Image Classification on CIFAR-10: Exploring CNN, ResNet-18 with SVM, and Deeper Networks like ResNet-34

Ankita Aswathanarayana, Hayley Hawkins, Harsha Rapuru, Saher Thekedar, Prachi Dudhe

***Abstract*—This study explores the classification of images of the CIFAR-10 dataset which consists of 50,000 training and 10,000 test images spanning over 10 classes. We use three approaches for the image classification: a baseline Convolutional Neural Network (CNN) model,a hybrid ResNet-18 with a Support Vector Machine (SVM) classifier model, and ResNet-34. Our goal is to assess the classification accuracy, training efficiency, and generalization ability of these models. We obtained 75% accuracy on the CNN model,85% accuracy on ResNet-18 + SVM and 95% accuracy on ResNet-34 on the test dataset. The analysis presented in our study can further help in constructing robust image recognition models.**

I. Introduction

A. Overview of the dataset

Image classification is a popular machine learning and computer vision technique that deals with categorizing images by placing labels on images based on preexisting training data. The applications of image classification are wide, from analyzing medical images to autonomous driving technology.

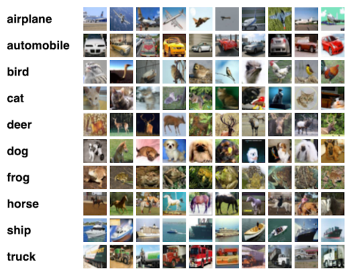


Fig. 1. A sample of CIFAR-10 images

CIFAR-10 is an established multi-class computer-vision dataset used for object recognition. It consists of 60000 32x32 color images in 10 classes, with 6000 images per class. The dataset includes 50,000 training images and 10,000 test images. The images were collected by groups from NYU and MIT, over a span of six months, using search engines such as Google and Flickr. The 10 different classes are Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck. All the classes are completely mutually exclusive. Fig.1 shows sample pictures from the CIFAR-10 dataset.

B. Literature Review

Machine and deep learning landscapes are constantly evolving, especially in recent years. Major steps have been made to develop algorithms with higher accuracy and generalizability, including experimentation of CNNs on the CIFAR-10 dataset.

The paper, Convolutional Neural Network (CNN) for Image Detection and Recognition, is a preexisting example of CNN architectures being implemented on the CIFAR-10 and MNIST datasets. In this study, researchers Chauhan, Ghanshala, and Joshi achieved an accuracy of 80.18% on a CPU. In the process of improving accuracy, the researchers used techniques such as data augmentation to generate variations of images, and dropout layers to prevent the model from memorizing the training data. These techniques enhanced model robustness, reduced overfitting, and improved the model's ability to generalize to new images.

Researchers Doon, Rawat, and Gautam also researched the effectiveness of deep learning models on the CIFAR-10 dataset, focusing on CNNs. They were able to achieve an accuracy of 90% on the training data and 87.57% accuracy on the test set. The model described in their research leverages convolutional layers and employs common data processing techniques such as augmentation and dropout to avoid overfitting. The study highlighted the importance of regularization, optimizers, and tuning of hyperparameters to achieve high accuracy, aspects that we implemented into our own experimentation.

C. Data Preprocessing

The data preparation process is an important step. First, we scaled each image to 32x32 pixels. Various data transformations were used on the training data to avoid the model from finding specific patterns in the features. These modifications included random rotations of up to 20 degrees and horizontal flips with a frequency of 0.1, followed by brightness, contrast, and saturation alterations utilizing color jitter. In addition, sharpness was randomly altered by a factor of two, and 75% of the images were erased at random. The primary goal of these changes was to help the model learn more invariant features. Finally, we transformed the images to tensors and normalized the data based on the mean and standard deviation values obtained from the training data. Similar steps were performed on the test data, including resizing and normalization, but without the transformations.

D. Exploratory Data Analysis

1) Descriptive Statistics: Initially, the class distribution was reviewed, and all classes were evenly distributed across the train and test data sets. In the following phase, we assessed the pixel distribution of colors (Red, Blue, and Green) in the train and test datasets. The study revealed that the pixel values for each color were evenly distributed across the 0–255 range, indicating that no outliers existed. This even distribution assists in discovering patterns and limiting the likelihood of overfitting.

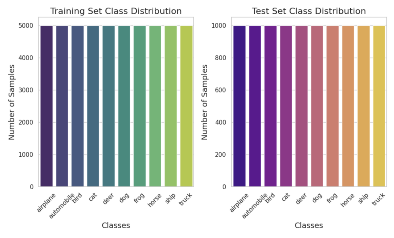


Fig. 2. Distribution of Classes for Train and Test

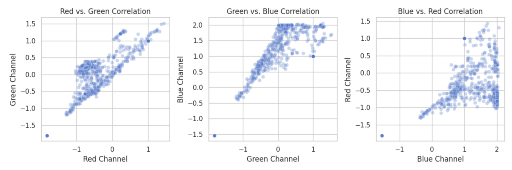


Fig. 3. RGB Channel Correlation

2) Data Visualization: For further analysis, few of the images were randomly checked for all the classes. Additionally, we used dimensionality reduction to see if the reduced components could maintain variance and minimize complexity. We tried to minimize the 3,072 components in the original dataset to just three, and then used a 3D scatter plot to visualize the outcome. For classes with distinctive characteristics, this study revealed some clear clustering behavior. The majority of the classes did, however, significantly overlap, suggesting that feature extraction is required for improved class separation. Furthermore, we chose not to lower the image size because the reduced components only accounted for 47% of the variance.

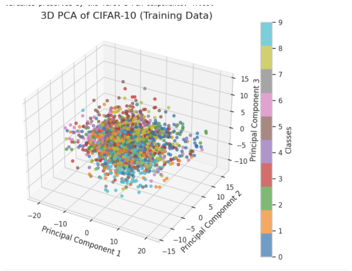


Fig. 4. Principal Component Analysis of CIFAR-10

II. Methodology

A. Convolution Neural Networks

1) Convolution Layer: A convolutional layer is used to extract features from the input data. It applies a kernel (small filter) across the input tensor, calculating an element-wise product and summing the results to form a feature map. Multiple kernels are used to extract different features. Key hyperparameters include kernel size and the number of kernels, which define the depth of the output feature map. Padding is applied to maintain the spatial dimensions, and stride defines the step size of the kernel.

2) Nonlinear Activation Function: The output of the convolution is passed through the ReLU activation function,to introduce nonlinearity, allowing the network to learn complex patterns.

3) Pooling Layer: Max pooling is used to downsample the feature maps, reducing dimensionality and computational load while maintaining essential information. A common configuration is a 2x2 filter with a stride of 2, which halves the dimensions of the feature map.

4) Fully Connected Layer: The feature maps from the final convolution or pooling layer are flattened into a 1D vector and passed through one or more fully connected layers. Each output is connected to every input through learnable weights. The final layer produces the classification probabilities, with the number of output nodes equal to the number of classes.

5) Loss Function: We use cross-entropy loss, a standard loss function for multiclass classification, which measures the difference between predicted and true labels.

6) Gradient Descent: The model is trained using Stochastic Gradient Descent (SGD), which updates the learnable parameters (kernels and weights) iteratively to minimize the loss. The learning rate determines the size of the steps taken during each update. Mini-batch gradient descent is used to optimize the model, where the gradient is computed using a subset of the data at each step.

B. SVM

We used SVM (Support Vector Machine) to train and classify the CIFAR10 images. SVMs are well-suited for high-dimensional spaces as their optimization focuses only on support vectors near decision boundaries. SVM uses a kernel trick to handle non-linear data. For our study we implemented the Radial Bias Function (RBF) kernel to capture complex class boundaries. The RBF kernel projects the complex, non-linearly separable CIFAR-10 classes into a higher-dimensional space, enhancing their separability, thereby making it suitable for multi-class classification. SVM with RBF kernel uses two important hyperparameters: C (regularization), that monitors the trade-off between margin maximization and error minimization and Gamma, which controls the influence of each training point. For tuning these hyperparameters, we used randomized cross validation.

C. Resnet

Convolutional neural networks are difficult to train because of their vanishing gradients. CNN relies on back propagation to determine gradients at the output using the chain rule of derivatives. Gradients are multiplied backwards, and the gradient shrinks too small when it reaches to the starting layers, resulting in a vanishing gradient. Resnet, a pre-trained model offers a solution to this problem: the concept of skip connection has been introduced, which adds the original input to the output of the convolution block. In general, convolution layers are layered, and we have added the actual input to the layer's output, known as the skip connection, which overcomes the problem of vanishing gradients. Here, the values will not be too small as the connections are skipped

TABLE I
RESNET-34 ARCHITECTURE

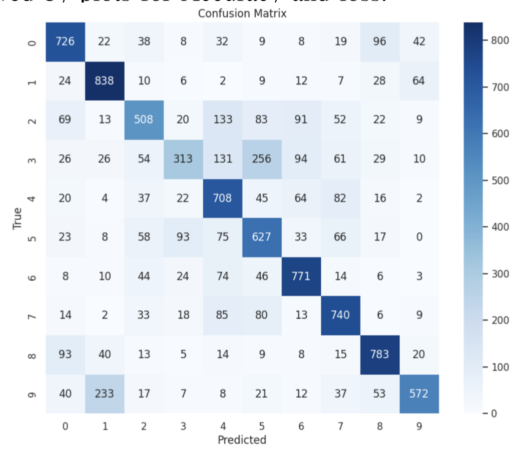
Layer Name	Output Size	Configuration
Conv1	112 × 112	7 × 7, 64, stride 2, Max Pool
Conv2_x	56 × 56	3 blocks, 2 × (3 × 3, 64)
Conv3_x	28 × 28	4 blocks, 2 × (3 × 3, 128)
Conv4_x	14 × 14	6 blocks, 2 × (3 × 3, 256)
Conv5_x	7 × 7	3 blocks, 2 × (3 × 3, 512)
Average Pool	1 × 1	7 × 7 avg pooling
Fully Connected	1 × 1	1000 (output classes)

III. EXPERIMENTAL RESULTS

A. Experiment 1: Convolutional Neural Networks

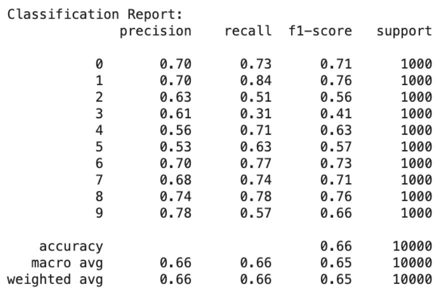
The main goal of this experiment is to classify the images by identifying the features of all the images with CNN. Convolutional Neural Networks (CNNs) are preferred for high-dimensional data because they efficiently extract features from raw images via convolutional layers. Pooling layers minimize image dimensions while preserving critical features, which improves accuracy. Convolutional and pooling layers extract features effectively independent of image resolution. While the computational cost increases dramatically with more dimensions, CNNs exploit GPUs for faster training, making them well-suited for these workloads.

Initially, we started with a simple CNN consisting of 2 Convolution layers, 1 pooling layer and 1 fully connected layer. This model was trained with 20 epochs. We observed a Validation Accuracy of 59.60 and Train Accuracy of 63.22. To improve this model further, we added 2 hidden layers in the fully connected layer. However, the validation and train accuracy dropped to 57.15% and 58.34% respectively. Below are the plots for confusion matrix, performance metrics followed by plots for Accuracy and loss.



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	726	22	38	8	32	9	8	19	96	42
1	24	838	10	6	2	9	12	7	28	64
2	69	13	508	20	133	83	91	52	22	9
3	26	26	54	313	131	256	94	61	29	10
4	20	4	37	22	708	45	64	82	16	2
5	23	8	58	93	75	627	33	66	17	0
6	8	10	44	24	74	46	771	14	6	3
7	14	2	33	18	85	80	13	740	6	9
8	93	40	13	5	14	9	8	15	783	20
9	40	233	17	7	8	21	12	37	53	572

Fig. 5. CNN: Confusion Matrix



```
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.73      0.71      1000
           1       0.70      0.84      0.76      1000
           2       0.63      0.51      0.56      1000
           3       0.61      0.31      0.41      1000
           4       0.56      0.71      0.63      1000
           5       0.53      0.63      0.57      1000
           6       0.70      0.77      0.73      1000
           7       0.68      0.74      0.71      1000
           8       0.74      0.78      0.76      1000
           9       0.78      0.57      0.66      1000

    accuracy                           0.66     10000
   macro avg       0.66      0.66      0.65     10000
weighted avg       0.66      0.66      0.65     10000
```

Fig. 6. CNN:Performance Metrics

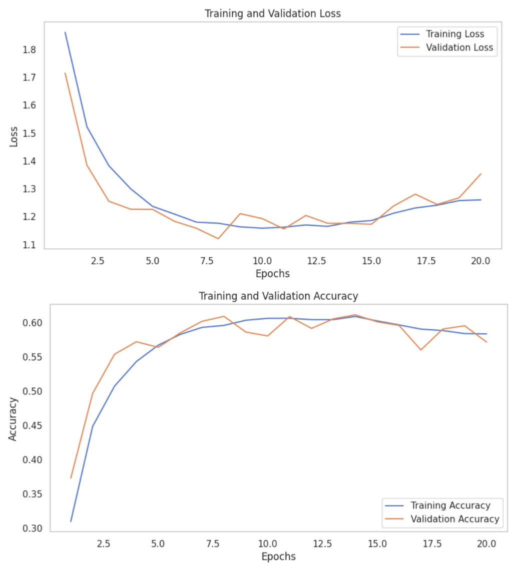


Fig. 7. CNN: Epochs vs Accuracy

The model's results showed that it could learn from the train data, although its performance was relatively good for a simple CNN. When we added hidden layers, training accuracy exceeded validation accuracy, indicating overfitting. However, there is still space for improvement in the accuracy of generalizing validation data. If we look at the confusion matrix, we can see that the model misidentified an automobile as an airplane, indicating that the model struggled to distinguish between two classes.The additional convolutional layer did not improve the learning process, and the increased model complexity appeared to make it more difficult for the network to generalize well.

B. Experiment 2: ResNet-18 + SVM Hybrid model

In our second experiment, we trained the CIFAR-10 dataset on a hybrid model by leveraging a pretrained ResNet18 model for feature extraction and SVM with an RBF kernel for classification. ResNet-18 excels at extracting high-quality, abstract features from raw data, (i,e, CIFAR-10 images), but can be computationally expensive and less interpretable. On the other hand, SVMs are well-suited for high-dimensional spaces, non-linear data and multi class classification, but they struggle with handling unstructured raw data (like images), leading to increased computational costs. Our aim was to increase the efficiency and performance of the model by leveraging the strengths of both ResNet and SVM. Additionally, we used randomized cross validation (5 iterations) on 20% of the training data for tuning the hyperparameters- Gamma and C (Regularization). The range for C was set to values between 0.1 and 10 and choices for gamma were 'scale', 0.1 and 1. The best values for Gamma and C obtained were: 'scale' and 7.566580531858055 respectively. Using these values, we trained the SVM model on the entire training set and achieved 95.13% accuracy on the validation set and 86.41% accuracy on test data. Below is a plot for confusion matrix and a tabulation of the different evaluation metrics on the training, validation and test sets.

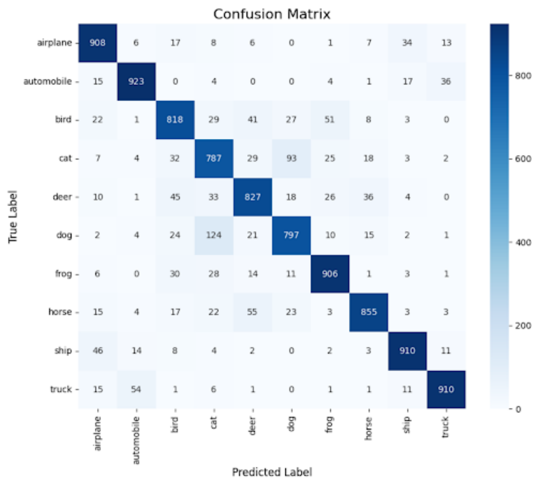


Fig. 8. CNN:Performance Metrics

TABLE II
PERFORMANCE METRICS

Data	Accuracy	Precision	Recall	F1 Score
Train	95.01%	95.02%	95.01%	95.01%
Validation	95.13%	95.13%	95.13%	95.13%
Test	86.41%	86.47%	86.41%	86.42%

C. *Experiment 3: Deeper Networks - ResNet-34*

The main goal of this experiment was to further improve the accuracy for the image classification, better than CNN and SVM. In order to achieve this, we used Resnet-34. Convolutional Neural Networks (CNNs) achieved an accuracy of 65%. However, adding more layers led to overfitting, primarily caused by the vanishing gradient problem. CrossEntropyLoss criteria was used for the classification job, which is common in multi-class situations. The optimizer utilized was Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and a momentum factor of 0.5, which allowed for quicker convergence by accumulating previous gradients.The model was trained across ten epochs, with performance measured on both the training and validation datasets. This approach involved tracking training/validation accuracy and loss in order to analyze learning behavior. With this, we obtained around 85% accuracy. Furthermore, we applied hyper parameter tuning by considering multiple values of learning rate(0.1,0.01,0.001) and momentum(0.2,0.5,0.9). We eventually obtained 95% accuracy for a learning rate of 0.01 and momentum of 0.2. Below are the plots for confusion matrix, performance metrics followed by plots for Accuracy and loss.

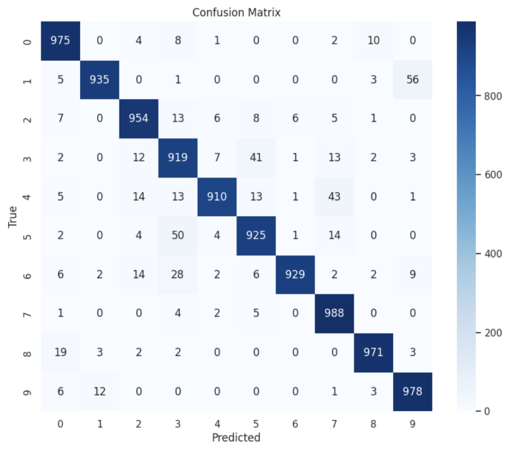
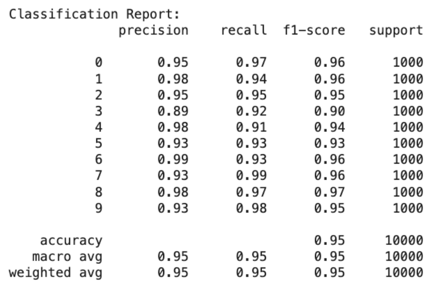


Fig. 9. Resnet: Confusion Matrix



```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      1000
           1       0.98      0.94      0.96      1000
           2       0.95      0.95      0.95      1000
           3       0.89      0.92      0.90      1000
           4       0.98      0.91      0.94      1000
           5       0.93      0.93      0.93      1000
           6       0.99      0.93      0.96      1000
           7       0.93      0.99      0.96      1000
           8       0.98      0.97      0.97      1000
           9       0.93      0.98      0.95      1000

    accuracy                           0.95     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.95      0.95      0.95     10000
```

Fig. 10. Resnet:Performance Metrics

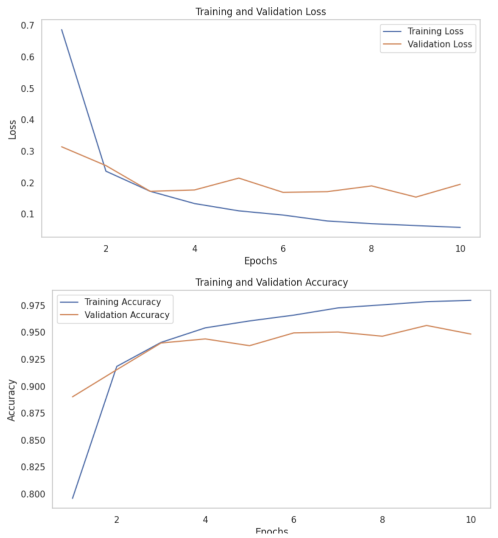


Fig. 11. Resnet: Epochs vs Accuracy

IV. Model Interpretation

A saliency map shows which pixels in the image contribute the most to the model's prediction, emphasizing regions that the model thinks important for categorization. In this scenario, we backpropagated the gradients from the output class (deer) to the input picture, capturing pixel-level sensitivity. The heatmap revealed that the model largely focused on the deer's form, notably the legs and body contour, implying that it relied on these important elements to correctly identify the class. This highlights our model's capacity to separate key characteristics from irrelevant background data, offering insight into the learned representations. This interpretability assures that the model is not generating predictions based on noise or misleading correlations in the information, but rather on significant visual patterns, giving credibility to it's corresponding predictions.

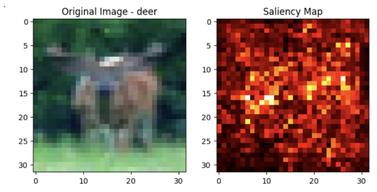


Fig. 12. Sailency Map

V. Business Insights

The methodologies explored in this study could be applied to a number of different industries. For example, in the context of healthcare, these models could be adapted for tasks such as medical imaging analysis, detecting diseases from X-rays or MRIs, or improving diagnostic accuracy. In autonomous systems, for example self-driving cars, the models explored in this study could be used to enhance object classification in real-time driving decision making. The insights from this research could help companies optimize machine learning pipelines, reduce costs pertaining to overfitting or retraining, and ultimately deliver more reliable solutions to customers.

More of this type of research will be crucial as the world becomes even more reliant on artificial intelligence. Ensuring that we have transparent and trustworthy models is an essential requirement, especially in fields like healthcare. This study provides a foundation for developing robust image classification systems and lays the groundwork for future research in a wide range of industries.

VI. Conclusion

This study examines the effectiveness of a few common deep learning approaches for image classification using the CIFAR-10 dataset. We compared CNN, ResNet-19 with SVM, and ResNet-34, demonstrating how each of these techniques produce significant differences in performance. The baseline CNN achieved a 63% accuracy. The ResNet-18 model that utilized SVM improved the accuracy to 86%, showcasing how deep feature extraction and robust classifiers can improve model performance. The ResNet-34 was able to achieve the highest accuracy, 95%, which demonstrates the benefits of deep architectures. ResNet-34 was most successful due to its ability to mitigate issues like vanishing gradients. Our findings highlight how balancing network depth and design can help improve generalization and prevent overfitting.

References

[1] R. Chauhan, K. K. Ghanshala and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCC), Jalandhar, India, 2018, pp. 278-282, doi: 10.1109/ICSCCC.2018.8703316.

[2] R. Doon, T. K. Rawat, and S. Gautam, "CIFAR-10 classification using deep convolutional neural network," in 2018 IEEE PuneCon, 2018, pp. 1-5.

[3] Q. A. Al-Haija, M. A. Smadi, and S. Zein-Sabatto, "Multi-Class Weather Classification Using ResNet-18 CNN for Autonomous IoT and CPS Applications," 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2020, pp. 1586-1591, doi: 10.1109/CSCI51800.2020.00293.

[4] Y. Chen, C. Lin. (2006). Combining SVMs with Various Feature Selection Strategies. In: Guyon, I., Nikravesh, M., Gunn, S., Zadeh, L.A. (eds) Feature Extraction. Studies in Fuzziness and Soft Computing, vol 207. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-35488-$8_1$3

[5] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Apr. 2009. https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf

[6] https://ieeexplore.ieee.org/document/8745428

[7] https://www.kaggle.com/c/cifar-10/overview

VII. Code and Project Links

The code for the CIFAR-10 project is available on Google Colab and can be accessed using the following link:

- Link to Google Colab Code

Additionally, the project's website provides further information and is accessible at:

- Link to Project Website