Final Report

Joshua Gordon & Arieh Norton

ABSTRACT

This report presents a comprehensive analysis of the implementation and performance evaluation of various deep learning models applied to the CIFAR-10 dataset, a widely used benchmark dataset for image classification tasks. The objective is to compare the effectiveness of different model architectures in accurately classifying images across ten distinct categories: (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). The study progresses from simpler models such as a basic Softmax model to more sophisticated Convolutional Neural Networks (CNNs) and Deeper Convolutional Neural Networks (CNNs) . Through detailed experimentation and analysis, the project aims to understand how model complexity impacts classification accuracy and to provide insights into best practices for tackling image classification tasks.

1. INTRODUCTION

The CIFAR-10 dataset consists of 60,000 32x32 RGB images distributed evenly across ten classes, with each class containing 6,000 images. Given this balanced distribution of classes, it is trivial that any effective classification model must outperform a naive baseline approach. In this project, we address the multiclass classification problem posed by CIFAR-10 by developing and evaluating various models. The models are trained on a subset of the dataset and evaluated on the remaining samples. The primary goal is to achieve high accuracy in classifying images into their respective classes, surpassing the performance of a dummy classifier that predicts the most common class with an accuracy of 10%. Additionally, we aim to improve upon simpler models that utilize only linear activation functions, such as logistic regression, by exploring more complex architectures and non-linear activation functions such as deep convolutional neural networks.

2. RELATED WORK AND REQUIRED BACKGROUND

Previous research in image classification has witnessed the successful application of deep learning techniques, particularly CNNs. CNNs have demonstrated superior performance in image classification tasks owing to their ability to automatically learn hierarchical features from raw pixel data. The non-linear activation functions employed in CNNs, such as ReLU, enable them to capture complex patterns and relationships within images more effectively compared to simpler models like logistic regression and softmax. Understanding key concepts such as neural network architectures, activation functions, loss functions, and optimization algorithms is crucial for effective implementation of deep learning models.

3. PROJECT DESCRIPTION

The final model in our project is a deep 22 layer Convolutional Neural Network (CNN) designed for image classification tasks, particularly on the CIFAR-10 dataset. The architecture of the CNN consists of multiple layers including convolutional layers, batch normalization layers, max-pooling layers, dropout layers, fully connected layers, and activation functions. Furthermore the model's loss function is defined as the CrossEntropyLoss function and is optimized with a learning rate of 0.001 in training.

At first the model passes a sanity check before any training is done by performing a forward pass with random inputs, we receive as expected a loss of around ~ Log(10) indicating that the model indeed roughly predicts as expected with a dataset of 10 classes that are evenly distributed.

for context a detailed breakdown of the CNN architecture is shown below:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 32, 32]	896
BatchNorm2d-2	[-1, 32, 32, 32]	64
Conv2d-3	[-1, 32, 32, 32]	9,248
BatchNorm2d-4	[-1, 32, 32, 32]	64
MaxPool2d-5	[-1, 32, 16, 16]	0
Dropout2d-6	[-1, 32, 16, 16]	0
Conv2d-7	[-1, 64, 16, 16]	18,496
BatchNorm2d-8	[-1, 64, 16, 16]	128
Conv2d-9	[-1, 64, 16, 16]	36,928
BatchNorm2d-10	[-1, 64, 16, 16]	128
MaxPool2d-11	[-1, 64, 8, 8]	0
Dropout2d-12	[-1, 64, 8, 8]	0
Conv2d-13	[-1, 128, 8, 8]	73,856
BatchNorm2d-14	[-1, 128, 8, 8]	256
Conv2d-15	[-1, 128, 8, 8]	147,584
BatchNorm2d-16	[-1, 128, 8, 8]	256
MaxPool2d-17	[-1, 128, 4, 4]	0
Dropout2d-18	[-1, 128, 4, 4]	0
Flatten-19	[-1, 2048]	0
Linear-20	[-1, 128]	262,272
Dropout-21	[-1, 128]	0
Linear-22	[-1, 10]	1,290

Input Layer: Accepts input images with dimensions 32x32x3 (RGB images).

Convolutional Layers: The CNN starts with two convolutional layers each followed by a ReLU activation function. These layers extract features from the input images. The first convolutional layer has

32 output channels and a kernel size of 3x3, while the second convolutional layer maintains 32 output channels with a similar kernel size. Both convolutional layers use padding to preserve spatial dimensions.

Batch Normalization Layers: Batch normalization is applied after each convolutional layer. implementing this helps stabilize and accelerate the training process by normalizing the input to the subsequent layer.

Max Pooling Layers: Following each pair of convolutional layers, max-pooling layers are utilized to reduce the spatial dimensions of the feature maps while retaining the most important information. These layers have a kernel size of 2x2 and a stride of 2.

Dropout Layers: Dropout regularization is applied after each max-pooling layer to prevent overfitting by randomly setting 25% of input units to zero during training.

Flatten Layer: After the second dropout layer, the feature maps are flattened into a vector using the flatten layer to prepare for the fully connected layers.

Fully Connected Layers: The flattened features are passed through two fully connected layers. The first fully connected layer has 128 units and utilizes a ReLU activation function. Dropout regularization is applied after this layer . The output layer consists of 10 units corresponding to the 10 classes in the CIFAR-10 dataset.

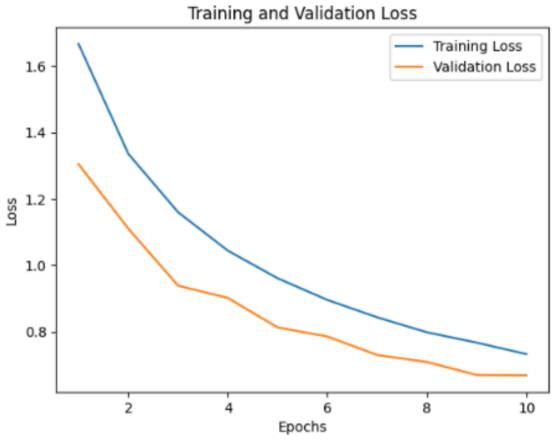
Activation Function: The output of the final fully connected layer is passed through a softmax activation function during inference to obtain class probabilities.

Loss Function and Optimizer: The model is trained using the cross-entropy loss function and optimized using stochastic gradient descent with a learning rate of 0.001 and momentum of 0.9.

Model training

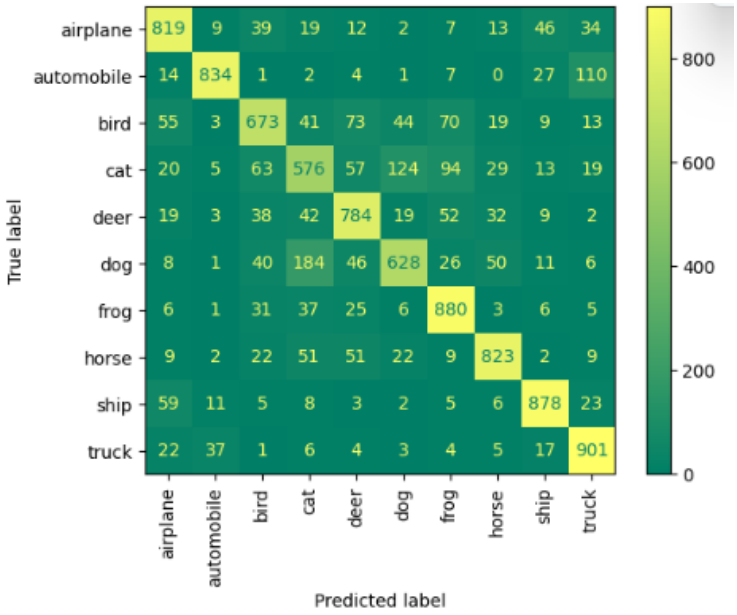
When training the model some information in regards to the train loss and validation loss is shown to reassure the model is not overfitting while training the training data. The graph shows that indeed the model is not overfitting. one can see that the training loss and validation loss decrease together throughout the training. In the case where one notices that indeed the model is overfitting, early stopping could be applied to fix this problem or by adding more regularization.

For context a detailed breakdown of the training of the model on the training data is shown below: one can see that in each epoch of the training the training loss and validation loss decrease accordingly and the model is not overfitting. This is a result of dropout of 25% as this helps the model learn the data in a more robust manner where the neurons are not merely memory machines but rather more generalizes the high hierarchy over look of the data.



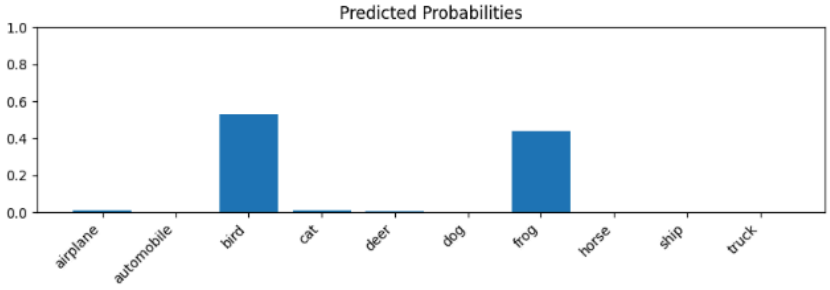

```
Epoch: 1, Validation Loss: 1.305
Epoch: 2, Validation Loss: 1.110
Epoch: 3, Validation Loss: 0.938
Epoch: 4, Validation Loss: 0.901
Epoch: 5, Validation Loss: 0.812
Epoch: 6, Validation Loss: 0.785
Epoch: 7, Validation Loss: 0.729
Epoch: 8, Validation Loss: 0.708
Epoch: 9, Validation Loss: 0.669
Epoch: 10, Validation Loss: 0.668
```

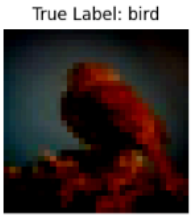
4. Experiments/Simulation Results

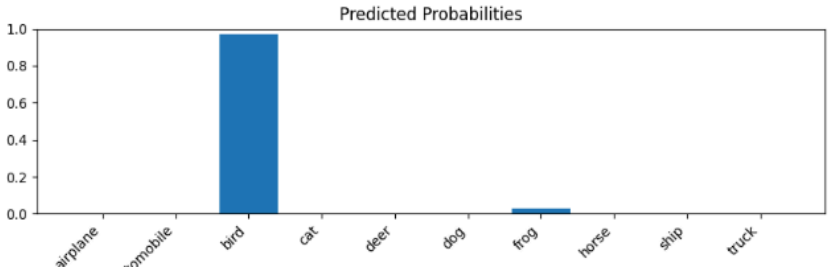


Next we show our model's confusion matrix. One can see some interesting insights here for example when the image is an automobile the model predicts it correctly most of the time however when it fails it mostly classifies this as a truck. We can also see that cats and dogs produce similar results. as opposed to, for example, airplanes and dogs. This is not surprising as these images are similar in some sense and would expect it to be harder for the model to classify correctly.

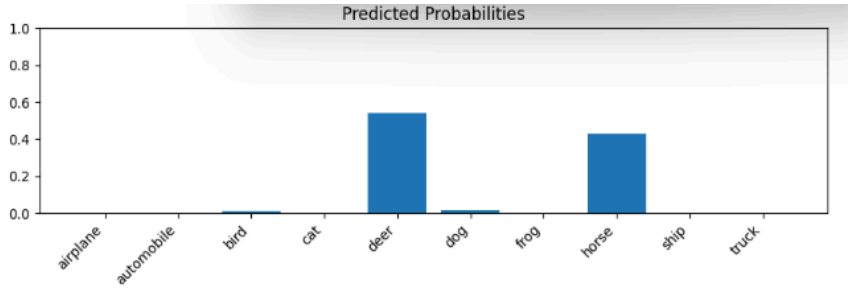
Finally we showcase some interesting cases of our model:



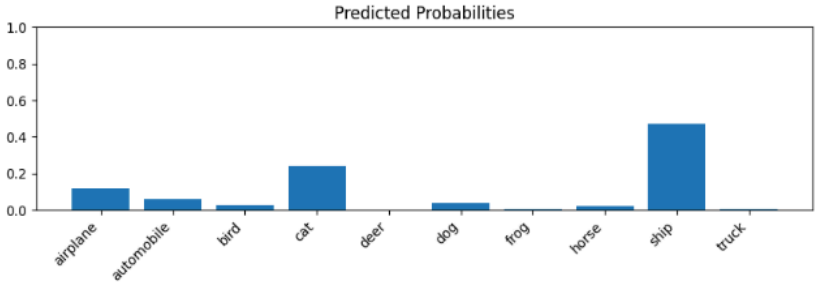




One can see in this first example our model classified the image as a bird although its ground truth is a frog, an image below shows why the might occur as images of frogs and birds can be very similar.



Here One can see our model classifies the image correctly as a deer. although it also states with a high probability (less than that of deer) that the image belongs to class of horse. this is understandable as deers and horses are similar in some sense



Here one can notice the ship's “pointy” edges, this might explain why the model seems to also state that the class has a certain small but not minor probability that the image belongs to the class of cats as cats have “pointy” ears as features as well.

5. Previous Attempts:

In earlier phases of the project (Part 1 and Part 2), we experimented with a simple SoftMax, SoftMax and a shallow CNN. However, these models failed to achieve satisfactory results in terms of classification accuracy and were outperformed by our deep CNN despite the fact that they indeed did outperform the dummy classifier and therefore do hold some merit.

The research conducted adopted a systematic approach to model development and evaluation:

Baseline Models:

- Dummy Classifier.
- Simple SoftMax: A basic SoftMax model for classification.
- Softmax: A shallow neural network with one hidden layer using the softmax activation function.

Complex Models:

- CNN: A Convolutional Neural Network with multiple convolutional layers followed by fully connected layers.
- Deep 22 Layer CNN: An extended version of the CNN model with additional convolutional layers and batch normalization.

Current Results:

Dummy Classifier: Achieved an accuracy of **10%**, serving as a baseline reference.
simple SoftMax: Improved accuracy to **40%**.
Softmax: Further improved accuracy to **52%**.
CNN: Achieved an accuracy of **59%**, outperforming previous models.
final result: Deep 22 Layer CNN: Significantly improved accuracy to **76%**, highlighting the efficacy of deeper architectures.

We monitored loss during training to ensure models were learning effectively and not overfitting the training data.

6. CONCLUSION

In conclusion, we have successfully implemented and evaluated a range of deep learning models for the CIFAR-10 dataset. Our findings demonstrate the critical role of model complexity in achieving higher classification accuracy. While simpler models provide a baseline, deeper architectures like CNNs exhibit superior performance. These results underscore the importance of leveraging deep learning techniques for image classification tasks, particularly when dealing with complex datasets like CIFAR-10.