

**College: Engineering and Information Technology**

**Department: Information Technology**

**Program: Data Analytics**

**Programing for Data Analytics II course Project**

**Pneumonia Image Classification**

**Prepared by:**

**Leena Alsalhi 202110613**

**Layan Ahmad 202110844**

**Haniyah Alzaben 202110616**

**Supervised by:**

**Dr. Salam Fraihat**

**Academic Year 2023- 2024 – Fall**

Image Classification using CNN

Leena Alsalhi   
*College of Engineering & IT  
Ajman University*Ajman, UAE  
202110613@ajmanuni.ac.ae

Layan Ahmad  
*College of Engineering & IT  
Ajman University*Ajman, UAE  
202110844@ajmanuni.ac.ae

Haniyah Alzaben *College of Engineering & IT  
Ajman University*Ajman, UAE  
202110616@ajmanuni.ac.ae

*Abstract*—When it comes to medical diagnosis, having the ability to quickly and accurately detect diseases could be a matter of life or death. Convolutional Neural Networks (CNNs) is one type of deep learning technology, which has proven impressive potential in this environment. The aim of our project is to classify X-ray images and identify if Pneumonia exists or not by using CNNs. We seek to enhance our model's accuracy and efficiency through an intricate process of data preprocessing, data augmentation, and thorough parameter tuning. We also conducted sensitivity analysis to encompass various critical parameters. The outcomes of this project demonstrate the important impact of CNNs in the field of medical image classification, highlighting the effect of parameter selection on model performance and effectiveness, as well as providing insightful information on optimizing Pneumonia diagnosis in medical imaging.

Keywords—CNN, pneumonia, Deep learning.

# Introduction

The use of deep learning techniques in the processing of medical images has radically altered the way we identify illnesses. Among these advancements, the adoption of Convolutional Neural Networks (CNNs) for medical image classification stands out as an effective tool. A common and possibly fatal lung disease, Pneumonia is a good candidate for this technical breakthrough. Traditional techniques for identifying this lung disease require experienced radiologists to visually assess X-ray images; this is a costly and human error-prone process. With CNNS ability to extract complex patterns and features from images, it provides an automated and potentially more accurate solution. The objective of our project is to apply CNNs' abilities for classifying X-ray images in relation to detecting Pneumonia. Our concentration goes beyond the simple use of CNNs. We go into the important field of sensitivity analysis, where we thoroughly examine how different parameters influence the performance of the model. The approach we're going to follow consists of a process that begins with a preparation of the data to ensure the dataset's consistency and quality. For the purpose of improving the model's capacity to generalize from training data, data augmentation methods are used. We also explore parameter optimization including batch size, regularization methods and weight initializers.

# Task and data

## Data Set Description

The X-ray pictures that were used in our project were divided into two classes: "pneumonia" and "normal." The data were obtained via Kaggle. Three folders were used to arrange the data: "train," "test," and "pred." A total of 7,315 images in the "train" folder , 3,883 of which were categorized as "pneumonia" and 3,432 as "normal". Of the 629 photos in the "test" folder, 234 are of normal condition and 386 are of Pneumonia. Nine more images for future model predictions were included in the "pred" folder. These were grayscale images of sizes that varied, requiring further data engineering techniques.We will also be evaluating the experiment with the accuracy metric as well as the confusion matrix to see precisely how many false positives and negatives our model’s going to produce.

## **Data Pre-processing:**

We have performed  a number of data pre-processing steps before training the CNN model. Initially, we observed that there were a few duplicate images. These were identified and removed to prevent redundancy and ensure data quality [3]. The following step is to divide the "train" data into training and validation sets, we split the "train" data into two groups: 80% of the set to be used for training and 20% of the set for validation Which resulted in 5,852 training images and 1,463 validation images. The splitting of the training set aids in evaluating, optimizing, and selecting the optimal model. It also helps in preventing overfitting and guarantee that the model works well on unseen images, which is essential for its practical application. Then we divided each pixel value by 255 to accomplish grayscale normalization, which aligned the intensity of the pixels range between 0 and 1. Grayscale normalization is applied to make sure that pixel values in grayscale images are scaled to a consistent range, assists in stabilizing training, improves gradient flow, assures compatibility with activation functions, and fosters consistency across datasets. In order to guarantee that the images matched the desired input shape for the model, they were reshaped to a consistent size of 150x150. This was required to achieve model compatibility as well as efficient training.

## Data Augmentation

When it comes to image preprocessing, data augmentation is a technique where we apply various transformations, including flips, rotations, and brightness adjustments to a dataset with the goal to enhance the variety of a dataset [3]. We can prevent overfitting as well as improve its ability to generalize to many shifts and scenarios in real-world problems.

We utilized the features of the ImageDataGenerator class to supplement data. This required using several kinds of transformations:

1. **Rotation** (rotation\_range = 10): determines the range, expressed in degrees, of possible random rotations that can be applied to the input images during the data augmentation phase.  setting the rotation range to 10 means that any angle between -10 and +10 degrees can be randomly applied to the images during the data augmentation process. This makes the model more flexible and able to handle various angles of the input data through the inclusion of variations in the orientation of the images.
2. **Width shift** (width\_range = 0.1) and a **height shift** (height\_shift\_range = 0.1) This means that the images can be randomly shifted horizontally by up to 10% of the total width and vertically by up to 10% of the total height. This kind of augmentation adds variances to the positions of the objects in the image. By exposing the model to a variety of spatial feature arrangements, it improves generalization and increases its ability to adapt to changes in object location. Using shifts can improve the model's recognition of changes in organ positions or abnormalities across multiple X-ray images, which is common in medical imaging.
3. **Shear** (shear\_range = 0.2) : When conducting shear transformations on the images during data augmentation. Shear transformation creates a "tilting" effect by moving a single portion of an image in a particular direction. The images can be randomly sheared by up to 20%. This makes the model able to adapt to object tilts or deformations in the image. When specific structures in the photos may vary in angle or orientation and the model has to identify these structures regardless of orientation, this kind of augmentation is essential.
4. **Zoom** (zoom\_range = [0.8, 1.2]): This parameter defines the range across which the images will be randomly zoomed in and out. This helps with controlling the degree of zooming that can be used on the pictures. The images can be dynamically zoomed in or out by a factor ranging from 0.8 to 1.2 . Here, 0.8 indicates a zoom out, or a smaller image, and 1.2 indicates a zoom in , or a larger image. During the augmentation process, the actual zoom factor is uniformly sampled for each image within the range. By doing this, the model becomes able to adapt to changes in the object scale within the images. Also, it enhances the model's capacity to generalize to different item sizes and positions.
5. **Horizontal flip** (horizontal\_flip = True): When horizontal\_flip=True, the input images will go through random horizontal flips as part of the data augmentation process. Images are randomly and 50% likely flipped horizontally. This implies that there is a chance that every image in the training dataset will be flipped horizontally while being trained.
6. **Fill mode** (fill\_mode = 'nearest'): The technique is used for filling in newly formed pixels that may be introduced during various imagine transformations, such as rotations or width/height shifts, is determined by the fill\_mode parameter. The value of the closest nearby pixel will be filled in the newly created pixels. This is a simple yet widely used technique that fills in the gaps by using the pixel value of the closest existing pixel.

# Methodology

## To solve the image classification challenge, we used a Convolutional Neural Network (CNN) algorithm in our methodology. Given its ability to recognize spatial patterns and hierarchies in images, CNNs are especially well-suited for tasks involving images [1]. The goal of our CNN-based method was to make use of the model's capacity to recognize and identify characteristics from chest X-ray pictures, which are essential for differentiating between Pneumonia patients and normal cases.

## Layers in the CNN Model

* Convolutional Layers: The first convolutional layer, or Conv2D with 16 filters, is essential for extracting the input images' basic spatial information. It makes use of sixteen 3x3 pixel-sized filters. To make sure that the feature maps' output dimensions correspond to the dimensions of the input images, the "Valid" padding approach is used. In order to add non-linearity, this layer also uses the Rectified Linear Unit (ReLU) activation function. This first layer starts the process of identifying the basic elements of an image. To help capture features that are more complex and abstract, we gradually add more convolutional layers to the initial layer's base. The architecture of each of these layers is the same: Conv2D with a 3x3 filter size, "same" padding, ReLU activation, and a predetermined number of filters (e.g., 32, 64, and 128). These later layers are meant to enhance and improve the input images' feature representation. The model learns to identify complex structures, textures, and patterns in the chest X-ray pictures as data moves through these layers.
* **Max-Pooling Layer:** There is a max-pooling layer (MaxPooling2D()) added after every convolutional layer. By choosing the largest value in each zone, max-pooling down-sampled the feature maps and reduces the spatial dimensions without sacrificing important information. This procedure lowers computational complexity and helps with feature abstraction.
* **Dropout Layer:** After the convolutional layers, a dropout layer with a rate of 0.2 is added. This layer encourages model generalization by randomly setting 20% of the input units to zero during training. This helps avoid overfitting.
* **Flatten Layer:** We utilize a flatten layer to convert the 2D feature maps into a 1D vector before moving on to fully connected layers. In order to make the data compatible with dense layers, this change is required.
* **Dense Layers:** We thoroughly investigated the effects of various arrangements for dense layers. A variety of dense layers with different neuron counts and activations were engaged in this. We specifically tested one dense layer, then two dense layers: one with 128 neurons and another with 256 neurons. We then progressively increased the depth to four dense layers with 64, 128, 128, and 256 neurons each. These dense layers aid in the classification and learning process [2].
* **Output Layer:** The last dense layer.Dense(1, activation='sigmoid'), appropriate for binary classification applications, is composed of a single neuron with sigmoid activation. The likelihood that an image is in the pneumonia class is output by it.
* **Model Compilation**: The accuracy metric, binary cross-entropy loss function, and Adam optimizer are used in the compilation of the model. By defining the optimization plan and performance standards, this configuration gets our model ready for training.

## Sensitivity Analysis

It is a useful technique that is essential to our work and allows us to assess our convolutional neural network (CNN) model's performance in-depth. Each dense layer, neuron in each layer, batch size, weight initializers, type of padding, the type of pooling layers and regularization methods are among the many model parameters and hyperparameters that are methodically changed one at a time. The model is retrained on the training data following each adjustment, and a validation dataset is used to assess the model's performance. We are able to comprehend the relationship between these parameters and the model's predictive power better by examining the effects of these modifications on the model's performance. Sensitivity analysis helps us make well-informed decisions on architectural selections and parameter tweaking.

* **Number of Dense Layers:** We systematically changed the number of dense layers in order to investigate the effect of model depth. As a baseline, we started with a single dense layer that had 256 neurons, then we tried one dense layer with 256 neurons, we have also tried many tests that will be later explained in details. This gave us the opportunity to look at how the model's capacity to capture complex features in the data was impacted by depth and complexity The model we created had three dense layers with 128, 256 and 64 neurons..
* **Batch Size**: We assessed how the size of the batch affected the training process. Three distinct batch sizes—32, 64, and 128—were tried. Larger batch sizes, such as 64 and 128, are anticipated to enhance computational efficiency and possibly result in smoother convergence, therefore improving generalization. Smaller batch sizes, such as 32, are meant to be more memory-efficient and may operate as implicit regularization, enabling the exploration of various solutions. The selection of batch size is a crucial hyperparameter, and this study tries to determine its effects on both efficiency and model generalization.
* **Weight Initialization Techniques**: We used Glorot normal, truncated normal, random normal, and He normal weight initialization approaches on the model's layers. The model's initial weight setting at the start of training is influenced by weight initialization [3]. To determine which weight initialization produced the best results for our work, we examined the results of several approaches. During training, every strategy has a different effect on how well the model learns and adapts.
* **Type of Pooling Layer:** The experiment will also investigate the influence of different pooling layers, particularly Max and Average Pooling. The purpose of this analysis is to determine how pooling layer selection influences the model's ability to capture and represent significant characteristics during the image classification process. We intend to determine the pooling method that best matches with the details of the dataset and contributes to optimal model performance by thoroughly testing both Max and Average pooling.
* **Stride Number:** We will also expand our research by examining the impacts of various stride values in the pooling layer. By experimenting with different stride numbers, we can see how the level of data compression in the pooling layer influences the model's feature extraction and classification abilities. Smaller strides preserve more spatial information but raise computing demands, whereas bigger strides lead to more aggressive down-sampling. We expect to determine the ideal configuration that achieves an appropriate equilibrium between computational efficiency and the preservation of critical features in the CNN image classification process by testing stride values.
* **Padding Type:** We are going to investigate the effects of two different padding methods, specifically "Valid" and "Same," in the convolutional and pooling layers deeper. We are seeking to see how these padding strategies affect the network's capacity to gather relevant features and spatial information during the image classification process. The study will shed light on the role of padding in optimizing the computational efficiency in our CNN architecture[4].

# Experiments

We performed a variety of experiments in this section that are intended to offer important new information and enhance the functionality of our Convolutional Neural Network (CNN) for the classification of X-ray images. Every experiment has been motivated by a purposeful investigation of different hyperparameters, architectural choices, and regularization strategies. By careful manipulation of factors such as the number of dense layers, batch size, techniques for weight initialization, stride number and padding type. Our objective is to fully comprehend the various manners in which these factors affect the model's sensitivity, convergence, and generalization capabilities. Every experiment is thoroughly explained, including the basic objectives for our investigation in addition to the technical details. The outcomes offer insightful information about the subtleties of CNN training for jobs involving the classification of medical X-ray images.

### Experiment 1(Number of Dense Layers):

#### Motivation:

The main objective of this experiment is to examine how the number of dense layers and number of neurons influence model performance and to evaluate the sensitivity of our model to changes in the architecture. We seek to understand how different configurations of dense layers impact classification accuracy[2]. We are going to systematically adjust the number of dense layers and neurons, providing a thorough understanding of the elements influencing classification accuracy in our CNN image classification challenge.

#### Description:

We have thoroughly examined different density layer configurations: one dense layer with 128 neurons, one dense layer with 256 neurons, two dense layers with 128 and 256 neurons, three dense layers with 256, 128, and 64 neurons, and four dense layers with 256 ,128, 128 and 64 neurons, These layouts were selected in an effort to satisfy a range of architectural complexities. To prevent overfitting and ensure effective convergence, each experiment was subjected to an early stop with a patience of 3 epochs and a maximum of 100 epochs. To provide for an unbiased and informative comparison, the hyperparameters—such as the learning rate, optimizer, and activation functions—were maintained constant across all tests.

#### Results:

We explored with different dense layer configurations to determine the best architectural configuration for our model that was to be used for the classification of X-ray images. The model showed an accuracy of 94.19% in the case of a single dense layer with 128 neurons. However, misclassifications were noted, indicating the need for more examination. subsequent to that, an individual dense layer including 256 neurons showed enhanced accuracy at 95.32%; however, there was a rise in false positives. After deciding to switch to two dense layers, the model's accuracy was 93.39% while maintaining a steady misclassification pattern. The best accuracy of 95.97% was obtained with the addition of a third dense layer, which also reduced misclassifications. We noticed the accuracy of the four-layered dense model decreased somewhat (to 94.84%) as the number of misclassifications grew.

#### Interpretation and Discussion :

The in-depth examination of results across various dense layer configurations provides significant insight into how our model performs when it applies to classifying the X-ray images. We have concluded that our model is somewhat sensitive to the number of dense layers. The accuracy of the single dense layer with 256 neurons significantly increased, highlighting the importance of neuron count in feature extraction. Consistent misclassification patterns, as seen in the 2-layer configurations, promote the exploration of alternate architectures to reduce these problems. The network demonstrated a significant increase in accuracy upon the addition of a third dense layer, illustrating its ability to use hierarchical characteristics to enhance discrimination. On the other hand, the four-layered model exhibits remarkable accuracy but a diminishing return on complexity. We analyzed the results meticulously and reached a conclusion that the 3-layer structure scored the best overall, which was in line with our main objective of reducing false positives and false negatives in the critical field of medical image classification.

### Experiment 2 (Batch Size):

#### Motivation:

The purpose of experimenting with various batch sizes was to determine how they would affect our model's overall performance and learning dynamics. The size of the batch plays a critical role in the stochastic nature of gradient descent, which affects the training speed, convergence, and generalization. Our goal was to figure out the batch size that best balances all these factors for precise and effective X-ray image classification and to see how sensitive our model is to the batch size.

#### Description:

Similar to our previous experiment, we performed attempts with three distinct batch sizes: 32, 64, and 128. We also maintained the maximum number of epochs equal to 100 and used an Early Stopping mechanism with a patience of 3. A range from smaller, more frequent updates (32) to bigger, less frequent updates (128) can be observed by the selection of these batch sizes. To isolate the effect of batch size on performance, the architecture of the model and other hyperparameters remained unaltered where we used 3 dense layers based on the results of the previous experiment and the same loss function and optimizer.

#### Results:

The experiment we carried out into different batch sizes for the CNN model has revealed important information on how to best optimize the learning dynamics and overall performance of the model in the classification of X-ray images. Batch size 64 stood out with the accuracy of 96.29% and showed a balanced confusion matrix. Nonetheless, this configuration's computing cost encouraged a strategic assessment of other options. With a slightly less efficient computational profile, batch size 128 showed competitive performance with an accuracy of 95.65%. Batch size 32, on the contrary, offered a convincing trade-off favoring computational efficiency without compromising classification accuracy, with an accuracy of 95.97%.

#### Interpretation and Discussion:

The in-depth analysis of different batch sizes suggested important factors that could enhance the performance of our model in the classification of X-ray images. Batch size 64 showed a well-distributed confusion matrix. However, this resulted in higher computing demands, demanding a critical assessment of computational efficiency. Compared to batch size 128 revealed a somewhat less efficient computational profile. Batch size 32 struck an intriguing mix between computational efficiency and high accuracy. Seeing the similar accuracies between different batch sizes indicates that our model isn’t sensitive to the batch size. However, we made the choice to continue the rest of the experiments with batch size 32 because of this complex trade-off, which falls in line with our main objective of maximizing practical applicability. This architecture was chosen as a strategic compromise between processing needs and classification accuracy. Together with our selection of batch size, the early stopping mechanism highlights our dedication to creating a CNN that excels in both accuracy and the real-world limits of medical image diagnoses.

### Experiment 3 (Weight Initialization Techniques):

#### Motivation:

The motivation for examining several weight initialization techniques comes from the significant impact that initialization techniques have on the progress of our model and its performance. The starting point of the optimization process will be set by weight initialization, thereby affecting the ability of the model to learn and generalize well. We wanted to analyze and compare four different weight initialization techniques - Glorot normal, Truncated normal, Random normal, and He normal [3]- to see how they influenced the training dynamics, the sensitivity of our model to different weight initialization techniques and overall performance of the model. Each initialization technique has different characteristics, and comprehending their implications is critical when creating a robust and efficient model.

#### Description:

Within a uniform methodology, we conducted an in-depth evaluation of four distinct weight initialization techniques. Among the techniques chosen are:

**Glorot Normal:** This technique selects samples from a normal distribution with mean 0 and variance computed based on the number of input and output units to initialize weights.

**Truncated Normal:** This method draws samples from a truncated normal distribution and restricts the results within a given range to prevent outliers.

**Random Normal:** Weights are created by randomly selecting samples from a normal distribution having a mean of zero and a standard deviation equal to one.

**He Normal**: This method modifies the variance based on the number of input units, but the scaling factor is different.

The model had been set up using three dense layers, a binary cross-entropy loss function, an Adam optimizer, a batch size of 32, and 100 epochs for each weight initialization method. To avoid overfitting, Early Stopping was implemented with a patience of 3. The use of Glorot, Truncated, Random, and He normalization techniques allows for a thorough assessment of their impact on accuracy, loss, and classification accuracy. We aim to discover the best weight initialization approach through this experiment.

#### Results:

The testing of various weight initialization methods for our model has revealed different outcomes, offering light on the nuanced impact of initialization methods on X-ray image classification. Glorot Normal achieved an accuracy of 93.87%, demonstrating a balanced prediction profile. Truncated Normal and Random Normal had greater misclassifications, with accuracies of 87.58% and 87.26%. However, He Normal appeared as the most effective initialization technique, with a 95.16% accuracy. The He Normal confusion matrix produced fewer misclassifications, highlighting its practicality for our specific medical image classification challenge.

#### Interpretation and Discussion:

The experiment we conducted into various weight initialization techniques gave remarkable insights into their consequences for our CNN model. Glorot Normal had a well-balanced prediction performance. The substantial rise in precision found with He Normal indicates the important relevance of suitable weight initialization for model effectiveness. However, after observing the accuracies and the loss of each weight initialization technique, we have concluded that our model is sensitive to different techniques of weight initialization. The selection of He Normal is consistent with our initial motivation, underlining the importance of selecting initialization approaches that improve convergence and overall model performance. The various outcomes highlight the complex interaction between weight initialization and the accuracy of our CNN in medical image detection.

### Experiment 4 (Max vs Average Pooling Layer)

#### Motivation:

The selection of a pooling layer in Convolutional Neural Networks (CNNs) is an important architectural decision that has a major effect on the network's ability to extract significant features from input data. The objective of this experiment was to evaluate the performance of two commonly used pooling techniques, Max Pooling and Average Pooling. Both pooling layers operate as down-sampling approaches, but they do so in different ways. Max Pooling selects the maximum activation within a particular area to highlight the dominating features, whereas Average Pooling computes the average activation to provide a more generalized representation [8]. We wanted to figure out the most effective pooling layer for our unique medical image classification challenge by examining their effects on the model's accuracy and convergence.

#### Description:

We evaluated Max Pooling and Average Pooling within a CNN layout. Three dense layers with 256, 128 and 64 neurons, a batch size of 32, and He Normal weight initialization are among the parameters that were chosen. We have also maintained consistency with a loss function of Binary Cross Entropy, 'Adam' as our optimizer and Early Stopping with a patience of 3. The subtleties of pooling layer behavior are the focus of this experiment. Max Pooling picks the maximum activation where dominant features and edges are accentuated, while Average Pooling calculated the average activation, where it captures a more generalized representation. The accuracy and convergence characteristics of the CNN model will be assessed under these different pooling methods. This experiment seeks to provide insights on the best pooling layer for optimizing the extraction of relevant features in medical image classification.

#### Results:

The comparison of pooling layer approaches, Average Pooling and Max Pooling in our model's architecture, revealed significant results. Average Pooling achieved 94.19% accuracy with a loss of 0.15, indicating an impressive performance. Max Pooling, on the other hand, significantly outperformed with an accuracy of 94.68% and a loss of 0.13. Both functioned admirably, showing the model's resistance and insensitivity to the specific pooling approach employed. The confusion matrices displayed complex misclassification patterns, underscoring each approach's strengths in discriminating between normal and pneumonia cases. Despite the model's overall insensitivity to pooling layer type, a slight advantage with Max Pooling has motivated us to use this configuration in future tests.

#### Interpretation and Descision:

The contrasting results of Average Pooling and Max Pooling in our CNN model suggests low sensitivity to pooling layer selection, with both obtaining high accuracy rates. The confusion matrices exhibit sophisticated misclassification patterns, with both pooling algorithms showing strengths in differentiating between normal and patients with Pneumonia. In the larger context, our approach looks to be robust and not unduly sensitive to the specific pooling layer selection. Despite this, a minor performance benefit noticed with Max Pooling motivated us to stick with this configuration for further research. The decision indicates a preference for the pooling layer, which slightly improves the model's discriminative power.

### Experiment 5 (Stride Number in CNN Model):

#### Motivation:

The selection of stride in Max pooling layers is an essential variable affecting the down-sampling process in CNNs. Stride determines the step size at which the pooling process occurs, therefore affects the spatial resolution of the learned features. The purpose of this experiment is to see how sensitive our current CNN architecture, our goal is to variable stride values (1 and 2) in the max pooling layers. We want to examine the influence of stride on accuracy, loss, and model performance in order to discover the best configuration for our challenge in classification.

#### Description:

Within our CNN architecture, we investigated stride values in max pooling layers while keeping consistency with three dense layers, Binary Cross Entropy as our loss function, ‘Adam’ as our optimizer, Early Stopping was implemented with a patience of 3 over 100 epochs, a batch size of 32, and He Normal weight initialization. The experiment was limited to two stride values: 1 and 2. A stride of 1 indicates that the pooling operation is non-overlapping, but a stride of 2 introduces overlapping pooling, which affects the spatial resolution of the learned features.

#### Results:

Our model showed stable performance with minimal sensitivity to stride value in the exploration of multiple stride values in max pooling layers. Stride 1 obtained an accuracy of 95.16% and a loss of 0.13, showing precise classification. Stride 2 followed closely behind, with an accuracy of 94.03% and a loss of 0.17. Given the fact that the confusion matrices highlighted complex misclassification patterns, overall model performance remained consistent across both configurations. These results imply that the stride number in the max pooling layers has no effect on our CNN architecture.

#### Interpretation and Descision:

The examination of different stride values in max pooling layers indicated insignificant variations in accuracy and loss, demonstrating that the choice of stride is not significantly sensitive to our model. Across both configurations, the overall model performance remained stable. After careful evaluation, we chose to stick with Stride 1 for the subsequent experiments since it performed somewhat better. This choice illustrates our dedication to improve the CNN architecture for maximum precision in medical image classification.

### Experiment 6 (The Type of Padding in the Convonutional Layers):

#### Motivation:

The type of padding applied to convolutional layers has an important effect on the spatial dimensions of the learned features. Padding affects how the convolutional operation interacts with the input data. We explored two types of padding: "Same" and "Valid." "Same" padding adds zeros to the input data, making sure that the convolutional layer's output size remains the same as the input size. In contrast, "Valid" padding does not include any zeros [5], resulting in a smaller output size as the convolutional filter goes over the input with no padding. The experiment's goal was to rigorously analyze how padding types affect accuracy and loss, providing insight into their impact on our model's overall performance.

#### Description:

We did a comparison of the "Same" and "Valid" padding types in our CNN architecture's convolutional layers. With three dense layers with 256, 128 and 64 neurons, a batch size of 32, He Normal weight initialization. The loss function was Binary Cross Entropy, with the Adam optimizer, and Early Stopping was carried out with a patience of 3 over 100 epochs. "Same" padding keeps the convolutional layer's output size constant, but "Valid" padding allows for decrease. The experiment's goal was to rigorously evaluate the model's sensitivity to padding types, offering insight into their impact on accuracy, loss, and overall model performance.

#### Results:

Both padding choices did exceptionally well in this experiment, illustrating the robustness of our CNN model to padding type option. The "Same" padding produced an accuracy of 95.16% with a loss of 0.13, whereas the "Valid" padding produced an accuracy of 95.32% with a loss of 0.14. The confusion matrices indicated subtle misclassification patterns, with both configurations doing decently overall.

#### Interpretation and Descision:

The examination of different padding types in convolutional layers revealed that our CNN model performed well in both "Same" and "Valid" configurations. Given the subtle performance gain noticed with "Valid" padding, as well as its computationally efficient nature (requiring fewer computations due to the absence of zero-padding), we strategically chose to employ the "Valid" padding configuration in the subsequent experiments. This change is consistent with our commitment to improving the CNN architecture for maximum precision in medical image classification, acknowledging the importance of both performance and computational efficiency in real-world applications.

# Related Work

Exploration of related work proves essential context and expanding on the existing knowledge landscape. Various studies in the field of medical image classification have offered useful insights, extending our understanding and directing the development of new techniques and methodology. We have come across a notable contribution from Anoop's Singh's article "Classification of Cardiomegaly using Convolutional Neural Network" which was published on February 8, 2018, where he delves into the implementation of CNNs for X-ray image classification when it comes to Cardiomegaly. Singh's approach correlates with current medical image analysis methods, exhibiting commonalities with our own approach.

Notably, dataset preprocessing included splitting the dataset into training and validation sets, which enhanced the robustness of model evaluation. The use of the "ReLU" activation function introduces non-linearity, which is essential for learning complex patterns. As well as the inclusion of a dropout layer with a rate of 0.2 contributes to regularization while addressing overfitting problems. Anoop Singh made use of Keras' ImageDataGenerator class, a powerful tool for data augmentation, extends the dataset and enhances the model's generalization capabilities. The study found a noteworthy accuracy of 0.95, demonstrating the CNN architecture's effectiveness in successfully classifying X-ray images associated with Cardiomegaly [6].

Anoop Singh's work stands out as a valuable guidance, offering insights into architectural decisions and preprocessing methods that contribute to effective model outcomes in X-ray image classification.

[7]

# Conclusion

To conclude, the persistent examination of hyperparameters for our CNN model for Pneumonia recognition in X-ray pictures produced a refined model configuration. Our improved model obtained an astounding 95% accuracy through the use of three dense layers with 256, 128, and 64 neurons, a batch size of 32, He Normal weight initialization, a Max pooling layer with a stride of 2, and "Valid" padding based on our experiments. With 212 true negatives, 22 false positives, 377 true positives, and 9 false negatives, the outcome of the confusion matrix indicates the model's ability to distinguish between normal and patients with Pneumonia. This comprehensive strategy, which includes optimal hyperparameters as well as the Adam optimizer, Binary Cross Entropy loss function, and Early Stopping, highlights the significance of careful parameter tweaking in accomplishing robust performance in medical image classification. It's also worth noting that we've secured our optimized model by saving it for use in the future. This strategic method ensures that the model's meticulously established parameters and gained patterns may be easily applied in real-world scenarios, increasing its adaptability beyond the experimental phase. The journey we took not only led to a highly effective Pneumonia detection model, but it also showed the importance of the complex relationship of hyperparameters, providing essential knowledge for future tasks at the confluence of AI and healthcare.

# Refrences

[1] M. Mishra, “Convolutional Neural Networks, Explained,” *Medium*, Aug. 27, 2020. <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>

[2] Y. Verma, “A Complete Understanding of Dense Layers in Neural Networks,” Analytics India Magazine, Sep. 19, 2021. <https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/>

[3] J. N. Jan 26 and Read  〈 ◊2020 6 M., “Why should I do pre-processing and augmentation on my computer vision datasets?,” *Roboflow Blog*, Jan. 26, 2020. https://blog.roboflow.com/why-preprocess-augment/

[4] K. Team, “Keras documentation: Layer weight initializers,” keras.io. <https://keras.io/api/layers/initializers/>

[5] A. S. Raut, “Padding in Neural Networks: Why and How?” *Medium*, Mar. 27, 2023. <https://blog.gopenai.com/padding-in-neural-networks-why-and-how-b076ab0a4fc2>.

[6] “Classification of cardiomegaly using Convolutional Neural Network,” *www.linkedin.com*. https://www.linkedin.com/pulse/classification-cardiomegaly-using-convolutional-neural-anoop-singh/ (accessed Nov. 11, 2023).

‌[7] P. Gavrikov, “visualkeras for Keras / TensorFlow,” *GitHub*, Apr. 13, 2022. <https://github.com/paulgavrikov/visualkeras>

[8] C. Versloot, “Machine learning articles,” *GitHub*, Jul. 27, 2022. https://github.com/christianversloot/machine-learning-articles/blob/main/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling.md

‌

‌

‌

‌

‌

‌

‌

‌