

**DEPARTMENT: Computer Science**

**Module Code: Deep Learning**

**Academic Year: 2024**

**Submission Deadline: 28/04/2024**

**Actual Submission Date: 28/04/2024**

|  |
| --- |
| **Student Registration Number** |
| ARY23030 |

**Declaration**: *I have completed and submitted this work by myself without assistance from or communication with another person either external or fellow student or any AI type of content generator. I understand that not working on my own will be considered grounds for unfair means and will result in a fail mark for this work and might invoke disciplinary actions. This piece of assessment will be continuously checked for its academic integrity until my graduation and the mark will be revised if it is found to breach the unfair means policy. It is at the instructor’s discretion to conduct an oral examination which will result in the award of the final grade for that particular piece of work.*

**Abstract**

This research talks about early detection and classification of lung cancer as it is an important research domain in the field of medical imaging and therefore helps in selecting the most convenient treatment method to save patients life therefore the application of deep learning approaches in context to improve health diagnosis is providing impactful solutions. According to the World Health Organization (WHO), proper lung disease diagnosis involves detection, lung cancer location identification, and classification of disease type. This method has experimented in terms of utilizing one model for classifying lung diseases on different classification tasks rather than an individual model for each classification task. The Convolutional Neural Network (CNN) based multi-task classification is equipped for the classification and detection of lung diseases. The identification of lung cancer location is also done using a CNN-based model by segmenting the lung diseases.

Contents

[**1. Introduction** 4](#_Toc164859459)

[**2. Data Preprocessing** 4](#_Toc164859460)

[**3. Training the data** 5](#_Toc164859461)

[**4. Model performance** 8](#_Toc164859462)

[**5. Testing the data** 8](#_Toc164859463)

[**6. Confusion matrix** 8](#_Toc164859464)

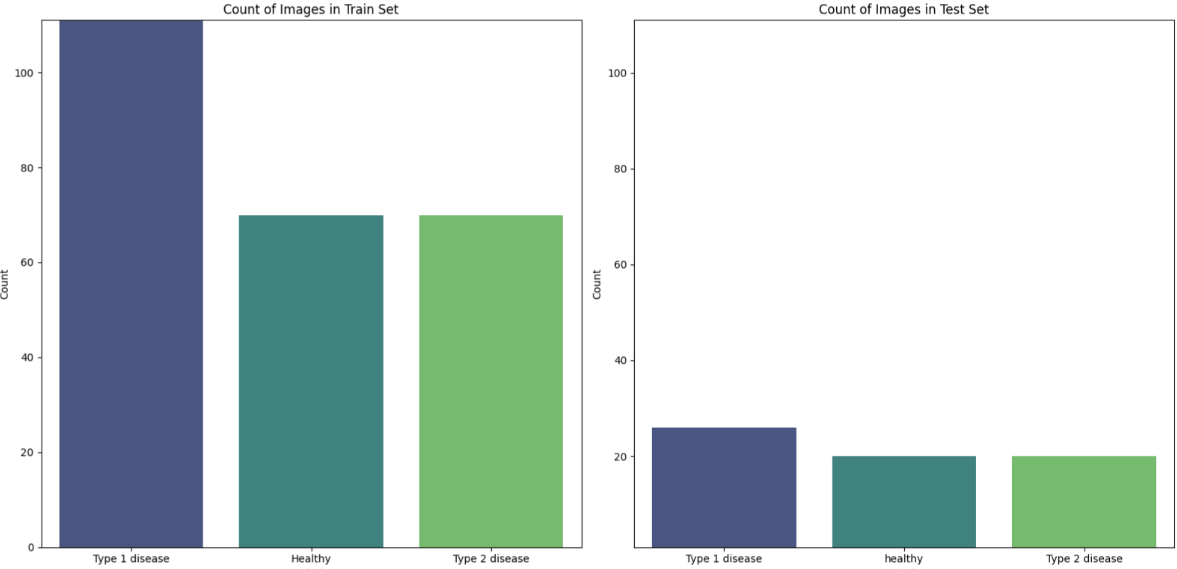
[**7. Conclusion** 9](#_Toc164859465)

# **1. Introduction**

The research is about detection of lung disease diagnosis, specialized in lung cancer, and classification of disease types. In this report, you can find methods used for data preprocessing, visualization, model structuring, training, testing, performance evaluation as well as classification analysis. Each step is used to evaluate productiveness and accuracy in the model's predictions. The importance of hyperparameter tuning, including optimizers, kernel sizes, filter sizes, and learning rates and more, is mentioned for the identification of the optimal model configuration (finding best combination). Confusion matrices and classification table are used to show a strong understanding of the model's classification capabilities across different classes of disease. Finally, this research aims to contribute to the ongoing process in advancing medical imaging technologies, particularly in the domain of lung disease diagnosis with the help of CNN.

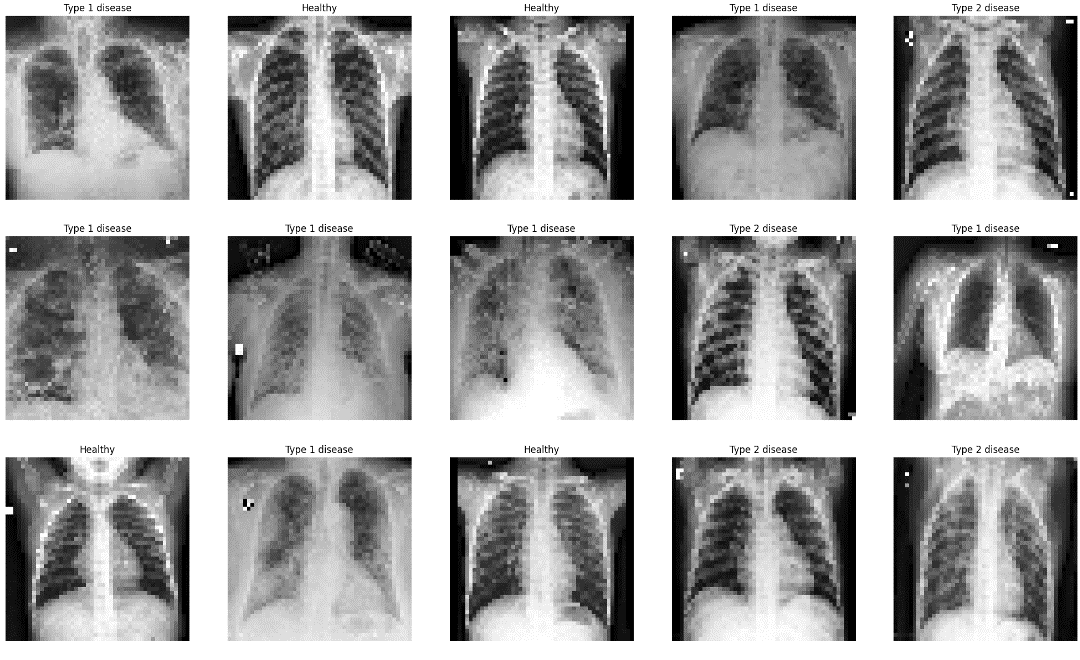
# **2. Data Preprocessing**

Before processing the data, it is important to check missing values but as the data consist of images and the report aims on CNN there cannot be missing values like in tabular data. The only missing values there can be are labels of images, fortunately there are 0 missing. It is important to make the machine read image files as in numbers and put the numbers in a data frame. The first step is to identify images in labels. Making a data frame from the images by concatenating the label and an empty list which will add containing images. This step must be used for training and test datasets. The initial datasets are split in 80 / 20 percent respectively for training and test datasets. This part prepares training data by collecting file paths and labels into a data frame, which can be used for training the CNN model. Later in the report it will be also divided into validation set. After, understanding the distribution of the datasets is necessary just for a visual picture.



***Fig.1:*** *Distribution of diseases in labels for each dataset.Top of Form*

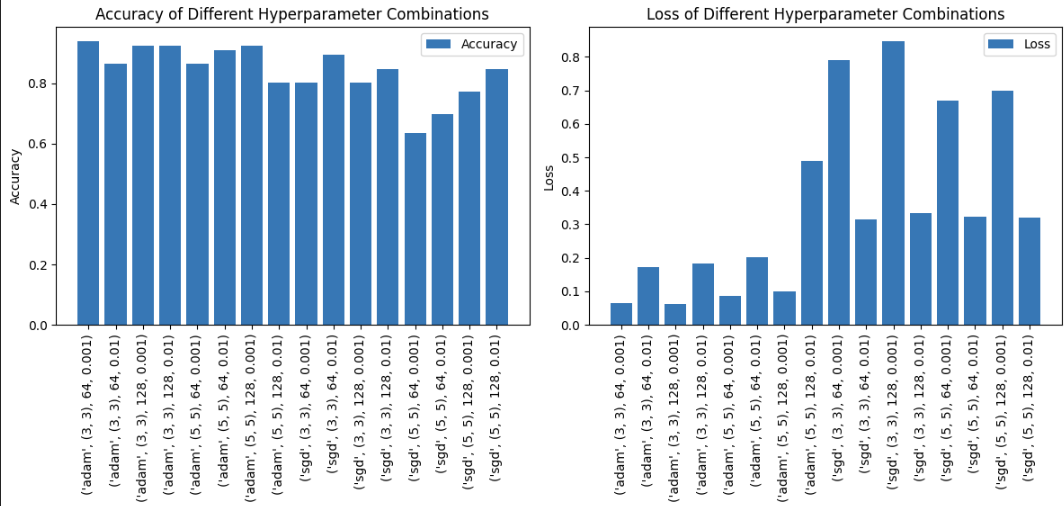
It is important to prepare training data by collecting file paths and labels into a DataFrame, which can be used for training CNN model. Top of FormThis visualization provides us with the distribution of labels in the training dataset, which is crucial for understanding the class distribution and for addressing class variations during model training. One can use this information for further analysis or visualization to gain insights into the data. In order to read the files properly it must take an image file path, read the image, resize it to a square shape with the specified size, convert its color space to RGB, normalize the pixel values, and return the preprocessed image with the preprocess function. This type of preprocessing is often used as a base step before feeding images into machine learning models. The next step is to load image data from the specified directory, preprocess the images, perform data augmentation (if training), and return the preprocessed data along with their corresponding labels. You can see what the data looks like after processing the images [1], [2].



***Fig.2:*** *Images with labels.*

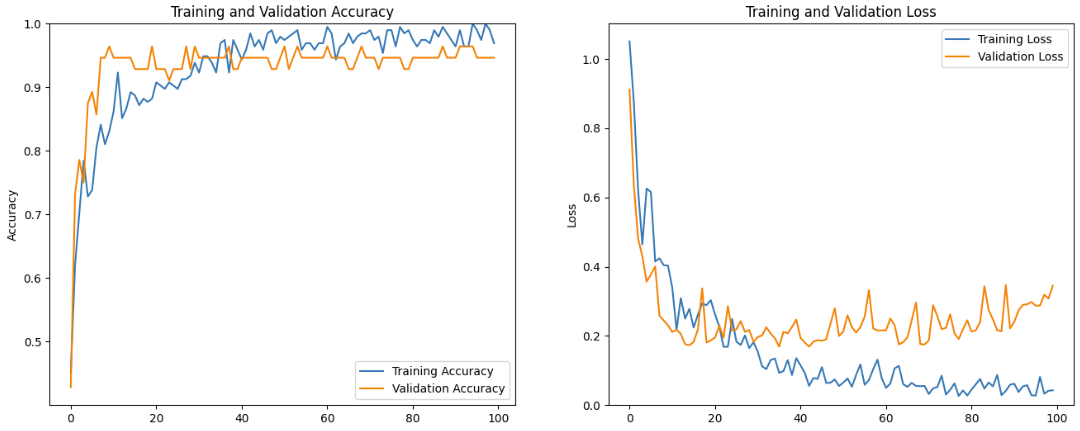
# **3. Training the data**

Before building the model, it is necessary is to ensure reproducibility in experiments involving random number generation, which can be crucial for debugging, testing, and comparing different machine learning models or algorithms. The model building starts with converting categorical labels into a binary format (0 and 1) called encoded labels for train, test and validation sets. In this part the model will be trained on validation set to see how does it perform in order to fine tune the model for testing. Validation set was split from training set having 19 percent of the overall data. The purpose is to find the best combination from different hyperparameters defined from the start such as optimizers, kernel sizes, filter sizes, and learning rates, all of them having 2 value options. So, the overall number of combinations will be 2 to the power of 4 which is 16. The model took too much time with more than 4 options of hyperparameters that is why only 4 hyperparameters will be used. Some hyperparameters were added manually in the same model such as pooling, flattening, number of epochs or activation function. An optimizer is used to change the attributes of neural network such as weights and learning rate in order to minimize the losses. Kernel size is the size of the convolutional filter. Filter size determines the number of filters to apply in the convolutional layer. Learning rate controls how much we are adjusting the weights of network with respect to the loss gradient. Activation function is used to learn complex patterns in the data: ReLu is used to fight the vanishing gradient problem and at last layer softmax is used for multi-class classification that convert scores to probabilities. Given the hyperparameters, the code goes through all and finds the best combination out of all 16 models. The best combination can be seen in the figure below.



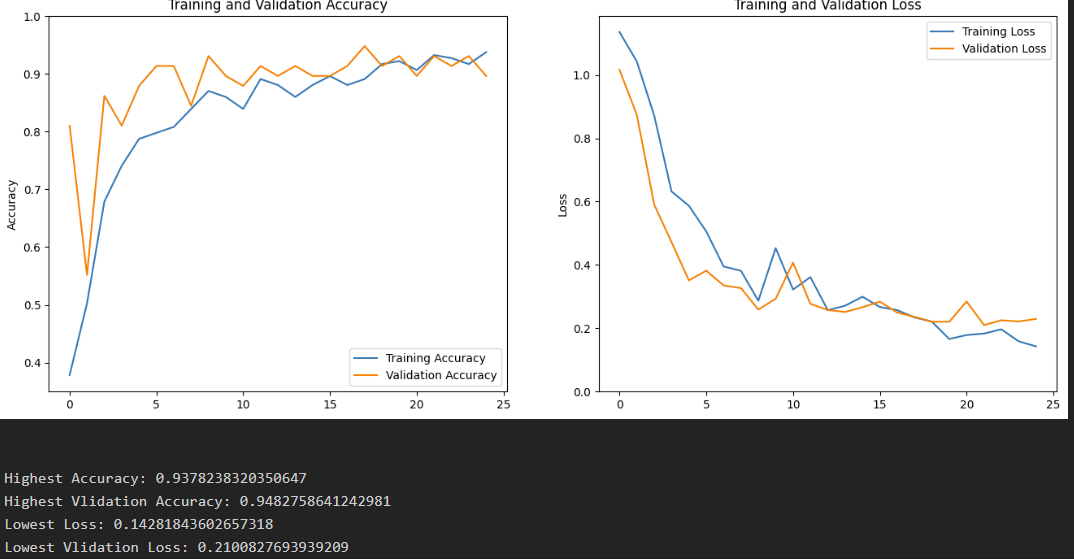
***Fig.3:*** *It shows accuracy and loss for each possible combinations and the best combination from 64 options is: optimizer = adam, kernel = 3,3, filter = 64, learning rate = 0.001.*

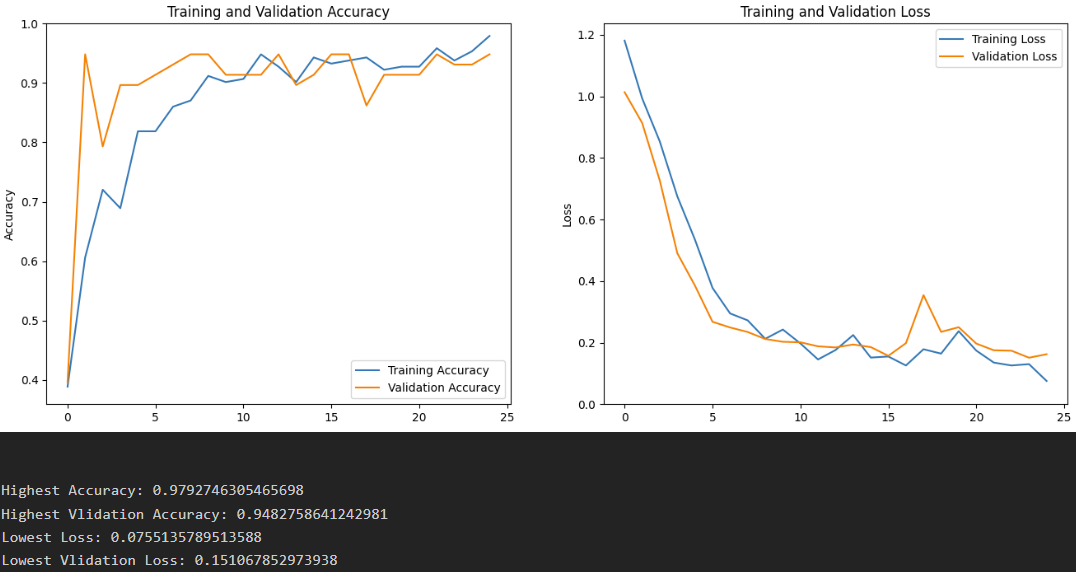
After, manually adding all parameters in a model such as (dropout level, padding, also change the size of epochs and polling method) which will be trained by having a key metric as accuracy. And here the model is created with more suitable hyperparameters and here it is important to find out the difference of results in max and avg pooling, adding or not adding padding, and checking different values in dropout rate. Since in CNN in Conv2D does not specify the number of neurons, instead it wants us to specify the filter size and after flattening it is important to provide the number of neurons for a dense layer which is 64 and finally 3 layers in the output layer for classifiers. When the model is trained it can be seen that from one-point validation loss graph goes up and train loss goes down. It means that the model is overfitting and the epoch size shall be reduced to approximately 25 to avoid overfitting (See figure 3 below) [3], [4].



***Fig.4:*** *Shows the accuracies and losses for train and validation sets.*

After changing the epoch size to 25, adding padding, having a dropout rate as 0.5, keeping number of neurons 64 and changing from avg pooling to max pooling graph is clearer and the accuracy is higher. As random seed is changing the values each time, this model needed to be tested multiple times to avoid confusions and to see the result different times to be sure that new mentioned hyperparameters works better on this model and that the overall model with all its hyperparameters is working precisely. That is why random seed is specified to have different results and not rely on just one model’s result [5].





***Fig.5:*** *Difference in AVG pooling and MAX pooling, 0.3 and 0.5 dropout rate, with padding and without, Max pooling, 0.5 dropout rate and using padding has a very good result.*

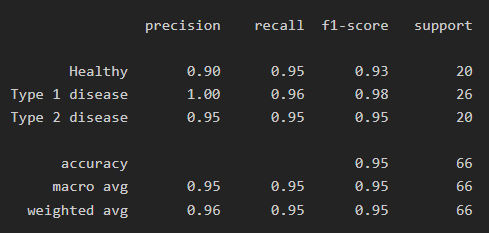
In the image above you can see the critical values (max and min) for accuracy and loss. This means where the model was performing the best according to epochs. As the epoch size was reduced according to overfitting, by looking at the graph it is clear that the best epoch was the 25th one. When adding a batch size to this model, the accuracy goes down and the predictions keep getting worse, instead of 3 False predictions it gives more than 5 or 6, which means that the batch size shall remain the default size in this case, based on dataset size.

# **4. Model performance**

After picking the very best model it is important to evaluate model only on a part of the whole dataset. The purpose of this is to provide a quicker assessment of the model’s performance. Therefore, only first 20 batches will be used for evaluation. Next job was to change the encoded labels to the original labels in order to understand the results and each prediction is converted into a one-hot encoded format and stored in a list [6].

# **5. Testing the data**

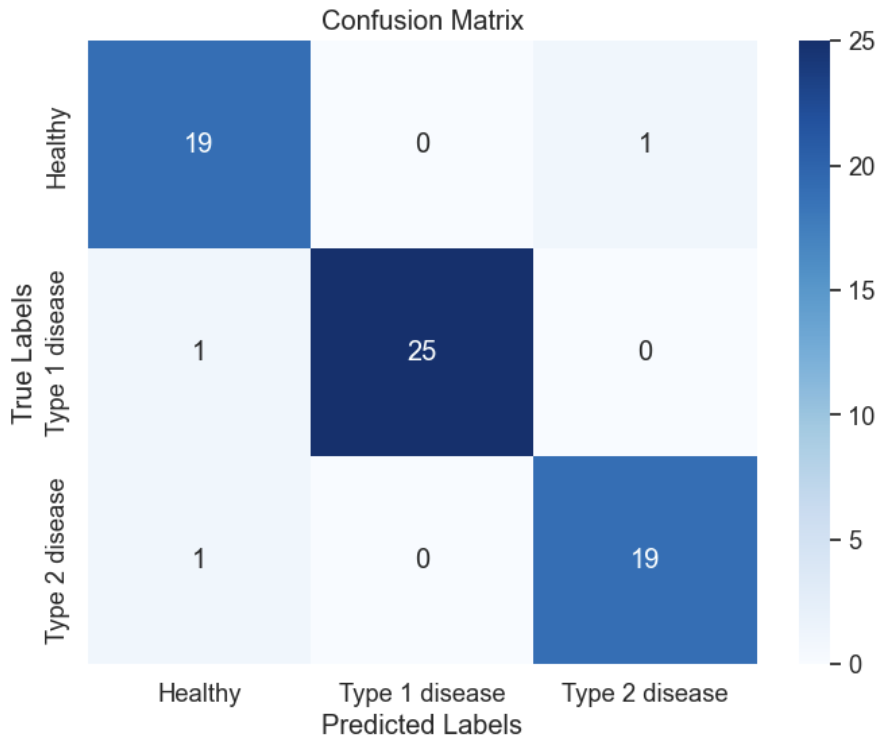
When it is time to test the data, prediction function takes a set of input images, predicts their labels using a pre-trained model, converts the binary-encoded predictions back to their original categorical labels, and returns the predicted labels. Before conducting a confusion matrix, there is another matrix shat shows the metrics for classifiers. The table below shows that the accuracy of the model is 95 percent, and basically all of metrics are quite high which indicates that the model is working well and we can rely on the model when testing on new datasets [7].



***Fig.6:*** *Best metrics for the model so far.*

Precision shows how often model is correct when predicting the class, it shows the accuracy of positive predictions, where Recall identifies all positive instances out of all actual positive instances in the dataset. Meanwhile, F1 score is the harmonic mean of these 2 measures. For accuracy, it shows how often the model is predicted correctly.

# **6. Confusion matrix**



***Fig.7:*** *Only 3 False predictions with Max pooling and all other hyperparameters.*

After predictions confusion matrix is shown to understand predictions such as False Positives/Negative, True Positives/Negative.

Healthy (True Label: Healthy): There are 20 instances of Healthy (label 0) in the test set. Out of 19 instances are correctly classified as Healthy (true positives), and 1 instance is incorrectly classified as Type 1 disease (false negative).

Type 1 disease (Actual Class 1): There are 26 instances of Type 1 disease (label 1) in the test set. Out of 25 instances are correctly classified as Type 1 disease (true positives), and 1 instance is incorrectly classified as Healthy (false negative).

Type 2 disease (Actual Class 2): There are 20 instances of Type 2 disease (label 2) in the test set. Out of 19 instances are correctly classified as Type 2 disease (true positives), and 1 instance is incorrectly classified as Healthy (false negative).

In summary, this confusion matrix provides insight into how well the model performs in classifying each class, showing the distribution of correct and incorrect classifications [8].

# **7. Conclusion**

In conclusion, this research talks about going into the critical domain of early detection and classification of lung diseases using deep learning approaches using medical imaging data. The report indicates the challenges of accurate diagnosis, localization of lung cancer, and classification of disease types through data preprocessing, visualization, and model development. By automatically and manually exploring various hyperparameters and model configurations, it can be identified optimal settings that improve the robustness and accuracy of our neural network model. The research also goes through significance of hyperparameter tuning, including optimization algorithms, kernel sizes, filter sizes, learning rates, and pooling methods, learning rate and etc. Through iterations of hyperparameters and evaluation of model performance, we can see the effectiveness of convolutional neural network structure in as much as possibly accurately classifying lung diseases specifically in cancerous regions within lung images. Furthermore, the use of confusion matrices and performance metrics has provided very valuable insights about the model's classification capabilities and areas for improvement. By using different hyperparameter settings, the report can contribute to the improvement of medical imaging technologies and the ongoing efforts to strengthen lung disease diagnosis.

**References**

[1] <https://github.com/iamtrask/Grokking-Deep-Learning>

[2] <https://www.superdatascience.com/blogs/the-ultimate-guide-to-artificial-neural-networks-ann>

[3] <https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn>

[4] <https://www.geeksforgeeks.org/vanishing-and-exploding-gradients-problems-in-deep-learning/>

[5] <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

[6] <https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning>

[7] <https://www.analyticsvidhya.com/blog/2020/11/popular-classification-models-for-machine-learning/>

[8] <https://github.com/DipakMajhi/Machine-Learning-CheatSheets/blob/master/Deep%20Learning%20Cheat%20Sheet.pdf>