Deep Learning- Final Course Project

July 6, 2024

Deep Learning- Final Course Project

Submitted as final project report for the DL course, BIU, 2024

By Omer Bar

Date:
July 6, 2024

Supervised by:
Dr. Moshe Butman

Click here to download the Network's Weights

Contents

1 Introduction

In this project, we aim to tackle the task of classifying chest X-ray images into different categories using deep learning techniques. The dataset we use for training is the Kaggle Chest X-ray Pneumonia Dataset, which contains 5,863 X-ray images categorized into two main classes: healthy and pneumonia. The pneumonia class is further divided into bacterial and viral pneumonia, which can be inferred from the image name. Our objective is to develop deep neural networks (DNNs) capable of accurately classifying these images, thereby aiding in medical diagnoses.

1.1 Related Works

Several works in the field of medical image analysis using deep learning have provided valuable insights into various techniques and methodologies. Some relevant resources include:

- A review paper on deep learning for medical image analysis
  - Diagnosis of Pediatric Pneumonia with Ensemble of Deep Convolutional Neural Networks in Chest X-Ray Images
  - Deep Learning on Chest X-ray Images to Detect and Evaluate Pneumonia Cases at the Era of COVID-19
  - Advances in Deep Learning-Based Medical Image Analysis
  - Medical image analysis using deep learning algorithms
  - Deep Learning in Medical Image Analysis

These resources have informed our approach and methodology for implementing deep learning techniques in our medical image classification project.

2 Solution

2.1 *General approach*

Our approach to the classification task prioritizes training deep neural networks via the Keras library. We aim to experiment with different network architectures, notably convolutional neural networks (CNNs), to identify the optimal model. Additionally, we'll delve into methods like embedded vector classification with KNN, autoencoder-based anomaly detection, and explainability techniques to improve model performance and interpretability.

2.1.1 Task 1.A - Classification of Normal/Sick

Our approach to classifying healthy and sick individuals centers around constructing a convolutional neural network (CNN) architecture. CNNs excel in image classification tasks because they can autonomously discern pertinent features from input images. For this task, we aim to devise a straightforward CNN model featuring an output layer equipped with two neurons, each dedicated to one of the categories—healthy or sick. This simplified architecture will facilitate efficient learning and decision-making, aligning with our goal of achieving accurate and swift classifications.

2.1.2 Task 1.B - Classification of Normal/Viral/Bactrial

To enhance the CNN architecture for classifying healthy, bacterial pneumonia, and viral pneumonia cases, we will build upon the framework established in Task 1.1. Our approach involves modifying the output layer of the network to accommodate the three distinct classes and refining the loss function accordingly. Additionally, we acknowledge the significance of restructuring the provided data folder to ensure proper organization and facilitate the loading of training data for each category. This restructuring aims to optimize the training process and enhance the model's accuracy in distinguishing between different pneumonia types. (Please note, we experimented with several different model architectures due to low model accuracy. The architecture that closely resembles the Task 1.A model yielded the best results.)

2.1.3 Task 2 - Utilizing Embedding Vectors and KNN for Classification

To classify new images using embedding vectors and KNN, we'll utilize CNNs from Tasks 1.1 and 1.2. We'll extract embedding vectors from these CNNs' intermediate layers as feature vectors for classification. With KNN, we'll find nearest neighbors in the embedding space to label new images. Additionally, we'll visualize the embedding space using t-SNE to interpret classification results, aiming for a robust framework for image classification.

2.1.4 Task 3 - Anomaly Detection for Identifying Sick Individuals

For anomaly detection, we'll employ autoencoder neural networks, renowned for capturing data structure and reproducing normal instances. Our strategy involves training an autoencoder exclusively on "healthy" image data to assess its reconstruction performance with unseen images. Anomalies, representing "sick" cases, will be detected based on reconstruction error magnitude, identifying deviations from typical patterns. Additionally, tailored data augmentation techniques will be applied to enrich the "normal" X-ray dataset, enhancing the autoencoder's capacity to learn diverse normal patterns and enhance anomaly detection accuracy.

2.1.5 Task 4 - Explainability of Model Decisions

To interpret the decisions made by the model trained in Task 1.A, we plan to implement Grad-CAM (Gradient-weighted Class Activation Mapping) for visualizing the regions of input images that contribute the most to the model's predictions. Grad-CAM provides valuable insights into the features learned by the CNN and helps understand the rationale behind its decisions. By analyzing the generated heatmaps, we can identify important image regions and validate the model's predictions, enhancing its explainability and interpretability.

2.2 *Design*

We implemented our deep learning models using Keras, leveraging its simplicity and flexibility. The training process involved preprocessing the dataset, experimenting with different architectures and hyperparameters, and evaluating the models' performance on validation and test sets. We encountered challenges related to data imbalance, model convergence, and interpretability, which we addressed through careful experimentation and analysis.

2.2.1 Task 1.A - Design

I chose this architecture (Figure 1) for normal/sick classification because it strikes a balance between complexity and performance:

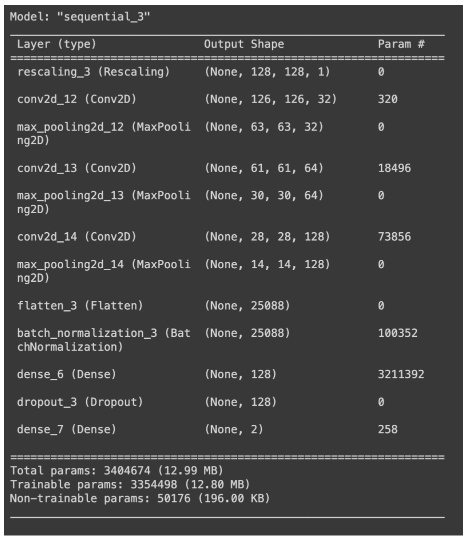
- Convolutional Layers: These layers capture hierarchical features from images.
- Increasing Depth: The network progressively increases filters, capturing more complex features.
- Batch Normalization: Stabilizes training by normalizing layer activations.
- Dropout: Prevents overfitting by randomly dropping neurons.
- Final Dense Layer: Outputs probabilities for the two classes (normal/sick).

Overall, this architecture is suitable for chest X-ray classification due to its depth, regularization techniques, and manageable parameter count, also because of binary classification we chose the sigmoid activation function at the last layer.

2.2.2 Task 1.B - Design

The chosen architecture (Figure 2) for the classification of normal/bacterial/viral pneumonia in chest X-ray images is a sequential neural network composed of convolutional layers followed by dense layers. The architecture is as follows:

- The input layer performs rescaling to normalize pixel values.
- Three convolutional layers with increasing depth and max-pooling layers for downsampling.



```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_3 (Rescaling)     (None, 128, 128, 1)       0

 conv2d_12 (Conv2D)          (None, 126, 126, 32)      320

 max_pooling2d_12 (MaxPooli  (None, 63, 63, 32)        0
 ng2D)

 conv2d_13 (Conv2D)          (None, 61, 61, 64)        18496

 max_pooling2d_13 (MaxPooli  (None, 30, 30, 64)        0
 ng2D)

 conv2d_14 (Conv2D)          (None, 28, 28, 128)       73856

 max_pooling2d_14 (MaxPooli  (None, 14, 14, 128)       0
 ng2D)

 flatten_3 (Flatten)         (None, 25088)             0

 batch_normalization_3 (Bat  (None, 25088)             100352
 chNormalization)

 dense_6 (Dense)             (None, 128)               3211392

 dropout_3 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 2)                 258

=================================================================
Total params: 3404674 (12.99 MB)
Trainable params: 3354498 (12.80 MB)
Non-trainable params: 50176 (196.00 KB)
_________________________________________________________________
```

Figure 1: Task 1.A Model Architecture

- A flattening layer to convert the 3D feature maps into a 1D vector. Two dense layers with dropout regularization.
- The final dense layer with three output units, representing the probabilities of the three classes (normal, bacterial pneumonia, viral pneumonia).

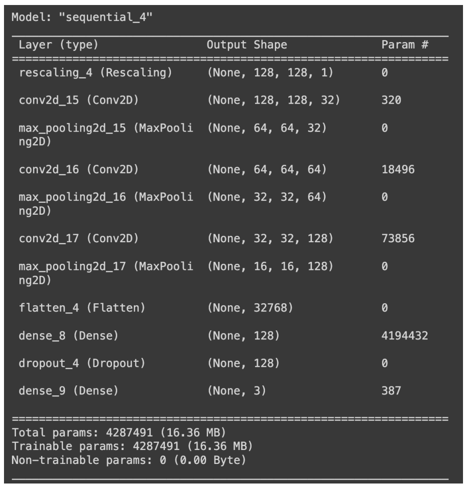
This architecture is selected for its ability to hierarchically extract features from images, capture increasingly complex patterns, and regularize the network to prevent over-fitting. With a total of 4,287,491 parameters, it strikes a balance between model complexity and manageability, making it suitable for multi-class classification tasks, This task is for categorical classification, so we leverage the power of the softmax activation at the last layer, that is known for better performance in categorical cases.

2.2.3 Task 2 - Design

For this task, we didn't need new neural network architectures. Instead, we used models from tasks 1.A and 1.B to generate embedded vectors for KNN classification. Vectors were (1, 128)-sized per image, and (n, 128)-sized for the test dataset. We applied t-SNE for visualization, revealing clustering patterns. However, results were inconsistent at times, with KNN and neural network predictions differing.

2.2.4 Task 3 - Design

We chose this autoencoder architecture (Figure 3) for anomaly detection because it consists of convolutional and max-pooling layers followed by upsam-



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_4 (Rescaling)     (None, 128, 128, 1)       0

 conv2d_15 (Conv2D)          (None, 128, 128, 32)      320

 max_pooling2d_15 (MaxPooli  (None, 64, 64, 32)        0
 ng2D)

 conv2d_16 (Conv2D)          (None, 64, 64, 64)        18496

 max_pooling2d_16 (MaxPooli  (None, 32, 32, 64)        0
 ng2D)

 conv2d_17 (Conv2D)          (None, 32, 32, 128)       73856

 max_pooling2d_17 (MaxPooli  (None, 16, 16, 128)       0
 ng2D)

 flatten_4 (Flatten)         (None, 32768)             0

 dense_8 (Dense)             (None, 128)               4194432

 dropout_4 (Dropout)         (None, 128)               0

 dense_9 (Dense)             (None, 3)                 387

=================================================================
Total params: 4287491 (16.36 MB)
Trainable params: 4287491 (16.36 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

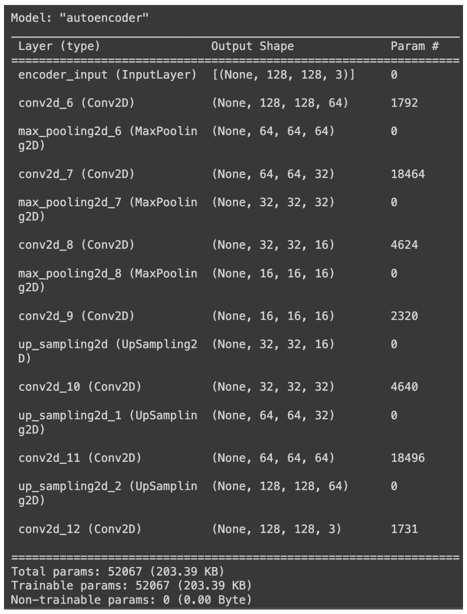
Figure 2: Task 1.B Model Architecture

pling layers. This architecture is well-suited for capturing spatial dependencies and patterns within the input images while reducing their dimensionality.

- The autoencoder architecture was selected for anomaly detection due to its ability to capture spatial dependencies and patterns in input images effectively.
- It consists of an encoder that compresses input images into a lower-dimensional latent space and a decoder that reconstructs the original images from this latent representation.
- This architecture is well-suited for anomaly detection tasks as it can identify significant deviations from learned normal patterns in the reconstruction errors.
- With a total of 52,067 parameters, it strikes a balance between model complexity and computational efficiency.
- The convolutional layers in the encoder and decoder enable the model to learn hierarchical representations of the input images, facilitating the detection of anomalies.

2.2.5 Task 4 - Design

For this task, we are utilizing the model trained in task 1.A and applying the technique known as "Visualizing patches that maximize activation," which we learned during the course. This method involves selecting a random image from both the "Normal" and "Pneumonia" datasets. We then generate heatmaps for each convolutional layer in the model to visualize which parts of the input image



```
Model: "autoencoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_input (InputLayer)  [(None, 128, 128, 3)]     0

 conv2d_6 (Conv2D)           (None, 128, 128, 64)      1792

 max_pooling2d_6 (MaxPoolin  (None, 64, 64, 64)        0
 g2D)

 conv2d_7 (Conv2D)           (None, 64, 64, 32)        18464

 max_pooling2d_7 (MaxPoolin  (None, 32, 32, 32)        0
 g2D)

 conv2d_8 (Conv2D)           (None, 32, 32, 16)        4624

 max_pooling2d_8 (MaxPoolin  (None, 16, 16, 16)        0
 g2D)

 conv2d_9 (Conv2D)           (None, 16, 16, 16)        2320

 up_sampling2d (UpSampling2  (None, 32, 32, 16)        0
 D)

 conv2d_10 (Conv2D)          (None, 32, 32, 32)        4640

 up_sampling2d_1 (UpSamplin  (None, 64, 64, 32)        0
 g2D)

 conv2d_11 (Conv2D)          (None, 64, 64, 64)        18496

 up_sampling2d_2 (UpSamplin  (None, 128, 128, 64)      0
 g2D)

 conv2d_12 (Conv2D)          (None, 128, 128, 3)       1731

=================================================================
Total params: 52067 (203.39 KB)
Trainable params: 52067 (203.39 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 3: Autoencoder Model Architecture

contribute most to the activation of neurons in those layers. By comparing the heatmaps generated for images from different classes, we aim to understand how the neural network discerns between normal and pneumonia cases based on the features it learns.

3 Experimental results

We conducted experiments using various model architectures, loss functions, and optimizers to optimize performance. **All details on the tasks result's are in the Notebook**

3.1 Task 1.A - Experimental result

In this project, we aimed to classify chest X-ray images into two categories: normal and pneumonia. We trained a convolutional neural network (CNN) on the Kaggle Chest X-ray Pneumonia Dataset and evaluated its performance on a test set. The model achieved a test accuracy of 77.40%, with a notable difference in performance between normal and pneumonia cases. Here are the key findings and insights:

- The model demonstrates high accuracy during training and validation but exhibits lower performance on the test set, indicating potential overfitting or difficulties in generalization.

- Disparities between precision and recall for normal and pneumonia cases suggest challenges in accurately classifying normal X-rays compared to pneumonia cases.

In Figure 4, we illustrate the learning process (over 40 epochs) of the model by visualizing both the training and validation losses, as well as the training and validation accuracies.

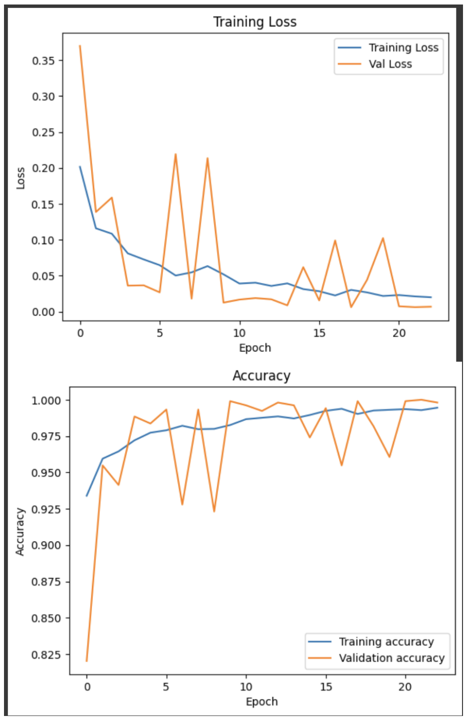


Figure 4: Loss and Accuracies - Model 1.A

3.2 Task 1.B - Experimental result

The model achieved test accuracy of 63.46%. (It's not much but it's the best we could achieved for this task :( ) During training, the model attained a training accuracy of 97.96%, while the validation accuracy reached 99.61%. The precision, recall, and F1-score metrics for each class are as follows:

- For the "BACTERIA" class, the precision is 0.66.
- For the "NORMAL" class, the precision is 0.96.
- For the "VIRAL" class, the precision is 0.46.

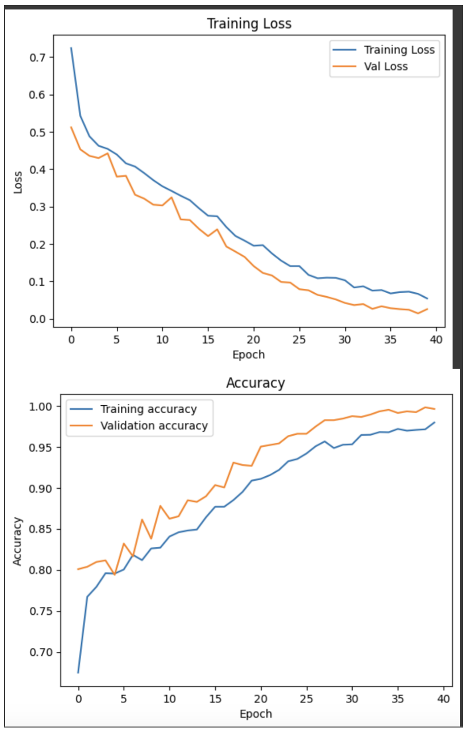


Figure 5: Loss and Accuracies - Model 1.B

In Figure 5, we illustrate the learning process (over 40 epochs) of the model by visualizing both the training and validation losses, as well as the training and validation accuracies.

3.3 Task 2 - Experimental result

This task focused on visualizing and classifying the embedded vector. The results occasionally aligned with the network's predictions, but there were inconsistencies at times. Further details can be found in the notebook. Here, we will present the T-SNE graph for both tasks.

3.3.1 Model 1.A

Here in Figure 6, we present two embedded vectors for each classification, followed by a test image depicted in green color, sourced from the internet.

3.3.2 Model 1.B

In Figure 7, we observe a distinct sparse pattern resembling a snake pattern for Task 1.B.

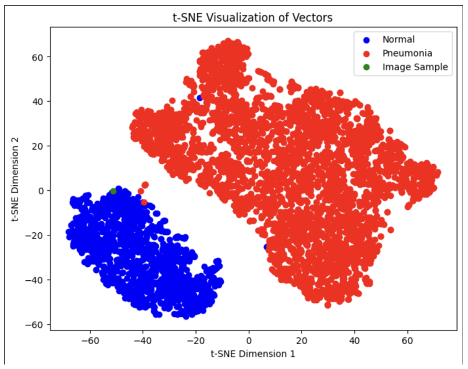


Figure 6: T-SNE for Model 1.A Embedded Vectors

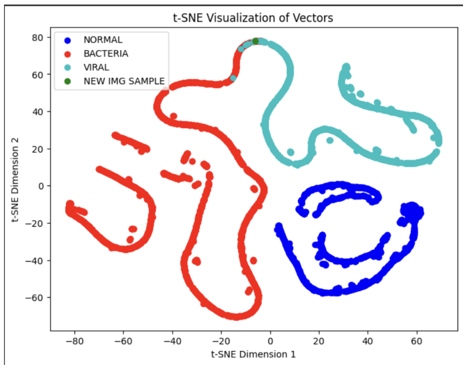


Figure 7: T-SNE for Model 1.B Embedded Vectors

3.4 Task 3 - Experimental result

This task posed significant challenges initially, notably with all reconstructed images initially appearing black, indicating generation failure. Remedying this by transitioning from grayscale to RGB channels resulted in notable improvements. Despite these hurdles, the autoencoder effectively reconstructed ”Normal” X-rays with minimized loss, showcasing its efficacy. Following the successful construction of the autoencoder, performance evaluation ensued. Utilizing the test dataset for ”Normal” classification, mean squared error (MSE) calculations were conducted between each original image and its reconstructed counterpart. Subsequent computations involved deriving the mean and standard deviation of these errors. Images were classified as ”anomalies” if their errors exceeded one standard deviation from the mean. Analysis revealed that 27% of the ”Normal” dataset was identified as anomalies, indicating deviations from the norm. A larger proportion, 63%, of the ”Sick” dataset (Pneumonia) was classified as anomalies, suggesting a higher incidence of abnormal features in these images.

3.5 Task 4 - Experimental result

In this task, we visualize each convolutional layer of Model 1.A using a random input image from both the "Normal" and "Pneumonia" datasets to generate heatmaps overlaid on the original images. Additionally, we observe that the model is focusing on the center of the lungs for healthy images, while for sick images, it tends to focus on the outer edges of the lungs.

As illustrated in Figure 8, the impact of the convolutional layers on a test image is evident.

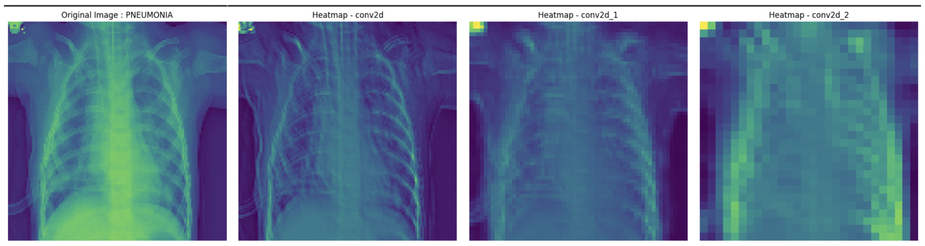


Figure 8: Random test image (Pneumonia this case) passing through each convolutional layer

And on Figure's 9-11 you can see the same original image, but visual with Heat-Map effect for each layer.

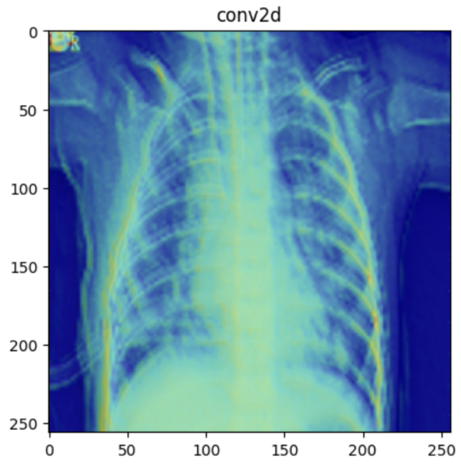


Figure 9: First Convolutional Layer Effect

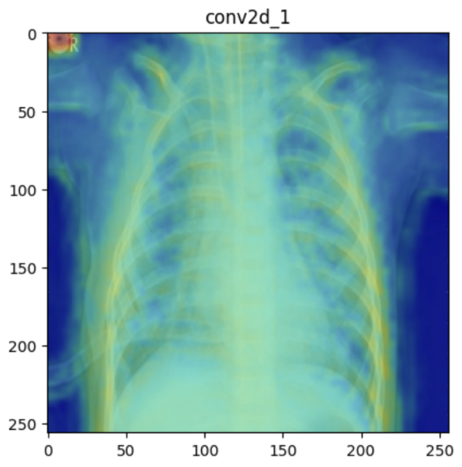


Figure 10: Second Convolutional Layer Effect

4 Discussion

Our project journey has been challenging yet enlightening, exploring deep learning techniques for medical image analysis. We faced hurdles in determining optimal hyperparameters like learning rates, loss functions, and optimizers. Image size choice was also challenging, balancing detail capture with computational resources. Despite challenges, rigorous experimentation and methodological integrity allowed effective navigation. Systematic testing and analysis provided valuable insights into hyperparameter impact. Data preprocessing and augmentation were critical, requiring careful technique and parameter selection. Overall, our project highlights the importance of systematic experimentation. Challenges provide insights, refining our understanding and advancing medical image analysis for improved healthcare outcomes.

5 Code

5.1 Notebook link (With change permission)

Train Notebook Test Notebook

5.2 Notebook link (Without change permission)

Train Notebook

This action serves to preserve the current state and prevent any alterations. Both notebooks are identical.

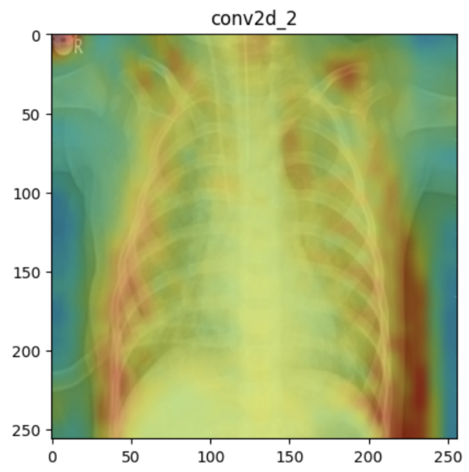


Figure 11: Third Convolutional Layer Effect

References

- TensorFlow CNN tutorial Convolutional Neural Network (CNN) — TensorFlow Core
- TensorFlow autoencoder tutorial Intro to Autoencoders — TensorFlow Core
- Anomaly detection with Keras and TensorFlow tutorial Anomaly Detection in TensorFlow and Keras Using the Autoencoder Method
- Grad-CAM visualization tutorial Understand the Grad-CAM: A Complete Guide With Example
- Heatmap generation from CNN article COVID-19 detection and heatmap generation in chest x-ray images