Images Classification

1.Introduction

Image classification is gaining attention in the research field, especially in medical analysis. This report aims to investigate the performance of a Fully Connected Neural Network (MLP), a Convolution Neural Network (CNN) and a classification or regression algorithm when building an image classification. Image classification applications have a variety of usage that helps complete the task of categorizing and assigning classes to picture in pixels or vectors format. It is widely used in the medical and astronomy field, which is mainly used for classifying images like X-rays, MRI scans and possible galaxies. It is beneficial for the medical diagnosis accuracy and examining the unknown stars or galaxies in space.

2.Data

2.1. Data description and exploration

In this report, we are investigating OrganCMNIST (part of the MedMNIST suite), which is a dataset for 2D and 3D biomedical image classification that contains23660 images (18928 in the train set and 4732 in test set) of coronal CT scans of 11 body organs. Each image is divided into 28×28 pixels. Our targeting variables are 11 class labels ranging from 0 to 10. Each class of label represents a body organ being scanned. From Github, we found the name for each class label and we assigned them to each class. In addition, each of our predicting variables, pixel values, indicates the brightness of the pixel on that location. These values can span from 0 to 255, where 0 indicates complete black whereas 255 represents complete white. All other values are grayish colors in between. In ***Figure 1***, we visualize the first 20 images in the dataset to have a preliminary understanding about the dataset. The challenge that we face is distinguishing between classes, as we can barely obtain any useful information from visualizing due to the unfamiliarity with human organs.

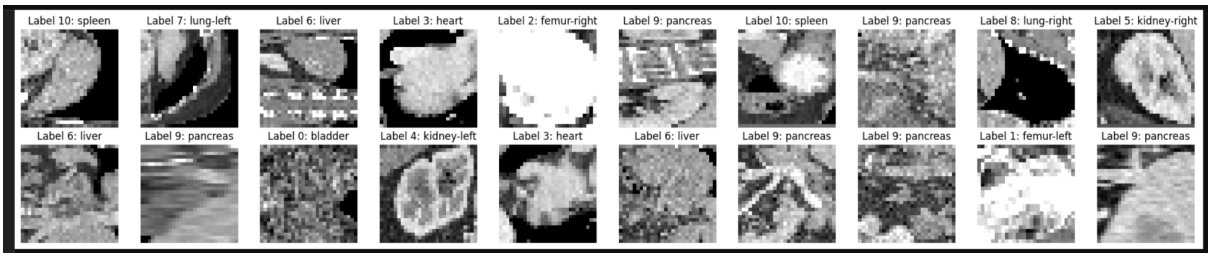

Figure 1: First 20 images in the dataset

By eyeballing these images, it seems like they are overall centered. After calculating the pixel intensity statistics(Minimum, Max, Mean and standard deviation) of each image we obtain a better sense of the brightness, variability and informality among them. For example, imagine 5 looks very bright, it indeed has a high mean intensity of 239.82. Image 4 and Image 9 have relatively high standard deviations (95.97 and 85.14, respectively), suggesting a higher level of variability in pixel intensity, indicating a more textured image.

2.2. Pre-processing

Before building the classification model, in order to mitigate the burden of our computer to run the model with a distinct classifier, more importantly, handling hyperparameter tuning in the classification model, we decided to normalize the pixel. Normalization can be done by dividing the pixel value by 255, it helps train the model faster. The next thing to do is to count the number of each class in both the train and testing dataset. ***Figure 2*** illustrates a disparity in counts between class 6 and other classes. It can potentially lead to a class imbalance. However, the distribution of each class is also similar between train and test sets. If they are randomly shuffled, we may assume this being the true pattern in the border population. Thus we leave the data as they are for the time being.

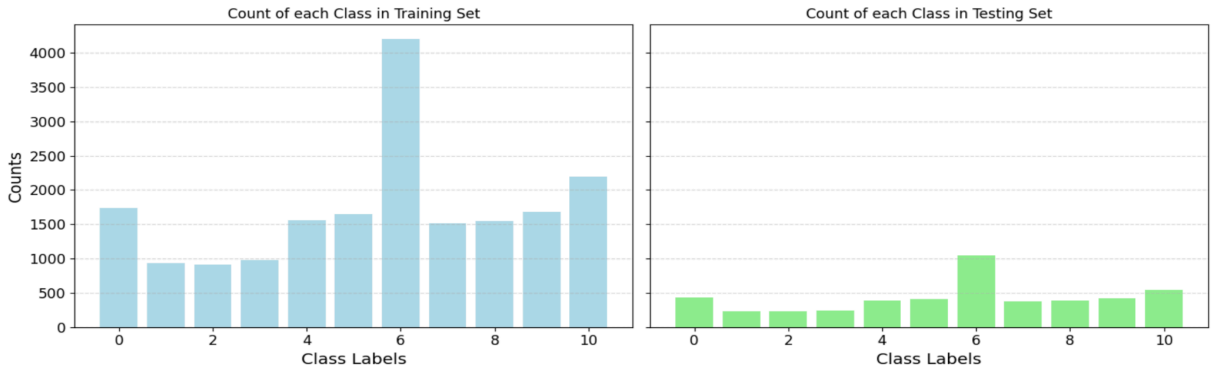

Figure 2: Side-by-Side barplot showing distribution of each class in training and testing set

3. Methods

In this part, we are going to build three models to perform image classification. The first model we choose is random forest. According to an article investigating image classification using machine learning, it stated that random forest performed the best against Naive Bayes (NB) and k-nearest neighbors (KNN) [1]. Based on finding, we tested the accuracy of all three classifiers, random forest showed the best accuracy score with 0.90, KNN scored 0.79 and NB scored 0.68. As a result, we decided to choose a random forest to be our classification model besides MLP and CNN.

3.1 Theory

3.1.1 Random Forest

Random forest is an ensemble method most used on decision trees. It relies on bootstrap sampling where each resample data using original sample with replacement. For each of the bootstrapped samples, we will build a decision tree using a proportion of the total attributes in the dataset at each step of splitting.This will increase the diversity among the trees, which enhances the model's robustness. For classification tasks, the prediction by the forest is the majority vote of the predictions made by the individual trees. By using random forest, we can reduce variance or overfitting by taking a majority vote from trees while not increasing bias using all the trees.

3.1.2 MLP

Multilayer Perceptron (MLP) is a type of artificial neural network (ANN) that consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node, or neuron, in these layers (except for input nodes) uses a nonlinear activation function. MLPs are well-suited for addressing problems that require consideration of complex patterns and relationships.An MLP consists of multiple layers. The input layer receives a raw input vector and each neuron represents one input from the utilized feature. One or more hidden layers followed by the input layer perform computations and feature transformations. The depth (number of layers) and breadth (number of nodes per layer) of the network can significantly influence its ability to model complex functions.The final layer, output layers produces the network's output, which can be a single value (for regression) or multiple values (for classification). A neuron or node is a non linear activation function that processes data from previous layers.

3.1.3 CNN

A Convolutional Neural Network (CNN) is another type of neural networks, highly effective for imagining data classification.. CNN can leverage the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. A typical CNN has a series of layers, each of which transforms one volume of activations to another through an activation function. Convolutional Layer, followed by the input layer is the building block of a CNN that performs a convolutional operation. This layer applies several filters to the input to create feature maps that summarize the presence of detected features in the input. Following each convolutional operation, an activation function such as ReLU is applied to introduce nonlinearity into the model, helping it to learn more complex patterns. After that a pooling layer is added to reduce the dimensionality of each feature

map while retaining the most important information. After several convolutional and pooling layers, the output reduced to smaller dimension will serve as input to a fully connected neural network, the same as MLP model.

3.2. Strengths and weaknesses

3.2.1 Random Forest

Random Forests, as a machine learning algorithm, offer unique strengths and weaknesses depending on the context of their application. On the one hand, Random Forests typically achieve very high accuracy in many scenarios, especially for complex classification problems, because it combines the predictions of several decision trees and reduces the impact of errors in any single tree. Secondly, unlike individual decision trees, Random Forests are less likely to overfit because they average the results of many trees, which individually may overfit different parts of the data. Thirdly, Random Forest can handle large datasets with high dimensionality. It can handle thousands of input variables without removing them, so it has an advantage in this high-dimensional image classification scenario.

However, there are also some disadvantages in using Random Forests during the process. Firstly, training a random forest model can be very time consuming because each tree is constructed independently and the complexity increases as the number of trees and the depth of each tree grows. Secondly, the performance of a random forest is heavily dependent on settings such as the number of trees, the depth of each tree, and the number of features considered for segmentation at each node. Tuning these parameters can be both time consuming and requires a good understanding of how they affect overall model performance.

3.2.2 MLP

The advantages and disadvantages of MLP will be described in the following sentences. On the one hand, MLP has strong nonlinear fitting ability and can handle complex input-output relationships. Secondly, tt can improve the performance of the network by increasing the number of hidden layers and neurons. Thirdly, it has good generalization ability and can avoid overfitting problems. Finally, it is suitable for large-scale data processing and parallel computing.

Nonetheless, there are certain drawbacks to using Random Forests in the process. Firstly, training can be slow and requires significant computational resources, particularly as the number of layers and neurons increases. Secondly, it is easy to fall into the local minimum and cannot find the global optimal solution.

3.2.3 CNN

CNN, as an advanced machine learning method, presents distinct advantages and disadvantages based on their application context. On the one hand, the convolution operation of CNN can effectively reduce the number of parameters of the model and improve the training speed and generalization ability of the model. Secondly, CNNs are great for processing 2D data such as images and videos, especially for the task of image classification.

However, there are certain limitations to using CNN in the process. Firstly, CNNs have a large computational complexity and require a lot of computational resources, especially in scenarios with large-scale models and large datasets. Secondly, CNNs have difficulty in extracting global information. Since the convolution operation is used in processing the image, the processing of global information in the image is relatively weak.

3.3. Architecture and hyperparameters

In this part, we are discussing the steps for the architecture and hyperparameter tuning.

Random Forest

Random forest is a combination of distinct decision trees, each tree is constructed independently and randomly sampled with replacement. ***Figure 3*** describes the architecture of a random forest, indicating that the dataset is split into a training and testing set. For the training set, first split n training data, then create a decision tree for each subset. From each decision tree, choose the model. By majority voting, make the prediction.

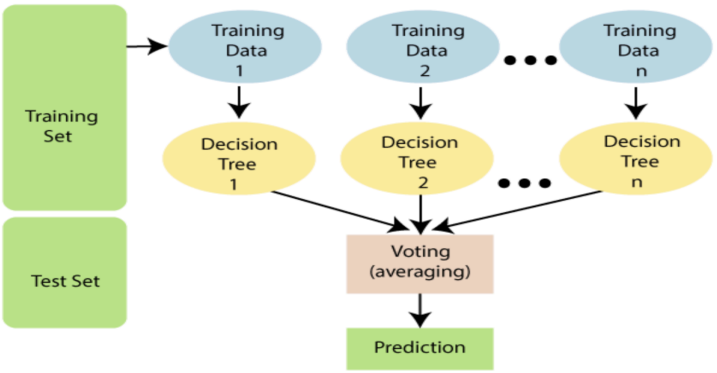

Figure 3: Architecture of Random Forest [2]

We decided to use *RandomForestClassifier* from *sklearn.ensemble* and *GridSearchCV* from *sklearn.model_selection* to perform the classification. Considering the processing time for the code, only 3-fold cross-validation is used. In terms of hyperparameter tuning, we decided to choose hyperparameters that have a huge impact on the model, such as *n_estimators*, *min_samples_split and max_depth*. The range of Random Forest hyperparameter (***Figure 4***):

n_estimators	-	50	100
min_samples_split	None	10	20
max_depth	2	10	20

Figure 4: Table for best hyperparameter value for Random Forest

MLP

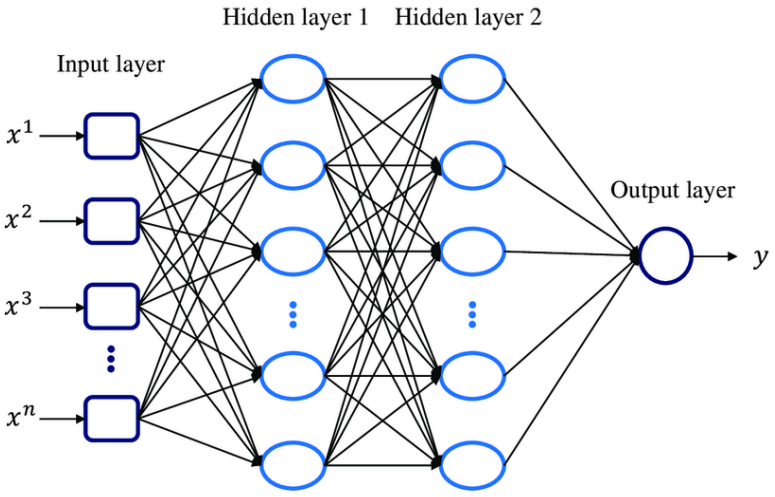

Figure 5: Architecture of MLP [3]

MLP starts with the input layer, propagates data forward to the output layer, then calculates the error based on the output—primarily the difference between the predicted and known outcome—which needs to be minimized, and finally back propagate the error to find its derivative with respect to each weight in the network and update the model [4]. ***Figure 5*** illustrates an input layer that connects to every hidden layer , and produces an output layer based on the final hidden layer. Output layer will give the classification result. First define a Feedforward neural network, then implement fixed parameters with *neurons1*, *neurons2*, *epochs*, *batch_size* inside *CustomKerasClassifier*. Then set the hyperparameter, use *GridSearchCV* to find the best hyperparameter combination. Considering the processing time for the code, only 3-fold cross-validation is used. In terms of

hyperparameter tuning, we decided to choose hyperparameters that have a huge impact on the model, such as *optimizer, batch_size and epochs*. The range of MLP hyperparameter (***Figure 6***):

optimizer	adam	sgd
batch_size	200	500
epochs	20	50

Figure 6: Table for best hyperparameter value for MLP

CNN

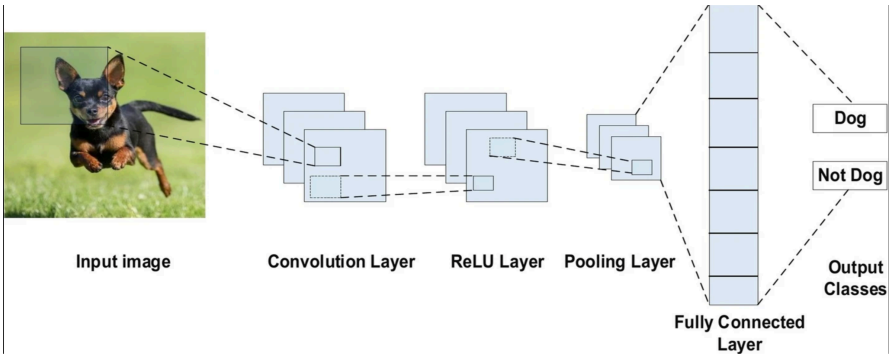

Figure 7: Architecture of CNN [5]

From ***Figure 7***, the architecture of CNN is divided into the following key steps or layers, and then shows the conversion from image input to classification output sequentially. Step 1: Input an image of a dog with areas of interest marked, which we want the CNN to recognize and process. Step 2: The input image is passed to the first convolutional layer, which outputs an activation map. The filters in the convolutional layer extract relevant features from the input image and pass them forward. Step 3: After the convolutional layer, a ReLU layer is applied. The ReLU layer introduces non-linearity to the model by replacing all negative pixel values in the feature map with zero. Step 4:After the ReLU layer, a pooling layer is used. This layer mainly reduces the spatial size of the incoming feature map and reduces the number of parameters, thereby reducing the computational load. Step 5: The output layer of CNN is a fully connected layer, and inputs from other layers are transformed and transmitted in order to obtain the desired classification output through the network. Step 6: The final output of the CNN shows two categories: "dog" and "non-dog". The output layer uses an activation function to calculate the probability that the input image belongs to each category. First build a convolution neural network model with optimizer 'adam', then define the CNN model with KerasClassifier. After that, create a hyperparameter grid containing optimizer, batch_size, epochs, then use GridSearchCV to find the best hyperparameter combination. Considering the processing time for the code, only 3-fold cross-validation is used. In terms of hyperparameter tuning, we decided to choose hyperparameters that have a huge impact on the model, such as *optimizer, batch_size and epochs*. The range of CNN hyperparameter (***Figure 8***):

optimizer	adam	sgd
batch_size	200	500
epochs	20	50

Figure 8: Table for best hyperparameter value for CNN

4.Results and discussion

Random Forest

max_depth	20
min_samples_split	2
n_estimators	50

Figure 9: Table for best hyperparameter value for random forest

The runtime for random forest is approximately 16 minutes, when *n_estimators* = 500, the runtime increased to over 30 minutes. Therefore, we decided to set {10 30 50} as the range of n_esitmators. After searching for the best combination of hyperparameters, it is the best parameter of the random forest. We can observe that with an increase in *n_estimators*, the model becomes more predictive. However, *min_samples_split*, has an opposite trend, the higher the value of *min_samples_split* the poorer the performance of the result. Additionally, *max_depth* has a similar trend as *n_esitmators*, with the higher value of *max_depth*, the better the model performs.

MLP

batch_size	200
epochs	50
optimizer	sgd

Figure 10: Table for best hyperparameter value for MLP

The runtime for random forest is approximately 34 seconds, when we add one more optimizer, the runtime increased to over 20 minutes. Therefore, we decided to include only two *optimizers* for hyperparameter tuning. After searching for the best combination of hyperparameters, it is the best parameter of the MLP. From grid search, sgd is the better optimizer for this dataset, with smaller the *batch_size*, better the performance and the bigger the *epochs*, the better the model.

CNN

batch_size	200
epochs	50
optimizer	sgd

Figure 11: Table for best hyperparameter value for CNN

The runtime for random forest is approximately 17 minutes, when we add one more optimizer, the runtime increased to over 40 minutes. Therefore, we decided to include only two optimizers for hyperparameter tuning. After searching for the best combination of hyperparameters, we discovered that the best parameter of CNN is the same as MLP, which indicated that the value of hyperparameter is suitable for image classification.

	F1 score	Precision	Recall
Random Forest	0.8913	0.8943	0.8922

MLP	0.8660	0.8778	0.8677
CNN	0.9389	0.9298	0.9389

Figure12: Table for F1 score, Precision, Recall for random forest, MLP and CNN

Figure 12 compares the performance of random forest, MLP and CNN based on three evaluation measurements: F1 score, Precision and Recall. CNN has the best performance among all three measurements. MLP has the second best performance for all three measurements and random forest has the worst performance among all measurements. CNN is the best performing model, showing superior ability to identify and classify the correct categories accurately. Random Forest and MLP perform commendably but do not reach the effectiveness of the CNN in this evaluation. This analysis suggests that CNN might be more suitable for tasks where both precision and recall are critically important.

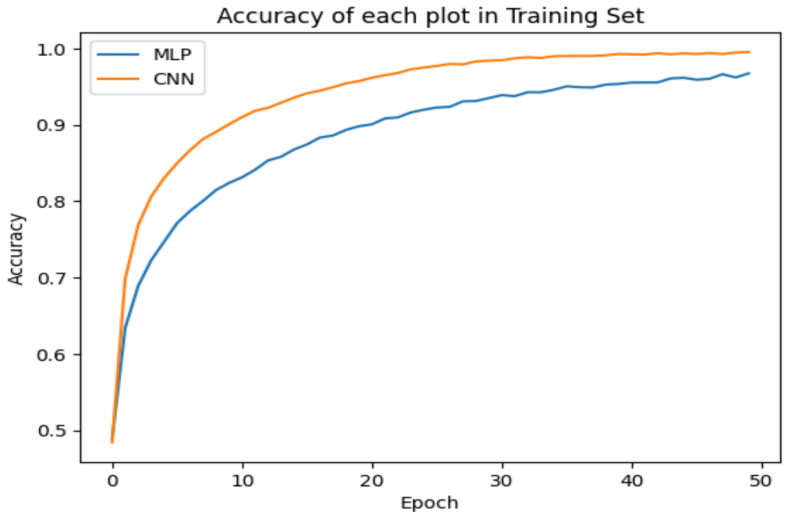

Figure 13: Line plot illustrating the accuracy of MLP and CNN for different value of epoch

From a report investigating the comparison of the performance of CNN and MLP in image classification, it stated that CNN has lower cross-entropy, which makes it a better classifier than MLP [6]. According to ***Figure 13***, it shows that CNN enjoys more accuracy with the increase in number of epochs, it gets the same justification as the report that support our research result.

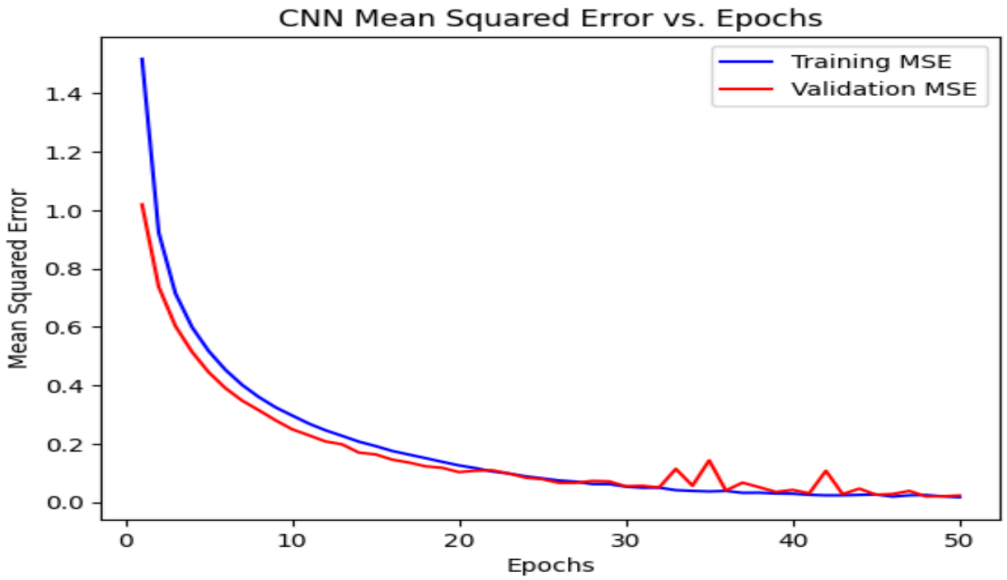

Figure 14: Line plot illustrating the differences of Training MSE and Validation MSE from CNN

Figure 14 shows effective learning dynamics as both training and validation errors decrease. The model appears to be well-tuned in the training set without significant evidence of overfitting. If needed, hyperparameters could be further tuned to see if the validation MSE can be reduced further or if the model can learn faster. Adjustments could include changing the learning rate, modifying layer configurations, or experimenting with different optimizers.

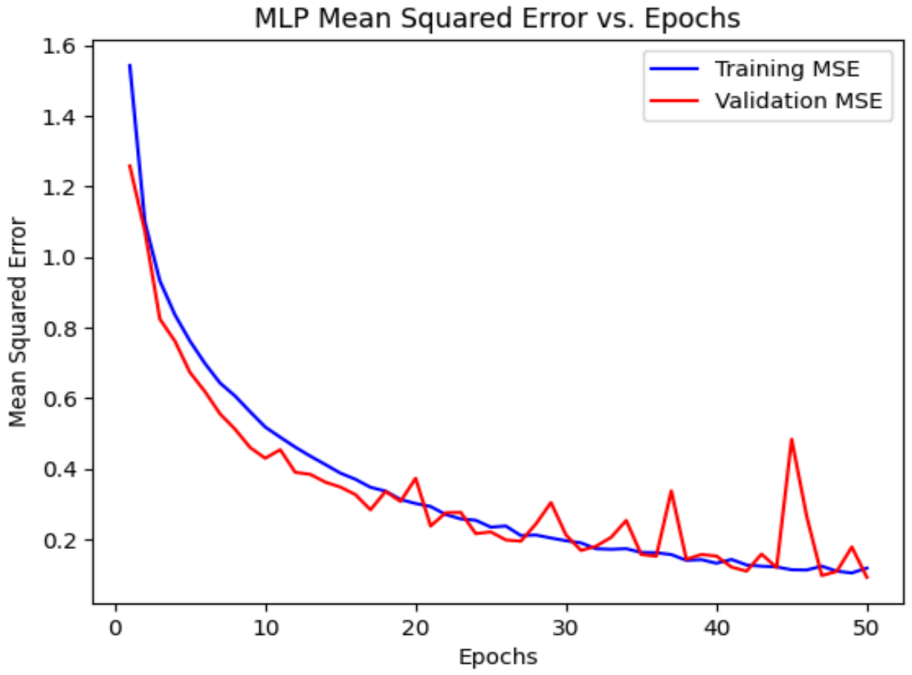

Figure 15: Line plot illustrating the differences of Training MSE and Validation MSE form MLP

Figure 15 shows the Mean Squared Error (MSE) for MLP final model during training over 50 epochs. Both the training and validation MSE decrease sharply in the initial epochs. This indicates that the model is quickly learning from the training data, and significant learning is occurring, which is typical as the model begins to fit to the data structure.The sharp decrease suggests that the learning rate and model architecture are effectively capturing the underlying patterns in the data without major initial hurdles. Additionally, there are spikes in the validation MSE at later epochs (notably around epoch 45), which is somewhat unusual. This could indicate that the model encounters some samples in the validation set that are significantly different from what it has learned so far, or it might be reacting to minor overfitting or instability in the learning process.

5.Conclusion

The analysis clearly demonstrates that CNN outperforms the MLP and Random Forest in terms of F1 score, precision, and recall, making it the superior model for tasks requiring high accuracy, such as image and speech recognition. The CNN architecture, designed to mimic the human visual cortex, allows it to excellently capture hierarchical patterns in complex inputs, which contributes to its high performance.

While MLP offers faster runtime due to its simpler architecture, it does not achieve the same level of accuracy as the CNN. Random Forest, although robust and less prone to overfitting, also falls short in comparison. Thus, while CNN requires more resources and training time, their effectiveness in precise classification makes them the preferable choice for critical tasks, whereas MLP might be suited for scenarios where speed and resource efficiency are prioritized.

During our research on multiple models, we found that although *GridSearchCV* could help us find the optimal hyperparameters, it included hyperparameter tuning that was restrictive on computer hardware, resulting in model training taking too long. To speed up our training process, *RandomizedSearchCV* from

sklearn.model_selection is a method worth considering for investigating hyperparameters, as it has faster processing speed.

6.Reflection

According to this report, we realize that time management is crucial. We underestimated the running time of MLP and CNN, so we were pressed for time on hyperparameter tuning, such as trying higher parameter values. While trying different hyperparameter combinations may lead to better model results, we chose to prioritize faster runtimes at the expense of some model complexity.

In addition, it is important for us to understand the specific characteristics and requirements of the data when choosing the appropriate model. Because different models need to configure corresponding environments and libraries for machine learning, otherwise errors will occur.

7.References

[1]. *V, N. (2023, April 5). Image Classification using Machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/01/image-classification-using-machine-learning/#:~:text=Random%20Forest%20Classifier%20shows%20the*

[2]. *JavaTpoint. (2021, March 31). Machine Learning Random Forest Algorithm - Javatpoint. Www.javatpoint.com. https://www.javatpoint.com/machine-learning-random-forest-algorithm*

[3].Sarraf Shirazi, Alireza & Frigaard, Ian. (2021). SlurryNet: Predicting Critical Velocities and Frictional Pressure Drops in Oilfield Suspension Flows. Energies. 14. 1263. 10.3390/en14051263.

[4]. *Banoula, M. (2021, May 20). An Overview on Multilayer Perceptron (MLP). Simplilearn.com. https://www.simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron*

[5]. *Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data **8**, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8*

[6]. *Damle, A. (n.d.). Comparing the Performance of CNN and MLP in Image Classification.*