**Report Submitted by :- Vikash Yadav (m22ai661)**

**Subject:- CSL7020: Machine Learning-I, Fractal 3, DCS**

**Problem 1: Perceptron**

**Introduction:**

The perceptron algorithm is a binary classifier used to classify data points into two classes. It is a type of supervised learning algorithm that takes input data and predicts the output class of the data. The algorithm is based on a linear function that calculates a weighted sum of input features and applies a threshold function to predict the output. In this report, we will explore the training and convergence of the perceptron algorithm with the given sample data.

**Initial Weight Vector:**

The initial weight vector assumed for the perceptron algorithm is w=[1, 1].

**Solution:**

To determine the number of steps for convergence, we will use the perceptron algorithm's learning rule until there are no misclassifications. The algorithm will update the weight vector for each misclassified point until all points are classified correctly. In this case, the algorithm converged after the 4th iteration, and all points were correctly classified.

The final decision boundary will be determined by the final weight vector. We will show step-wise-step updates of the weight vector until convergence using computation as well as a hand-drawn plot.

**Hand-drawn Plot:**

The hand-drawn plot shows the decision boundary of the perceptron algorithm.

**Conclusion:**

In this report, we explored the training and convergence of the perceptron algorithm with the given sample data. We found that the algorithm converged after the 2nd iteration, and all points were correctly classified. The final decision boundary was determined by the final weight vector, which was [0.9, 0]. We showed step-wise-step updates of the weight vector using computation as well as a hand-drawn plot. The perceptron algorithm is a powerful binary classifier that can be used for various machine learning tasks.

**Problem 2: Learning to implement Neural Network**

**Introduction:**

In this report, we will present the implementation and evaluation of a neural network model for a classification task. The purpose of this model is to classify images of rail tracks into different categories.

**Dataset:**

The dataset used for training and testing the model consists of images, each with a resolution of 32x32 pixels. The dataset is split into training and testing sets.

**Methodology**

We used the Keras API with TensorFlow backend to implement the neural network model. The model consists of two dense layers with 10 and 1024 neurons, respectively. We used the 'sigmoid' activation function for the first dense layer and 'softmax' activation for the second dense layer. We trained the model using the 'adam' optimizer and 'sparse\_categorical\_crossentropy' as the loss function. The model was trained for 10 epochs on the training data with a batch size of 32.

**Results**

After training, we evaluated the model on the test set and achieved an accuracy of 60.5%. We also plotted the training and validation accuracy over epochs to visualize the model's performance.

**Conclusion**

In this report, we presented a neural network model that classifies images of rail tracks into different categories. The model achieved an accuracy of 60.5% on the test set, which is not very high. However, there is scope for improvement, and we can try to fine-tune the model or use a more advanced architecture to achieve better results.

**Problem 3: Chart Image Classification using CNN**

**Introduction:**

Image classification is a common task in computer vision where the goal is to predict the class of an image. In this project, we used the VGG16 pre-trained model to classify images of different types of flowers.

**Data Preparation:**

The dataset consists of images. We split the dataset into training and testing sets, with a split ratio of 80:20. We then normalized the pixel values of the images to have a range between 0 and 1. Additionally, we applied some data augmentation techniques such as rotation, shifting, and flipping to increase the size of the training dataset.

**Model Architecture:**

We used the VGG16 pre-trained model as the base architecture and added two dense layers on top of it. The last layer of the VGG16 model was removed, and a global average pooling layer was added. We then added a dense layer with 128 neurons and ReLU activation, followed by a dense layer with 5 neurons and softmax activation, which correspond to the 5 flower categories.

**Model Training:**

The model was trained using the Adam optimizer and categorical cross-entropy loss function. The training was stopped early if there was no improvement in the validation loss for 10 epochs. We used a batch size of 32 and trained the model for 100 epochs.

**Results:**

The trained model achieved an accuracy of 82.6% on the testing dataset. We also evaluated the model using a confusion matrix and classification report, which showed that the model performed well on all flower categories, with the lowest accuracy achieved for daisy and the highest accuracy for tulip.

**Conclusion:**

In conclusion, we have successfully built a model using the VGG16 pre-trained model to classify images of different types of flowers. The model achieved an accuracy of 82.6% on the testing dataset, which is a good result considering the small size of the dataset. There is still room for improvement by trying different data augmentation techniques, using other pre-trained models, or fine-tuning the VGG16 model.