Optimizing Neural Networks for CIFAR-10 Image Classification

Roman Bukreev
Student, Computer Science
Salem State University
Salem, MA, USA
romanbukreev@icloud.com

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert.**

I. Introduction

This project focuses on the classification task performed on the CIFAR-10 dataset, using neural networks. The project will focus on training three types of neural network models: Deep Neural Network (DNN), Deep and Wide Neural Network, and Convolutional Neural Network (CNN). They will be implemented using Keras' Sequential API. The primary goal of the project is to develop an effective Neural Network model for image classification and to understand the trade-offs and performance differences between the architectures used. We will also experiment with optimization and regularization techniques to improve accuracy and avoid overfitting for each model.

II. Dataset Overview

The dataset used for this project is the CIFAR-10 dataset. It consists of 60000 32×32 color images, each corresponding to one of the ten classes. The classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

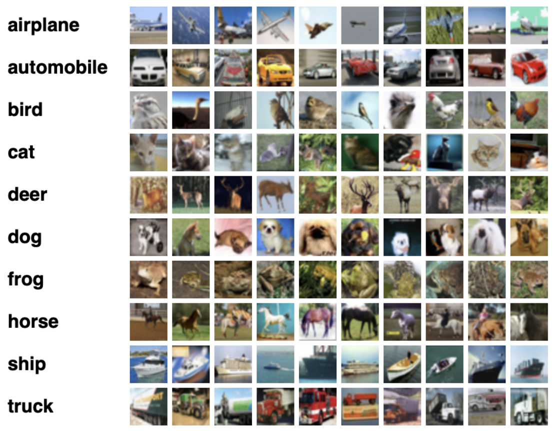

Fig. 1. 10 random images from each class.

The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. ”Automobile” includes sedans, SUVs, things of that sort. ”Truck” includes only big trucks. Neither includes pickup trucks.

The dataset is divided into training and test sets. The training set contains 50000 images, with 5000 images from each class. The test set consists of 1000 randomly selected images from each class. The CIFAR-10 dataset can be loaded directly from the Keras library. It is already divided into training and test sets, as well as separated into features and labels.

III. Problem Statement

As mentioned before, our main goal is to compare the performance of different neural network models. By performance, we mean the classification accuracy. Since we are performing a classification task and the CIFAR-10 dataset is balanced, accuracy is the most suitable metric for the result evaluation. Therefore, our goal is to achieve higher accuracy.

It is also important to mention that this project is focused solely on classification with supervised learning, using the labels that we already have. We will not try to complete more complex tasks such as object detection, for example.

Another challenge that the project may present is the images we have in our dataset. They are very low-resolution and, unlike some other low-resolution datasets (e.g., the MNIST dataset), where it is relatively easy to tell the classes apart visually, CIFAR-10 images can sometimes be very similar, even if they belong to different classes.

IV. Methodology

We can divide our project into two big sections: data prerocessing and models training. Both of these steps are very important in order to achieve the best result. So, before exploring how the models will work and perform on our dataset, let's dive into data preprocessing.

A. Data Preprocessing

The CIFAR-10 dataset is well-known and relatively clean, meaning it does not require extensive preprocessing. However, it is still important to ensure the dataset is optimized for our models in the best possible way.

The first important preprocessing step is data scaling. In our case, the images already have pixel intensity values ranging from 0 to 255. However, this range is quite large and can affect both model performance and training time. To address this, we can normalize the data by simply dividing each value by 255. This brings all features to a common scale, which often leads to more efficient training.

After scaling, there is not much else to do with the features. However, we still need to consider the labels. The labels in the CIFAR-10 dataset are numerical values from 0 to 9. While we could leave them in this form without significantly impacting performance, such a numerical encoding implicitly imposes an order on the classes. To avoid this, we can apply one-hot encoding. This will transform each label into a 10-dimensional vector, where exactly one element (corresponding to the correct class) is set to 1, and all other elements are set to 0. This ensures that no inherent priority or ordering is assigned to any particular class.

B. *Deep Neural Network*

After preprocessing step is done, we can move on to training the models. The first model we will look at is Deep Neural Network. Deep Neural Network is an artificial neural network that contains a deep stack of hidden layers. In our case, the number of hidden layers is already suggested by the project specifications. So we will use 20 hidden layers, each containing 100 neurons. Not having to experiment with the number of layers and neurons will significantlly speed up the training process.

The deep structure of the network allows it to learn increasingly complex features. In early layers, networks often learn simple features like edges and color gradients, while deeper layers combine these basic patterns into more sophisticated representations, helping the network differentiate between classes in a more robust way.

C. *Deep and Wide Neural Network*

Our second model is a Deep and Wide Neural Network, which is an example of a non-sequential neural network architecture. The idea behind this approach is that it connects all or some of the inputs directly to the output layer, allowing the model to learn both simple and complex patterns simultaneously.

In our implementation, we set up two separate branches. The wide branch consists of two dense layers, each with 200 neurons. Its primary goal is to capture simpler, more direct relationships between input features and output classes. This branch effectively "memorizes" dominant patterns in the data.

The deep branch has 20 dense layers, each with 50 neurons. This additional depth enables the network to learn more hierarchical and abstract representations of the data, uncovering more complex patterns.

Finally, these two branches are combined by a single dense layer with 50 neurons, which then leads to the output layer. This structure allows the model to integrate both the broad, direct patterns from the wide branch and the intricate, layered representations from the deep branch, ultimately aiming to improve overall performance.

D. *Convolutional Neural Network*

Our final model is a Convolutional Neural Network. Convolutional Neural Networks are a specialized class of deep neural networks particularly effective at handling image data. The key idea behind them is the use of convolutional layers, which apply multiple filters across the spatial dimensions of the input. These filters can automatically detect low-level features at earlier layers and increasingly complex patterns in deeper layers. By learning these features directly from the data, Convolutional Neural Networks eliminate the need for handcrafted feature extraction and can adapt to various image types.

This model consists of three sequential convolutional blocks. Each block follows a similar pattern: it begins with a convolutional layer that applies filters of a given kernel size (2×2 matrix in our case) to the input, extracting spatial patterns from the images. This is followed by a pooling layer that reduces the spatial dimensions by taking the maximum value from each 2×2 region, helping the network focus on the most essential features while lowering computational complexity.

After these convolutional blocks, we use a flatten layer to convert the resulting 3D feature maps into a 1D vector, making them suitable for the subsequent dense layers. The dense layer then processes these extracted features, combining them into more interpretable representations for the final classification.

E. *Learning Rate*

Now that we have described all the models, we can delve deeper into the details for each of them. We will begin by exploring the learning rate, which is one of the most critical hyperparameters in neural network training. It is essential to find an optimal learning rate for the model to converge effectively. A learning rate that is too high may cause the model's training to fluctuate and fail to converge, while a rate that is too low can slow down the training process significantly, increasing computational costs and delaying meaningful improvements in accuracy.

In our project, we examined the impact of different learning rates on the Deep Neural Network model. We conducted experiments with rates ranging from 0.1 down to 0.000001. Among these, a rate of 0.0001 yielded the best results. Consequently, we used the same value for both the Deep and Wide Neural Network and the Convolutional Neural Network models as well.

F. *Optimizer*

When it comes to optimizer, we are using Nadam optimizer as it is one of the specification for the project. Nadam is a variant of the Adam optimizer that incorporates Nesterov momentum. This combination helps the optimizer navigate the loss surface more effectively by looking ahead and adjusting the parameter updates accordingly. Compared to standard Adam, Nadam tends to converge more quickly and consistently. It provides smoother and more predictable learning curves, enabling the model to reach higher accuracy levels in fewer epochs.

G. Activation Function

The activation function is another important hyperparameter. In our project we are following the specifications giving to us and using the swish activation function for every dense layer that we have in our models. The swish function is defined in following way.

$$Swish(z) = z\sigma(\beta z)$$

Swish combines a smooth, unbounded, and non-monotonic form to create a more flexible mapping than many traditional activation functions. Unlike ReLU, which returns zero for all negative inputs, swish transitions smoothly from negative to positive values and scales outputs by their input magnitudes, allowing it to maintain information about the sign and relative magnitude of the input.

Another big advantage of swish is that it is continuous and differentiable across all real values. Its gradient does not abruptly change as it does with ReLU at zero, potentially leading to more stable and effective training by reducing issues associated with non-differentiable points or flat gradients.

H. Early Stopping

Early stopping is a regularization technique used to prevent overfitting and improve the generalization ability of neural networks. Instead of training a model for a fixed number of epochs, early stopping involves monitoring a validation accuracy during training and halting the process as soon as that metric no longer improves.

Its main advantage is that by stopping before the model has a chance to memorize noise or irrelevant details in the training data, early stopping helps maintain better generalization. Early stopping can also reduce computational costs by halting training as soon as further improvements are unlikely, saving time and resources.

I. Batch Normalization

Batch normalization is a technique designed to improve the stability and speed of neural network training. It normalizes the inputs of each layer by adjusting and scaling the activations to have a more favorable distribution.

For each mini-batch during training, batch normalization computes the mean and variance of the layer's inputs. These statistics are then used to normalize the activations, ensuring they have zero mean and unit variance. The network also learns additional parameters (scale and shift) that allow it to recover or adjust the normalized values, ensuring flexibility and preserving the expressiveness of the model.

Batch normalization can act as a form of regularization, slightly reducing the need for other regularization techniques. It helps the network generalize better to unseen data.

In our project we add Batch Normalization to all the three models, and in each case it leads to the improvement of performance.

J. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

K. Dropout

Dropout is a simple yet effective regularization technique used to reduce overfitting and improve the generalization of neural networks. The key idea is to randomly ”drop out” or deactivate a certain fraction of neurons during the training phase. By doing this, the network is encouraged to learn more robust, distributed representations of the data rather than relying on a few specialized neurons.

Same as with Batch Normalization, we applied Dropout for every model we have, and the accuracy score improved in each case.

V. Results

Remember that our main goal is to achieve the best classification accuracy and compare the models. Let's compare the performance of each model with all the regularization and optimization techniques that we tried.

A. Validation Accuracy

TABLE I
Validation Accuracy for Each Model

	DNN	**DWNN**	**CNN**
-	0.4983		
BN	0.4691	0.4800	
BN+Dropout	0.4602	0.4908	0.7616

First thing to take a look at is that the accuracy for Deep Neural Network seems to get worse with addition of Batch Normalization and Dropout, which contradicts what we mentioned before. But if we could look at the training accuracy, we would see that the base model is overfitting a lot, when the one with Batch Normalization and Dropout does not have this issue.

Next, we can notice that Deep and Wide Neural Network improved the performance on validation set, but not too much compared to Deep Neural Network. One important thing to mention here is that it was overfitting a lot, and the dropout rate was increased from 0.2 to 0.3 in order to avoid it. But we will come back to this issue later.

And finally, Convolutional Neural Network has the best validation accuracy among all of the models. Notice also that we didn't implement Convolutional Neural Network without Batch Normalization and Dropout. The main reason for it is that it takes a lot of time to finish training, and since we've seen on previous models that Batch Normalization and Dropout only improved the results, it made sense to add them right the way.

B. Testing Accuracy

But the most important metric that should give us the best understanding of model performance is testing accuracy, which measures the model accuracy on the test set. So let's compare the models again.

Deep Neural Network got testing accuracy of 0.4649, which is about same as was validation accuracy. Deep and Wide Neural Network showed accuracy of 0.3912 on the test set, that is much lower than validation accuracy. So it looks like the issue with overfitting is not completely resolved for this model. And in the end, Convolutional Neural Network confirmed its best result by getting testing accuracy of 0.7560, which clearly indicates that it is the best model from all that we had.

VI. Conclusion

So after performing the classification task on CIFAR-10 dataset, using three types of neural network models, we've got better understandig of how they work. But even though we've got a pretty good result with Convolutional Neural Network, it still feels like there is more potential in it. So in future, it is possible to experiment with hyperparameters a bit more to try to get better accuracy. Another improvement to consider is performing data augmentation that can increase number of training instances, which is crucial for Convolutional Neural Network.

References

[1] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd ed., O'Reilly Media, 2022.

[2] Keras, "The Sequential class," [Online]. Available: https://keras.io/api/models/sequential/

[3] "The CIFAR-10 dataset," [Online]. Available: https://www.cs.toronto.edu/ kriz/cifar.html

[4] Keras, "CIFAR10 small images classification dataset," [Online]. Available: https://keras.io/api/datasets/cifar10/

[5] Salem State University, "CSC 455 – Machine Learning, Fall 2024 Lecture Slides."

[6] "ChatGPT," [Online]. Available: https://chatgpt.com

[7] "Google Gemini," [Online]. Available: https://gemini.google.com