CIFAR10 classification using Machine Learning and Deep Learning Models

***Abstract*—This study explores the classification on the Cifar10 dataset, using machine learning algorithm - Random Forest and deep learning algorithms - CNN, ResNet. Random Forest works well on structured data, CNN extracts intricate features, ResNet uses residual connections enabling more efficient training. A comparative analysis of accuracy and efficiency of each of the models sheds light on the strengths and limitations of these approaches for image recognition. The accuracies achieved were - 44.97%, 81.1% and 83.6% for Random Forest, CNN and ResNet respectively.**

I. Introduction

The classification of images in the CIFAR-10 dataset is a crucial task in computer vision, with applications in areas such as object recognition, autonomous systems, and digital content analysis. This project focuses on classifying the CIFAR-10 dataset, a widely used benchmark for image classification [1], using Machine Learning model (like Random Forest) and Deep Learning models (like Convolutional Neural Networks (CNNs) and ResNet). The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Random Forest is a simple, machine learning model based on decision trees that works well for structured data. We included it to serve as a baseline for comparison, as it is relatively easy to train and does not require extensive hyperparameter tuning. This model is useful for understanding the performance gap between traditional machine learning models and more sophisticated deep learning approaches. CNNs [2] are well-known for their ability to automatically extract features from raw pixel data, utilize convolutional layers to capture hierarchical spatial patterns and local features. They are a natural choice for tasks like image classification and have been proven to perform well on image datasets. ResNet [3] is a deeper, more advanced CNN architecture that addresses the vanishing gradient problem in deep networks by incorporating residual connections, enabling more efficient training of deep models. ResNet was chosen because it has shown state-of-the-art performance on various image classification tasks. This project leverages these three methodologies to classify the CIFAR-10 dataset, with an emphasis on their comparative performance in terms of classification accuracy and computational efficiency. By implementing Random Forest, CNNs and ResNet separately, the project aims to provide a comprehensive analysis of their strengths and weaknesses, offering valuable insights into their suitability for different computer vision tasks. This comparative study has practical implications for industries that rely on automated image recognition, including e-commerce, robotics, and autonomous driving, where accurate and efficient image classification is key to improving both user experience and operational performance.

II. Method

A. Dataset Description

The CIFAR-10 dataset, a widely used benchmark for image classification, consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The 10 classes are completely mutually exclusive and consist of a wide range of images like airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The Cifar10 dataset serves as an essential resource for evaluating and benchmarking machine learning and deep learning models, particularly in the realm of computer vision.

B. Model Selection

Random Forest is one of the most powerful machine learning algorithms, and performs well for classification tasks. By combining the predictions of multiple decision trees, it reduces the risk of overfitting and makes the model robust to outliers, as outliers are unlikely to influence all trees in the forest. This approach ensures more stable and accurate predictions, making it a reliable choice for classification problems.

The hierarchical feature learning of CNNs allows for the extraction of complex patterns, textures, and shapes within the images. This results in a more intricate, robust and nuanced understanding of images, increasing the efficiency and thereby potentially improving the user experience and operational performance.

ResNet (Residual Neural Network) is a deep learning architecture which uses residual connections and mitigates the vanishing gradient problem. These connections allow gradients to flow directly through the network, ensuring deeper networks can learn effectively. The residual blocks in ResNet helps to learn identity mappings and preserve essential information across layers, this ensures that important low-level and high-level features are retained, improving its ability to differentiate between classes in a classification task.

C. Model Implementation

The Random Forest implementation was chosen to leverage its ability to handle high-dimensional data effectively while being robust to overfitting. Using the scikit-learn library, the model was optimized through hyperparameter tuning, including parameters like the maximum depth of trees, the number of features to consider, and the minimum samples per leaf.

To improve feature extraction, PCA was applied to reduce dimensionality and retain essential features, ensuring the model efficiently handles the CIFAR-10 dataset. The training phase involved fitting the model to the preprocessed data, followed by validation to fine-tune its performance. The Random Forest Algorithm provided valuable insights into feature importance and preprocessing techniques for this dataset.

A Convolutional Neural Network (CNN) was selected due to its proven effectiveness in processing high-dimensional image data and its capability to capture complex, non-linear patterns. Utilizing the PyTorch framework, the CNN was fine-tuned to improve its performance on the CIFAR-10 dataset. Key hyperparameters such as the network depth, number and size of filters, activation functions, and max pooling strategies were crucial in optimizing the architecture. Furthermore, hyperparameters like learning rate, batch size, and dropout rates were meticulously adjusted to strike a balance between generalization and overfitting, resulting in an optimized model. The training process involved preprocessing the raw pixel data from the Cifar10 images, transforming them into a format compatible with the CNN (&ResNet) architecture. The model was designed to automatically learn hierarchical features from the input images, enabling it to discern complex patterns and representations within the dataset.

A ResNet-18 architecture was chosen for its ability to address the vanishing gradient problem through the use of residual connections, enabling the effective training of deeper networks. The ResNet model was implemented using the PyTorch framework for the CIFAR-10 dataset. The model's initial convolutional layer was customized with a 3×3 kernel and a variable number of filters to enhance feature extraction. The fully connected (fc) layer was modified to output 10 classes corresponding to the CIFAR-10 categories. Experimented with hyperparameters such as learning rate, optimizer type (Adam, SGD, or Adadelta), and activation functions (ReLU, Sigmoid, or Tanh), to improve model performance. Batch normalization and max pooling were employed to stabilize learning and reduce overfitting. The training process involved preprocessing raw image data into a format compatible with the ResNet(& CNN) architecture, followed by iterative updates using backpropagation and gradient descent. The model was trained over multiple epochs using cross-entropy loss as the objective function. Performance was evaluated on validation and test datasets, achieving generalization through careful tuning of hyperparameters. ResNet's feature extraction capabilities enabled it to effectively capture complex patterns in image data, making it well-suited for CIFAR-10 classification tasks.

III. Experimental Results and Discussion

A. Data Preparation

In data preparation, PyTorch was used to divide the training dataset into 80:20 ratio for training and validation sets. The test dataset given was used for testing.1 To ensure feature uniformity, normalized the pixel values. Transformed the data using auto augment for Cifar10 and created tensors compatible with the models. Also used Principal Component Analysis (PCA) to strategically reduce dimensionality, streamlining the dataset, expediting the training and increasing efficiency of the Random Forest, by capturing essential features and minimizing redundancy.

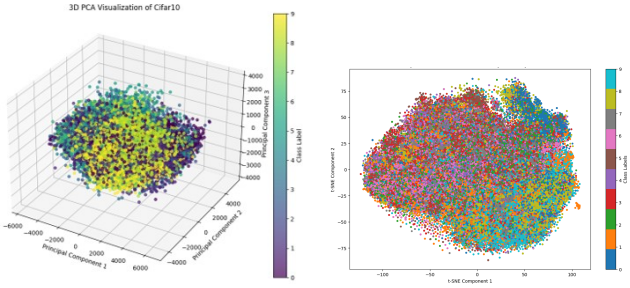
```
x_train: (50000, 3072)
y_train: (50000,)
x_test: (10000, 3072)
y_test: (10000,)
Shape of X_train: (40000, 3072)
Shape of y_train: (40000,)
Shape of X_val: (10000, 3072)
Shape of y_val: (10000,)
Shape of X_test: (10000, 3072)
Shape of y_test: (10000,)
{'frog': 4020, 'truck': 3978, 'deer': 4003, 'automobile': 3986, 'bird': 4048, 'horse': 4023, 'ship': 3997, 'cat': 3984, 'dog': 3975, 'airplane': 3986}
```

Fig. 1: Data Statistics after Train, Valid and Test split

B. Exploratory Data Analysis

We employed multiple data visualization techniques in our research. We visualized one sample image from each class.2 Subsequently, we incorporated Principal Component Analysis (PCA) to project the data into a lower-dimensional space and obtained a 3D visualization with 3 principal components.3a To further enhance the understanding of the dataset's structure, we applied t-Distributed Stochastic Neighbor Embedding (t-SNE) on the PCA-reduced data for 2D visualization.3b By setting the perplexity to 40 and the number of iterations to 2000, we achieved improved separation of data points in the 2D space. The resulting t-SNE plot, as shown, clusters the CIFAR-10 dataset into distinct regions corresponding to class labels, providing a clear representation of class separability and underlying patterns in the data. We also reconstructed the images with 250 principal components and visualized one sample image from each class.4

Fig. 2: Sample Images from each class



(a) PCA 3D Visualization (b) t-SNE 2D visualization

Fig. 3: PCA Visualizations

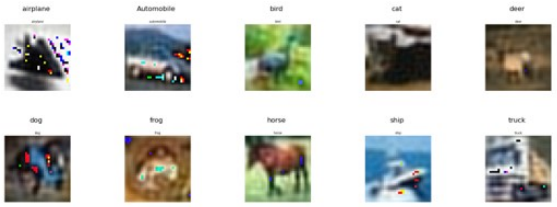


Fig. 4: PCA reconstructed sample images from each class

C. Models

- **Random Forest** The algorithm constructs multiple decision trees during training and outputs the class that is the mode of the classes predicted by the individual trees, effectively preventing overfitting. In the implemented model, hyperparameter tuning was performed using a parameter grid to explore various configurations, including criterion (gini and log_loss), max_depth (values such as None for unlimited depth, 5, 10, 100, etc.), min_samples_leaf (1, 5, 10, and 50), and max_features (sqrt and log2). The criterion determines how splits are evaluated, with gini yielding better results. The max_depth controls the depth of the trees, with None allowing the model to capture complex patterns. The min_samples_leaf ensures that each leaf has enough samples, preventing overfitting, while max_features determines the subset of features to consider for splits, balancing efficiency and performance. After experimentation, the optimal parameters that yielded the highest accuracy were: criterion = gini, max_depth = None, min_samples_leaf = 10, and max_features = sqrt. The model was trained on the training dataset using these parameters and validated on the validation set to test its capability on unseen data. The final trained model achieved the following accuracies: training accuracy = 90.48%, validation accuracy = 45.29%, and testing accuracy = 44.97%. 5 As we can see the actual and predicted classes in 6. CIFAR-10 has 10 classes, so the random guessing accuracy would be 10% if we were to randomly guess between the classes. Therefore, an accuracy of 50% is significantly better than random guessing, but it is still far below what deep learning models can achieve. While it is powerful for structured/tabular data, we can say that Random Forest is not naturally suited for high-dimensional image data like CIFAR-10 as it does not leverage spatial hierarchies or local patterns in images like deep learning models do.

```
Model Loaded
RandomForestClassifier(min_samples_leaf=10, random_state=42)
Test Accuracy: 44.97%
Precision: 0.44
Recall: 0.45
F1 Score: 0.44
```

Fig. 5: Random Forest on Test dataset - Metrics

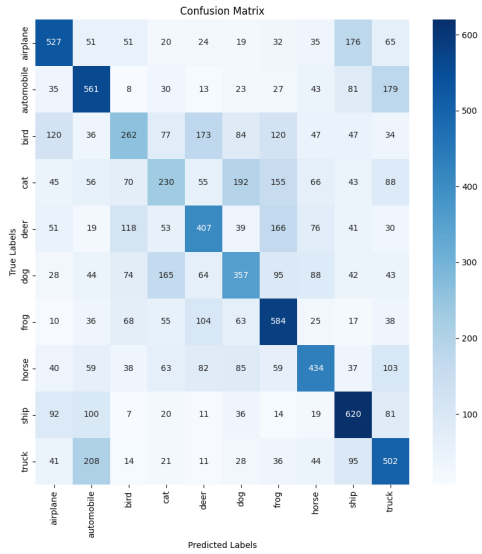


Fig. 6: Random Forest on Test dataset - Confusion Matrix

- **CNN** The network consists of three convolutional layers, each followed by batch normalization, an activation and max pooling operation to reduce the spatial dimensions and capture hierarchical features from the input images. Progressively increased the number of filters in the convolutional layers - second layer having 64 and third layer having 128. Experimented with 32 and 64 filters for the first layer. After the convolutional layers, the output is flattened and passed through two fully connected layers, where the first one maps the flattened features to a 128-dimensional space, and second one outputs the final 10-dimensional classification result, corresponding to the CIFAR-10 classes. Added a a dropout layer (with a 50% dropout rate) for regularization, in between the two fully connected layers. Experimented with ReLU, Sigmoid, and Tanh as the activation functions, for both the hidden layers and the output. 7 In the training process, the model was trained over 20 epochs using the training dataset. During each epoch, the model was set to training mode, and the forward pass was performed for each batch in the training data. The loss was calculated using Cross-Entropy Loss, which is appropriate for multi-class classification tasks. The optimizer used was Adam, which adjusts the learning rate during training to help achieve faster convergence. After computing the gradients via backpropagation, the model's parameters were updated to minimize the loss. At the end of each epoch, the model was validated on a separate validation dataset to assess its performance and avoid overfitting. Experimented with different batch sizes (32, 64, and 128), to evaluate their impact on the model's performance and training dynamics. Additionally, a learning rate scheduler (StepLR) was employed to adjust the learning rate every 5 epochs, gradually reducing

it by a factor of gamma(0.1). This helped refine the model's convergence. The average training loss for each epoch was printed to monitor progress, and the learning rate after each epoch was also displayed to track the adjustments made during training. The model with the best validation accuracy was used to test on the test dataset and it yielded an accuracy of 81.11% 8. As we can see from 9, the CNN had difficulties in distinguishing cats and dogs. A mojority of the misclassifications are dog being classified as cat and cat being classified as dog.

- **ResNet** The residual connections (skip connections) in the ResNet architecture allow the model to learn residual mappings instead of directly learning the underlying function. We used the predefined ResNet-18 model from the torchvision library for the classification problem. The ResNet architecture was trained with different numbers of filters in the first convolution layer, such as 32, 64, and 128. Following the convolution layer, a standard ResNet-18 architecture with residual blocks was used, each containing two 3x3 convolution layers with batch normalization and activation. A max-pooling layer followed the initial convolution, and global average pooling was applied before the fully connected layer, which mapped features to 10 output classes for CIFAR-10 classification 10. The model was experimented with using multiple activation functions such as ReLU, Sigmoid, and Tanh. It was trained using cross-entropy loss, an optimizer with momentum (0.9), and a StepLR scheduler that reduced the learning rate by 0.1 every 5 epochs for improved convergence, over 20 epochs with batch sizes of 32, 64, and 128 from the dataset loader. Various optimizers, including SGD, Adam, and Adadelta, were tested to evaluate their impact on performance, with Adam achieving the best results. Validation was performed on a separate validation set after each epoch to monitor performance and prevent overfitting, while testing on unseen data demonstrated high accuracy and generalization ability. Compared to the CNN model, ResNet showed better performance due to its ability to train deeper architectures effectively. Experiments showed that ReLU activation, the Adam optimizer, and a 128-filter size provided the best results in terms of convergence speed and accuracy(83.62%) 11, while the confusion matrix highlighted consistent improvements in classifying CIFAR-10 classes. As we can see from the confusion matrix 12, compared to CNN the overall misclassification is less.

```
CNN(
  (activation): ReLU()
  (conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=4096, out_features=256, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=256, out_features=10, bias=True)
)
```

Fig. 7: CNN Architecture

```
Loaded model with best accuracy
Hyperparameters: {'filter_size': 3, 'num_filters': 64, 'activation': 'relu'}
Test Accuracy: 81.11%
Precision: 0.81
Recall: 0.81
F1 Score: 0.81
```

Fig. 8: CNN on Test dataset - Metrics

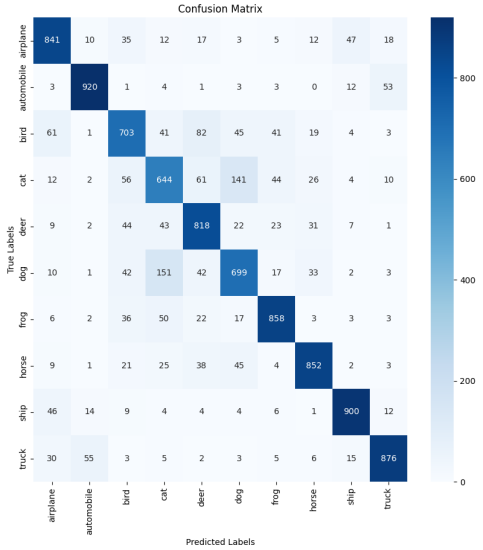


Fig. 9: CNN on Test dataset - Confusion Matrix

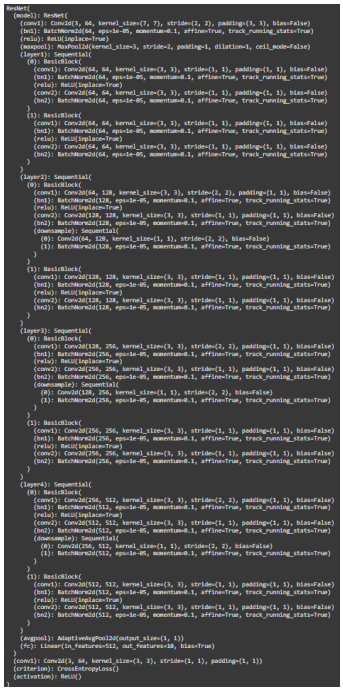


Fig. 10: ResNet Architecture

```
Testing ResNet:
Test Accuracy: 83.62%
Precision: 0.84
Recall: 0.84
F1 Score: 0.84
```

Fig. 11: ResNet on Test dataset - Metrics

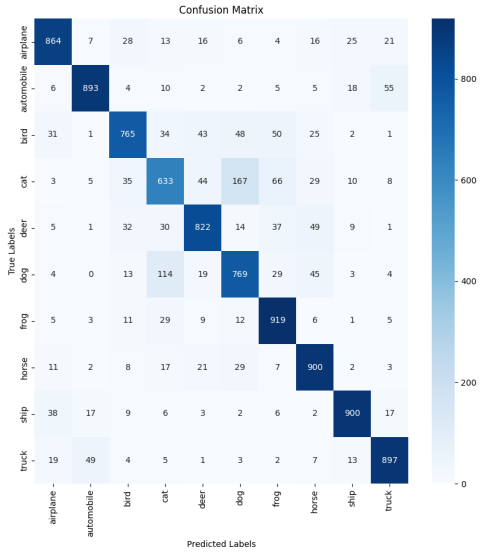


Fig. 12: ResNet on Test dataset - Confusion Matrix

IV. Conclusion

We selected these three models to compare - a baseline (Random Forest) with a traditional deep learning (CNN) and a more advanced deep learning technique (ResNet). The results justify the increasing complexity of the models: Random Forest performs poorly (44.97% accuracy on the Test set) compared to CNN and ResNet, due to its inability to capture spatial features, CNN provides a solid improvement (81.1% accuracy on Test set) by utilizing convolutional layers for feature extraction, and ResNet achieves the best performance (83.6% accuracy on the Test set) by leveraging deeper architectures with residual connections, demonstrating its suitability for complex image classification tasks. For future works a DenseNet [4] could be considered, which improves upon the ResNet by connecting each layer to every other layer in a dense manner, facilitating better feature reuse and improving gradient flow across layers.

References

[1] Cifar10 Dataset - Learning Multiple Layers of Features from Tiny Images, Chapter 3, Alex Krizhevsky, 2009.

[2] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," in IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 12, pp. 6999-7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[3] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016

[4] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017

[5] Louppe, Gilles. "Understanding random forests: From theory to practice." arXiv:1407.7502, 2014.