Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 11, No. 3, September 2023, pp. 883~895
ISSN: 2089-3272, DOI: 10.52549/ijeei.v11i3.5009

Optimizing U-Net Architecture with Feed-Forward Neural Networks for Precise Cobb Angle Prediction in Scoliosis Diagnosis

Mohamad Iqmal Jamaludin[1], Teddy Surya Gunawan[2], Rajandra Kumar Karupiah[3], Suriza Ahmad Zabidi[4], Mira Kartiwi[5], Zamzuri Zakaria[6]

[1,2,4] Department of Electrical and Computer Engineering, International Islamic University Malaysia, Malaysia
[3,6] Department of Orthopaedics, Traumatology & Rehabilitation, International Islamic University Malaysia, Malaysia
[5] Department of Information Systems, International Islamic University Malaysia, Malaysia

Article Info

Article history:

Received Aug 8, 2023
Revised Sep 27, 2023
Accepted Sep 28, 2023

Keywords:
Artificial Intelligence
Deep Learning
U-Net Architecture
Scoliosis Diagnosis
Cobb Angle Prediction
PyTorch Framework

ABSTRACT

In the burgeoning field of Artificial Intelligence (AI) and its notable subsets, such as Deep Learning (DL), there is evidence of its transformative impact in assisting clinicians, particularly in diagnosing scoliosis. AI is unrivaled for its speed and precision in analyzing medical images, including X-rays and computed tomography (CT) scans. However, the path does not lack obstacles. Biases, unanticipated outcomes, and false positive and negative predictions present significant challenges. Our research employed three complex experimental sets, each focusing on adapting the U-Net architecture. Through a nuanced combination of feed-forward neural network (FFNN) configurations and hyperparameters, we endeavored to determine the most effective nonlinear regression model configuration for predicting the Cobb angle. This was done with the dual purpose of reducing AI training time without sacrificing predictive accuracy. Utilizing the capabilities of the PyTorch framework, we meticulously crafted and refined the deep learning models for each of the three experiments, focusing on an FFFN dropout rate of p=0.45. The Root Mean Square Error (RMSE), the number of epochs, and the number of nodes spanning three hidden layers in each FFFN were utilized as crucial performance metrics while a base learning rate of 0.001 was maintained. Notably, during the optimization phase, one of the experiments incorporated a learning rate scheduler to protect against potential pitfalls such as local minima and saddle points. A judiciously incorporated Early Stopping technique, triggered between the patience range of 5-10 epochs, ensured model stability as the Mean Squared Error (MSE) plateau loss approached approximately 1. Consequently, the model converged between 50 and 82 epochs. We hypothesize that our proposed architecture holds promise for future refinements, conditioned on assiduous experimentation with an array of medical deep learning paradigms.

Corresponding Author:
Teddy Surya Gunawan
Department of Electrical and Computer Engineering,
International Islamic University Malaysia, Malaysia.
Email: tsgunawan@iium.edu.my

1. INTRODUCTION

Scoliosis is characterized by vertebral rotation and lateral curvature that deviates from the typical vertical alignment of the spine [1]. On a posterior-anterior radiograph, a definitive diagnosis of scoliosis requires a spinal angulation of at least 10° accompanied by vertebral rotation. The diverse causes of scoliosis include congenital, neuromuscular, syndrome-related, idiopathic, and secondary conditions. Clinicians predominantly diagnose idiopathic scoliosis as the most prevalent form. In severe cases, scoliosis can cause

life-threatening complications. For instance, an excessive curvature of the spine may obstruct the thoracic cavity, limiting lung function.

The introduction of Artificial Intelligence (AI), specifically its subset Machine Learning (ML), has the potential to revolutionize medical diagnostics. ML enables computers to refine predictions based on previous discrepancies, utilizing vast biological datasets to improve healthcare [2]. Essentially, ML can enhance a machine's task execution prowess with suitable datasets. AI can guarantee that vital clinical indicators are not overlooked by assisting in interpreting X-rays, MRIs, and CT scans. Nevertheless, it is essential to recognize that AI's role in medical diagnostics is supplementary. It aids clinicians, particularly radiologists, without replacing their knowledge. The final diagnosis is the exclusive domain of medical professionals.

In the pursuit of advancing and improving Cobb angle detection for scoliosis, researchers have extensively used AI techniques. These investigations demonstrate AI's adaptability and potential impact on the healthcare paradigm. Convolutional Neural Networks (CNN) were used in [3] to evaluate spinal alignment using moiré patterns extracted from photographs. These patterns are the result of the imaging sensor's limited resolution. They predicted the locations of 17 vertebral bodies in 10,788 images using the AlexNet CNN model. The mean absolute error (MAE) of the model is closely aligned with the analyses of the experts, based on a comparison between the model's outputs and their analyses. However, factors such as the clinician's ability to interpret images and the limitations of image acquisition and equipment affected the accuracy of the model.

The intraobserver and interobserver error of 3 to 10° inherent to traditional Cobb angle measurement techniques was emphasized in [4]. These patterns are the result of the imaging sensor's limited resolution. They predicted the locations of 17 vertebral bodies in 10,788 images using the AlexNet CNN model. The mean absolute error (MAE) of the model is closely aligned with the analyses of the experts, based on a comparison between the model's outputs and their analyses. However, factors such as the clinician's ability to interpret images and the limitations of image acquisition and equipment affected the accuracy of the model.

Scoliosis screening using portable ultrasound was implemented in [5], reducing patients' exposure to radiation-laden MRIs and X-rays. Inception v1 (GoogleNet) was trained on their dataset of 2,752 ultrasound images, with an additional 747 images set aside for testing. The model's consistent and reliable performance suggested the potential for ultrasound-based spinal detection, though expanding the dataset could further refine accuracy. Similarly, the Mask R-CNN model was deployed in [6] to measure the Cobb angle from chest X-rays. With a dataset of 248 X-ray images from lung cancer patients, they identified vertebral bodies and the respective endplates of each vertebral section. Two radiologists manually cross-verified the model's results, underscoring the potential of a computer-aided approach. However, training the CNN on a more extensive dataset, especially of standard spinal curve radiographs, could enhance outcomes.

An automated technique using anteroposterior (AP) X-ray images was proposed in [7]. Using U-Net, Dense U-Net, and Residual U-Net deep learning CNN models, they segmented vertebrae from 35 AP spinal X-ray images and estimated the Cobb angle. Residual U-Net exhibited the best segmentation results among the three, reflecting manual evaluations. The limitation of this method was its emphasis on Cobb angle calculation rather than the identification and classification of spinal deformities. A deep learning keypoint detection technology was developed by [8] that demonstrated high reliability in automated Cobb angle measurement, especially when the angle did not exceed 90°, offering the advantage of locating multiple curves in scoliosis cases simultaneously in a short period. However, the study acknowledged the need for verification in more cases in the future. A deep learning algorithm was introduced in [9] with a three-dimensional (3D) depth sensor, considering the influence of garments, which showed a high correlation with actual Cobb angles and offered a valuable alternative for scoliosis examination with underwear, addressing the psychological burden associated with naked body imaging in children and adolescents. On the other hand, a method for locating vertebral center points was proposed in [10] and measuring the Cobb angle based on deep learning, which exhibited good reliability compared to traditional manual measurement methods, indicating its effectiveness in quick scoliosis detection. However, the paper did not extensively discuss the limitations or potential areas of improvement for the proposed method.

Deep transfer learning was utilized in [11] to detect scoliosis and spondylolisthesis, achieving high diagnostic accuracy without requiring any measurements, thereby streamlining the diagnostic process. However, the study did not delve into the potential limitations or challenges of the approach. A convolutional neural network was introduced in [12] for automated Cobb angle measurement in adolescent idiopathic scoliosis, demonstrating up to 93.6% accuracy and excellent reliability compared to clinicians' measurements. This method holds promise for clinical application, although its adaptability to varied clinical settings remains to be explored. SpineHRNet was developed in [13], a neural network capable of predicting endplate landmarks and computing Cobb angles from smartphone-acquired scoliosis radiograph images, showcasing its potential for telemedicine and auto-diagnosis. However, the study's applicability across different smartphone devices

and image qualities warrants further investigation. Collectively, these studies underscore the potential of deep learning in enhancing scoliosis diagnosis, while also highlighting areas necessitating further research.

While AI's capabilities are vast, obstacles remain. Its output quality is inextricably tied to the training data, making it susceptible to biases introduced by non-representative data. Overfitting, in which an excessively complex model memorizes rather than learns, can inhibit performance on new data [14]. In addition, AI predictions can manifest as false positives and false negatives. These inaccuracies, which result from data biases, overfitting, or obsolete models, require cautious interpretation. Therefore, this paper aims to design and evaluate a deep learning architecture for early scoliosis detection, balancing prediction accuracy with efficient training time, by developing a regression model to predict Cobb angle values from medical images. We will assess the performance of our model using mainly regression metrics. This paper is organized as follows: Section 2 delves into the fundamentals of scoliosis, the Cobb angle, and the intricacies of deep learning frameworks and hyperparameters. Section 3 introduces and outlines the proposed U-Net architecture. Results and discussions are presented in Section 4, and the paper concludes with Section 5.

2. SCOLIOSIS AND DEEP LEARNING

This section provides a comprehensive overview of the foundational concepts relevant to our investigation. Following an explanation of the nature and implications of scoliosis, we examine the Cobb angle and its significance in depth. In addition, we explore the nuances of deep learning frameworks, elucidating their applicability and significance. In addition to shedding light on the complexities of hyperparameters, this section lays the groundwork for subsequent discussions on model architecture and evaluations.

The human spine, depicted in Figure 1, is a crucial anatomical structure responsible for bearing the body's weight and safeguarding the spinal cord and its encompassed nerves. The human spine comprises 33 vertebrae, which are categorized into five distinct regions: the cervical (C1–C7), thoracic (T1–T12), lumbar (L1–L5), sacrum (S1-S5), and coccyx (Co1–Co4). The top 24 vertebrae are individual and movable, giving the spine essential flexibility. Conversely, the remaining 9 vertebrae are static; the sacral vertebrae merge to constitute the sacrum, while the coccygeal vertebrae typically unite post-adolescence to form the coccyx [15].

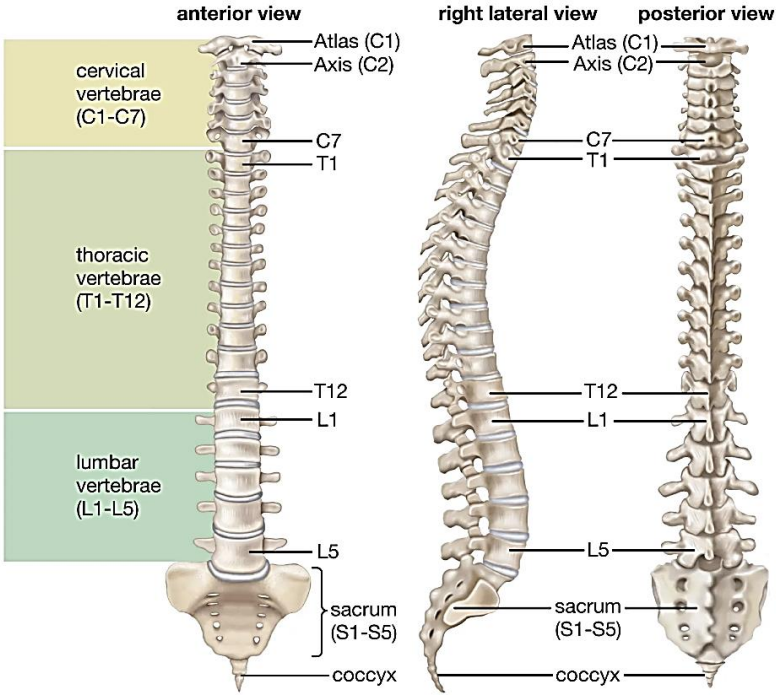
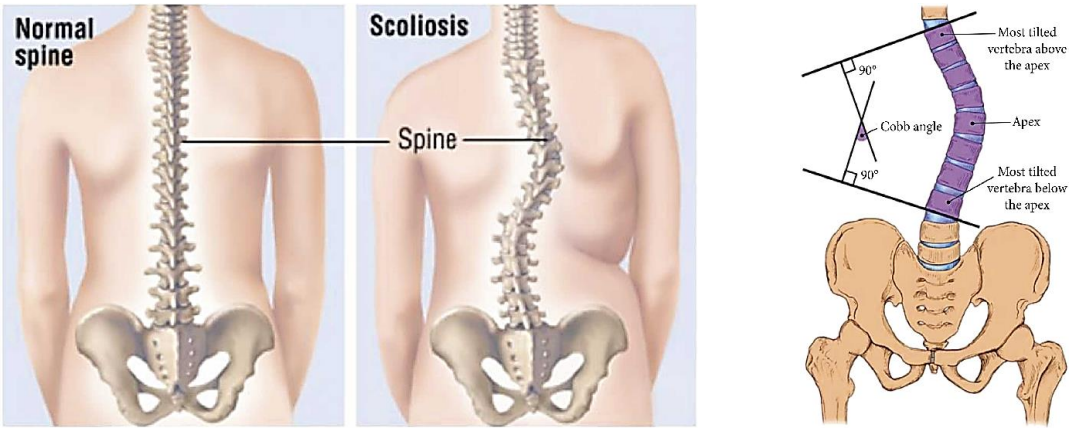


Figure 1. Vertebral Column [15]

2.1. Scoliosis and Cobb Angle

Scoliosis, derived from the Greek word for "crookedness," is characterized by a deviation from the normal vertical alignment of the spine [16]. This deviation manifests as a lateral curvature accompanied by a rotation of the vertebrae within that curve [1]. When viewed from the front or the back, the spine should ideally appear straight and centered over the pelvis. In individuals with scoliosis, however, the spine displays a pronounced lateral curve, either to the right or left. Scoliosis is indicated by a curvature greater than 10°, which gives the spine a C- or S-shaped appearance, as shown in Figure 2(a). There are numerous causes for the onset of scoliosis, including neurological disorders, muscular abnormalities, and other syndromes.



(a) (b)

Figure 2. Scoliosis and Cobb angle [7]: (a) Normal spine and scoliosis, (b) Cobb angle measurement

Clinicians use imaging tests such as X-rays, CT scans, and MRIs to quantify the degree of spinal curvature when diagnosing and assessing scoliosis. Cobb's angle, a technique developed by the American orthopedic surgeon John Robert Cobb, is the most common method for measuring this angle. In 1966, the Scoliosis Research Society (SRS) acknowledged this method as the standard for quantifying scoliosis [7]. As shown in Figure 2(b), the procedure involves measuring the angle between tangents drawn to the upper and lower endplates of the vertebrae with the greatest degree of inclination at the curve's extremities. Table 1 summarizes the classifications based on the Cobb angle measurements.

Table 1. Cobb Angle Classification

Cobb angle	Classification
0° - 10°	Spinal curve
10° - 20°	Mild scoliosis
20° - 40°	Moderate scoliosis
> 40°	Severe scoliosis

2.2. Deep Learning Frameworks

The second generation of Google's open-source deep learning framework, TensorFlow, is an evolution of its predecessor, DistBelief. It embodies the tech titan's core principles for artificial intelligence systems [17]. TensorFlow's adaptability across diverse platforms, ranging from handheld devices such as smartphones and tablets to vast distributed systems involving hundreds of computers and various computational devices such as GPU cards, is one of its defining characteristics. It maintains its position as the most popular deep learning framework due to several distinguishing characteristics, such as unrivaled performance, true portability, and support for multiple languages.

PyTorch, a Python-based package, possesses two revolutionary features. It facilitates GPU-assisted tensor computations, making it a viable alternative to NumPy in many situations. Second, it features a dynamic computation graph that gives developers the freedom to design complex neural networks using the simplicity of Python. In stark contrast, TensorFlow employs a static computation model in which, once a neural network is created, it must be rebuilt from scratch for any modifications [17]. PyTorch's implementation of Reverse-mode auto-differentiation enables programmers to modify network behavior on the fly without cumbersome overhauls. This flexibility reinforces PyTorch's status as one of the most rapidly evolving deep learning tools.

PyTorch's dynamic computation graph is a boon for researchers and developers interested in iterative and experimental modeling. This adaptability permits network modifications in real time and can result in quicker prototyping and more intuitive model modifications. In addition, the seamless transition between CPU and GPU computations allows developers to optimize performance based on available hardware resources. Lastly, integrating Python, a widely adopted programming language in the scientific and machine learning communities, ensures an easier learning curve and immediate familiarity for many users. Given these benefits and PyTorch's ecosystem's rapid development, it stands out as the best option for this research.

2.3. Deep Learning Hyperparameters

Neural networks have demonstrated their efficacy in various applications; however, their propensity for overfitting renders them unsuitable for some modeling tasks. Training a network, which begins with a random state and requires voluminous data, can be viewed as a brute-force technique. The deployment of these networks necessitates meticulous model architecture, algorithmic design, and hyperparameter tuning.

Typically, hyperparameters are selected based on prior experience and network training knowledge. Nonetheless, the empirical nature of these decisions, devoid of solid logical justification, can result in suboptimal outcomes, producing merely adequate hyperparameters instead of the optimal set.

Before training a machine learning model, essential hyperparameters must be set. They can affect the model's architecture, such as the number of hidden layers and activation functions, and the efficiency and precision of training, such as the learning rate of stochastic gradient descent (SGD), batch size, and optimizer selection [18].

Learning Rate (LR) determines the magnitude of model weight adjustments made during Stochastic Gradient Descent (SGD) [19]. Typically, it is manually adjusted throughout training to achieve optimal results. A variable LR during training, also known as LR scheduling or LR decay, is sometimes more effective than a fixed LR. This adaptive method employs learning algorithms to adjust the LR based on model performance or specific criteria [20, 21].

The learning rate profoundly impacts the training of deep learning models. It governs the steps to optimize the loss function with the model's weights. The optimization algorithm adjusts weights iteratively according to the gradient of this loss function. Larger learning rates result in larger adjustment steps, which may cause optimal weight values to oscillate or overshoot. In contrast, extremely low rates may trap the model in suboptimal solutions, known as local minima. Consequently, striking the proper balance is crucial.

By default, frameworks like PyTorch utilize a fixed LR. However, finding the optimal rate can be difficult. Convergence may be hindered if the convergence rate is too high, particularly at the end of training. A common strategy is to begin with a larger LR and then reduce it as the model approaches an accuracy plateau [22]. StepLR, MultiStepLR, OneCycleLR, CosineAnnealingLR, and CosineAnnealingWarmRestartsLR are among the schedulers for dynamic LR adjustments offered by PyTorch.

Regularization mitigates overfitting, a common concern in large, intricate neural networks. By adding a penalty term or "noise" to the loss value, models can be prevented from merely memorizing training data, fostering better generalization. The regularization term's addition is mathematically represented as:

$$Modified\ loss\ value = loss\ function + regularization\ form \quad (1)$$

The concept is to modify the computed loss value by adding noise to the original loss value. It is crucial because overfitting is a problem in which the model cannot generalize the training data and instead begins to "memorize" it. This can be detected as the training loss decreases while the validation loss increases during training. By introducing some tolerable noise, it is hoped that the model is not overly dependent on training data alone for accurate prediction.

Two commonly used regularization techniques are $\mathcal{L}1$ and $\mathcal{L}2$. The $\mathcal{L}1$ method, called Lasso Regression, penalizes the sum of absolute weight values and is shown in Eq. (2).

$$w^* = \sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_i j W_j\right)^2 + \lambda \sum_{j=0}^{M} |W_j| \quad (2)$$

In contrast, L2 or Ridge Regression penalizes the square sum of weights and is shown in Eq. (3).

$$w^* = \sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_i j W_j\right)^2 + \lambda \sum_{j=0}^{M} W_j^2 \quad (3)$$

where λ is the regularization hyperparameter determining the amount of noise added to the weight. Larger λ simplifies the structure of deep learning because most weights are nearly equal to zero, resulting in less information (weight update) being passed from one layer of the neural network to the next. Lesseningλ diminishes the $\mathcal{L}1$ or $\mathcal{L}2$ regularization effect.

Dropout is a strategy for randomly selecting neurons not used during training with a predetermined probability, making the network less sensitive to the precise weights of neurons [23]. In other words, some nodes in each layer will be nullified (weights will be set to zero) based on a predetermined probability. It is then said that the neural network becomes condensed or less complex, which reduces overfitting. Estimates indicate that 20 to 50 percent of neuron deactivation may occur during each step of the weight update procedure. A high dropout rate will oversimplify the model, whereas a low value will have little effect. Regularization is integral to deep learning, ensuring accurate and generalizable models. By understanding and effectively utilizing these techniques, researchers and practitioners can harness the full power of neural networks.

3. PROPOSED DEEP LEARNING ARCHITECTURES

This study will exploit the potential of a hybrid deep learning algorithm incorporating both the Convolutional Neural Network (CNN) and Feed Forward Neural Network (FNN) to establish a robust execution system (FFNN). Figure 3 depicts the comprehensive flow of the proposed algorithm.

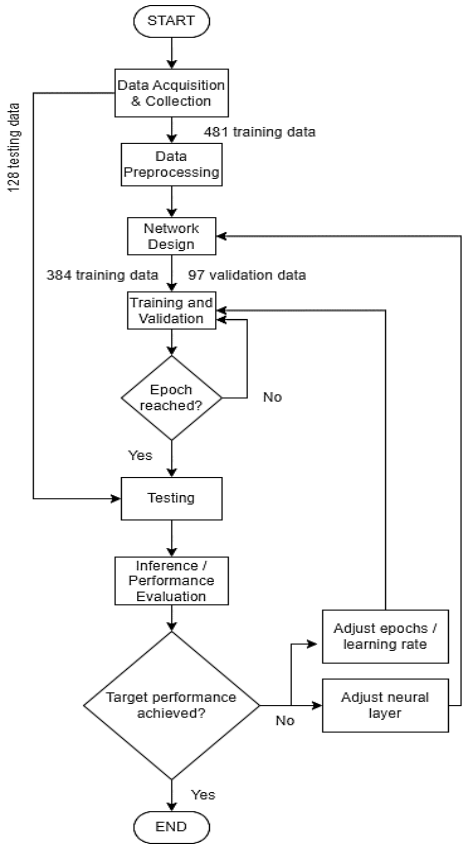


Figure 3. Summary of Research Methodology

This algorithmic procedure is divided into six sequential steps to ensure clarity and progression:

- **Data Acquisition & Collection**: This preliminary phase entails collecting the necessary data for training and validating the model. The data may be sourced from various datasets or platforms depending on the project's objectives.
- **Data Preprocessing**: In this phase, raw data is transformed into a format more conducive to analysis. This entails dealing with missing values, normalizing data, augmenting images (if applicable), and separating the dataset into training, validation, and test sets.
- **Network Design**: Here, the deep learning model's architecture is established. Regarding layers, nodes, and activation functions, the hybrid combination of CNN (to process spatial hierarchies in data, especially images) and FFNN (for generalized learning tasks) will be mapped out.
- **Training and Validation**: The designed network is trained using the previously prepared training dataset during this critical phase. The validation set iteratively evaluates the model to ensure that it does not overfit and that its learning applies to unseen data.
- **Testing**: The model is tested after achieving satisfactory performance metrics during validation. It is exposed to an unseen test dataset to evaluate the model's predictive abilities in real-world scenarios.
- **Performance Evaluation**: In this final phase, the model's efficacy is determined using the preferred evaluation metrics, which, depending on the problem statement, may include accuracy, precision, recall, F1-score, or any other pertinent metrics.

3.1. Dataset and Data Preprocessing

Deep learning algorithms thrive on large labeled datasets, essential for recognizing patterns and producing accurate predictions. This research utilized a dataset of grayscale x-ray images, each associated with one of three Cobb angles: Proximal Thoracic (PT), Mid Thoracic (MT), and Thoracolumbar (TL) (TL). The study utilized a dataset of 481 anterior-posterior spinal X-ray images provided by [24], all exhibiting varying degrees of scoliosis, provided by local clinicians. Focusing on 17 vertebrae of the thoracic and lumbar spine, 68 landmarks were manually annotated on the corners of each image. Based on their position in the original image, the scale for the landmarks was set to 0–1. The dataset was separated into a training/validation set (Trainset) of 431 images and a testing set (Testset) of 50 images, with no overlap of patients between the two sets. Using these subsets, the model was trained, validated, and tested to evaluate its performance in scoliosis

detection. In addition, some X-ray images were obtained from the IIUM Kuantan Hospital during algorithm development.

Effective deep learning requires meticulous data preprocessing, which can substantially improve model results. A custom dataset class was created to manage the necessary preprocessing for this research. The x-ray images, which were in various jpg dimensions and contained RGB channels, were initially resized. For compatibility with the U-Net architecture, these images were scaled to a standard (1, 572, 572) dimension, with 1 indicating a grayscale channel and 572×572 denoting the image's width and height. The subsequent steps involved normalizing each image's Z-score based on its standard deviation and mean. The next step includes transforming each image into a float32-type PyTorch tensor.

The training dataset was split into an 80:20 ratio, with 20 % designated for validation, which was essential for a preliminary objective evaluation of the model's performance. The 481-instance training dataset was subdivided into 384 instances for training and 97 instances for validation. The training subset was shuffled to promote unbiased learning and optimize the training procedure; this ensures that the model learns from various instances, thereby enhancing generalization. In contrast, neither the validation nor the 128-item testing subsets were mixed. The latter serves as a pristine reserve for the final evaluation of the model following training.

3.2. Deep Learning Architectures Design

The neural network under consideration is bifurcated into two primary components: the Convolutional Neural Network (CNN) and the Feed Forward Neural Network (FFNN). The CNN segment utilizes the U-Net architecture for biomedical image segmentation, as shown in Figure 4. Configured to accept an input of dimensions (1, 572, 572), which corresponds to a 1-channel image of size 572×572 pixels, it generates an output of dimensions (2, 388, 388). Nevertheless, the version utilized in this experiment is not a standard U-Net but an enhanced version. Dropout layers have been strategically incorporated after each downsampling and upsampling stage to improve model generalization and prevent overfitting. Concurrently, batch normalization layers are incorporated into the encoder subprocess of U-Net to ensure that inputs to subsequent convolutional layers remain normalized.

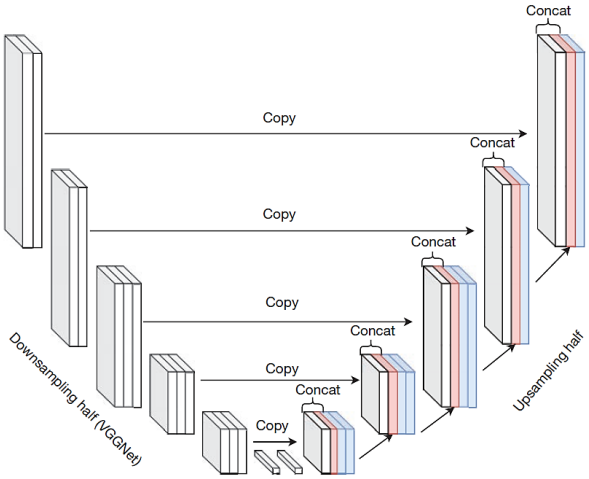


Figure 4. U-Net Architecture for Image Segmentation [25]

Table 2. Feed Forward Neural Network Architecture

	FFNN begins	
Linear Feed Forward 1	Input = 301088 (Flattened U-Net)	Output = 128
	Batch Normalization	
	ReLU	
	Dropout	
Linear Feed Forward 2	Input = 128	Output = 32
	Batch Normalization	
	ReLU	
	Dropout	
Linear Feed Forward 3	Input = 32	Output = 8
	Batch Normalization	
	ReLU	
	Dropout	
Linear Feed Forward 4	Input = 8	Output = 3 (3 Cobb angles)
	FFNN ends	

The CNN is complemented by a custom-built FFNN that interfaces seamlessly with the CNN output. This network comprises four feed-forward layers, each adorned with batch normalization, the ReLU activation function, and dropout layers. The only exception to this rule is the final layer, which lacks these enhancements. As shown in Table 2, a thorough examination of the FFNN's architecture reveals the following::

- The **initial layer** accepts the U-flattened Net's output, corresponding to 301,088 input nodes on a 2×388×388 grid. However, the data is immediately compressed, reducing the number of output nodes to 128.
- The **second layer** continues this consolidation pattern. This layer reduces its output to 32 nodes using the 128 nodes from the preceding layer as input.
- As we progress to the **third layer**, the input is the 32 nodes from the previous layer, which undergo another compression cycle; the output nodes are now number 8.
- In the **final layer**, the system takes the output of the third layer's eight nodes and reduces it to three nodes. These three nodes are illustrative because they predict the crucial Cobb angles: PT, MT, and TL.

3.3. Hyperparameters Design

The learning progression of the deep learning model is guided by a base learning rate (base lr) of 0.001. This rate affects how the model updates its parameters to reduce the Mean Squared Error (MSE) loss, a metric chosen for its effectiveness in regression tasks. MSE measures the squared deviations between observed and predicted numerical values. This model is optimized using the Adam optimizer, which offers adaptive learning rates for parameters, resulting in improved convergence and optimization. Initialized with the specified base lr, the optimizer also incorporates weight decay. This mechanism imposes a mild penalty on model weights during training, thereby encouraging the development of more generalized solutions.

The primary phases of training procedures are training and validation. The model operates in "train" mode during the training phase, permitting weight and bias adjustments. It computes a forward pass to predict outcomes for each data batch, determines the associated loss, and then adjusts the weights based on the gradients derived from the loss. In the meantime, the validation phase calculates the validation loss by iterating through the validation data and comparing it to the model's predictions while the model is in evaluation mode.

The number of epochs (num epochs) and the patience level for early stopping are important training parameters. The latter indicates when to terminate training if validation loss does not improve over several epochs. Metrics such as the training and validation losses and the learning rate values are recorded in lists for subsequent analysis as training progresses. In addition, a tracking mechanism is in place to determine and store the optimal model state based on the lowest observed validation loss.

Over the specified epochs, the training loop repeats, updating and validating as it cycles. If the model fails to demonstrate improved performance after a predetermined number of epochs (as specified by the patience parameter), the training process is terminated early. Finally, the total training duration is determined by subtracting the start and end times.

The optimal model, as determined by its lowest validation loss during training, is loaded and evaluated using the test dataset during the testing phase. The state of this model, referred to as the best model, is loaded from "best training 8.pt," which contains the best-obtained weights and biases. The model is then transferred to the appropriate device, typically a GPU, to ensure computation efficiency. Testing then commences, with the model set to evaluation mode via the best model.eval() to prevent any parameter modifications. Using its forward pass, the model processes the input images and predicts target angles. Then, these predictions are compared to actual values, and the test loss is aggregated across batches. The final step is calculating the average test loss, which provides insight into the model's performance on unseen data.

Evaluation of performance is essential, but nonlinear regression models, such as the one used in this study, lack the straightforward inference procedures of linear regression models. Nevertheless, linear regression metrics can provide insight into the model's performance by plotting predicted values versus actual values. In this context, the Root Mean Squared Error (RMSE) is used to evaluate the model's performance. While Mean Squared Error (MSE) accurately measures the difference between predicted and actual values, Root Mean Squared Error (RMSE) is preferred. This is because MSE values can become excessively high, making comparisons difficult. By taking the square root, the magnitude of the prediction error becomes more interpretable.

4. RESULTS AND DISCUSSION

In this section, we delve into the experimental setup, investigate a variety of neural network architectures, and thoroughly discuss the findings and their implications.

4.1. Experimental Setup

We utilized the NVIDIA GTX GeForce 1660 SUPER graphics card as the primary hardware resource for the experimental framework. This card is renowned for its powerful computational capabilities, making it ideal for data-intensive tasks such as training deep learning models and processing high-resolution data. The GTX GeForce 1660 SUPER, which serves as the backbone of our experimental platform, ensures rapid computations, drastically reducing the time required for iterative processes and model evaluations.

We chose JupyterLab as our interactive development environment for software. The user-friendly interface of JupyterLab enables researchers, academics, and data scientists to integrate live code execution with rich text narratives, mathematical equations, and dynamic visualizations. This combination of live code and rich media facilitates a holistic analytical approach, enhancing the interactivity and insight of experiment documentation, data visualization, and result interpretation.

Combining the immense computational power of the GTX GeForce 1660 SUPER with the adaptability of JupyterLab yields a state-of-the-art setup. This configuration accelerates the experimentation phase and improves the reproducibility and dissemination of findings. This arrangement is especially advantageous in academic and collaborative research environments where clarity, transparency, and efficacy are paramount.

4.2. Experimentation Profiles

In this section, we delve into the details of three distinct experimental configurations, each encapsulated by Table 3. The cornerstone of all three configurations is the U-Net architecture, a renowned neural network structure pivotal for image segmentation tasks. The U-Net's significance stems from its ability to meticulously capture intricate details and adeptly reconstruct target outputs from compressed feature representations.

Upon receiving the input, the U-Net in each experiment processes it to yield feature maps. These maps are then flattened, converting them into a one-dimensional structure. Following this transformation, the flattened outputs are smoothly incorporated into their corresponding Feed-Forward Neural Networks (FFNN). The FFNN takes the helm at this stage, utilizing the features extracted by the U-Net for further processing and eventually generating the final results.

A noteworthy point of distinction lies between Profile A and Profile C. While they are architecturally akin, variations in their hyperparameter configurations set them apart, as explicitly outlined in Table 4. This nuanced differentiation is instrumental in understanding the unique characteristics and outcomes of each experimental profile.

Table 3. FFNN Experimentation Profiles

	Profile A		**Profile B**		**Profile C**	
Linear Feed Forward 1	Input = 301088 (Flattened U-Net)	Output = 128	Input = 301088 (Flattened U-Net)	Output = 64	Input = 301088 (Flattened U-Net)	Output = 128
			Batch Normalization			
			ReLU			
			Dropout			
Linear Feed Forward 2	Input = 128	Output = 32	Input = 64	Output = 16	Input = 128	Output = 32
			Batch Normalization			
			ReLU			
			Dropout			
Linear Feed Forward 3	Input = 32	Output = 8	Input = 16	Output = 4	Input = 32	Output = 8
			Batch Normalization			
			ReLU			
			DropOut			
Linear Feed Forward 4	Input = 8	Output = 3 (3 Cobb angles)	Input = 4	Output = 3 (3 Cobb angles)	Input = 8	Output = 3 (3 Cobb angles)

Table 4. Hyperparameter Configurations

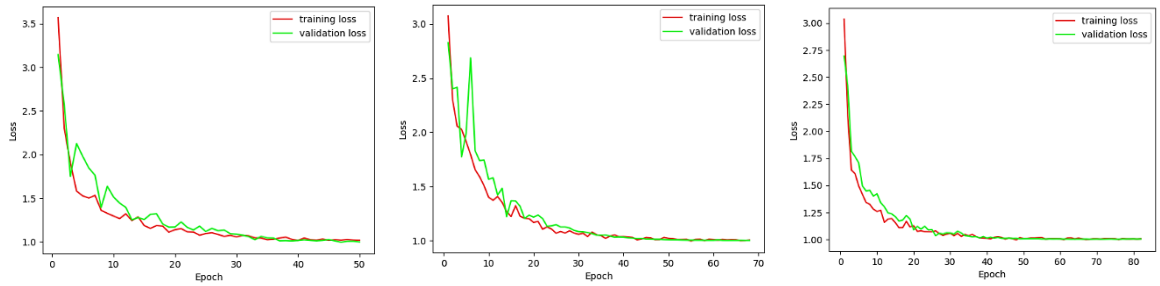
	Profile A	**Profile B**	**Profile C**
criterion (loss func.)	Mean Squared Error		
optimizer	ADAM		
number of epochs	50	75	1000
base learning rate	0.001		
patience	10	5	10
learning rate scheduler	No	Cosine Annealing Warm Restarts	No
weight decay	5e-6		
FFNN Dropout *p value*	0.45		

4.3. Experiments on Profile A, B, and C

For the Profile A experiment, the best possible model based on the lowest validation loss is loaded in this experimental evaluation. After extensive analysis, the calculated testing loss is determined to be 1.1978, using the Mean Squared Error (MSE) loss function. This carefully chosen evaluation metric, known for its capacity to quantify the differences between predicted and actual values, enables a thorough evaluation of how well the model performed on the unseen test dataset, confirming the reliability and validity of the reported results. The learning rate is maintained at 0.001 throughout all 50 epochs.

In an effort to refine the prediction accuracy of the PT and TL Cobb angle values from the Profile A experiment, the Profile B strategy integrates three essential modifications: the incorporation of learning rate scheduling, a reduction in hidden nodes within the Feed Forward Network (FFN) to curtail overfitting, and an extended number of training epochs. As a result, Profile B achieved a slightly improved testing loss of 1.1402 using the Mean Squared Error (MSE) loss function, indicating a closer alignment between predicted and actual values. The experiment employed the Cosine Annealing Warm Restarts learning rate scheduler, which fuses learning rate annealing with cyclical learning rates. The model can explore diverse solutions by periodically resetting the learning rate to its original value, thus avoiding plateaus and suboptimal outcomes during optimization. Specifically, after every 4 epochs, the learning rate is refreshed, transitioning from its base value of 0.001 to its minimum value of 0.00023.

The maximum number of epochs in the Profile C experiment is set at 1000, with an early stopping mechanism and a patience level of 10. Unlike previous profiles, this experiment does not incorporate a learning rate scheduler. As a result, the testing loss further diminishes to 1.1252, with the Mean Squared Error employed as the loss function. The learning rate is consistently maintained at 0.001 throughout the training. Notably, the training process ceases around the 81st epoch due to the early stopping criteria, signifying that the model failed to register any substantial improvements over 10 consecutive epochs.



(a) Profile A (b) Profile B (c) Profile C

Figure 5. Loss versus epoch for Profile A, B, and C experiments.

Figure 5(a) shows that the training and validation loss converged by the final epoch in the Profile A experiment. However, there were noticeable fluctuations in validation loss during the initial 15 epochs. These oscillations can be attributed to the network's regularization strategy, which introduced noise by altering node weights, momentarily impacting the training and validation losses. By employing image normalization (z-score normalization) and batch normalization, the model's loss converged rapidly within the first 5 epochs, signifying the consistent scaling of learning features. Beyond the 20th epoch, the loss reduction rate decelerated, a typical pattern in machine learning training. The slowing down indicates the model reaching a local minimum in the loss function and potentially becoming trapped there due to a set, modest learning rate. Despite this, the early stopping condition (with a patience of 5) was not triggered, as the model consistently found a new lowest validation loss approximately every 4 epochs throughout its 50 epoch duration.

In the Profile B experiment, as depicted in Figure 5(b), the training and validation loss trajectories closely mirror those observed in the Profile A experiment. However, a notable distinction is the reduced initial epoch losses in Profile B compared to its predecessor. The noticeable spike in validation loss around the 8th epoch can be attributed to the regularization effect. As the training progresses, early stopping mechanisms come into play just before the final epoch, resulting from the validation loss failing to improve over five consecutive epochs. Similar to the Profile A experiment, the training and validation losses exhibit marginal reductions before converging to an approximate loss of 1.

As depicted in Figure 5(c), the training and validation losses exhibit a minimal divergence throughout the training process. Unlike the noticeable fluctuations observed in the Profile A and B experiments, the validation loss in Profile C remains relatively stable. Interestingly, the initial losses in the first epoch closely mirror those in the Profile B experiment. Consistent behavior and the minimal disparity between training and validation losses are promising indicators. They suggest that the model in Profile C achieves a balanced

representation, avoiding underfitting and overfitting, which is essential for robust and generalizable performance.

4.4. Comparison of Predicted and Ground Truth Angles

Table 5 shows a discernible trend in the RMSE values across the three profiles when examining the PT, MT, and TL angles. Specifically, the RMSE for the PT angle exhibits a consistent decline from Profile A through to Profile C, indicating improved accuracy in predicting this particular angle. Conversely, the MT angle presents a different scenario, with its RMSE consistently escalating across all three experiments, signaling a challenge in its precise prediction. For the TL angle, the progression is a bit more nuanced; there is an initial decrease in the RMSE moving from Profile A to Profile B, suggesting an enhancement in prediction accuracy. However, this is followed by a slight uptick when transitioning to Profile C, though the reasons for this minor regression would warrant further investigation.

Table 5. RMSE for each profile experiment

	Proximal Thoracic (PT)	Mid Thoracic (MT)	Thoracolumbar (TL)
Profile A	4.5938	2.0329	2.6203
Profile B	4.4552	2.0478	2.5526
Profile C	4.3924	2.0493	2.5809

4.5. Relation between training and validation losses to epochs

Table 6 provides insightful trends regarding the relationship between the number of epochs and the resultant loss across Profiles A to C. Notably, as the number of epochs increases through these profiles, there is a subtle decline in training loss, suggesting that the model slightly benefits from extended training. Conversely, the validation loss exhibits an opposing trend. Instead of benefiting from the increased number of epochs, the validation loss consistently rises across the profiles. This divergence in patterns between training and validation losses might indicate that while the model is becoming adept at fitting the training data, it might be at the risk of not generalizing well to unseen data, a classic indication of overfitting. Further investigation and potential corrective measures might be necessary to address this discrepancy.

Table 6. Final epochs of every profile and its training and validation loss

	Training Loss	Validation Loss
Profile A (Epoch 50/50)	1.0183	0.9995
Profile B (Epoch 68/75)	1.0048	1.0018
Profile C (Epoch 82/1000)	1.0051	1.0038

4.6. Discussion

Table 5 illustrates the model's improved ability to predict the PT angle, as well as subsequent enhancements in the TL and MT angles. The data suggests that increasing the number of epochs improves the accuracy of PT angle predictions, but this advantage is not shared by MT angle predictions. The effect of learning rate scheduling on the model's overall performance appears to be relatively negligible. The consistent nature of the training and validation losses may be indicative of the model's struggle to escape local minima or saddle points during gradient descent computations, as suggested by this observation. Our model demonstrates a commitment to improving the precision of Cobb angle measurements, a challenge also addressed by [8] through the comparison of manual and automated methods. The high reliability and automation of measurements in their study highlight the potential and necessity for advancements in this field, thereby highlighting the importance of our research endeavors.

The correlation between the number of epochs and the resulting losses across the various profiles is examined in greater detail in Table 6. The transition from Profile A to Profile C signifies a subtle emergence of overfitting, as evidenced by a decrease in training losses and an increase in validation losses. This pattern is indicative of the model's increasing aptitude for predicting the training dataset, juxtaposed with a diminishing efficacy on unseen validation data. It is essential to note, however, that the difference in losses between Profiles A and C remains negligible, indicating the model's consistent convergence to a nearly identical minimum loss over a range of 50 to 82 epochs.

The results of Profiles A through C collectively illustrate the complexities and difficulties inherent in accurately predicting Cobb angle values. Given the nonlinear nature of the regression problem and the intricate patterns and relationships that characterize such scenarios, the task is extremely challenging. These complexities add multiple layers to the modeling process, necessitating a nuanced and meticulous approach to model creation and refinement.

Our findings, along with the insights of [8], highlight the ongoing difficulties and the need for continued innovation in the field of Cobb angle prediction. The pursuit of more accurate and trustworthy methods continues to be of paramount importance, with the potential to significantly impact the diagnosis and treatment of scoliosis.

5. CONCLUSIONS AND FUTURE WORKS

In this exhaustive study, we embarked on a complex journey to elucidate the predictive ability of various neural network models on Cobb angle values, a task made more difficult by the nonlinear regression characteristics of neural networks. We investigated the effects of varying epochs, learning rate scheduling, and neural architectures on model performance using three distinct profiles: A, B, and C. Even though increasing epochs improved predictions for the PT angle, it was not beneficial for all angle types, particularly the MT angle, as indicated by our findings. In addition, the study emphasized the nuanced role of learning rate scheduling, whose impact appeared somewhat muted, possibly due to difficulties in escaping local minima during gradient descent. The onset of overfitting, particularly during the transition from Profile A to C, was a significant finding throughout our experiments. Such overfitting nuances are crucial, highlighting the model's prowess in predicting training data but revealing its reluctance when confronted with unknown validation datasets. Despite these variations, all models converged to a nearly identical minimal loss, indicating consistent performance across profiles. This paper highlights the complexities and subtleties of nonlinear regression in predicting Cobb angle values. While significant progress has been made in comprehending the various influencing factors, it is evident that the search for the optimal predictive model remains difficult. We believe that our findings provide a solid basis for future research, with the potential to inspire innovations in neural network architectures and training strategies tailored to complex regression scenarios.

ACKNOWLEDGMENTS

This research received generous support from the Malaysian Ministry of Education, facilitated through the Fundamental Research Grant for Research Acculturation of Early Career Researcher under the grant number RACER/1/2019/SKK06/UIAM//5, specifically RACER19-016-0016. Further, our heartfelt gratitude extends to the Faculty of Engineering at IIUM and IIUM Kuantan Hospital, who have graciously provided essential facilities and resources that significantly contributed to this research endeavor.

REFERENCES

[1] J. A. Janicki and B. Alman, "Scoliosis: Review of diagnosis and treatment," *Paediatrics & child health*, vol. 12, no. 9, pp. 771-776, 2007.

[2] N. A. Makhdoomi *et al.*, "Development of Scoliotic Spine Severity Detection using Deep Learning Algorithms," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 2022: IEEE, pp. 0574-0579.

[3] K. Watanabe, Y. Aoki, and M. Matsumoto, "An application of artificial intelligence to diagnostic imaging of spine disease: estimating spinal alignment from Moiré images," *Neurospine*, vol. 16, no. 4, p. 697, 2019.

[4] P. Bernstein, J. Metzler, M. Weinzierl, C. Seifert, W. Kisel, and M. Wacker, "Radiographic scoliosis angle estimation: spline-based measurement reveals superior reliability compared to traditional COBB method," *European spine journal*, vol. 30, pp. 676-685, 2021.

[5] V. Wu, T. Ungi, K. Sunderland, G. Pigeau, A. Schonewille, and G. Fichtinger, "Automatic segmentation of spinal ultrasound landmarks with U-net using multiple consecutive images for input," in *Medical Imaging 2020: Image-Guided Procedures, Robotic Interventions, and Modeling*, 2020, vol. 11315: SPIE, pp. 572-577.

[6] Y. Pan *et al.*, "Evaluation of a computer-aided method for measuring the Cobb angle on chest X-rays," *European Spine Journal*, vol. 28, pp. 3035-3043, 2019.

[7] M. Horng, C. Kuok, M. Fu, C. Lin, and Y. Sun, "Cobb angle measurement of spine from X-Ray images using convolutional neural network. Comput Math Methods Med. 2019; 2019: 6357171," ed: Epub 2019/04/19. https://doi. org/10. 1155/2019/63571 71. PubMed PMID: 30996731.

[8] Y. Sun, Y. Xing, Z. Zhao, X. Meng, G. Xu, and Y. Hai, "Comparison of manual versus automated measurement of Cobb angle in idiopathic scoliosis based on a deep learning keypoint detection technology," *European Spine Journal*, pp. 1-10, 2022.

[9] Y. Ishikawa *et al.*, "Prediction of Cobb Angle Using Deep Learning Algorithm with Three-Dimensional Depth Sensor Considering the Influence of Garment in Idiopathic Scoliosis," *Journal of clinical medicine*, vol. 12, no. 2, p. 499, 2023.

[10] Z. Zhou, J. Zhu, and C. Yao, "Vertebral Center Points Locating and Cobb Angle Measurement Based on Deep Learning," *Applied Sciences*, vol. 13, no. 6, p. 3817, 2023.

[11] M. Fraiwan, Z. Audat, L. Fraiwan, and T. Manasreh, "Using deep transfer learning to detect scoliosis and spondylolisthesis from X-ray images," *Plos one*, vol. 17, no. 5, p. e0267851, 2022.

[12] W. Caesarendra, W. Rahmaniar, J. Mathew, and A. Thien, "Automated Cobb angle measurement for adolescent idiopathic scoliosis using convolutional neural network," *Diagnostics*, vol. 12, no. 2, p. 396, 2022.

[13] T. Zhang, Y. Li, J. P. Y. Cheung, S. Dokos, and K.-Y. K. Wong, "Learning-based coronal spine alignment prediction using smartphone-acquired scoliosis radiograph images," *IEEE Access*, vol. 9, pp. 38287-38295, 2021.
[14] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, 2019, vol. 1168: IOP Publishing, p. 022022.
[15] E. Britannica. "Vertebral Column." https://www.britannica.com/science/vertebral-column [accessed 8 August 2023],
[16] M. N. Choudhry, Z. Ahmad, and R. Verma, "Adolescent idiopathic scoliosis," *The open orthopaedics journal*, vol. 10, p. 143, 2016.
[17] Z. Wang, K. Liu, J. Li, Y. Zhu, and Y. Zhang, "Various frameworks and libraries of machine learning and deep learning: a survey," *Archives of computational methods in engineering*, pp. 1-24, 2019.
[18] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv preprint arXiv:2003.05689*, 2020.
[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[20] J. Brownlee, "How to configure the learning rate when training deep learning neural networks," *Machine Learning Mastery*, vol. 6, 2019.
[21] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*, 2017: IEEE, pp. 464-472.
[22] N. Tanksale. "Finding Good Learning Rate and The One Cycle Policy." https://towardsdatascience.com/finding-good-learning-rate-and-the-one-cycle-policy-7159fe1db5d6 [accessed 8 August 2023],
[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of machine learning research*, vol. 15, no. 1, pp. 1929-1958, 2014.
[24] H. Wu, C. Bailey, P. Rasoulinejad, and S. Li, "Automatic landmark estimation for adolescent idiopathic scoliosis assessment using BoostNet," in *Medical Image Computing and Computer Assisted Intervention− MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part I 20*, 2017: Springer, pp. 127-135.
[25] M. Ekman, *Learning deep learning: Theory and practice of neural networks, computer vision, NLP, and transformers using TensorFlow*. Addison-Wesley, 2021.