INTRODUCTION

This project attempted to use Caffe to run two convolutional neural networks with function and parameter adjustments. The goal was to accurately classify a variety of animal images as either a cat or a dog. The implications of creating a successful model could be used in many industries, especially those related to health and animal services.

Group work on this project included agreement on selection of topic, data set, and frameworks required. We were each responsible for working on portions of the group report, writing code, and debugging efforts throughout project implementation. Debugging included coordinated research and communication to move through each phase of the project.

DESCRIPTION OF INDIVIDUAL WORK

The initial topic and tool selection were straightforward. Both team members have dogs so animal classification was a relatable and mutually interesting focal point. The first data set I suggested using was through querying PetFinder's API with results returned in JSON format. Unfortunately, only 1,000 records per day are accessible through this method, which would have limited the quantity and quality of the unrefined data set. Additionally, the ability to download images was convoluted so this attempt was discarded. Ultimately, the data set used was found on Kaggle and had been previously assessed for image classification projects, which suggested it was a vetted data set.

The steps in deciding how to analyze this data set was by selecting the framework (Caffe) and determining a few networks to try. The first LeNet network used was taken from Exam 2. I recommended it as a baseline for simplicity and with the assurance that it had been successful with other image classification datasets. From this point, another LeNet structure was created with modified functions and parameters. Main changes were selected as a response to overfitting concerns and precision of the learning rate parameters. It was expected that selecting classification-specific functions (such as Cross-Entropy for loss) and minimizing the momentum and learning rate would encourage a higher accuracy score and lower the final loss value. The trade off would be the time it takes to train and test the model with a smaller learning rate step size. I created recommended prototxt files to try based on the various models and parameter changes. I also updated the training and testing Python file to reference the appropriate architecture and solver files for each network.

Additional work was required to preprocess the data for the models to run in Caffe. I assisted by

creating a script to rename the files and split them into train and test folders. I attempted to create a bash script to download the files and a bash script to create the lmdb. These two scripts were not successfully implemented. It was recommended to us to follow the example provided in Dr. Jafari's GitHub Deep-Learning repository on creating the lmdb so we edited the create_lmdb_tutorial.py file located on the site instead. Within this process, debugging was required to make the lmdb functional. I spent time researching the error and considered that not all images were in the proper format. I was unable to locate the specific images; however, Leslie was able to successfully correct the error and implement the lmdb's in her environment.

Additional contributions I made were heavy assistance in writing the paper, README.md file, and creating the PowerPoint presentation. The paper required researching and locating applicable references for our project's goal. I worked heavily on the introduction, data, and model design and instrumentation sections. I also contributed to the conclusion section and recommendation for project improvement. I attempted to contribute to the results section and recommended parameter values to test. I was limited by my inability to create the lmdb's required to run the code.

RESULTS

Due to the errored code, my contribution to the results were in my recommendations made for function and parameter adjustments. Below are tables associated to the three model variations that I suggested to Leslie during the initial and mid-stages of our project. Edited versions of these tables can be found in the group report with the versions implemented in the models.

Model	Batch Size	Backend DB	Convl Layer Count	Fully Connected Layer Count	Dropout Layers	Net Input Functions	Transfer Function	Loss Function	Output Function
LeNet	64	lmdb	2	2	0	Inner Product	ReLU	Stochastic Gradient Descent (SGD)	Softmax

Table 1. Baseline model and functions.

Test Iteration	Test Interval	Base Learning Rate	Momentum	Weight Decay	Gamma	Power	Max Iterations
100	500	0.01	0.9	0.0005	0.001	0.75	100

Table 2. Baseline parameters and values.

The variations in layer count, functions, and parameters were all adjustments made with the intention of affecting the accuracy and providing control measures for the risk of overfitting.

Model	Batch Size	Backend DB	Convl Layer Count	Fully Connected Layer Count	Dropout Layers	Net Input Functions	Transfer Function	Loss Function	Output Function
LeNet	1	lmdb	3	3	2	Inner Product	ReLU	Cross Entropy	Softmax

Table 3. Updated LeNet model and functions.

Test Iteration	Test Interval	Base Learning Rate	Momentum	Weight Decay	Gamma	Power	Max Iterations
50	200	0.001	0.5	0.0001	0.1	0.75	100

Table 4. Updated LeNet parameters and values.

AlexNet was chosen as a comparative model due to its' popularity after achieving top five results in a 2012 image classification competition. The basis relies on the specifics outlined in Table 5. Parameters from the LeNet baseline were applied to the AlexNet to enhance the performance comparisons based on layer and function selections.

Model	Batch Size	Backend DB	Convl Layer Count	Fully Connected Layer Count	Dropout Layer	Net Input Functions	Transfer Function	Loss Function	Output Function
AlexNet	1	lmdb	5	3	5	Inner Product	ReLU	SGD	Softmax

Table 5. AlexNet model and functions.

Test Iteration	Test Interval	Base Learning Rate	Momentum	Weight Decay	Gamma	Power	Max Iterations
100	500	0.01	0.9	0.0005	0.001	0.75	100

Table 6. AlexNet parameters and values.

CONCLUSION

Initially, I was motivated to select Caffe for our convolutional neural network due to the ease and clear approach of structuring and running the model. In the process, I learned how much nuance existed in the data preprocessing stage in order to successfully run a model on Caffe. This was further complicated by the sparse documentation available to understand the code and debug issues. Caffe is certainly an effective framework for convolutional neural networks; however, I will be more cautious about whimsically selecting tools and frameworks for future projects. It's commonly understood that documentation is important in programming, and yet, it often seems to be the afterthought. This experience has taught me the value of clear communication and thorough documentation.

Table 7 below includes each of the coding files I created, and the percentage of code borrowed from other sources. Other sources include the Exam 2 files, Dr. Jafari's GitHub repositories, and the internet.

Code Files	Borrower's Percentage (%)
README.md	0
Train_oxford.py, Tran_oxford_var1.py	90
create_lmdb.py	100
renameFiles.py	0
traintestsplit.py	0
Lenet_solver_oxford.prototxt	100
Lenet_train_test_oxford.prototxt	97
Lenet_solver_oxford_var1.prototxt	73
Lenet_train_test_oxford_var1.prototxt	21
Alexnet_solver_oxford.prototxt	96
Alexnet_train_test_oxford.prototxt	97
Get_oxford.sh, Create_oxford.sh	23, 80

Table 7. AlexNet parameters and values.

REFERENCES

dr. Avicenna (2019). Cats and Dogs Breeds Classification Oxford Dataset. Retrieved from https://www.kaggle.com/zippyz/cats-and-dogs-breeds-classification-oxford-dataset#annotations.tar.gz

Motta, D., Santos, A. Á, Winkler, I., Machado, B. A., Pereira, D. A., Cavalcanti, A. M., . . . Badaró, R. (2019). Application of convolutional neural networks for classification of adult mosquitoes in the field. Plos One,14(1). doi:10.1371/journal.pone.0210829

Sun, T., Sun, L., & Yeung, D. (2017). Fine-grained categorization via CNN-based automatic extraction and integration of object-level and part-level features. Image and Vision Computing,64, 47-66. doi:10.1016/j.imavis.2017.06.003

Taheri, S., & Toygar, Ö. (2018). Animal classification using facial images with score-level fusion. IET Computer Vision,12(5), 679-685. doi:10.1049/iet-cvi.2017.0079

*20190428 23:44: Update – I have been able to successfully create lmdb's and run the various versions of our network! An additional network, LeNet version 2 has also been added to the mix as the optimal model we were able to achieve.