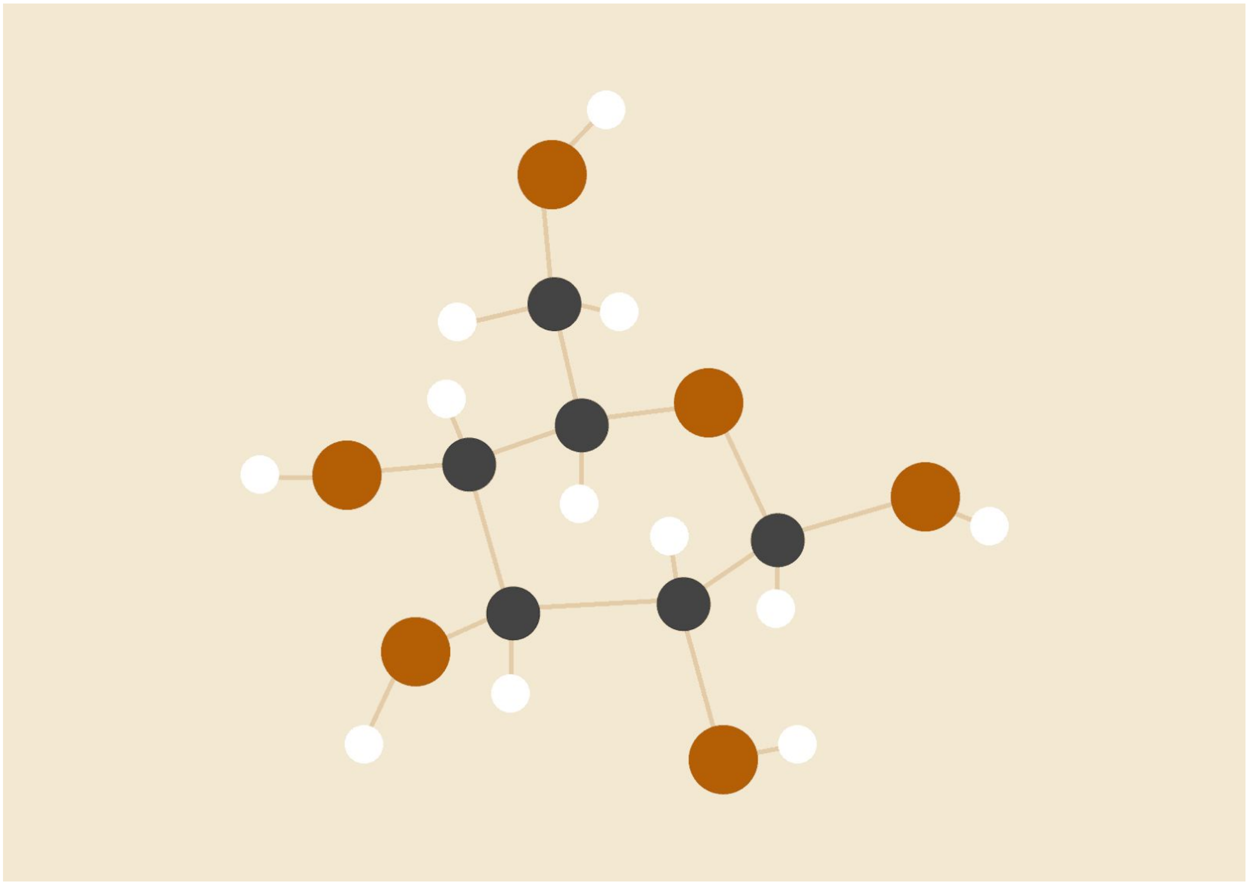
Advanced Machine Learning

Project Report

Lab Exam 3



Sumeet Padavala	01FB15ECS315
Ankit Anand	01FB15ECS041
Abhijay Gupta	01FB15ECS005

INTRODUCTION

The goal of this assignment is to explore the effect of various activation functions in a multi-layer perceptron.

In this report, we show our results from testing a standard MLP with the ReLU, cos, and ELU activation functions. We analyze the performance of each network, looking for key indicators like accuracy and convergence time. We also present data to back up our claims.

INPUT DATA

We have performed this analysis on the standard MNIST dataset. We used the original idx3 and idx1 formatted files. We used a third-party python package `idx2numpy` to convert then to numpy arrays.

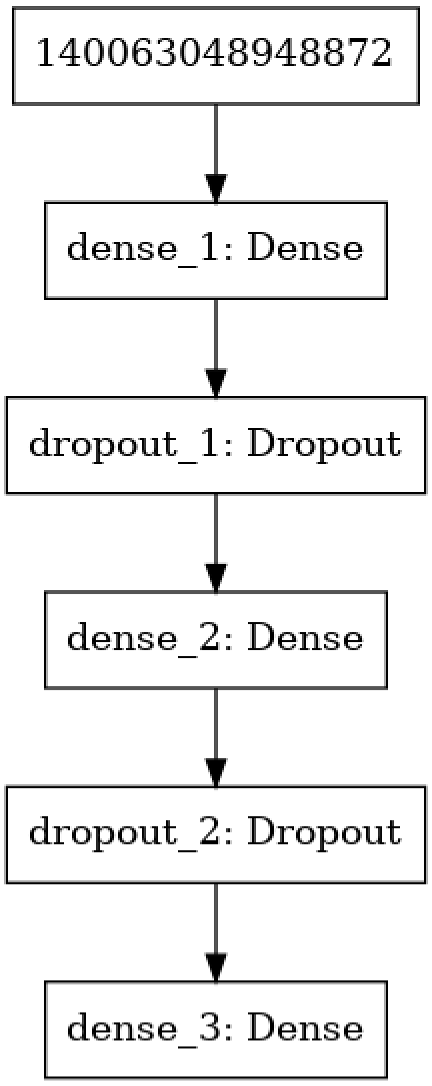
In our preprocessing step, we flattened each image to a 784-length vector, and one-hot-encoded the output.

We obtained a total of 60,000 training samples and 10,000 testing samples.

MODELS

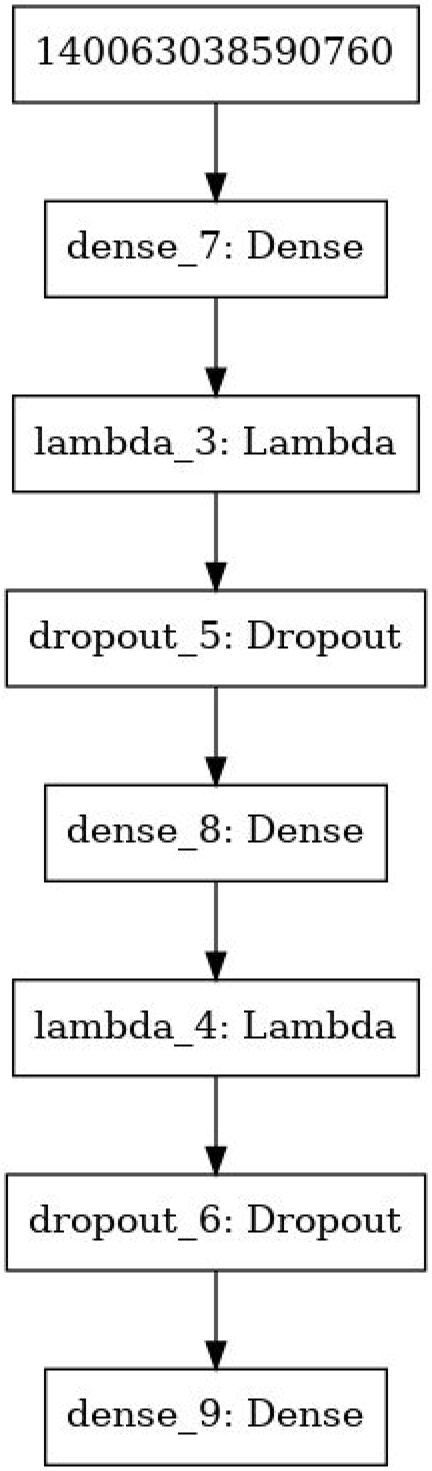
In this experiment, we have used a total of three models. The first is a simple baseline model, copied from Keras sample code.

This model consists of two blocks consisting of a Dense layer with the ReLU activation function and a dropout layer. Finally, the output layer of the model is a Dense layer with the softmax activation layer.



The second and third model are fundamentally the same as the baseline model, except that they replace the ReLU activation function with the cosine and ELU functions respectively.

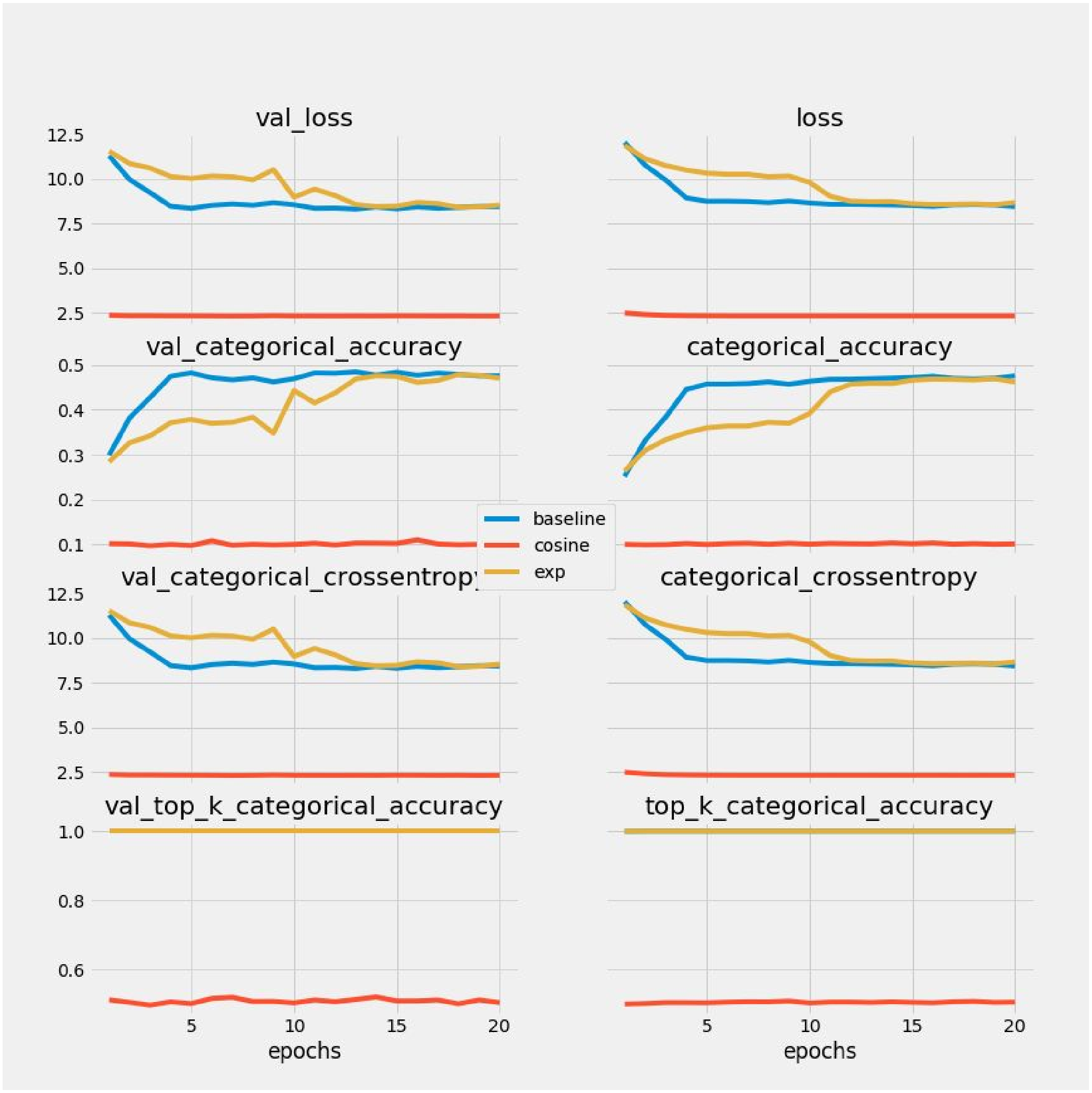
Since the Keras Dense layer does not support either of these functions as the activation function, we have solved this problem by changing the activation functions of the Dense layers to linear (identity), and then implementing the loss function through a Lambda layer. The layers were created using the keras.backend.cos and keras.backend.elu functions respectively.


140063038590760
dense_7: Dense
lambda_3: Lambda
dropout_5: Dropout
dense_8: Dense
lambda_4: Lambda
dropout_6: Dropout
dense_9: Dense

RESULTS

We ran each of the three models on 60,000 data samples for 20 epochs, with 10,000 samples used for validation.

The data we collected is summarized in the chart below.



From these charts, we can make a few inferences:

1. There is little to no overfitting, and each model has eventually converged to its optimal state. This removes all of the potential confounding factors related to improper training.
2. The cosine-based model performs terribly, failing to improve at all and showing a significantly worse performance than the baseline.
3. The baseline model converges faster than the ELU model.
4. After convergence, the baseline and ELU models perform almost identically.

CONCLUSION

In this analysis, we have compared the performance of three different activation functions on the performance of a simple multi-layer perceptron network trained on the MNIST dataset.

From our results, we have concluded the following:

1. The cosine function is an absolutely terrible function for neural networks. The model using a cosine function is unable to train at all, and performs no better than random chance. This indicates that periodic activation functions are probably a poor choice for neural networks.
2. The ReLU-based network converges faster than the ELU-based network, contrary to what should be expected according to literature. We also find that the two activations function almost identically after convergence. We believe that this occurs because the training data is very small and the MNIST task is very easy, making a subtle difference in performance between the two activations impossible to distinguish. The convergence speed difference is likely due to the fact that the ReLU function is steeper in some parts, producing a greater gradient. This would usually cause overshooting, but probably works well in this case due to the simplicity of the task.

REFERENCES

1. Keras documentation

   https://www.keras.io

2. Keras MLP code:
   https://github.com/keras-team/keras/blob/master/examples/mnist_mlp.py