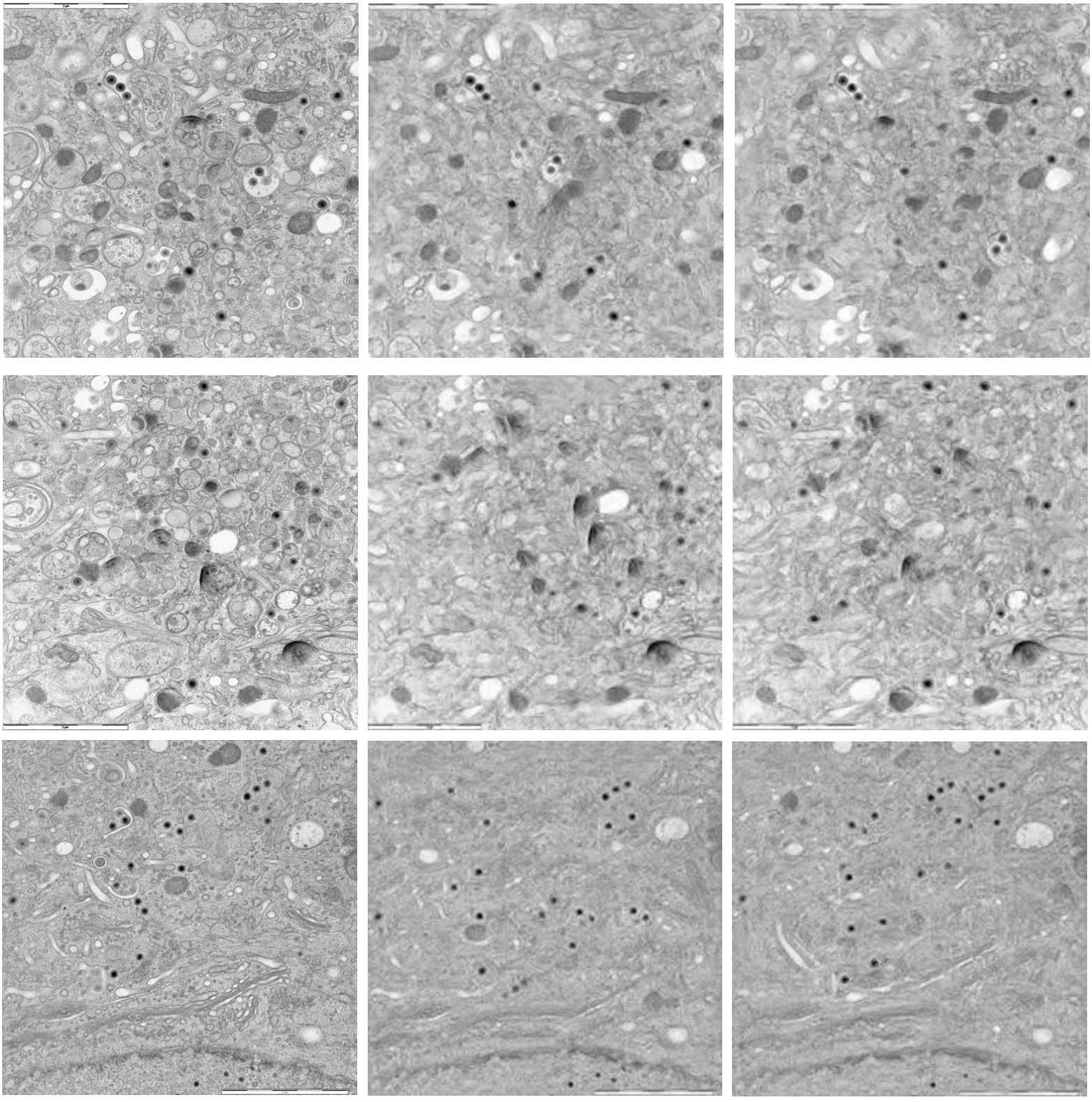


Supplementary Figure S2

Sample of synthetic generated images used as part of the training dataset for the Faster R-CNN model. Column A shows the original ground truth images. Column B and C shows the corresponding synthetic images at different scales.



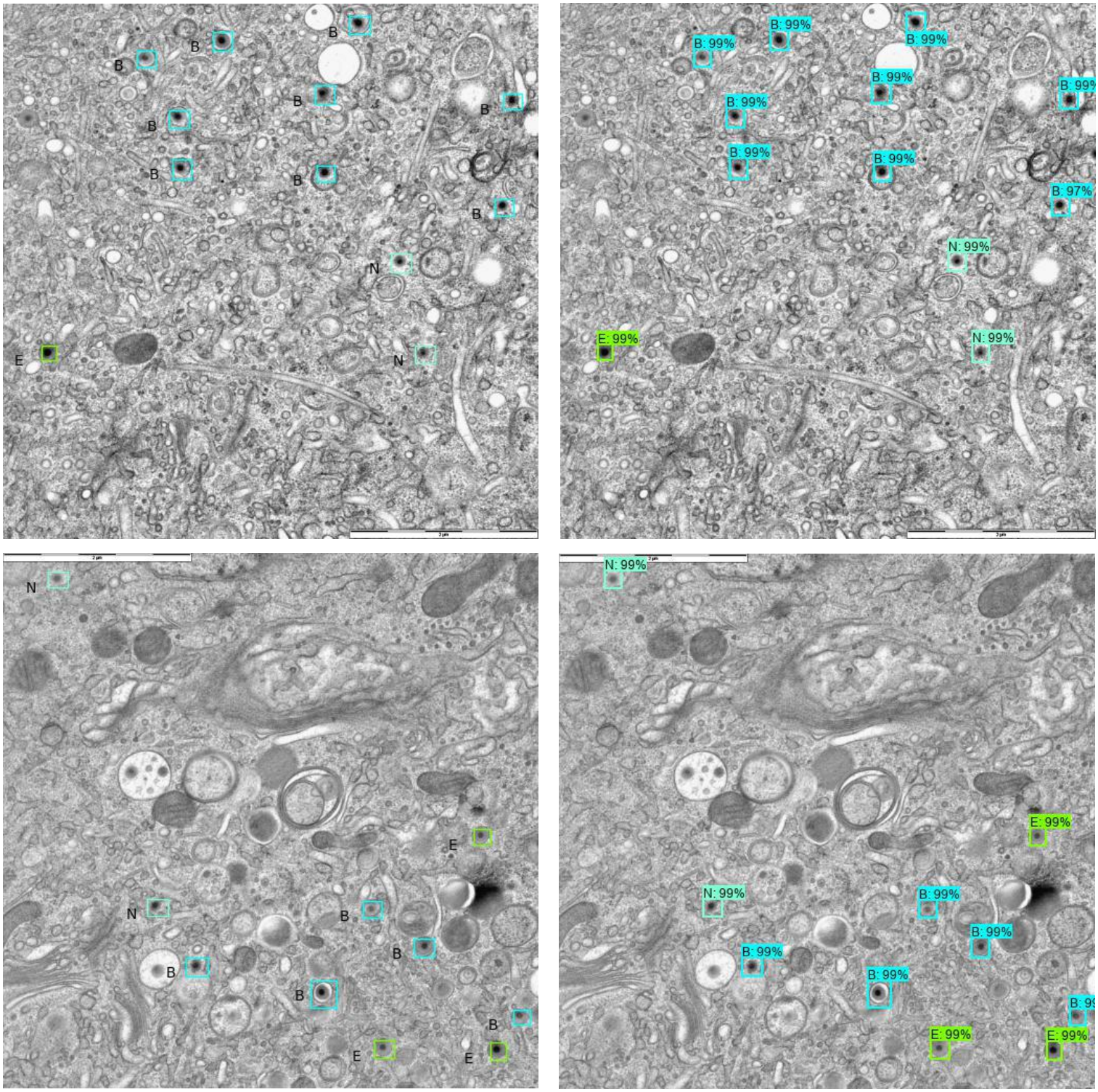
A B C

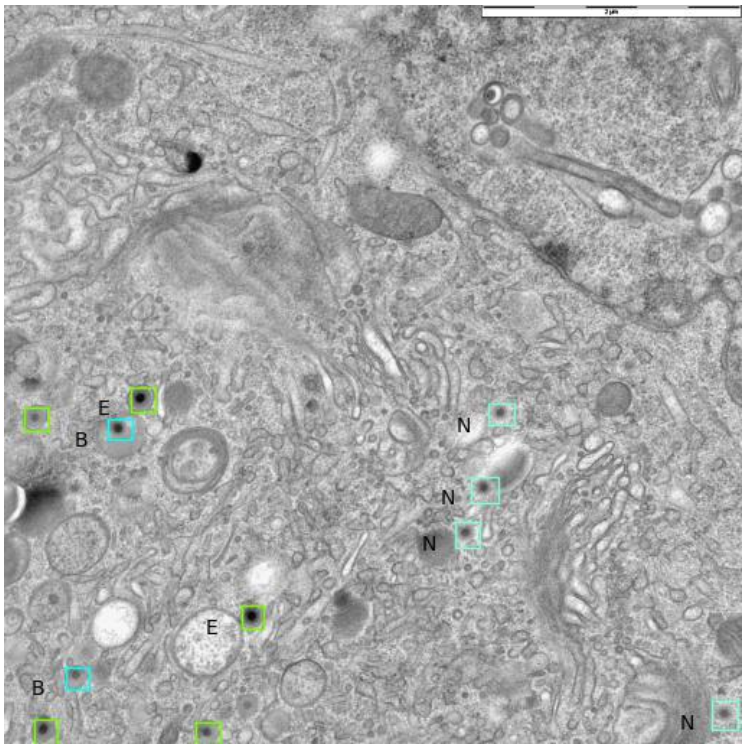


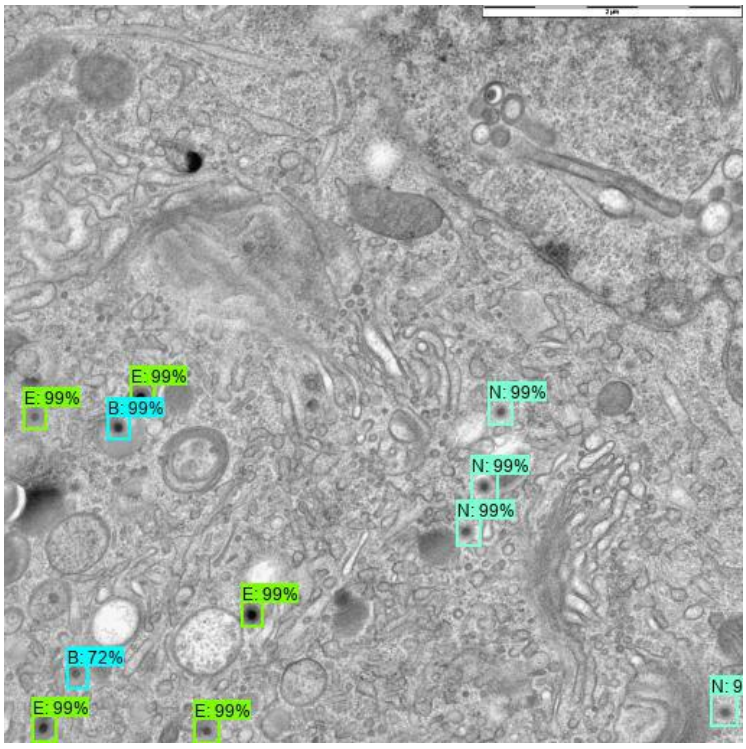
A B C

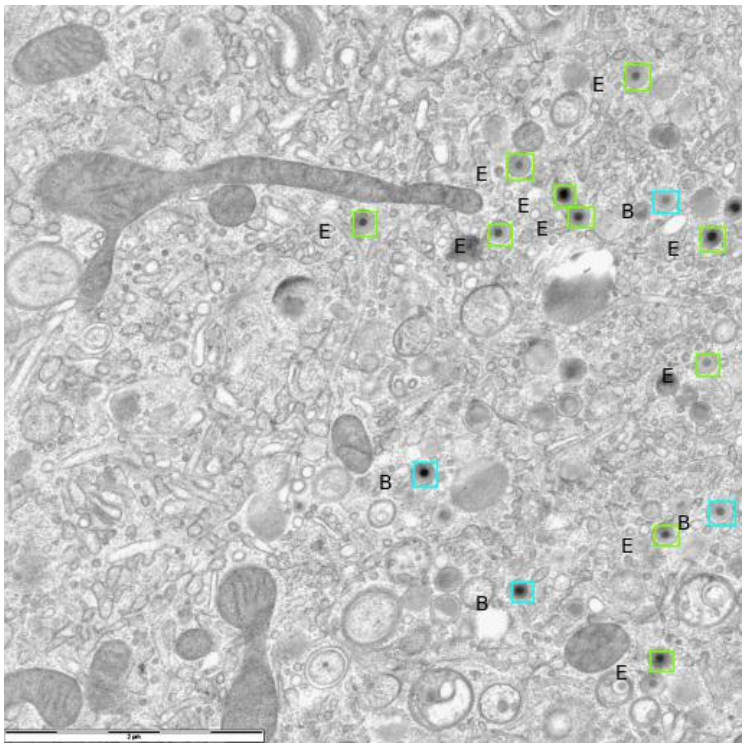
Supplementary Figure S3

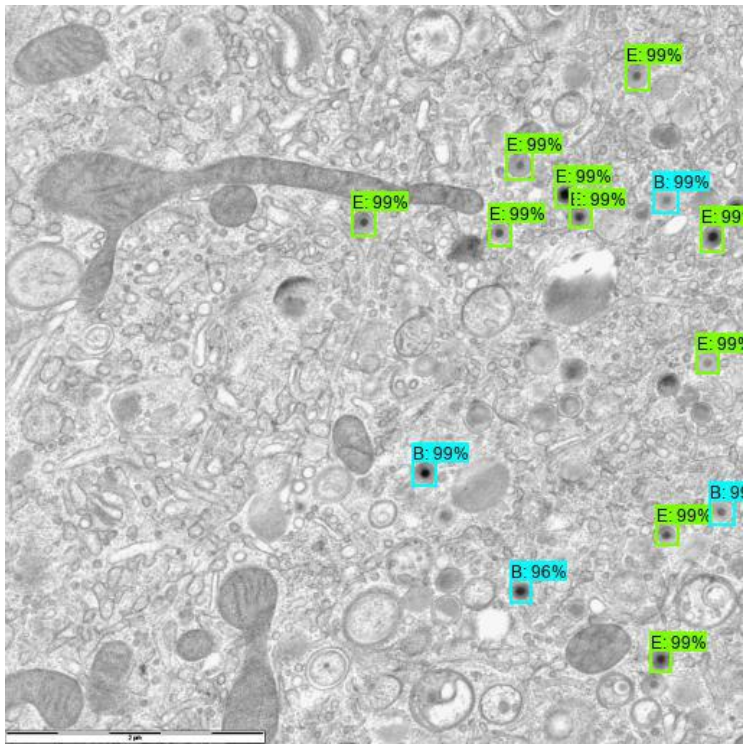
The left column shows expert labeled ground truth images and the right column shows automatically detected HCMV capsids with confidence scores in the validation images. N-Naked, B-Budding and E-Enveloped.

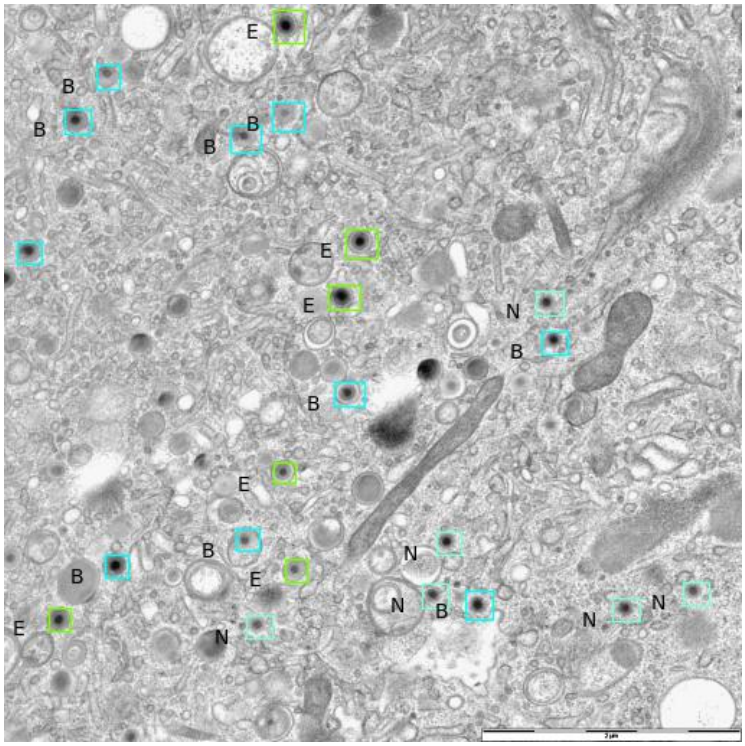


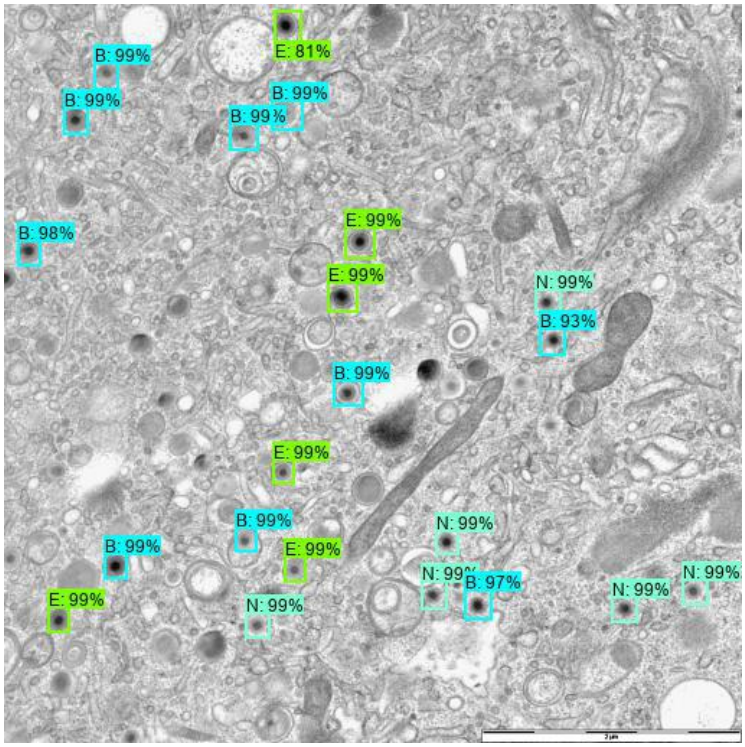

E
B
N
N
N
E
B
N

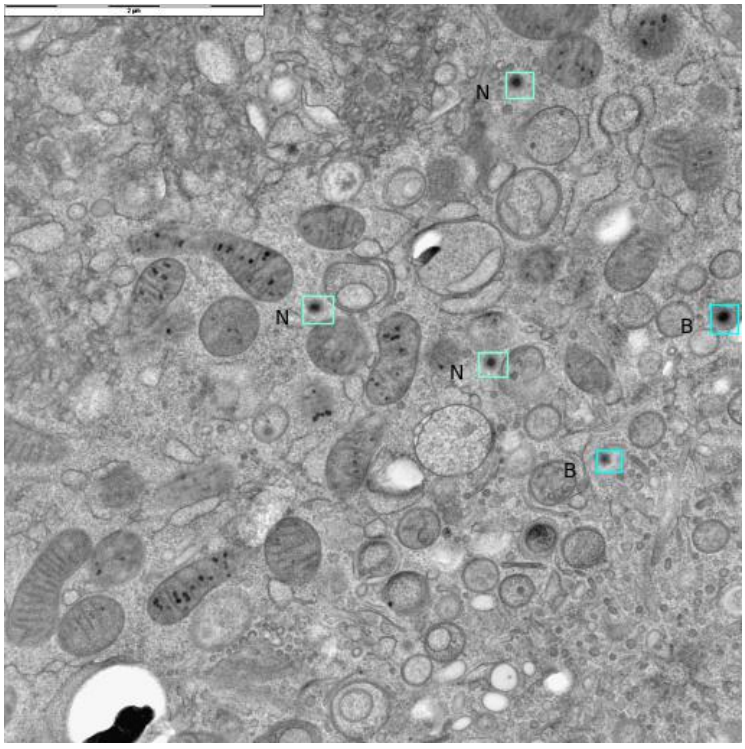

E: 99%
E: 99%
B: 99%
N: 99%
N: 99%
N: 99%
E: 99%
B: 72%
E: 99%
E: 99%

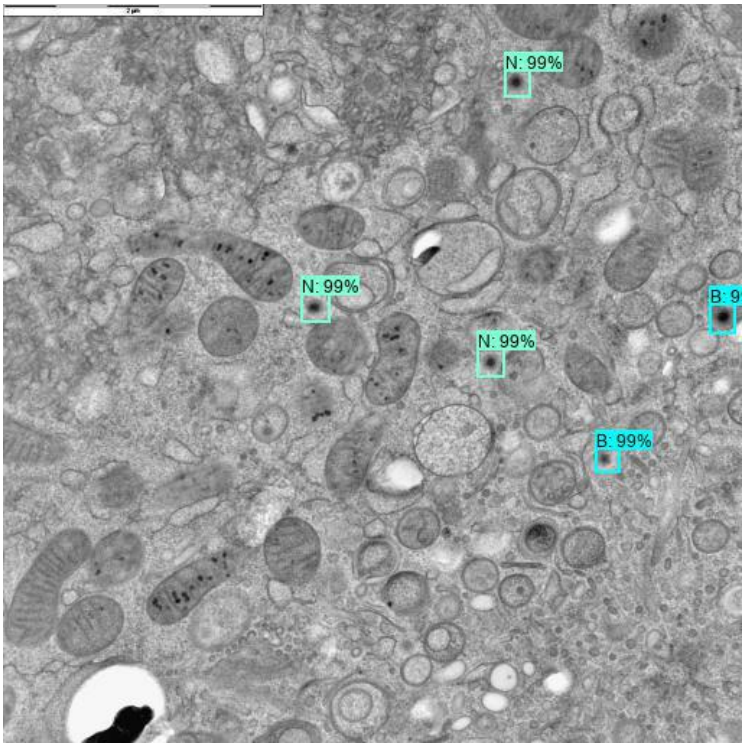

E
E
E
E
E
B
E
E
E
B
E
B
B
E

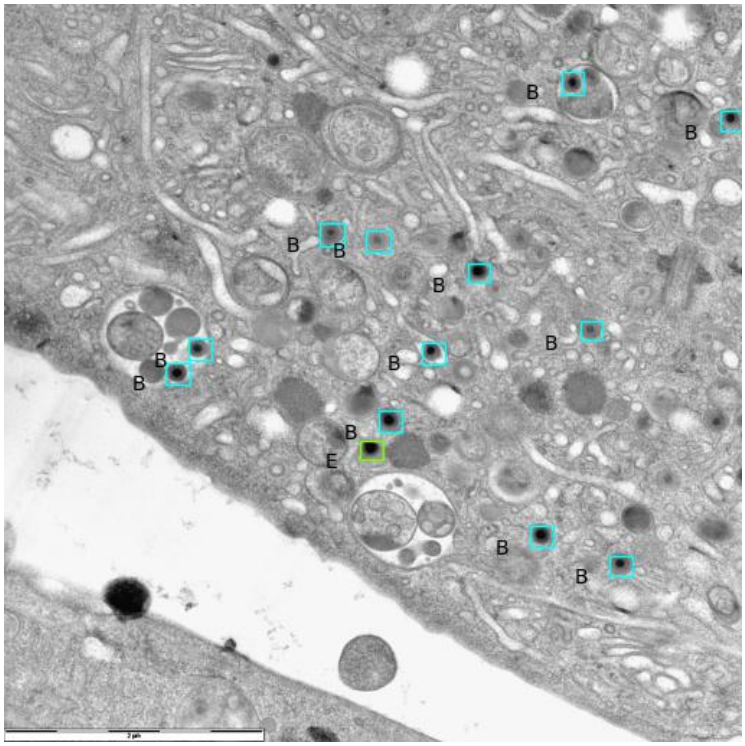

E: 99%
E: 99%
E: 99%
E: 99%
E: 99%
B: 99%
E: 99%
E: 99%
B: 99%
E: 99%
B: 96%
E: 99%

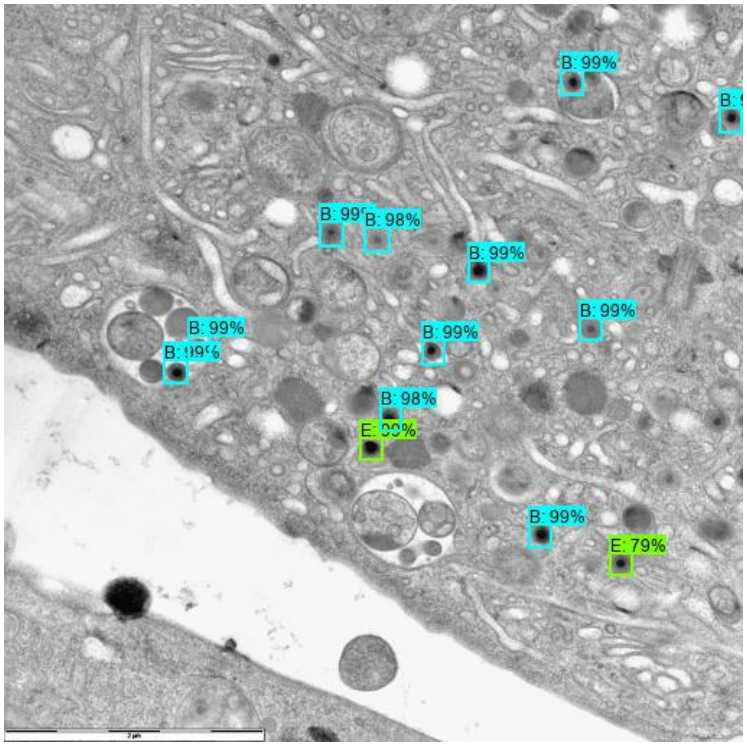

E
B
B
B
B
E
E
N
B
B
E
B
E
B
E
N
N
B
N
N
N

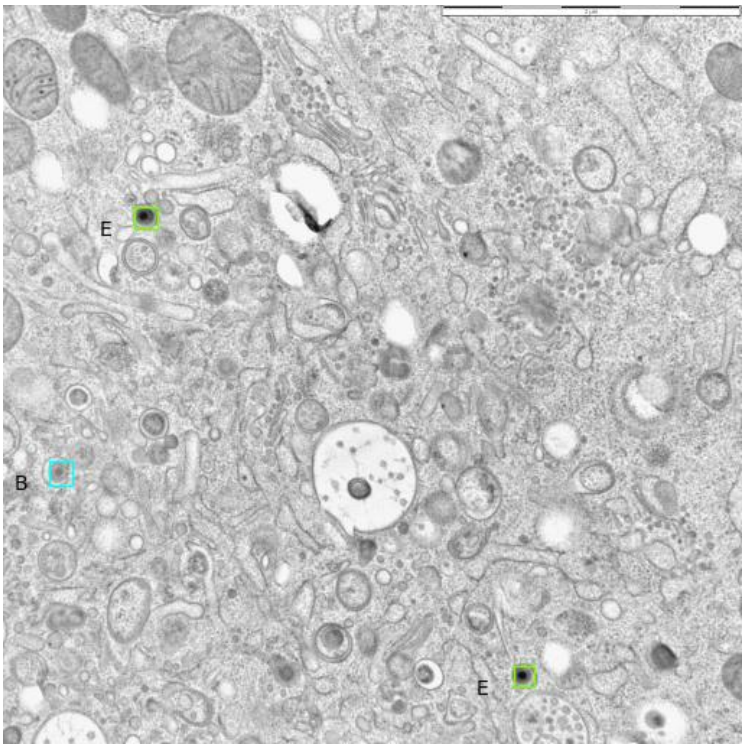

B: 99%
E: 81%
B: 99%
B: 99%
B: 99%
B: 98%
E: 99%
E: 99%
N: 99%
B: 93%
B: 99%
E: 99%
B: 99%
N: 99%
B: 99%
E: 99%
N: 99%
B: 97%
N: 99%
N: 99%
E: 99%
N: 99%

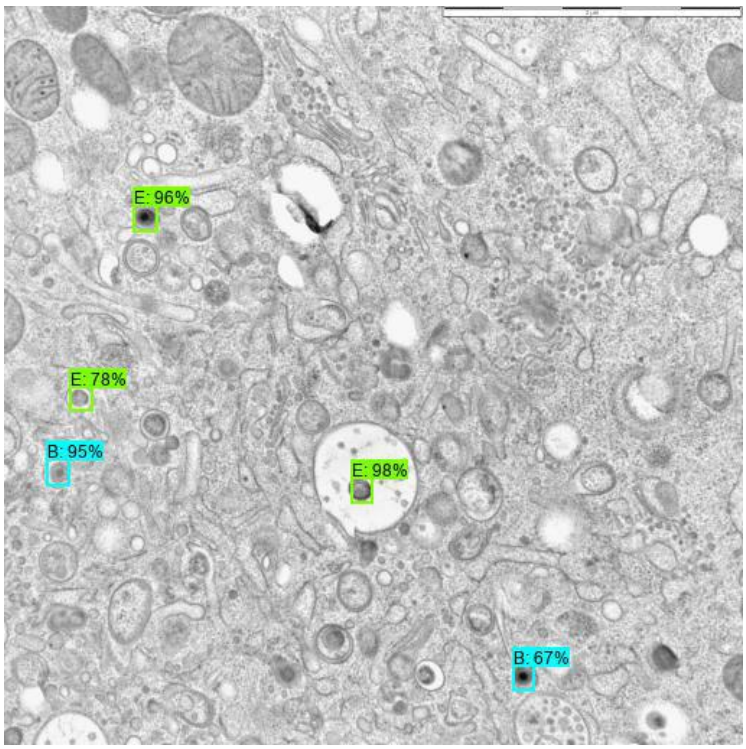

N
N
B
N
B

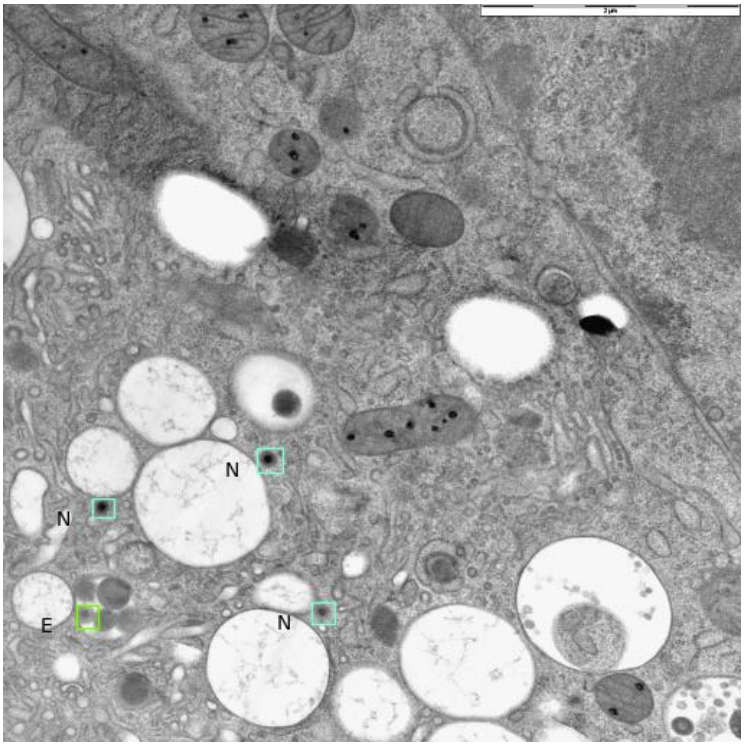

N: 99%
N: 99%
N: 99%
B: 99%

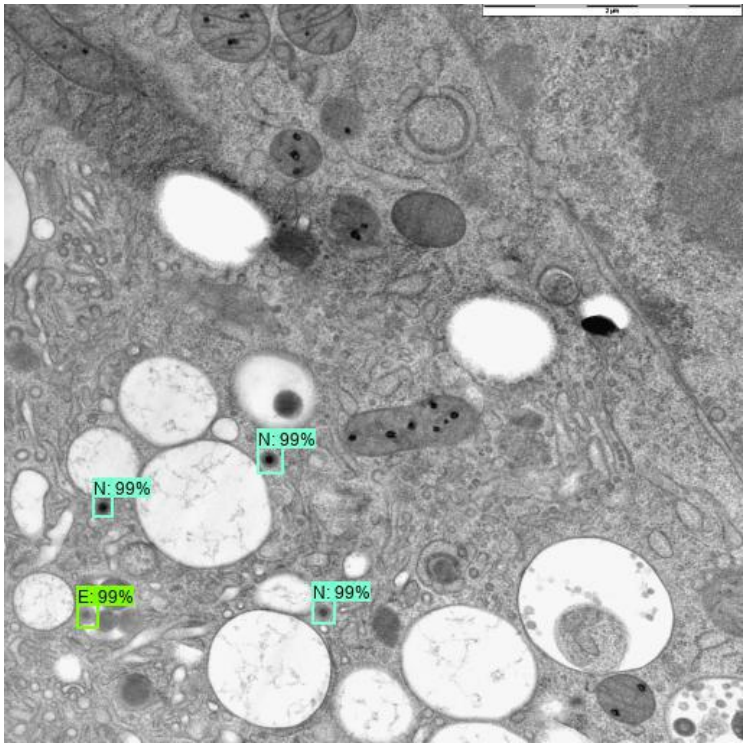

B
B
B
B
B
B
B
B
B
B
B
E
B
B

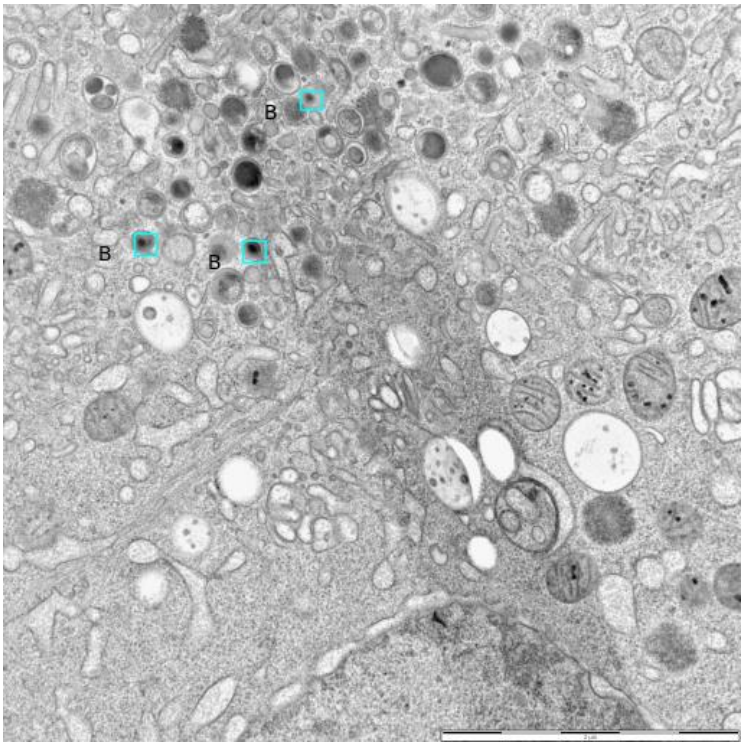

B: 99%
B: 98%
B: 99%
B: 99%
B: 99%
B: 99%
B: 98%
B: 99%
E: 79%

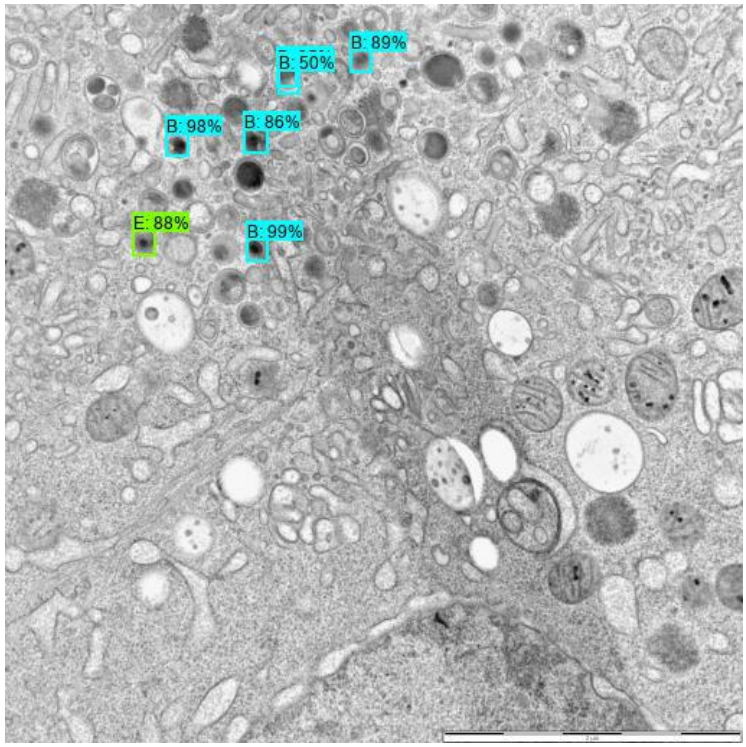

E
B
E

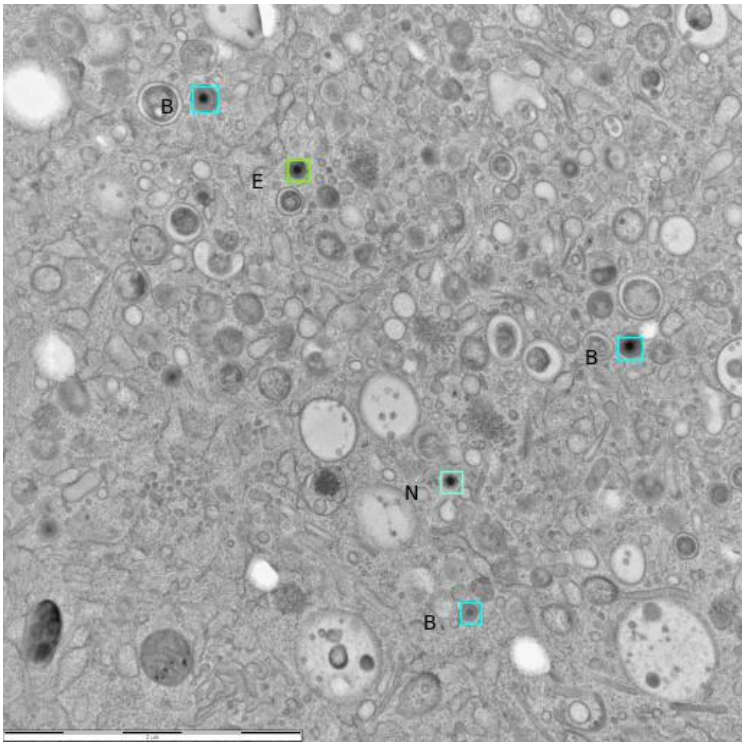

E: 96%
E: 78%
B: 95%
E: 98%
B: 67%

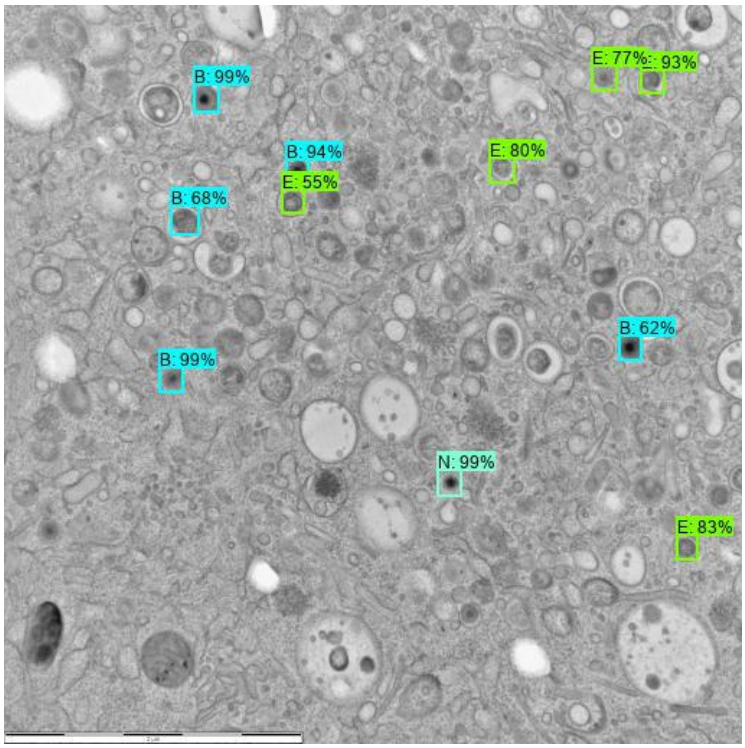

N
N
E
N

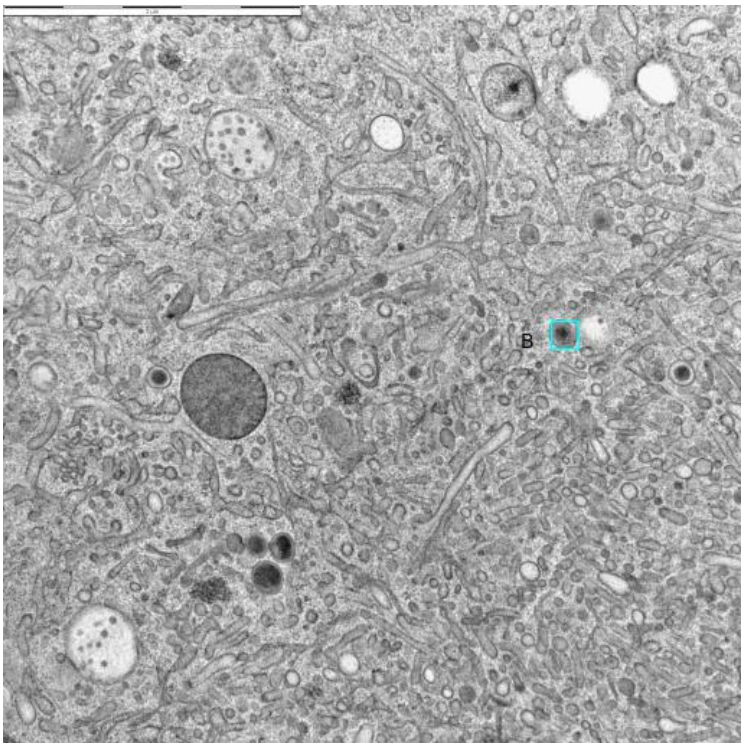

N: 99%
N: 99%
E: 99%
N: 99%

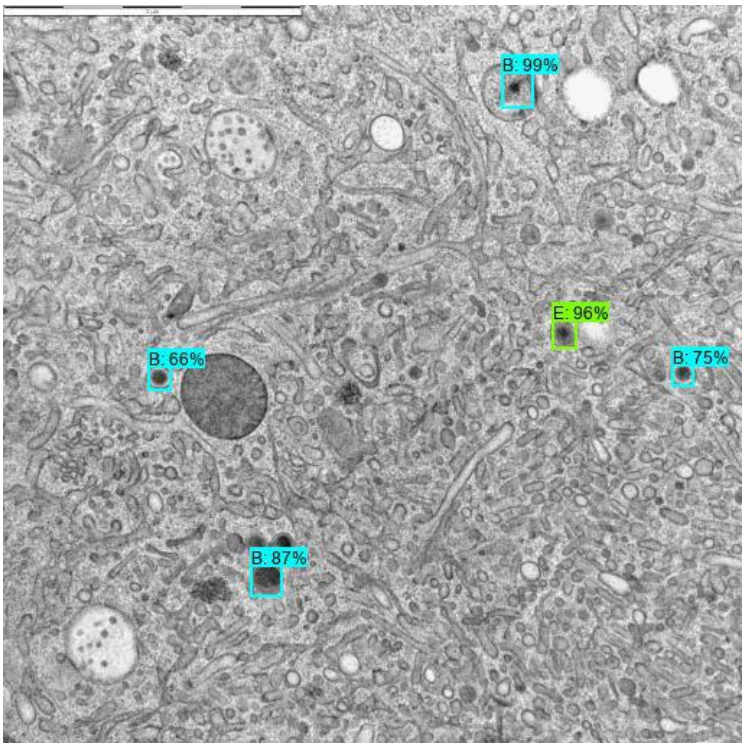

B
B
B


B: 89%
B: 50%
B: 98%
B: 86%
E: 88%
B: 99%


B
E
B
N
B

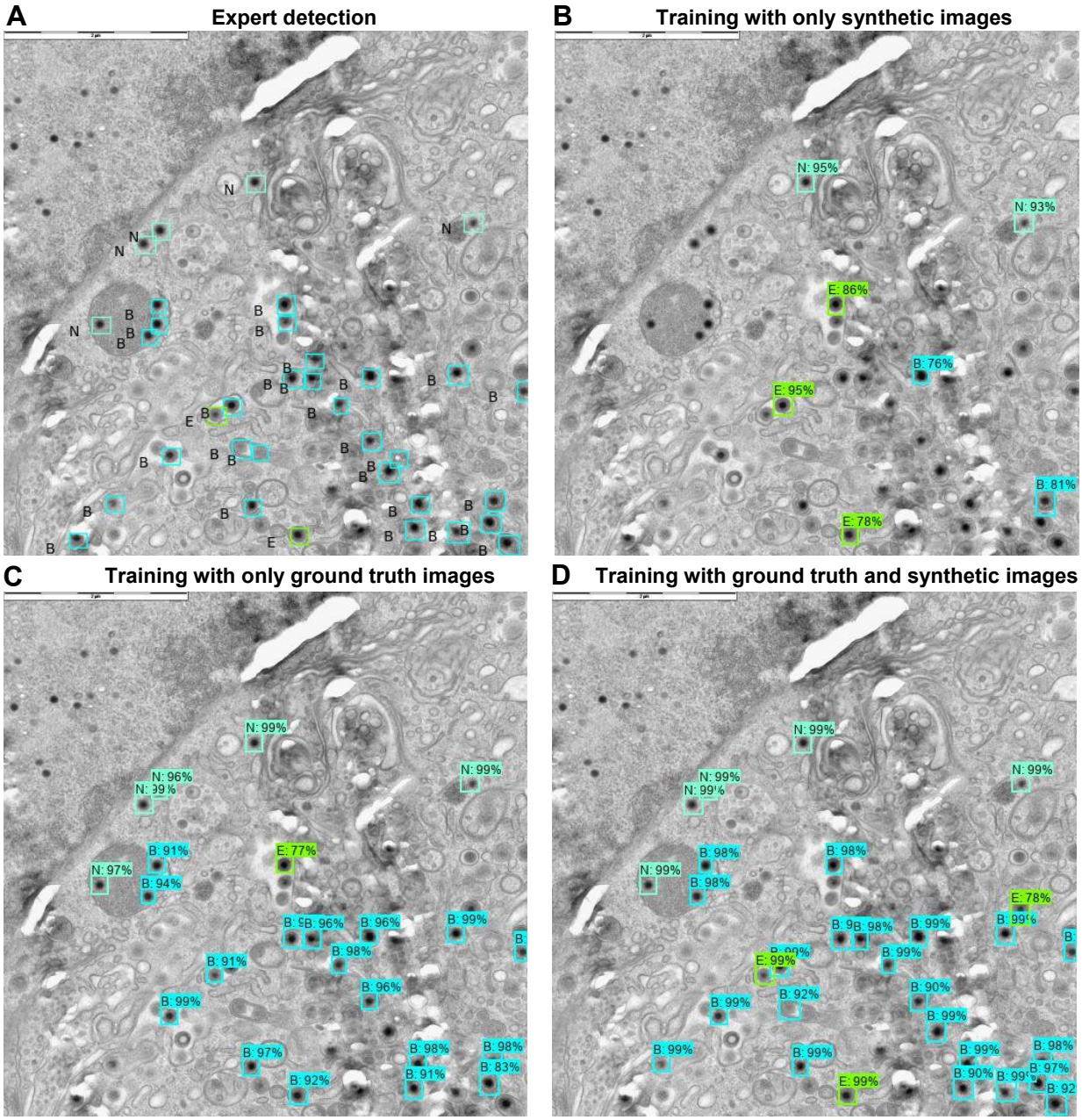

B: 99%
E: 77%
E: 93%
B: 94%
E: 80%
E: 55%
B: 68%
B: 62%
B: 99%
N: 99%
E: 83%


B


B: 99%
E: 96%
B: 66%
B: 75%
B: 87%

Supplementary Figure S4

This figure is supplementary to Figure 6. Validation image after (A) manual and (B-D) automatic capsid detection for comparison of detection efficiencies. Confidence scores in %. Manual detection was performed by a virologist, automatic detection was performed by Faster R-CNN model trained with different training datasets (B-D): (B) Model trained with only synthetic images, (C) model trained with only ground truth image and (D) model trained with both, ground truth and synthetic images. E: enveloped, B: budding, N: naked.

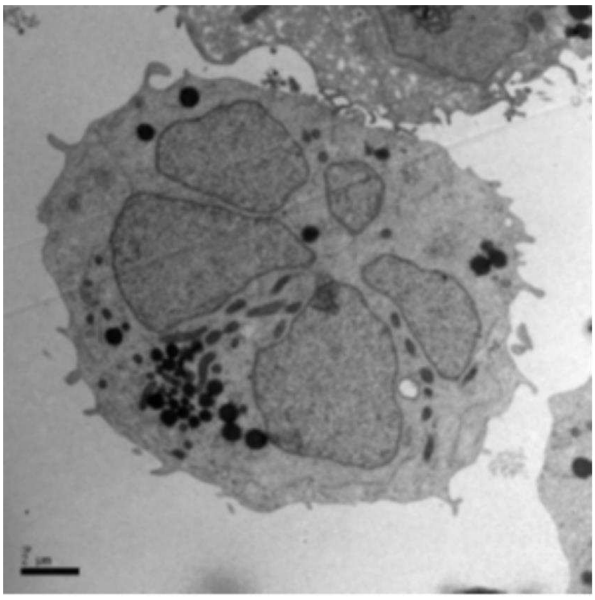


Supplementary Materials: Weighted average ensemble-based semantic segmentation in biological electron microscopy images [SD-3] (Chapter 7)

Shaga Devan K, Kestler HA, Read C, Walther P. Weighted average ensemble-based semantic segmentation in biological electron microscopy images. Histochem Cell Biol. 2022. 158:447–462. https://doi.org/10.1007/s00418-022-02148-3. PMID: 35988009

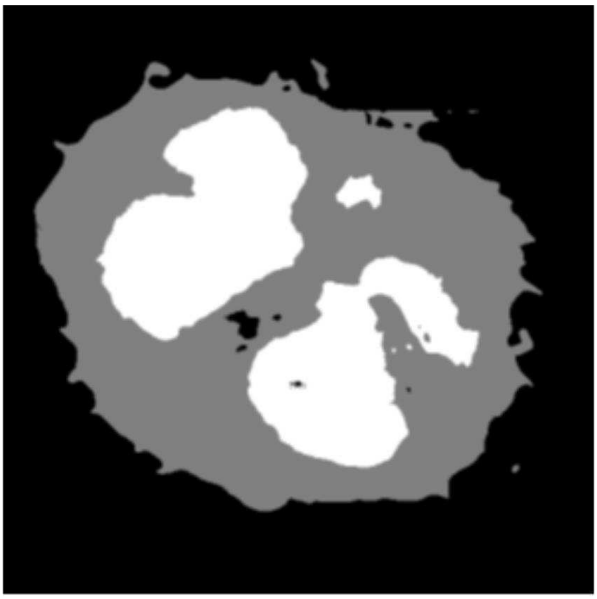
Supplementary Figure S1

Representative test image from dataset 4. Bar 2µm



Input image

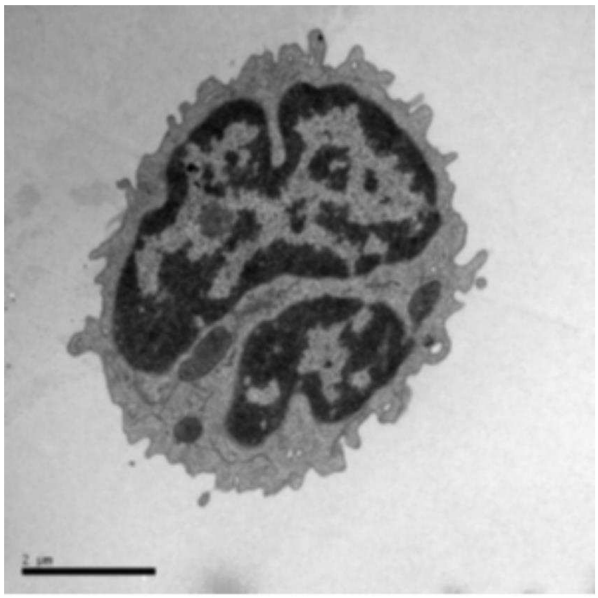
Ground truth



U-Net segmentation

WAE-Net segmentation

Representative test image from dataset 5. Bar 2μm



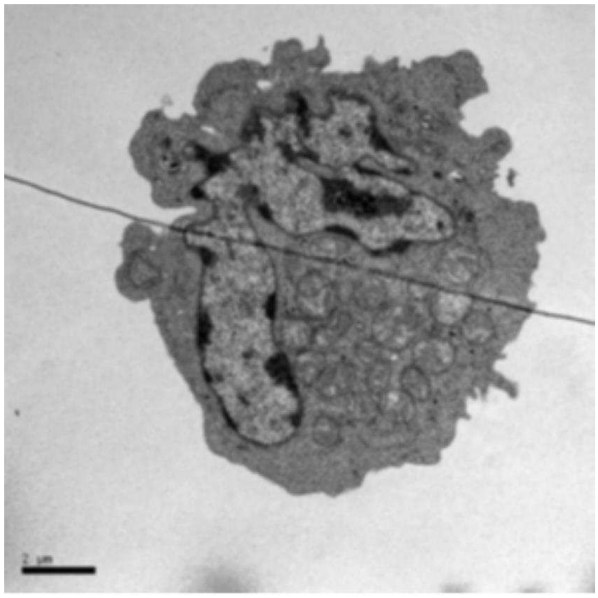
Input image

Ground truth

U-Net segmentation

WAE-Net segmentation

Representative test image from dataset 6. Bar 2µm



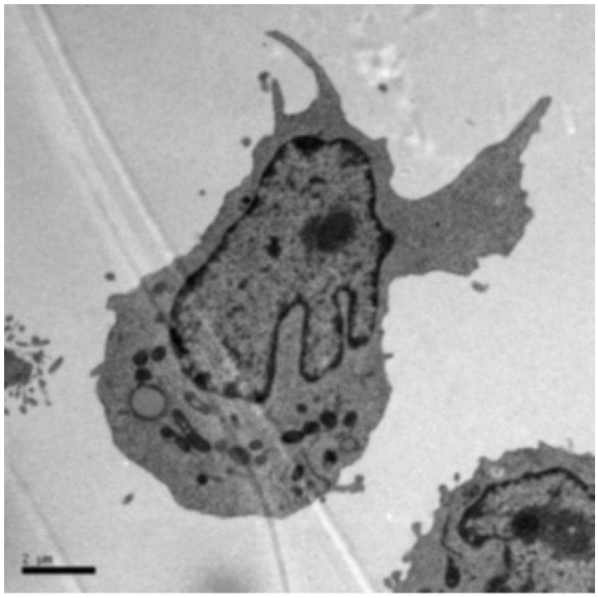
Input image

Ground truth

U-Net segmentation

WAE-Net segmentation

Representative test image from dataset 7. Bar 2μm



Input image

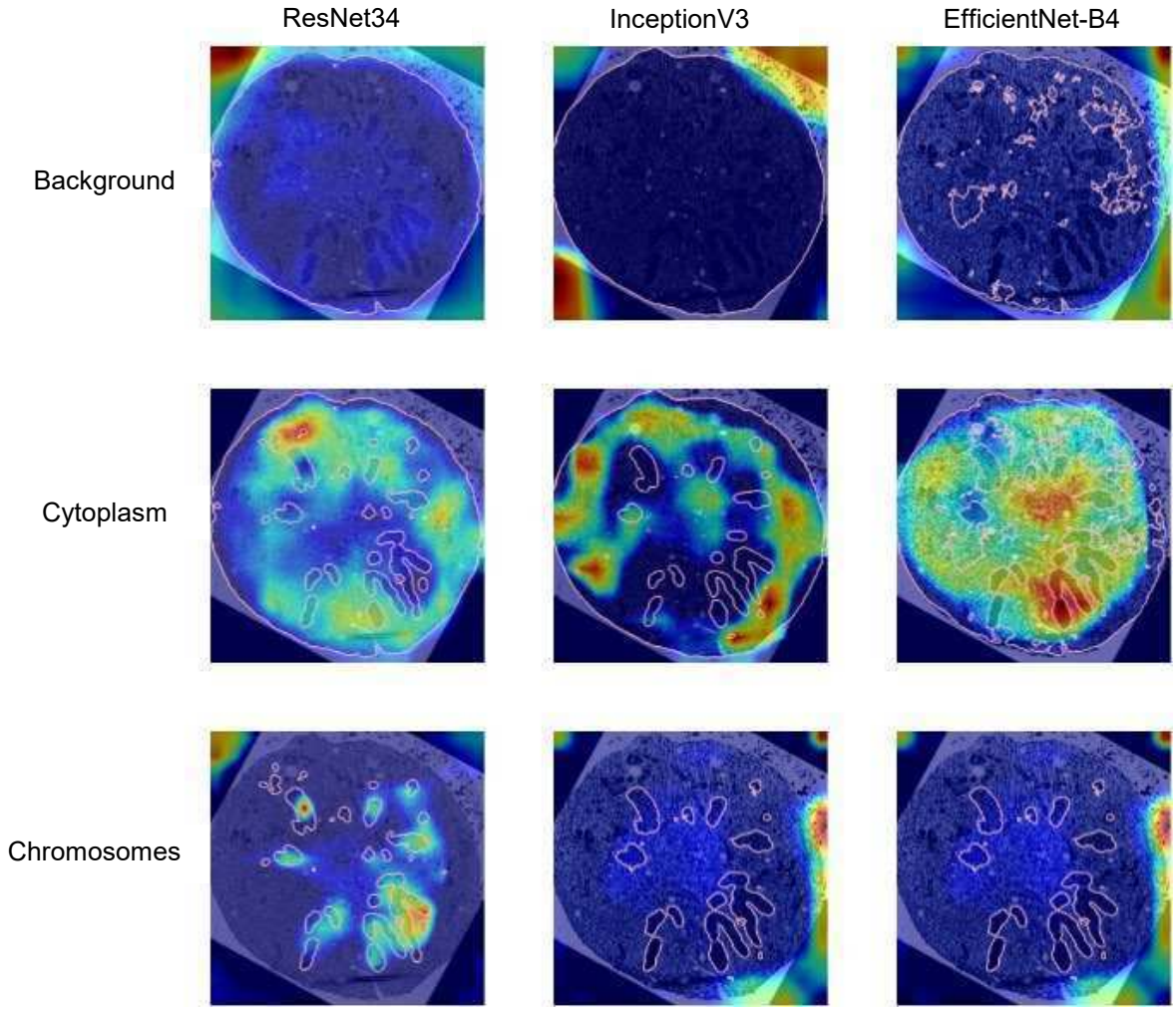
Ground truth

U-Net segmentation

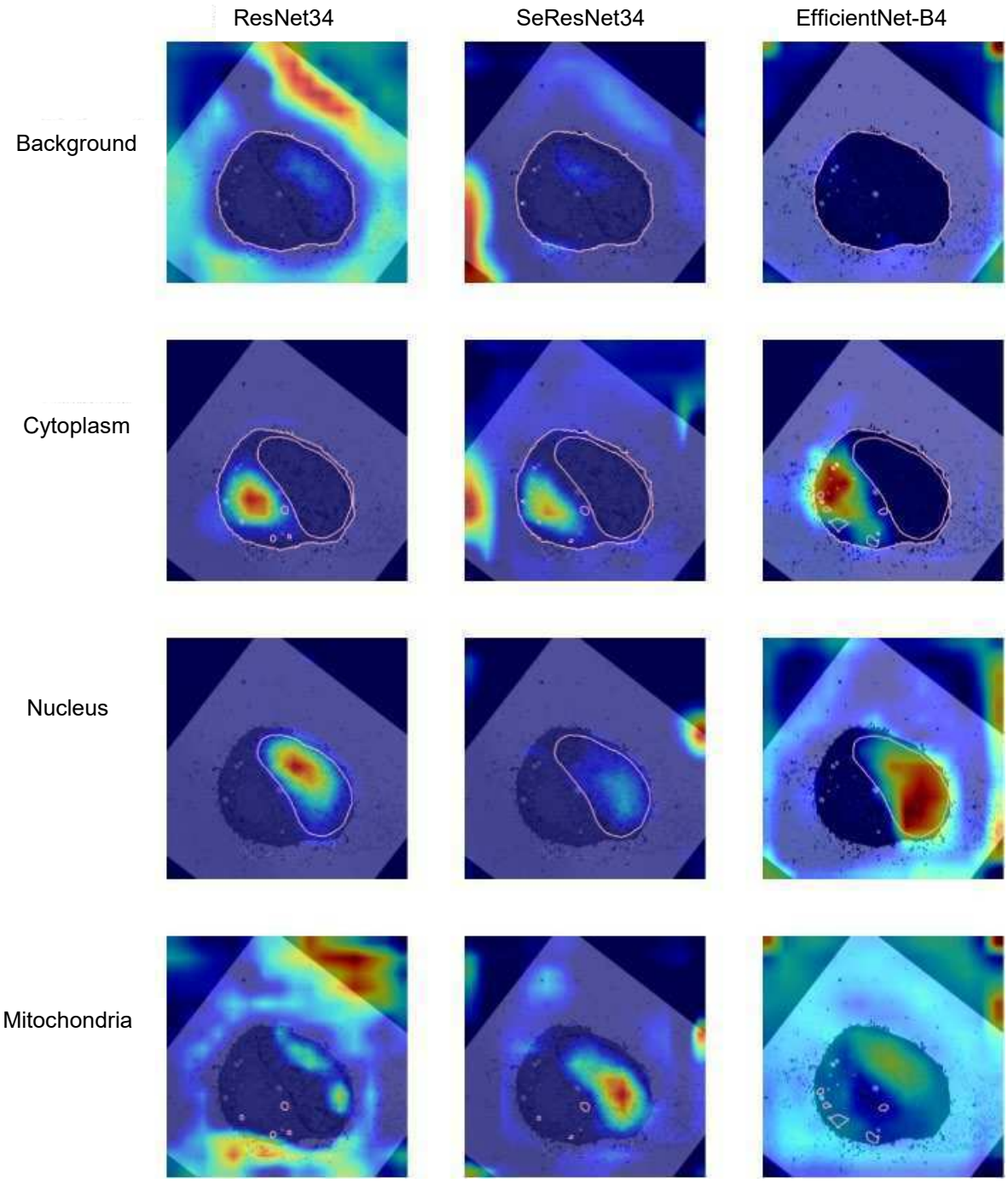
WAE-Net segmentation

Supplementary Figure S2

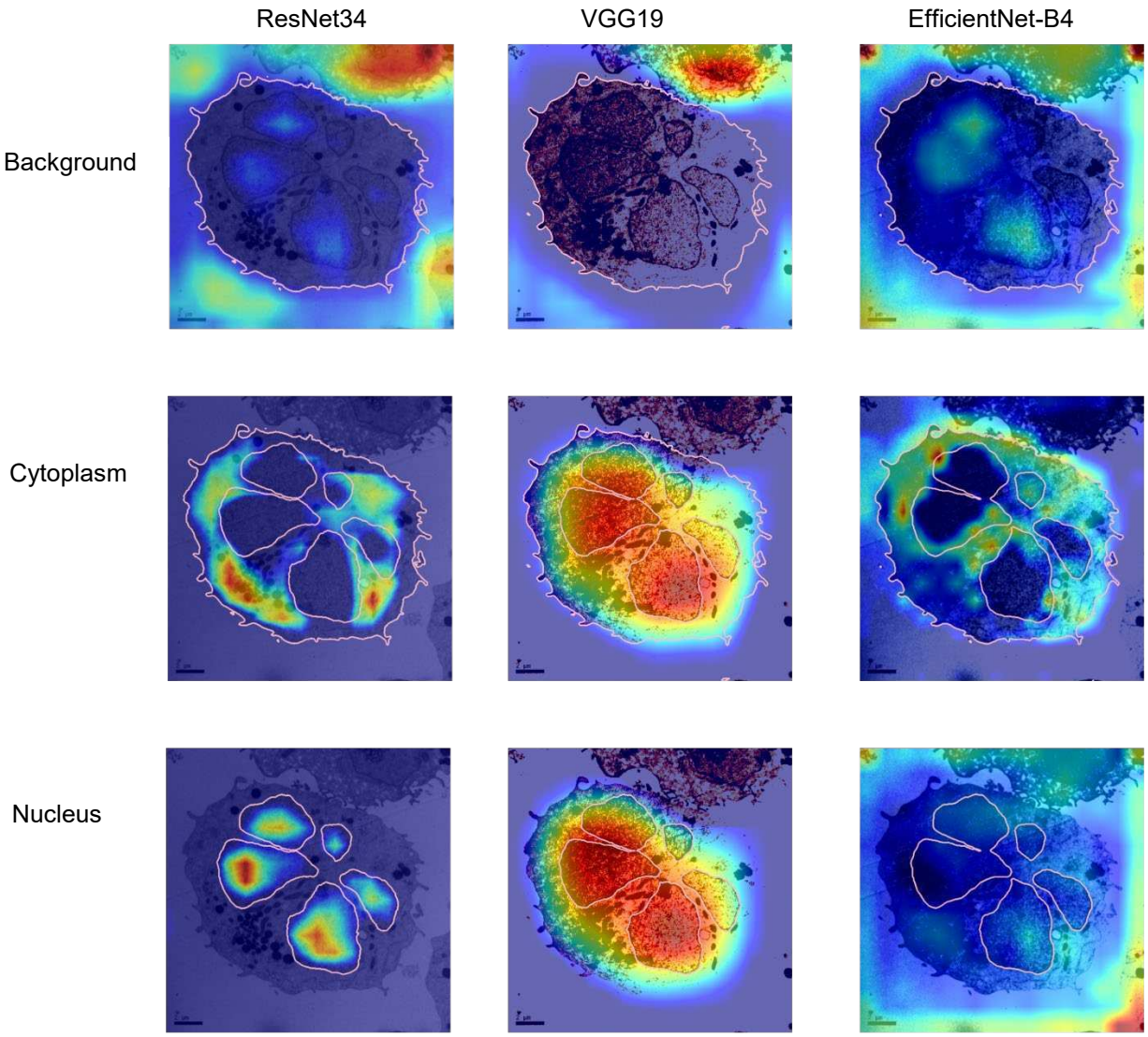
Representative Grad-CAM images from Dataset 2



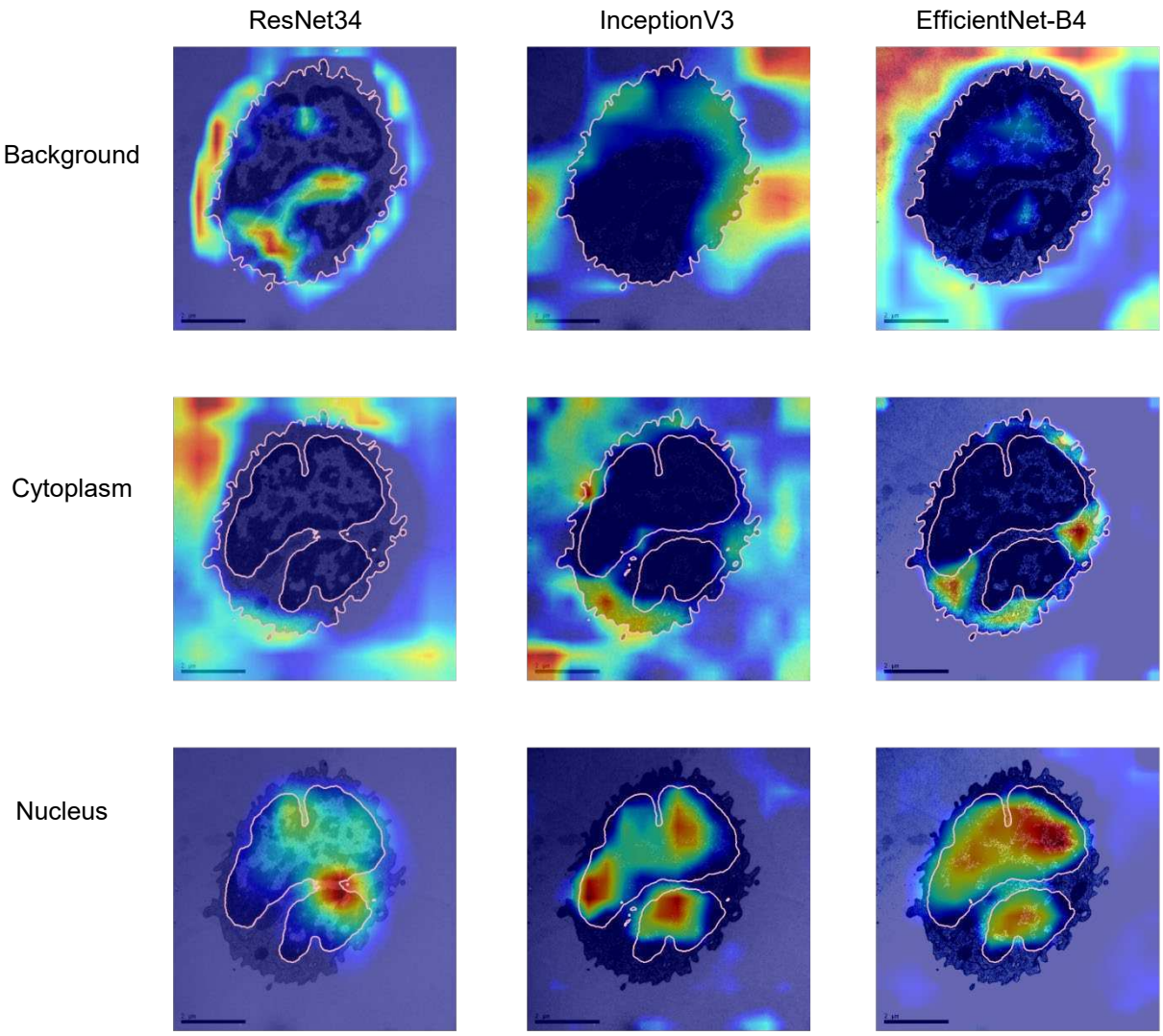
Representative Grad-CAM images from Dataset 3



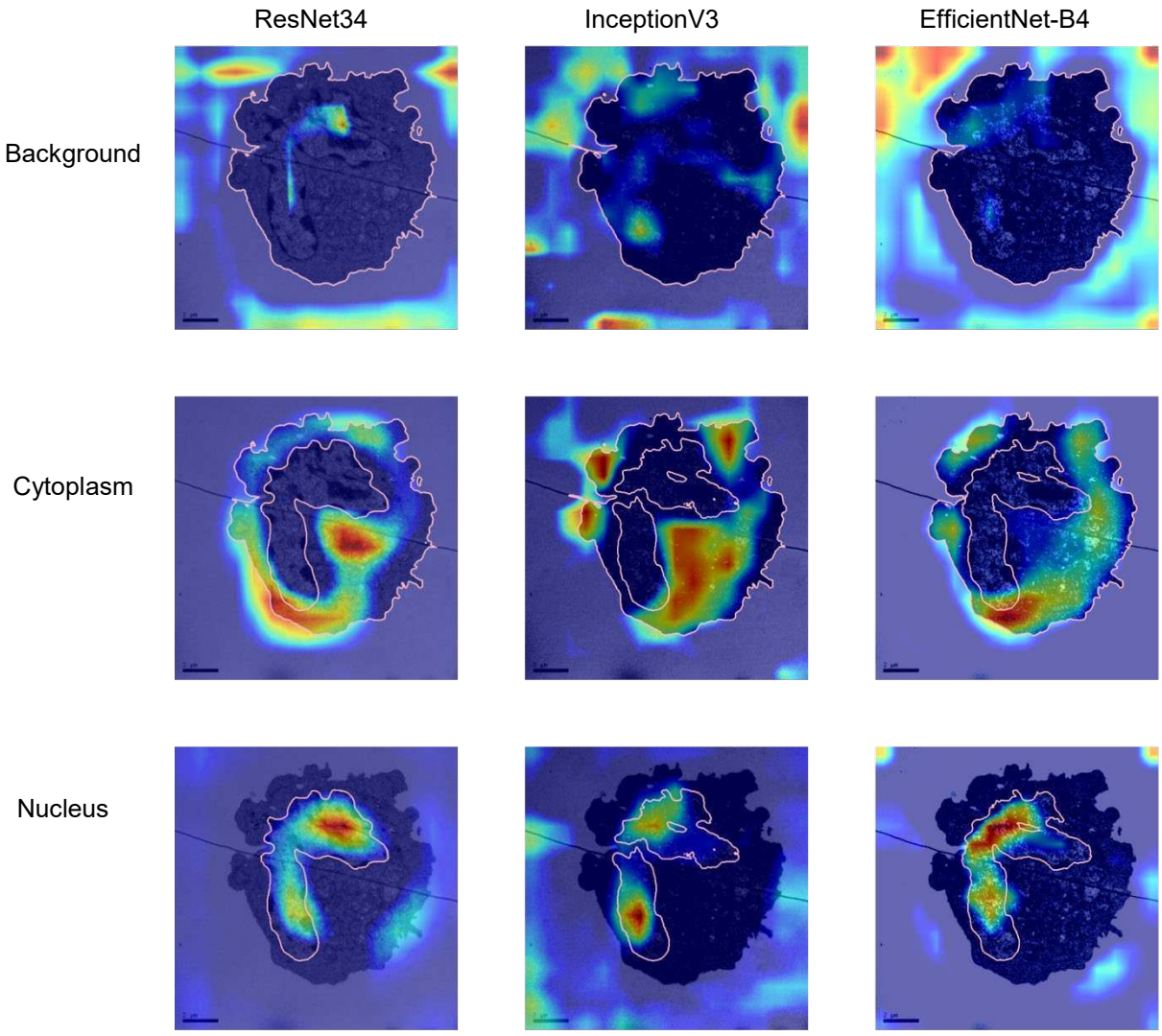
Representative Grad-CAM images from Dataset 4



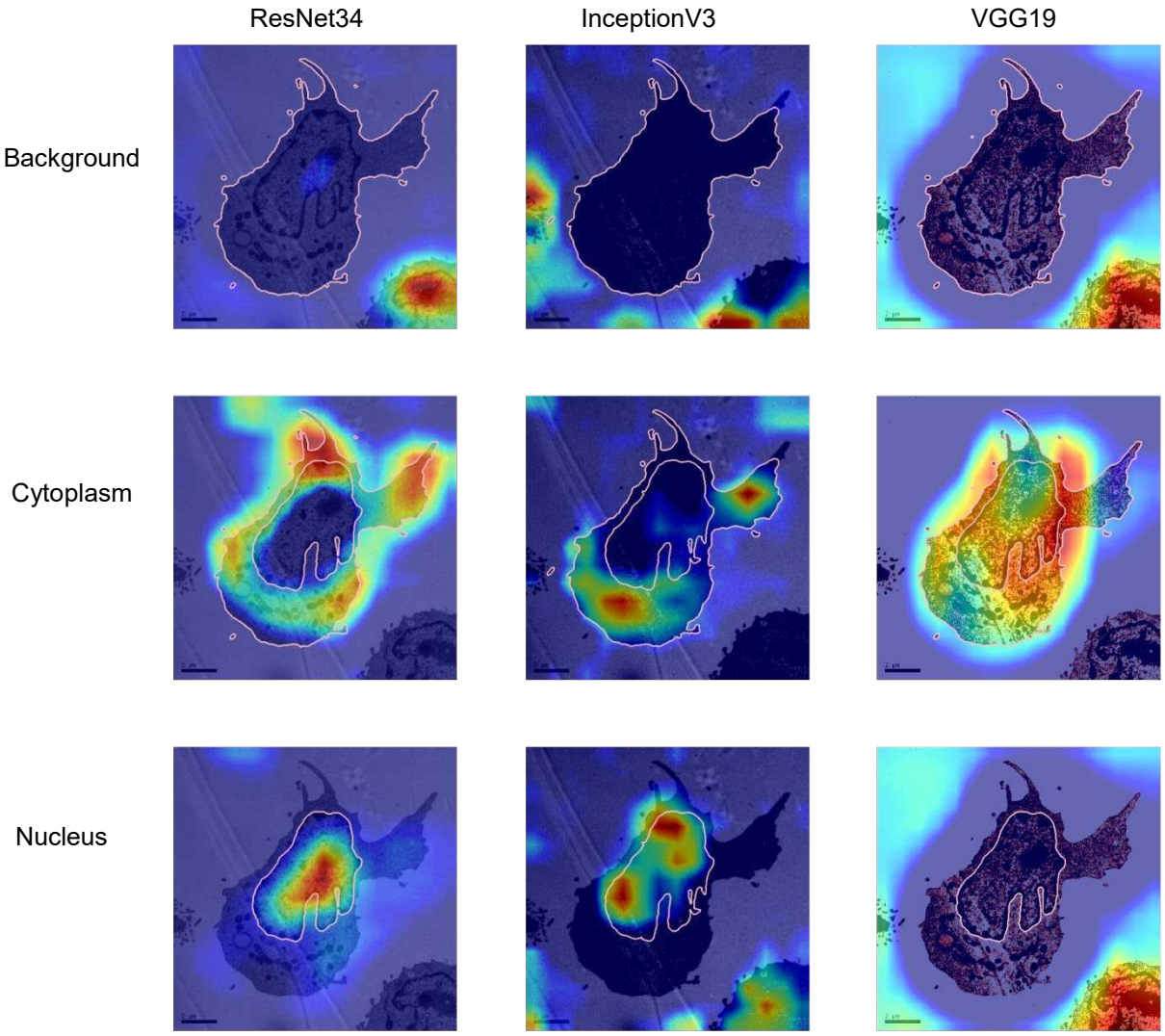
Representative Grad-CAM images from Dataset 5



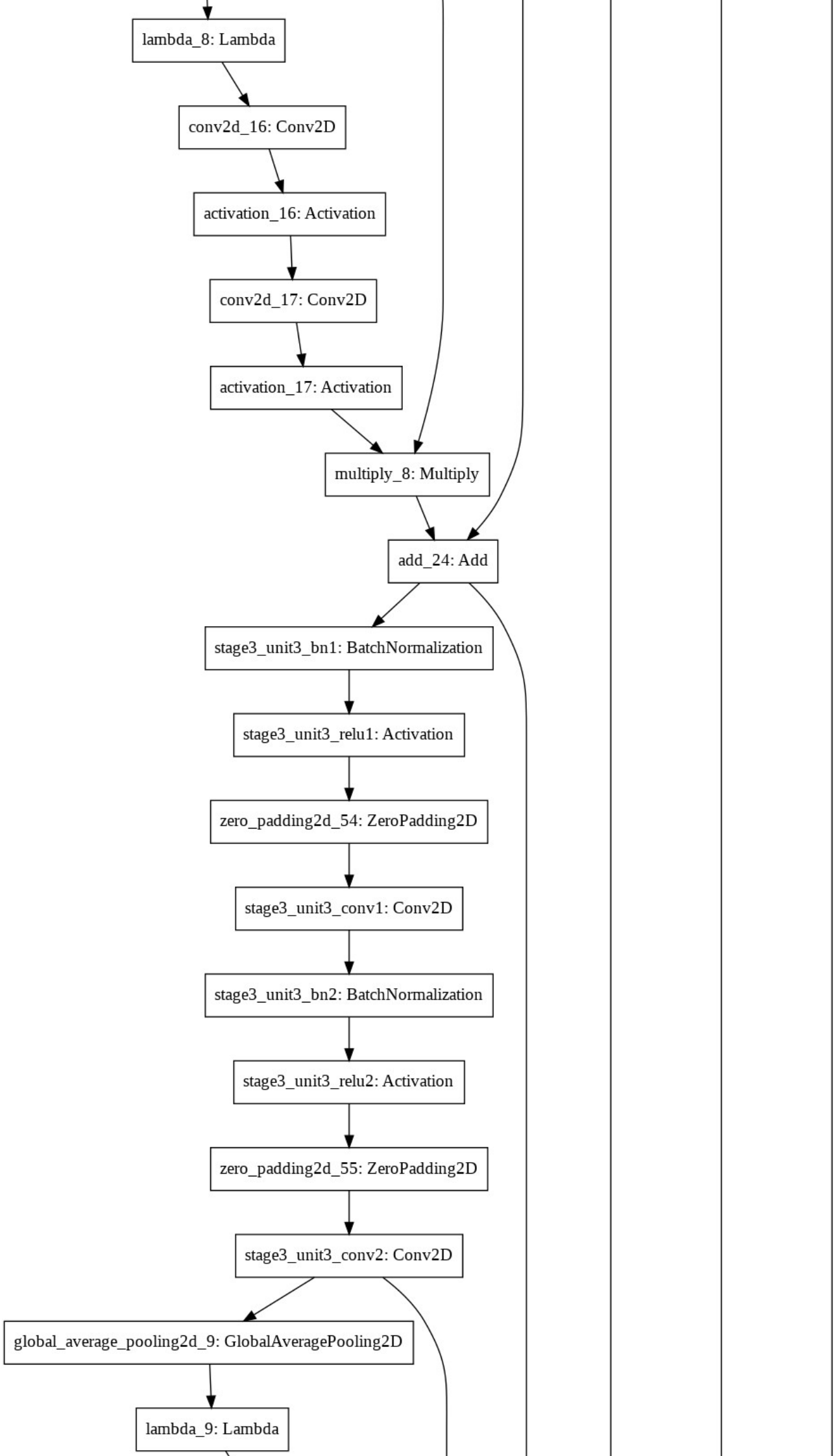
Representative Grad-CAM images from Dataset 6

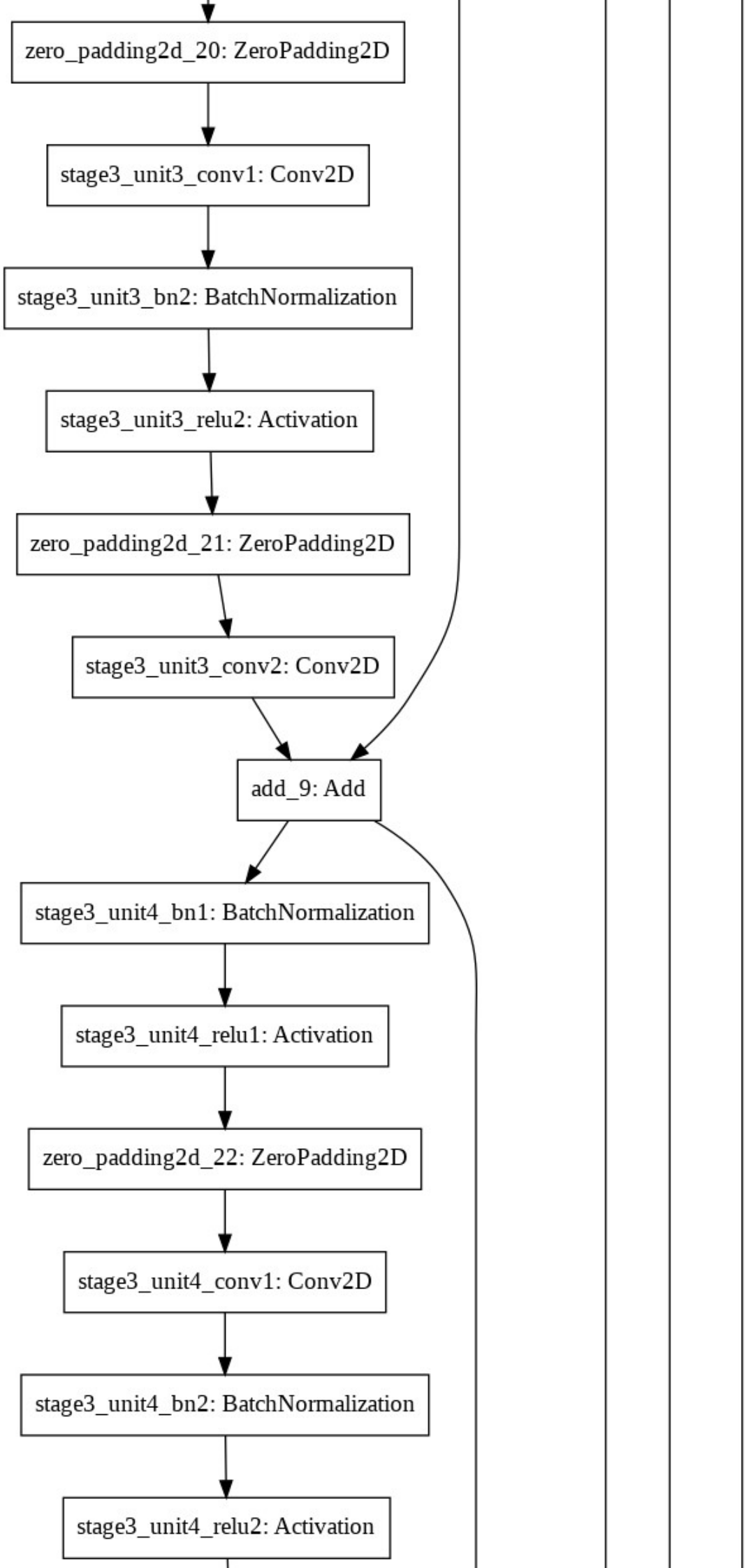


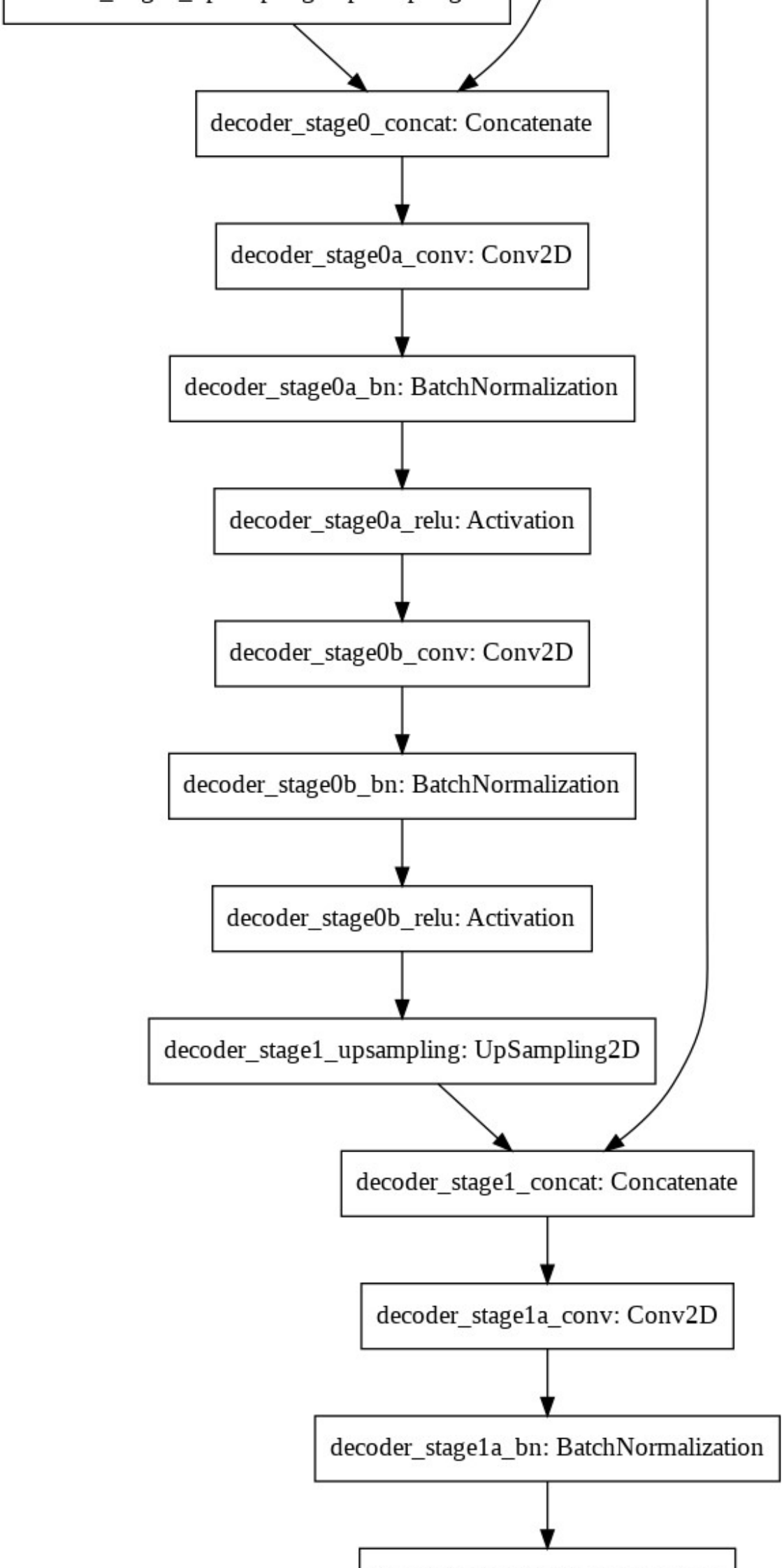
Representative Grad-CAM images from Dataset 7

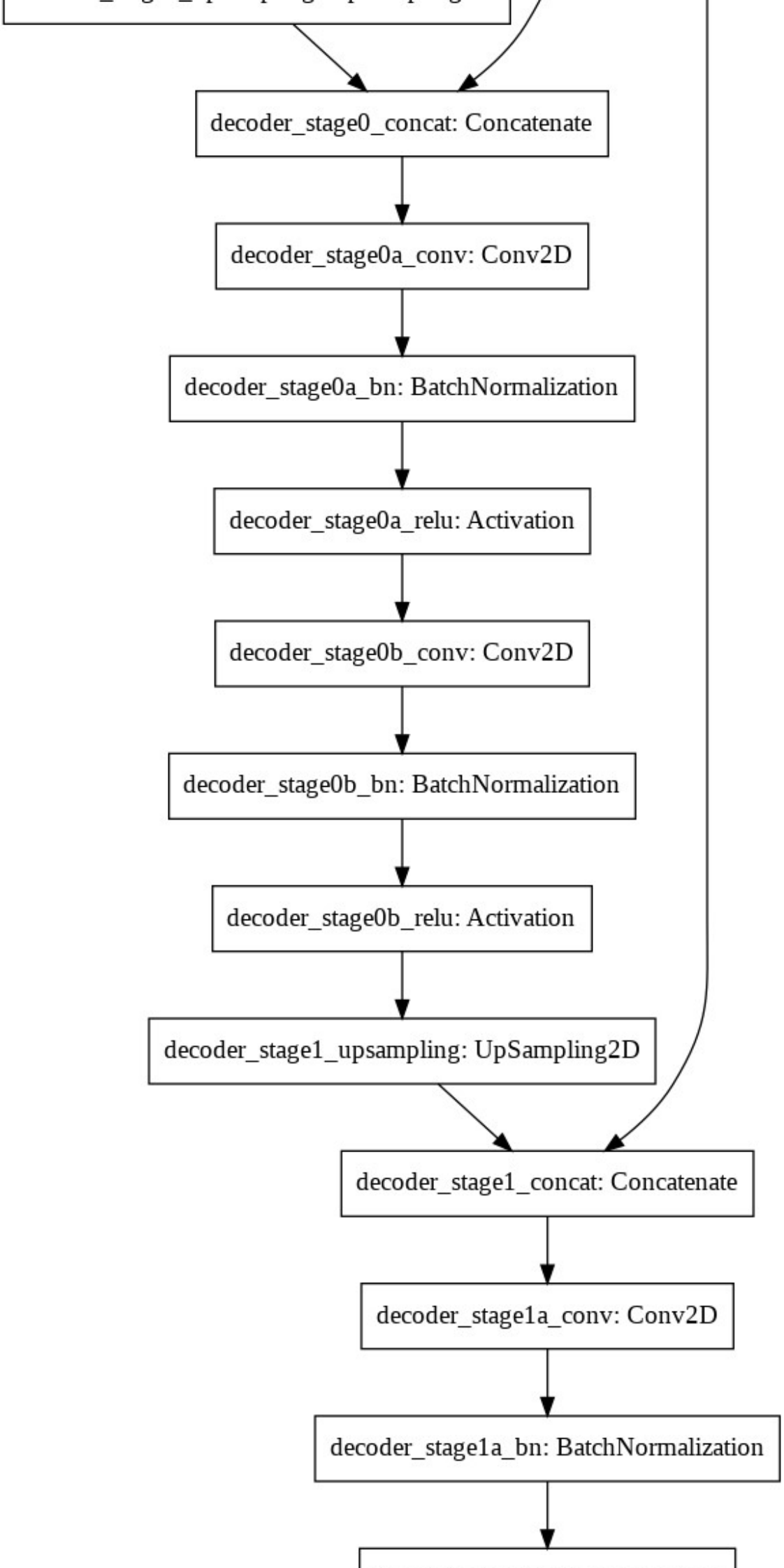


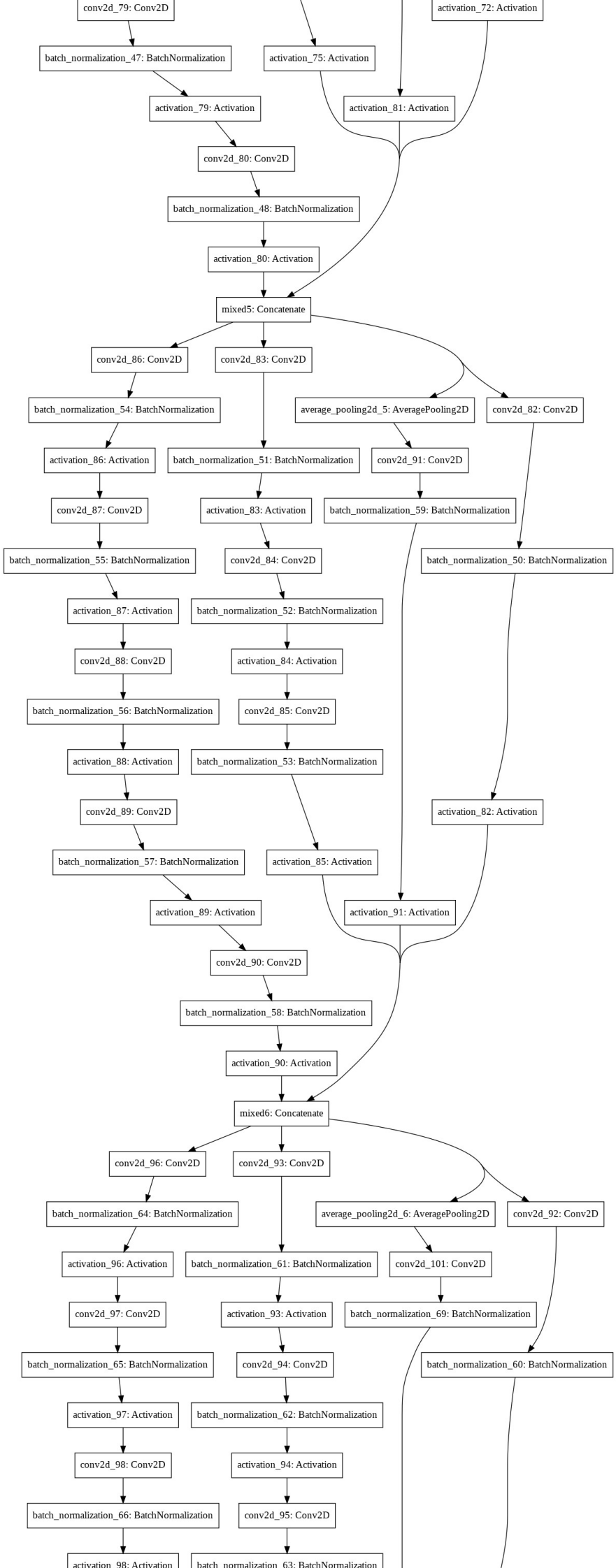
The following pages show representative sections of all the model architectures used in the work. These model architectures are of large dimensions (average image height of 18943 pixels) and are best viewed in electronic format using large image viewer tools.

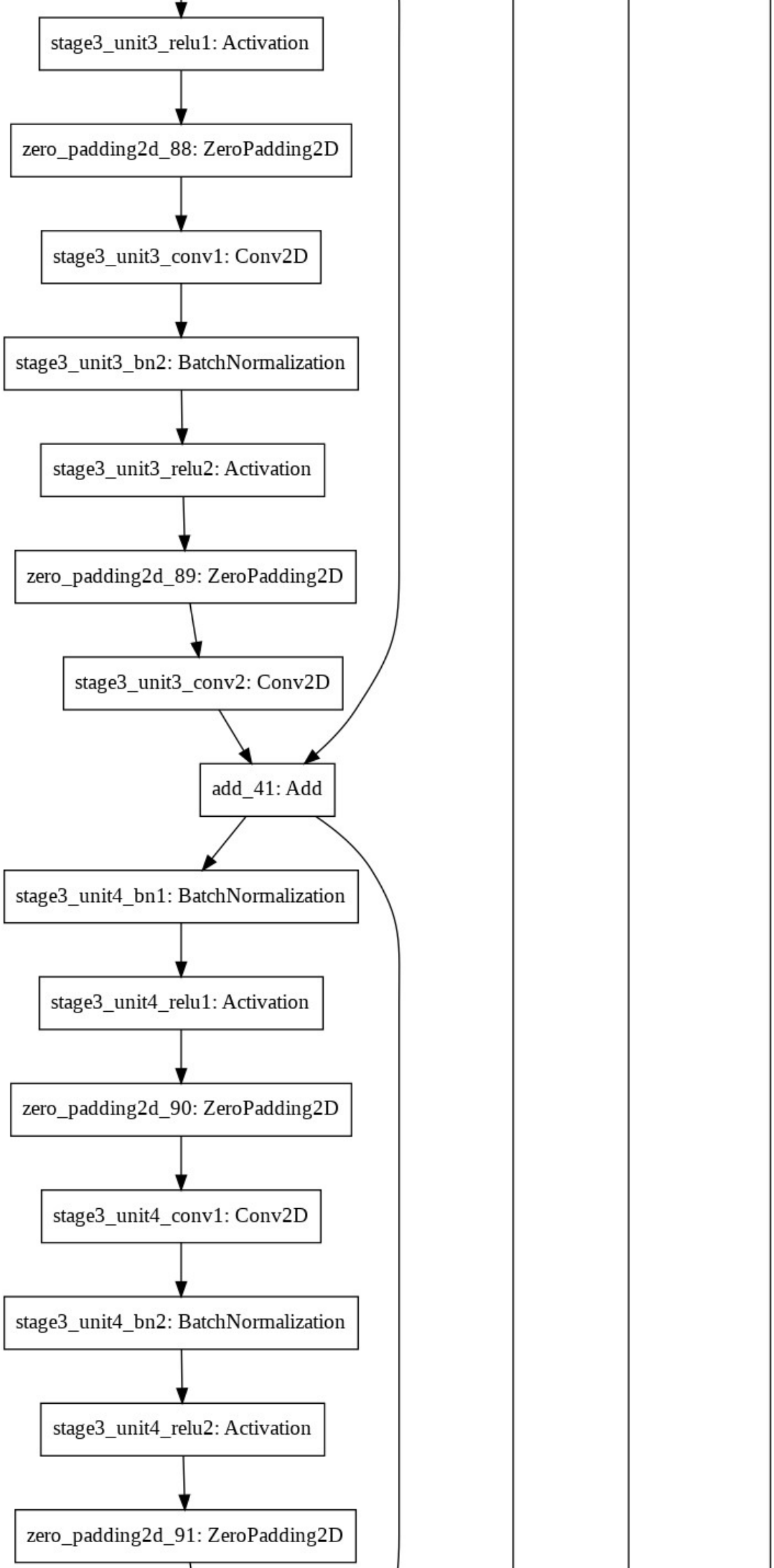

lambda_8: Lambda
conv2d_16: Conv2D
activation_16: Activation
conv2d_17: Conv2D
activation_17: Activation
multiply_8: Multiply
add_24: Add
stage3_unit3_bn1: BatchNormalization
stage3_unit3_relu1: Activation
zero_padding2d_54: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_55: ZeroPadding2D
stage3_unit3_conv2: Conv2D
global_average_pooling2d_9: GlobalAveragePooling2D
lambda_9: Lambda

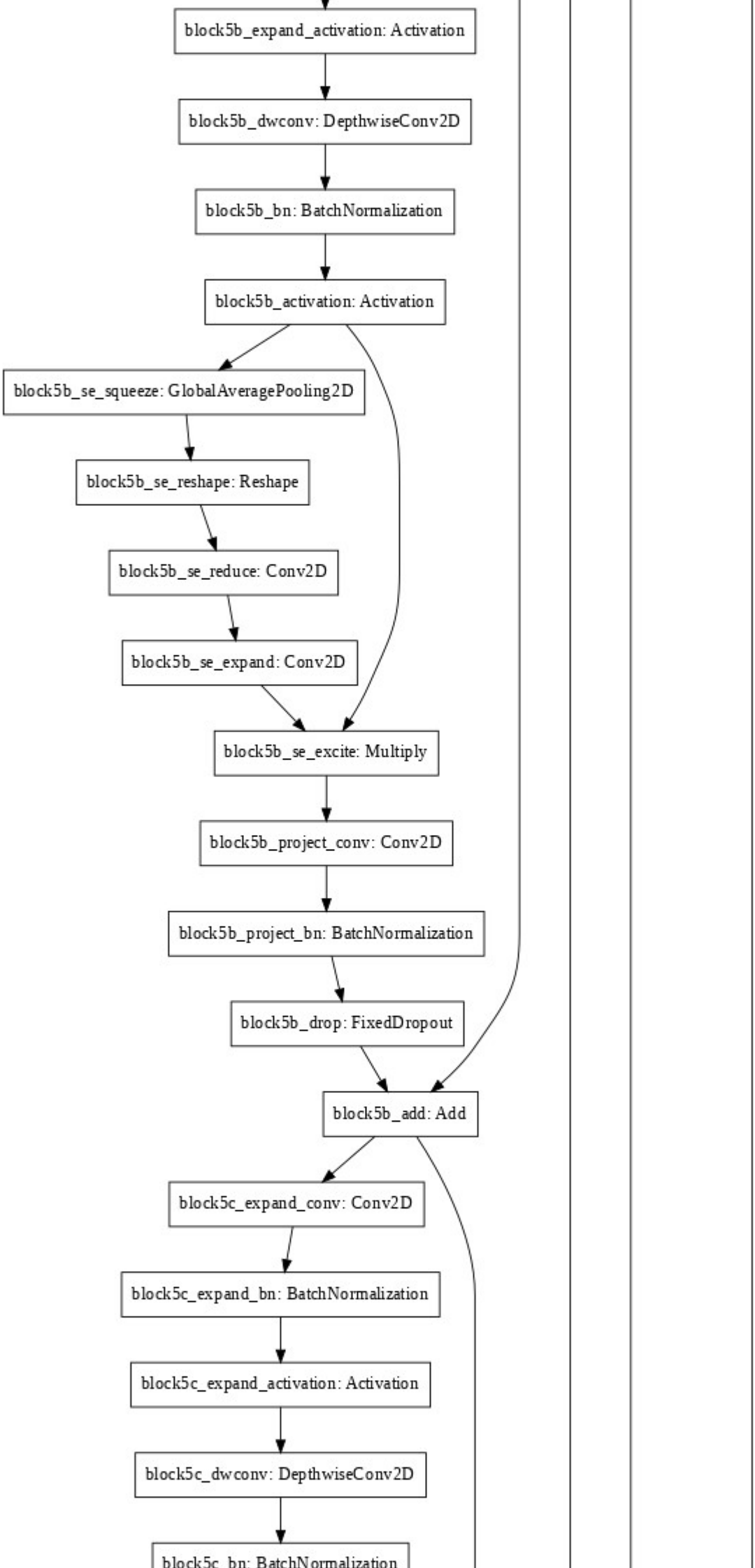

zero_padding2d_20: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_21: ZeroPadding2D
stage3_unit3_conv2: Conv2D
add_9: Add
stage3_unit4_bn1: BatchNormalization
stage3_unit4_relu1: Activation
zero_padding2d_22: ZeroPadding2D
stage3_unit4_conv1: Conv2D
stage3_unit4_bn2: BatchNormalization
stage3_unit4_relu2: Activation

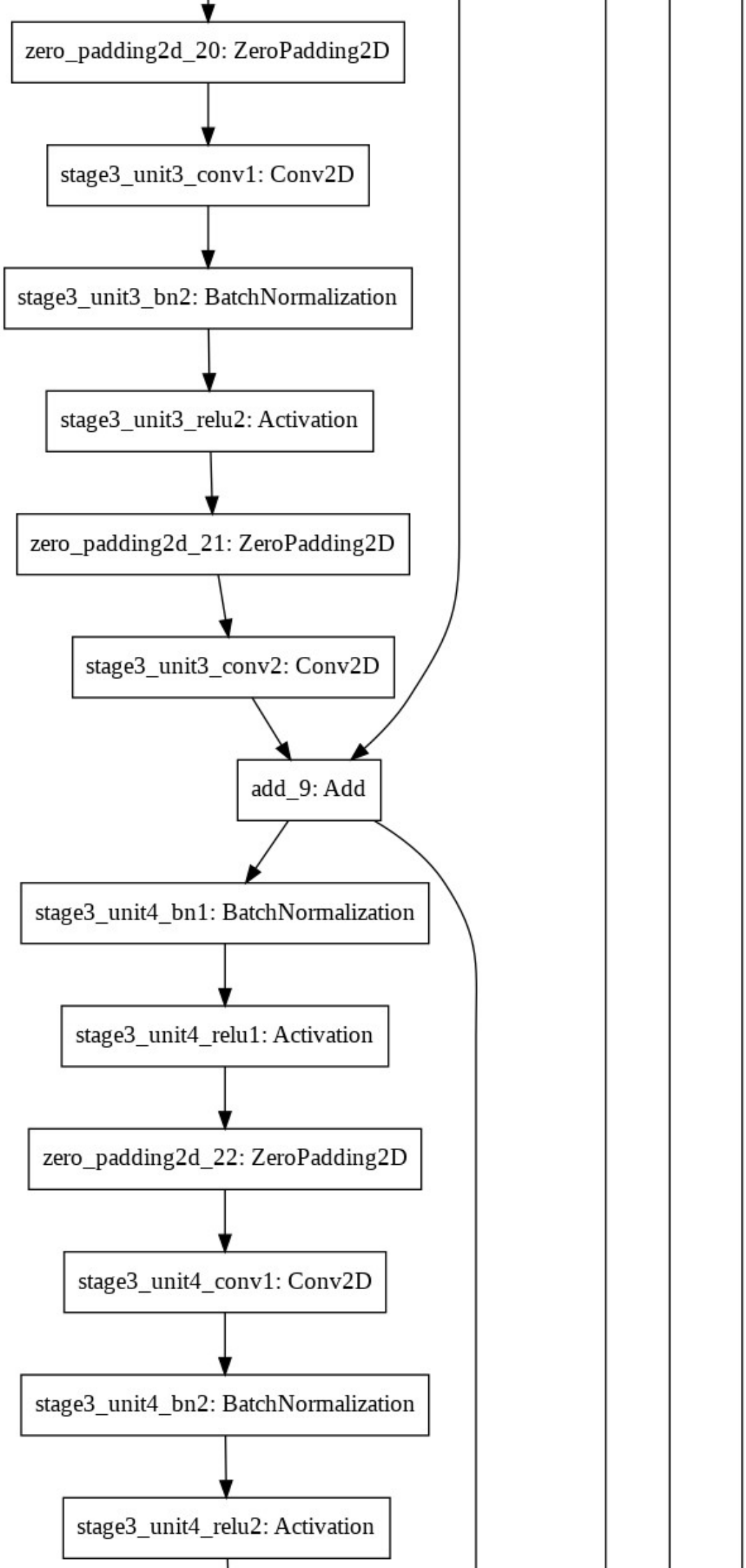

decoder_stage0_concat: Concatenate
decoder_stage0a_conv: Conv2D
decoder_stage0a_bn: BatchNormalization
decoder_stage0a_relu: Activation
decoder_stage0b_conv: Conv2D
decoder_stage0b_bn: BatchNormalization
decoder_stage0b_relu: Activation
decoder_stage1_upsampling: UpSampling2D
decoder_stage1_concat: Concatenate
decoder_stage1a_conv: Conv2D
decoder_stage1a_bn: BatchNormalization

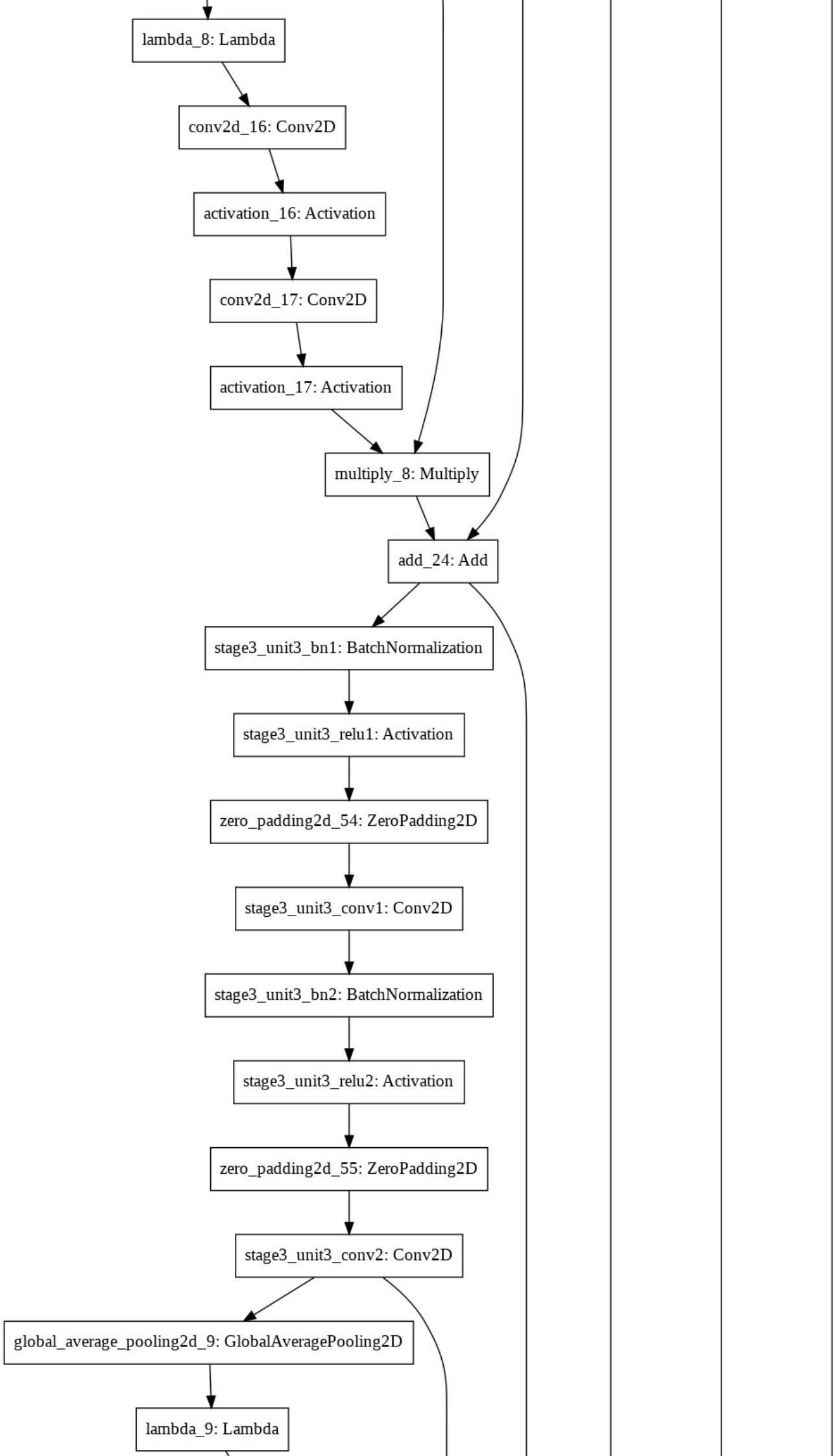

decoder_stage0_concat: Concatenate
decoder_stage0a_conv: Conv2D
decoder_stage0a_bn: BatchNormalization
decoder_stage0a_relu: Activation
decoder_stage0b_conv: Conv2D
decoder_stage0b_bn: BatchNormalization
decoder_stage0b_relu: Activation
decoder_stage1_upsampling: UpSampling2D
decoder_stage1_concat: Concatenate
decoder_stage1a_conv: Conv2D
decoder_stage1a_bn: BatchNormalization

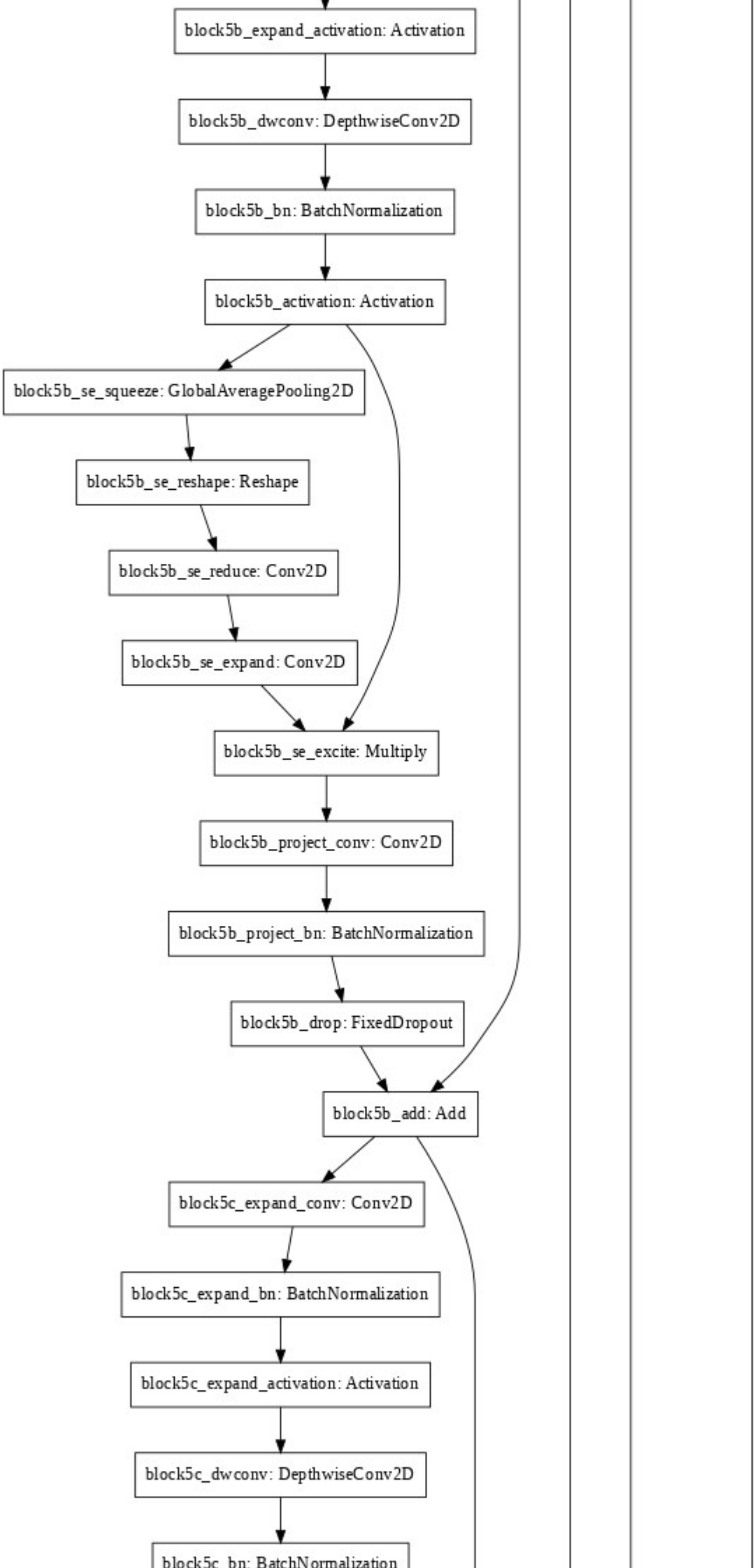

conv2d_79: Conv2D
activation_72: Activation
batch_normalization_47: BatchNormalization
activation_75: Activation
activation_79: Activation
activation_81: Activation
conv2d_80: Conv2D
batch_normalization_48: BatchNormalization
activation_80: Activation
mixed5: Concatenate
conv2d_86: Conv2D
conv2d_83: Conv2D
batch_normalization_54: BatchNormalization
average_pooling2d_5: AveragePooling2D
conv2d_82: Conv2D
activation_86: Activation
batch_normalization_51: BatchNormalization
conv2d_91: Conv2D
conv2d_87: Conv2D
activation_83: Activation
batch_normalization_59: BatchNormalization
batch_normalization_55: BatchNormalization
conv2d_84: Conv2D
batch_normalization_50: BatchNormalization
activation_87: Activation
batch_normalization_52: BatchNormalization
conv2d_88: Conv2D
activation_84: Activation
batch_normalization_56: BatchNormalization
conv2d_85: Conv2D
activation_88: Activation
batch_normalization_53: BatchNormalization
conv2d_89: Conv2D
activation_82: Activation
batch_normalization_57: BatchNormalization
activation_85: Activation
activation_89: Activation
activation_91: Activation
conv2d_90: Conv2D
batch_normalization_58: BatchNormalization
activation_90: Activation
mixed6: Concatenate
conv2d_96: Conv2D
conv2d_93: Conv2D
batch_normalization_64: BatchNormalization
average_pooling2d_6: AveragePooling2D
conv2d_92: Conv2D
activation_96: Activation
batch_normalization_61: BatchNormalization
conv2d_101: Conv2D
conv2d_97: Conv2D
activation_93: Activation
batch_normalization_69: BatchNormalization
batch_normalization_65: BatchNormalization
conv2d_94: Conv2D
batch_normalization_60: BatchNormalization
activation_97: Activation
batch_normalization_62: BatchNormalization
conv2d_98: Conv2D
activation_94: Activation
batch_normalization_66: BatchNormalization
conv2d_95: Conv2D

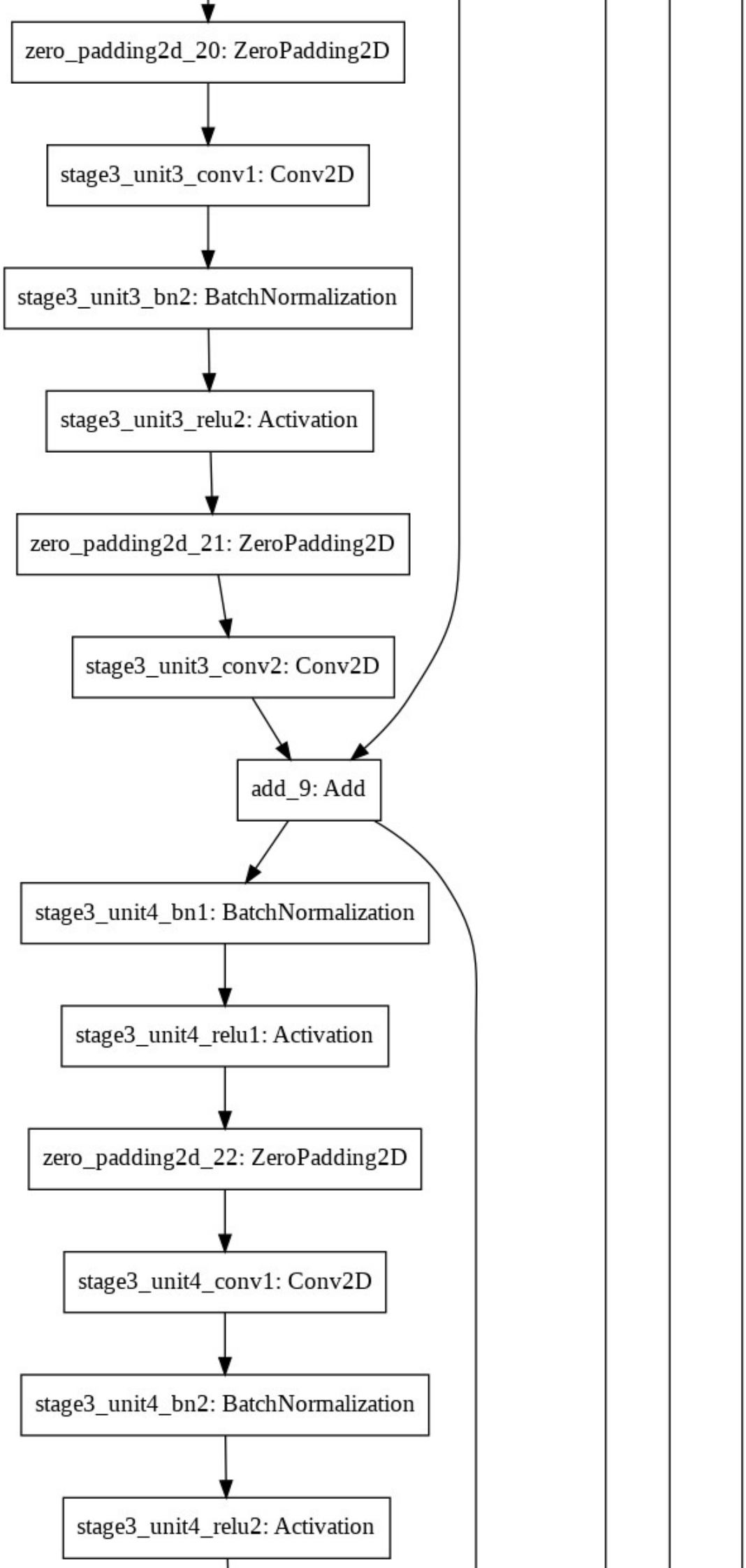

stage3_unit3_relu1: Activation
zero_padding2d_88: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_89: ZeroPadding2D
stage3_unit3_conv2: Conv2D
add_41: Add
stage3_unit4_bn1: BatchNormalization
stage3_unit4_relu1: Activation
zero_padding2d_90: ZeroPadding2D
stage3_unit4_conv1: Conv2D
stage3_unit4_bn2: BatchNormalization
stage3_unit4_relu2: Activation
zero_padding2d_91: ZeroPadding2D

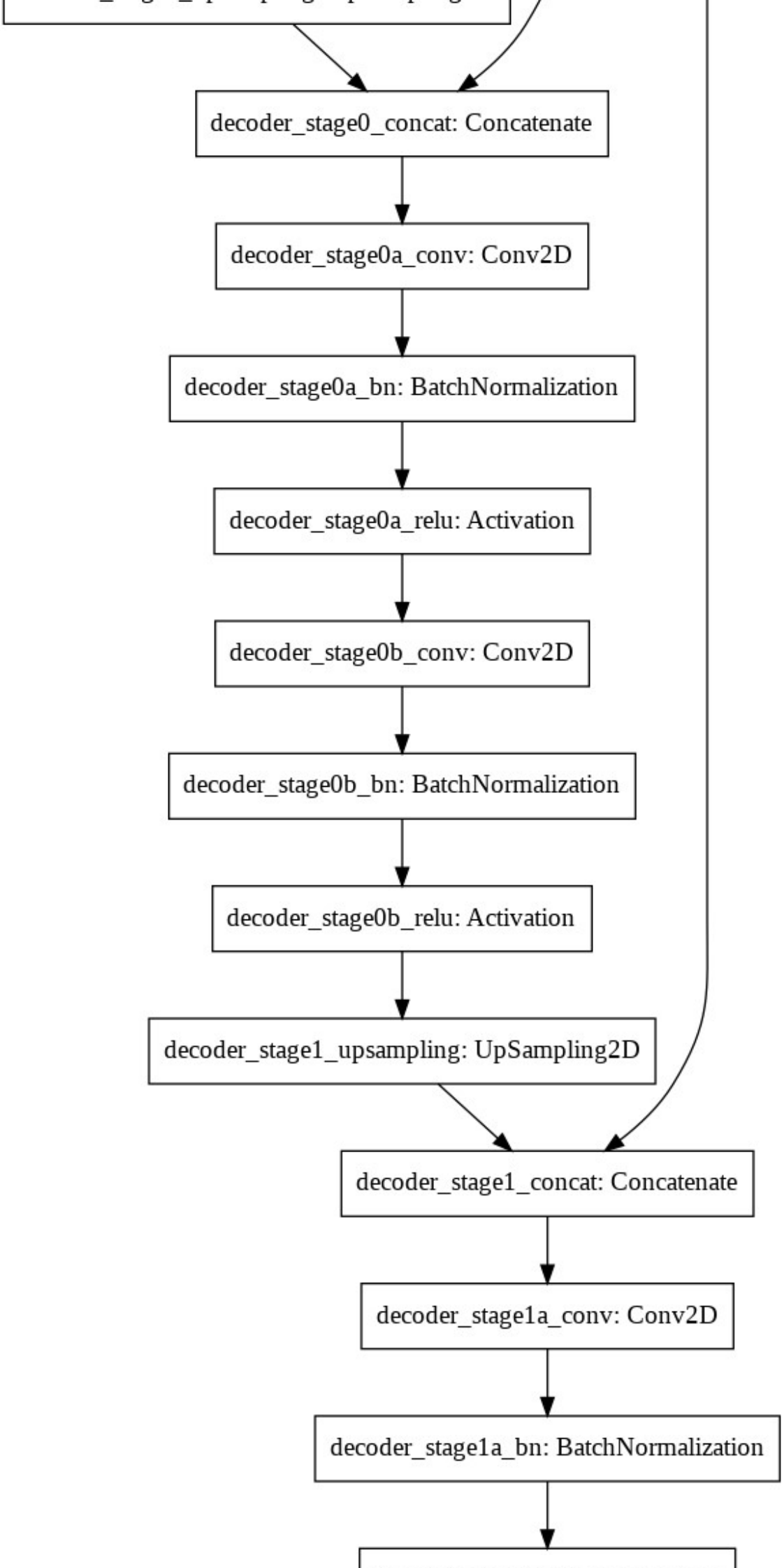

block5b_expand_activation: Activation
block5b_dwconv: DepthwiseConv2D
block5b_bn: BatchNormalization
block5b_activation: Activation
block5b_se_squeeze: GlobalAveragePooling2D
block5b_se_reshape: Reshape
block5b_se_reduce: Conv2D
block5b_se_expand: Conv2D
block5b_se_excite: Multiply
block5b_project_conv: Conv2D
block5b_project_bn: BatchNormalization
block5b_drop: FixedDropout
block5b_add: Add
block5c_expand_conv: Conv2D
block5c_expand_bn: BatchNormalization
block5c_expand_activation: Activation
block5c_dwconv: DepthwiseConv2D
block5c_bn: BatchNormalization

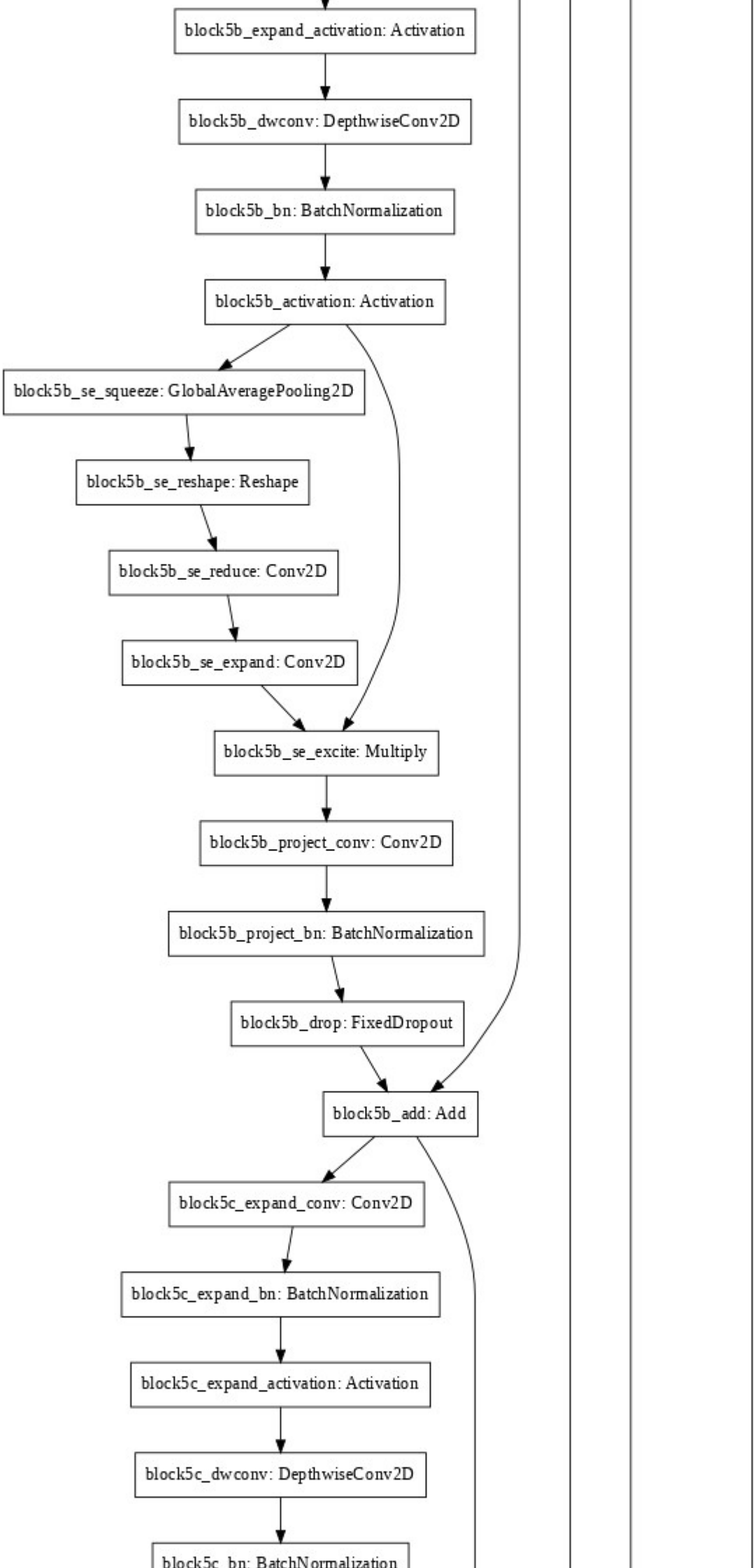

zero_padding2d_20: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_21: ZeroPadding2D
stage3_unit3_conv2: Conv2D
add_9: Add
stage3_unit4_bn1: BatchNormalization
stage3_unit4_relu1: Activation
zero_padding2d_22: ZeroPadding2D
stage3_unit4_conv1: Conv2D
stage3_unit4_bn2: BatchNormalization
stage3_unit4_relu2: Activation

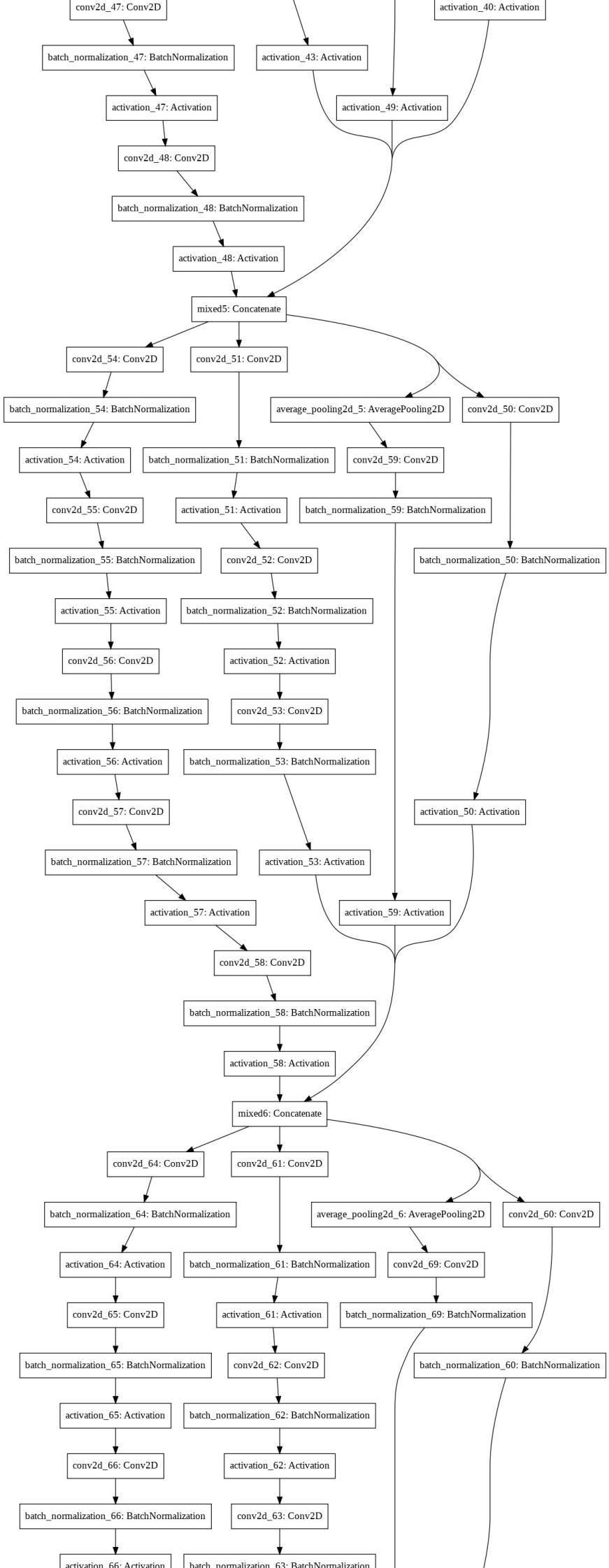

lambda_8: Lambda
conv2d_16: Conv2D
activation_16: Activation
conv2d_17: Conv2D
activation_17: Activation
multiply_8: Multiply
add_24: Add
stage3_unit3_bn1: BatchNormalization
stage3_unit3_relu1: Activation
zero_padding2d_54: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_55: ZeroPadding2D
stage3_unit3_conv2: Conv2D
global_average_pooling2d_9: GlobalAveragePooling2D
lambda_9: Lambda

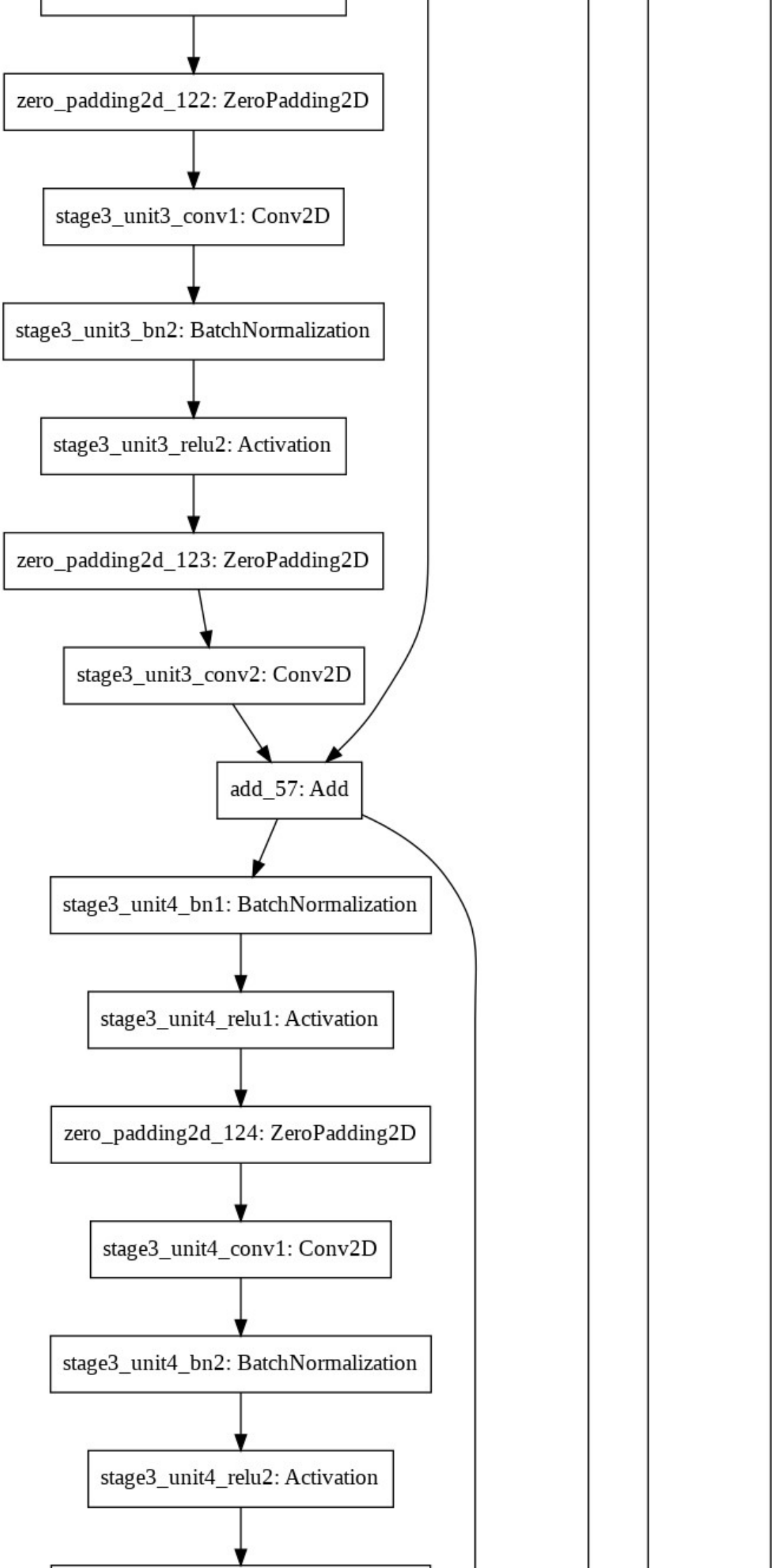

block5b_expand_activation: Activation
block5b_dwconv: DepthwiseConv2D
block5b_bn: BatchNormalization
block5b_activation: Activation
block5b_se_squeeze: GlobalAveragePooling2D
block5b_se_reshape: Reshape
block5b_se_reduce: Conv2D
block5b_se_expand: Conv2D
block5b_se_excite: Multiply
block5b_project_conv: Conv2D
block5b_project_bn: BatchNormalization
block5b_drop: FixedDropout
block5b_add: Add
block5c_expand_conv: Conv2D
block5c_expand_bn: BatchNormalization
block5c_expand_activation: Activation
block5c_dwconv: DepthwiseConv2D
block5c_bn: BatchNormalization

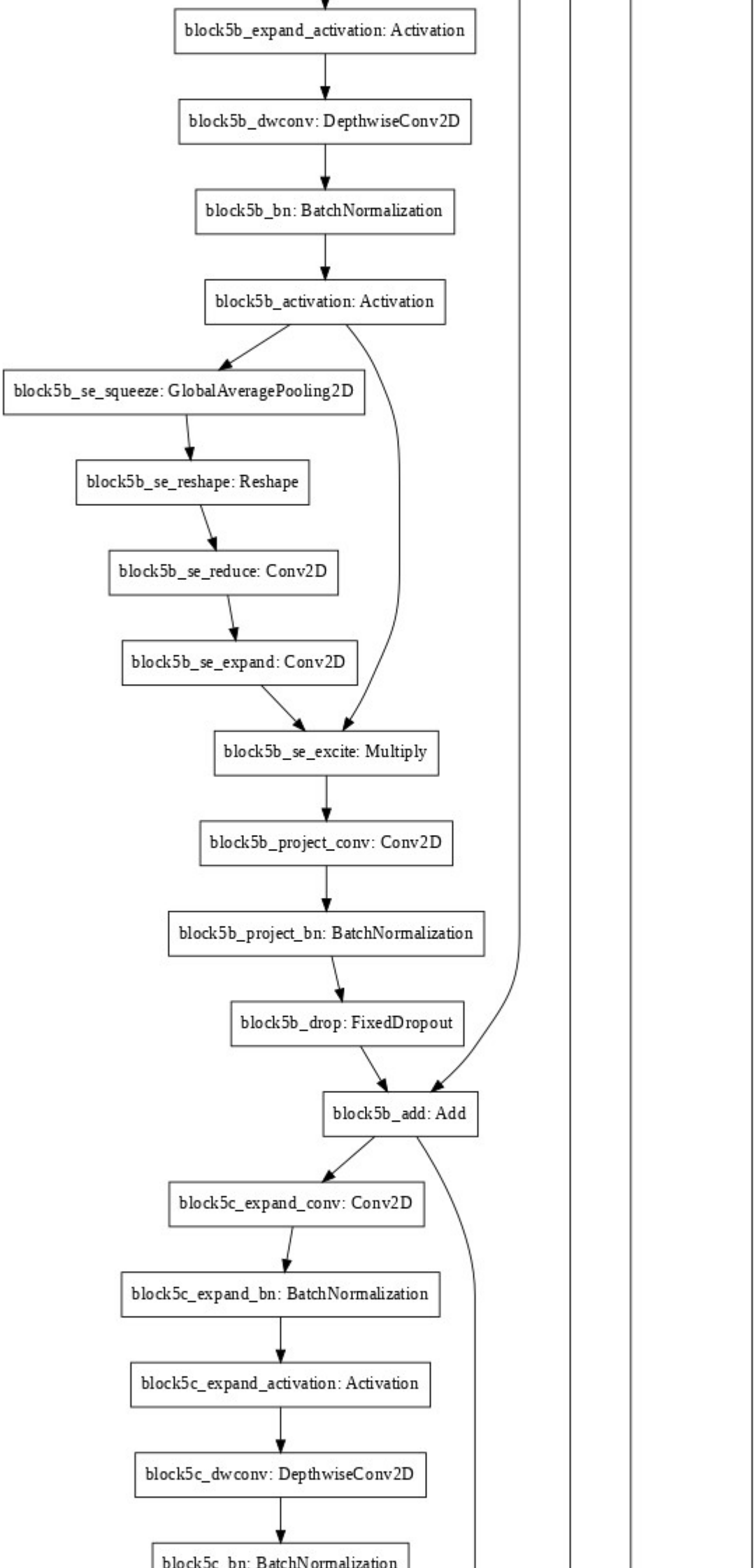

zero_padding2d_20: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_21: ZeroPadding2D
stage3_unit3_conv2: Conv2D
add_9: Add
stage3_unit4_bn1: BatchNormalization
stage3_unit4_relu1: Activation
zero_padding2d_22: ZeroPadding2D
stage3_unit4_conv1: Conv2D
stage3_unit4_bn2: BatchNormalization
stage3_unit4_relu2: Activation

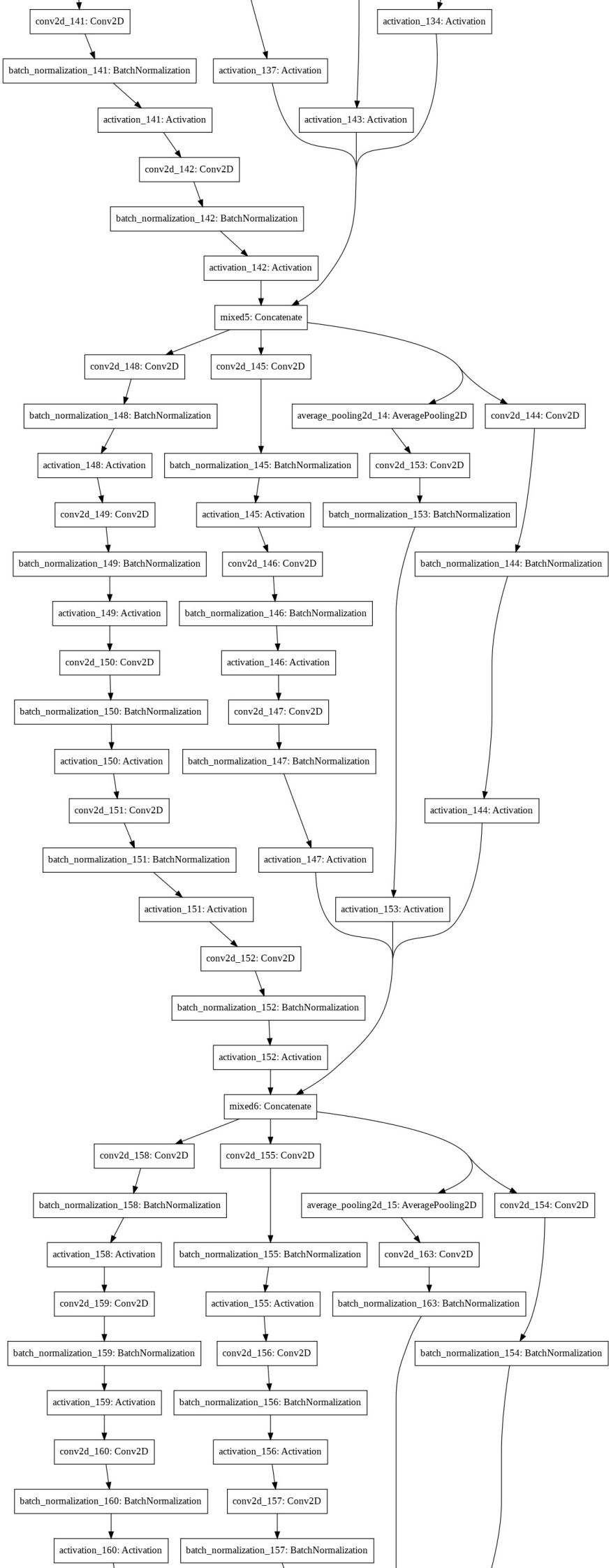

decoder_stage0_concat: Concatenate
decoder_stage0a_conv: Conv2D
decoder_stage0a_bn: BatchNormalization
decoder_stage0a_relu: Activation
decoder_stage0b_conv: Conv2D
decoder_stage0b_bn: BatchNormalization
decoder_stage0b_relu: Activation
decoder_stage1_upsampling: UpSampling2D
decoder_stage1_concat: Concatenate
decoder_stage1a_conv: Conv2D
decoder_stage1a_bn: BatchNormalization

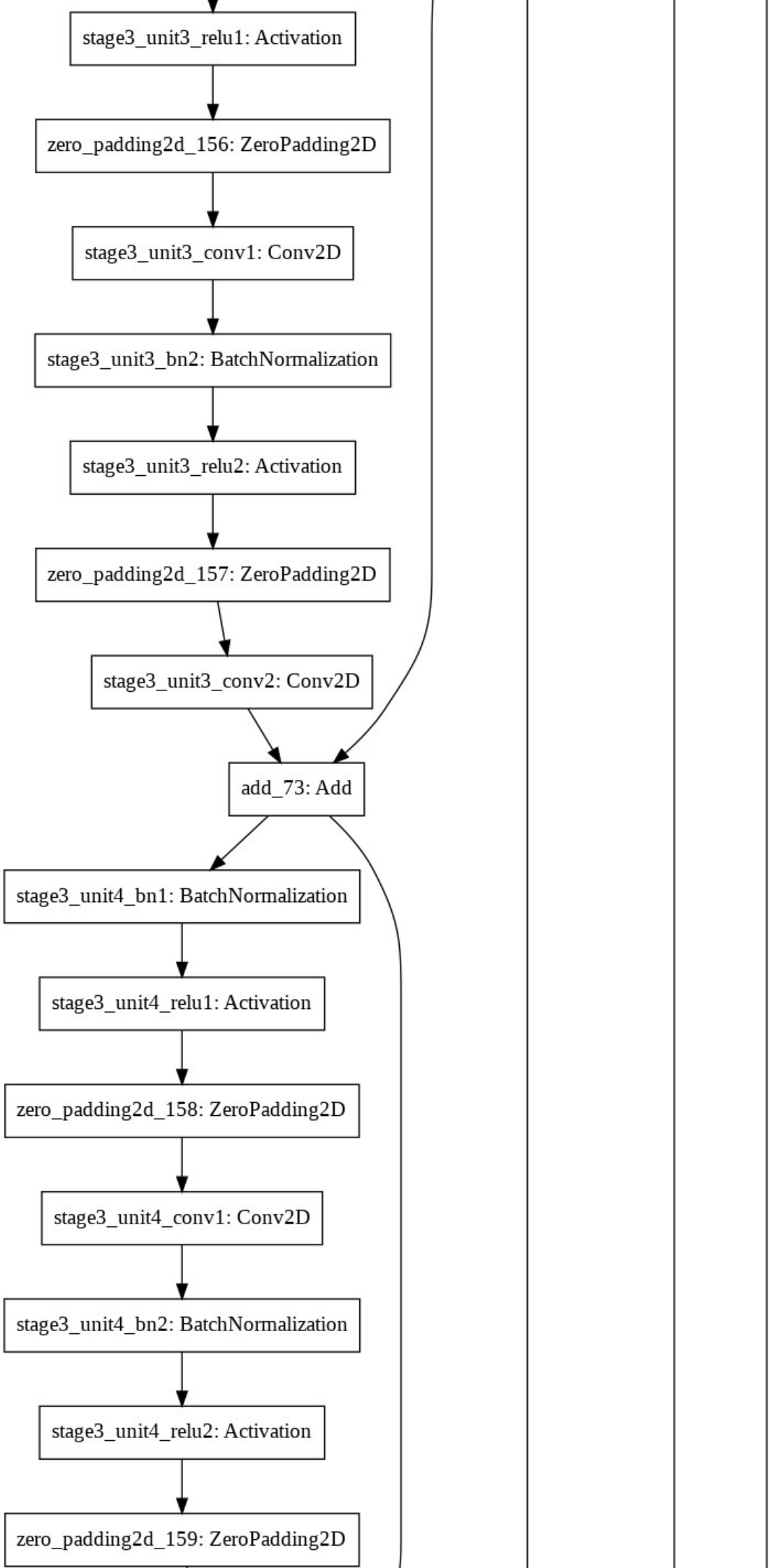

block5b_expand_activation: Activation
block5b_dwconv: DepthwiseConv2D
block5b_bn: BatchNormalization
block5b_activation: Activation
block5b_se_squeeze: GlobalAveragePooling2D
block5b_se_reshape: Reshape
block5b_se_reduce: Conv2D
block5b_se_expand: Conv2D
block5b_se_excite: Multiply
block5b_project_conv: Conv2D
block5b_project_bn: BatchNormalization
block5b_drop: FixedDropout
block5b_add: Add
block5c_expand_conv: Conv2D
block5c_expand_bn: BatchNormalization
block5c_expand_activation: Activation
block5c_dwconv: DepthwiseConv2D
block5c_bn: BatchNormalization

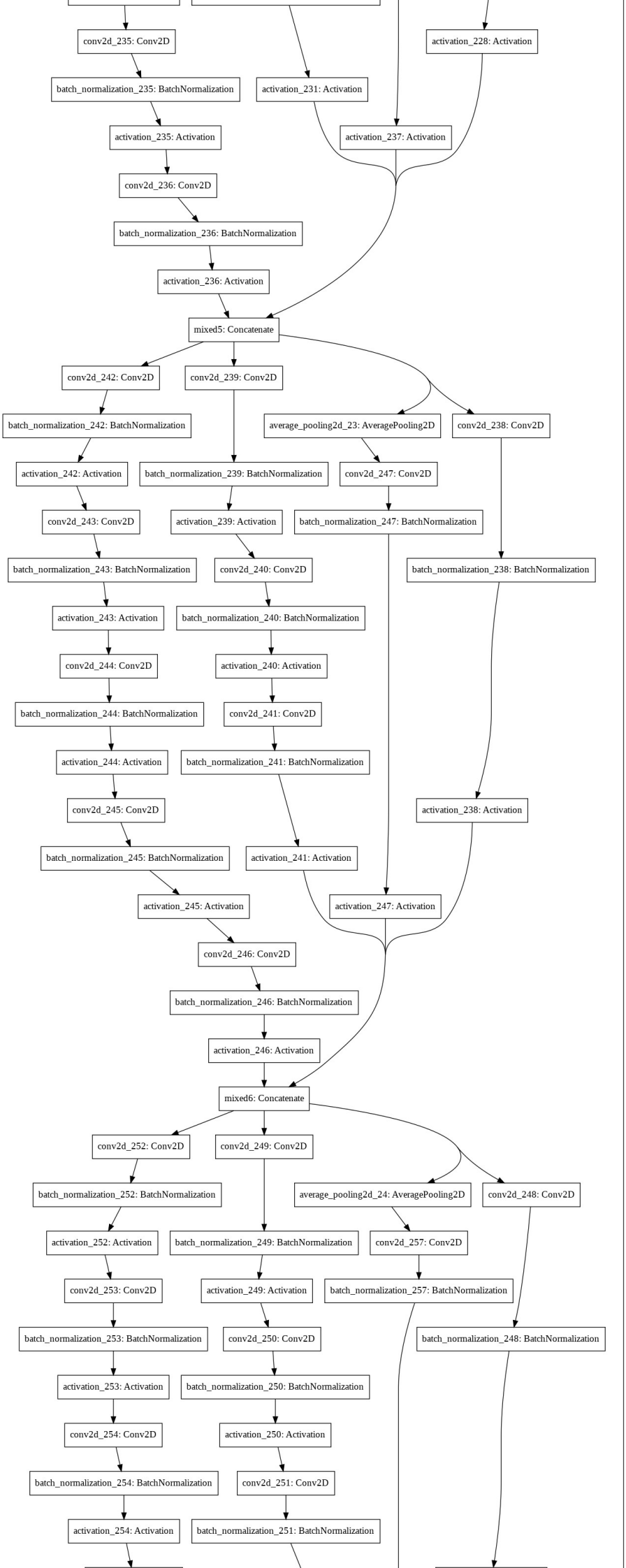

conv2d_47: Conv2D
activation_40: Activation
batch_normalization_47: BatchNormalization
activation_43: Activation
activation_47: Activation
activation_49: Activation
conv2d_48: Conv2D
batch_normalization_48: BatchNormalization
activation_48: Activation
mixed5: Concatenate
conv2d_54: Conv2D
conv2d_51: Conv2D
batch_normalization_54: BatchNormalization
average_pooling2d_5: AveragePooling2D
conv2d_50: Conv2D
activation_54: Activation
batch_normalization_51: BatchNormalization
conv2d_59: Conv2D
conv2d_55: Conv2D
activation_51: Activation
batch_normalization_59: BatchNormalization
batch_normalization_55: BatchNormalization
conv2d_52: Conv2D
batch_normalization_50: BatchNormalization
activation_55: Activation
batch_normalization_52: BatchNormalization
conv2d_56: Conv2D
activation_52: Activation
batch_normalization_56: BatchNormalization
conv2d_53: Conv2D
activation_56: Activation
batch_normalization_53: BatchNormalization
conv2d_57: Conv2D
activation_50: Activation
batch_normalization_57: BatchNormalization
activation_53: Activation
activation_57: Activation
activation_59: Activation
conv2d_58: Conv2D
batch_normalization_58: BatchNormalization
activation_58: Activation
mixed6: Concatenate
conv2d_64: Conv2D
conv2d_61: Conv2D
batch_normalization_64: BatchNormalization
average_pooling2d_6: AveragePooling2D
conv2d_60: Conv2D
activation_64: Activation
batch_normalization_61: BatchNormalization
conv2d_69: Conv2D
conv2d_65: Conv2D
activation_61: Activation
batch_normalization_69: BatchNormalization
batch_normalization_65: BatchNormalization
conv2d_62: Conv2D
batch_normalization_60: BatchNormalization
activation_65: Activation
batch_normalization_62: BatchNormalization
conv2d_66: Conv2D
activation_62: Activation
batch_normalization_66: BatchNormalization
conv2d_63: Conv2D

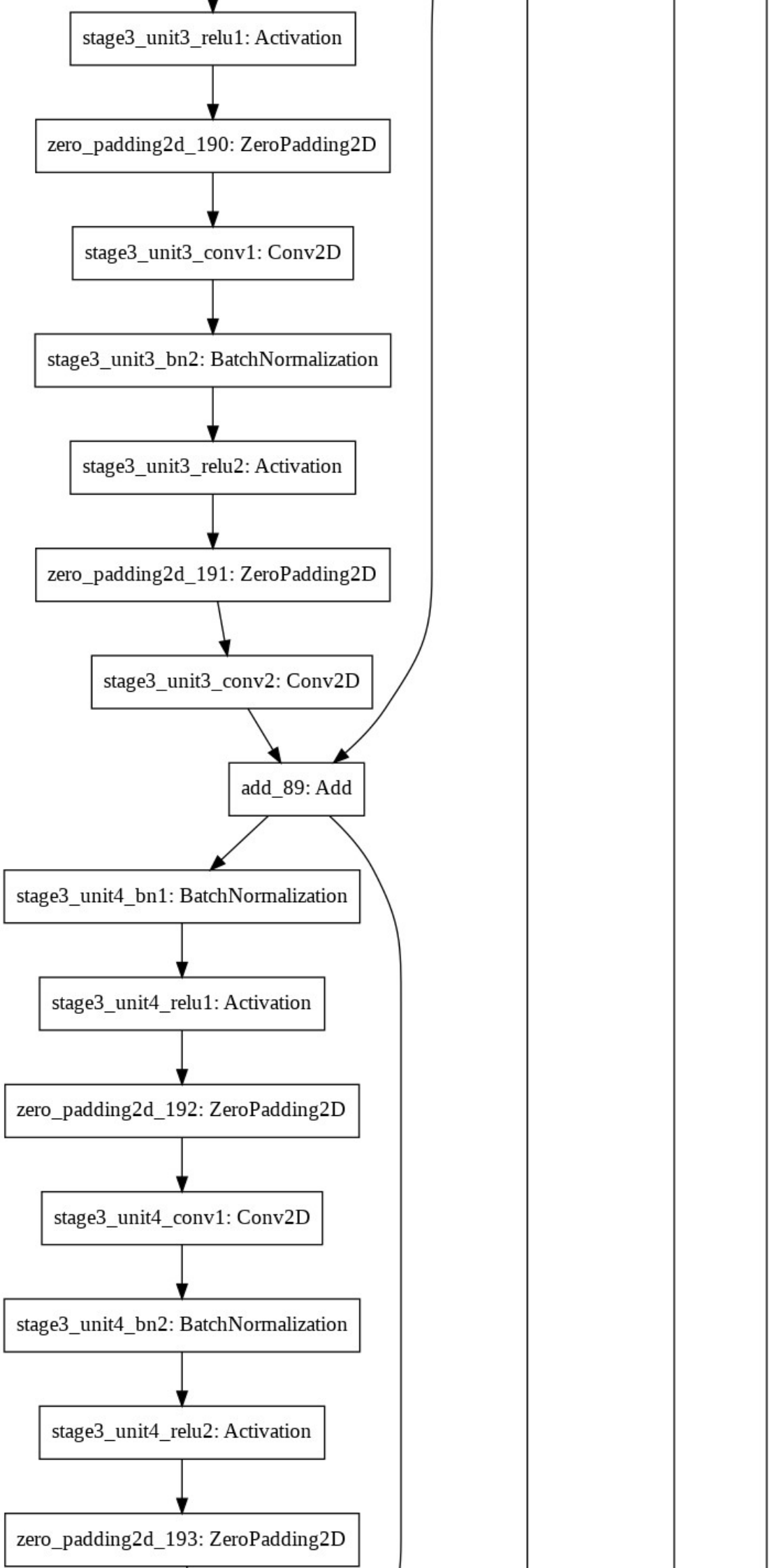

zero_padding2d_122: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_123: ZeroPadding2D
stage3_unit3_conv2: Conv2D
add_57: Add
stage3_unit4_bn1: BatchNormalization
stage3_unit4_relu1: Activation
zero_padding2d_124: ZeroPadding2D
stage3_unit4_conv1: Conv2D
stage3_unit4_bn2: BatchNormalization
stage3_unit4_relu2: Activation

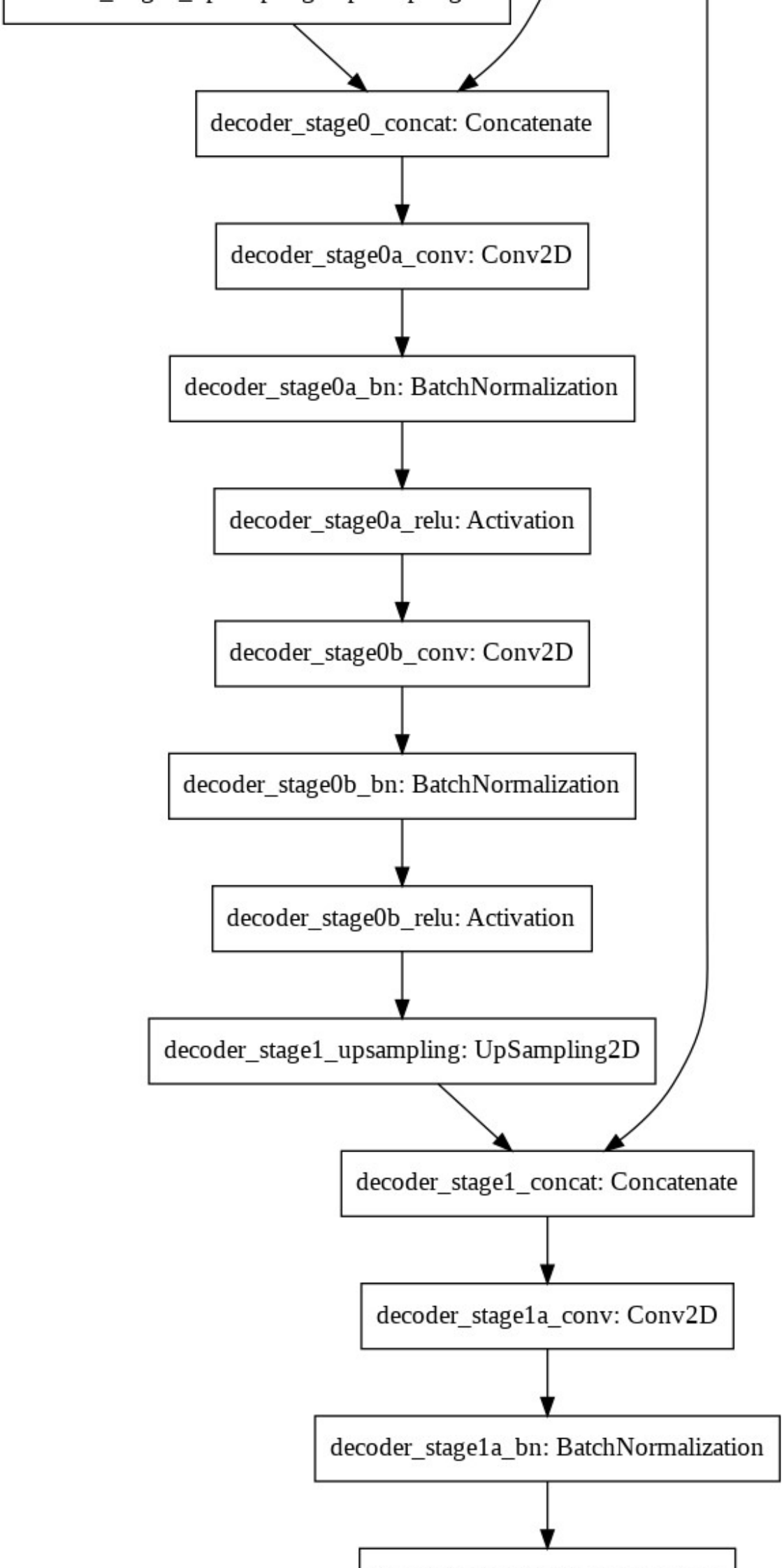

block5b_expand_activation: Activation
block5b_dwconv: DepthwiseConv2D
block5b_bn: BatchNormalization
block5b_activation: Activation
block5b_se_squeeze: GlobalAveragePooling2D
block5b_se_reshape: Reshape
block5b_se_reduce: Conv2D
block5b_se_expand: Conv2D
block5b_se_excite: Multiply
block5b_project_conv: Conv2D
block5b_project_bn: BatchNormalization
block5b_drop: FixedDropout
block5b_add: Add
block5c_expand_conv: Conv2D
block5c_expand_bn: BatchNormalization
block5c_expand_activation: Activation
block5c_dwconv: DepthwiseConv2D
block5c_bn: BatchNormalization


conv2d_141: Conv2D
activation_134: Activation
batch_normalization_141: BatchNormalization
activation_137: Activation
activation_141: Activation
activation_143: Activation
conv2d_142: Conv2D
batch_normalization_142: BatchNormalization
activation_142: Activation
mixed5: Concatenate
conv2d_148: Conv2D
conv2d_145: Conv2D
batch_normalization_148: BatchNormalization
average_pooling2d_14: AveragePooling2D
conv2d_144: Conv2D
activation_148: Activation
batch_normalization_145: BatchNormalization
conv2d_153: Conv2D
conv2d_149: Conv2D
activation_145: Activation
batch_normalization_153: BatchNormalization
batch_normalization_149: BatchNormalization
conv2d_146: Conv2D
batch_normalization_144: BatchNormalization
activation_149: Activation
batch_normalization_146: BatchNormalization
conv2d_150: Conv2D
activation_146: Activation
batch_normalization_150: BatchNormalization
conv2d_147: Conv2D
activation_150: Activation
batch_normalization_147: BatchNormalization
conv2d_151: Conv2D
activation_144: Activation
batch_normalization_151: BatchNormalization
activation_147: Activation
activation_151: Activation
activation_153: Activation
conv2d_152: Conv2D
batch_normalization_152: BatchNormalization
activation_152: Activation
mixed6: Concatenate
conv2d_158: Conv2D
conv2d_155: Conv2D
batch_normalization_158: BatchNormalization
average_pooling2d_15: AveragePooling2D
conv2d_154: Conv2D
activation_158: Activation
batch_normalization_155: BatchNormalization
conv2d_163: Conv2D
conv2d_159: Conv2D
activation_155: Activation
batch_normalization_163: BatchNormalization
batch_normalization_159: BatchNormalization
conv2d_156: Conv2D
batch_normalization_154: BatchNormalization
activation_159: Activation
batch_normalization_156: BatchNormalization
conv2d_160: Conv2D
activation_156: Activation
batch_normalization_160: BatchNormalization
conv2d_157: Conv2D
activation_160: Activation
batch_normalization_157: BatchNormalization


stage3_unit3_relu1: Activation
zero_padding2d_156: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_157: ZeroPadding2D
stage3_unit3_conv2: Conv2D
add_73: Add
stage3_unit4_bn1: BatchNormalization
stage3_unit4_relu1: Activation
zero_padding2d_158: ZeroPadding2D
stage3_unit4_conv1: Conv2D
stage3_unit4_bn2: BatchNormalization
stage3_unit4_relu2: Activation
zero_padding2d_159: ZeroPadding2D


conv2d_235: Conv2D
activation_228: Activation
batch_normalization_235: BatchNormalization
activation_231: Activation
activation_235: Activation
activation_237: Activation
conv2d_236: Conv2D
batch_normalization_236: BatchNormalization
activation_236: Activation
mixed5: Concatenate
conv2d_242: Conv2D
conv2d_239: Conv2D
batch_normalization_242: BatchNormalization
average_pooling2d_23: AveragePooling2D
conv2d_238: Conv2D
activation_242: Activation
batch_normalization_239: BatchNormalization
conv2d_247: Conv2D
conv2d_243: Conv2D
activation_239: Activation
batch_normalization_247: BatchNormalization
batch_normalization_243: BatchNormalization
conv2d_240: Conv2D
batch_normalization_238: BatchNormalization
activation_243: Activation
batch_normalization_240: BatchNormalization
conv2d_244: Conv2D
activation_240: Activation
batch_normalization_244: BatchNormalization
conv2d_241: Conv2D
activation_244: Activation
batch_normalization_241: BatchNormalization
conv2d_245: Conv2D
activation_238: Activation
batch_normalization_245: BatchNormalization
activation_241: Activation
activation_245: Activation
activation_247: Activation
conv2d_246: Conv2D
batch_normalization_246: BatchNormalization
activation_246: Activation
mixed6: Concatenate
conv2d_252: Conv2D
conv2d_249: Conv2D
batch_normalization_252: BatchNormalization
average_pooling2d_24: AveragePooling2D
conv2d_248: Conv2D
activation_252: Activation
batch_normalization_249: BatchNormalization
conv2d_257: Conv2D
conv2d_253: Conv2D
activation_249: Activation
batch_normalization_257: BatchNormalization
batch_normalization_253: BatchNormalization
conv2d_250: Conv2D
batch_normalization_248: BatchNormalization
activation_253: Activation
batch_normalization_250: BatchNormalization
conv2d_254: Conv2D
activation_250: Activation
batch_normalization_254: BatchNormalization
conv2d_251: Conv2D
activation_254: Activation
batch_normalization_251: BatchNormalization


stage3_unit3_relu1: Activation
zero_padding2d_190: ZeroPadding2D
stage3_unit3_conv1: Conv2D
stage3_unit3_bn2: BatchNormalization
stage3_unit3_relu2: Activation
zero_padding2d_191: ZeroPadding2D
stage3_unit3_conv2: Conv2D
add_89: Add
stage3_unit4_bn1: BatchNormalization
stage3_unit4_relu1: Activation
zero_padding2d_192: ZeroPadding2D
stage3_unit4_conv1: Conv2D
stage3_unit4_bn2: BatchNormalization
stage3_unit4_relu2: Activation
zero_padding2d_193: ZeroPadding2D


decoder_stage0_concat: Concatenate
decoder_stage0a_conv: Conv2D
decoder_stage0a_bn: BatchNormalization
decoder_stage0a_relu: Activation
decoder_stage0b_conv: Conv2D
decoder_stage0b_bn: BatchNormalization
decoder_stage0b_relu: Activation
decoder_stage1_upsampling: UpSampling2D
decoder_stage1_concat: Concatenate
decoder_stage1a_conv: Conv2D
decoder_stage1a_bn: BatchNormalization

Curriculum Vitae

Personal Data

Name: Kavitha Shaga Devan

Year and place of birth: 1979, Ipoh

Work experience

02/2017 to 03/2023

PhD candidate
Ulm University, Germany

- Conducted research on deep learning methods for biological electron microscopy images.
- Developed and evaluated algorithms using deep learning and computer vision for the effective analysis and quantification of bio-medical images.
- Designed and implemented automated end-to-end predictive machine learning models to quantify viral and biological particles using electron microscopy data for life sciences applications that optimized workflows and improved efficiency.

02/2013 to 10/2014

Researcher
Linköping University, Sweden

- Developed and evaluated algorithms using machine learning and computational imaging for cancer diagnosis using medical imaging.
- Designed and implemented automated end-to-end predictive machine learning models to diagnose breast cancer stages using digital histopathological biopsy specimen images for optimization of pathologists' workflows and improved efficiency.

08/2011 to 08/2012

Researcher
Auckland University of Technology, New Zealand

- Developed and evaluated algorithms using computational imaging for skin cancer diagnosis using dermatological images.
- Conducted research on spectrophotometric intracutaneous analysis-based skin cancer diagnostics.

10/2011 to 06/2012

Research and development engineer
Fulton Hogan NZ, New Zealand

- Designed and implemented an automated end-to-end image analysis tool for the analysis of road surface textures that were used on the roads of Auckland, New Zealand.

- Worked on a collaborative project with a cross-functional team of engineers and image analysts from industry.

01/2007 to 06/2011

Lecturer (Engineering department)
UOW Malaysia KDU, Malaysia

- Created and evaluated lesson plans and course contents for electrical and communications engineering subjects to moderate classroom discussions and student-centered learning.
- Enhanced students' learning by optimizing a wide range of instructional approaches and innovative classroom approaches.
- Remote lecturing and lesson plan development for Northumbria University, United Kingdom.

02/2006 to 12/2006

Lecturer
HELP University, Malaysia

- Created and evaluated lesson plans and course contents for electrical and communications engineering subjects to moderate classroom discussions and student-centered learning.
- Enhanced students' learning by optimizing a wide range of instructional approaches and innovative classroom approaches.
- Remote lecturing and lesson plan development for Hertfordshire University, United Kingdom.

01/2003 to 12/2005

Research Officer
Universiti Teknologi PETRONAS, Malaysia

- Developed and evaluated computational imaging algorithms for the analysis of medical images for breast cancer.
- Designed and implemented an image analysis tool to detect cancer biomarkers in radiology images for cancer diagnosis to be used by radiologists in local hospitals.

Education

02/2017 till 06/2023

PhD in Computer Science
Ulm University, Germany
Area: Computer Science /Artificial Intelligence
PhD Thesis: Deep learning applications for biological electron microscopy

01/2004 to 01/2006

Masters of Science in Electrical & Electronic Engineering
University Teknologi PETRONAS, Malaysia
Major: Computer Systems
Research thesis: Expert system with an embedded imaging module for the diagnosis of lung diseases

05/1997 to 05/2003

Bachelor of Electrical & Electronic Engineering
University Teknologi PETRONAS, Malaysia
Major: Computer Systems

Skills

- **Programming:** Python, C/C++, MATLAB
- **Machine Learning:** Convolutional Neural networks, Deep learning, Classification, Detection, Segmentation, Clustering, Regression, MLOPs, Ensemble, Boosting, Random Forest, CI/CD, AutoML, Feature Engineering
- **Data:** Data analytics, Data Pipeline, Visualization, Feature engineering, Explanatory data analysis
- **Packages:** OpenCV, SciKit, Pandas, NumPy, Jupyter Notebook
- **Frameworks:** Tensorflow, Keras, PyTorch, AWS, Microsoft Azure, Amazon Sagemaker, Google Could Platform, MLFlow, Kubernetes, Git
- **Computer vision:** Computational imaging and analysis, Digital pathology, Image analysis for life sciences, Microscopy data analysis, Pattern Recognition, Algorithm development, Model development and deployment, Predictive modelling
- **Management:** Agile, Scrum, Lean

Patents

- Methods, systems and circuits for generating magnification-dependent images suitable for whole slide images. US20150055844A1. Issued Aug 9, 2016

Certifications

- Deep Learning Specialization, Deeplearning.AI, 2020
- Machine Learning Engineering for Production (MLOPs) Specialization, Deeplearning.AI, 2022
- Python for Data Science, AI & Development, IBM, 2020
- Google Project Management: Professional Certification, Google, 2022
- Agile Project Management, Google, 2022

Publications

- Weighted Average Ensemble-Based Semantic Segmentation in Biological Electron Microscopy Images. Histochemistry & Cell Biology. 2022
- Clean Implicit 3D Structure from Noisy 2D STEM Images, IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2022
- Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labeled images generated by a generative adversarial network. Cellular Microbiology. 2021
- Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. Histochemistry & Cell Biology. 2018
- Rotation Forest Ensemble Classification for Nuclear Pleomorphism Scoring in Breast Cancer. Poster Proceedings of Nordic Symposium on Digital Pathology. 2014

- Feature Enhancing Zoom to Facilitate Ki-67 Hot Spot Detection. Proceedings of SPIE Medical Imaging. 2014
- Expert System with an Embedded Imaging Module for Diagnosing Lung Diseases. Proceedings of 7th International Workshop on Enterprise Networking and Computing in Healthcare Industry (HEALTHCOM). 2005