CECS 456 Final Report

Authors: Adrian Seth, Kelly Duangrudeeswat, Luke Sunaoka, Matthew Quinn
https://github.com/sethadre/MachineLearningFinalProject

Introduction

In this project, we created a deep learning project that used various neural networks to classify animal images provided by Kaggle's website. To achieve this, each member trained a unique neural network using various methods, with the goal of optimizing accuracy on a dataset of 10 types of unique animals. Each one of our four group members independently implemented their own neural network model in python using Google Colab, Keras, and Tensorflow. Upon completion, we compared our work and analyzed the aspects of the models that led to the various levels of success achieved.

Dataset/Related work

Our group trained four different convolutional neural networks using the Animals-10 data set from Kaggle. The dataset included twenty-eight thousand medium-quality images belonging to 10 categories of animals, each with varying amounts of images. The animal categories and number of images are as follows: dog 4863, horse 2623, elephant 1446, butterfly 2112, chicken 3098, cat 1668, cow 1866, sheep 1820, spider 4821, and squirrel 1862. An issue that needed to be addressed was the fact that the image dataset was imbalanced and stored in a foreign language. To assist our report, we used Animal Classifier by Ayushi Mishra and VGG19_classifier by Abanob Morgan, as well as documents, lecture slides, and homework provided by CECS 456.

Methodology

Image recognition is an ongoing practice used to train neural networks. Convolutional neural networks, or CNNs, demonstrate their superiority through improved performance and more accurate results over other neural networks that can perform this task. Each member designed and created a CNN in order to classify our dataset. Two members, Matt and Adrian, designed their model using the guidance of homework and the CNN demo provided in class. The other two members, Luke and Kelly, designed their respective models based on VGG-16 and

AlexNet. Each member had to split the dataset, normalize it, and incorporate layers of convolution, pooling, flattening, and full connection in their model.

Experimental Setup

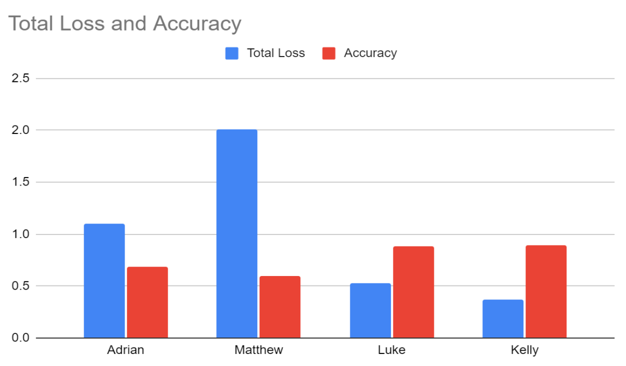
Our first model, designed by Adrian, first implemented eight layers of convolution. Each convolution was paired followed by a layer of MaxPooling. The convolution started with 16 filters, doubling after each pair and ending in 128 filters. This was all then flattened and followed by three fully connected layers. To train the model, the image set was split, 20% was used for testing, and 20% of the remaining set was used for validation, resulting in 16,721 images used for training, 5,226 for testing, and 4,181 for validation. For this model, the imbalance was not taken into account. The model was then compiled using a batch size of 200 and ten epochs where each epoch on average took 7 seconds.

The second model was created by Luke in which he recreated a design similar to that of VGG-16. First, the dataset was split and made sure to undersample the 26179 images down to 20943 in order to account for the imbalance dataset. Secondly, one hot encoding was applied to all of the data, each assigned to binary values, making them easier to sort for a better prediction. The model started with a convolution of 16 filters and would be pooled after. Thirdly, there were 2 convolutional layers each with 64 filters followed by another pooling layer. From there, there were 3 more convolution layers with 512 filters followed by a pooling layer repeated twice. This was all flattened and connected 3 times using ReLu as the activation function. The model was then compiled with a learning rate of 0.001. Finally, the model would be fit with a batch size of 100 alongside an epoch of 20.

The third model was developed by Matthew through his own personal design. This design also started with 16 filters and would work its way up to 256 filters, totaling to 9 convolution layers and 5 pooling layers. Next, the layers were flattened and fully connected 3 times with a ReLu activation function followed by 2 dropouts. The model was then fit with a batch size of 100 and epoch of 20.

The last model, created by Kelly, was based on AlexNet. The dataset was extracted and split into 2 categories: training (20,858 images) and testing (5,320 images). Next, feature scaling was performed to boost the model's performance. To ensure the data dimensions were approximately the same scale, both datasets were converted to floats and normalized to the range

0-1. The batch size was set to 53 and the epoch was 69, with the accuracy increasing along with the number of epochs. The training parameters consisted of 8 layers total, 5 being convolutional layers, and the last 3 were fully connected layers. After using the ReLu activation function, normalization is performed. Layer 6 was flattened before it was passed to the dense layers, 7 and 8. For the last output layer, the softmax activation function was applied to account for probability distribution.



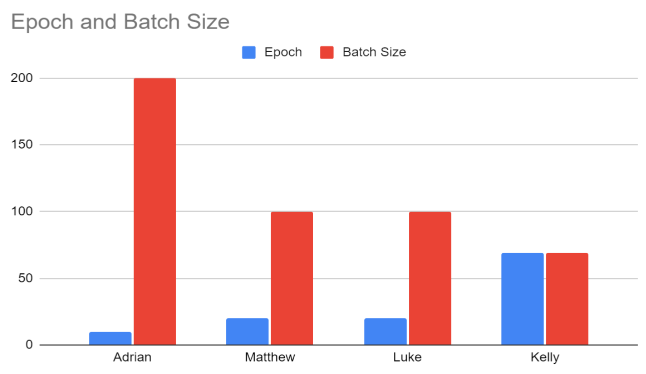


Figure 1. Total Loss and Accuracy comparison

Figure 2. Epoch and Batch size comparison

Results Measurement/Analysis

Once complete, each model was evaluated using the testing set. Adrian's design had a total loss of 1.1 with an accuracy of 0.69, Luke's model had a total loss of 0.52 with an accuracy of 0.88, Matthew had a total loss of 2.01 and an accuracy of 0.6, and Kelly had a total loss of 0.37 and an accuracy of 0.89.

Table 1. Test Set Metrics

CNN model	**Author**	**Accuracy**	**Total Loss**
Custom	Adrian	0.69	1.1
VGG-16	Luke	0.88	0.52
Custom	Matthew	0.6	2.01
AlexNet	Kelly	0.89	0.37

Comparison

As observed by Figure 1, the members who implemented their own design had an overall higher total loss and a lower accuracy. The least accurate model, which was developed by Matthew, had an accuracy score of 0.60 and was not modeled using any existing architecture. On the other hand, our best model, which was created by Kelly, achieved an accuracy score of 0.89 and was modeled after AlexNet. This was expected as Kelly had implemented a known CNN architecture while Matthew made a model from scratch. Another interesting point of comparison could be made between Luke's VGG-16 implementation and Kelly's AlexNet model. Despite it being known that VGG is more accurate than AlexNet, the data says otherwise. We believe that this is due to Kelly's model running more epochs compared to Luke's model (as shown in figure 2).

Conclusion

Our team implemented different variations of CNN (VGG-16, AlexNet, and two different original designs) in order to understand how much discrepancy there was between each model. Through this project, we were able to conclude that both the VGG-16 and AlexNet were the superior designs with AlexNet taking the edge, considering that it had a higher epoch size. One main obstacle that we faced was how to address the imbalance within our dataset that is reflected in some of our models' results. In the future, it would be interesting to test our designs through different image samples and see how we can improve on these models based on our discoveries and newfound knowledge.

Team Contribution

Team member name	**Task completed**
Matthew	Introduction, Results Measurement
Kelly	Conclusion, Images (Figures 1 and 2, Table 1), Team Contribution
Luke	Dataset/Related work, Analysis
Adrian	Methodology, Comparison
All members	Experimental Setup