Using Machine Learning Techniques to Understand the Biophysics of Demyelination

Ahmed Hany Mohamed Hassan Rezk

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the
Master of Science in Physics

Department of Physics
University of Ottawa

Abstract

Demyelination is the process where the insulating layer of axons known as myelin is damaged. This affects the propagation of action potentials along axons which can have deteriorating consequences on the motor activity of an organism. Thus it is important to understand the biophysical effects of demyelination to improve the diagnostics of its diseases. We trained a Convolutional Neural Network (CNN) on Coherent anti-Stokes Raman scattering (CARS) microscope images of mice spinal cord inflicted with the demyelinating disease Experimental Autoimmune Encephalomyelitis (EAE). Our CNN was able to classify the images reliably based on clinical scores assigned to the mice. We then synthesized our own images of the spinal cord regions using a 2D Biased Random Walk. These images are simplified versions of the original CARS images and show homogenously myelinated axons, unlike the heterogeneous nerve fibres found in real spinal cords. The images were fed into the trained CNN as an attempt to develop a clinical connection to the biophysical effects of demyelination. We found that the trained CNN was indeed able to capture structural features related to demyelination which can allow us to constrain demyelination models such that they include the simulated parameters of the synthesized images.

Acknowledgements

I would like to thank my supervisor Dr. Andre Longtin for his mentorship and for providing me with the chance to work on this project. I have learnt a countless amount of skills through this opportunity.

I would like to thank Dr. Daniel Côté (Université Laval, Department of Physics) for providing us with the image dataset used in this project. Without him we would not have been able to explore ideas with great potential.

I would like to thank Nicolas Brodeur for his constant participation and discussions with my supervisor and I on brainstorming ideas for my project.

I would like to thank the University of Ottawa for providing me with the environment to acquire my education, knowledge, and skills.

I would like to thank Dr. Andre Longtin and NSERC for providing the financial support required to investigate in this project.

Last but not least, I would like to thank my family for their financial and emotional support throughout my educational journey.

Contents

List of Figures

List of Tables

Chapter 1

Thesis Motivation

1.1 Introduction

Multiple Sclerosis (MS) is the highest prevalent neurological disease that affects humans and can lead to critical physical, cognitive, and neurological complications for young adults [1]. The disease is caused by an autoimmune disorder where the human's own immune system attacks the nerve fibres in the Central Nervous System (CNS), leading to the degradation of an insulating layer known as myelin and lesion formation [2, 3]. The lesions disrupt the structural integrity of the nerve fibres which lead to motor (i.e. movement) dysfunction [1].

MS has several symptoms and there exists many diagnostic techniques for disease detection [4], however, it is difficult to figure out how far the disease has progressed. It is useful to understand the severity of the disease in order to plan out the necessary treatments for patients. It is also complicated to properly diagnose MS since there is no single clinical feature that is enough to detect MS, which is the reason why the diagnostic criteria of MS includes both clinical and paraclinical (i.e. techniques that are not purely clinical such as radiology) studies [5, 6]. Thus, it is important to investigate what are the important clinical features of demyelination to better diagnose and characterize the disease.

In this thesis, we will be using an image dataset to extract the important and discriminative features of demyelination. The dataset consists of Coherent anti-Stokes Raman scattering (CARS) Microscope images that visualize mice spinal cord regions [7]. The mice are induced with an animal model of MS known as Experimental Autoimmune Encephalomyelitis (EAE) and the images show different degrees of de-

myelination. We decided to use a deep learning technique known as Convolutional Neural Networks (CNN) in this thesis as they have proven to be effective in the classification of microscope images [8–10].

The CNN would attain reliable classification results if the classifier model has indeed detected the discriminative features of demyelination and disease progression. To extract what the CNN has learnt from the dataset, we simulated our own images using a 2D Biased Random Walk simulation. The purpose behind this CNN-Simulation joint approach was to synthesize images in the context of the CARS images with simulation parameters, creating images with known physics encoded within them.

The joint approach could be a first step in the guidance towards the important features of demyelination which would assist in constraining complex demyelination models. Demyelination models are extensions of models that show the dynamics of the electrical activities of neurons, taking into account the damaging effects of the lesions. Using constrained models would add potential in designing new diagnostic methods for more accurate and feasible detection of MS, and thus narrowing the diagnostic criteria of MS and allowing for a better understanding of the disease severity and progression. Our goal for this thesis is to attempt to bridge the gap between the biophysics of demyelination and the clinical aspects of MS to establish more effective diagnostics of the disease.

1.2 Biophysics of Neurons

Neurons are the components responsible for transmitting information across the nervous system. Given an input stimulus, a neuron produces an electrical signal known as an action potential (AP) which can be conveyed to several neurons. The input stimulus is received by the dendrites of a pre-synaptic neuron and the AP propagates along the axon of the neuron until it reaches the terminal bouton. In this area, the AP stimulates the release of several chemicals known as neurotransmitters at the synapse. The neurotransmitters are packed into fluid-filled sacs known as vesicles. The synapse consists of the pre-synaptic cleft which the vesicles are released into. The neurotransmitters bind with the receptors of the post-synaptic neuron to initiate an AP in the axon of the post-synaptic neuron. Thus the post-synaptic neuron becomes a pre-synaptic neuron to several other neurons.

This transmission of APs starts from the peripheral nervous system (PNS) and continues to the central nervous system (CNS) for processing and then back to its final destination in the PNS to produce the required motor action. Suppose a person comes into contact with the surface of a hot object. The input stimulus is received by the person through temperature receptors on the surface of skin cells. The stimulus received by the receptors allow for the production of an AP which is passed along to a network of neurons. The brain, which also extends into the CNS, processes the information and further propagates the transmission of APs. The APs eventually return back to the PNS and reach the muscles at the contact point, stimulating the muscles to move away from the hot object. In this thesis, we are looking at the spinal cord of mice which is part of the CNS and is involved in the bidirectional transmission of APs between the brain and the PNS.

Figure 1.1 shows the time variation of the membrane potential of a typical neuron during the production of an AP. At equilibrium, the inside of a neuron has a negative membrane potential of approximately -70 mV compared to its outer region due to the presence of more Na^+ ions outside of the neuron than inside and to more K^+ ions inside than outside. This state is maintained by the Na^+/K^+ transporter found in the cell membrane which pumps three Na^+ ions out of the cell and two K^+ ions into the cell. Since more positive charges are present outside, the neuron is left with a negative membrane potential. Along with the Na^+/K^+ transporter, there are voltage-gated channels for both types of ions. These are ion transporting channels that are activated at certain membrane potential values and play an important role in AP production. Since more Na^+ ions are present outside of the neuron, there is a tendency for these ions to diffuse into the cell to achieve chemical equilibrium and vice versa for the K^+ ions.

The membrane potential needs to reach a threshold value which is approximately -55 mV to produce an AP. A subthreshold stimulus is not enough to initiate an AP while a threshold stimulus is able to. Once the threshold potential is reached, voltage-gated Na^+ channels open and the Na^+ ions diffuse into the cell. This further increases the membrane potential in a process known as depolarization. When the membrane potential reaches approximately +30 mV, the voltage-gated Na^+ channels start to close and K^+ voltage-gated channels open. This causes K^+ ions to diffuse out of the cell, decreasing the membrane potential in a process known as repolarization. There is a period known as the refractory period where the K^+ channels remain open

for sometime and decreases the membrane potential beyond the resting value. This mechanism is essential in preventing another AP from initiating and ensures that APs propagate in one direction.

Figure 1.2 describes the structure of a neuron. Typically, the dendrites are the sites where stimuli are received and the cell body, known as the soma, is where the voltage-gated channels are present. The APs travel across neurons through axons. Several regions of the axon are covered with an insulating layer known as the myelin sheath. An AP propagating along a myelinated axon loses its amplitude, similar to an electrical pulse travelling down a cable. In between each myelinated region, there is a structure known as the Node of Ranvier which is responsible for resetting the AP back to its original height, due to the presence of Na^+ and K^+ channels as in the soma. The myelin maintains a fast enough propagation velocity such that the electrical signal reaches its destination before decaying. If axons are not myelinated well enough, the signal could die out during its transmission and the required action, such as the initiation of a motor command, is not performed. Such is the case with the human autoimmune disease Multiple Sclerosis (MS) where the human's own immune system attacks the myelin surrounding the axons [2, 3]. This damage to the myelin is known as demyelination. The severity of demyelination is the determinant of how far the disease has progressed.

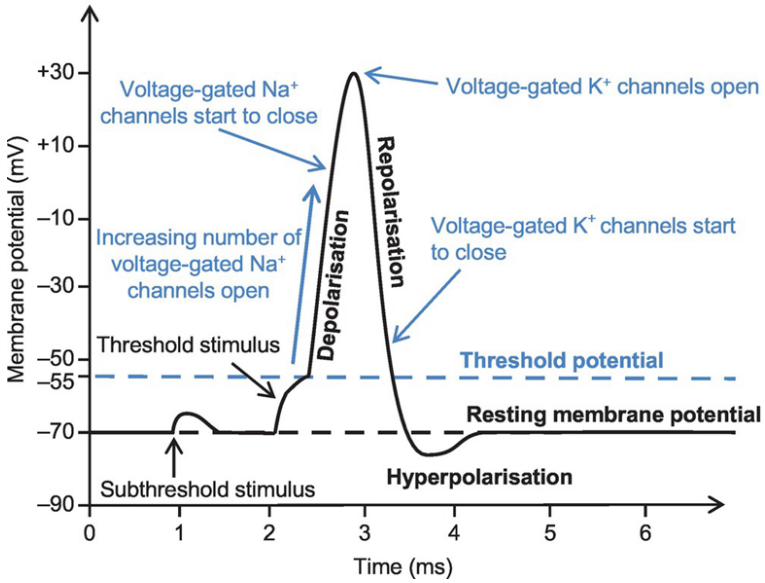

Figure 1.1: Action potential production in neurons. The graph shows the dynamics of the membrane potential along with labels of the mechanisms taking place during action potential initiation. Adapted from [11].

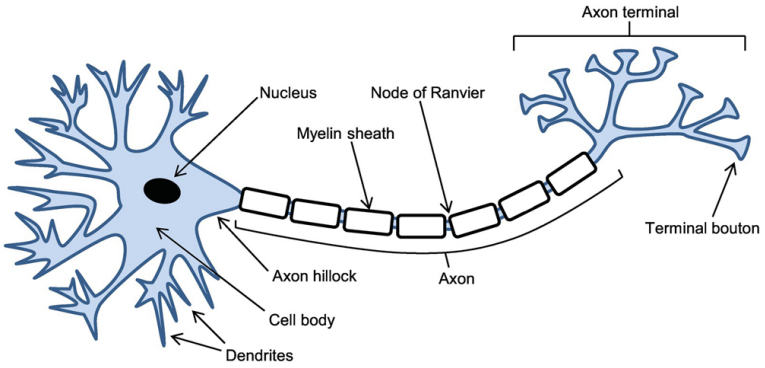

Figure 1.2: Structure of a Neuron. A single nerve consists of several parallel axons emanating from as many single neurons. Adapted from [11].

1.2.1 Saltatory propagation

Input stimuli are generally received at the dendrites of a neuron. The input stimulus activates voltage-gated channels which increase the membrane potential of the neuron, a process known as depolarization. A strong stimulus produces an AP that propagates along the axon. For effective propagation, the AP needs to be transmitted through a material with low resistance, i.e. a good conductor, a property that the axon possesses. We also want to prevent the loss of membrane potential caused by charges diffusing out of the axon transversally through the membrane to the external environment of the neuron. The myelin insulating layer has a high resistance so it restricts the transmission of the AP to propagate along the axon. It is possible for myelin to wrap around the axon in order to form multiple layers of myelin. This decreases the effective membrane capacitance since it is equivalent to multiple capacitors in series. Having multiple layers would also increase the resistance of the membrane.

If the entire axon were surrounded by a single myelin sheath, the signal would become significantly decayed by the time it reaches the Axon Terminal, which is the destination where the neuron transmits its AP to the dendrites of other neurons. Instead, there are several sections on the axon surrounded by the myelin sheath and in between each section there is a Node of Ranvier. Each Node of Ranvier contains voltage-gated channels which are stimulated by the incoming AP, thus allowing charges to flow into the axon and resetting the strength of the signal. This ensures signals with enough strength would reach other neurons and the AP that originated due to the input stimulus is able to propagate across the PNS and CNS and reaches its destination. Each section of the axon surrounded by a myelin sheath is known as a Schwann cell in the PNS and an oligodendrocyte in the CNS.

1.2.2 Integrate-and-Fire Model

We next discuss mathematical models of AP generation, with the eventual goal of incorporating into them some information about demyelination. The details of this section were obtained from [12]. The Integrate-and-Fire (IF) Model was first proposed by Lapicque in 1907 [13]. The IF model is a simplification of a neuron model and includes a few biophysical parameters. An AP is generated when the membrane potential of the neuron reaches a threshold value V_{th}. After the AP is produced, the membrane potential resets to a value V_{reset}.

AP models typically rely on the conductance of different voltage-gated channels, for example, Sodium (Na^+) or Potassium (K^+) gated channels to depolarize and repolarize the neuron during AP generation. Each channel has a membrane current which is positive when positive ions leave the neuron. The total membrane current per unit area i_m is the sum of each channel's membrane current. The membrane current of channel i linearly depends on its conductance g_j and the membrane potential V, obeying Ohm's law. Each channel has its own equilibrium potential E_j at which no membrane current flows through the channel. The equation of the total membrane current per unit area is:

$$i_m = \sum_j g_j(V - E_j),$$

where the sum is over different ion species. The IF model does not take into account the biophysical mechanisms of channels leading to APs, but only subthreshold currents. In its simplest form, the membrane current takes a single conductance and equilibrium potential value that is dependent on the cell being studied. In this case, the total membrane current per unit area would be:

$$i_m = g_L(V - E_L),$$

where g_L is the total leak conductance and E_L is the associated equilibrium potential. This IF model is referred to as a Leaky Integrate-and-Fire (LIF) Model. This model is suitable for cases when there are small fluctuations in the resting membrane potential as well as an approximation for the case of large fluctuations bringing the membrane potential to the threshold value. The neuron acts as a resistor and capacitor in parallel and thus the membrane potential with an input current I_e and neuron of area A can be found to be governed by the following equation:

$$c_m \frac{dV}{dt} = -g_L(V - E_L) + \frac{I_e}{A}.$$

The membrane resistance per unit area of the neuron r_m is related to the conductance through the equation $r_m = 1/g_L$. And the equation of the membrane time constant is $c_m r_m = \tau_m$. So it is useful to multiply the above equation by r_m to simplify the equation to the following where R_m is the total membrane resistance:

$$\tau_m \frac{dV}{dt} = E_L - V + R_m I_e.$$

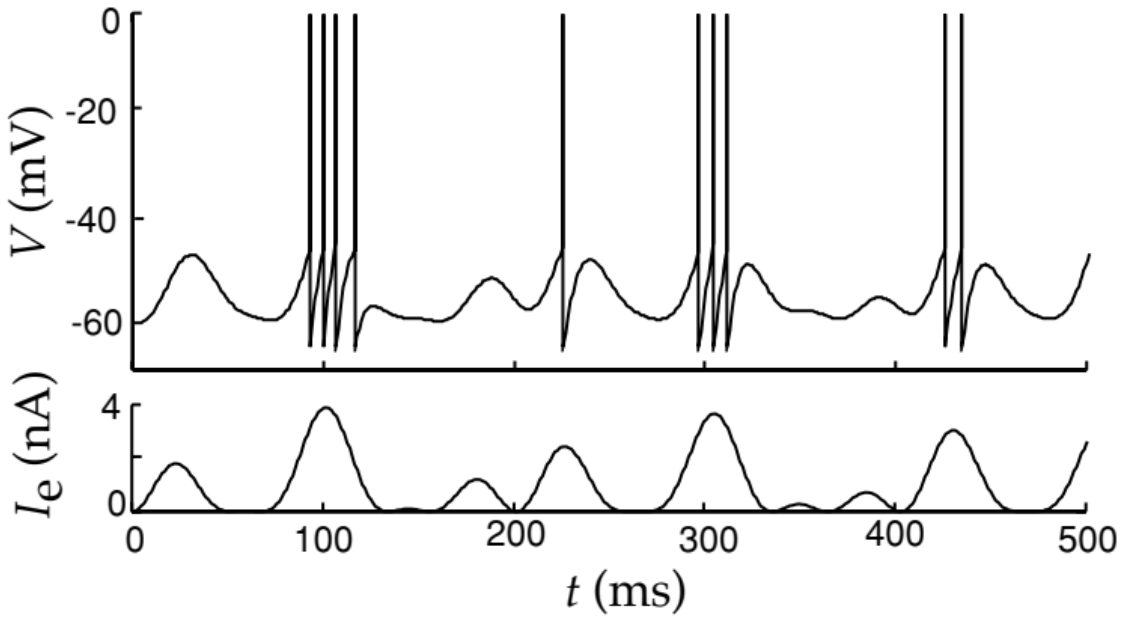

Figure 1.3: LIF model with the top graph showing how the membrane potential varies with time and the bottom graph shows the input current. The parameters of the model are $E_L = V_{reset} = -64mV$, $V_{th} = -50mV$, $\tau_m = 10ms$, and $R_m = 10M\Omega$. Adapted from [12]

1.2.3 Hodgkin-Huxley Model

The details of this section were obtained from [12]. A single K^+ channel involves four different identical subunits. All subunits need to undergo a structural change to allow the opening of the channel. In the Hodgkin-Huxley (HH) Model, the probability of one subunit changing is denoted as n. Thus, the probability that all four subunits will change is n^4. Such a K^+ channel is known as a delayed-rectifier.

The Na^+ channel has an activation variable m and an inactivation variable h. They describe the probability that the channel will be open for Na^+ ion flow. Like n, these variables are functions of the voltage V. In the HH model, the probability that the Na^+ channel will be open is m^3h. The variables have opposite voltage dependencies i.e. m increases and h decreases with depolarization, while a decrease in membrane potential (known as hyperpolarization) causes the opposite. This type of Na^+ channel is known as a transient channel.

Unlike the LIF model, the HH model attempts to capture the biophysical mechanisms of the K^+ and Na^+ channels. Thus in our membrane current equation, we would have a term for the K^+ and Na^+ channels and a third term similar to that in the LIF model which takes into account the current leakage of the remaining channels in the neuron:

$$i_m = g_L(V - E_L) + g_K n^4(V - E_K) + g_{Na} m^3 h(V - E_{Na}).$$

The biophysical modelling of the channels already adds complexity to the coupled dynamics of the model since now we need to solve not only for the membrane potential, but also for the three probability values n, m, and h. But with more complexity, we observe a more detailed and realistic behavior of AP production.

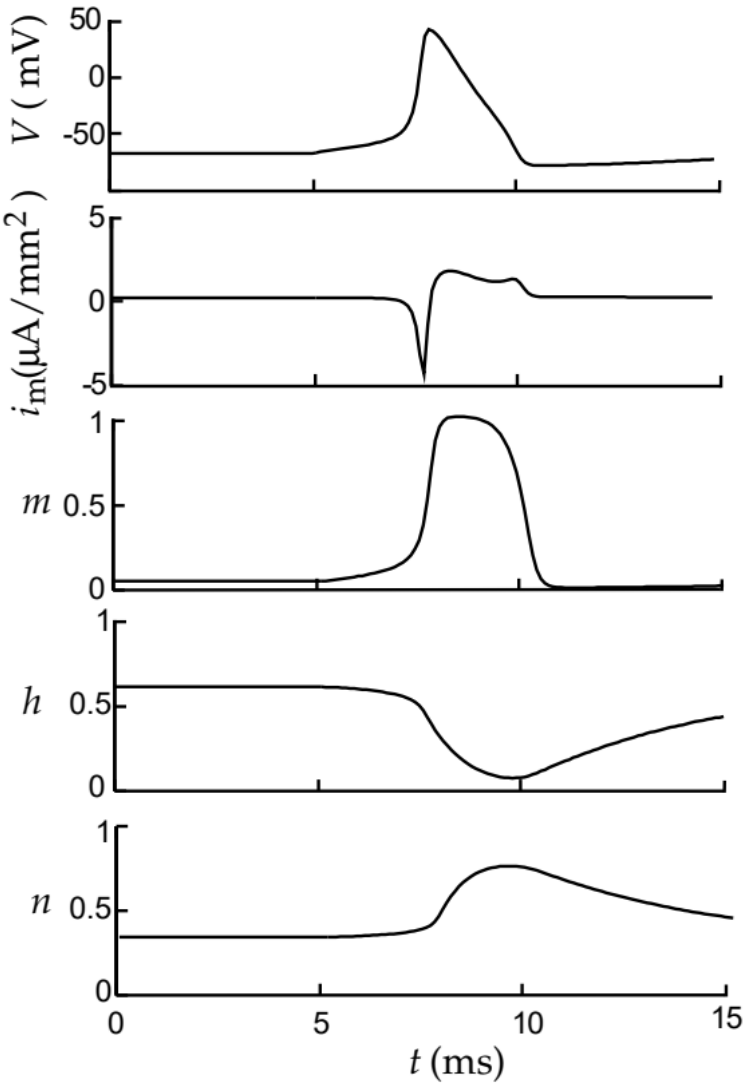

Figure 1.4: HH model with the dynamics of V, m, h, and n. i_m is the total membrane current produced by the sum of the HH K^+ and Na^+ conductances. Input current was injected at $t = 5ms$. Adapted from [12]

1.2.4 AP Propagation

In the neuron models discussed, membrane conductance is simplified and as such the models are known as single-compartment models. When complexities are to be included, the neuron needs to be split into different compartments and the model becomes a multi-compartment model. Each compartment should be small enough such that the membrane potential does not vary considerably across it. We will discuss how multi-compartment models can describe AP propagation. Since we are interested in demyelination, we look at how propagation differs between unmyelinated and myelinated axons to have an intuition on how the propagation velocity is affected. The details of this section were obtained from [12].

1.2.4.1 Propagation along an Unmyelinated Axon

In this example, a neuron is divided into compartments having the same membrane conductances as the single-compartment HH model. APs typically propagate along the axon away from the soma, a process known as orthodromic propagation. Antidromic propagation takes place in the reverse direction and would cancel out with the orthodromic propagation which, for example, could happen in the middle of the axon. After a compartment fires an AP, partial inactivation of Na^+ channels leads to a refractory period where another AP cannot be initiated in either direction. By the time the refractory effects disappear, the AP would have travelled far enough that the compartment cannot initiate an AP.

The propagation velocity v_{prop} along an unmyelinated axon is proportional to the ratio between the electrotonic length constant λ and membrane time constant τ_m. The electrotonic length constant depends on the membrane resistance per unit area r_m, intracellular resistivity r_L per unit area, and axon radius a as follows:

$$\lambda = \sqrt{\frac{ar_m}{2r_L}}.$$

The ratio mentioned above would then be:

$$v_{prop} \propto \frac{\lambda}{\tau_m} = \sqrt{\frac{a}{2c_m^2 r_L r_m}}.$$

The propagation velocity is proportional to the square root of the axon radius and thus grows slowly with the thickness of an unmyelinated axon. For living organisms, axons are typically not thick enough to allow a sufficiently high conduction velocity. As we will discuss in the next section, the insulating layer myelin can speed up AP propagation.

1.2.4.2 Propagation along a Myelinated Axon

Propagation along a myelinated axon takes place in the saltatory behavior discussed in section 1.1.1. Myelinated regions allow for faster propagation and Nodes of Ranvier reset the AP. Axons are treated as coaxial cables with inner core radius a_1 and outer radius a_2. The thickness of a single layer of cell membrane is d_m. If the axon is treated as a cylinder then $C_m = 2\pi c_m d_m L$. The length of the axon is L and x describes a coordinate point along the axon assumed to be 1D structure. Using the membrane capacitance C_m, the propagation can be described in the form of a linear cable equation:

$$\frac{C_m}{L}\frac{\partial v}{\partial t} = \frac{\pi a_1^2}{r_L}\frac{\partial^2 v}{\partial x^2} \quad .$$

The equation has the same format as the diffusion equation where the diffusion constant D is:

$$D = \pi a_1^2 L/(C_m r_L) = a_1^2 \ln(a_2/a_1)/(2c_m r_L d_m).$$

The derivative of the diffusion constant with respect to the inner radius is:

$$\frac{dD}{da_1} = a_1(2\ln(a_2/a_1) - 1)/(2c_m r_L d_m).$$

If we set the derivative to zero, we are left with the inner and outer radius:

$$2\ln(a_2/a_1) - 1 = 0.$$

By rearranging, we get:

$$a_1/a_2 = e^{-1/2} \approx 0.6.$$

The ratio between the inner and outer radius is referred to as the g-ratio. Thus, according to the solution to the cable equation, the ideal g-ratio is 0.6 as it leads to the maximal diffusion constant D. There exists literature that investigates the ideal g-ratio [14–17] and research has shown that the ideal g-ratio could fall between 0.5 and 0.7 [18, 19]. The g-ratio is an important feature that will be studied in this thesis. We were able to observe the ideal g-ratios for axon diameters which will assist in understanding when would demyelination become critical to AP propagation.

We will now discuss previous work done on modelling demyelination and how important it is to constrain such models to focus on fewer and more impactful parameters. Model constraining will help in understanding the physical effects on myelin health and AP propagation.

1.2.5 Demyelination Models

We have seen how neurons and AP propagation can be mathematically modelled. It is possible to simulate these models with different values of axon diameters and myelin thickness, i.e. to simulate AP propagation along axons with different g-ratio values. Since demyelination involves thinning out the myelin surrounding the axons, it is possible to perform simulations with varying myelin thickness and observe how the electrical behavior of AP propagation changes. An important missing piece in the modelling is the association between the damage inflicted on the myelin sheath and the clinical implications of such demyelination.

An interesting recent model of demyelination is based on cable theory and Green's functions for propagation [20]. The model makes use of changes in the electrotonic length constant of two adjacent myelinated segments: the preceding segment is the antidromic internode and the second segment is the orthodromic internode. Simulations were performed to compare the electrical activity of AP propagation. The results show the activity of a myelinated antidromic internode and a demyelinated orthodromic internode, demyelinated antidromic and orthodromic internodes, and a demyelinated antidromic internode and myelinated orthodromic internode. The results have shown that the electrical activity is significantly affected in the second case where both internodes are demyelinated. The first case shows an increase in membrane potential spiking and the third case shows an insignificant difference from the regular uniformly myelinated axon.

In living systems, demyelination is not uniform and it is not clear at what level of severity would the disease become clinically significant. The AP model in [20] is capable of accommodating real-world data such as myelin thickness. However, we are unsure which features of the real-world data are relevant to demyelination since we would need to first simulate the effects of the features and observe if they have an impact on the electrical activity of neurons. By constraining demyelination models, we can switch our focus to fewer and more important parameters such that the task of understanding the consequences of demyelination is made simpler.

There exists a demyelination dataset of microscope images that could help us constrain demyelination models. In this thesis, we try to uncover the important features by studying these microscope images. They are obtained from longitudinal nerve fibres found in the spinal cord of mice that are assigned a clinical score based on the severity of their demyelination. The features should reflect structural aspects of demyelination. Finding such features would allow us to have a better clinical understanding of demyelination and better capabilities of disease diagnosis.

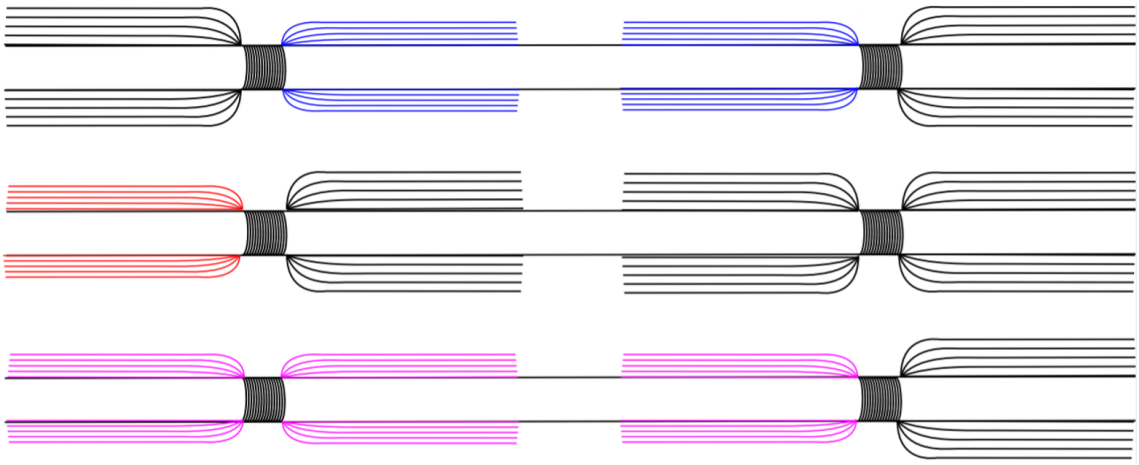

Figure 1.5: Schematic of different demyelination cases. Axons are shown in black, blue, red, and purple. In between each axon is a Node of Ranvier. Black axons have normal myelination. The blue axon is a demyelinated orthodromic internode. The red axon is an antidromic internode. The purple axons are demyelinated antidromic and orthdromic internodes. Electrical activity propagates from left to right. Adapted from [20]

.

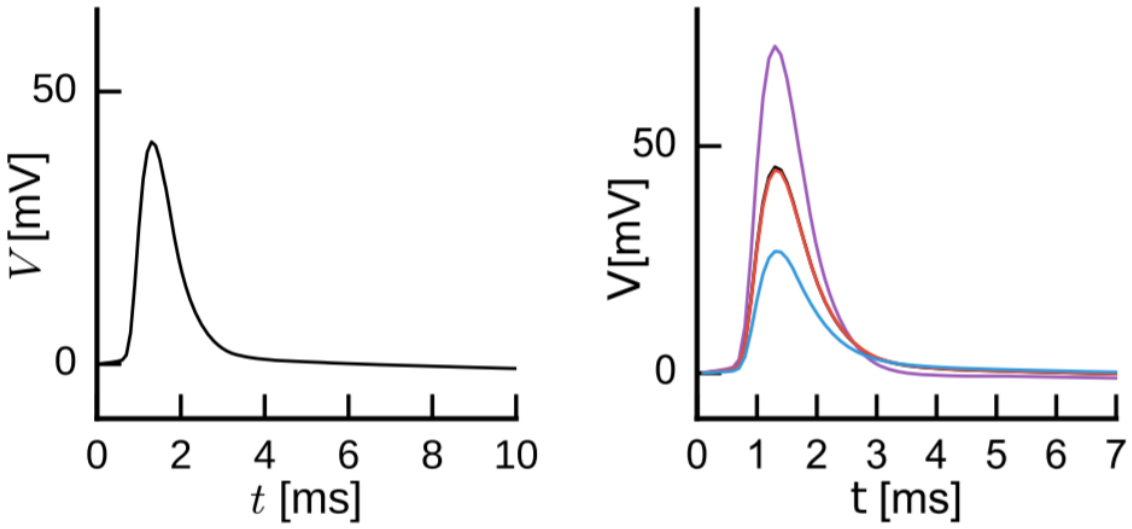

Figure 1.6: AP Propagation using Green's function method. The panel on the left shows the electrical activity excitation on the left side of the left-most Node of Ranvier from Figure 1.5. The panel on the right shows the electrical activity excitation on the right side of a Node of Ranvier. Colored curves in the right panel represent the different cases shown in Figure 1.5. Adapted from [20]

.

1.3 Dataset Description

We first describe the technique leading to the images in our dataset were taken. Coherent anti-Stokes Raman scattering (CARS) Microscopy is a label-free technique that is capable of real-time imaging of living cells and organisms using molecular vibrational spectroscopy [21]. CARS was first used at the Ford Motor Company in 1965 by Maker and Terhune [22]. Its first usage to image spinal cord tissue was reported by Wang *et al.* [23]. CARS has made use of the lipid content in myelin to provide internal contrast of the structure [21, 24, 25]. Using a wave-mixing process, a pump beam at frequency w_p and a Stokes beam at frequency w_s interact with the microscope sample. An anti-Stokes signal of magnitude equal to $2w_p - w_s$ is produced once the beat frequency $w_p - w_s$ is equivalent to the frequency of a Raman active molecular vibration [21].

Since myelin plays a critical role in allowing organisms to perform essential basic functions such as locomotion, it is important to understand the consequences of demyelination on the biophysics of neurons and construct a link between the physics and clinical implications of the disease. To study demyelination, we use an image dataset of mice spinal cord induced with Experimental Autoimmune Encephalomyelitis (EAE) [7] which is a suitable animal model for the human disease [26, 27]. The images are of size 500x500 and are acquired using CARS. The following methods are adapted from [7].

The mice were split into two control groups and four groups with varying degrees of the EAE disease. The control groups include mice administered with either Saline (Sal) or complete Freund's adjuvant (CFA). The CFA consisted of incomplete Freund's adjuvant and killed mycobacterium tuberculosis H37Ra (2 mg/mL). Along with CFA, the mice were also administered with pertussis toxin (PTX, 0.7-10 μ g/mL of phosphate buffer solution (PBS)). The CFA control group were used to study the effects of both CFA and PTX on myelin. The third group consisted of mice on which clinical scores were assigned based on their neural activity. The score (Sc) of a mouse shows the clinical progression of the disease and is given on a scale of 0 to 5 (Sc0-Sc5). The following is the breakdown of the clinical score assignment: Sc0 = no signs, Sc1 = limp tail, Sc2 = limp tail and weakness in hind legs, Sc3 = limp tail and paralysis of hind legs, Sc4 = limp tail, paralysis of hind legs and weakness of front legs, Sc5 = complete paralysis of both hind and front legs. Only scores of up to Sc3 were observed

in the mice dataset possibly due to the progressive clinical limitations of the EAE disease.

The orientation of the nerve fibres was analyzed to investigate the correlation between neighbouring domains of the images in the dataset. Healthy images are expected to have similar nerve fibre orientations across the spinal cord region, i.e. there would be a strong correlation between the orientations of different image domains [28]. The Fourier transform of an image domain produces an elliptical shape of a specific orientation and aspect ratio (AR). This analysis has been previously used to study microscope images of collagen tissue [29–31]. The AR of the ellipse is the ratio between its minor and major axis. Image domains with randomly oriented nerve fibres would show an ellipse with an AR close to 1, indicating that the ellipse is close to being a circle. The AR is used as a measure of the uncertainty of the fibre orientation, therefore the closer the AR is the value of 1 the more disordered the spinal cord region is.

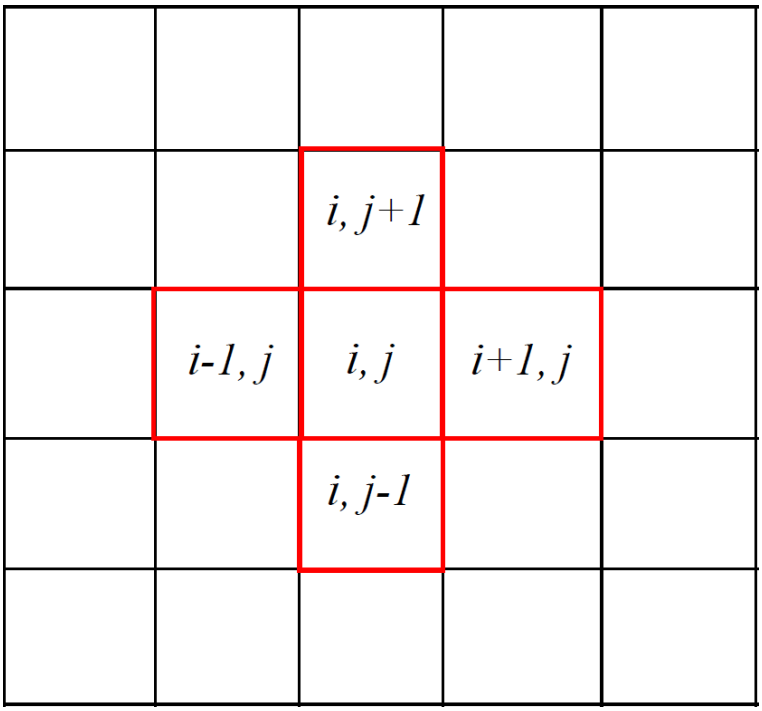

Figure 1.7: Schematic of image domain i, j and its neighbouring domains. The domains are the regions with red outlines.

The orientation of the domain θ_i is the angle of its ellipse in the Fourier domain. The correlation of the orientation between an image domain *i,j* and its neighbour *i+1, j*, i.e. the collinearity between them, is:

$$C_{i\rightarrow i+1} = 2[\cos^2(\theta_{i,j} - \theta_{i+1,j}) - 0.5].$$

The collinearity is strongest for perfectly parallel domains ($C_{i\rightarrow i+1} = 1$) and weakest when the domains are orthogonal to each other ($C_{i\rightarrow i+1} = -1$). The correlation parameter (CP) describes the collinearity between the domain *i,j* and its four neighbouring domains. This is modelled as the average of the four collinearities:

$$CP_{i,j} = \frac{C_{i\rightarrow i-1} + C_{i\rightarrow i+1} + C_{i\rightarrow j-1} + C_{i\rightarrow j+1}}{4}.$$

The CP could then be generalized as follows where nb represents the coordinates of the four possible neighbours and N_{nb} is the number of neighbours:

$$CP_{i,j} = \frac{1}{N_{nb}} \sum_{N_{nb}} 2[\cos^2(\theta_{i,j} - \theta_{nb}) - 0.5].$$

The value of CP is in the range of ± 1. When $CP = 0$, the orientations of the domains are random and are uncorrelated with each other. To better detect lesions, the uncertainty provided by the AR is used to compute the corrected correlation parameter (CCP):

$$CCP = CP[1 - AR].$$

For a low uncertainty of the fibre orientation (low AR), the value of CP is not affected significantly. If the uncertainty is high (high AR), then a correction could be applied to a high value of CP using a correction factor $(1 - AR)$. Without the correction, the analysis could indicate that a region is healthy despite the existence of lesions. Figure 1.8 shows the Fourier transform of three different images. Elliptical fits are applied to measure the AR of each image. Fits that are of a closer resemblance to an elongated ellipse are expected to have a low AR and vice versa for fits that show a more circular shape.

In figure 1.10, image domains that have low uncertainty and strong correlations with their neighbours would be considered healthy as they would have a larger value for the CCP while high uncertainty and weak correlations are the domains that are afflicted the most by lesions and have a lower CCP. The CCP is averaged across all image domains of a single mouse from each group. For each group, two mice were used to compute the mean and median CCP and to plot the distributions in Figure 1.9. The domains of the healthiest mice were observed to have a mean and median CCP of approximately 0.75 while the mice with the most severe demyelination seem to have a mean and median CCP close to 0.5.

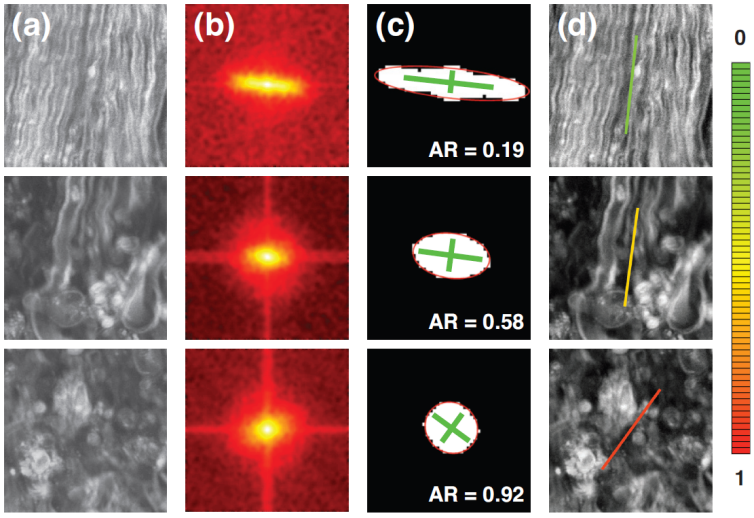

Figure 1.8: (a) Three different spinal cord regions with a progressive increase in EAE lesions from top to bottom. (b) Fourier Transform of each spinal cord region. (c) Segmentation of Fourier Transform fitted with an ellipse with the corresponding minor and major axes. (d) Color-coded fibre orientation superimposed on original images. Adapted from [7].

1.4 Deep Learning Interpretation of Microscope Images

There could be more information encoded in the images, other than what the AR and CCP were able to capture, that is correlated with disease severity. The goal of this thesis is to train a Convolutional Neural Network (CNN), which is a deep learning method for image classification [32], that can detect and extract the important image features that uniquely define the severity of demyelination. CNNs use supervised learning to classify the images, i.e. the algorithm is given labels for the images and the model attempts to learn how to correctly classify the labels. This is the subject of chapter 2. The labels will include the two control groups Sal and CFA as well as each observed clinical score in the mice EAE experiment (Sc0 to Sc3). Thus, a total of six labels will be used.

Performing classification on these six labels could prove to be a challenging task especially since some labels do not show a clear distinction between each other. For example, it is simple to visually classify a Sal image from a Sc3 image but it is difficult to figure out the differences between a Sc0 and a Sc1 image which is the onset of the disease. In fact, it would be expected that most of the misclassifications would be performed on the Sc1 images since the onset of the disease might have a range of demyelination severity.

It is also possible to synthesize our own images as spatial patterns simulated using a 2D Biased Random Walk [33–35]. A biased random walk is selected since it can be clearly seen that the nerve fibres have a preference to form a path in the positive y direction. By encoding images with known physics using different simulation parameters, we can feed the synthesized images into our reliable classifier model. This is the subject of chapter 3. Examples of simulation parameters used were the axon diameter, myelin thickness, and degree of disorder. The CNN model evaluates the synthesized images and based on the classification results, we can understand the consequences of demyelination on the structural integrity of nerve fibres in the form of a random walk model. This is a work in progress method as the synthesized images are far from being identical to the real microscope images. This method only attempts to encompass the main features of the spatial patterns observed in the microscope images but does not take into account the noise and varying pixel intensity values. The synthesized images are binary images, i.e. the pixel value is either 255 or 0 which

means that the generated nerve fibres are perfectly planar.

The physical parameters of the random walk attempt to model different structural features. After understanding the underlying relationships between the simulation parameters and the severity of demyelination, it is possible to incorporate the values of the parameters into mathematical models that describe AP propagation. A connection can then be developed between the clinical scores and the electrical activities within the nervous system. This connection can be applied in a clinical setting for humans diagnosed with MS. It would be possible to indicate how far the disease has progressed using the electrical measurements performed by medical devices. The random walk model aims to evaluate the healthiness of the synthesized spinal cord regions and analyze how the healthiness correlates with the simulation parameters used to generate the images. We aim to see how much can tortuosity explain the CNN results in the context of homogeneous axon diameter and myelin thickness. Tortuosity is the ratio of the actual nerve fibre length to the displacement between the end and starting points.

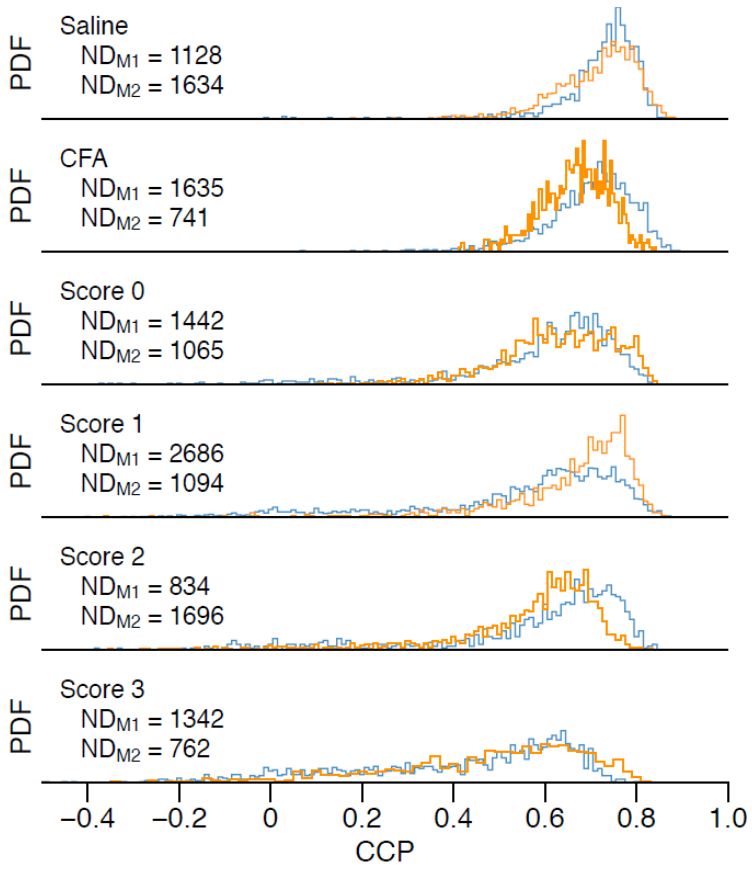

Figure 1.9: Distribution of CCP values across the six different groups. Images were collected from two mice (M1 and M2) for each group. The variable ND shows the number of images collected from each mouse. The width of the distribution increases as the severity of demyelination increases. Adapted from [7].

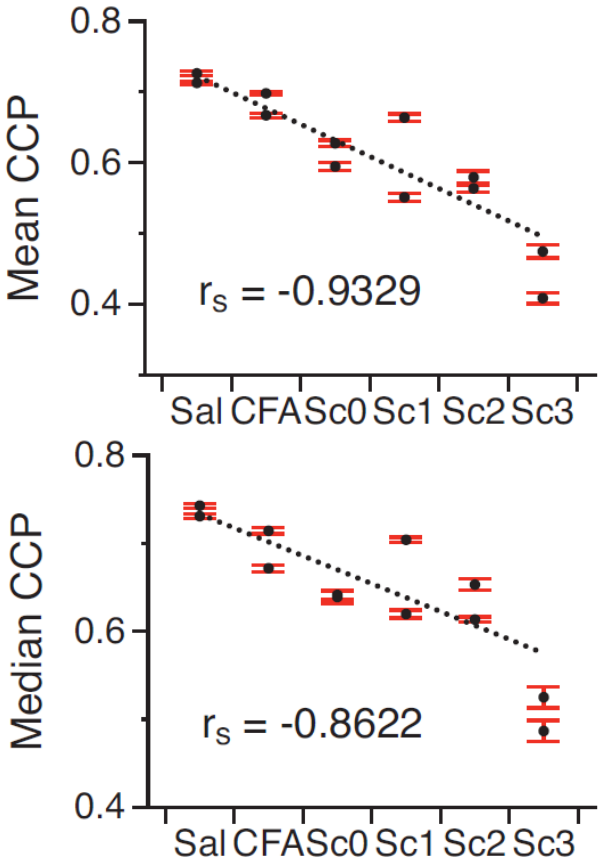

Figure 1.10: Mean and Median CCP plots for all six mice groups. Both metrics decrease as the severity of demyelination increases. Both plots show a strong negative correlation value for r_s. Adapted from [7].

Chapter 2

Deep Learning and Neural Networks

2.1 Artificial Neural Networks (ANNs)

2.1.1 Simple Perceptron

The most fundamental form of an artificial neural network is the Simple Perceptron from the work of McCulloch and Pitts [36]. The Simple Perceptron is based off of a biological neuron. A neuron consists of the dendrites, soma and axon. The dendrites receive input stimuli from the neuron's environment. The soma performs a weighted sum of these inputs. Two neurons can communicate together through a synapse. The synapse is the structure where a presynaptic neuron passes its electrical pulse to postsynaptic neurons. The strength of the connection between a presynaptic and postsynaptic neuron is the synaptic weight. The weight determines the strength of the connection between both neurons. If the stimulus is strong enough, the neuron is able to produce an electrical pulse which is then transmitted along its axon to other neurons. In deep learning, this is referred to as an activation. To avoid confusion, we refer to a deep learning neuron as a node throughout this thesis. Figure 2.1 shows a schematic of a Simple Perceptron.

The Simple Perceptron consists of three different layers of nodes: the input layer, the hidden layer, and the output layer. The number of layers L of a Simple Perceptron is 3. In a Simple Perceptron, a node is used to receive a set of data from an input layer through weighted connections and allows data to flow in one direction; this is referred to as a feed-forward neural network. The weighted connections determine the

strength and type of influence the input will have on the node's output. The strength of the input is determined by the magnitude of the connection's weight, with a large magnitude indicating that the input will have a strong contribution to the output. The influence can be of two types: positive or negative. This is analogous to biological neurons which can receive inputs that can either excite or inhibit the inactivity. In the case of the Artificial Neural Network, a positive weight excites the node and a negative weight inhibits the node.

The Simple Perceptron performs a linear combination of each input with its respective weighted connection to produce an output z as follows:

$$z = \sum_{i=1}^{N} w_i x_i + b.$$

The value of z is then passed onto the output layer for further computation using an activation function $\sigma(z)$. A node in layer L is activated by using the value of z as an input:

$$a^L = \sigma(z).$$

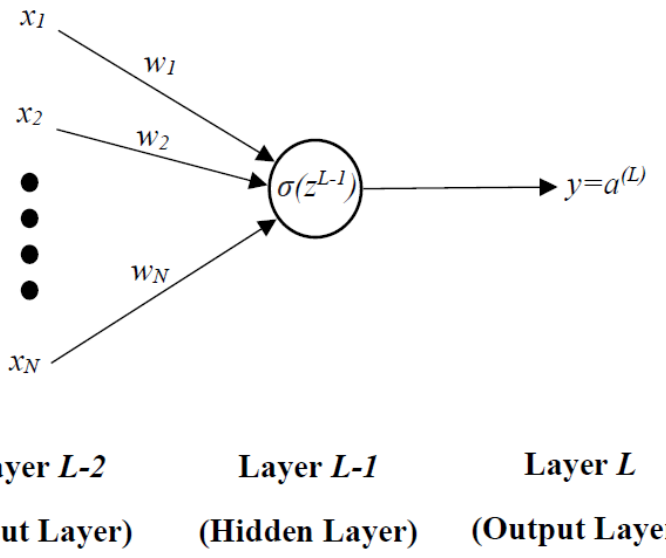

Figure 2.1: A Simple Perceptron with $L=3$ layers based on the McCulloch-Pitts model of a neuron receiving N different inputs and producing a single output y using an activation function $\sigma(z)$.

It is possible that $a^L = z$. Nodes that produce outputs in such a fashion have a linear activation function. An activation function allows the node to capture the patterns within the data received from the input layer. The performance of an ANN depends on the activation functions used as well as the network connectivity. It is interesting to note that a Simple Perceptron with a linear activation function is learning to perform a Univariate Linear Regression given that there is a single input, i.e. $N = 1$. If there exists more than one input, the network would learn to perform a Multivariate Linear Regression.

It is good practice in deep learning to use nonlinear activation functions. This allows nodes to capture more complex patterns in the data which are typically nonlinear in nature. Examples of nonlinear activation functions include the Sigmoid, Hyperbolic Tangent, and Rectified Linear Unit (ReLu) functions. Figure 2.2 shows examples of activation functions. Activation functions allow the output of a node to be mapped between 0 and 1 or -1 and 1. Depending on the deep learning problem, different activation functions could lead to different learning trajectories. However, it has been shown that it is highly likely that a properly trained model can be obtained with different activation functions [37]. Obtaining a reliable model typically depends on which activation function can capture the important patterns and features from the data in an easier manner, for example, by having less computational time, less layers, or a smaller width. In some cases, a linear activation function can perform well enough on the deep learning task.

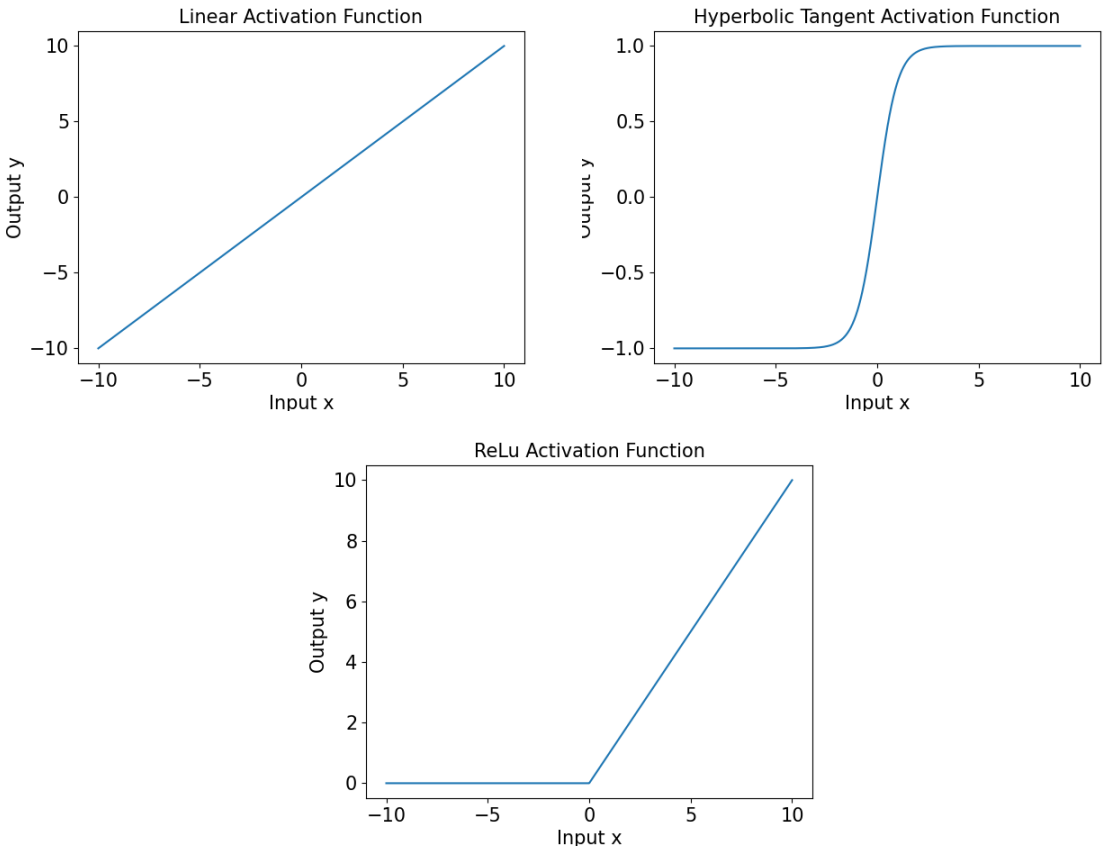

Figure 2.2: Different types of activation functions. Input values are equivalent to output values for the linear activation function. For the hyperbolic tangent activation function, the output y is computed using the equation: $y = tanh(x)$. The ReLu activation function converts negative values to zero and positive values are linearly mapped onto the output.

2.1.2 Multilayer Perceptron (MLP)

Since the Simple Perceptron is only restricted to a single node in a single hidden layer, it is limited on the amount of features that it can learn and represent. In an MLP, it is possible to have more than one node in the hidden layer. This can significantly improve the neural network's performance as each node is able to detect its own unique feature. MLPs have proven to be effective in approximating any smooth function [38] and perform better than other traditional statistical techniques for classification [39]. Figure 2.3 shows a schematic of an MLP. Throughout the learning process of the neural network, each hidden layer node will have connections of different weights with each input node. In this type of ANN, we can control how deep the network is, i.e. how many hidden layers the neural network will have.

Having multiple hidden layers allows the neural network to breakdown learned features in between each subsequent hidden layer. The features detected in the earlier hidden layers are known as shallow features i.e. they learn a representation of the features that is difficult for the user to interpret. The features found in deeper hidden layers, known as deep features, are represented in a way that simplifies the learning process for the neural network. Usually, each node of one layer (including the input layer) is connected to each node of the following layer (including the output layer). This type of connection is known as a Fully Connected Layer or a Dense Layer. An MLP neural network consisting of these layers is referred to as a Fully Connected Network (FCN) or Dense Network.

Not only can we increase the number of hidden layers, we are also able to adjust the number of nodes in each hidden layer. Since each node is able to learn a specific feature from its preceding layer, increasing the number of nodes allows the neural network to learn more features in a specific layer. The number of nodes found in each hidden layer determines the width of the neural network: the greater the number of nodes, the wider the neural network layer is. Deep features typically require more width compared to shallow features since deep features are more abstract and will require a more complex representation.

Although increasing the depth and/or the width of a neural network can improve the performance of the MLP, this comes at the cost of computational speed of the deep learning algorithm. It is important to balance the performance of a deep learning algorithm and the amount of time it takes for the neural network to achieve the required performance. This comes with trial and error by adjusting the depth and the width of the neural network, which is referred to as the art of building a neural network architecture. The parameters that determine the architecture of a neural network, for example the depth and width, are known as hyperparameters.

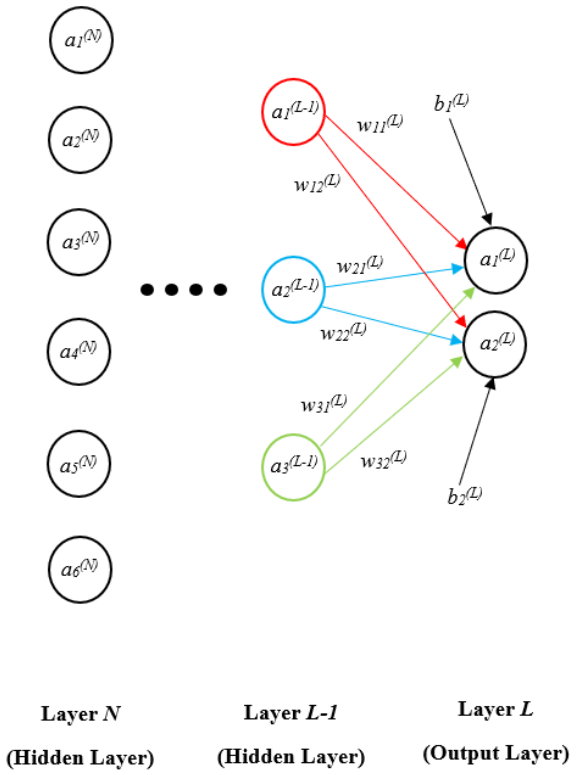

Figure 2.3: Part of an MLP Neural Network with L layers. The figure shows the *Nth* layer having 6 nodes with their respective activations. Layer *L-1* has 3 nodes with their weighted connections shown in red, blue, and green. The bias connections are only shown for the output nodes but exist in all nodes (and may equal to zero for certain nodes). The superscripts represent the layer number of the activations, weights and biases. The subscripts represent the node that the activations and biases belong to. The weights from a preceding layer to the next layer are in the form w_{ij} where i represents the node of the previous layer and j represents the node of the next layer.

2.2 Convolutional Neural Networks (CNNs)

2.2.1 Introduction to CNNs

An MLP is effective in deep learning tasks that require classification or regression of input data up to a certain dimensionality of the input data. However, in tasks that involve image classification or regression, the dimensionality of the input data can become extremely large. Suppose we have an ANN receiving input images where each image has a size of 50x50 pixels. The dimensionality of the input would be 2,500 i.e. we would need to define 2,500 nodes in the input layer of the ANN. This would be very computationally expensive. To tackle such a problem, one can use a CNN which aims to extract the important features of an image to decrease the dimensionality of the original image. It is common to see wide and deep CNNs when solving image classification problems [40–45]. Previous work on microscope images has shown the effectiveness of CNN classification [8–10]. This is achieved using a series of Convolution and two-dimensional Max Pooling layers (defined below). The resulting image with smaller dimensionality is then fed into an FCN after being flattened into a 1D vector representation of the feature maps. This flattening is performed using the Flatten layer.

2.2.2 The Convolution Layer

The Convolution layer is responsible for extracting the important features of the original image and separating each feature into its own image channel where each channel is known as a feature map. Although this initially increases the dimensionality of the original image, the 2D Max Pooling will reduce it to a value lower than it was initially.

The Convolution layer learns and extracts a feature using a two-dimensional filter (also referred to as a kernel). The filter will have a defined number of nodes. Each node consists of weight values that are adjusted during the learning process of the CNN. The filter acts as a sliding window and performs a linear operation on the input image as it slides across different regions of the image using the following equation:

$$z = \sum_{j=1}^{C} \sum_{i=1}^{R} w_{ij} x_{ij} + b.$$

R and C are the number of rows and columns of the filter, respectively. Since the weights of a single filter are fixed throughout the Convolution operation, the filter searches for its learnt feature across the entire image.

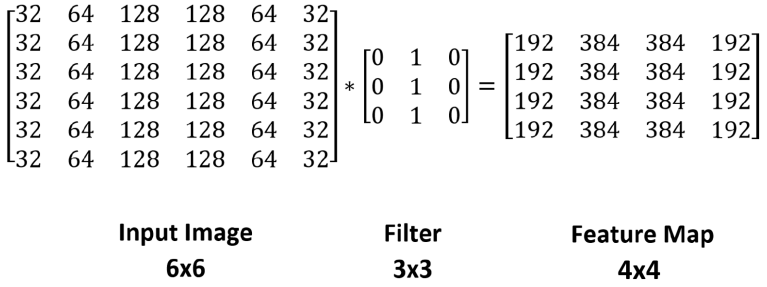

Figure 2.4: A vertical line filter performing a Convolution on an input image in 3x3 regions. The filter reduces the dimensionality of the input image and the feature map shows the activated regions.

Computers represent images in the form of a two-dimensional matrix. The matrix serves as a coordinate plane for different pixel values. The pixel values describe the intensity of the image in different regions. Since we are dealing with grayscale images, all our images have one channel, i.e. they only require one matrix to represent the pixel values.

The regions of the image where the filter detects the feature are thus activated and are represented on the feature map. The stronger the activation is, the more prominent the feature is. In Figure 2.4, a vertical line of width 2 pixels (the two columns filled with the pixel value 128) is seen in the input image. The filter is designed to detect vertical line features. The feature map now shows vertical lines detected in the input image. The output of the Convolution operation in Figure 2.4 is activated using a linear activation function. It is possible to try one of the other previously mentioned activation functions if the representation of the features needs to be improved.

There are hyperparameters associated with each Convolution layer. Firstly, there is the number of filters which is analogous to the number of nodes in a hidden layer. This determines the number of feature maps that the Convolution layer can extract from the input image. As the number of the filters in a Convolution layer increases, more features can be detected from the input image. The second hyperparameter is

the filter size (also referred to as kernel size). The filter size determines the size of the region where the linear operation is applied to detect features in the input image. A size of 3x3 was used for the convolution filters in this thesis. These hyperparameters also affect the computational speed of the algorithm: having more filters and/or larger filters will add on to the computational cost. Finally, it is important to note that in our model, we applied zero padding to each convolution layer. Zero padding adds additional rows and columns containing zero pixel values to the input image such that the size of the feature map matches the original size of the input image [46–48].

2.2.3 The 2D Max Pooling Layer

The feature maps produced after the Convolution layer are different representations of the important features detected by the Convolution layer. This increases the dimensionality of the data, however, this is a required step which will allow the CNN to pick out the dominating feature in a specified region. The output of the Convolution layer consists of the extracted feature maps. The number of extracted feature maps corresponds to the number of filters defined for the Convolution layer. The feature maps are then added together to form one single image which serves as the input of the 2D Max Pooling Layer. This procedure is demonstrated in Figure 2.5.

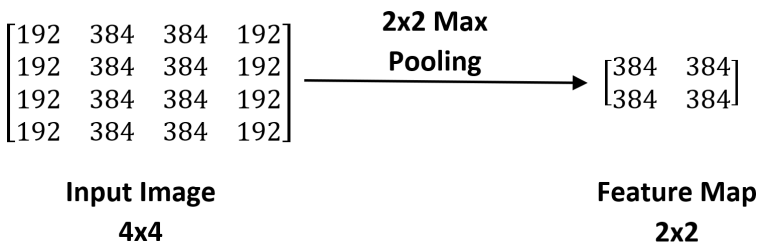

Figure 2.5: A 2D Max Pooling operation reducing the dimensionality of the input image by selecting the strongest activation in 2x2 regions of the image.

The Max Pooling Layer has a hyperparameter which defines the size of the 2D pooling operation in a similar fashion to the filter size of the Convolution Layer. The pooling operation then slides across regions of the input image. These regions have the same size as the pooling operation. The pooling operation then picks out the maximum pixel value within the image region and replaces this whole region with the maximum pixel value. Since there are no weights involved, this layer requires

no learning unlike the Convolution and Fully Connected Layers which involve filter weights and weighted connections, respectively. A size of 2x2 was used for the Max Pooling layers. In deep learning, there is another type of pooling known as Average Pooling. Instead of taking the maximum pixel value of the region, Average Pooling outputs the average pixel value. Max Pooling is able to detect the prominent features [49] of an image while average pooling has a smoothing effect on images [50].

2.2.4 The Softmax Layer

The Softmax layer produces the output of a CNN classifier. The output is a decision on which label the input data belongs to based on the features extracted by the CNN. The decision produced by the Softmax layer depends on the activations and feature maps extracted throughout the CNN. The number of nodes in the Softmax layer matches the number of labels in the classification task. The activation function used in the Softmax layer is known as the Softmax function which maps the input into a probability. The probability values are ordered in an array. Discrete scores ranging from 0 to 5 are assigned to the labels in the sequence of demyelination severity. The array produces as many discrete scores as demyelination labels. The array index containing the highest probability value is the output of the CNN, i.e. the classification decision, since the index represents the discrete score.

Suppose we have an architecture with an output layer similar to that in Figure 2.3 which would be referred to as a Softmax Layer in this case. If this is used in classification, then the problem being solved is a binary image classification task with two classification labels. The most famous example is the Cat vs. Dog classification where a CNN is trained to classify whether an image shows a Cat or a Dog. The first node represents the classification of a Cat while the second node represents that of a Dog. Each node computes the value of z using the typical linear operation and passes it to the Softmax Activation function. The activation of the Softmax Function in layer L for node i in a classification task consisting of K labels is defined as follows:

$$a_i^L = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}.$$

This function normalizes the outputs of the nodes to map the data onto a probability distribution. The activations then represent the likelihood that the image belongs to one of the labels. In classification, the labels are usually assigned an integer value starting from zero. In the case of Cat vs. Dog, the Cat is assigned a discrete score of 0 and Dog is assigned the discrete score 1. The output of the Softmax Layer for the Cat vs. Dog is a vector of size 2 and the vector components are in the order of the assigned discrete scores, i.e. the first component of the vector is the probability that the image is a Cat and the second component is the probability that it is a Dog image. The position of the vector component with the maximum probability, i.e. the discrete score representing the most likely label, is the final output.

2.2.5 The Dropout Layer

The Dropout layer is responsible for dropping out nodes in the hidden layers of a Fully Connected Network during training [51]. Dropping out means that the algorithm switches off any activation of a node. The selection of the nodes that drop out is random and the drop out is temporary since it lasts for only a single Epoch (this term is defined in 2.3.1). In each new Epoch, nodes are again randomly selected to be dropped out during the learning process. The benefit this provides is it helps to avoid overfitting the model which takes place when the classifier learns to memorize the data used in the learning process but fails to generalize when presented with new test data. Dropout also allows the ANN optimization to focus on the important connections between different nodes.

2.3 Model Optimization and Learning

2.3.1 Gradient Descent

We have seen that the classification decision of the CNN is dependent on the input data and the weights. The weights are randomly initialized and then input data are classified by the CNN. Since the weights are randomized, it is expected that the CNN will perform poorly on classifying. These include both the filter weights of the Convolution layers and the weighted connections of the Fully Connected layers. It is necessary to implement an optimization algorithm for the CNN such that the weights are updated in a way that improves on the classification results.

The optimization is performed by minimizing the loss (also referred to as the error or uncertainty) of the classification results. The loss is modelled as the cross-entropy of the classification:

$$C = -\sum_{i=1}^{K} t_i log_e(a_i^L).$$

Given a specific label a_i^L, the value of t is a discrete value of 0 or 1. 0 means that the image does not belong to that label and the opposite is true for 1. In our Cat vs. Dog problem, an image of a Cat would have $t_0 = 1$ and $t_1 = 0$. If a label i is not the correct label, then $t_i = 0$ and we would be left with the term where $t_i = 1$. Thus, the value of the loss can be simplified to the categorical cross-entropy for the activation of a node representing the correct label $a_{correct}^L$:

$$C = -log_e(a_{correct}^L).$$

The aim of the optimization is to minimize the loss function since we want to reduce the uncertainty on the classification decisions of the CNN. The loss is an important metric to improve since a lower uncertainty means that the model was able to extract the unique and hidden features within the image. This allows us to have trust in the decisions of the CNN. The optimization is performed using Gradient Descent (GD). The loss depends on both the weights and biases, and the optimization process involves adjusting their values.

The deep learning algorithm is required to go over the training dataset several times, updating the weights and biases after each run. A single iteration of observing the dataset and updating the parameters is referred to as an Epoch. The number of Epochs required for the algorithm to have the required performance will depend on the deep learning task and the architecture of the neural network. Suppose we have a vector of all the weights W and a vector of all the biases B with a loss function $C(W,B)$. In between each Epoch, the gradient $\nabla C(W, B)$ is computed. The method on how the gradient is found is known as Backpropagation which will be discussed in the following subsection.

Consider that with the random initialization of the weights and biases, the CNN is at a loss state that is far from the minimum. GD allows the CNN to take a step along the gradient in order to get closer to the optimized state. The weights and biases are updated as follows. α is a hyperparameter known as the learning rate. The learning

rate is used to scale the size of the step taken along the gradient. A higher learning rate decreases the overall runtime of the CNN learning process since the step sizes will be larger and the CNN is able to reach the minimum using a fewer number of Epochs. However, this could lead to the CNN overshooting and failing to reach the minimum. A smaller learning rate is more suitable to use when approaching the minimum to ensure that the CNN settles as close to the minimum as possible. The step is taken in the opposite direction of the gradient. This is repeated for the number of Epochs specified in the algorithm.

Since GD would go over the entire dataset during each Epoch, this would let the algorithm run for an unrealistically long time until the loss converges. An alternate scheme for GD is the Minibatch Stochastic Gradient Descent (MSGD) which uses a randomly selected batch of data points from the dataset and updates the weights and biases based on the average gradient of the minibatch. In this case, the parameters are updated successively for each data point in the batch where one Epoch is completed. If we define a batch size of 32, this means that in one Epoch, the weights are updated after observing 32 data points from the entire dataset.

2.3.2 Backpropagation

The CNN algorithm requires an efficient way of computing $\nabla C(W, B)$. Backpropagation uses the chain rule to compute $\nabla C(W, B)$ for each weight and bias in our CNN model. Suppose we have a neural network with L layers. Backpropagation starts out by calculating $\nabla C(W, B)$ for the last layer using the chain rule as follows:

$$\frac{\partial C}{\partial w} = \frac{\partial z}{\partial w}\frac{\partial a}{\partial z}\frac{\partial C}{\partial a}.$$

Similarly for the bias:

$$\frac{\partial C}{\partial b} = \frac{\partial z}{\partial b}\frac{\partial a}{\partial z}\frac{\partial C}{\partial a}.$$

We are able to find the partial derivatives from the defining equations for a and z as follows:

$$z = wa^L_{correct} + b, \quad \frac{\partial z}{\partial w} = a^L_{correct},$$

$$a = \sigma(z), \quad \frac{\partial a}{\partial z} = \sigma'(z),$$

$$C = -log_e(a^L_{correct}), \quad \frac{\partial C}{\partial a} = \frac{1}{a^L_{correct}},$$

therefore

$$\frac{\partial C}{\partial w} = \sigma'(z).$$

For the bias, we only need to find one more partial derivative which is:

$$\frac{\partial z}{\partial b} = 1,$$

therefore

$$\frac{\partial C}{\partial b} = \frac{\sigma'(z)}{a^L_{correct}}.$$

The $\nabla C(W, B)$ for each weight and bias is averaged over the dataset from which the CNN is learning from during an epoch. The weights and biases are adjusted using the following learning rules:

$$w_{new} = w_{old} - \alpha \frac{\partial C}{\partial w},$$

$$b_{new} = b_{old} - \alpha \frac{\partial C}{\partial b},$$

where α is the learning rate.

2.4 Dataset Preparation and Training

2.4.1 Dataset Split

For the CNN model to learn, a portion of the dataset has to be used for the learning process of the model and the rest of the images would serve as unseen data. The dataset used by the model to update the weights and biases is known as the Training Dataset and the unseen data constitute the Test Dataset. In machine learning, the typical ratio split is 0.8:0.2 (Training:Test). This split allows the model to be exposed to several amounts of data during training such that the model is able to learn the features unique to each label, thus allowing it to classify the data accurately and with certainty. The test dataset aims to make sure that the model did not overfit on the training data. Overfitting is present when the model performs well on the training data but fails to deliver the same performance on the test dataset. In deep learning, this means that the model has learned to memorize the training dataset but fails to generalize on the training data. To monitor for overfitting during the training process, we introduce the Validation Dataset.

At the end of each Epoch of the learning process, the CNN model goes through the Validation Dataset to compute the accuracy and loss for this dataset (the accuracy metric is define in section 2.4.2.2 below). After the CNN completes the learning process, the evolution of both the Training and Validation metrics are compared. Overfitting takes place when the validation loss starts to increase. The typical split with the validation is 0.6:0.2:0.2 (Training:Validation:Test). If overfitting is detected, it is possible to adjust the CNN architecture and restart the learning process. The adjustment could be repeated until overfitting is no longer observed. Figure 2.6 shows an example of overfitting where the validation loss starts to increase after approximately ten epochs. This plot was an overfitting result on the CARS microscope dataset.

In our work here, the training dataset had 8,435 images while both the validation and test dataset had 2,812 images, adding up to a total of 14,059 images. Data augmentation is the process of synthesizing new training data and thus increases the number of training samples that the CNN is exposed to. Examples of data augmentation techniques are vertical and/or horizontal flipping, image rotation, and image rescaling. As long as the synthesized training data can exist in nature, i.e. it is possible to observe the augmented image in real life, then the augmentation technique can be used. This has been shown to improve the performance of the CNN and helps to avoid overfitting [20].

The training process starts out with the learning rate choice $\alpha = 0.01$. If no improvement is observed after ten epochs then α is reduced by a factor of 10. The lack of improvement is a possible indication that the model is starting to require smaller step sizes to descend through the loss function landscape so that the loss can be further minimized. This technique is known as a learning rate scheduler [52].

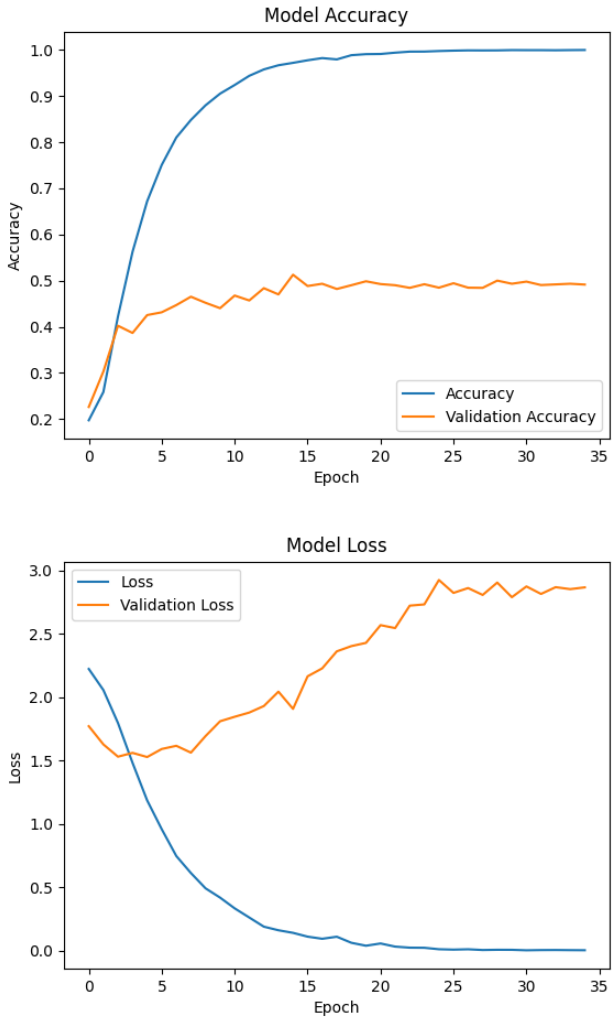

Figure 2.6: Example of an overfitted CNN model. The validation accuracy is the fraction of correctly made predictions across the validation dataset and it starts to plateau while the training accuracy keeps increasing. The validation loss is the uncertainty on the validation dataset. It initially decreases, then starts to increase while the training loss is monotonically decreasing.

2.4.2 Performance Metrics

It is essential to evaluate the classification performance of the CNN to ensure that the decisions made by the model are to be trusted. In this section, we discuss various performance metrics [53] that provide useful insights into the decisions made by the classification model.

2.4.2.1 True/False Positives and True/False Negatives

Suppose we have a binary classification problem, for example, the Cat vs. Dog classification. We would like to evaluate the performance of a trained CNN using a test dataset. We assign the Cat label as a positive class and the Dog as a negative class i.e. the classifier will output whether an image is a Cat or not a Cat. When a classifier attempts to make predictions on data, the decision can either be positive or negative.

A positive decision is made when the classifier outputs that the image is a Cat. If the CNN believes that the image is not a Cat (i.e. it is a Dog image), then the decision is negative. However, it is possible that a positive decision is made when it should have been negative or vice versa. In such a case, the decision is deemed to either be a False Positive (FP) or False Negative (FN). If the prediction is correct, then the decision will either be a True Positive (TP) or a True Negative (TN). The predicted label is correct if it matches the actual label (known as the Ground Truth) of the data point. Figure 2.7 shows an example of a Confusion Matrix which is the type of visualization used to observe classification results.

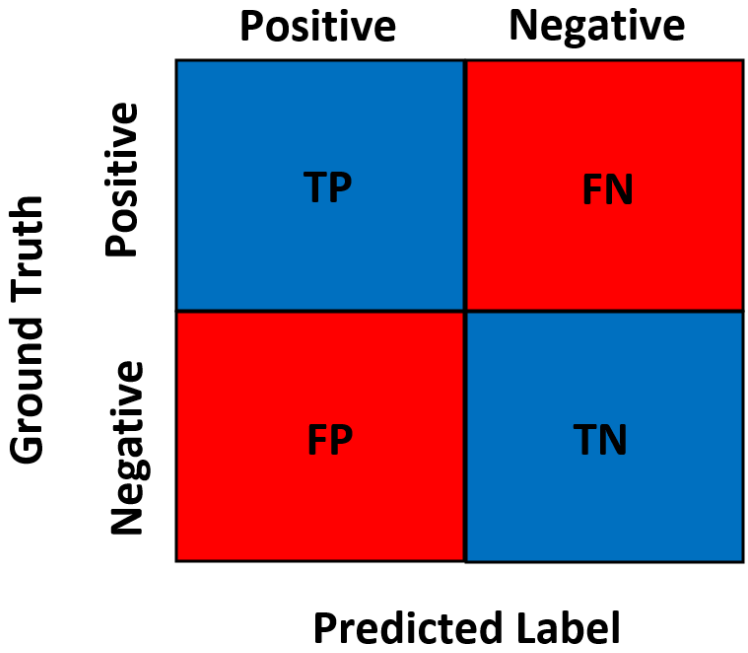

Figure 2.7: Confusion Matrix Visualization of the Positive and Negative Predictions. The blue squares are the Positive decisions while the red squares are the Negative decisions.

Now suppose we extend our classification task to include Hamsters. Instead of a Binary Classification, we now have a Multi-label Classification problem. This extends the meaning of TNs, FPs, and FNs. The definition of TP remains the same since there is only one possibility of receiving a TP decision.

Since this is no longer a binary classification task, there is no clear distinction as to which type of decision the classifier is making. The fundamental definitions of True/False Positives and True/False Negatives need to be used. Suppose we would like to look at the metrics for the Cat label. A TP will only take place when the predicted label is equal to the ground truth. In Figure 2.8, 13 TP decisions were made for the Cat label. TN decisions occur when both the predicted label and ground truth are not equal to the Cat label. Thus, the number of TN decisions from figure 2.8 would be:

$$TN = 11 + 6 + 7 + 15 = 39.$$

FP decisions have the predicted label equal to Cat while the ground truth is equal to either the Dog label or the Hamster label. Therefore, in our example the number of FP decisions is:

$$FP = 8 + 2 = 10.$$

Finally, FN decisions would have the predicted label equal to either the Dog label or the Hamster label while the ground truth is equal to the Cat label. In this case, the number of FN decisions is:

$$FN = 3 + 5 = 8.$$

The same procedure can be followed for the Dog and Hamster Classes. In the upcoming subsections, we will see how the True/False Positives and True/False Negatives are used to compute important metrics that assess the performance of the CNN classifier. For our Cat vs. Dog vs. Hamster problem, the Positive and Negative decisions for each label would be:

$$TP_{Cat} = 13 \qquad TP_{Dog} = 11 \qquad TP_{Hamster} = 15$$

$$TN_{Cat} = 39 \qquad TN_{Dog} = 35 \qquad TN_{Hamster} = 35$$

$$FP_{Cat} = 10 \qquad FP_{Dog} = 10 \qquad FP_{Hamster} = 11$$

$$FN_{Cat} = 8 \qquad FN_{Dog} = 14 \qquad FN_{Hamster} = 9.$$

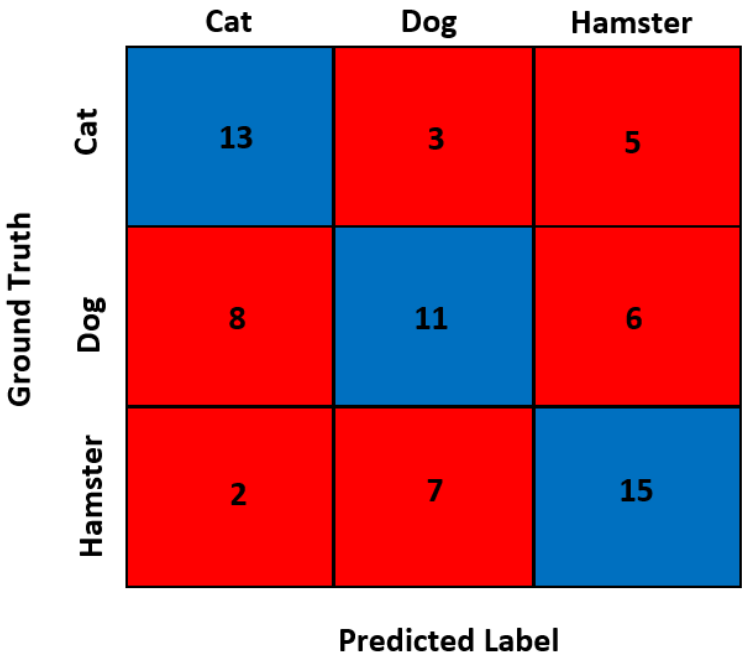

Figure 2.8: Confusion Matrix Visualization of a Multi-label Classification problem. Each blue square shows the number of correct/positive predictions while the red squares show the false predictions.

2.4.2.2 Accuracy

The accuracy metric is a measure of the correct predictions made within a dataset. It is a general metric that describes how well the model performs on each label. Accuracy is the most used classification metric for classification problems [54–57]. The accuracy has the following equation:

$$A_{label} = \frac{TP_{label} + TN_{label}}{TP_{label} + TN_{label} + FP_{label} + FN_{label}}.$$

For our Cat vs. Dog vs. Hamster problem, the Positive and Negative decisions for each label would be:

$$A_{Cat} = 0.743,$$

$$A_{Dog} = 0.657,$$

$$A_{Hamster} = 0.714.$$

Based on our example, the model seems to perform best on the Cat label and has the lowest accuracy on the Dog label. The overall accuracy of the model is taken as the average of the label accuracies:

$$A = \frac{0.743 + 0.657 + 0.714}{3} = 0.705.$$

The accuracy metric only measures the correctly made predictions and thus provides less distinctive results compared to other metrics [58, 59]. We move on to discuss stronger performance metrics that can give better indications on the classification performance.

2.4.2.3 Precision

The precision metric measures the fraction of positive decisions that were correct, i.e. the ratio of the TPs to the total number of positive decisions. Thus, the equation for the precision metric is:

$$P_{label} = \frac{TP_{label}}{TP_{label} + FP_{label}}.$$

The precision of a classifier model determines on how well the model can classify a positive decision. A classifier will be more precise if it generates more TPs or less FPs. For our example, the precision of each label is:

$$P_{Cat} = 0.565,$$

$$P_{Dog} = 0.523,$$

$$P_{Hamster} = 0.577.$$

We can already see that the precision provides a different view of the model's performance than what the accuracy shows. Accuracy measures the fraction of correct predictions which include both TPs and TNs. Precision focuses on the correct positive predictions and misclassifying negative samples as positive. The overall precision of our example is:

$$P = \frac{0.565 + 0.523 + 0.577}{3} = 0.555.$$

2.4.2.4 Recall

The recall metric measures the ratio of the correct positive predictions to the total number of data points assigned with the positive label, i.e. the ground truth label. Here we are measuring the number of positive samples that the classifier can detect as opposed to precision which measures the number of positive samples that were actually positive. FNs are decisions deemed negative by the classifier but in fact are positive samples, thus they are taken into consideration when measuring recall. The equation for the recall metric is:

$$R_{label} = \frac{TP_{label}}{TP_{label} + FN_{label}}.$$

The recall for each label in our example is:

$$R_{Cat} = 0.619,$$

$$R_{Dog} = 0.440,$$

$$R_{Hamster} = 0.625.$$

The overall recall of the model is:

$$R = \frac{0.619 + 0.440 + 0.625}{3} = 0.561.$$

2.4.2.5 Precision vs. Recall and the F1-Score

The usage of either precision or recall as our performance metric of choice depends on the type of classification problem. Precision is used when the classification task is sensitive to misclassifications [60]. For example, classification of a medical image showing either a cancerous or non-cancerous cell would require the maximization of Precision. Recall is used when the goal is just to detect all the positive samples with little importance given to misclassifications [60]. An example would be using

a camera that detects small vehicles passing by a road. The classification model of the camera may misclassify some mid-sized cars as small but we only care about correctly detecting all of the cars with the small ground truth label, thus recall would be maximized for this task.

If both metrics are of interest, it is possible to compute the F1-Score of the labels. The F1-Score is the harmonic mean of the Precision and Recall. A high F1-Score indicates low FPs and FNs. It is calculated as follows:

$$F = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}.$$

The F1-Scores of our example are:

$$F_{Cat} = 0.590,$$

$$F_{Dog} = 0.478,$$

$$F_{Hamster} = 0.600.$$

The overall F1-Score is:

$$F = \frac{0.590 + 0.478 + 0.600}{3} = 0.556.$$

2.4.3 CNN Results and Discussion

The architecture of the CNN is adjusted to achieve a better performing classifier. In this thesis, the classifier's performance was deemed reasonable for our task if most of the predictions were correct with the lowest achievable uncertainty. A line search method was used to optimize the hyperparameters of the architecture. The hyperparameters include the learning rate, width of the CNN (number of filters), depth of the CNN, and the number of nodes in the FCNs. Line search was used to adjust the values of the parameters individually until they no longer showed improvement on the CNN performance. Once a CNN architecture has been designed such that overfitting is not observed and a reasonable accuracy and loss are achieved, the CNN is able to make reliable classification predictions. These predictions are performed on data that the CNN did not come across during the learning process, i.e. unseen data such as the test dataset. To confirm the model is able to generalize well enough, we evaluate its performance on the test dataset using a Confusion matrix. For the Convolution portion of the CNN is shown in Figure 2.9. The number of feature detecting filters found in each Convolution layer were the same and three different CNNs were tested.

The first CNN had 8 filters in the Convolution layers, the second had 16 filters, and the third had 32 filters. A Flattening layer followed the Convolution portion of the CNN and an FCN with a Softmax layer was connected as the last portion of the network. The FCN consisted of four dense layers with each layer having 128 nodes. All the Convolution and Dense layers used the hyperbolic tangent activation function.

To ensure the model is indeed able to generalize, the accuracy achieved on the training dataset should closely match that of the validation and test dataset. The losses on both the training and validation datasets should also match. When visualizing the training and validation results, the gaps between the training and validation accuracy/loss defines the degree of overfitting. The larger the gap is, the more the CNN model has learned to memorize, i.e. the more it overfits.

It is important to note that the model could achieve a reasonable accuracy and loss during training, and then it can start to overfit in the latter part of training. Instead of readjusting the architecture, it is possible that the model retains the weights and biases that provided the highest accuracy or lowest loss after the training process is completed. So instead of using the weights achieved at the last Epoch, optimal weights from the best Epoch are saved for the model. This can be done in the programming language used to create the CNN algorithm. When deciding to code it in such a way, the user would have to decide whether to save the weights that gave the highest accuracy or the lowest loss. A high accuracy would mean that the classifier predicts the correct label most of the time and a low loss indicates the model is certain of its predictions. In most cases, once the model reaches its learning cap i.e. when the accuracy and loss start to plateau, that is when the highest accuracy or lowest loss will be saved.

The CNN architecture used is adapted from the VGG16 model which was used in classifying image data belonging to ten different labels [61]. For our image dataset, several architectures were used. However, VGG16 achieved the balance of reasonable performance and computational cost based on the available specifications of the computer used to run the code. In our work, the computer used to run the algorithm had an intel core i7 processor, 32 gigabytes of RAM, and a GTX GeForce 1070 graphics card. To achieve desirable performance metrics, the network needs to have a large enough depth such that the model is able to eventually extract abstract features from our microscope images that would significantly improve the CNN's performance.

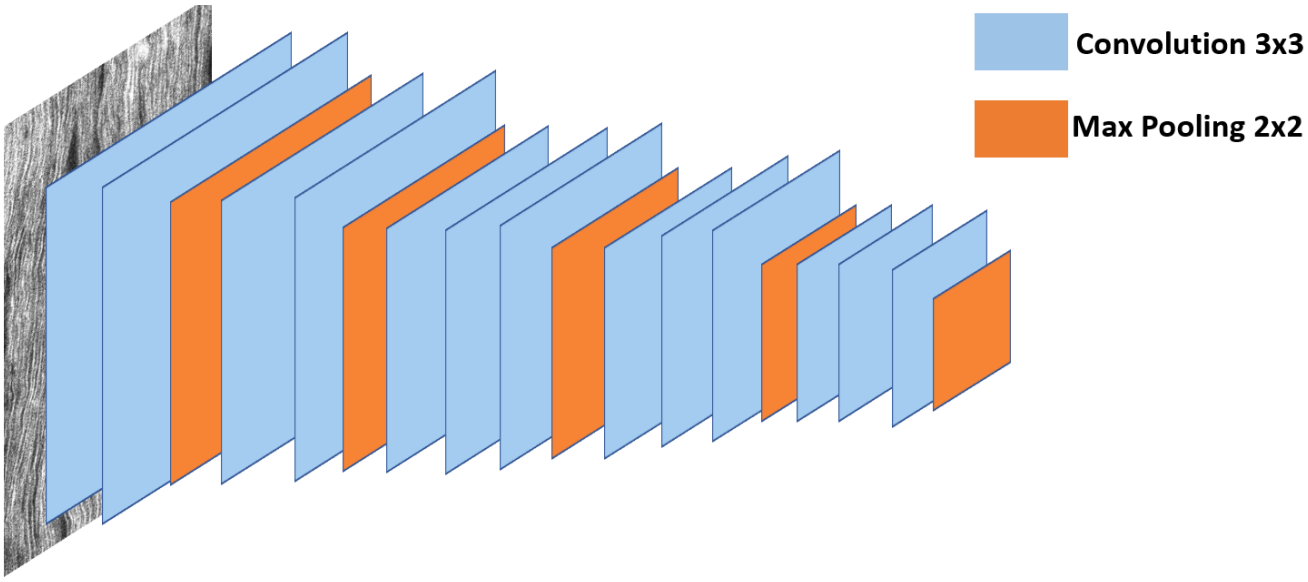

Figure 2.9: The Convolution Network of our CNN. Each Convolution layer has a filter size of 3x3 and each 2D Max Pooling layer has a window size of 2x2.

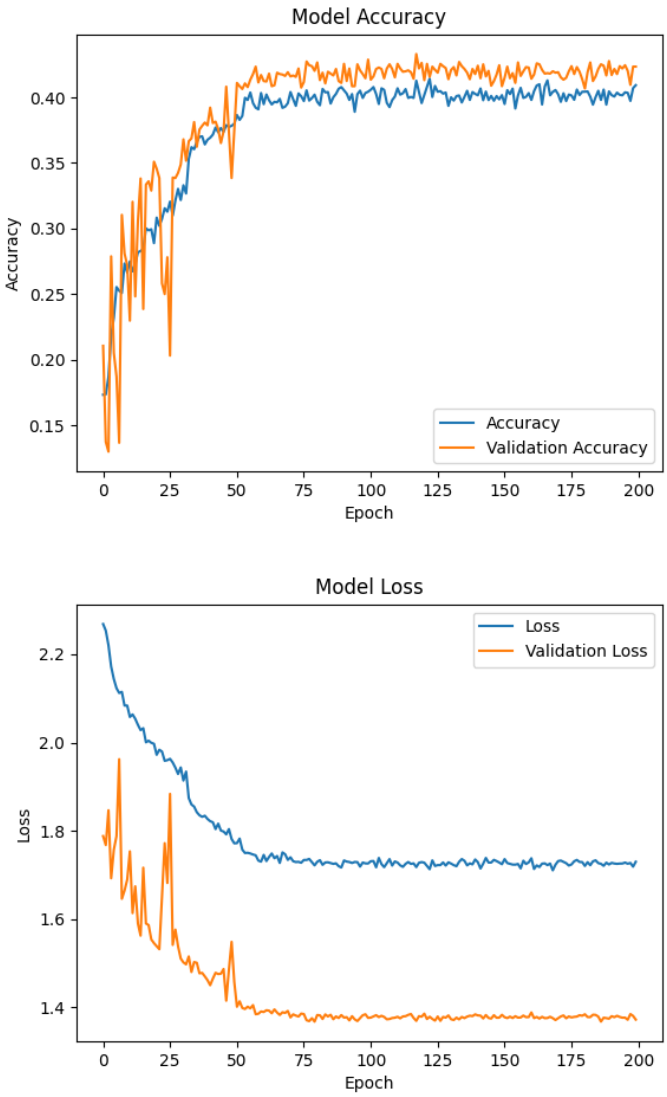

Figure 2.10: Model Accuracy and Loss Function defined by Categorical Cross-Entropy for CNN model with 8 filters/nodes in each convolutional layer. Each filter is a 3x3 matrix of weights that detect image features.

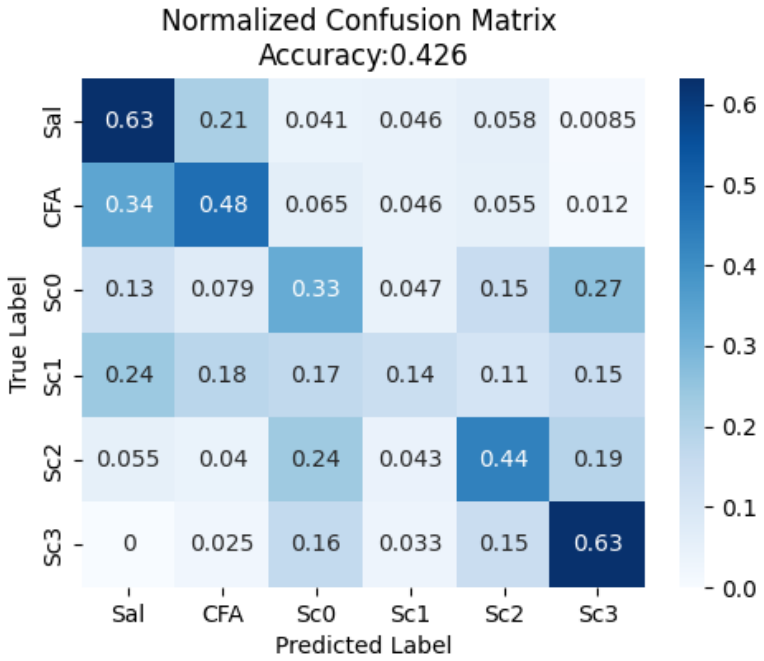

Figure 2.11: Normalized Confusion Matrix for the CNN model with 8 filters. The training, validation, and test dataset sizes are in the 0.6:0.2:0.2 ratio of the spinal cord images. The Confusion Matrix values are normalized by dividing the TPs, TNs, FPs, and FNs by the total number of decisions.

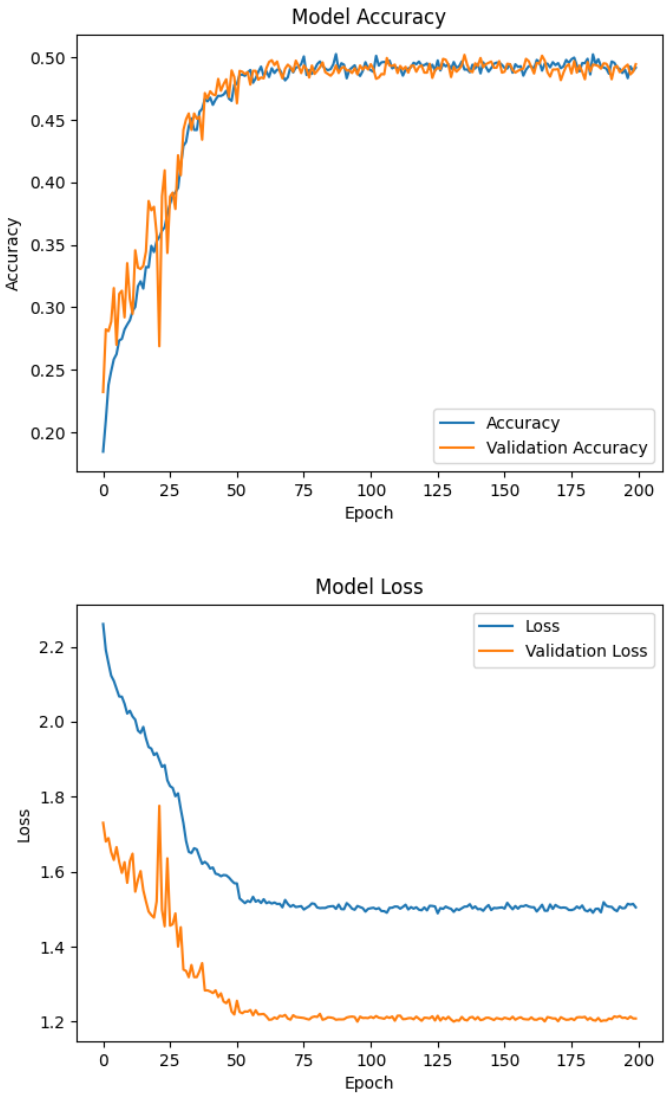

Figure 2.12: Model Accuracy and Loss for CNN model with 16 filters.

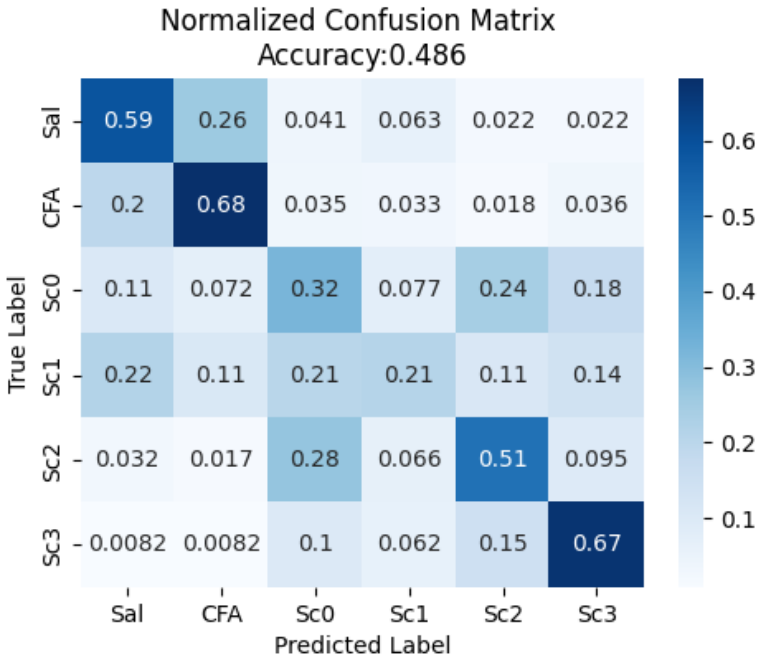

Figure 2.13: Normalized Confusion Matrix for CNN model with 16 filters.

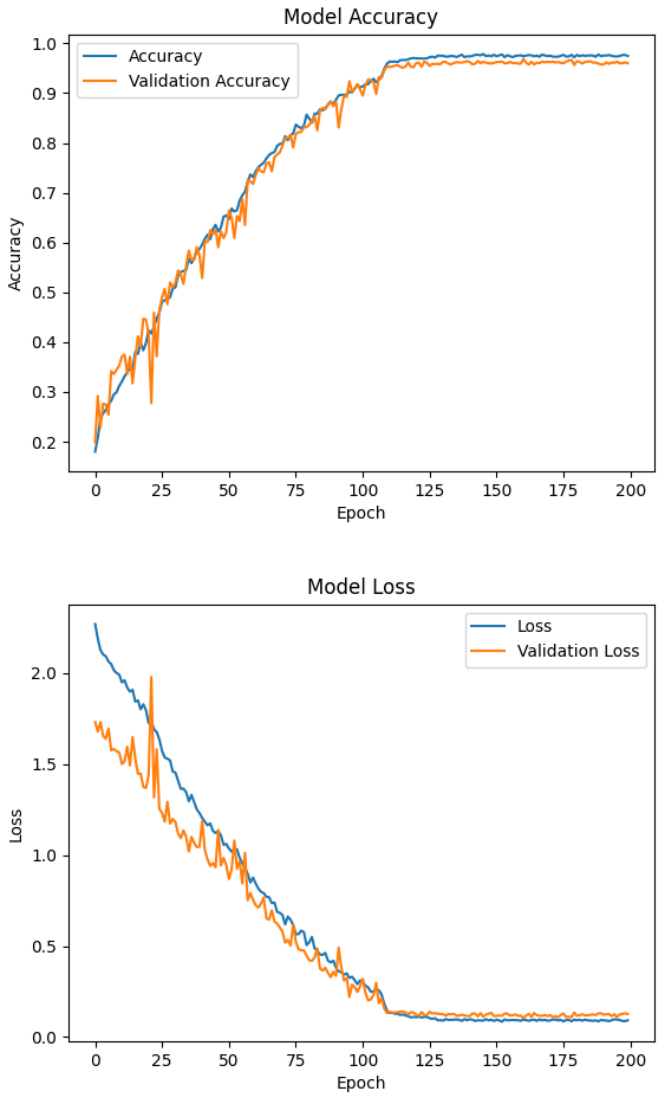

Figure 2.14: Model Accuracy and Loss for CNN model with 32 filters.

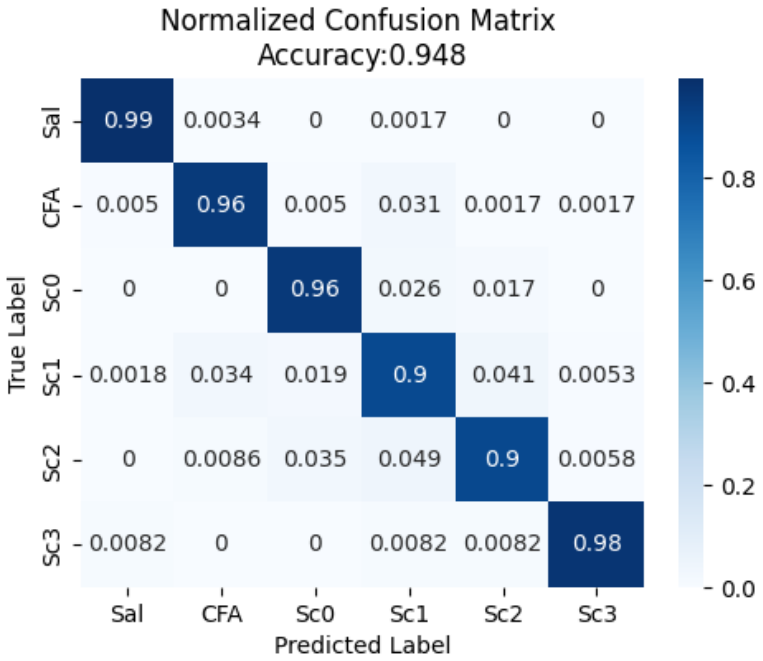

Figure 2.15: Normalized Confusion Matrix for CNN model with 32 filters.

	Precision	Recall	F1-Score
Sal	0.99	0.99	0.99
CFA	0.96	0.97	0.96
Sc0	0.95	0.95	0.95
Sc1	0.92	0.90	0.91
Sc2	0.91	0.91	0.91
Sc3	0.98	0.97	0.98

Table 2.1: Performance metrics of CNN model with 32 filters.

Three different CNN models were trained. Figure 2.9 summarizes the convolution portion of the archeticture. The number of filters used for all convolutions was either 8, 16, or 32. The FCN portion consisted of four dense layers each having 128 nodes. The activation function used for the applicable layers was the hyperbolic tangent function. All of the weights including the filter nodes of Convolution layers were trained. Training time ranged from 3 hours to 7 hours. The learning rate α was initially set to 0.01. If after 10 epochs no significant improvement is observed in the loss of the model, the learning rate was reduced by a factor of 10. The architectures were identical however the width of the Convolutional portion, i.e. the number of convolutional filters was varied. This was done to investigate the minimum number of filters needed to reach an accuracy of approximately 90%. Figures 2.10, 2.13, and 2.14 show the results on the training and validation datasets for 8, 16, and 32 filters respectively. Figures 2.11, 2.13, and 2.15 show the Confusion Matrix results on the test dataset for 8, 16, and 32 filters respectively.

The performance shown in Figure 2.12 with 8 filters seemed to be very poor as the model was not able to surpass 50% accuracy. The large loss indicates that the model is uncertain with its decisions. The predictions made by a model with such metrics would not be trusted. The model did not have enough feature representations to properly classify the images. This implies that more filters need to be added to the Convolutional layers. The Confusion Matrix in Figure 2.13 shows that the classifier predicted few TPs which indicates that the CNN has not made sufficient correct predictions. Since the diagonal elements represent the TPs, We want to see warmer colors for the diagonal elements and cooler colors for the remaining elements.

By having 16 filters, we can see a slight improvement in the model's performance and metrics as shown in Figure 2.14. This comes at the expense of a longer training time. The model reaches an accuracy of approximately 50% and the lower loss indicates that the model is a bit more certain in its decisions. Figure 2.15 shows insignificant improvement as the diagonal elements in the heatmap are not warm enough.

The classification performance is nearly maximized with 32 filters as shown in Figures 2.16 and 2.17. The high accuracy implies that the model is mostly making correct predictions. The loss indicates low uncertainty in these predictions, giving an insight into the feature extraction capabilities of the CNN. The CNN model nearly

achieves perfect classification and shows desirable results for the performance metrics which are summarized in Table 2.1. The Accuracy results shown in the confusion matrix as well as the Precision and Recall are all 0.9 or above. This is highly indicative of a reliable CNN. The Precision values indicate a small number of false positives so the CNN does not misclassify a label too often. This means the model is able to mostly detect the unique features in each image and associate them to the correct label. This confidence in the CNN decisions is further solidified by the high recall values.

It should be noted that the improvements provided by the additional filters could result from specific layers and not necessarily all of them. Ideally, each layer should not have an excessive number of filters, since at some point there is no longer a benefit towards the CNN's performance. Also, the results do not necessarily show that the classification performance will not improve past 32 filters. Better performance could be observed if more than 32 filters are used. However, the computational cost was not possible to be accommodated due to constraints on the available hardware. With 8 filters, each epoch takes approximately 50 seconds. This amount of time was raised to approximately 80 and 120 seconds for 16 and 32 filters respectively. For 200 epochs, the time it would take to train the model with 8 filters would be approximately 3 hours, approximately 4 hours for 16 filters, and approximately 7 hours for 32 filters. Since the test dataset has a lower number of images and only requires one epoch to produce the Confusion Matrix results, it takes significantly less time to run the test dataset using our model. For the model with 32 filters, it takes around 20 seconds to go through one epoch of the test dataset. Thus, it is the training process that would require a significant amount of time to be completed.

It is interesting to point out that the CNN model is not capable of achieving a perfect accuracy due to the labelling process of the data. The mice were assigned the label based on a visual assessment of their behavior and not on the actual spinal cord images. Due to the heterogeneity of the spinal cord and demyelination, a single mouse can have spinal cord regions with different degrees of disorder and myelin damage. If the labels were assigned based on visual inspection of the actual CARS images, then a perfect accuracy might be achievable. It might be worthwhile to thoroughly analyze the images that contribute to the misclassification rate.

There is ongoing research on the uncertainty of CNN classification and investigating the features that the CNN model has actually learnt regardless of its performance [59, 62–65]. It is important to know that the classifier is able to extract distinguishing features. Biophysical models can become complex as we try to include several parameters where only a few are essential. The challenge here is figuring out which parameters have the most impact in the model outcomes. In our case, we are interested in the severity of demyelination which we study using images. Out trained CNN must have picked up the important structural features that are present in different degrees of demyelination. By figuring out the features that the CNN is detecting, we might be able to constrain demyelination models such as [20] to better understand the disease severity based on action potential behavior. Alongside understanding the physical parameters affected by the damage inflicted through demyelination, synthesizing our own images could prove to be effective in interpreting CNN decisions. In short, the myelin images that we have access to can be satisfactorily classified. Occasionally there would be difficulty in distinguishing Sc0 and Sc1 images and Sc2 and Sc3 images, but the model can clearly distinguish healthy from unhealthy images.

Chapter 3

Image Synthesis using 2D Biased Random Walks

3.1 Interpreting the Decisions of our CNN

3.1.1 Class Activation Maps

The performance metrics justifies the credibility of the decisions made by the CNN which was an important step before attempting to uncover the physics behind the images. We had to search for a technique that would allow us to visualize and understand the convoluted features learnt by the CNN. Feature extraction is a technique used to visualize the feature maps constituting the different representations of the original image.

CNNs are capable of classifying partial objects remarkably well [62–65]. Occlusion experiments involve "zero-masking" parts of the original image (i.e. setting the pixel values in the masked region to zero) before feeding it into the CNN model and compare how much the loss increases in the decision [59]. This is used to measure the contribution of the masked region on the distinction that the CNN performs. Class Activation Maps (CAMs) expand on such localized contributions by indicating which regions in the image contributed the most to the CNN's decision by superimposing a heatmap on the original image [65].

One would then hope to use this technique on the spinal cord images to measure demyelination severity across different parts of the image. CAMs are specific for CNNs that have a Global Average Pooling (GAP) layer in place of a Flattening layer at the end of the Convolutional network. The GAP layer averages the pixel values of a feature map to represent the feature as a single value.

An extension to CAMs, known as Gradient-CAMs [33], generalizes the method to work with any CNN architecture, not only to those having a GAP layer. In fact, if Gradient-CAM is applied in the architecture used for regular CAM (i.e. with the GAP), the mathematics would simplify to that of regular CAM. Gradient-CAM is usable in any CNN architecture since it uses the gradients found using backpropagation to produce the heatmap on the original image.

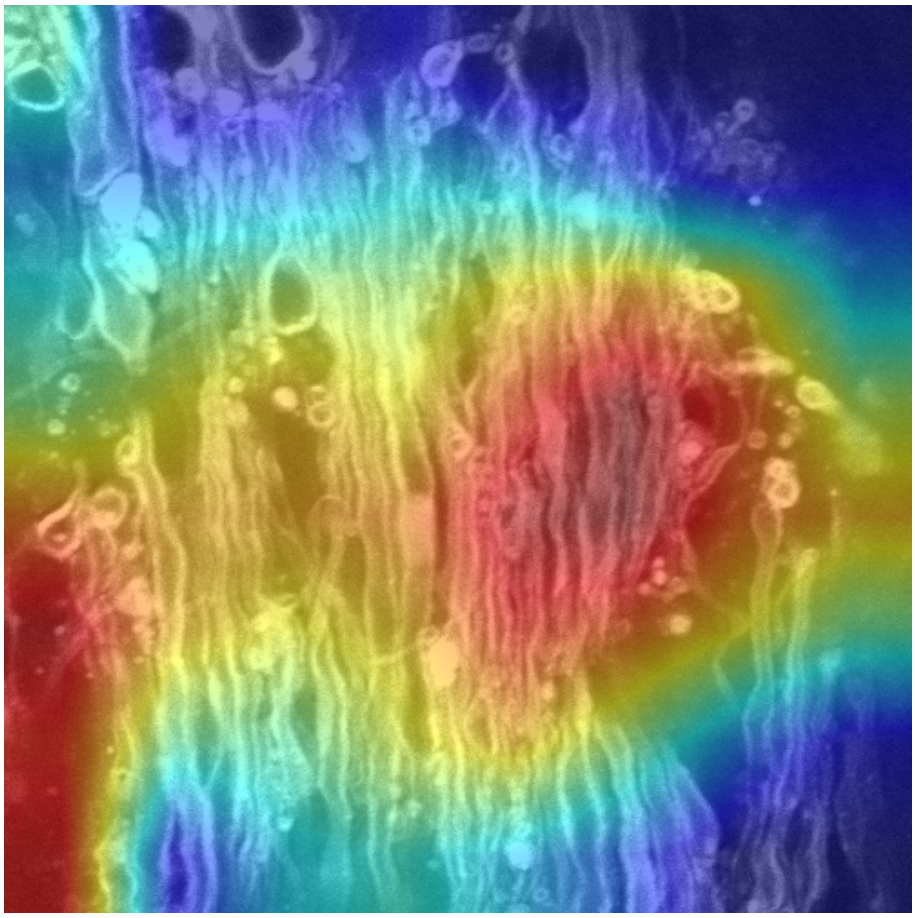

Figure 3.1: Gradient-CAM applied on a Sc3 image. Warmer colors show the regions that contributed the most to the CNN decision, while cooler regions show the least contributions.

Although the Gradient-CAM seems to distinctly highlight regions in the microscope image, it is visually difficult to tell what the CAM is detecting. We were hoping that the CAM would highlight both lesions and randomly oriented nerve fibres. However, we were not able to figure out a way to understand the physics encoded in the image features.

Therefore, we decided to proceed by synthesizing our own images using a 2D Biased Random Walk. Previous work have used 2D Random Walks to study 2D spatial patterns for animal movements [33, 34] and nerve fibre tracts [35, 66, 67]. We thus aim to simulate 2D images of longitudinal nerve fibres and encode the physics using 2D random walk parameters. Then we will apply the CNN which was trained on the CARS images to extract the relevant features from synthesized images. From this we would be able to connect the encoded physics with the images.

3.1.2 Introduction to Random Walks

Random Walks are a type of stochastic process. Consider a person standing on an integer number line. At each discrete time step, the person is allowed to either take a step to the right or to the left i.e. $\Delta x = 1$ or $\Delta x = -1$. The direction of the step depends on the probability p that the person takes a step to the right. Suppose that $p = 0.5$; this means that the person has an equal likelihood of stepping in either directions since the probability of taking a step to the left is $q = 1 - p = 0.5$. Such a process is considered to be Markovian i.e. the steps are independent of each other.

The example given above is a one-dimensional (1D) random walk. It is possible to extend the random walk to a two-dimensional (2D) lattice. The person's steps would then have components in both the x (horizontal direction) and y (vertical direction) axes. Suppose that the random walker has an equal likelihood of taking a step upwards, downwards, right, or left. This means that $p = 0.25$ for any of the four possible directions. The random walk process evolves with time, thus creating a spatial pattern of the journey embarked on by the random walker. We will be simulating the evolution of a population of random walkers, each performing a specified number of steps. The initial positions of the random walkers would be distributed uniformly across the bottom of the lattice. By sampling many random walkers, one can find the mean displacements at each step in the x and y directions $\mu_{x,i}$ and $\mu_{y,i}$, where i is the step number. The means describe the average position across all walkers.

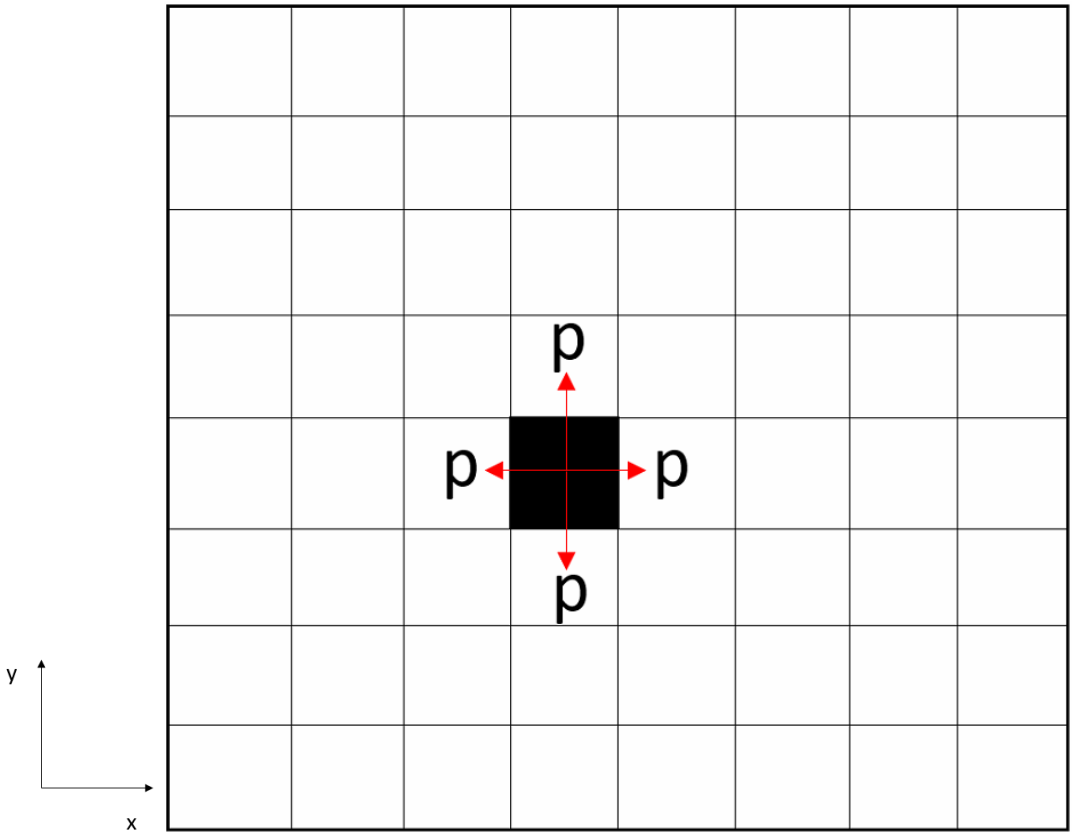

Figure 3.2: A random walker at its starting point (black box) with an equal likelihood p of stepping in one of four allowed directions.

The simulation gives insights into the statistical properties of the random walker. Before extracting useful statistics, it is important that enough sampling be performed. Lack of sampling might lead to anomalies being included in the results.

Suppose the 2D random walk steps are each stored in a vector of size N, where N is the number of time steps taken by the random walker:

$$x = [x_1, x_2, ..., x_N],$$

$$y = [y_1, y_2, ..., y_N].$$

If the simulation is ran M times, then we would have M random walkers each labelled by a superscript:

$$x^{(1)} = [x_1^{(1)}, x_2^{(1)}, ..., x_N^{(1)}], \quad y^{(1)} = [y_1^{(1)}, y_2^{(1)}, ..., y_N^{(1)}],$$

$$x^{(2)} = [x_1^{(2)}, x_2^{(2)}, ..., x_N^{(2)}], \quad y^{(2)} = [y_1^{(2)}, y_2^{(2)}, ..., y_N^{(2)}],$$

$$\vdots$$

$$x^{(M)} = [x_1^{(M)}, x_2^{(M)}, ..., x_N^{(M)}], \quad y^{(M)} = [y_1^{(M)}, y_2^{(M)}, ..., y_N^{(M)}].$$

When enough steps are sampled and enough random walkers are generated, we can ensemble average across the M number of samples to obtain the mean displacement at each time step:

$$\mu_{x,i} = \frac{1}{M}\sum_{j=1}^{M} x_i^{(j)}, \quad \mu_{y,i} = \frac{1}{M}\sum_{j=1}^{M} y_i^{(j)}.$$

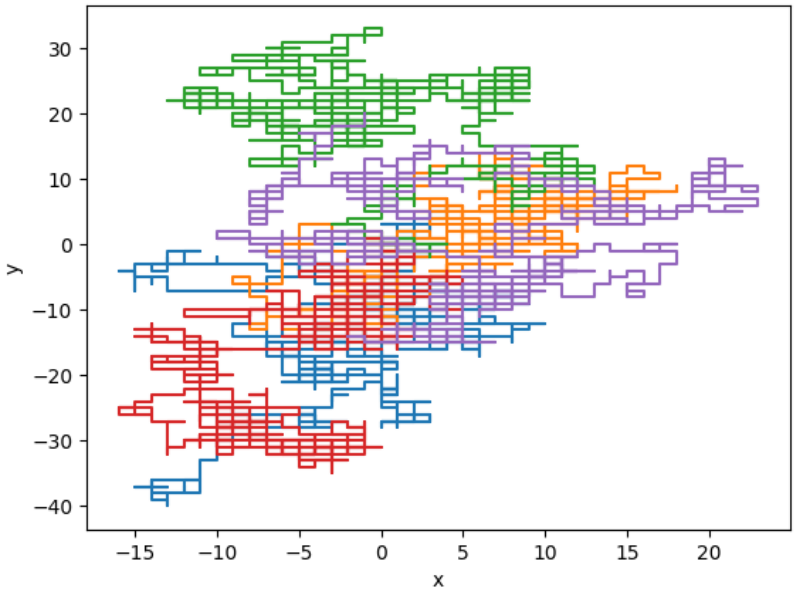

Figure 3.3: Displacements of 5 random walkers. Each random walker has taken 1000 steps from its initial position at (0, 0). A different color is used for each random walker.

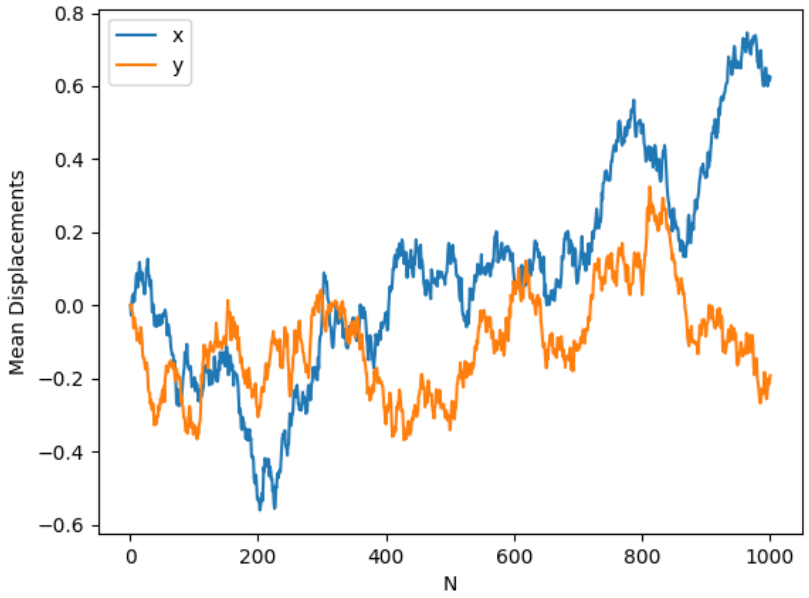

Figure 3.4: Mean displacements in the x and y directions of random walkers that took 1000 steps. The displacements were ensemble averaged across 1000 random walkers. We expect that as $M \to \infty$, the average tends to zero. But the variance should increase with time.

In the example shown in Figure 3.2, the random walker is restricted to either step in the x direction or the y direction with equal likelihood i.e. $p = 0.25$. That was dependent on the value of p which was the same in all 4 directions. Now suppose that the walker has a bias of stepping in the right direction. Here we will consider that p is the probability of taking a step in the right direction and q is the probability of stepping in the other directions. To have a bias towards a certain direction, p must be greater than q. If $p = 0.7$, then $q = 0.3$ and the walker only has a probability of 0.1 to take a step in one of the three directions.

3.1.3 Nerve Fibres depicted as Random Walkers

We attempt to synthesize our own microscope images by simulating 2D biased random walks where the bias is simply applied to the positive y direction as shown in Figure 3.5. The lattice is analogous to the spinal cord region with the y-axis being the longitudinal direction along the spinal cord and the x-axis being the transverse direction that sections the cord. Each walker represents a nerve fibre travelling upwards across the spinal cord region towards the CNS. In this random walk model, each nerve fibre is allowed to take a step in all neighbouring discrete sites except in the negative y direction. The value of p is the probability of the nerve fibre taking a step in the positive y direction, while q is the probability of stepping in any of the remaining allowed directions. Recall that $q = 1 - p$. Bright white pixels are used to represent myelin and dark black pixels represent axons in between myelin and empty regions. The bias simply reflects the fact that the nerve fibres develop by growing in a longitudinal direction.

An empty lattice of size 1000x1000 is defined for the simulation. In a single simulation, nerve fibres are generated and each nerve fibre is allowed to travel 1000 steps across the lattice. After the simulation is completed, a region of size 500x500 is selected from the lattice and is used as a synthetic image of longitudinal nerve fibres. The selected region is of size 500x500 in order to match the size of the images used to train the CNN model. The random walkers all start at the bottom of the 1000x1000 lattice. The initial position of each walker is separated by one discrete space.

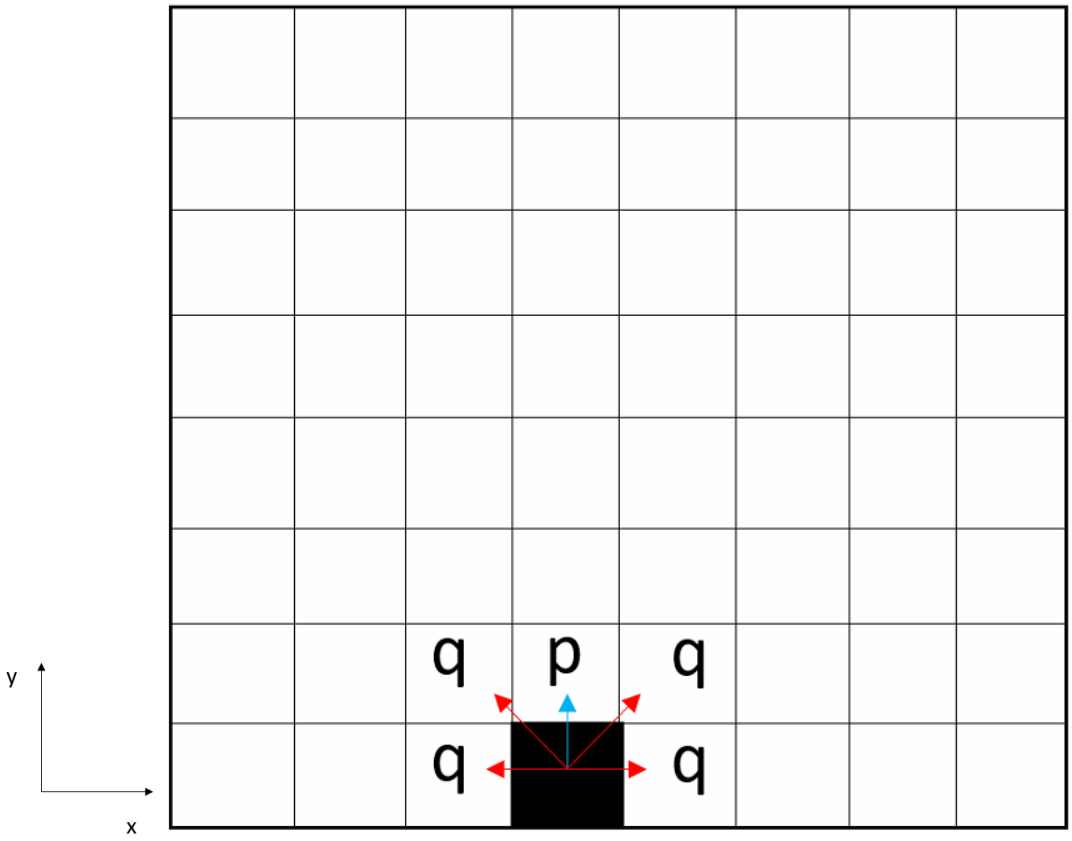

Figure 3.5: A random walker at its starting point (black box) with a likelihood p of stepping in the positive y direction and a likelihood q of stepping in the remaining allowed directions.

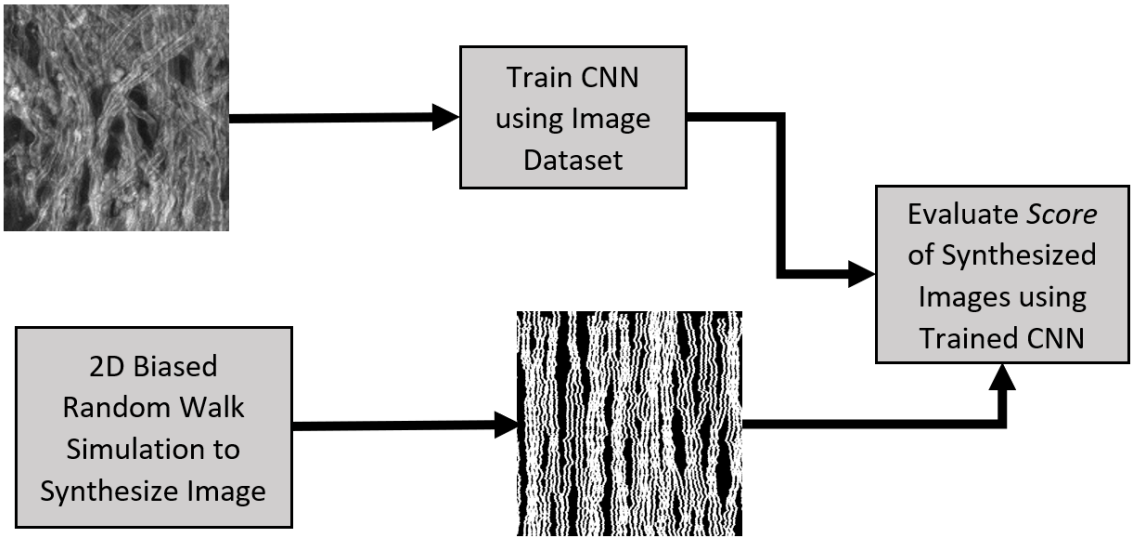

Figure 3.6: Flow Chart for CNN-Simulation joint approach.

3.2 A continuous score for CNN outputs

Since the CNN outputs a discrete score ranging from 0 to 5 in the order of demyelination severity, we are not able to distinguish how far the disease has progressed within a specific labelled decision. Thus, we attempt to produce a continuous score based on the weighted average of each label's probability value [68]. The CNN outputs one of the following discrete scores for its decision:

$$s = [0, 1, 2, 3, 4, 5],$$

$$Sal = 0, \quad CFA = 1, \quad Sc0 = 2, \quad Sc1 = 3, \quad Sc2 = 4, \quad Sc3 = 5.$$

The CNN makes a decision based on a list of probability values as the following example:

$$l = [0.00, 0.00, 0.05, 0.70, 0.15, 0.10]$$

The above list indicates that the model is 70% certain that the image will have a Sc1 label and since Sc1 has the largest probability value, the CNN model would output a discrete score of 3. But now, we produce a continuous score value as follows:

$$Score = \sum_{s=0}^{5} sl_s.$$

Here, the probability values act as weightings for the discrete scores. In the example we have for l, we would obtain the following score:

$$Score = 0.00(0) + 0.00(1) + 0.05(2) + 0.70(3) + 0.15(4) + 0.10(5) = 3.30.$$

With this score, the CNN now outputs its decision along with a measure of how close the disease severity is to the actual label. So we are able to both classify the different labels from each other and differentiate images that have the same label from each other. Healthier images will have a lower CNN score.

3.3 Modelling Nerve Fibres

In the CARS images, the pixels with high intensities, i.e. bright white pixels, represent the myelin sheath covering the axon [69–73]. The pixels with low intensities, i.e. dark black pixels, show the axons that are surrounded by the myelin. With these details, we set the pixel intensities of the random walkers to be 255 for myelin and 0 for axons. The filters of the trained CNN model in chapter 2 should be able to detect higher

intensities as myelin and lower intensities as the axons. The random walk analysis was performed in the context of longitudinal CARS images. Since the myelin is found on both edges of the axons in a 2D sectional image, a single simulated nerve fibre would consist of two adjacent random walkers experiencing identical journeys, each representing one side of the myelin sheath in a 2D cross-section. The gap between these two random walkers would have zero pixel intensity and thus will represent the axon of the nerve fibre.

The parameters of the simulation that can be adjusted are the myelin thickness m, the axon diameter d, and the value of p. By adjusting the values of m and d, we alter the g-ratio of the nerve fibre which is defined as follows [74]:

$$g\text{-}ratio = \frac{d}{2m + d}.$$

It is the ratio of the axon diameter to the thickness of the nerve fibre. Since axons are surrounded by myelin from both sides, the total myelin thickness is $2m$, thus the total nerve fibre thickness is $2m + d$. It has been observed that ideal values of the g-ratio for healthy nerve fibres fall between 0.5-0.7; larger healthy nerve fibres are expected to have a larger g-ratio. Axon diameters may range from $1\mu m$ to $6\mu m$ in mice [75], with smaller diameters being more common. A single pixel in a CARS image is $225nm$ [7] which is approximately $0.2\mu m$. If we want to simulate a nerve fibre with $d = 1\mu m$, then the axon would have to be 5 pixels wide.

In the human's nervous system, axons were observed to have axon diameters as thin as $0.3\mu m$ [76] and as large as $20\mu m$ [77] with most of them being smaller than $2\mu m$. Although the axon diameter range for humans is wider than that of mice, the assessment of demyelination in mice can still give insights into that of humans due to the similar physiology. This also opens up a need to explore demyelination image datasets for humans to look for differences between the mice and human dataset results. From our image dataset we decided to choose three different values of d. These values were chosen based on visual inspection of the images of what would seem to be the possible axon diameters. These diameters range from approximately $1\mu m$ to $3\mu m$. A simulation is performed for each value of $d \, \epsilon \, (1, 2, 3)\mu m$. By selecting a specific range for axon diameters, we are able to provide a constrained and realistic exploration of the synthetic random walk model. In each simulation, we generate images with the same value of d and vary the value of m. The value of m always starts at 1 pixel and stops changing when $m = d$. For example, when $d = 1\mu m$ i.e.

$d = 5$ pixels, m will be an integer value from 1 pixel to 5 pixels. This generates five simulated images with $d = 1\mu m$. Each image will then be fed into our trained CNN in order to analyze how the *Score* value varies with the g-ratio for $d = 1\mu m$. The number of nerve fibres n generated is determined using the following equation:

$$n = \frac{1000}{2m + d + D}.$$

This equation ensures that nerve fibres' starting points span across the entire lattice and are separated by D. The value of D is a parameter that controls the spacing between the synthetic nerve fibres. In this thesis, D was set to 1 pixel given how closely packed the nerve fibres are.

In the results below, we will be simulating images varying the myelin thickness m for different values of p while keeping the axon diameter d constant. We repeat these simulations for other values of d. The images would then be fed into the trained CNN in order to investigate the how the *Score* value is affected by the simulation parameters.

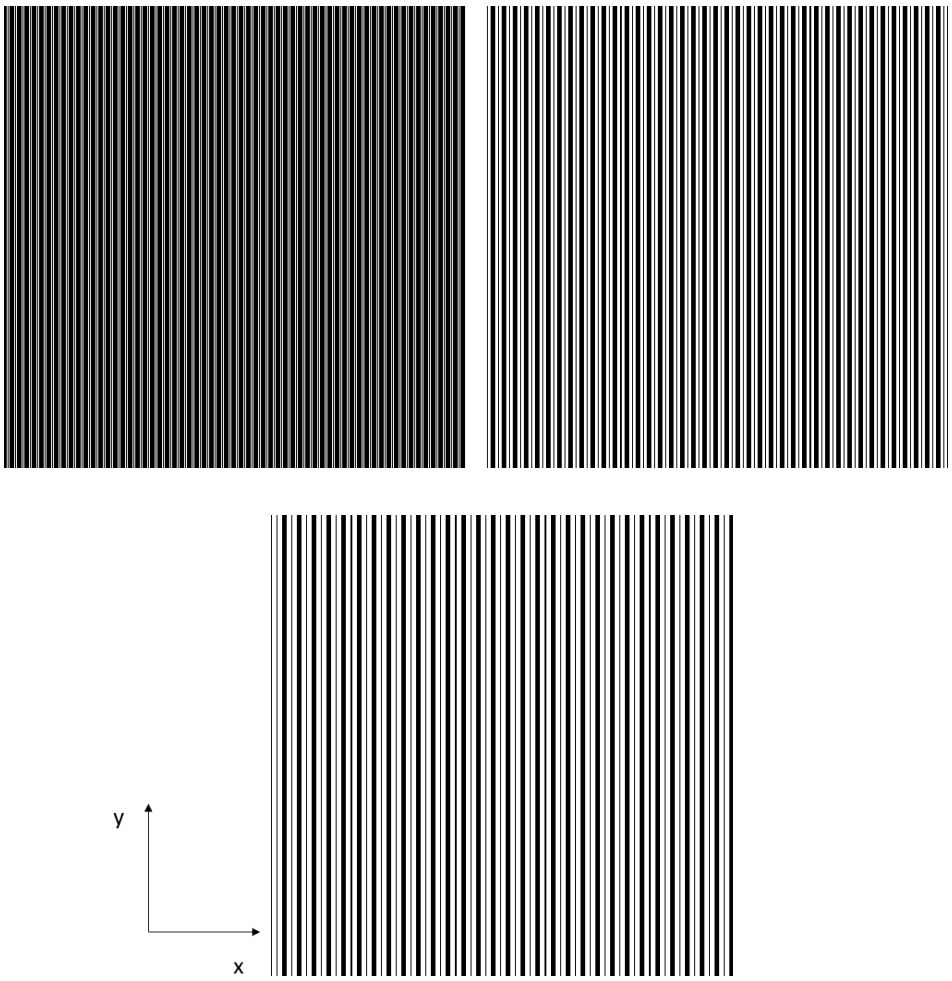

Figure 3.7: Synthesized image of longitudinal nerve fibres with size 500x500 pixels, *p=1*, and *d=5* pixels. Black pixels represent axons and white pixels represent myelin. The values of m used were 1, 3, and 5 pixels. One pixel separates each fibre at their starting points, e.g. the gap between the outer right side of one sheath and the outer left side of the neighbouring sheath to its right. The thick black lines represent the axons surrounded by the myelin sheath and the thin black lines represent the one pixel gap.

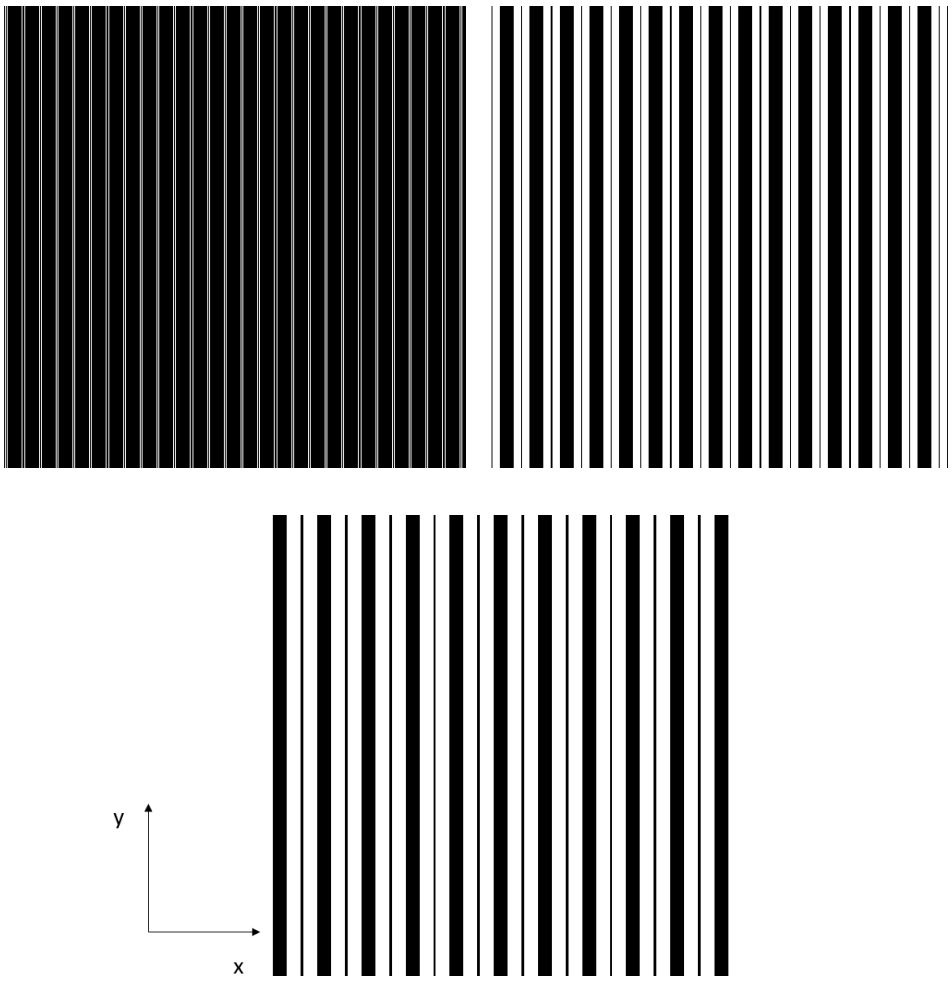

Figure 3.8: Synthesized image of longitudinal nerve fibres with *p=1* and *d=15* pixels. Black pixels represent axons and white pixels represent myelin. The values of m used were 1, 8, and 15 pixels. One pixel separates each fibre at their starting points.

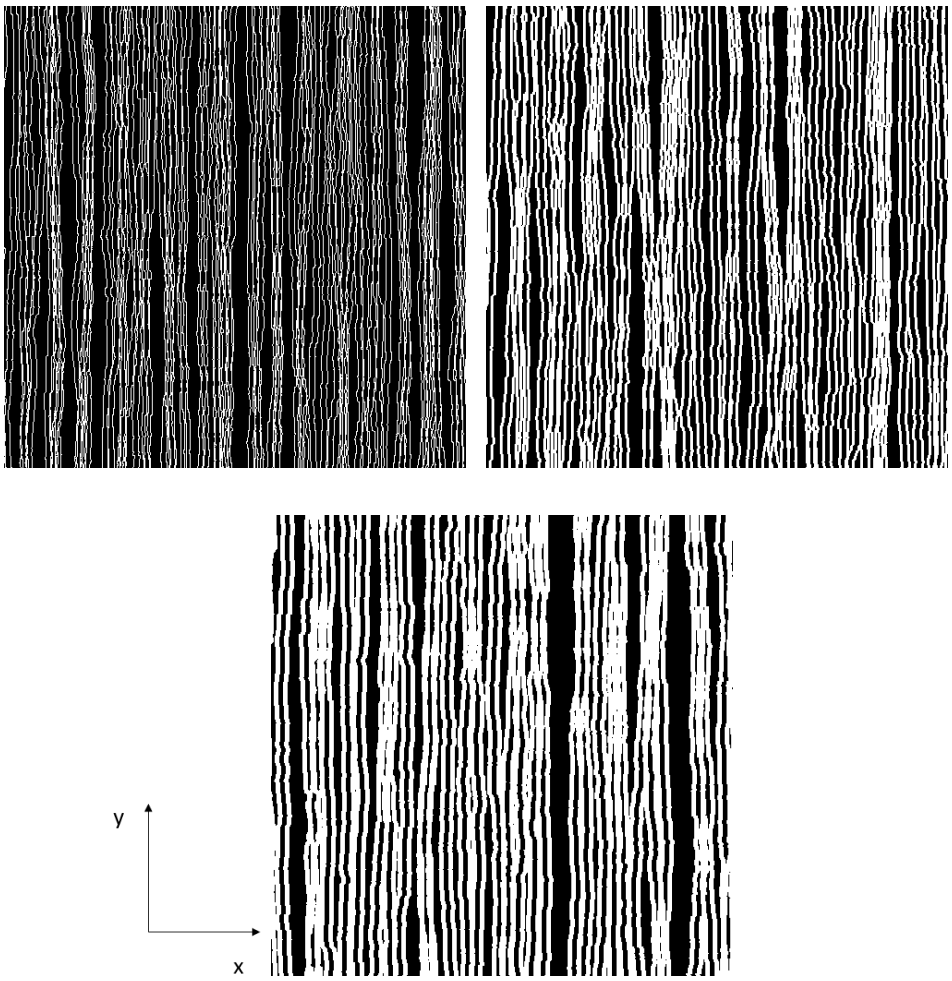

Figure 3.9: Synthesized image of longitudinal nerve fibres with *p=0.9* and *d=5* pixels. Black pixels represent axons and white pixels represent myelin. The values of m used were 1, 3, and 5 pixels. One pixel separates each fibre at their starting points.

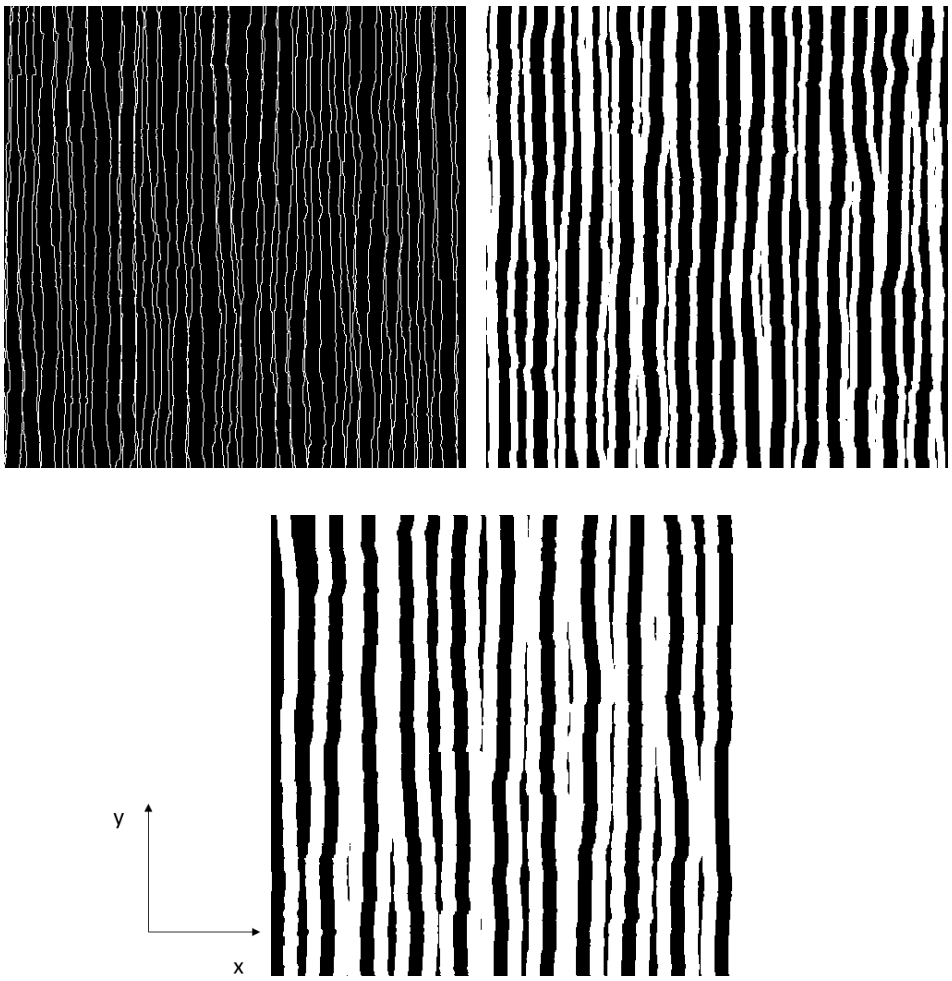

Figure 3.10: Synthesized image of longitudinal nerve fibres with *p=0.9* and *d=15* pixels. Black pixels represent axons and white pixels represent myelin. The values of m used were 1, 8, and 15 pixels. One pixel separates each fibre at their starting points.

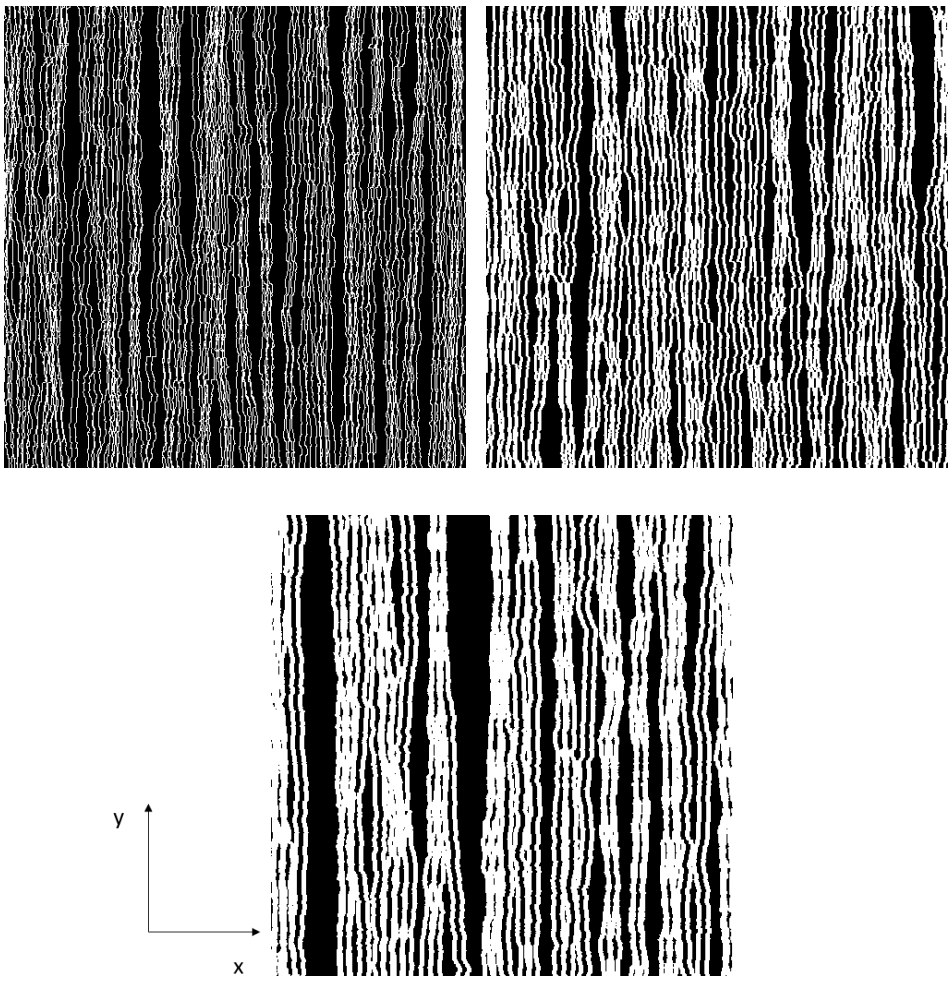

Figure 3.11: Synthesized image of longitudinal nerve fibres with *p=0.8* and *d=5* pixels. Black pixels represent axons and white pixels represent myelin. The values of m used were 1, 3, and 5 pixels. One pixel separates each fibre at their starting points.

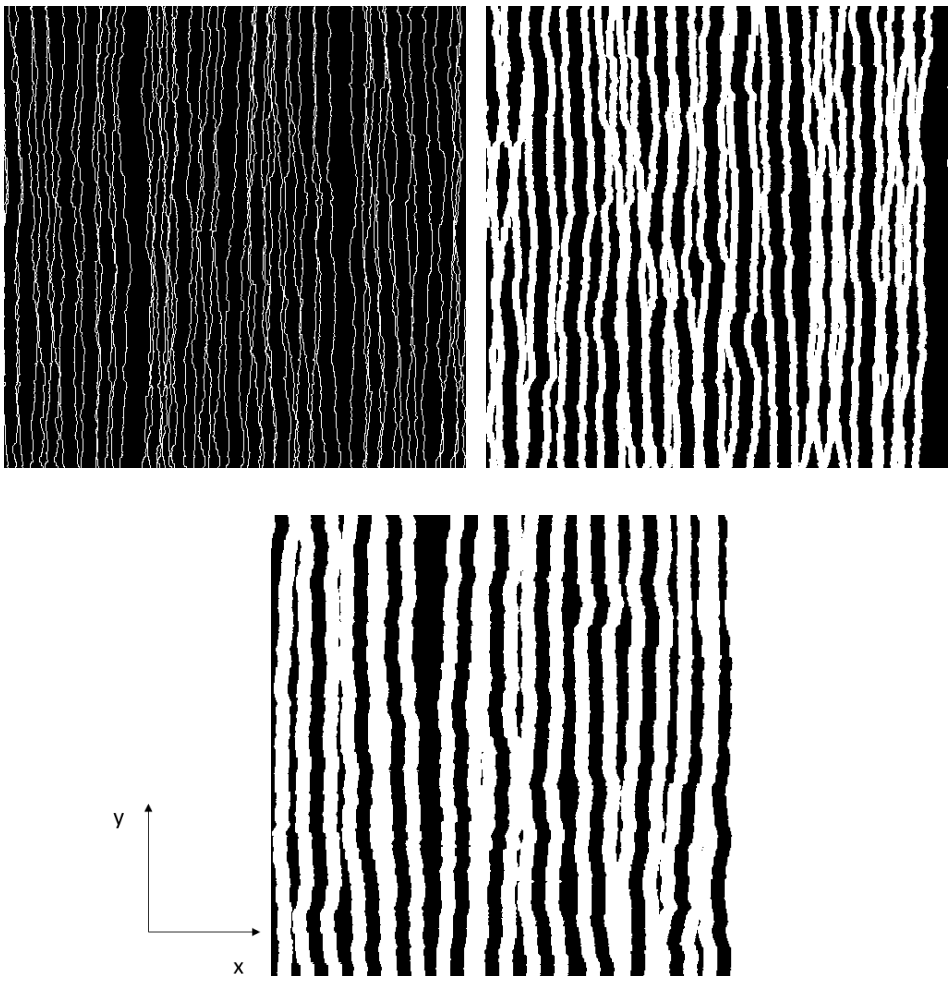

Figure 3.12: Synthesized image of longitudinal nerve fibres with *p=0.8* and *d=15* pixels. Black pixels represent axons and white pixels represent myelin. The values of m used were 1, 8, and 15 pixels. One pixel separates each fibre at their starting points.

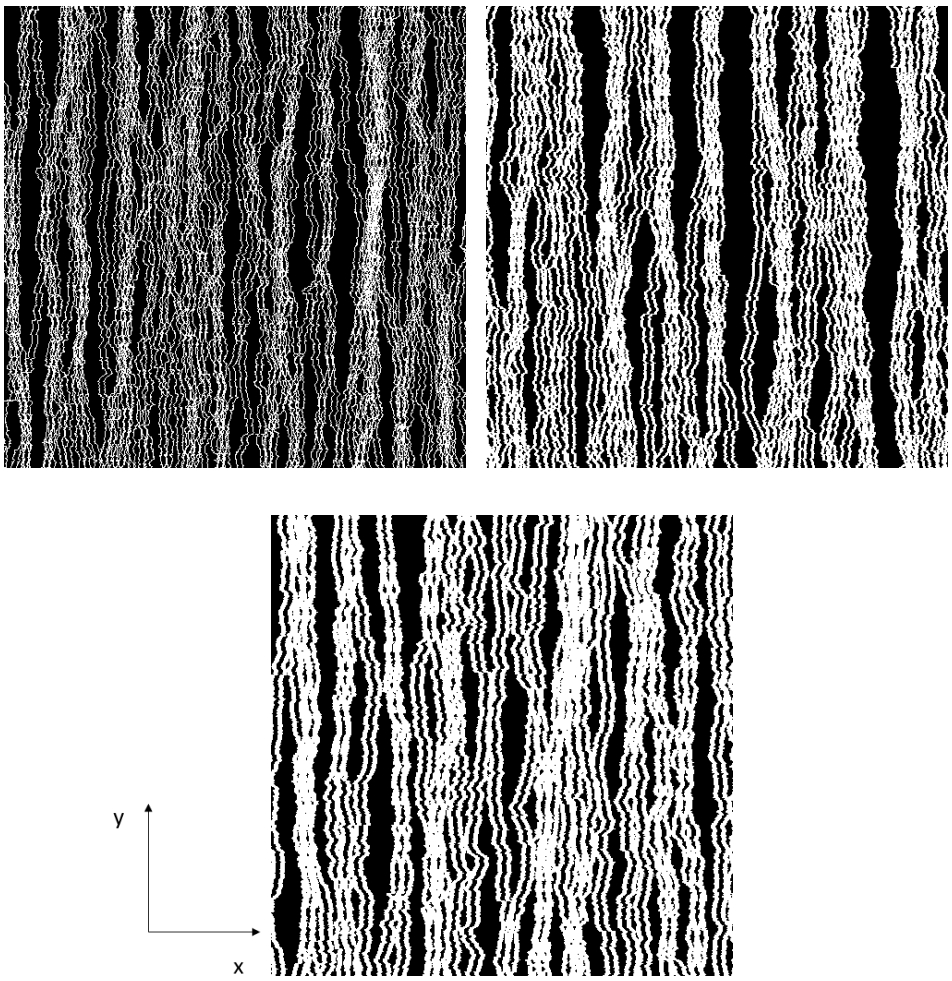

Figure 3.13: Synthesized image of longitudinal nerve fibres with *p=0.5* and *d=5* pixels. Black pixels represent axons and white pixels represent myelin. The values of m used were 1, 3, and 5 pixels. One pixel separates each fibre at their starting points.

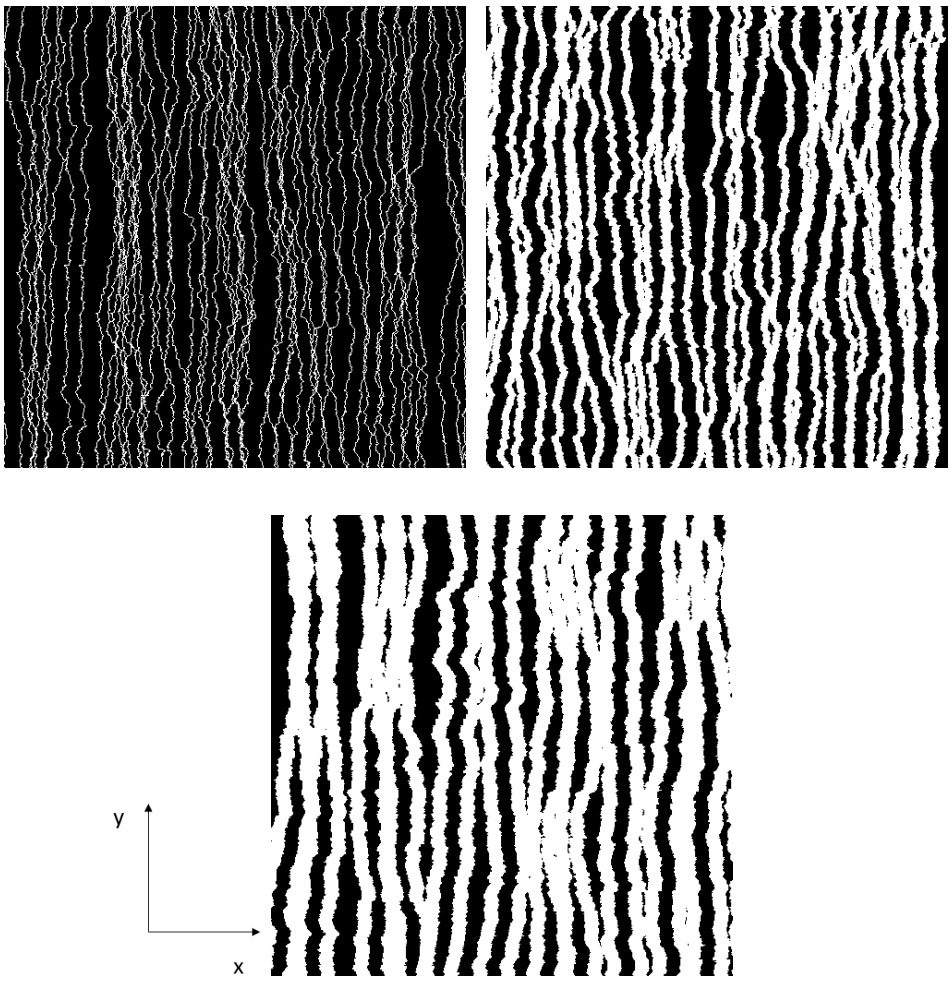

Figure 3.14: Synthesized image of longitudinal nerve fibres with *p=0.5* and *d=15* pixels. Black pixels represent axons and white pixels represent myelin. The values of m used were 1, 8, and 15 pixels. One pixel separates each fibre at their starting points.

3.4 Simulation Results and Discussion

In this thesis, we have trained a CNN on CARS microscope images showing regions of mice spinal cord. Each set of images from a mouse shows nerve fibres inflicted with EAE that show different degrees of demyelination. The degree of demyelination of the mice was evaluated through visual observation of their clinical behavior. The evaluation was based on a clinical scoring system ranging from 0 to 3. Alongside the images of mice that start to show signs of EAE, we have two control groups, CFA and Sal, that are considered healthy. These mice had their spinal cords undergo imaging before demyelination started to affect the nerve fibres. We then used the images to train a CNN using supervised learning where we had six labels in total, two control groups and four clinical scores.

The trained CNN has learnt weights through optimization using Minibatch Stochastic Gradient Descent (MSGD) and Backpropagation where the categorical cross-entropy of the model was minimized. The loss function of the model measures the uncertainty of the model's predictions. This trained and tested CNN was used to perform predictions on our synthesized images using the weights that gave the lowest uncertainty for the experimental longitudinal spinal images.

The images were synthesized using a 2D Biased Random Walk simulation where random walkers represented the nerve fibres. The aim of this experiment was to use simulation parameters to model the structural features of the real images and encode them into the synthetic images. By feeding the synthesized images into our classifier model, we expected the CNN to capture these features and perform a prediction using our *Score* value. The CNN *Score* is a continuous prediction value that gives a precise indication of how far the disease has progressed.

The simulation used the probability parameter p to define the likelihood of the random walker taking a step in the positive y-direction, i.e. taking an upwards longitudinal step through the spinal cord region. This is the parameter that provides the bias in the positive y-direction since that is observed in the original CARS images. To study how the size of a nerve fibre affects its health, we introduced two other simulation parameters. These parameters are the axon diameter d and myelin thickness m. Both d and m can be used to find the g-ratio of an axon which is the ratio of the axon diameter to the total thickness of a nerve fibre. Each value used for the

probability $p \in (1, 0.9, 0.8, 0.5)$ was used to study how the CNN *Score* is affected as the g-ratio varies. The parameters d and m are adjusted to change the g-ratio.

The value of p determines how consistent the journey of the nerve fibre is. When $p = 1$, the nerve fibre will only travel forward indicating that the bias towards the positive y direction is at its full effect. p seems to be responsible for what tortuosity each nerve fibre will have. The tortuosity t is the ratio of the actual length of the fibre L_T to the Euclidean distance between the starting and end points L:

$$t = \frac{L_T}{L}.$$

A tortuosity greater than 1 indicates the presence of disorder while a tortuosity of exactly 1 shows no disorder. Since it was observed that healthier fibres have stronger collinearity, then synthetic images with $p = 1$ will have $t = 1$ and are expected to receive the lowest CNN *Score* values.

The g-ratio shows how large the diameter of the axon is relative to the thickness of the myelin sheath surrounding it. In this thesis, we aim to find the ideal g-ratio for axonal propagation by varying the value of d. The ideal g-ratio is the g-ratio value that minimizes the *Score* value of an image i.e. what is the healthiest score an image can receive given its tortuosity. Theoretically, it has been found that the g-ratio value that is ideal for action potential propagation varies between 0.5 and 0.7 [14–19]. By feeding our simulated images into the CNN trained on the microscope images, we are able to study how the *Score* varies with the g-ratio of the nerve fibres. From the training results of the CNN, we are able to develop trust in the decisions made by the classifier as it seems to indeed have learned the important features that classify demyelination amongst different nerve fibres.

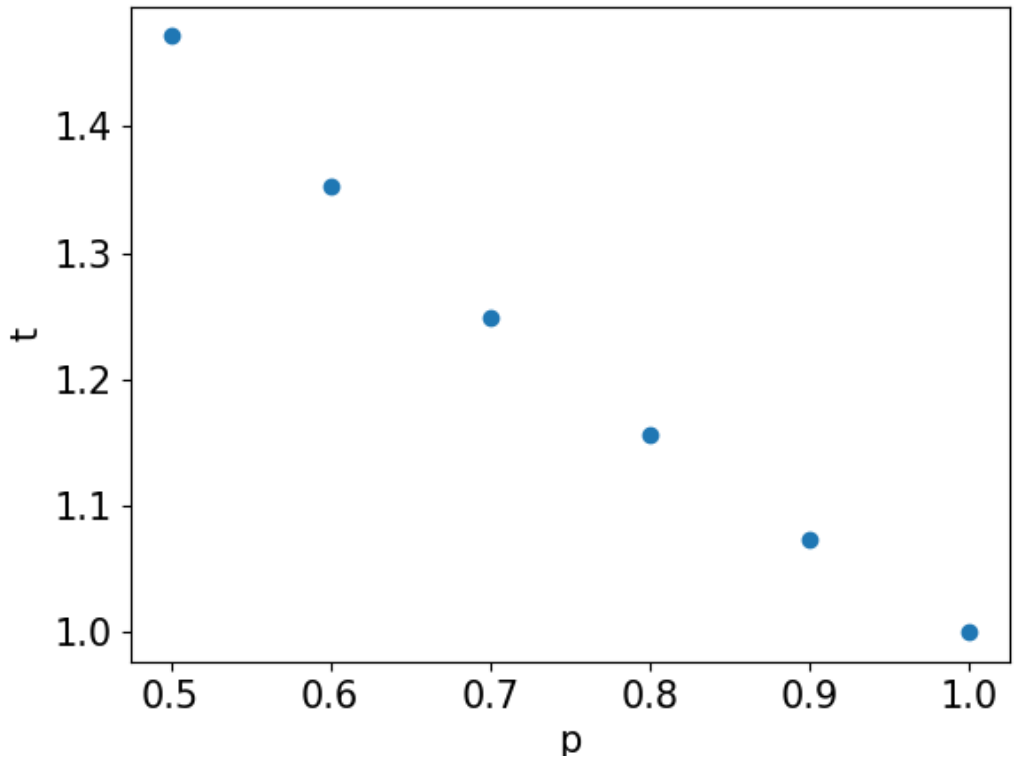

Figure 3.15: Tortuosity t varying with the the simulation parameter p. As expected, $t \to 1$ as $p \to 1$ since the simulated fibre disorder decreases.

Figures 3.7 to 3.14 show examples of the images that were fed into the trained CNN in chapter 2. The CNN outputs the continuous *Score* value as a way to assess the healthiness of the synthetic images. Figure 3.15 shows how the tortuosity varies with different values of p. Figures 3.16 to 3.20 show that when $p = 1$, i.e. when the tortuosity is 1 as seen in Figure 3.15, nerve fibres can have several g-ratio values and be considered healthy. Once the axon diameter reaches $2\mu m$ or more, lower g-ratio values received a higher score. Specifically, they seem to be recognized by the CNN as Sc1 images. In the clinical assessment of the mice, Sc1 was given to mice that had a limp tail. The remaining g-ratio values received *Score* values that were around 0 or 1, which represent the control group labels CFA and Sal. The Sal control group was found to be healthier than the CFA group. The results show that the healthiest synthesized fibres will most likely have an ideal or close to ideal tortuosity.

Just by changing the value of p from 1 to 0.9, we notice a significant change in the distribution of *Score* values. On a first look, one can see that shifting from an ideal tortuosity to a value greater than one has a strong impact on the perceived healthiness of the fibres. The fibres now start to have a range of g-ratio values where they are found to be in their healthiest state based on the value of d for the axons in the synthesized images. For g-ratios close to 0.90, the fibres are always considered unhealthy.

The starting point for the three values of p differs, becoming unhealthier as p decreases. It is interesting to note that despite having an ideal g-ratio, the fibres might still not be considered healthy. A spinal cord region is considered healthy if the *Score* value represents a label where mice do not exhibit clinical abnormalities. In the clinical assessment's case, these labels would be Sal, CFA, and Sc0. We see that this is likely to be achieved for $p = 0.9$ and $p = 0.8$ especially at ideal g-ratios, however, it is difficult for $p = 0.5$ as it seems that the tortuosity at this point is too severe.

For low g-ratios, we can see that the fibres are considered to be less healthy compared to moderate values of the g-ratio. This is an indication that there is a certain range of values for the g-ratio where nerve fibres are considered healthy. The transition from moderate to low g-ratios mimics the effect of demyelination on the healthiness of the nerve fibres. The simulation results are capable of quantifying the decrease in health due to the damage incurred to the myelin. As the value of d increases, the ideal g-ratio starts increasing, thus widening the range of unhealthy g-ratio values that are below the ideal g-ratio. Although we have seen the value of p affecting the healthiness of a region for a given g-ratio value, it seems to not have an influence on the pattern of Figures 3.14 and 3.15. It only has affects on the pattern when the value of p changes from 1 but aside from that, the *Score* value just increases when p decreases.

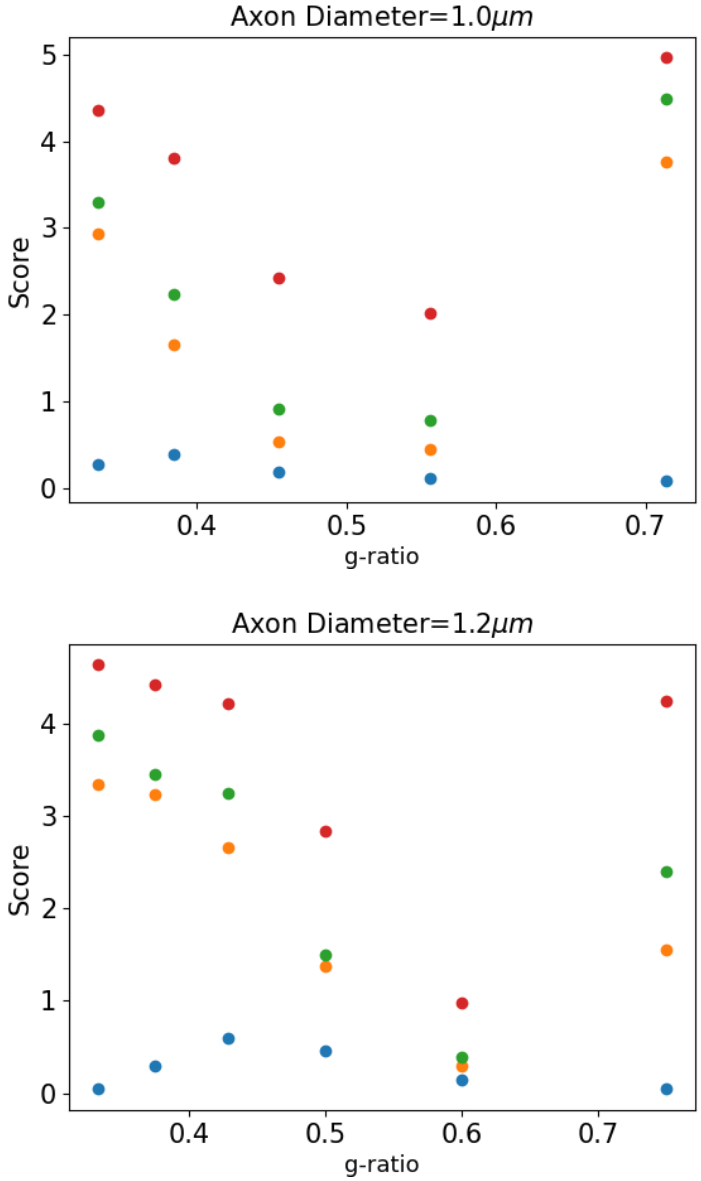

Figure 3.16: The value of the *Score* varying with the g-ratio for axon diameters $1\mu m$ and $1.2\mu m$. Different values of p were used: Blue shows $p = 1$, Orange shows $p = 0.9$, Green shows $p = 0.8$, and Red shows $p = 0.5$. Fibres with $p = 1$ seem to remain healthy despite the g-ratio. At high values of the g-ratio, fibres with $p < 1$ show a minimum peak for the *Score* value, indicating that the fibres are at their best health around the minimum *Score* value and less healthy outside of the peak.

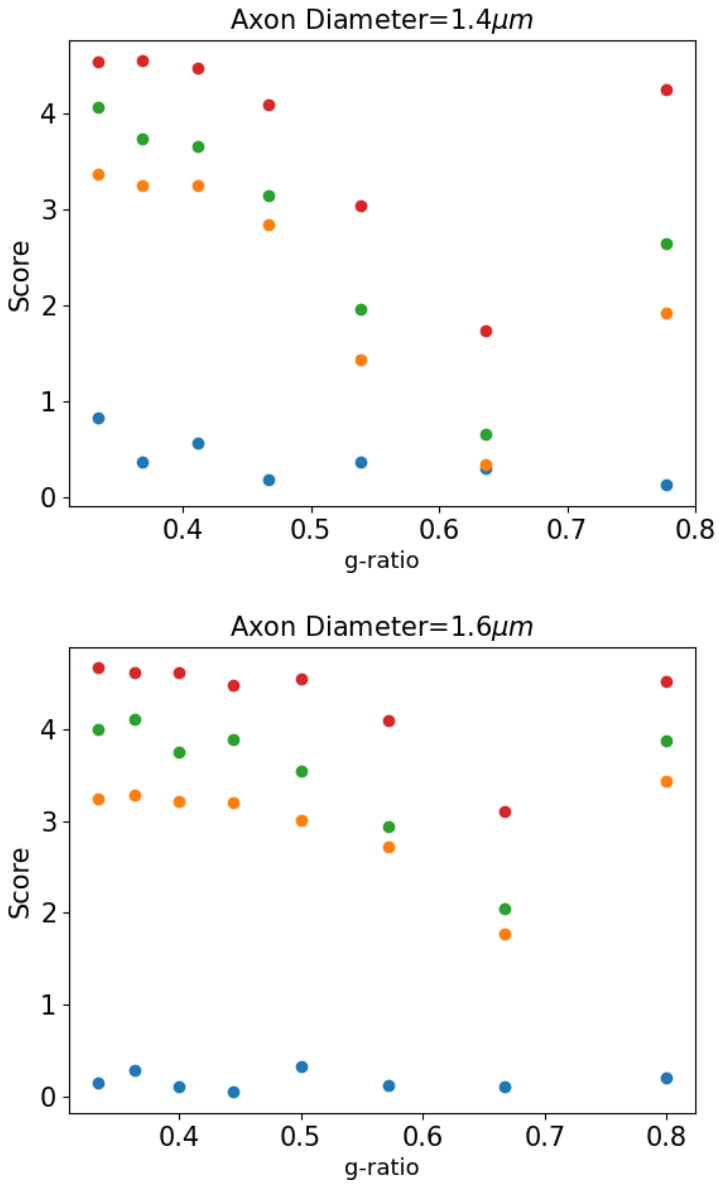

Figure 3.17: The value of the *Score* varying with the g-ratio for axon diameters $1.4\mu m$ and $1.6\mu m$. Different values of p were used: Blue shows $p = 1$, Orange shows $p = 0.9$, Green shows $p = 0.8$, and Red shows $p = 0.5$.

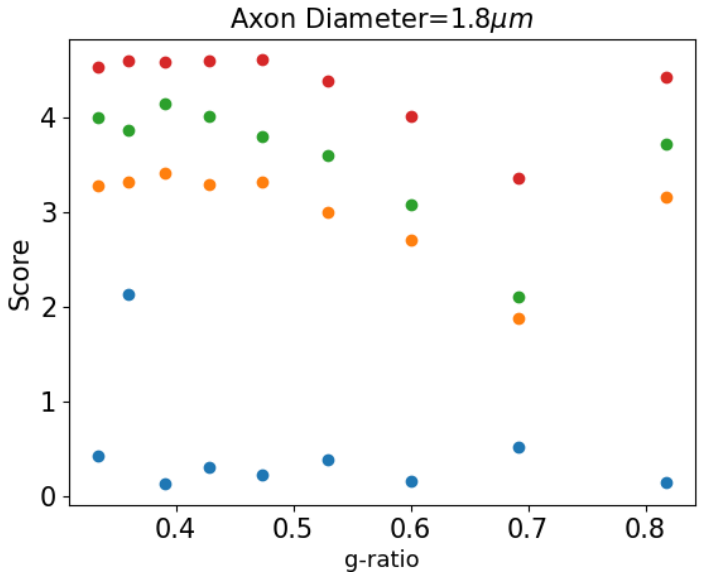

Figure 3.18: The value of the *Score* varying with the g-ratio for axon diameter $1.8\mu m$. Different values of p were used: Blue shows $p = 1$, Orange shows $p = 0.9$, Green shows $p = 0.8$, and Red shows $p = 0.5$. For $p = 1$, there is an abrupt increase in the *Score* value when the g-ratio is 0.36. It is unclear why the *Score* value behaves this way at this g-ratio.

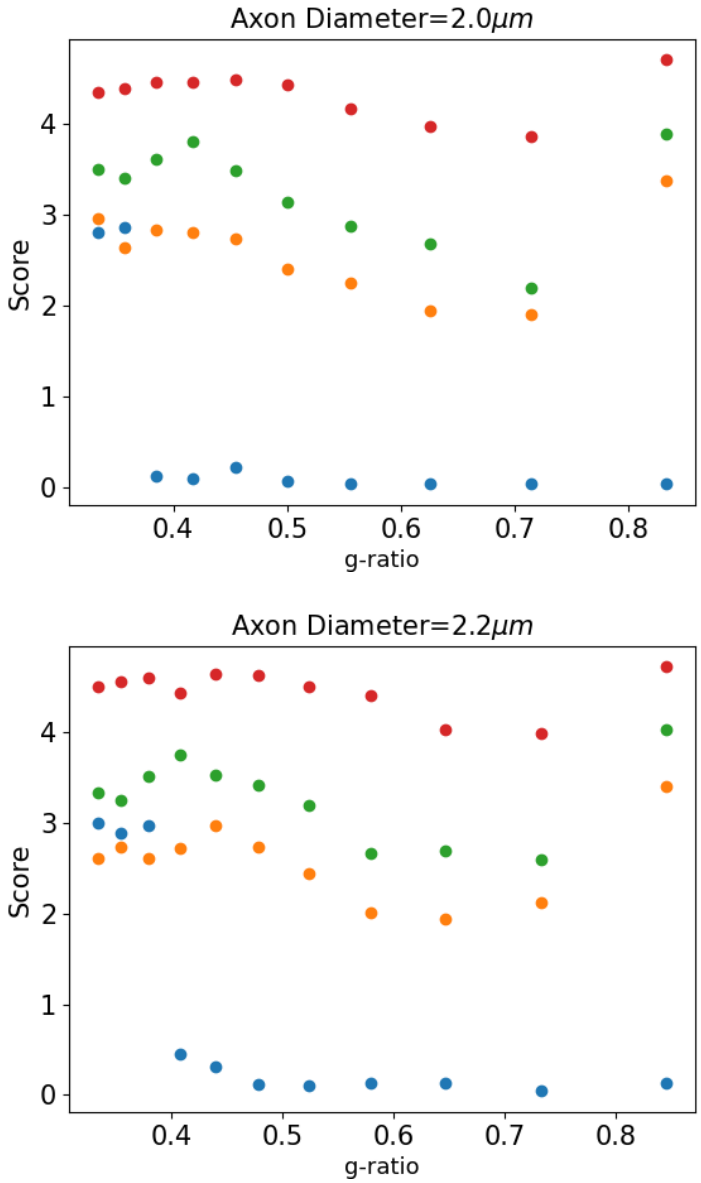

Figure 3.19: The value of the *Score* varying with the g-ratio for axon diameters $2\mu m$ and $2.2\mu m$. Different values of p were used: Blue shows $p = 1$, Orange shows $p = 0.9$, Green shows $p = 0.8$, and Red shows $p = 0.5$. The *Score* values for $p = 1$ have an abrupt decrease at low g-ratio values. It is not clear why the *Score* value decreases in such a fashion.

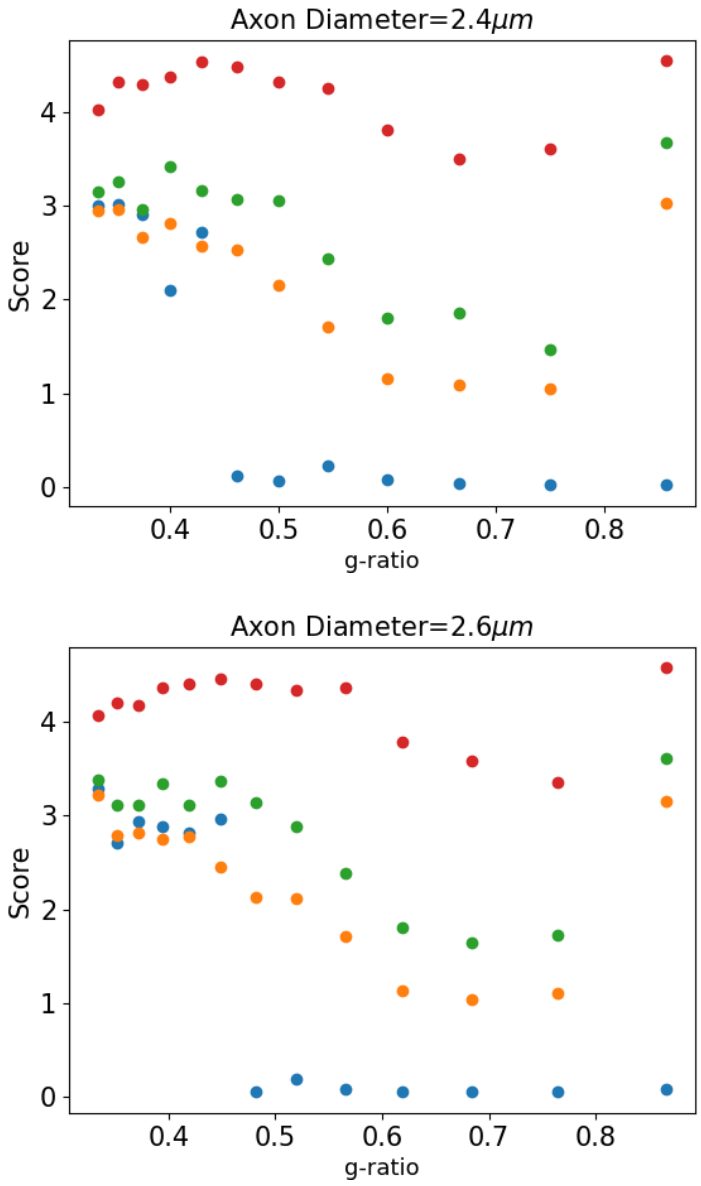

Figure 3.20: The value of the *Score* varying with the g-ratio for axon diameters $2.4\mu m$ and $2.6\mu m$. Different values of p were used: Blue shows $p = 1$, Orange shows $p = 0.9$, Green shows $p = 0.8$, and Red shows $p = 0.5$.

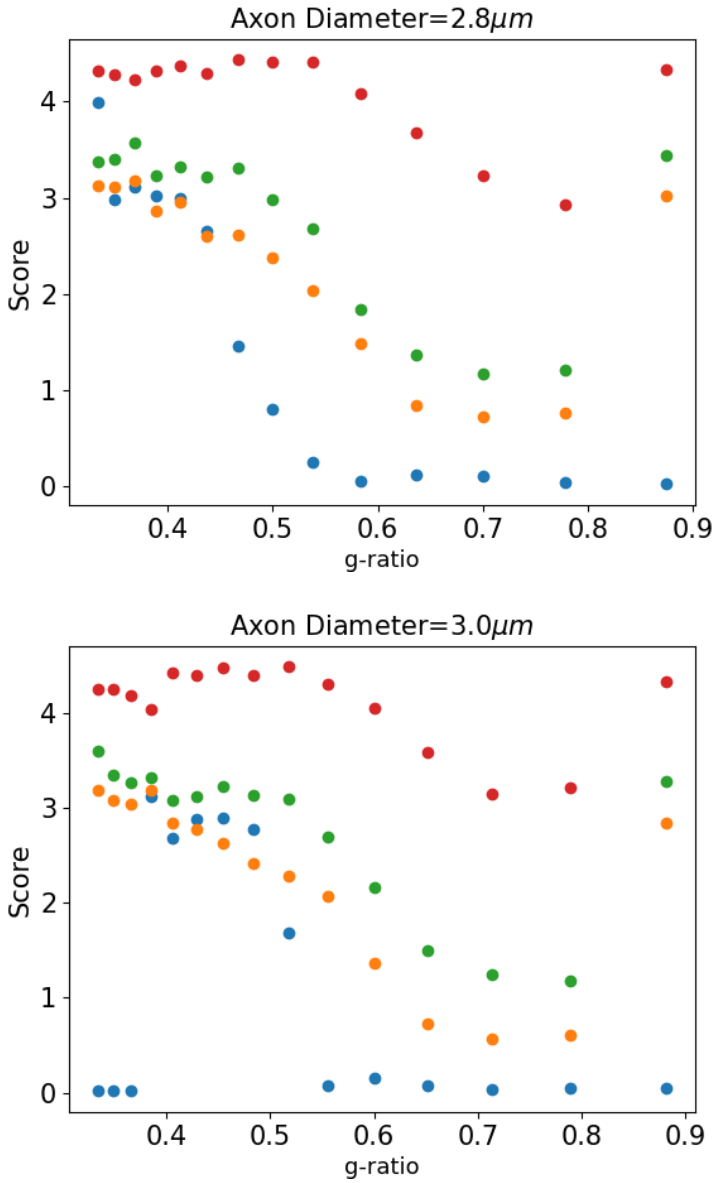

Figure 3.21: The value of the *Score* varying with the g-ratio for axon diameters $2.8\mu m$ and $3\mu m$. Different values of p were used: Blue shows $p = 1$, Orange shows $p = 0.9$, Green shows $p = 0.8$, and Red shows $p = 0.5$. The *Score* values for $p = 1$ have an abrupt increase at low g-ratio-values. It is not clear why the *Score* value increases in such a fashion.

Chapter 4

Conclusion and Future Work

4.1 Summary of Problem and Methods

In this thesis, we used CARS Microscope images to train a CNN classifier and feed our own synthesized images into the trained model to study demyelination diseases such MS. The microscope images showed longitudinal nerve fibres of mice spinal cord regions representing different degrees of demyelination, which is an accurate representation of MS. By better understanding the consequences of demyelination on the biophysical properties of nerve fibres, we are able to develop a connection between the physics and the clinical implications of the disease. With this information we will be able to take structural measurements from patients using Diffusion Tensor Imaging (DTI) and figure out exactly how far the disease has progressed.

The images belonged to six different labels: Sal, CFA, Sc0-Sc3. The clinical scores are assigned based on the visually assessed behavior of the mouse. Previously, measurements of the collinearity of neighbouring domains were used to classify and assess the images. While this is effective for quantifying demyelination from images, it does not give insights into the characteristics of individual nerve fibres.

Our CNN was trained on the microscope image dataset using a modified version of the VGG16 architecture. Three different models were trained with different number of filters. It was found that 32 filters were sufficient to achieve a model with reliable results. The performance metrics are all indicative that the CNN has actually learned some features embedded in the images due to low number of FPs and FNs.

The images were fed to the trained CNN and are assigned a score value to indicate the severity of demyelination. The score values were in the range of 0 to 5 with a larger score representing more severe demyelination. A 2D Biased Random Walk simulation was then used to synthesize images with known physical parameters. The 2D random walk allowed us to simulate the degree of disorder by controlling p which is the likelihood of a random walker stepping in the positive y-direction. In this case, the spatial pattern formed by the simulation using the assigned value of p establishes the healthiness of the nerve fibres. Images with lower values of p received higher scores from the CNN, indicating that the CNN observed a stronger presence of demyelination.

This method has allowed us to simulate the degree of disorder for demyelination using the value of p and relate it to the *Score* value which is the CNN's quantification of the axonal damage in the image. We have also varied the g-ratio to understand what is the optimal myelin thickness for different axon diameters. This will help in understanding when demyelination could be observed in axons of specific diameters. If a region of myelin deviates from the optimal g-ratio, then this region could be experiencing demyelination. As the axon diameter increases, it seems that the optimal g-ratio increases. Interestingly, we see that the degree of disorder greatly affects the healthiness of the images. A slight change from $p = 1$ to $p = 0.9$ seems to push the nerve fibres into the onset of the disease when $Score = 3.00$, especially for smaller axons. This value of the *Score* is equivalent to the Sc1 label where symptoms of EAE start to be observed. So according to the CNN, a slight change in the degree of disorder can lead to diseased conditions.

The CNN-simulation joint approach from chapter 3 Figure 3.6 showed that it is possible to constrain demyelination models to the g-ratio and degree of disorder of the spinal cord region. The approach was useful as an attempt to understand the features that the CNN has learnt to detect and to quantify the parameters involved in the demyelination process. This could open a wide variety of research questions and future analysis in using microscope images of biological structures and CNNs to relate damage to dysfunction.

4.2 Limitations of Model

The synthesized images are of course not identical to the real microscope images which are the data that the CNN used to train on. The original images have varying pixel values since the actual nerve fibres do not traverse through the spinal cord within the same plane. They will tend to diverge from their path leading to smaller or larger pixel values compared to the nerve fibres in the center plane. This divergence could be associated with demyelination since it is disorder along the z-axis and our method only takes into account disorder in the x and y axes.

The fact that both edges of a single nerve fibre follow the same random walk path could be an over simplification of the trajectory of real nerve fibres. It could be observed in the microscope images that it is not always the case where both edges of a nerve fibre follow a mirrored path in the horizontal direction. In fact in some cases, they could follow almost identically opposite paths. This could also be an effect caused by demyelination.

Our simulations also do not simulate Nodes of Ranvier. In the CARS images, Nodes of Ranvier would appear as a small region of black pixels between myelinated axons. Although they are small in size compared to the axon lengths and diameters, the CNN must have picked up on the occurrence of the Nodes of Ranvier. It would be interesting to study these occurrences across the different labels, and it could certainly assist in further developing existing demyelination models.

In the simulations we performed, all the nerve fibres had the same expected degree of disorder. What if we had nerve fibres with different degrees of disorder in an image, then how would that affect the decision making of the CNN? Perhaps the CNN could have a learnt to quantify demyelination based on the overall disorder of the region. This could be modelled by applying a probability distribution for the degree of disorder.

The modelling performed in this thesis simulates nerve fibres with the same myelin thickness and axon diameter in a single image. In reality, these variables are stochastic in nature and should vary after each random walk step according to a probability distribution in order to generate heterogeneous regions across the synthetic spinal cord. This is possible to model using a moving average technique. The moving average would involve sampling several values of the myelin thickness m from a defined probability distribution. The myelin thickness of the nerve fibre before the step would, for example, be the average of the first 5 sampled values. The averaging window would shift by one for every step taken. The moving average would ensure that demyelination varies with a controllable degree of smoothness along the nerve fibres, which could be a better representation for the heterogeneous nerve fibres in the real images.

4.3 Future Work

Future work could include applying the collinearity analysis on the synthetic images. With the simulated images, we know the degree of disorder as well as the structural properties of the nerve fibres, such as the myelin thickness and axon diameter. One could also associate degrees of disorder to different labels by cross-validating the *CCP* values of the synthetic images. This would solidify the relationship between the degree of disorder and the correlation of neighbouring domains.

There also exists a dataset of the images with a transverse view of the nerve fibres. This dataset could be used in cross-validation with the longitudinal images while training the CNN to have a better understanding of the structural features of the nerve fibres. In this thesis, we have not investigated how demyelination damages nerve fibres in the transverse side. We might be able to obtain a better measure of axon diameter and myelin thickness, a knowledge that could be used in our synthesized longitudinal images. Also, there is potential in synthesizing the transverse images and feeding them into the CNN with the synthesized longitudinal images,

We plan to incorporate the simulation parameters into existing models that describe action potential propagation with different amounts of myelin. In a recent publication, a cable equation model based on Green's function was developed to study the effects of demyelination on axonal transmission of action potentials between two internodes [20]. Internodes are two myelinated regions separated by a Node of Ranvier. The internode where the action potential is being propagated from is the antidromic internode while the internode receiving the action potential is the orthodromic internode. The model has parameters that control the amount of myelin in a given internode of a nerve fibre. Three different scenarios were investigated. The first case is a myelinated antidromic internode and a demyelinated orthodromic internode. The second case simulates a demyelinated antidromic internode and myelinated orthodromic internode. Finally, the last case simulates demyelination for both internodes .

The results of this model show different dynamical properties for each case of electrical transmission. The ratio of myelin thickness between the orthodromic and antidromic is used to measure the difference in myelin content. Since we can control the myelin thickness of the nerve fibres in our simulation, we can cross-validate the results from our simulations and the results of the Green's function model. This would allow us to bridge the gap between the physics of action potential propagation and the clinical implications of MS. Another question worth pursuing is whether the tortuosity feature can be implemented in the biophysical models, perhaps as an increase in length of different internodes or extra randomness of axon diameter and/or myelin thickness. All of these parameters can affect the Green's function propagation.

It remains to be seen in future work how tortuosity leads to propagation failure. The cause of faliure could be due to mechanical effects of the environment surrounding the nerve fibres such as the cytoskeleton, supporting glial cells, and the surface tension from the cerebrospinal fluid. The electrical influence that neighbouring nerve fibres could have on each other may contribute to the propagation of action potentials. At low tortuosity, nerve fibres are close together which could assist propagation while at high tortuosity, fibres are far apart which might have a negative impact on propagation. Finally, tortuosity may cause changes in the extra cellular concentrations of ions that are essential in initiating action potentials. These points remain unresolved and further research on them could provide more insights on the biophysical effects of demyelination.

In certain neurological diseases such as MS, an axon may have sections surrounded by myelin while some regions are left uncovered. This is due to the myelin pulling back from its target despite the axon reaching the target. In this case, we are left with a nerve fibre where regions are myelinated and would be evaluated as having a healthy g-ratio by our CNN-Simulation joint approach, thus considered to be a healthy fibre, where in reality, the remaining unmyelinated section of the fibre would be evaluated as unhealthy. This is another variant of demyelination that could be explored in future studies.

Finally, Magnetic Resonance Imaging (MRI) and Diffusion Tensor Imaging (DTI) can make use of this type of analysis. The degree of disorder and g-ratio can be measured from patients' images and clinical decisions could be inferred using our technique. Since the CNN was trained on the CARS image and may not recognize the MRI or DTI images, a new CNN could be trained on a patient image dataset and a similar study could be performed. Past research has shown that CNNs are capable of performing classification and segmentation of MRI images in hopes to identify demyelination [78–80], but do not attempt to connect the biophysics of the disease to the clinical information shown in the patient images. The demyelination assessment of MRI [81, 82] and DTI [83–85] can give rise to new simulation parameters and thus a clearer view of the structural and functional consequences of demyelination.

Bibliography

[1] Alasdair Coles. Multiple sclerosis: the bare essentials. *Practical Neurology*, 9(2):118–126, 2009.

[2] Paul van der Valk and Sandra Amor. Preactive lesions in multiple sclerosis. *Current Opinion in Neurology*, 22(3):207–213, 2009.

[3] Massimo Filippi, Wolfgang Brück, Declan Chard, Franz Fazekas, Jeroen JG Geurts, Christian Enzinger, Simon Hametner, Tanja Kuhlmann, Paolo Preziosa, Àlex Rovira, et al. Association between pathological and MRI findings in multiple sclerosis. *The Lancet Neurology*, 18(2):198–210, 2019.

[4] Nazem Ghasemi, Shahnaz Razavi, and Elham Nikzad. Multiple sclerosis: pathogenesis, symptoms, diagnoses and cell-based therapy. *Cell Journal*, 19(1):1, 2017.

[5] George A Schumacher, Gilbert Beebe, Robert F Kibler, Leonard T Kurland, John F Kurtzke, Fletcher McDowell, Benedict Nagler, William A Sibley, Wallace W Tourtellotte, and Thomas L Willmon. Problems of experimental trials of therapy in multiple sclerosis: report by the panel on the evaluation of experimental trials of therapy in multiple sclerosis. *Annals of the New York Academy of Sciences*, 122(1):552–568, 1965.

[6] Charles M Poser, Donald W Paty, Labe Scheinberg, W Ian McDonald, Floyd A Davis, George C Ebers, Kenneth P Johnson, William A Sibley, Donald H Silberberg, and Wallace W Tourtellotte. New diagnostic criteria for multiple sclerosis: guidelines for research protocols. *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society*, 13(3):227–231, 1983.

[7] Steve Bégin, Erik Bélanger, Sophie Laffray, Benoît Aubé, Émilie Chamma, Jonathan Bélisle, Steve Lacroix, Yves De Koninck, and Daniel Côté. Local as-

sessment of myelin health in a multiple sclerosis mouse model using a 2d fourier transform approach. *Biomedical optics express*, 4(10):2003–2014, 2013.

[8] Sheng Weng, Xiaoyun Xu, Jiasong Li, and Stephen TC Wong. Combining deep learning and coherent anti-stokes Raman scattering imaging for automated differential diagnosis of lung cancer. *Journal of Biomedical Optics*, 22(10):106017, 2017.

[9] Masayasu Toratani, Masamitsu Konno, Ayumu Asai, Jun Koseki, Koichi Kawamoto, Keisuke Tamari, Zhihao Li, Daisuke Sakai, Toshihiro Kudo, Taroh Satoh, et al. A convolutional neural network uses microscopic images to differentiate between mouse and human cell lines and their radioresistant clones. *Cancer Research*, 78(23):6703–6707, 2018.

[10] Ronald Wihal Oei, Guanqun Hou, Fuhai Liu, Jin Zhong, Jiewen Zhang, Zhaoyi An, Luping Xu, and Yujiu Yang. Convolutional neural network for cell classification using microscope images of intracellular actin networks. *PloS one*, 14(3):e0213626, 2019.

[11] David Chambers, Christopher Huang, and Gareth Matthews. *Basic Physiology for Anaesthetists*. Cambridge University Press, 2 edition, 2019.

[12] Peter Dayan, Laurence F Abbott, et al. Theoretical neuroscience: computational and mathematical modeling of neural systems. *Journal of Cognitive Neuroscience*, 15(1):154–155, 2003.

[13] Louis Édouard Lapicque. Louis lapicque. *J. physiol*, 9:620–635, 1907.

[14] WAH Rushton. A theory of the effects of fibre size in medullated nerve. *The Journal of Physiology*, 115(1):101, 1951.

[15] L Goldman and James S Albus. Computation of impulse conduction in myelinated fibers; theoretical basis of the velocity-diameter relation. *Biophysical Journal*, 8(5):596–607, 1968.

[16] Sid Deutsch. The maximization of nerve conduction velocity. *IEEE Transactions on Systems Science and Cybernetics*, 5(1):86–91, 1969.

[17] RICHARD S Smith and ZOLY J Koles. Myelinated nerve fibers: computed effect of myelin thickness on conduction velocity. *American Journal of Physiology-Legacy Content*, 219(5):1256–1258, 1970.

[18] PL Williams and CP Wendell-Smith. Some additional parametric variations between peripheral nerve fibre populations. *Journal of Anatomy*, 109(Pt 3):505, 1971.

[19] A Bischoff. Microscopic anatomy of myelinated nerve fibers. *Peripheral neuropathy*, 1:104–130, 1975.

[20] Richard Naud and André Longtin. Linking demyelination to compound action potential dispersion with a spike-diffuse-spike approach. *The Journal of Mathematical Neuroscience*, 9(1):1–24, 2019.

[21] Conor L Evans and X Sunney Xie. Coherent anti-stokes Raman scattering microscopy: Chemical imaging for biology and medicine. *Annu. Rev. Anal. Chem.*, 1:883–909, 2008.

[22] PD Maker and RW Terhune. Study of optical effects due to an induced polarization third order in the electric field strength. *Physical Review*, 137(3A):A801, 1965.

[23] Haifeng Wang, Yan Fu, Phyllis Zickmund, Riyi Shi, and Ji-Xin Cheng. Coherent anti-stokes Raman scattering imaging of axonal myelin in live spinal tissues. *Biophysical Journal*, 89(1):581–591, 2005.

[24] Steve Bégin, Erik Bélanger, Sophie Laffray, Réal Vallée, and Daniel Côté. In vivo optical monitoring of tissue pathologies and diseases with vibrational contrast, 2009.

[25] John Paul Pezacki, Jessie A Blake, Dana C Danielson, David C Kennedy, Rodney K Lyn, and Ragunath Singaravelu. Chemical contrast for imaging living systems: molecular vibrations drive cars microscopy. *Nature chemical biology*, 7(3):137–145, 2011.

[26] Eilhard Mix, Hans Meyer-Rienecker, Hans-Peter Hartung, and Uwe K Zettl. Animal models of multiple sclerosis—potentials and limitations. *Progress in neurobiology*, 92(3):386–404, 2010.

[27] A Bert, Bruno Gran, and Robert Weissert. Eae: imperfect but useful models of multiple sclerosis. *Trends in molecular medicine*, 17(3):119–125, 2011.

[28] Ara Ghazaryan, Halley F Tsai, Gor Hayrapetyan, Wei-Liang Chen, Yang-Fang Chen, Myung-Yung Jeong, Chang-Seok Kim, Shean-Jen Chen, and Chen-Yuan

Dong. Analysis of collagen fiber domain organization by fourier second harmonic generation microscopy. *Journal of biomedical optics*, 18(3):031105, 2012.

[29] Paolo Matteini, Fulvio Ratto, Francesca Rossi, Riccardo Cicchi, Chiara Stringari, Dimitrios Kapsokalyvas, Francesco S Pavone, and Roberto Pini. Photothermally-induced disordered patterns of corneal collagen revealed by SHG imaging. *Optics Express*, 17(6):4868–4878, 2009.

[30] Raghu Ambekar Ramachandra Rao, Monal R Mehta, and Kimani C Toussaint. Fourier transform-second-harmonic generation imaging of biological tissues. *Optics express*, 17(17):14534–14542, 2009.

[31] Riccardo Cicchi, Nadine Vogler, Dimitrios Kapsokalyvas, Benjamin Dietzek, Jürgen Popp, and Francesco Saverio Pavone. From molecular structure to tissue architecture: collagen organization probed by SHG microscopy. *Journal of Biophotonics*, 6(2):129–142, 2013.

[32] Tianmei Guo, Jiwen Dong, Henjian Li, and Yunxing Gao. Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724.

[33] Pierre Bovet and Simon Benhamou. Spatial analysis of animals' movements using a correlated random walk model. *Journal of theoretical biology*, 131(4):419–433, 1988.

[34] Frederic Bartumeus, M G E da Luz, Gandhimohan M Viswanathan, and Jordi Catalan. Animal search strategies: a quantitative random-walk analysis. *Ecology*, 86(11):3078–3087, 2005.

[35] Patric Hagmann, J-P Thiran, Lisa Jonasson, Pierre Vandergheynst, Stephanie Clarke, Philippe Maeder, and Reto Meuli. DTI mapping of human brain connectivity: statistical fibre tracking and virtual dissection. *Neuroimage*, 19(3):545–554, 2003.

[36] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[37] P Sibi, S Allwyn Jones, and P Siddarth. Analysis of different activation functions using back propagation neural networks. *Journal of theoretical and applied information technology*, 47(3):1264–1268, 2013.

[38] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[39] Robert Schalkoff. *Pattern recognition: statistical, structural and neural approaches*. Wiley, New York Chichester, 1992.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[42] C Szegedy, W Liu, Y Jia, P Sermanet, S Reed, D Anguelov, and A Rabinovich. Going deeper with convolutions in: Proceedings of the ieee conference on computer vision and pattern recognition; 2015. *Google Scholar*, pages 1–9.

[43] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[45] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[46] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[47] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495, 2017.

[48] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[49] Juyang Weng, Narendra Ahuja, and Thomas S Huang. Cresceptron: a self-organizing neural network which grows adaptively. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 576–581. IEEE, 1992.

[50] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[51] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[52] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[53] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

[54] M Hossin, MN Sulaiman, A Mustapha, N Mustapha, and RW Rahmat. A hybrid evaluation metric for optimizing classifier. In *2011 3rd Conference on Data Mining and Optimization (DMO)*, pages 165–170. IEEE, 2011.

[55] Romesh Ranawana and Vasile Palade. Optimized precision-a new measure for classifier performance evaluation. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2254–2261. IEEE, 2006.

[56] Jin Huang and Charles X Ling. Constructing new and better evaluation measures for machine learning. In *IJCAI*, pages 859–864, 2007.

[57] Alain Rakotomamonjy. Optimizing area under ROC curve with svms. In *ROCAI*, pages 71–80, 2004.

[58] Loris Bazzani, Alessandra Bergamo, Dragomir Anguelov, and Lorenzo Torresani. Self-taught object localization with deep networks. In *2016 IEEE winter conference on applications of computer vision (WACV)*, pages 1–9. IEEE, 2016.

[59] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 685–694, 2015.

[60] Michael Buckland and Fredric Gey. The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19, 1994.

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[62] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):189–203, 2016.

[63] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[64] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[65] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[66] Derek K Jones. Studying connections in the living human brain with diffusion MRI. *cortex*, 44(8):936–952, 2008.

[67] Hellen Altendorf and Dominique Jeulin. Random-walk-based stochastic modeling of three-dimensional fiber systems. *Physical Review E*, 83(4):041804, 2011.

[68] Fabian Heinemann, Gerald Birk, and Birgit Stierstorfer. Deep learning enables pathologist-like scoring of nash models. *Scientific reports*, 9(1):1–10, 2019.

[69] Yan Fu, Haifeng Wang, Terry B Huff, Riyi Shi, and Ji-Xin Cheng. Coherent anti-stokes Raman scattering imaging of myelin degradation reveals a calcium-dependent pathway in lyso-PtdCho-induced demyelination. *Journal of neuroscience research*, 85(13):2870–2881, 2007.

[70] Jaime Imitola, Stine Rasmussen, Yingru Liu, Tanuja Chitnis, Samia Khoury, Daniel Côté, X Sunney Xie, Charles P Lin, and Richard L Sidman. Multimodal coherent anti-stokes Raman scattering microscopy reveals microglia-associated myelin and axonal dysfunction in multiple sclerosis-like lesions in mice. *Journal of Biomedical Optics*, 16(2):021109, 2011.

[71] Yan Fu, Ji-Xin Cheng, Terra J Frederick, Gwendolyn E Goings, Stephen D Miller, and Terry B Huff. Paranodal myelin retraction in relapsing experimental autoimmune encephalomyelitis visualized by coherent anti-stokes Raman scattering microscopy. *Journal of Biomedical Optics*, 16(10):106006, 2011.

[72] Yunzhou Shi, Riyi Shi, Ji-Xin Cheng, Delong Zhang, Terry B Huff, Xiaofei Wang, and Xiao-Ming Xu. Longitudinal in vivo coherent anti-stokes Raman scattering imaging of demyelination and remyelination in injured spinal cord. *Journal of biomedical optics*, 16(10):106012, 2011.

[73] Christian W Freudiger, Rolf Pfannl, Daniel A Orringer, Brian G Saar, Minbiao Ji, Qing Zeng, Linda Ottoboni, Wei Ying, Christian Waeber, John R Sims, et al. Multicolored stain-free histopathology with coherent Raman imaging. *Laboratory investigation*, 92(10):1492–1502, 2012.

[74] Taylor Chomiak and Bin Hu. What is the optimal value of the g-ratio for myelinated fibers in the rat cns? a theoretical approach. *PloS one*, 4(11):e7754, 2009.

[75] Claes Hildebrand and Robert Hahn. Relation between myelin sheath thickness and axon size in spinal cord white matter of some vertebrate species. *Journal of the neurological sciences*, 38(3):421–434, 1978.

[76] D Graf von Keyserlingk and U Schramm. Diameter of axons and thickness of myelin sheaths of the pyramidal tract fibres in the adult human medullary pyramid. *Anatomischer Anzeiger*, 157(2):97–111, 1984.

[77] WJC Verhaart. On thick and thin fibers in the pyramidal tract. *Acta Psychiatrica Scandinavica*, 22(3-4):271–281, 1947.

[78] Melika Maleki, M Teshnehlab, and M Nabavi. Diagnosis of multiple sclerosis (ms) using convolutional neural network (cnn) from mris. *Global Journal of Medicinal Plant Research*, 1(1):50–54, 2012.

[79] Wen Wei, Emilie Poirion, Benedetta Bodini, Stanley Durrleman, Nicholas Ayache, Bruno Stankoff, and Olivier Colliot. Predicting pet-derived demyelination from multimodal mri using sketcher-refiner adversarial training for multiple sclerosis. *Medical image analysis*, 58:101546, 2019.

[80] Alexandre Fenneteau, Pascal Bourdon, David Helbert, Christine Fernandez-Maloigne, Christophe Habas, and Remy Guillevin. Learning a cnn on multiple sclerosis lesion segmentation with self-supervision. *Electronic Imaging*, 2020(17):3–1, 2020.

[81] M Sailer, JI O'riordan, AJ Thompson, DP Kingsley, DG MacManus, WI McDonald, and DH Miller. Quantitative MRI in patients with clinically isolated syndromes suggestive of demyelination. *Neurology*, 52(3):599–599, 1999.

[82] Leonard H Verhey, Helen M Branson, Manohar M Shroff, David JA Callen, John G Sled, Sridar Narayanan, A Dessa Sadovnick, Amit Bar-Or, Douglas L Arnold, Ruth Ann Marrie, et al. MRI parameters for prediction of multiple sclerosis diagnosis in children with acute cns demyelination: a prospective national cohort study. *The Lancet Neurology*, 10(12):1065–1073, 2011.

[83] Sheng-Kwei Song, Jun Yoshino, Tuan Q Le, Shiow-Jiuan Lin, Shu-Wei Sun, Anne H Cross, and Regina C Armstrong. Demyelination increases radial diffusivity in corpus callosum of mouse brain. *Neuroimage*, 26(1):132–140, 2005.

[84] Emilia Sbardella, Francesca Tona, Nikolaos Petsas, and Patrizia Pantano. DTI measurements in multiple sclerosis: evaluation of brain damage and clinical implications. *Multiple sclerosis international*, 2013, 2013.

[85] Eric C Klawiter, Robert E Schmidt, Kathryn Trinkaus, Hsiao-Fang Liang, Matthew D Budde, Robert T Naismith, Sheng-Kwei Song, Anne H Cross, and Tammie L Benzinger. Radial diffusivity predicts demyelination in ex vivo multiple sclerosis spinal cords. *Neuroimage*, 55(4):1454–1460, 2011.