Univerzitet u Novom Sadu
Prirodno-matematički fakultet
Departman za matematiku i informatiku

Milica Papić

Recognition of Pediatric Joint Fracture Using Convolutional Neural Networks

Master Thesis

Supervisor:

Oskar Marko, PhD

Novi Sad, 2024.

Acknowledgements

I would like to express my sincere gratitude to my mentor, Mr. Oskar Marko PhD whose guidance has been invaluable throughout this journey. His expertise has not only provided clear direction but has also deepened my understanding and appreciation of the subject.

To my family, I extend my heartfelt thanks for your unwavering love and support. To my dear sisters, Katarina and Tijana, your constant encouragement has been instrumental in this accomplishment. Your patience and understanding during the most challenging times have meant the world to me.

Finally, I would like to express my sincere thanks to those special ones who stood by me when it mattered most. Your utmost professionalism, unwavering dedication, and above all, your gentle love and care, gave me the strength and faith to take the final leap.

Contents

List of Figures

List of Tables

1 Abstract

In recent years, innovations in Artificial Intelligence (AI) have profoundly impacted various domains, with healthcare being one of the most significant beneficiaries. The rapid advancements in AI, particularly in the field of Deep Learning (DL), have opened new opportunities for improving diagnostic accuracy and efficiency in medicine.

This thesis focuses on the diagnostic application of Convolutional Neural Networks (CNNs) for detecting pediatric wrist joint fractures. The goal is to train and fine-tune several models on X-ray images in order to enable them to make accurate predictions on previously unseen images, specifically to determine whether a patient has a visible fracture or not. The aim is to enhance the model's ability to accurately identify the presence or absence of fractures, thereby improving diagnostic precision.

Three well-established CNNs architectures were used: ResNet18, ResNet50 and ResNet101. There are 20,327 X-Ray images, 9,730 of which were used, sourced from the Department of Pediatric Surgery of the University Hospital in Gratz, Austria. They consist of both cases where fractures are visible and cases where no fracture is present. In addition to the images, a patient dataset is at our disposal. It contains patient information including a column that links each image to a corresponding label, where 1 indicates a visible fracture and 0 indicates the absence of a fracture. The images and their corresponding labels are fed into the CNNs models to train them to accurately classify the presence or absence of fractures. These images and labels are used as input to develop a model that is capable of predicting whether a fracture is present, based solely on the visual information in the image. The model that showed the best performance in this study is ResNet50, achieving an accuracy of 94,76% and F1 Score of 95,93% on the unseen images, demonstrating its strong ability to generalize. This research not only demonstrates the potential of CNNs in automating and improving the diagnostic process, but also provides a comparative analysis of the performance of different network architectures with varying complexities in medical imaging tasks.

2 Introduction

Artificial Intelligence (AI) plays a key role in transforming the way medical industry works. Diagnostics is one of the most important areas for applying AI in medicine. Different branches of AI are used as a modern tools in medicine due to the large amount of available data that can be analyzed and interpreted in a very effective way. It benefits both doctors and patients.

As stated in [1], AI can be defined as the effort to automate intellectual tasks that usually require human intelligence. AI is any technique that enables computers to mimic, simulate, or replicate human-like cognitive abilities such as learning, reasoning, problem-solving and decision making. According to [2], there are various domains where AI techniques are applied:

1. Theorem proving: this activity traces back to the earliest days of AI and has produced significant results despite no new theorems being proven by machines;

2. Natural Language Processing (NLP): this term covers AI abilities, such as interpreting and generating sentences, accessing databases, retrieving documents as well as computer-aided translation, requiring extensive knowledge and reasoning techniques;

3. Speech recognition and understanding: this domain deals with decoding speech signals, presenting challenges due to inherent indeterminism, but commercial products are available for tasks like dictation and transcription;

4. Image interpretation and vision: early image processing systems focused on simple tasks, but more advanced applications involve interpreting images for diagnosis or guiding robots, requiring knowledge-based reasoning;

5. Robotics: while early robots repeated learned movements, modern ones incorporate AI for perception, reasoning, planning and dynamic replanning of actions to reach a certain goal;

6. Expert systems: these knowledge-based systems, developed since the 1970s, demonstrate the power of small amounts of knowledge for intelligent decision-making in various domains, evolving beyond strict logical deduction to include reasoning modes like analogy and model-based reasoning;

7. Machine Learning (ML).

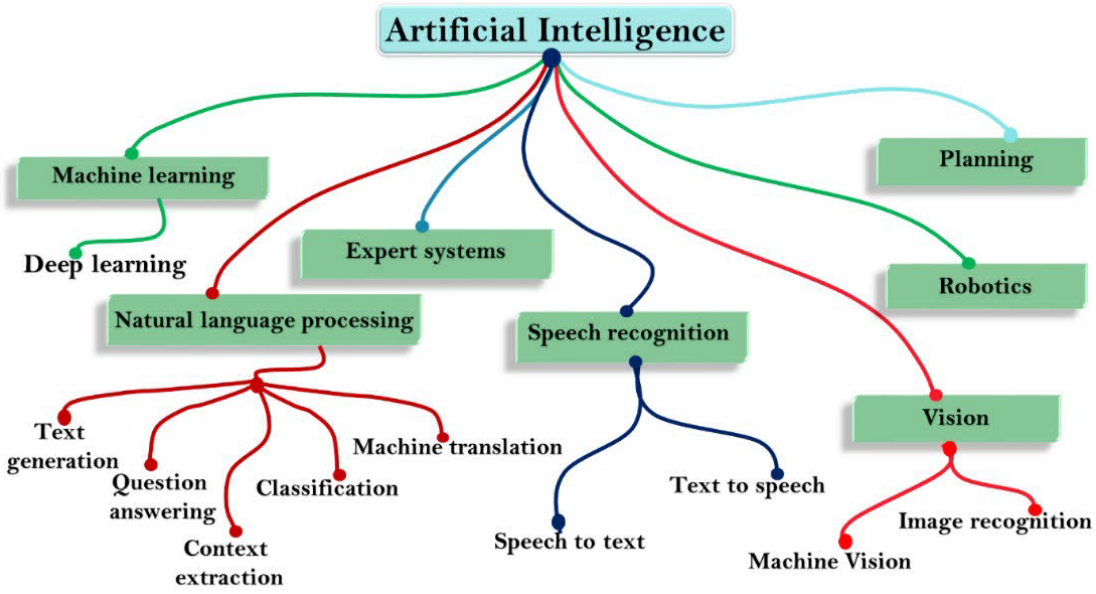

Figure 2.1: AI branches

ML algorithms play a major role in the analysis and later prediction of predispositions to genetically transmitted diseases, which allows the adaptation of therapy at an adequate preventive time for each patient individually. The concept of ML, as discussed in [2], comes from the curiosity of whether a computer can go beyond simply following explicit instructions and instead learn to autonomously perform tasks, including those it has never encountered before. In other words, ML provides intelligence to machines with the ability to automatically learn from experiences without being explicitly programmed. The goal of learning is not only memorization but also the ability to generalize on new, unseen situations. As mentioned in [3], there are two main approaches: Supervised Learning and Unsupervised Learning.

Supervised Learning, involves training a model on labeled data, where each example has a known outcome. The model learns to predict the outcome for new, unseen data based on patterns learned from the training set.

Unsupervised Learning operates on unlabeled data, aiming to uncover hidden patterns or structures within the data. Without predefined outcomes, algorithms cluster similar data points together or reduce the dimensionality of the data.

Over the past decade, Deep Learning (DL) has become one of the most popular topics in the field of ML. The history of DL is a journey over several decades, marked by numerous milestones that have shaped its development and success. The idea of DL and Artificial Neural Networks (ANNs) comes from an attempt to

build a computer system that can behave like a human brain. It is necessary to understand the functionality of human cognitive system in order to build such an artificial system.

As stated in [4], the idea came from Aristotle (384 B.C – 322 B.C) and his attempts to understand the brain around 350 B.C. He examined the processes of memory and recall and presented them with four laws of associations:

1. **Contiguity**: Things or events with spatial or temporal proximity tend to be associated in the mind.

2. **Frequency**: The number of occurrences of two events is proportional to the strength of association between these two events.

3. **Similarity**: Thought of one event tends to trigger the thought of a similar event.

4. **Contrast**: Thought of one event tends to trigger the thought of an opposite event.

Aristotle described the implementation of these laws in our mind as common sense. For example, the feel, the smell or the taste of an apple should naturally lead to the concept of an apple, as common sense. These laws still serve, 2000 years later, as the fundamental assumptions of ML methods.

DL, especially Convolutional Neural Networks (CNNs), allow accurate analysis of medical images such as CT, MRI and X-ray. Newly created algorithms can identify pathologies, tumors or bone fractures more precisely than the traditional methods we are used to, resulting in early disease detection.

Nowadays, digital radiography is widely available, representing a modern technology of imaging in medical diagnostic procedures. Unlike better-known X-ray films, this method uses digital sensors which allow more efficient and faster X-ray images of the patients. Therefore, the process of diagnosis is much faster. This combination of technologies brings a number of advantages. Primarily, the automation of result analysis reduces waiting time for results, enabling rapid diagnostics and emergency medical intervention. Additionally, the application of Neural Networks (NNs) improves the accuracy of problem identification, reducing the possibility of human error. Digital radiography contributes to the efficiency of the entire health care system since the sharing of X-ray images has never been faster and therefore communication and exchange of information between different medical experts can be much better and more efficient.

The availability of large-scale medical datasets, like the one used for the purpose of this thesis [5], has enabled further advancements in AI-driven diagnostics. These datasets have made it possible to train complex ML models and NNs to analyze a variety of medical data with remarkable precision.

In [6] the authors developed CheXNet, a DL model designed to detect pneumonia from chest X-rays. Trained on over 100,000 labeled X-ray images with 14 diseases, CheXNet surpassed the diagnostic accuracy of radiologists, providing a significant contribution to the field of radiology.

A study [7], illustrated the huge potential of AI in dermatology. The authors successfully trained a Deep Neural Network(DNN) to classify skin cancer using a dataset of 129,450 clinical images. The model demonstrated high accuracy in identifying both benign and malignant skin lesions, highlighting the capability of AI to enhance diagnostic precision in dermatology.

Following the steps of related papers, this thesis first covers the data and materials. The results and discussion include the process of image preprocessing, model training and final performance results in fracture detection of three architectures: ResNet18, ResNet50 and ResNet101.

ResNet was first introduced by Microsoft Research researchers in [8]. ResNet has been widely used in the classification of medical images across various studies due to its powerful feature extraction capabilities and ability to handle deep architectures effectively.

In [9] ResNet was applied to classify diabetic retinopathy stages from retinal fundus images and achieved accuracy of 98.32% for APTOS and 98.71% for Kaggle datasets.

In [10] ResNet-50 significantly improved the classification accuracy of different types of brain tumors from MRI images.

3 Materials and Methods

3.1 Data

As previously mentioned, automated fracture detection has become a topic of intense research, offering the potential to speed up the diagnosis process and ease the workload of medical professionals. In order to support research in this area, the publicly available GRAZPEDWRI-DX dataset is presented to encourage computer vision research [5].

3.1.1 Dataset Description

This dataset contains an extensive collection of X-rays of childrens' wrist joint fractures, collected over a period of ten years, between 2008 and 2018, at the Department of Pediatric Surgery of the University Hospital in Gratz, Austria.

20,327 images are available for analysis and research. They are spread over four folders of around 4GB each. All X-rays were de-identified and converted to 16-bit grayscale PNG images.

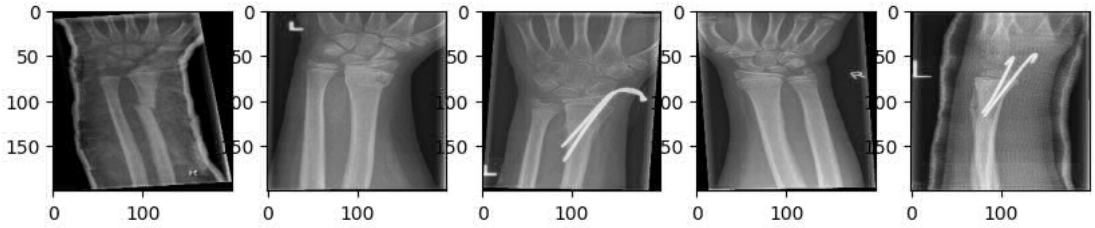

Figure 3.1: Five projections from the dataset

A dataset, with 20,327 rows × 17 columns, that contains information about patients, is also at the disposal. The columns are as follows: 'filestem', 'patient_id', 'study_number', 'timehash', 'gender', 'age', 'laterality', 'projection', 'initial_exam', 'ao_classification', 'cast', 'diagnosis_uncertain', 'osteopenia', 'fracture_visible', 'metal', 'pixel_spacing', 'device_manufacturer'.

The first column 'filestem' contains the names of the images, connecting the information from the dataset with the available images.

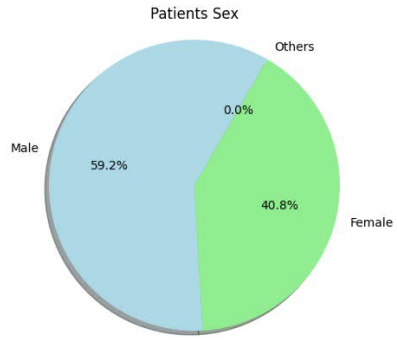

Figure 3.2: Gender distribution of the dataset

The majority of dataset patients are male, while one person defined herself/himself as a third gender, presented in figure 3.2.

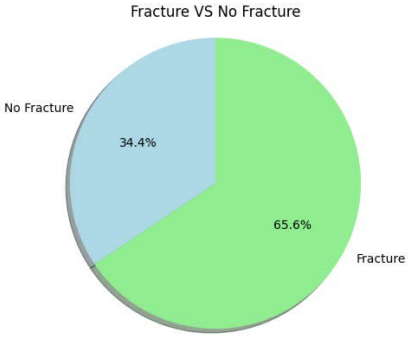

Figure 3.3: Fracture distribution of the dataset

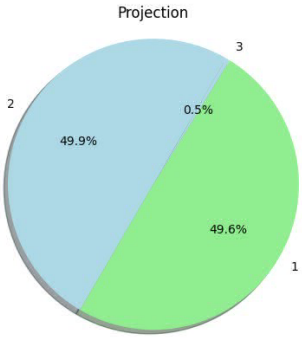

Figure 3.4: Projection distribution of the dataset

The most significant column for the thesis is the ‘fracture_visible’ column,

providing a label by which the NNs are trained. Distirbution of label 1 (fracture is visible) and label 0 (fracture is not visible) is presented in figure 3.3.

The 'projection' column contains values 1, 2 and 3 providing an information from which angle the X-rays were taken 3.5.

Figure 3.5: Projections of the patient ID=578

Prior to training, several preprocessing steps were applied to ensure that the input data was properly formatted for the learning process.

3.1.2 Data Preprocessing and Transformation

The first step involved the creation of a data loader, which was responsible for efficiently managing and feeding the images and their corresponding labels into the network during training. As a guide, the code available on GitLab [11] was utilized. Specific transformations, suggested by PyTorch documentation, were applied to each image in the dataset in order to standardize the input images and enhance the model's robustness.

The following transformations were implemented:

1. **Random Affine Transformation:** was applied to the images with a rotation range of -5 to 5 degrees, a translation range of 5% in both directions and a scaling factor between 0.95 and 1.05. This augmentation was applied with a probability of 70% to introduce variability and help the model become more robust to changes in image orientation, position and scale.

2. **Resizing to a Uniform Dimension:** all images were resized to a fixed dimension of 224×224 pixels to ensure consistency of the input data.

3. **Normalization:** the mean and standard deviation were computed across all dataset image pixels, after resizing to 224×224 pixels, resulting in a mean of

0.2987 and a standard deviation of 0.1566, which were then used to normalize the dataset for consistent model training.

4. **Random Horizontal Flip:** random horizontal flipping was applied to the images to augment the dataset and improve the generalization of the model. It artificially increased the diversity of the training data.

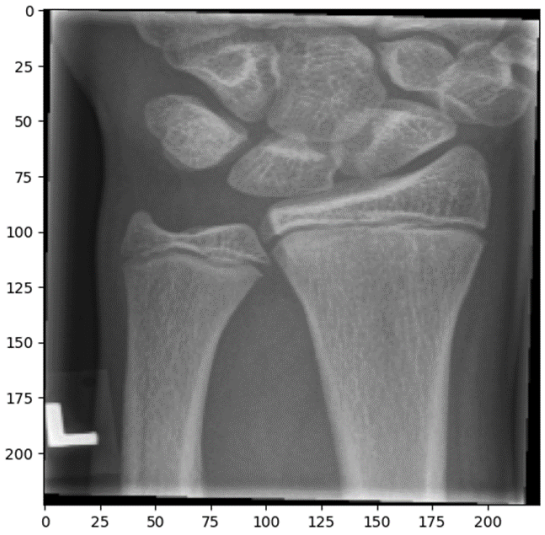

Figure 3.6: Image before transformations

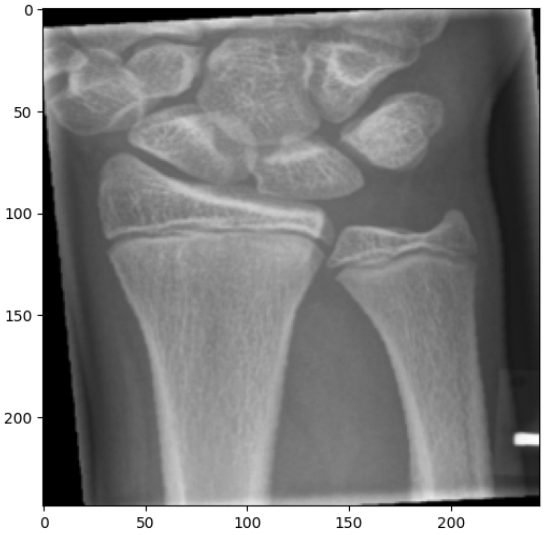

Figure 3.7: Image after transformations

Images labeled with value 1 in column 'projection' and with value 0 in column 'metal' were selected and used for more accurate results, reducing dataset to 9,730 rows × 17 columns.

The dataset is split into: a training set (80%-7,788 images), a validation set (10%-973 images) and a test set (10%-973 images).

A significant characteristic of the dataset is its class imbalance. This imbalance could potentially lead to biased training, where the model might favor the more frequent class, resulting in suboptimal performance on the minority class. To address this issue, a class balancing strategy was employed during the training phase. Custom sampler was implemented to balance the classes by ensuring that each batch contained a roughly equal number of images from both the fracture and non-fracture classes. A batch size of 64 was chosen for the training, validation and testing set offering a good balance between computational efficiency and model performance. For the training process, the Binary Cross-Entropy was selected, being the loss function to handle the binary classification task, AdamW optimizer to update the model's weights, with a learning rate of 0.0001 and a weight decay of

0.001 to prevent overfitting. Additionally, a learning rate scheduler was employed to adjust the learning rate dynamically based on the model's performance, reducing it by a factor of 0.1 if the validation performance plateaued for 5 consecutive epochs. All models were trained for a total of 20 epochs.

3.2 Artificial Intelligence Algorithms

In [12], in the chapter Capacity, Overfitting and Underfitting, it is presented that in ML algorithms, the biggest obstacle is making sure that models work well, not just on the data they have seen before, but also on brand-new data they have never encountered before. This ability to handle new situations and perform well on unseen data is called *generalization*.

When a model is trained, the goal is to make it better by adjusting its settings to reduce errors on the existing data. The aim is not only to reduce errors on the training data-the training error, but also to have the model make smallest possible errors on the new, unseen data. This is what it is call *generalization error* or the *test error*.

Thus, a trial environment is created and predicts how the model will perform in the real-world where it encounters different situations.

This process suggests to divide the input data into three sets: train, validation and test.

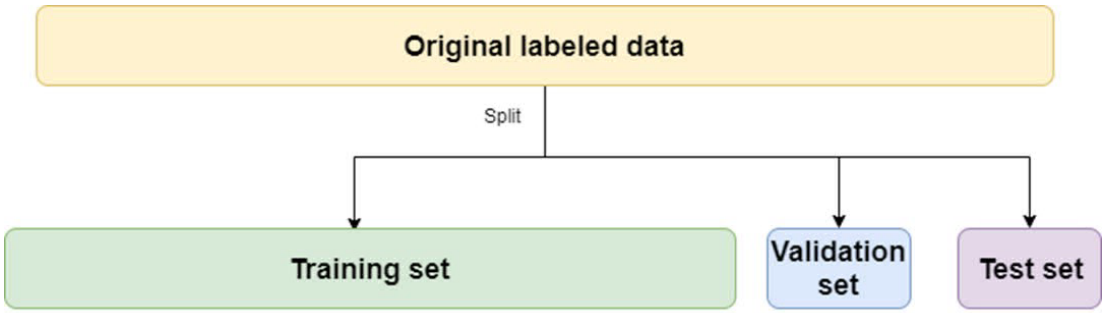

Figure 3.8: Splitting data into train validation and test subsets

The training set is used to fine-tune the model's parameters in order to minimize errors specific to the training data. The test set is used to evaluate how the model generalizes on new, unseen data. In this process, the expected error on the test set is typically equal to or greater than the expected error on the training set. This is because the model is optimized based on the training data and may not perform as well on data it has not seen before.

A validation set is introduced, consisting of examples that the training algorithm does not see. A separate test set, comprised of examples from the same data

distribution as the training set, is crucial for estimating the model's generalization error after training.

The efficacy of learning algorithms often depends on a multitude of adjustable parameters that regulate their behavior. These adjustable parameters, *hyperparameters* play a pivotal role in shaping the learning process and ultimately influencing the performance of the model. Unlike the internal parameters of the model, such as weights and biases, which are iteratively refined during the training process, hyperparameters remain constant throughout the training phase and are not adapted by the learning algorithm itself.

It is essential that the test set examples are not utilized in any way to make decisions about the model, including its hyperparameters. Therefore, none of the test set examples are included in the validation set. Instead, the validation set is created from the training data by splitting it into two distinct subsets. One subset is used for learning the model's parameters, while the other serves as the validation set for estimating the model's generalization error during or after training. This allows the update of the hyperparameters accordingly.

When the test set is small, it is hard to confidently compare algorithms. With large datasets, this is not a problem, but with small ones, k-fold *cross-validation* can be used. This involves splitting the data into k subsets, rotating which one is the test set in each iteration to get a more reliable estimate of performance.

Regularization is another important term which should be mentioned.

As stated in [13], regularization is any supplementary technique that aims at enhancing the model's ability to generalize, i.e. aims at producing better results on the test set. It encompasses various techniques aimed at reducing test error without significantly increasing training error. These techniques can include adjustments to the loss function, the optimization algorithms, or other supplementary methods.

There are two main factors that determine the success in the performance of a ML algorithm:

1. The ability of the algorithm to minimize errors on the training data, ensuring it learns the patterns present in the training set accurately.

2. The ability of the algorithm to generalize well to new, unseen data, which means minimizing the difference between the error on the training data and the error on the test data. It ensures that the model performs well in real-world applications beyond the training set.

At this point, it is important to mention core challenges in ML: *Underfitting and Overfitting*.

Underfitting arises when the model fails to achieve a sufficiently low error rate on the training set, indicating that it has not adequately captured the underlying

patterns in the data. On the other hand, overfitting occurs when the model becomes too complex and captures noise or irrelevant details present in the training data, resulting in a large gap between the training error and test error. 3.9

Underfitting and overfitting can be prevented by adjusting the model's capacity. Informally, a model's capacity refers to its ability to represent a wide range of functions or patterns in the data. Models with low capacity may struggle to capture the complexity of the underlying data distribution, leading to underfitting. Conversely, models with high capacity possess the flexibility to fit the training data extremely well, often by memorizing specific instances or noise present in the data. However, this increased capacity can also lead to overfitting, where the model fails to generalize effectively to new, unseen data.

Therefore, finding the right balance in model's capacity is crucial in ML. It involves selecting a model complex enough to capture the underlying patterns in the data while avoiding the pitfalls of both underfitting and overfitting.

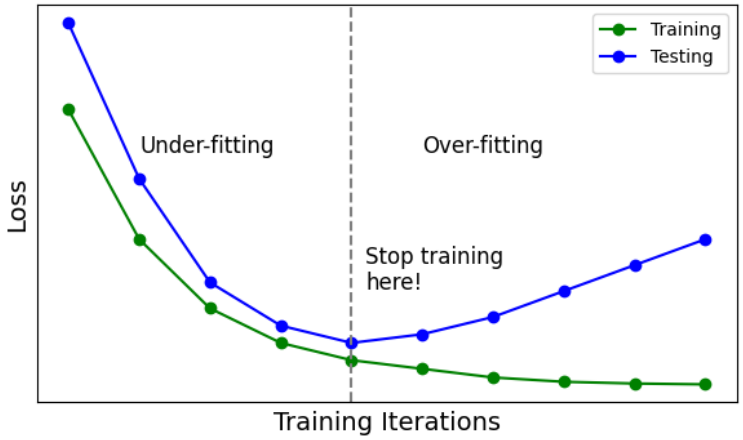

Figure 3.9: Underfitting and Overfitting

Dropout is an important and widely used regularization method to prevent overfitting.

The main idea behind dropout is to randomly "drop" or deactivate a percentage of neurons during training, ensuring the network does not depend on any neuron or a specific subset of neurons resulting in more reliable feature extraction and generalization.

3.2.1 Deep Learning

The first model of neuron was developed by neurophysiologist Warren McCulloch (1898–1969) and a mathematician Walter Pitts (1923–1969) in 1943. They speculated the inner workings of neurons and modeled a primitive electrical circuits' neural network based on their findings.

The model aimed to describe how individual neurons in the brain might work as binary logic gates, laying the foundation for ANNs. Key aspect of the McCulloch-Pitts neuron model is that the neuron is represented as a binary unit with a simple threshold activation function. It could be either "firing" (active or outputting a signal) or "not firing" (inactive or outputting no signal). Also the neuron received a fixed binary inputs from other neurons or external sources.

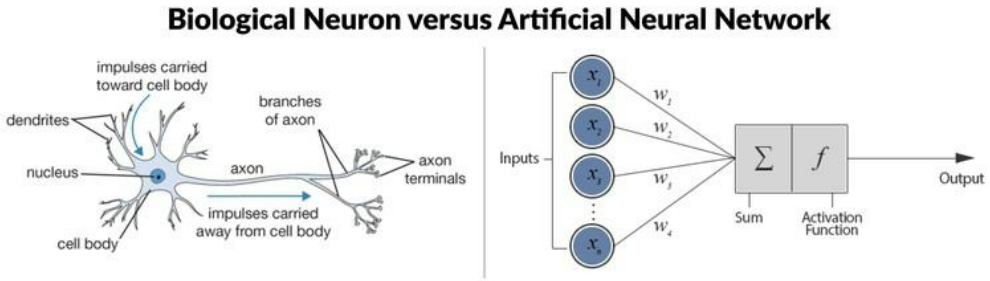

Figure 3.10: Biological vs Artificial Neuron

A simple single neuron is called *percepton*.

3.2.1.1 Perceptron Learning

The perceptron, a fundamental unit in NNs, functions as a scalar function of various variables, determined by weight coefficients representing the linear discriminant function. Mathematically, it operates by applying *weights* and *bias* to inputs, creating a new variable, which is then passed through an *activation function* to predict the final output.

Rosenblatt's perceptron model involves taking multiple binary inputs and producing a single binary output based on weighted sums of these inputs, compared with a predefined threshold value. This simple yet powerful concept forms the basis of how perceptrons work, where the weights assigned to each input determine their importance in influencing the final output.

Weights in NNs indicate the importance of each input x in the decision-making process, determining the amount of influence with which each input affects the neuron's output. Weights are also sometimes called the *parameters* of a layer.

Bias is an additional parameter, a constant, added to the weighted sum of inputs before passing it through the activation function. It allows the model to shift the activation function left or right, providing more flexibility in fitting and adapting to different patterns in the data.

The importance of Activation Functions (AFs) in NNs is emphasized in [14].

3.2.1.2 Activation Functions

AF enable NNs to learn abstract features through nonlinear transformations. The authors highlight several essential properties that AFs should possess, such as adding nonlinear curvature to the optimization landscape, without increasing computational complexity or hindering gradient flow during training. Some of the most crucial activation functions employed in NNs are: Logistic Sigmoid, Tanh and Rectified Linear Unit (ReLU) functions. These activation functions play a pivotal role in introducing non-linearity into NNs, each with its unique characteristics and implications for model performance. By understanding their properties, a valuable insights are gained into their suitability for various applications and their impact on network behavior.

The Logistic Sigmoid function maps inputs to the range $[0, 1]$, which is defined by the formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

However, it suffers from vanishing gradient issues due to saturation at higher and lower inputs. The *vanishing gradient problem* occurs when the gradients become extremely small during the training of NNs, impeding effective learning by causing earlier layers to receive minimal updates. The Sigmoid function is mainly used in binary classification problems, but its output is not zero-centered.

The Tanh function maps inputs to the range $[-1, 1]$, and is defined by the formula:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

This function addresses the zero-centered property, which helps in centering the data and potentially speeds up convergence. Despite this advantage, the Tanh function still faces vanishing gradient problems. Its derivatives are steeper than those of the Sigmoid function, which can help mitigate the issue to some extent, but does not entirely eliminate the vanishing gradient problem.

Addressing the limitations of Sigmoid and Tanh activation functions, the Rectified Linear Unit (***ReLU***) has emerged as a popular choice due to its simplicity and improved performance. This function is defined as:

$$\mathrm{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

ReLU overcomes the saturation problem and computational complexity associated with Sigmoid and Tanh functions. It produces outputs in the range $[0, \infty)$. ReLU activates only those neurons with positive values, making it computationally efficient. However, it faces the *Dying ReLU* problem, where units may consistently output zero due to negative inputs, which hinders learning by preventing weight updates during training. Several variants have been developed to address the Dying ReLU problem, one of them being *Leaky ReLU*. This variant allows a small, non-zero gradient for negative inputs, defined as:

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

where α is a small constant (e.g. 0.01). This modification helps maintaining the flow of information and preventing units from becoming inactive.

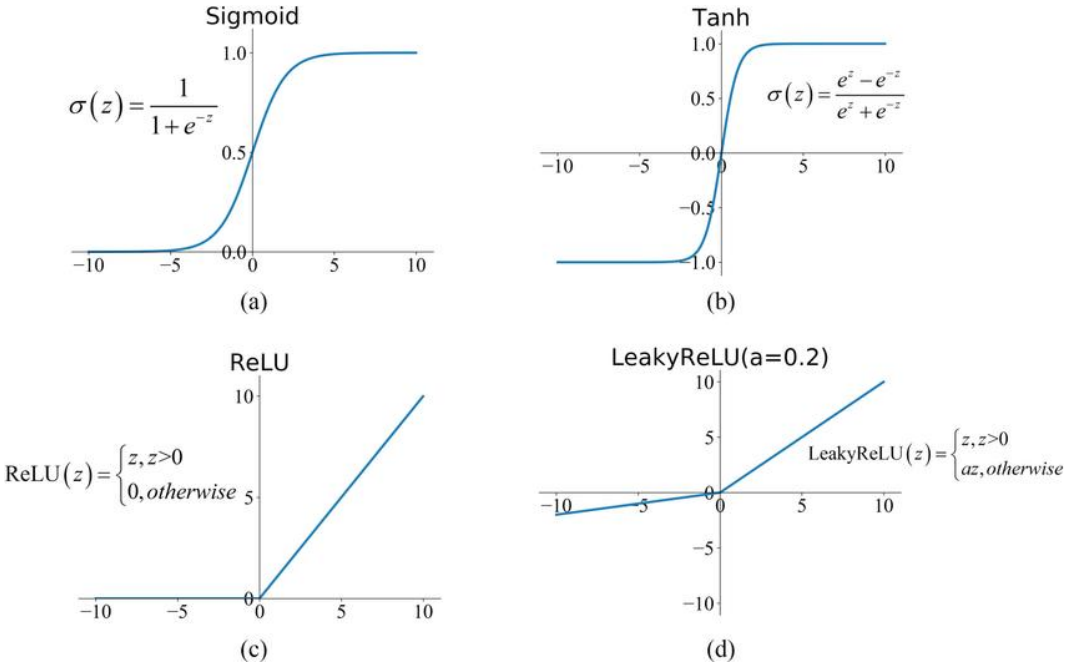

Figure 3.11: Activation Functions:Formulas and Slopes

Nonlinearity is crucial in NNs because it allows the network to model complex relationships between data points. To illustrate this, a hypothetical scenario, inspired by [15], should be considered. Dataset consisting of 100 points, some of which are red and others are green is shown in figure 3.12. With natural human intelligence, distinguishing and classifying these points based on color is a straightforward task. However, when it comes to machines, this process is not as intuitive.

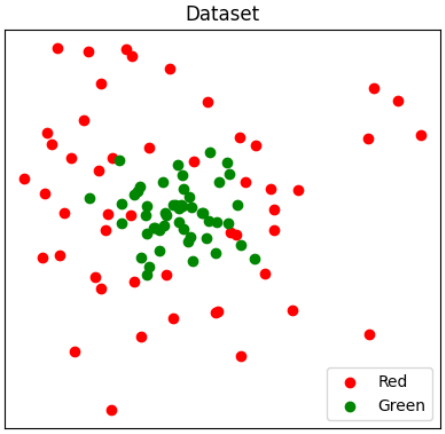

Figure 3.12: Dataset of 100 dots

If it is assumed that the NNs function is linear, the only option available to the network is to place a straight line at a position where it believes the division between the red and green points is most accurate. While this might provide some level of classification, it is inherently limited. A linear function can only create a simple, straight boundary, which might not effectively separate the points, as illustrated in figure 3.13. This limitation arises because linear functions can only capture linear relationships, which might not be sufficient for the task at hand.

On the other hand, by introducing nonlinearity into the network, we enable it to model more complex relationships between the data points. Nonlinear activation functions allow the network to create more sophisticated decision boundaries, potentially resulting in a classification that is closer to the true distribution of the data, as shown in figure 3.14. This enhanced capability is why the choice of activation function is critical in designing NNs. The right activation function can significantly improve the network's ability to learn and generalize, ultimately leading to more accurate and meaningful classifications.

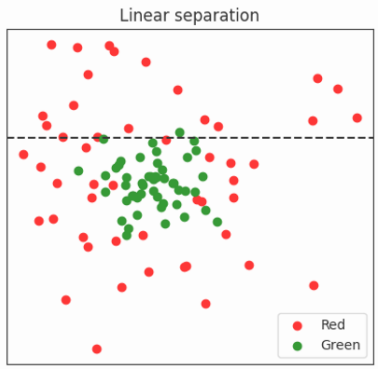

Figure 3.13: Linear Separation

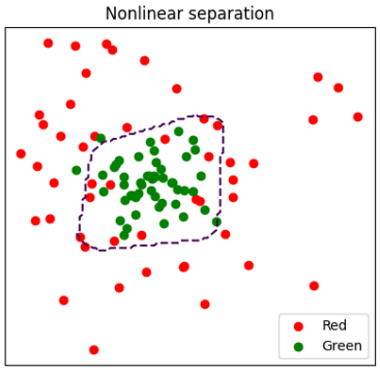

Figure 3.14: Nonlinear Separation

3.2.1.3 Multilayer Perceptrons

Activation functions are crucial in building NNs, particularly Deep Feedforward Networks (DFNs) or Multilayer Perceptrons (MLPs), which are the core of many deep learning models. These models are called feedforward because information flows through the function being evaluated from input x, through the computations in which we add weights to the input and bias, used to define function f and finally to the output y. There are no feedback connections where outputs loop back into the network, meaning the output of each neuron does not affect the neuron itself. This network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that results in the best function approximation.

When feedforward, NNs are extended to include feedback connections back into the network, they are called *Recurrent Neural Networks*.

3.2.1.3.1 Architecture

As stated in [1] the DL focuses on gradually building up different levels of understanding, each layer diving deeper into the underlying patterns present in the data, with the number of layers determining the depth of the model. Each layer processes information and passes it to the next layer, creating a stack of layers that work together to understand complex data. The simplest graphical representation is shown in figure 3.15.

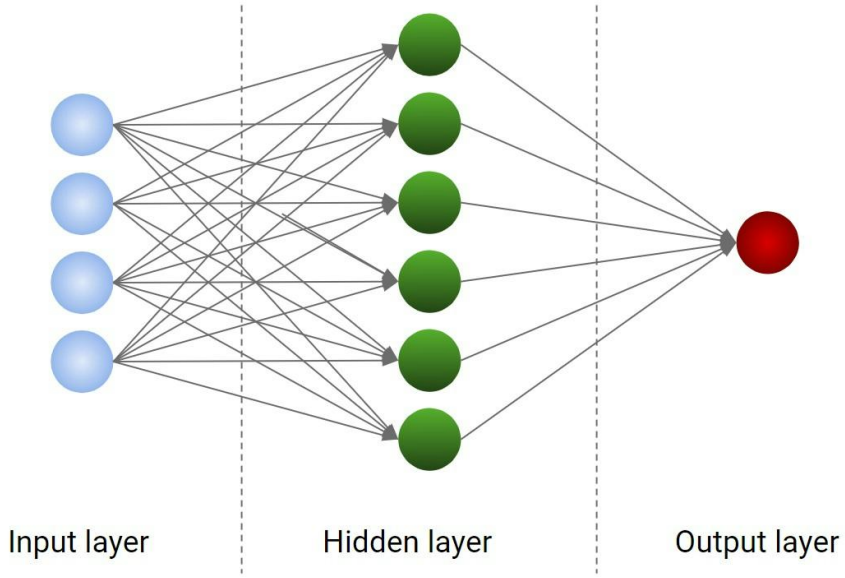

Figure 3.15: MLP Layers

The *Input layer* is the first layer of the multilayer perceptron. It serves as the entry point for the data into the network. Each neuron in this layer represents one feature of the input data, and it simply passes the data to the following layers without applying any transformations. The number of neurons in the input layer corresponds to the dimensionality of the input data.

Hidden layers are layers between the input and output layers. These layers consist of neurons that apply transformations to the input data by performing weighted sums followed by mentioned activation functions and finally passes it on the next layer. The purpose of the hidden layers is to enable the network to learn complex patterns and representations in the data.

The *Output layer* is the final layer of the MLP, responsible for producing the model's predictions. The number of neurons in this layer corresponds to the number of classes in a classification problem or the number of outputs in a regression problem. The output from this layer is typically passed through an activation function, to generate the final output probabilities or values.

3.2.1.3.2 Learning Process

Since the input data is parameterized by its weights, learning in NNs can be defined as finding a set of values for the weights of all layers in a network, such that the network will correctly assign optimal values to the input data and filter out the information that is crucial for making the final output decision. First, it is needed to compute how big the gap is between the network output and true output in order to adjust the values of the weights. This is accomplished by using the *loss function* of the network, also called the objective function or a cost function. The loss function is used as a feedback to adjust the value of the weights and biases, in a direction that will *minimize* the loss score. This adjustment is the task of the optimizer, which implements the *Backpropagation algorithm*: the central algorithm in DL. This iterative algorithm, during every epoch, adapts the weights and biases to minimize the loss by moving down toward the gradient of the error. Some of the most popular loss functions are MSE, MAE, Cross-Entropy Loss and Hinge Loss 3.16. These functions take true value y and a predicted value $\hat{y}$ as inputs.

1. **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2. **Mean Absolute Error (MAE)**:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

3. **Cross-Entropy Loss**:

$$\text{Cross-Entropy} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

4. **Hinge Loss**:

$$\text{Hinge Loss} = \sum_{i=1}^{n}\max(0, 1 - y_i \cdot \hat{y}_i)$$

Figure 3.16: Common Loss Functions

3.2.2 Convolutional Neural Networks

CNNs [12] are a specific type of NNs designed to efficiently process data that has a structured, grid-like arrangement, such as images. These networks are particularly well-suited for tasks involving image recognition and classification due to their ability to automatically and adaptively learn spatial hierarchies of features from input data. The term *convolutional* refers to mathematical operation convolution, performed within these networks. Unlike general NNs that rely on matrix multiplication to process data, CNNs replace this operation with convolution in at least one of their layers. This substitution is significant because convolution enables the network to focus on local patterns in the data, such as edges, textures, or more complex features as it moves deeper into the layers, mirroring how humans often recognize objects, first by identifying basic shapes and then understanding them as parts of a whole.

The CNNs architecture includes several building blocks, mentioned in [16], such as *convolution layers*, *pooling layers* and *fully connected layers*.

3.2.2.1 Convolution layer

Convolution layer plays a crucial role in feature extraction, which is the process of identifying important patterns and details in the input data. This layer combines both linear and nonlinear operations, specifically through convolution and activation functions, to transform and analyze the input data.

First, the input image is converted into pixels, where each pixel is represented by a numerical value between 0 and 255, indicating its brightness or darkness as shown in figure 3.17.

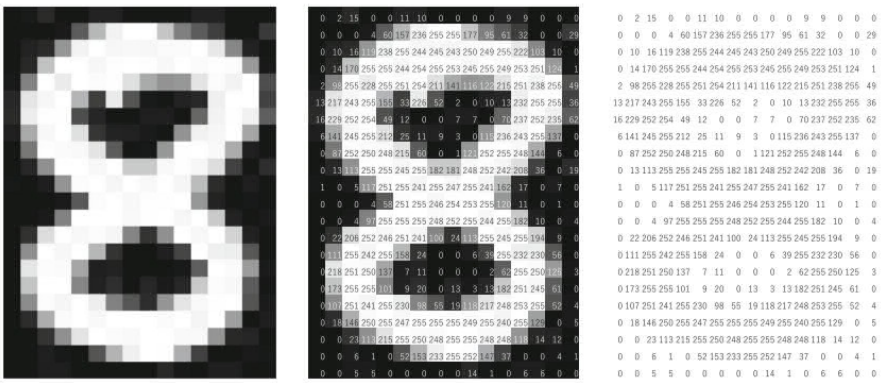

Figure 3.17: Example of pixel representation [16].

In mathematical terms, convolution is an operation that combines two functions to produce a third function, representing how the shape of one is modified by the

other:

$$h(t) = (f * g)(t) = \int f(\tau) \cdot g(t - \tau)\, d\tau$$

In CNNs, convolution is used to extract features from this grid of numbers (the pixel values in our case). It involves using a small matrix of numbers called a *kernel* or *filter*, which is applied across the entire image. The kernel slides over the grid, and at each position, it performs an element-wise multiplication with the corresponding pixels, followed by summing these products to produce a single value. This value represents a specific feature detected at that position, such as an edge or a texture. The resulting grid of these values forms what is known as a *feature map*, presented in 3.18.

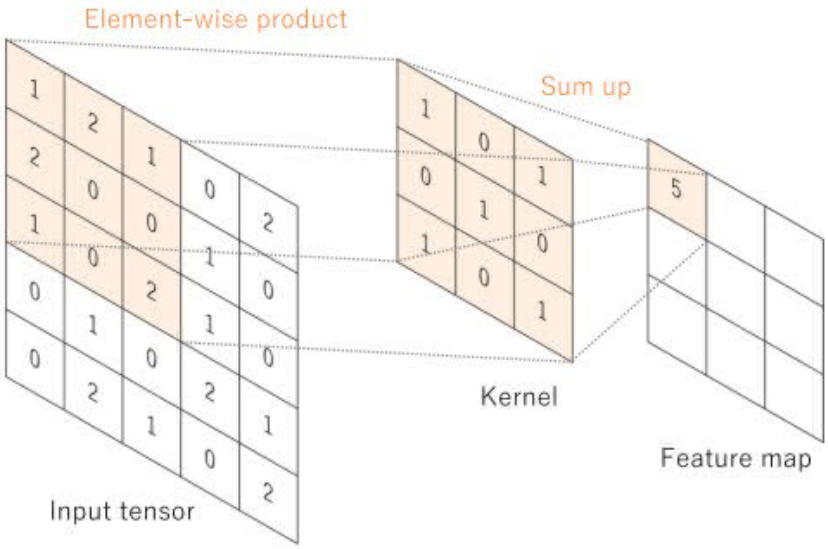

Figure 3.18: An example of convolution operation with a kernel size of 3x3, no padding [16]

Multiple kernels can be applied to the same input image to generate several feature maps, each highlighting different aspects of the image. Different kernels act as different feature extractors, allowing the network to capture various characteristics of the input data. Two critical hyperparameters that define this convolution process are the size of the kernels and the number of kernels used. The kernel size determines the area of the image that is analyzed at each step, while the number of kernels dictates how many different features the layer will attempt to extract.

Feature maps are passed through nonlinear activation function, usually $ReLU$, to introduce nonlinearity into the network.

The center of each kernel typically cannot overlap the outermost elements of the input tensor, which leads to a reduction in the height and width of the output

feature map compared to the input tensor. In [17] it is shown that the borders and corners of an image can have a substantial impact on model performance and so *padding* should be introduced. Padding involves adding extra data around the borders of an input image, before performing the convolution, enabling the original borders to be considered. Traditional padding techniques include replication padding, reflection padding and the most popular zero padding (figure 3.19), where rows and columns of zeros are added on each side of the input tensor, as to fit the center of a kernel on the outermost element and keep the same in-plane dimension through the convolution operation. The importance of padding is explained in [18] where the authors introduce the Padding Module, which can optimize itself without requiring or influencing the model's entire loss function.

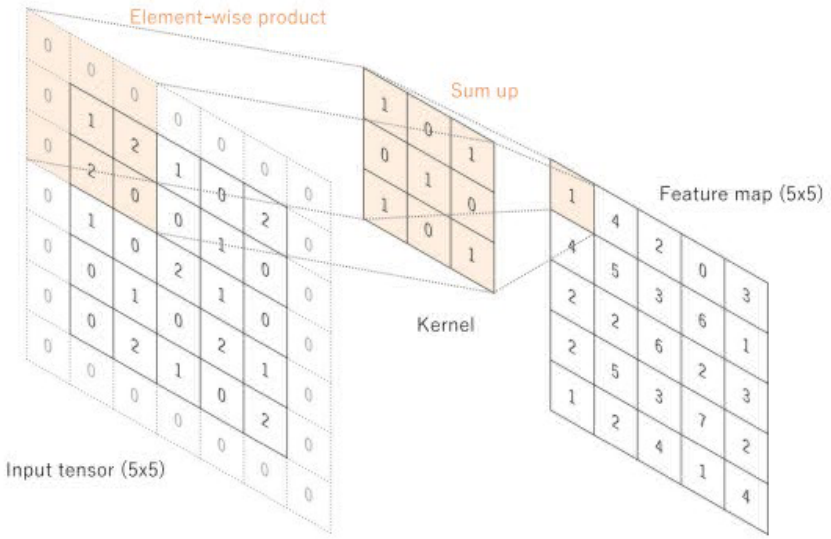

Figure 3.19: An example of convolution operation with zero padding [16]

3.2.2.2 Pooling layer

The resulting feature maps are typically processed by pooling layers. Pooling serves to downsample the feature maps, which helps in reducing their spatial dimensions while retaining essential information by focusing on small grid regions within the feature maps and produces a single value for each region. This value is usually derived using one of two common methods:

1. **Max Pooling:** selects the maximum value from each grid region 3.20
2. **Average Pooling:** computes the average value of the elements in each grid region

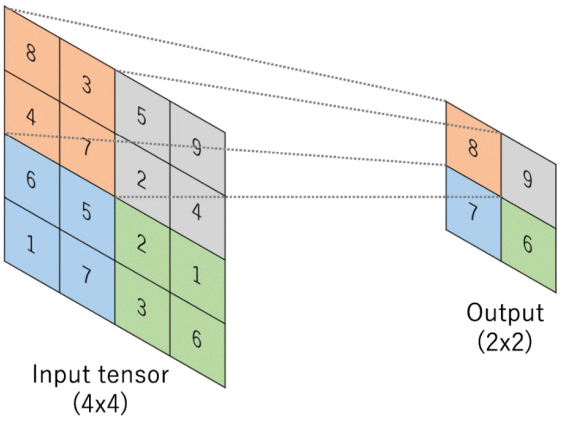

Figure 3.20: Max Pooling operation with filter size of 2x2 [16]

Pooling reduces the feature map size and introduces a degree of translational invariance to small shifts by summarizing each region into a single number. It means that small shifts or distortions in the input image result in minimal changes in the feature maps, helping the network become more robust to variations in the input, which brings it closer to generalization.

3.2.2.3 Fully Connected layer

Once the final convolution or pooling layer is reached, the common practice is to transform the output feature maps into vectorised feature maps and connect it to one or more fully connected layers, also known as dense layers, in which every input is connected to every output, adding learnable weight and passed through an activation function. By executing this final step, the extracted features are transformed to final outputs such as the probabilities for each class in classification tasks, having the same number of output nodes as the number of classes.

The process is captured in the figure 3.21

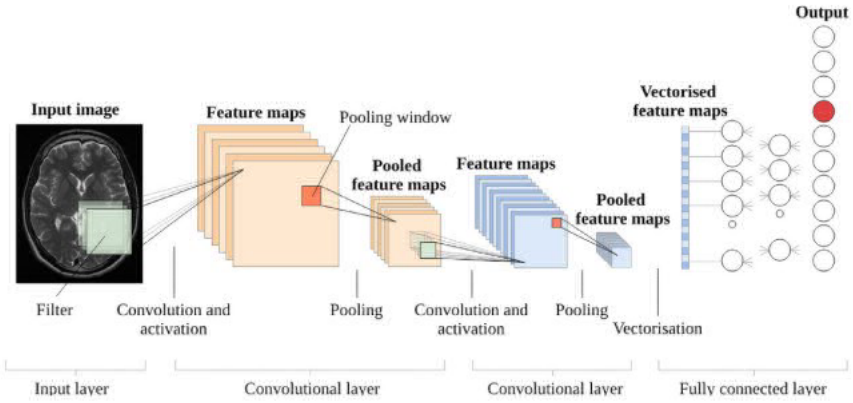

Figure 3.21: An overview of CNNs learning process [19]

Following the discussion on the fundamental principles of learning in CNNs, attention is now directed towards visualizing technique called Grad-CAM.

3.2.2.4 Gradient-weighted Class Activation Mapping (Grad-CAM)

Grad-CAM, proposed in [24], is introduced as a method for visualizing the importance of different parts of an image in the decision-making process of a model. The technique leverages feature maps with activation functions from CNNs layers to identify the regions of an image that most significantly contribute to the model's decision. During the backpropagation process, Grad-CAM uses gradients associated with a specific class to weight the feature maps with activation functions, thereby highlighting the critical regions of the image that influence the final decision, pesented in 3.22.

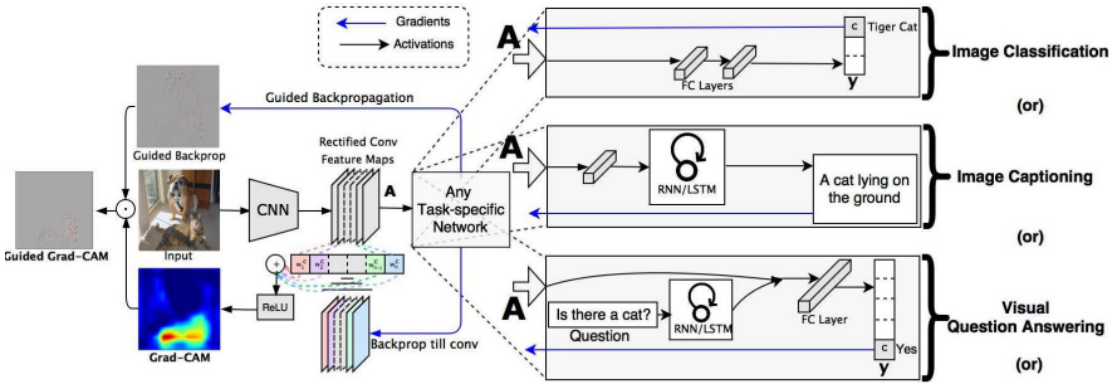

Figure 3.22: Grad-CAM overview: Given an image and a class of interest (e.g. 'tiger cat' or any other type of differentiable output) as input, we forward propagate the image through the CNNs part of the model and then through task-specific computations to obtain a raw score for the category. The gradients are set to zero for all classes except the desired class (tiger cat), which is set to 1. This signal is then backpropagated to the rectified convolutional feature maps of interest, which we combine to compute the coarse Grad-CAM localization (blue heatmap) which represents where the model has to look to make the particular decision. Finally, we pointwise multiply the heatmap with guided backpropagation to get Guided Grad-CAM visualizations which are both high-resolution and concept-specific [24]

This method has shown particular utility in the analysis of medical images, as demonstrated in [6]. In this study, Grad-CAM was employed to identify crucial regions on chest X-rays for the detection of pneumonia which is essential in medical applications where algorithm transparency is vital for clinical deployment.

Building upon the foundational concepts of CNNs discussed earlier, this section delves into the architecture and application of CNNs used for image recognition tasks: Residual Networks.

3.2.2.5 Residual Neural Networks (ResNets)

Residual Networks, commonly known as ResNet [8], are a type of CNNs architecture introduced to address the challenges associated with training very DNNs which present significant challenges, often making it difficult to optimize them effectively.

The architecture are mainly inspired by the philosophy of VGG nets, but instead of directly learning the desired underlying function at each layer, this architecture introduced the concept called Residual Blocks.

ResNet layers learn the residual function, which is the difference between the input to a layer and the desired output.

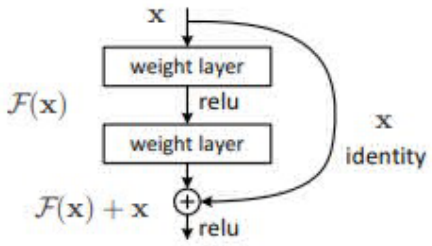

Figure 3.23: ResNet Structure [8]

Mathematically, if the desired output is $H(x)$, ResNet layers learn the function $F(x) = H(x) - x$. The original mapping then becomes $H(x) = F(x) + x$.

The core innovation in ResNet is the use of skip (or shortcut) connections, which bypass one or more layers by directly connecting the input of a block to its output. This allows the network to carry forward information from earlier layers, making it easier to train deeper networks.

Experiments on the ImageNet dataset showcase the effectiveness of this method, where authors successfully trained ResNets with up to 152 layers making them eight times deeper than the popular VGG networks, yet with lower computational complexity. ResNet achieved a 3.57% error rate on the ImageNet test set, securing first place in the ILSVRC 2015 classification challenge. The primary variants of ResNet include ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152 with the number in the name indicating the depth of the network(i.e. the number of layers). ResNet18 and ResNet34 are relatively simpe networks with 18 and 34 layers, respectively. They use basic residual blocks that consist of two convolutional layers, making them less complex and faster to train. ResNet50, ResNet101 and ResNet152 are deeper networks with 50, 101 and 152 layers, respectively. The complexity of their design allows them to have greater representational power, making them capable of capturing more intricate patterns in the data, but at the cost of increased computational complexity and training time. The key difference between these models lies in their depth and complexity, with deeper models generally offering better performance on complex tasks at the cost of higher computational demands.

3.3 Performance Metrics

Since this is a binary classification task, where the labels are represented as 0 and 1, the model's predictions can be categorized into four possible outcomes [23]:

1. **True Positive (TP):** the model correctly predicts a label 1-visible fracture when the actual label is 1-fracture is visible.
2. **True Negative (TN):** the model correctly predicts a label 0-nonvisible fracture when the actual label is 0-there is no fracture.
3. **False Positive (FP):** the model incorrectly predicts a label 1-visible fracture when the actual label is 0-there is no fracture present.
4. **False Negative (FN):** the model incorrectly predicts a label 0-no fracture when the actual label 1-the fracture is present.

False negatives, known as a "Type II error", are typically the most dangerous among these outcomes, especially in medical diagnostics, as they represent cases where a condition or disease is present but it is not detected by the model. On the other hand, false positives, known as a "Type I error", are often less dangerous but still undesirable, as they lead to the incorrect identification of a condition when it is not present, resulting in unnecessary stress and treatment.

The following metrics provide a comprehensive evaluation of the model's performance, offering insights into its ability to correctly classify both positive and negative instances using the above-mentioned terms: Accuracy, Sensitivity, Specificity, Precision and F1 Score.

Accuracy measures the overall correctness of the model by calculating the proportion of true results (both TP and TN) among the total number of cases [23]:

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

Sensitivity assesses the model's ability to correctly identify positive instances. It is calculated as the proportion of true positives out of the total actual positives:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Specificity measures the model's ability to correctly identify negative instances. It is the proportion of true negatives out of the total actual negatives:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Precision evaluates how many of the predicted positive instances were actually correct. It is calculated as the proportion of true positives out of all predicted positives:

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

F1 Score provides a balance between Precision and Sensitivity (Recall). It is particularly useful when there is an uneven class distribution, as it considers both false positives and false negatives:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$$

4 Results and Discussion

The results of the experiments are presented using CNNs. The experiments evaluate the performance of three ResNet architectures: ResNet-18, ResNet-50 and ResNet-101.

The first convolutional layer was modified to adapt ResNet models for grayscale images with a single channel. The original weights of the layer, which were designed for three-channel RGB images, were averaged across the channel dimension to create weights suitable for a single-channel input. The modified convolutional layer retains the same kernel size, stride and padding as the original, ensuring compatibility with the rest of the network.

Modifying only the first layer is sufficient for this adaptation, as the subsequent layers in the network operate on feature maps produced by the first layer, not directly on the input image.

Figure 4.1, presents the loss values on both the training and validation sets for three different ResNet architectures across all 20 epochs. The solid lines represent the training loss for each model, while the dashed lines represent the validation loss.

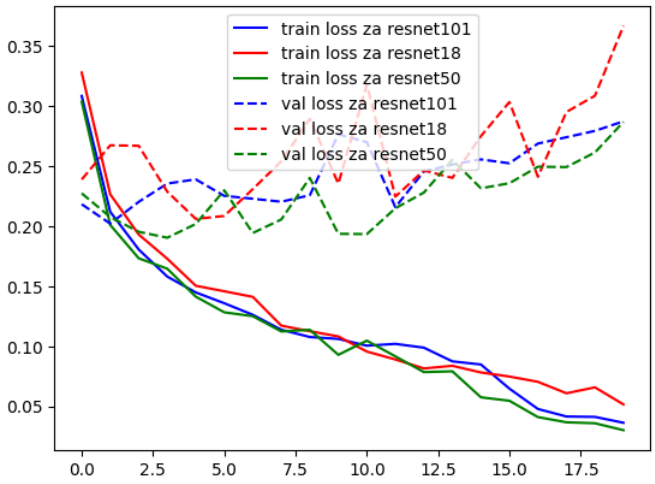

Figure 4.1: Training Loss vs Validation Loss over 20 epochs

All three models successfully learned the training data, as indicated by the steadily decreasing training loss curves. However, in terms of validation performance, ResNet50 achieves the lowest final validation loss (0.2867) and the highest validation accuracy (93.53%). ResNet101 follows closely with a similar validation loss (0.2872) and accuracy (93.01%). In contrast, ResNet18 has a noticeably

higher final validation loss (0.3667) and a slightly lower accuracy (92.29%), suggesting that it may not generalize as effectively as the other models.

The fluctuations in the validation loss curves, especially for ResNet18, suggest potential overfitting or sensitivity to the validation dataset, which could be linked to the lower complexity of the ResNet18 architecture compared to the deeper ResNet50 and ResNet101 models.

All training results, from figure 4.1 and figure 4.2, have been summarized in table 4.1 for better clarity and readability.

Model	**Train loss**	**Train Accuracy**	**Validation loss**	**Validation Accuracy**
ResNet101	0.0363	98.75%	0.2872	93.01%
ResNet18	0.0515	98.09%	0.3667	92.29%
ResNet50	0.0301	99.01%	0.2867	93.53%

Table 4.1: Training and Validation metrics at Epoch 20 for different models.

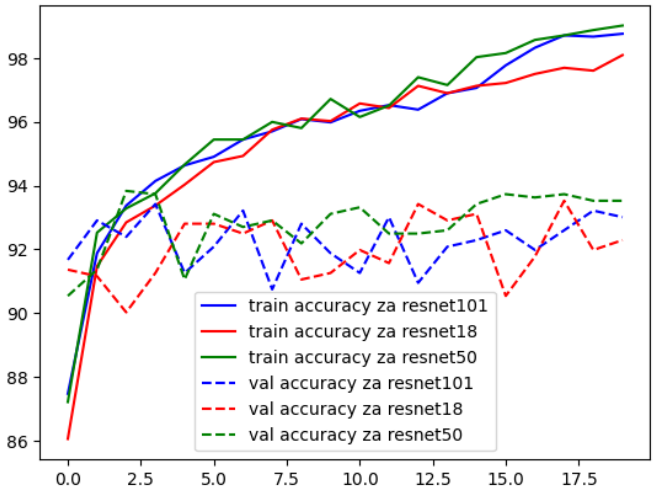

Figure 4.2: Training Accurracy vs Validation Accuracy over 20 epochs

Figure 4.2 presents the training and validation accuracies. All three models reached high training accuracy, with ResNet50 attaining the highest (99.01%), followed closely by ResNet101 (98.75%) and ResNet18 (98.09%). ResNet50 consistently maintains the highest validation accuracy, indicating its superior generalization to unseen data.

Confusion matrices for each model are presented in figure 4.3, figure 4.4 and figure 4.5.

ResNet50 demonstrated the best performance in avoiding the most dangerous "Type II error"-the false negatives (incorrectly classifying images with visible

fractures as no fracture visible label), with 33 such errors, compared to 46 for ResNet101 and 58 for ResNet18. All three matrices indicate that the models classify "no fracture" label well, with less than 20 misclassifications each.

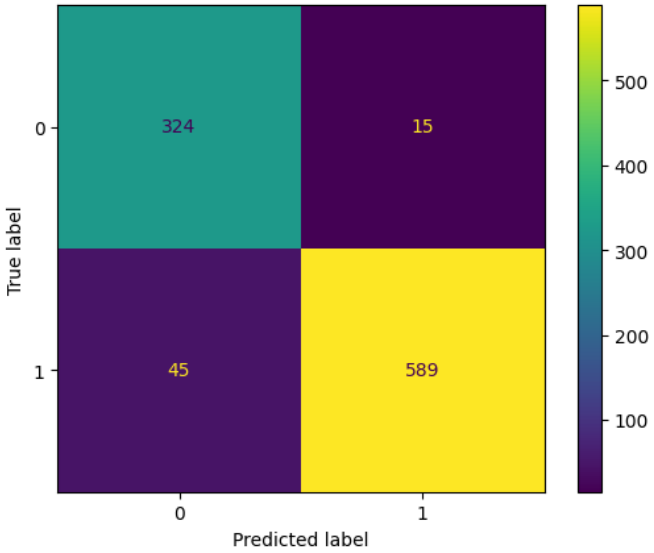

Figure 4.3: Confusion Matrix of Resnet 101

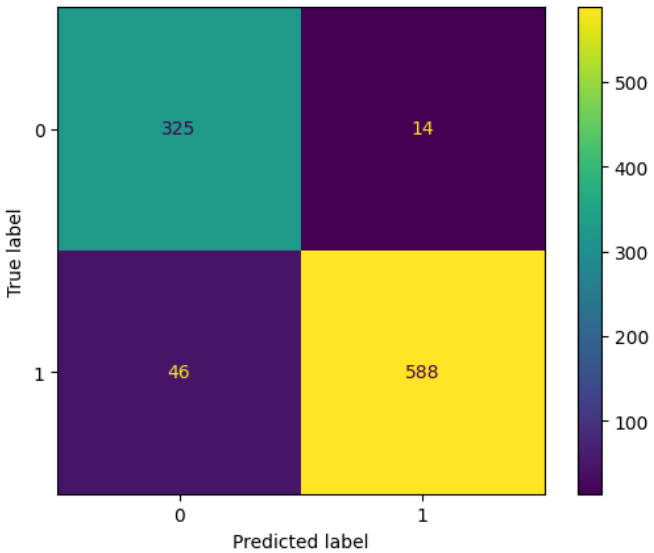

Figure 4.4: Confusion Matrix of Resnet 18

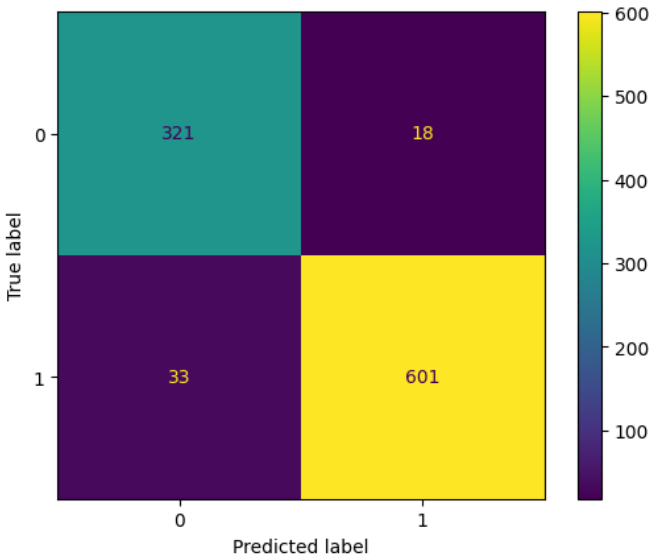

Figure 4.5: Confusion Matrix of Resnet 50

The following images are presented for better understanding of the visual patterns that contributed to both correct and incorrect model predictions.

Figure 4.6: X-ray images where none of the models predicted visible fracture while the fracture was visible

Figure 4.7: X-ray images where all the models predicted visible fracture while the fracture was visible

In both images in figure 4.6, all three models failed to predict the presence of a fracture, even though a fracture was present. It suggests that the indicative features fractures in this X-rays were either subtle or atypical, leading to a misclassification by the models. The misalignment or unusual presentation of the fracture could have contributed to this oversight. In contrast, as shown in figure 4.7, a scenario is presented where all three models correctly identified the presence of a fracture. It indicates that the features of the fracture in this X-rays were clear and aligned well with the patterns the models were trained to recognize. The fracture might have been more prominent, with clear discontinuity or displacement of a bone thus easily detectable by the models. The consistent success across all three

models indicates that the fracture characteristics in images are well-represented in the training data, leading to accurate predictions.

Subtle or complex fractures, as the one shown in figure 4.6, present a greater challenge for CNNs models, potentially due to a lack of similar examples in the training set. On the other hand, fractures that are more noticable, are easier for the models to detect, leading to consistent and accurate predictions.

Figure 4.8 presents heatmaps of five images generated using Grad-CAM, in order to gain a deeper understanding on how the CNNs models made its decision. These images illustrate how the ResNet18, the least complex CNN architecture in this study, detected the fracture across its layers.

In the first three rows, the model successfully identified the fracture, as indicated by the concentrated attention in the relevant areas of the images. In contrast, the last two rows present cases where the model did not detect the fracture, resulting in a broader and less focused distribution of attention. By examining the visualizations from figure 4.8, it becomes possible to observe which regions of the image were most influential in the model's decision-making process. The progression through the layers reveals how the model shifts its focus from low-level features, such as edges and textures, to higher-level, more abstract features, ultimately leading to the identification of the fracture. This visualization provides valuable insight into the internal workings of the model, highlighting the specific image regions that contributed to the final prediction.

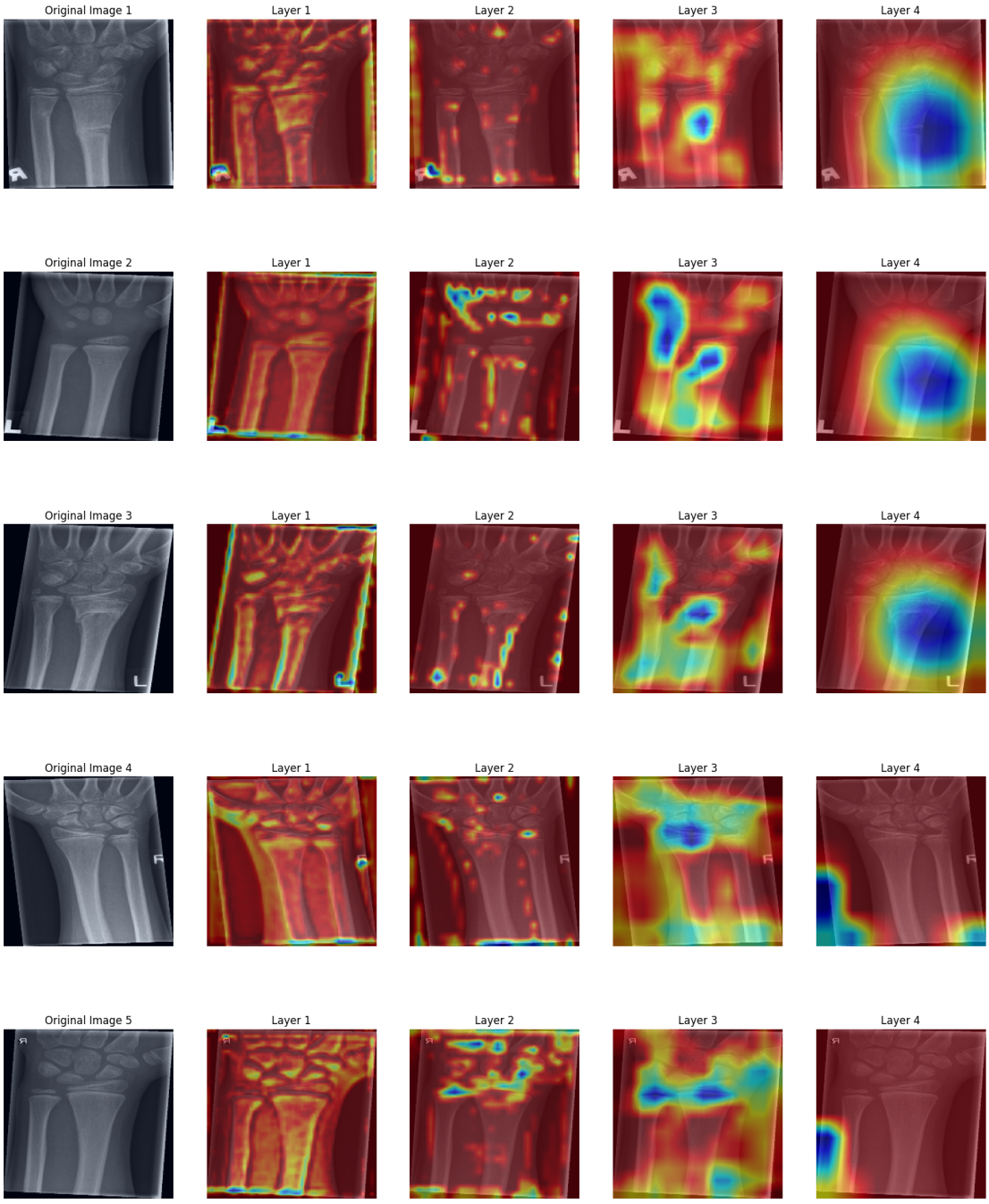

Figure 4.8: Heatmap visualizations of five images across different layers of ResNet18

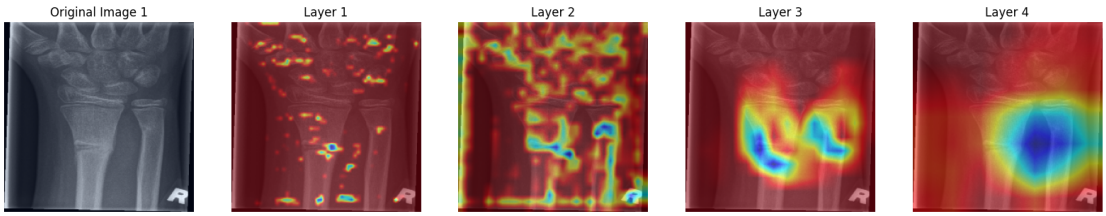

Figure 4.9: Heatmap visualizations across different layers of ResNet101

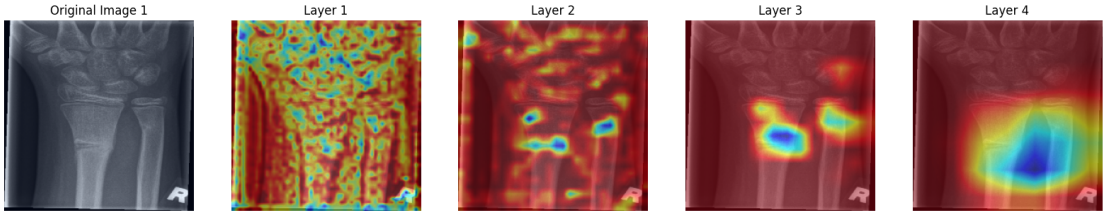

Figure 4.10: Heatmap visualizations across different layers of ResNet50

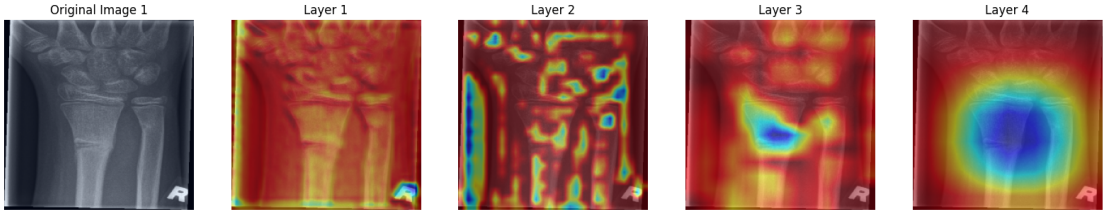

Figure 4.11: Heatmap visualizations across different layers of ResNet18

The heatmap visualizations for ResNet101 (figure 4.9), ResNet50 (figure 4.10) and ResNet18 (figure 4.11) demonstrate how each model processes and emphasizes different regions of the X-ray image across various layers. The last convolutional layer in each of the four main blocks was used in all three models to generate these visualizations.

In the first layers, all three models show scattered attention across the image, focusing on basic structures like edges and textures. As the models move to deeper layers, their attention becomes more focused on complex patterns, particularly those related to fractures. ResNet101, with its deeper architecture, begins to refine its focus earlier than ResNet50 and ResNet18.

By the third layer, both ResNet101 and ResNet50 show clear attention on the critical areas of the image that likely indicate fractures, with ResNet18 also identifying these regions, but with slightly less precision.

In the final, deepest layer, ResNet101 exhibits the most concentrated and accurate focus, capturing more abstract, high-level features. ResNet50 also shows strong focus, though slightly less pronounced, while ResNet18, while still effective, shows a broader and less sharp focus.

Presentation of training, validation and testing results of the models are shown in the tables bellow.

Model	**Train Accuracy**	**Validation Accuracy**	**Test Accuracy**
Resnet101	98.75%	93.01%	93.83%
Resnet18	98.09%	92.29%	93.83%
Resnet50	99.01%	93.53%	94.76%

Table 4.2: Models Accuracy on Train, Validation and Test set

ResNet50 achieves the highest test accuracy of 94.76%, as presented in table 4.2. ResNet101 and ResNet18 both achieve a test accuracy of 93.83%, slightly lower than ResNet50, suggesting that while they perform well, in this task, they are not as effective as ResNet50.

Model	**Accuracy for class fracture**	**Accuracy for class no fracture**
Resnet101	92.90%	95.58%
Resnet18	92.74%	95.87%
Resnet50	94.79%	94.69%

Table 4.3: Accuracy of class fracture visible and no fracture visible for different models

Table 4.3 offers more detailed insights into the models' performance on detecting specific classes. For the "fracture" class, ResNet50 again demonstrates the best performance, with an accuracy of 94.79%, outperforming ResNet101 (92.90%) and ResNet18 (92.74%). For the "no fracture" class, ResNet18 slightly edges out the others, achieving an accuracy of 95.87%, followed closely by ResNet101 at 95.58% and ResNet50 at 94.69%. Table 4.4 present the models' overall performance using metrics such as accuracy, sensitivity, specificity, precision and F1 Score.

Results presented in table 4.4, showcase that all three models demonstrate high performance across various metrics, with ResNet50 showing a slight advantage. Due to an uneven class distribution, F1 Score provides the most significant performance metric, where ResNet50 achieves the highest value of 95.93%, indicating its superior balance between precision (97.09%) and sensitivity (94.79%).

Model	**TN**	**FP**	**FN**	**TP**	**Acc**	**Sens**	**Spec**	**Prec**	**F1 Score**
ResNet101	324	15	45	589	93.83%	92.90%	95.58%	97.52%	95.15%
ResNet18	325	14	46	588	93.83%	92.74%	95.87%	97.67%	95.15%
ResNet50	321	18	33	601	94.76%	94.79%	94.69%	97.09%	95.93%

Table 4.4: Performance metrics of different ResNet models on the 9,730 image dataset

An additional table is presented with results using images from only the first folder from [5], for comparison. Originally this folder contained around 5,000 images. However after applying the same selection criteria used in the main training process (utilizing only projection 1 and excluding images with visible metal implants) a total of 2,407 images were used.

The resulting performance metrics are as follows:

Model	**TN**	**FP**	**FN**	**TP**	**Acc**	**Sens**	**Spec**	**Prec**	**F1 Score**
ResNet101	74	8	32	126	83.33%	79.75%	90.24%	94.03%	86.30%
ResNet18	73	9	15	143	90.00%	90.51%	89.02%	94.08%	92.26%
ResNet50	78	4	28	130	86.67%	82.28%	95.12%	97.01%	89.04%

Table 4.5: Performance metrics of different ResNet models on the 2,407 image dataset

The table 4.5 shows that ResNet18 achieved the highest F1 Score of 92.26% on this smaller dataset. ResNet50 followed with an F1 Score of 89.04%, while ResNet101 obtained an F1 Score of 86.30%. These results emphasize the role of dataset size in determining the optimal model architecture for a given task. In such cases, simpler architectures, such as ResNet18, are able to achieve superior performance on smaller datasets, likely due to their lower complexity. On the other hand, more complex models like ResNet50 and ResNet101 perform better with larger datasets, as they take advantage of the additional data to fully leverage their deeper layers and extract more detailed features.

5 Summary

In today's rapidly advancing medicine, Artificial Intelligence (AI) is becoming an indispensable tool for enhancing diagnostic accuracy and efficiency. The application of AI, particularly through Convolutional Neural Networks (CNNs), holds immense potential for transforming patient care by assisting doctors in making swift, precise decisions.

This master's thesis delves into the critical role of AI in medicine, focusing on the use of Neural Networks to recognize pediatric wrist joint fractures. The primary objective was to enable these networks to accurately identify and categorize unseen images - 0 for no visible fracture and 1 for visible fracture, thereby improving diagnostic processes and outcomes in pediatric care.

The dataset used was challenging, as it consisted of pediatric X-ray images of children aged 2 to 18 years. Due to the variations within this age group, careful and extensive parameter tuning was necessary to ensure that the models could generalize well across different cases.

The CNNs selected for this task were ResNet101, ResNet50 and ResNet18.

The models were trained over 20 epochs, during which a subset of 9,730 images out of the available 20,000 was utilized. This selection was made by applying only projection 1 and excluding images that displayed any metal implants to maximize precision and accuracy.

When trained on the dataset of 9,730 images, all models showed promising results. ResNet50 showed the best performance with an F1 Score of 95.93%, followed by ResNet101 and ResNet18 with F1 Scores of 95.15%

The results obtained from a smaller dataset of 2,407 images, resulted in slightly different performance metrics, with the F1 Score being the most telling metric.

In the smaller dataset, ResNet18 achieved the highest F1 Score of 92.26%, suggesting it is more robust when dealing with limited data due to its simpler architecture. In contrast, ResNet50 and ResNet101 demonstrated lower F1 Scores of 89.04% and 86.30% respectively, on the smaller dataset.

These results highlight the importance of dataset size in selecting the appropriate model. Simpler architectures like ResNet18 perform well on smaller datasets, while more complex models like ResNet50 and ResNet101 demonstrate superior results when more data is available. Additionally, the improved performance of all models on the larger dataset indicates that, with appropriate training, more data generally leads to better results.

Future research could benefit from incorporating age-specific data selection, considering that the appearance of children's joints varies significantly across different age groups.

Bibliography

[1] Francois Chollet, *Deep Learning with Python*, Manning Publications Co., 2018.

[2] Jean-Paul Haton, *A brief introduction to artificial intelligence*, LORIA-Université Henri-Poincaré, Nancy 1 Institut Universitaire de France, 2006.

[3] Gunnar Ratsch, *A Brief Introduction into Machine Learning*, Friedrich Miescher Laboratory of the Max Planck Society, Spemannstraße 37, 72076 Tubingen, Germany.

[4] Haohan Wang, Bhiksha Raj, *On the Origin of Deep Learning*, Language Technologies Institute, School of Computer Science, Carnegie Mellon University.

[5] Eszter Nagy, Michael Janisch, Franko Hržić, Erich Sorantin, Sebastian Tschauner, *A pediatric wrist trauma X-ray dataset (GRAZPEDWRI-DX) for machine learning.*

[6] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, Andrew Y. Ng, *CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning*, 2017.

[7] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, Sebastian Thrun, *Dermatologist-level classification of skin cancer with deep neural networks*, Nature, 2017.

[8] He K, Zhang X, Ren S, Sun J, *Deep residual learning for image recognition*, In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.

[9] Usharani Bhimavarapu, Nalini Chintalapudi and Gopi Battineni, *Automatic Detection and Classification of Diabetic Retinopathy Using the Improved Pooling Function in the Convolution Neural Network.*

[10] Arpit Kumar Sharma, Amita Nandal, Arvind Dhaka, *Brain tumor classification using the modified ResNet50 model based on transfer learning.*

[11] Aravind Sasidharan Pillai, *Multi-Label Chest X-Ray Classification via Deep Learning*, University of Illinois Urbana-Champaign, Champaign, IL, USA.

[12] Goodfellow I., Bengio Y., Courville A., *Deep Learning*, MIT Press, 2016.

[13] Jan Kukačka, Vladimir Golkov and Daniel Cremers, *Regularization for Deep Learning: A Taxonomy*, Computer Vision Group, Department of Informatics, Technical University of Munich, 2017.

[14] Shiv Ram Dubey, Satish Kumar Singh, Bidyut Baran Chaudhuri, *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*, Computer Vision and Biometrics Laboratory, Indian Institute of Information Technology, Allahabad, India.

[15] Alexander Amini, *Introduction to Deep Learning*, MIT 2022.

[16] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, Kaori Togashi, *Convolutional neural networks: an overview and application in radiology.*

[17] C. Innamorati, T. Ritschel, T. Weyrich and N. J. Mitra, *Learning on the edge: Explicit boundary handling in CNNs*, 2018.

[18] Fahad Alrasheedi, Xin Zhong, Pei-Chi Huang, *Padding Module: Learning the Padding in Deep Neural Networks*, Department of Computer Science, University of Nebraska at Omaha, Omaha, NE 68182, USA.

[19] Lundervold A., Lundervold A.S., *An overview of deep learning in medical imaging focusing on MRI*, 2018.

[20] Juan R. Terven, Edgar A. Chavez-Urbiola, *Loss Functions and Metrics in Deep Learning*, CICATA-Qro, Instituto Politecnico Nacional, Mexico, 2024.

[21] Dejun Zhang, Fuquan Ren, Yushuang Li, Lei Na, Yue Ma , *Pneumonia Detection from Chest X-ray Images Based on Convolutional Neural Network*, 2021.

[22] Brian Mwandau, *Investigating Keystroke Dynamics as a Two-Factor Biometric Security*, Strathmore University.

[23] Bradley J. Erickson, Felipe Kitamura, *Performance Metrics for Machine Learning Models.*

[24] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization.*

Biography

Milica Papić, born on February 24, 1998, in Novi Sad, Serbia, is a Database Engineer in making with a passion for growth and skill development. Milica has dedicated herself to the field of Information Technology, carving out a successful career path and contributing significantly to various projects. Milica's academic journey is marked by excellence. Her commitment to education is evident in her ongoing pursuit of a Master of Science in Applied Mathematics: Data Science, at the Faculty of Sciences, Department of Mathematics, Novi Sad, Serbia. In 2023, she embarked on an Advanced Database Course at the Faculty of Technical Sciences, Department of Informatics, Novi Sad, Serbia, achieving an outstanding grade of 10.

As a Bachelor in Applied Mathematics, Milica specialized in Mathematics in Finance during her undergraduate studies from 2016 to 2021, also at the Faculty of Sciences. Since October 2023, Milica has been employed as a Database Engineer at Navigator d.o.o, Novi Sad, Serbia. In this role, she has been showcasing her expertise in database management, particularly in SQL and PL/SQL.

In summary, Milica Papić is a dedicated and skilled professional, combining a strong academic background with practical experience in the dynamic field of database engineering and data science. Her commitment to personal and professional growth, coupled with a passion for making a positive impact, sets her apart as a valuable asset in the realm of Information Technology.

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj:
RBR
Identifikacioni broj:
IBR
Tip dokumentacije: monografska dokumentacija
BF
Tip zapisa: tekstualni štampani materijal
TZ
Vrsta rada: Master rad
VR
Autor: Milica Papić
AU
Mentor: dr Oskar Marko
MN
Naslov rada: Prepoznavanje preloma zglobova kod dece koristeći konvolucione neuronske mreže
NR
Jezik publikacije: engleski
JP
Jezik izvoda: engleski
JI
Zemlja publikovanjan: Republika Srbija
ZP
Uže geografsko područje: Vojvodina
UGP
Godina: 2024.
GO
Izdavač: autorski reprint
IZ
Mesto i adresa: Novi Sad, Trg Dositeja Obradovića 4
MA
Fizički opis rada:5/54/24/4/35
(broj poglavlja/broj strana /lit. citata/tabela/grafikona)
FO
Naučna oblast: matematika
NO
Naučna disciplina: primenjena matematika

ND
Predmetna odrednica/Ključne reči: konvolucija, neuralne mreže, resnet
PO
UDK:
Čuva se: u biblioteci Departmana za matematiku i informatiku, Prirodno-matematičkog fakulteta, u Novom Sadu
ČU
Važna napomena:
VN
Izvod:
U poslednjih nekoliko godina, inovacije u oblasti veštačke inteligencije (AI) značajno su uticale na različite sfere, a zdravstveni sektor je jedan od najvećih korisnika ovih tehnologija. Brzi napredak u AI, posebno u oblasti dubokog učenja (DL), otvorio je nove mogućnosti za poboljšanje tačnosti i efikasnosti u dijagnostici. Ovaj rad se fokusira na dijagnostičku primenu konvolucionih neuronskih mreža (CNNs) za otkrivanje preloma zgloba šake kod dece. Cilj je obuka mreže na rendgenskim snimcima kako bi oni mogli da daju tačne predikcije na prethodno neviđenim slikama, odnosno da odrede da li pacijent ima vidljiv prelom ili ne, čime bi se unapredila dijagnostička preciznost. Korišćene su tri dobro poznate arhitekture CNN-a: ResNet18, ResNet50 i ResNet101. Na raspolaganju je ukupno 20.327 rendgenskih snimaka, od kojih je 9.730 upotrebljeno, a preuzeti su sa odeljenja dečije hirurgije univerzitetske bolnice u Gracu, Austriji. Pored snimaka, dostupni su i podaci o pacijentima, koji uključuju kolonu koja povezuje svaku sliku sa odgovarajućom oznakom, gde 1 označava prisutan prelom, a 0 odsustvo preloma. Snimci i njihovi odgovarajući opisi koriste se kao ulaz za obučavanje mreže, kako bi mogao tačno da klasifikuje prisustvo ili odsustvo preloma. Ove slike i oznake koriste se kao ulaz za razvoj mreže koja je sposobna da predvidi da li je prelom prisutan, oslanjajući se isključivo na vizuelne informacije sa slike. Mreža koja je pokazala najbolje rezultate u ovom istraživanju je ResNet50, postigavši tačnost od 94,76% i F1 Score od 95,93% na novim slikama, čime je demonstrirana sposobnost generalizacije mreže. Ovo istraživanje ne samo da pokazuje potencijal CNN-a u automatizaciji i poboljšanju dijagnostičkog procesa, već pruža i komparativnu analizu performansi različitih mrežnih arhitektura sa različitim nivoima složenosti u zadacima medicinske obrade slika.
IZ
Datum prihvatanja teme od strane NN veća: 28.08.2024.
DP
Datum odbrane:
DO
Članovi komisije:

KO

Predsednik: dr Danijela Tešendić, vanredni profesor PMF-a u Novom Sadu, uža naučna oblast: Informacioni sistemi, izabrana 17.11.2020. godine

Mentor: dr Oskar Marko, naučni saradnik Instituta BioSens u Novom Sadu, uža naučna oblast: Elektronika, telekomunikacije i informacione tehnologije, izabran 22.01.2020. godine

Član: dr Nikša Jakovljević, vanredni profesor, Telekomunikacije i obrada signala, 11.10.2019, Fakultet tehničkih nauka u Novom Sadu

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES
KEY WORD DOCUMENTATION

Accession number:
ANO
Identification number:
INO
Document type: monograph type
DT
Type of record: printed text
TR
Contents code: Master thesis
CC
Author: Milica Papić
AU
Mentor: PhD Oskar Marko
MN
Title: Recognition of Pediatric Joint Fracture Using Convolutional Neural Networks
TI
Language of text: English
LT
Language of abstract: English
LA
Country of publication: Republic of Serbia
CP
Locality of publication: Vojvodina
LP
Publication year: 2024.
PY
Publisher: author's reprint
PU
Publ. place: Novi Sad, Trg Dositeja Obradovića 4
PP
Physical description:5/54/24/4/35
(chapters/pages/literature/tables/graphics)
PD
Scientific field: mathematics
SF
Scientific discipline: Applied mathematics

SD
Subject / Key words: convolution, neural network, resnet
SKW
UC:
Holding data: Department of Mathematics and Informatics' Library, Faculty of Sciences, Novi Sad
HD
Note:
N
Abstract: In recent years, innovations in Artificial Intelligence (AI) have profoundly impacted various domains, with healthcare being one of the most significant beneficiaries. The rapid advancements in AI, particularly in the field of Deep Learning (DL), have opened new opportunities for improving diagnostic accuracy and efficiency in medicine. This thesis focuses on the diagnostic application of Convolutional Neural Networks (CNNs) for detecting pediatric wrist joint fractures. The goal is to train and fine-tune several models on X-ray images in order to enable them to make accurate predictions on previously unseen images, specifically to determine whether a patient has a visible fracture or not. The aim is to enhance the model's ability to accurately identify the presence or absence of fractures, thereby improving diagnostic precision. Three well-established CNNs architectures were used: ResNet18, ResNet50 and ResNet101. There are 20,327 X-Ray images, 9,730 of which were used, sourced from the Department of Pediatric Surgery of the University Hospital in Gratz, Austria. They consist of both cases where fractures are visible and cases where no fracture is present. In addition to the images, a patient dataset is at our disposal. It contains patient information including a column that links each image to a corresponding label, where 1 indicates a visible fracture and 0 indicates the absence of a fracture. The images and their corresponding labels are fed into the CNNs models to train them to accurately classify the presence or absence of fractures. These images and labels are used as input to develop a model that is capable of predicting whether a fracture is present, based solely on the visual information in the image. The model that showed the best performance in this study is ResNet50, achieving an accuracy of 94,76% and F1 Score of 95,93% on the unseen images, demonstrating its strong ability to generalize. This research not only demonstrates the potential of CNNs in automating and improving the diagnostic process, but also provides a comparative analysis of the performance of different network architectures with varying complexities in medical imaging tasks.
AB
Accepted by the Scientific Board on: 28.08.2024.
ASB

Defended:
DE
Thesis defend board:
DB
Chair: Dr. Danijela Tešendić, Associate Professor at the Faculty of Sciences in Novi Sad, specialized field: Information Systems, elected on 17.11.2020.
Mentor: Dr. Oskar Marko, Research Associate at the BioSens Institute in Novi Sad, specialized field: Electronics, Telecommunications and Information Technology, elected on 22.01.2020.
Member: Dr. Nikša Jakovljević, Associate Professor, Telecommunications and Signal Processing, elected on 11.10.2019, Faculty of Technical Sciences in Novi Sad