Is It Raining Cats or Dogs?

Leslie Juengst & Samantha Roberts

DATS 6203: Group 5 Final Report

**INTRODUCTION**

Image classification and computer vision have grown increasingly more popular as new technologies have emerged. The evolution of robust tools for machine learning has led to new boundaries of more intricate hypotheses and novel practical applications of results. For example, the ability to properly classify images of different species of animals is a complex and delicate problem. The refinement and satisfactory solution to a problem such as this has wide application, such as predictive capabilities for agriculture, health services and animal services (Motta, et al., 2019).

The goal of this project was to correctly classify a variety of cat and dog images according to species. An assortment of models has been adapted to handle the inconsistent nature of images and feature extraction, varying specifically in neural layers and input functions to capture the most specific degree of variation in multiclass problems (Taheri & Toygar, 2018; Sun, Sun, & Yeung, 2017). This project implemented two separate convolutional neural network (CNN) algorithms with a varied combination of parameters to find the prime classification strategy for the selected data set.

**DATA**

The data set selected is “Cats and Dogs Breeds Classification Oxford Dataset” on Kaggle.com (dr. Avicenna, 2019). It is comprised of 37 categories of animals, with approximately 200 images per class for a total of 7,393 images. It was expected that this sizing would provide enough samples for each category to potentially identify unique feature maps for accurate classification purposes. The images themselves were stored as jpg files, which required no nuanced processing. All files were resized to 80 x 80 to be unified and fulfill Caffe’s general requirements. The images were also renamed and separated based on whether it was a cat (0) or dog (1). Premade training and testing files were discarded in lieu of a custom 70/30 split train and validation sets. The new training file contained 5,175 records and the new testing file contained 2,218 records.

**MODEL DESIGN AND INSTRUMENTATION**

The neural network model class best suited to address image classification was a CNN. The main algorithm selected for evaluation was LeNet. Customization was introduced by attempting a variety of output layer transfer and optimization functions, although the basic structure of the CNN remained the same. The framework for implementing this network was Caffe due to its processing ability specific to this type of neural network.

The first iteration of the neural network used the new training and testing sets that were created with an architecture and solver identical to the recent mini-project provided in academia. The experiment design was kept consistent to develop an initial baseline with the new data set based on a previously successful structure. The performance of the network was measured by training and testing times, loss function output, and the test set accuracy rating.

Model and parameter selection were used as a main source of manipulation to obtain the network’s maximum accuracy score. A baseline was developed using a combination of functions and, which can be found in Table 1 (models) and Table 2 (parameters). These values were chosen based on default values often selected when using CNN’s, were implemented in similar projects conducted by the authors, or were corroborated with other animal classification CNN studies when information was available (Taheri & Toygar, 2018; Motta, et al., 2019). A singular run of the data was also passed into the pre-trained AlexNet network with no customization efforts for result comparison.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 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 risk of overfitting on the small data set was mitigated through the implementation of multiple parameter and architectural design combinations. The baseline model and parameter choices provided an ideal selection to negate overfitting by using dropout layers, the ReLU transfer function with a cross-entropy loss function, and the Softmax output function. Additionally, a low number of layers is optimal for a less robust data set.

**RESULTS**

The first iteration was the baseline LeNet model using previously established model functions and parameters. Without any amendments to the existing model architecture and parameters, the accuracy score was ##% with a final loss value of ##. It took ## seconds to train the model and ## seconds to test it. Running this version created a baseline that could be used for comparison with function and parameter modifications. Additionally, this provided additional basis for comparison with the AlexNet convolutional neural network model.

The second iteration was the LeNet model with updated architecture and parameters. (This version did worse than the baseline, which was unexpected due to the updated learning rate decay parameters and updated loss function to cross entropy.) This version did better than the baseline, which was expected due to refining the learning weight factors and updated loss function to cross entropy. The accuracy score was ##% with a final loss value of ##. It took ## seconds to train the model and ## seconds to test it. The greatest difference from the baseline model was in the loss function value at ###. Updated LeNet functions and parameters are defined in Table 3 and Table 4.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Batch Size | Backend DB | Convl Layer Count | Fully Connected Layer Count | Dropout Layers | Net Input Functions | Transfer Function | Loss Function | Output Function |
| LeNet | 50 | 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 | 0.01 | 0.75 | 100 |

Table 4. Updated LeNet parameters and values.

The third iteration was the AlexNet model with baseline architecture and parameters. (This version did worse than the LeNet baseline and the LeNet variation, which was unexpected because it is a newer model to image classification. AlexNet’s strength is the use of GPU’s to run five convolutional layers and three fully connected layers with dropout layers. ) This version did better than the baseline, which was expected due to refining the learning weight factors and updated loss function to cross entropy. The accuracy score was ##% with a final loss value of ##. It took ## seconds to train the model and ## seconds to test it. The greatest difference from the baseline model was in the loss function value at ###. Updated LeNet functions and parameters are defined in Table 5 and Table 6.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Batch Size | Backend DB | Convl Layer Count | Fully Connected Layer Count | Dropout Layer | Net Input Functions | Transfer Function | Loss Function | Output Function |
| AlexNet | 64 | lmdb | 5 | 3 | 2 | Inner Product | ReLU | Stochastic Gradient Descent | 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**

Summarize the results you obtained, explain what you have learned, and suggest improvements that could be made in the future.

The results of the (LeNet baseline/LeNet updated/AlexNet) network were the optimal results with an accuracy rating of ##%. At this percentage, it is/is not feasible to continue refining the model for more useful tasks, such as subspecies identification. Applications of subspecies classification have wide and immediate applicability in the fields of health and animal sciences. It will continue to be a worthwhile endeavor to improve the ability to accurately and efficiently classify animal species and subspecies.

Throughout the development of this project, we learned

One recommendation for improvement with this data set would be to implement a fine-grained categorization method using part-based features to determine the minute differences between breeds of the dogs and cats in the data set (Sun, Sun, & Yeung, 2017). This approach would focus on the feature maps located within the hidden convolutional layers. This technique would likely require a larger data set so random oversampling techniques would be required for optimal performance.

**REFERENCES**

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

**APPENDIX A**

|  |  |
| --- | --- |
| **Documented Computer Listings** | **Description** |
| README.md | Code description and technical requirements overview |
| Train\_oxford.py | Python file designed to train and test the network |
| Create\_lmdb.py | Processes images and creates the lmdb databases |
| renameFiles.py | Iterates through directory to rename all the raw images |
| Traintestsplit.py | Creates 70/30 split of files to create train and validation |
| Lenet\_solver\_oxford.prototxt | Baseline LeNet file that assigns parameters to the network’s functions |
| Lenet\_train\_test\_oxford.prototxt | Baseline LeNet file that provides the network architecture through layer design and order |
| Lenet\_solver\_oxford\_var1.prototxt | Updated parameters for comparison |
| Lenet\_train\_test\_oxford\_var1.prototxt | Updated architecture for comparison |
| Alexnet\_solver\_oxford.prototxt | AlexNet Model w/ baseline parameters for comparison |
| Alexnet\_train\_text\_oxford.prototxt | AlexNet architecture for comparison |
| Get\_oxford.sh | Bash script; Retrieves the data files |
| Create\_oxford.sh | Bash script; Creates the lmdb databases required for training/testing |
| Images.tar.gz | Zipped image data file; 7,393 .jpg files |
| Annotations.tar.gz | Zipped annotations data file; trimaps (7,390 .png files), xml (3,686 xml files), list.txt, README.md, test.txt, trainval.txt |