**IMAGE CLASSIFICATION USING THE CIFAR-10 DATASET**

*Authors: Indhuja Muthu Kumar, Venkat Srinivasa Raghavan*

**1. OBJECTIVE:**

The CIFAR-10 dataset is a widely used benchmark dataset for image classification tasks. It consists of 60,000 images, with 50,000 images used for training and 10,000 images used for testing. Each image is a 32x32 pixel RGB image, with 10 possible labels corresponding to the 10 different object classes. The objective of this project is to build four different classification models to accurately classify the images into their respective classes. Specifically, Support Vector Machines (SVM), Random Forest Classifier, Deep Neural Networks (DNN) with Convolutional Neural Networks (CNN), and DNN without CNN were utilized for classification. The performance of each model was evaluated using standard evaluation metrics, such as accuracy, precision, recall, and F1-score, on the test set. Additionally, the convergence speed of the models during training will be tracked by measuring the time taken to reach a certain accuracy level.

**2. METHODS**

**2.1. CONVOLUTION NEURAL NETWORK:**

A convolutional neural network (CNN) is a type of artificial neural network designed to process and analyze images. It is composed of one or more convolutional layers, which apply a set of filters (also known as kernels) to the input data. The filters convolve across the input data, performing a mathematical operation that highlights certain features or patterns in the data.

***2.1.1 Data Preprocessing*:** The initial stage of any machine learning analysis involves data preprocessing. To begin with, the CIFAR-10 dataset was imported, and the images were transformed into a format that is compatible with machine learning algorithms. Additionally, the pixel values were normalized to enhance the speed of convergence of the algorithms.

***2.1.2 Feature Extraction:*** The Convolutional layers aid in extracting features from the images. Specifically, a pre-trained VGG16 model was used to extract features from the images, and a fully connected neural network was trained on top of the extracted features. Furthermore, the most used pooling technique, max pooling, was utilized in the model. It also aids in extracting the most important and relevant information from the previous convolutional layer, which makes the network more efficient and robust.

***2.1.3 Model Building*:** Upon extracting feature, the model is built by introducing activation functions. Since convolution is a linear operation that may not effectively capture the nonlinear relationships in image data, non-linear activation functions such as the Rectified Linear Unit (ReLU) has been to the model applied after convolutional layers to introduce nonlinearity into the activation maps. After training, the pre-trained VGG16 model was fine-tuned using SGD with Cross-entropy loss function.

***2.1.4 Evaluation*:** The performance of the trained model was then evaluated on the test set based on the accuracy metrics like precision, recall, and F-1 score. The test loss for each epoch was also tracked and the overall accuracy of 81% was obtained on the test data.

**2.2 DNN without CNN**

A Deep Neural Network (DNN) is a type of neural network that can be used to process a variety of data types, including images. Unlike CNN, DNN is not specialized for image processing and does not use convolutional or pooling layers. Instead, a DNN typically consists of fully connected layers that are stacked on top of each other.

***2.2.1 Data Preprocessing*:** Once the CIFAR-10 dataset was imported, the images were then transformed into a format that is compatible with DNN. Additionally, the pixel values were normalized to enhance the speed of convergence of the algorithm and to mitigate issues arising from weight matrix initialization.

***2.2.2 Feature Extraction*:** Unlike DNN with CNN, the DNN model doesn’t include Convolution layers to extract appropriate features. Once the features are engineered, they can be fed into the DNN as input data. The DNN then learns to map the input features to the desired output through a series of layers that perform nonlinear transformations on the input data.

***2.2.3 Model Building*:** Upon extracting feature, the model is built by introducing activation functions. To achieve this, the model has utilized non-linear activation functions such as the Rectified Linear Unit (ReLU) after the convolutional layers. This aids the network to capture more complex relationships in the data by introducing non-linear transformations. After training, the fully connected layer was fine-tuned using SGD with Cross-entropy loss function.

***2.2.4 Evaluation*:** To evaluate the performance of the trained model, accuracy metrics such as precision, recall, and F-1 score were used to assess its performance on the test set. Additionally, the test loss for each epoch was monitored, and the final model achieved an overall accuracy of 45% on the test data.

**2.3. RANDOM FOREST CLASSIFIER**

The Random Forest algorithm is an ensemble technique that employs multiple decision trees to construct a stronger and more precise model. The principle behind random forest is to create a forest of decision trees, where each tree is trained on a random subset of the training data (bootstrapped data) and a random subset of features. Once the trees are constructed, they each independently classify the data, and the algorithm combines their predictions to obtain a final prediction.

***2.3.1 Data Preprocessing*:** The process of data preprocessing for the CIFAR-10 dataset involved utilizing the "load\_data" function available within the Keras library. This function was utilized to load the dataset and subsequently split it into both training and testing sets. To enable compatibility with the random forest classifier, the 3D image arrays were flattened to 2D. Further, to ensure consistency and comparability, the pixel values were normalized by dividing them by 255.

***2.3.2 Modeling*:** In the modeling stage, the random forest classifier is constructed by initializing the "RandomForestClassifier" class from the “sklearn.ensemble” library. The training of the classifier is then executed by fitting it with the training data. Next, the classifier randomly selects a subset of features and bootstraps the samples to distribute the data across each decision tree within the forest. The final step involves predicting the test data by utilizing the "predict" function.

***2.3.3 Evaluation*:** The evaluation of the model is done by calculating the accuracy. The confusion matrix is also plotted to find the number of true and false positives and negatives predicted. Finally, the precision, recall and f1-score of the predictions are found using the classification report.

**2.4. SUPPORT VECTOR MACHINE**

Support Vector Machines (SVMs) are a class of machine learning algorithms that can be employed for classification tasks. One of the key strengths of SVMs is their capacity to learn intricate associations between features and labels, even in cases where the data is not linearly separable.

***2.4.1 Data Preprocessing*:** To begin with, the data is loaded into the model by utilizing the "load\_dataset" function available within the Keras library. Once the CIFAR-10 dataset was imported the images were flattened from a 32x32x32 tensor into a 1024-dimensional vector. This has been done to make the data more manageable for the SVM classifier. The data is then split into training and test data.

***2.4.2 Feature Extraction*:** In the feature extraction stage, dimensionality reduction was performed through Principal Component Analysis (PCA). The purpose of this technique is to reduce the dimensionality of the data while preserving as much information as possible. The primary benefit of this step is to improve the computational efficiency of the SVM model. The PCA was conducted using the "PCA" function from the sklearn.decomposition library, where the number of principal components was determined to be 100. This approach enables the identification of the most salient features of the dataset while minimizing the effects of noise and redundancy in the data.

***2.4.3 Model Building*:** In the model building stage, a non-linear SVM classifier was created by initializing it with a 'rbf' kernel. The regularization parameter C was set to 10, and a random state of 42 was specified. To combine the PCA and SVM in a single process, a pipeline was constructed. The pipeline is a sequential set of operations where the output of one step is used as the input for the next. In this case, the pipeline included PCA as a preprocessing step followed by SVM. The pipeline was then fitted with the training data, with an additional scaling step performed using 'StandardScaler'. Finally, predictions were made on the test data using the trained model.

***2.4.4 Evaluation:*** The trained model's effectiveness is evaluated by measuring its accuracy. To gain further insight into the model's performance, a confusion matrix is generated, which displays the number of true and false positive and negative classifications. Additionally, the precision, recall, and f1-score of the model output are calculated using the classification report. These evaluation metrics provide a comprehensive overview of the model's ability to correctly classify new, unseen data.

**3. RESULTS**:

* 1. Chart, bar chart

     Description automatically generated ***Preliminary analysis results***: Before data modeling, the data was analyzed to check whether there was any imbalances in the class distribution. The analysis showed the there was no class imbalance. The same can be seen in the figure below.

Figure 1: Class distribution in the training and test data

Images belonging to different categories in the dataset were also visualized as shown in figure 2.

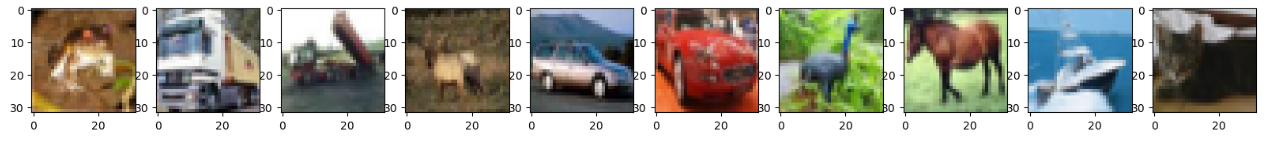


Figure 2 : Images of different classes of the output variable

Chart

Description automatically generated

Figure 3: Change in the Model's loss and Accuracy for each epoch for the DNN model with CNN

* 1. ***DNN with CNN:*** The DNN model with CNN achieved a test accuracy of 0.81, meaning that 81% of the test data was correctly classified. The test loss was 0.605, indicating the difference between predicted and actual labels was relatively low. The change in the model’s loss and accuracy for every epoch can be seen in figure 3.

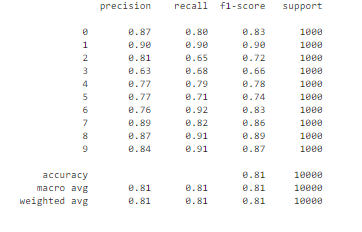
The precision of the model was 0.81, which represents the proportion of true positive classifications in relation to all positive classifications. The recall was also 0.81, which represents the proportion of true positive classifications in relation to all actual positive samples. The f1 score was 0.81, which is the harmonic mean of precision and recall. To summarize these values, a classification report has been printed out as shown in figure 4.

Figure 4: Classification report for the DNN model with CNN

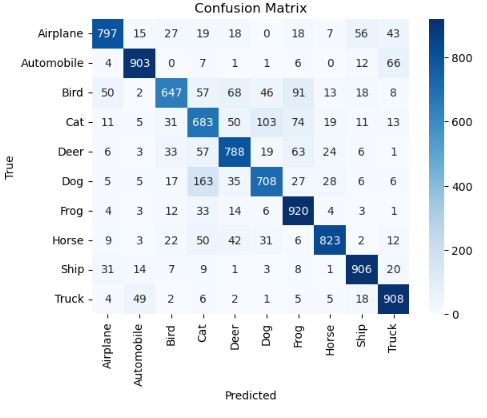
To visualize the number of true and false positive and negative classifications in this classification task, the confusion matrix has been plotted as shown in figure 4.

Figure 5: Confusion matrix for the Deep Neural network model with CNN.

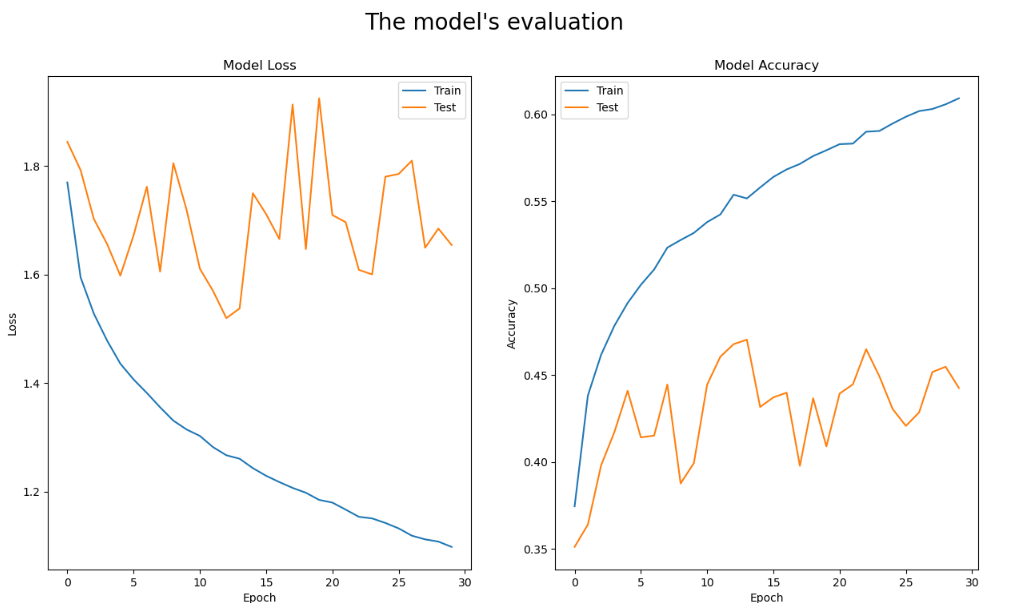
* 1. ***DNN without CNN:*** The DNN model without CNN achieved a test accuracy of 0.4425, meaning that only 44.25% of the test data was correctly classified. The test loss was 1.6543, indicating a relatively high difference between predicted and actual labels. The variation in the model’s loss and accuracy with each epoch can be seen in figure 6.

Figure 6: Change in the model's loss and accuracy per epoch for the DNN model without CNN

Table

Description automatically generatedThe precision of the model was 0.47, which represents the proportion of true positive classifications in relation to all positive classifications. The recall was 0.45, which represents the proportion of true positive classifications in relation to all actual positive samples. The F1-Score was 0.45, which is the harmonic mean of precision and recall. To summarize these values, a classification report has been printed out as shown in figure 7.

Figure 7: Classification report for the DNN model without CNN

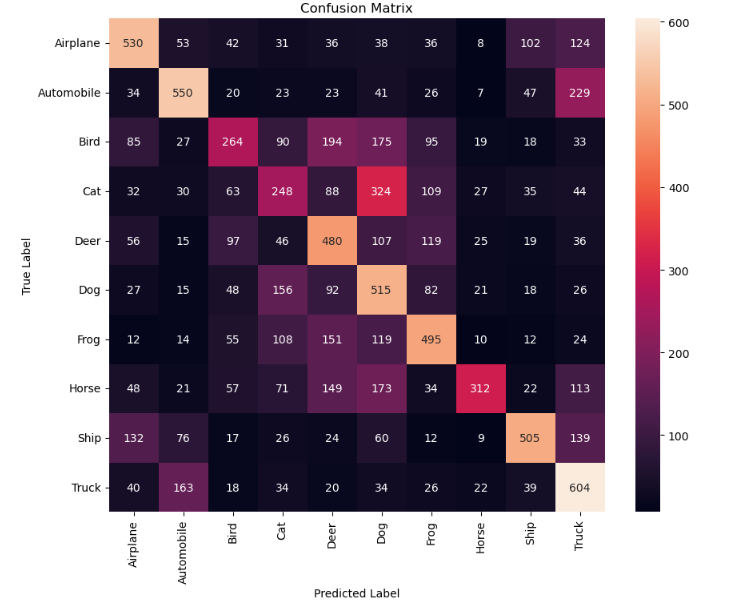
To visualize the number of true and false positive and negative classifications in this classification task, the confusion matrix has been plotted as shown in figure 8.

Figure 8: Confusion matrix for the DNN model without CNN

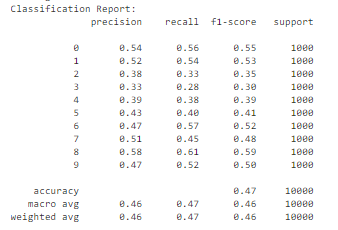
***3.3 Random Forest:*** The random forest model achieved a test accuracy of 0.4654, meaning that only 46.54% of the test data was correctly classified. The precision of the model was 0.46, which represents the proportion of true positive classifications in relation to all positive classifications. The recall was 0.47, which represents the proportion of true positive classifications in relation to all actual positive samples. The f1 score was 0.46, which is the harmonic mean of precision and recall. To summarize these values, a classification report has been printed out as shown in figure 9.

Figure 9: Classification report for the Random Forest classifier model

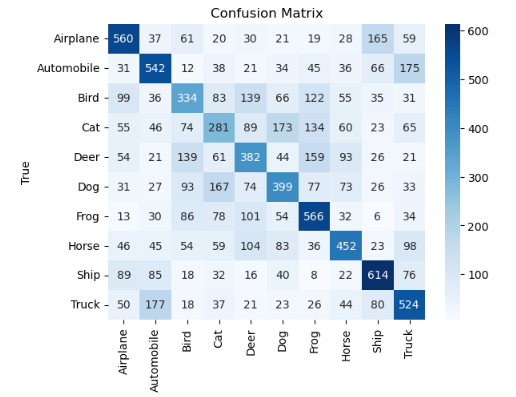
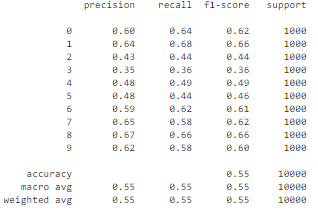
To visualize the number of true and false positive and negative classifications in this classification task, the confusion matrix has been plotted as shown in figure 10.

Figure 10: Confusion matrix for the Random Forest classifier model

***Support Vector Machine:*** The SVM model achieved a test accuracy of 0.55, meaning that only 55% of the test data was correctly classified. The precision of the model was 0.55, which represents the proportion of true positive classifications in relation to all positive classifications. The recall was also 0.55, which represents the proportion of true positive classifications in relation to all actual positive samples. The f1 score was 0.55, which is the harmonic mean of precision and recall. To summarize these values, a classification report has been printed out as shown in figure 11.

Figure 11: Classification report for the SVM Classifier with rbf Kernel



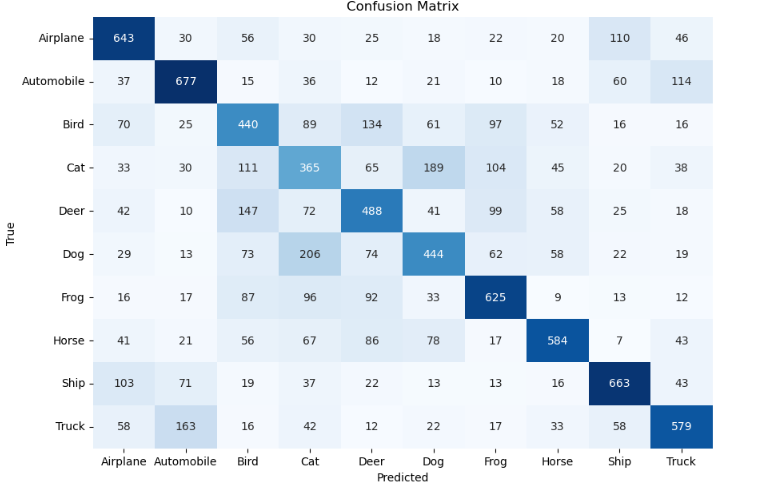
To visualize the number of true and false positive and negative classifications in this classification task, the confusion matrix has been plotted as shown in figure 12.

Figure 12: Confusion matrix for classification with SVM with non-linear rbf Kernel

**4. COMPARISON**

The convergence time of each model is as follows:

* DNN with CNN: 364.07 seconds for 50 epochs
* DNN without CNN: 70.6456 seconds for 30 epochs
* Random Forest: 234.1155 seconds
* SVM: 318.452 seconds

From these metrics, it can be observed that the DNN with CNN takes significantly longer than the other models to converge, as it requires 50 epochs to train and there are multiple convolution layers. The SVM takes longer than the Random Forest, but shorter than the DNN with CNN. The DNN without CNN has the shortest convergence time of all the models.

However, it is important to note that convergence time alone cannot be used to determine the effectiveness of a model. Other factors such as accuracy, precision, recall, and f1 score should also be considered to conclude the performance of a model. The summary of the performance metrics is given in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| DNN With CNN | **81%** | **81%** | **81%** | **80.7%** |
| DNN Without CNN | 44.25% | 45% | 44% | 43% |
| SVM | 55% | 55% | 55% | 55% |
| Random Forest | 46.54% | 46% | 47% | 46% |

In terms of accuracy, the DNN with CNN achieved the highest performance with a test accuracy of 0.81 followed by SVM with an accuracy of 0.55, while DNN without CNN and Random Forest had lower accuracy at 0.4425 and 0.4654, respectively. When looking at precision, recall, and f1 score, the DNN with CNN outperformed the other models with scores of 0.81, 0.81, and 0.807, respectively. SVM performed slightly better than Random Forest in terms of precision, recall, and f1 score with scores of 0.55 for all three metrics, while Random Forest achieved scores of 0.46, 0.47, and 0.46, respectively.

Overall, the DNN with CNN outperformed the other models in terms of accuracy and precision, recall, and f1 score. SVM also showed reasonable performance with an accuracy of 0.55 and similar scores for precision, recall, and f1 score. However, the DNN without CNN and Random Forest had lower performance in all metrics. Therefore, based on these results, the DNN with CNN can be considered the better model for CIFAR-10 dataset image classification.

**5. INFERENCE**

Comparing the performance of all the 4 models, the Deep Neural Network with CNN model achieved an impressive test accuracy of 0.80, which means that it correctly classified 80% of the images in the test set. The test loss of 0.605 indicates that the model's predictions were on average, very close to the true labels. Additionally, the precision, recall, and f1-score for the model were all 0.80, which suggests that the model was effective at identifying both true positives and true negatives, with a balance between precision and recall. Thus, the high-test accuracy achieved by the Deep Neural Network with CNN model, as well as its high precision, recall, and f1-score, demonstrate that it was able to accurately classify images across all classes, making it the best model for CIFAR-10 image classification.

**6. CODE**

The code implementation for the above project can be found in the subsequent pages.