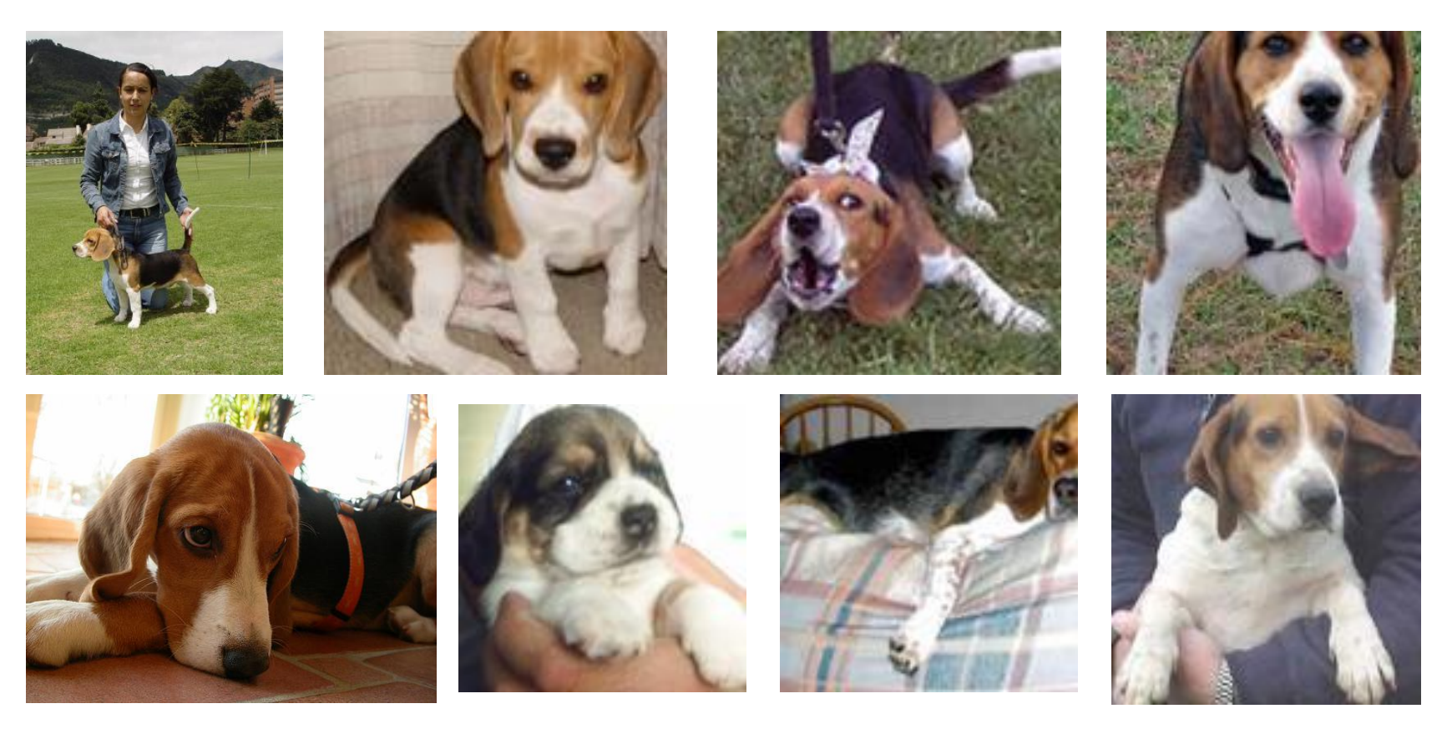
How Accurately Can a Convolutional Neural Network Identify a Dog's Breed

Ben Fritzeen, Dr. Art White
Taylor University

Introduction

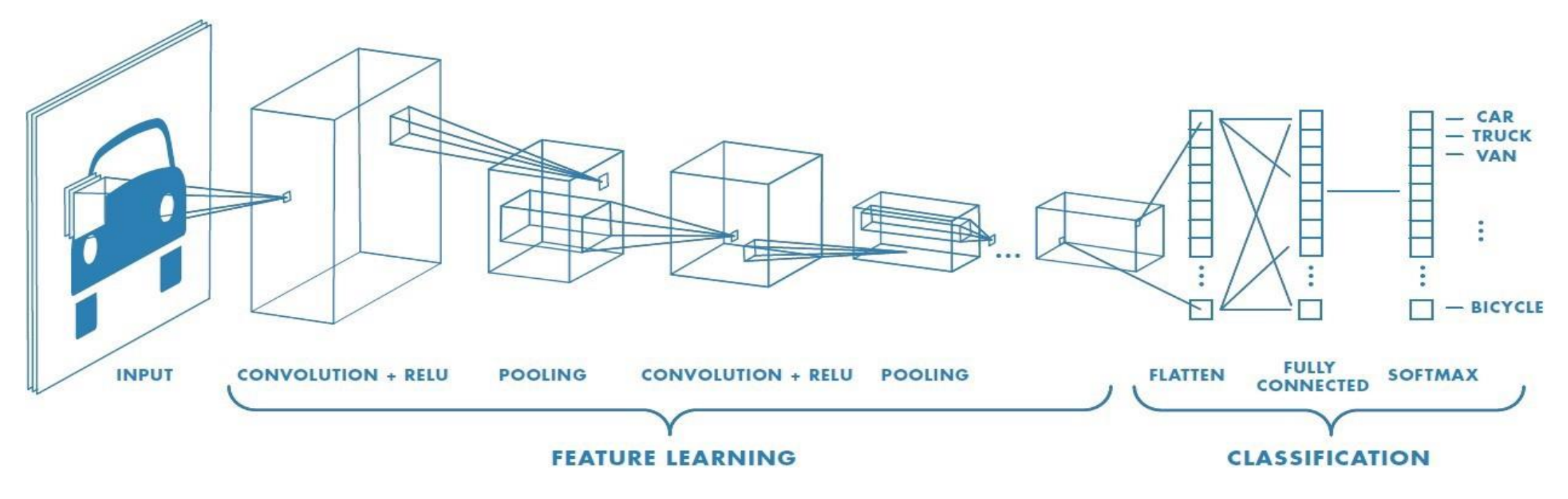
Kaggle is a data science and machine learning community, where large, interesting data sets are collected and posted, and data scientists model the data to solve difficult problems. One of the problems posted on Kaggle is to classify a dog's breed from an image, utilizing a small provided dataset of training and testing images. To solve this problem, we will be using Tensorflow, which is a tool that can be used to implement convolutional neural networks. we will also be using the training and testing datasets that are provided by the Kaggle competition. The dataset that Kaggle provided consists of 120 different dog breeds, and each breed has around 70-110 unique images to use for training.



These are just a handful of images of beagles from the provided dataset from Kaggle, which demonstrates the varying angles, lightings, and activities that each image depicts.

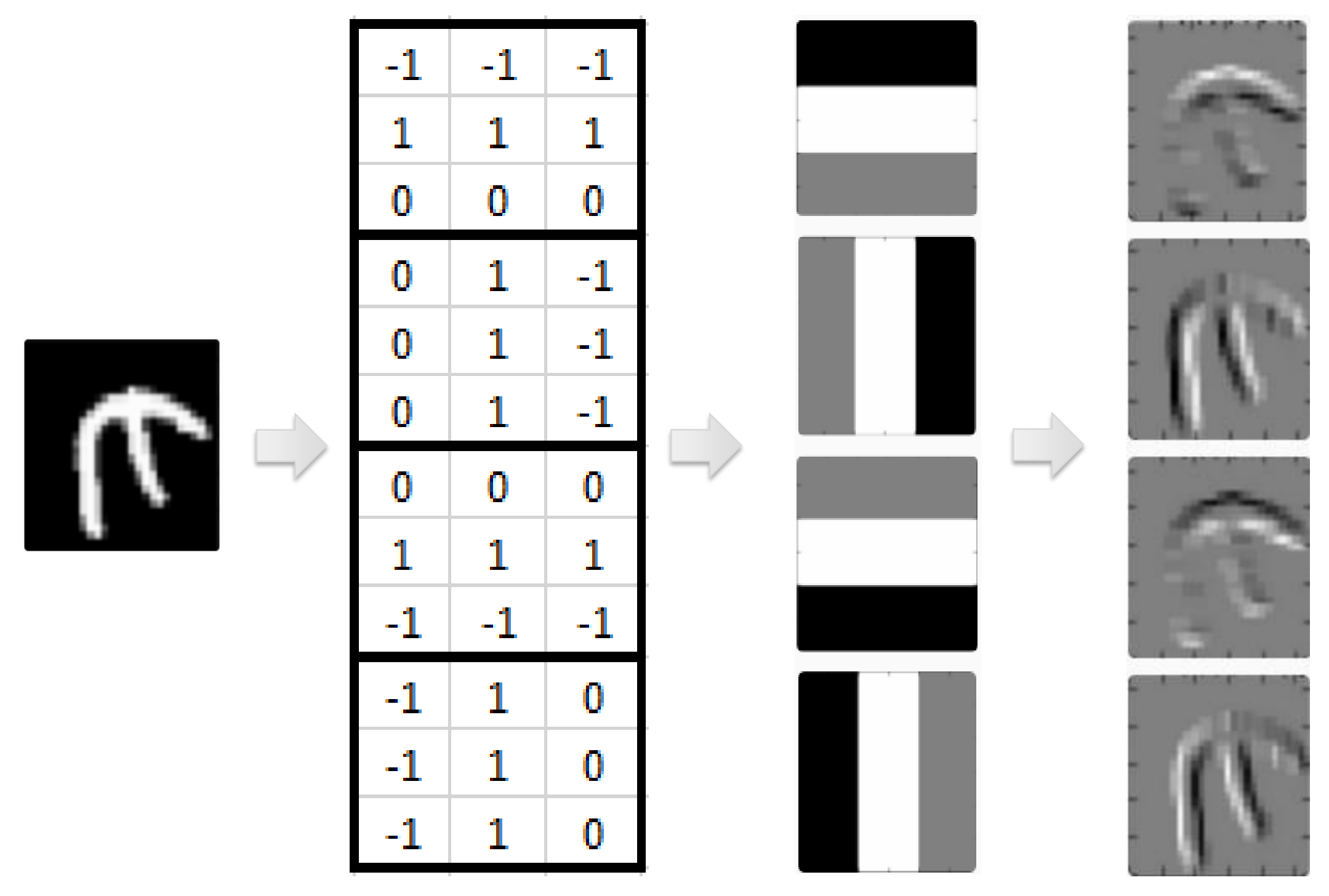
Methods

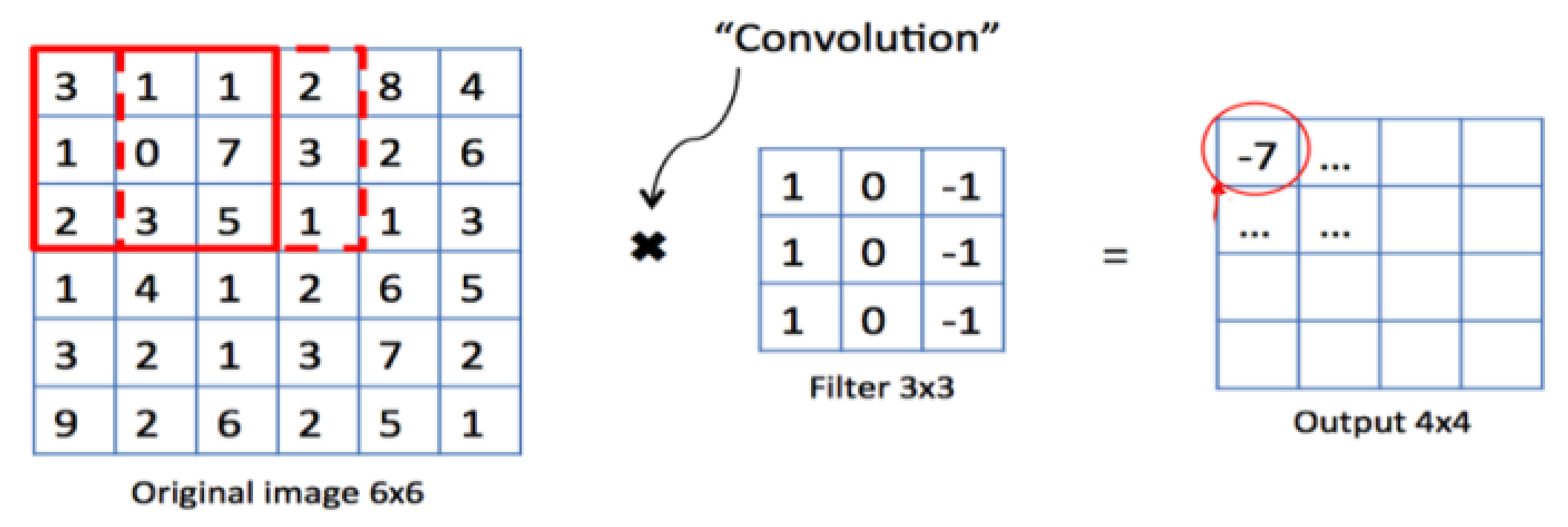
We plan to use a standard CNN pipeline for solving this classification problem. Our convolutional neural network consists of three convolutional layers, each of which use *Rectified Linear Units* (ReLU). ReLU is an activation function used in neural networks that will take any negative numbers in the image matrix, and convert them to zero. This helps make computation simpler, and can help training/testing take less time to run. After each convolutional layer, we use max pooling before we pass it into the next layer. After the last convolutional layer/max pool, we flatten the results. Flattening converts the image matrix into a vector of values. We then pass that into two fully connected layers. A fully connected layer essentially is a cluster of neurons that will over time capture elements of the input data. This data is then used in the classification prediction at the softmax layer. After the second fully connected layer outputs its results, we use an activation function called softmax, which will give a decimal value (0-1) based on the likelihood of it being a certain dog breed.

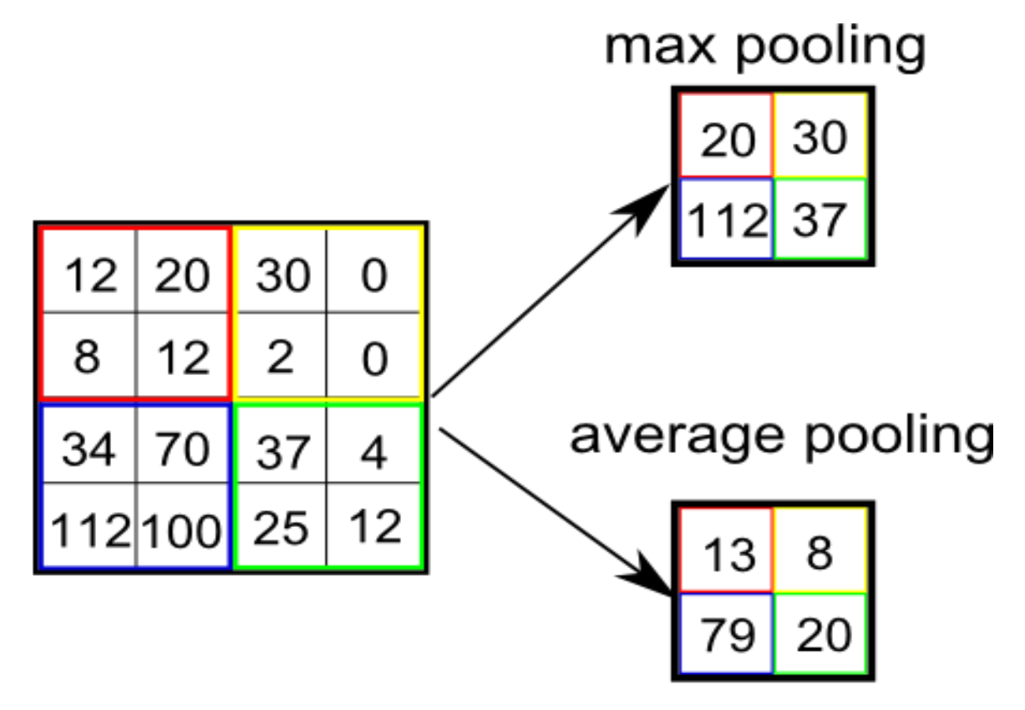


As stated previously, this is simply a standard image classification setup, so we will try to optimize the classification with several different optimizations. In regards to using softmax and ReLU, see the Activation Function Alternatives section for more information. Also see the Future Work section for more information concerning the possible changes to the actual implementation.

Convolutional Neural Network Details

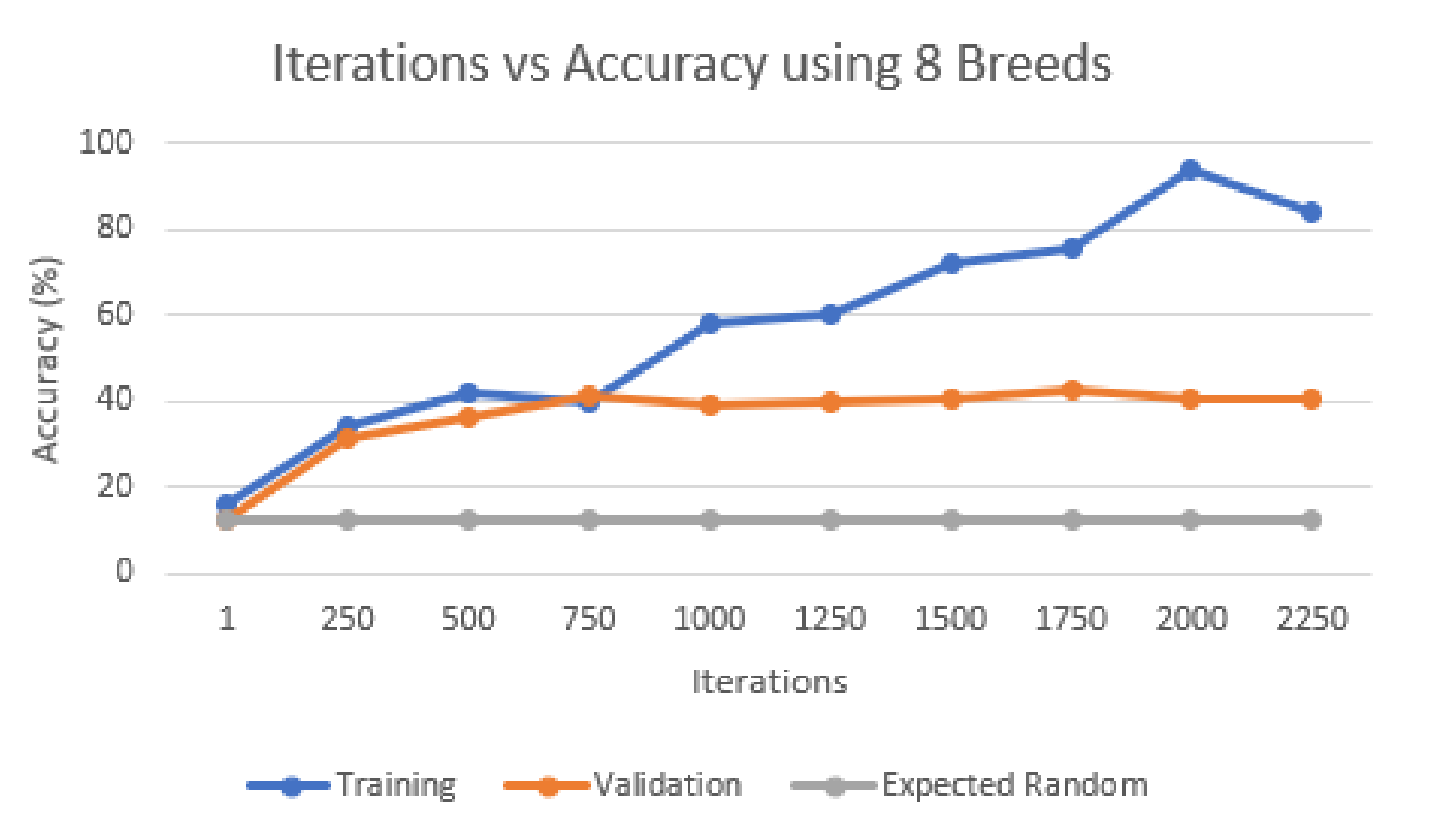




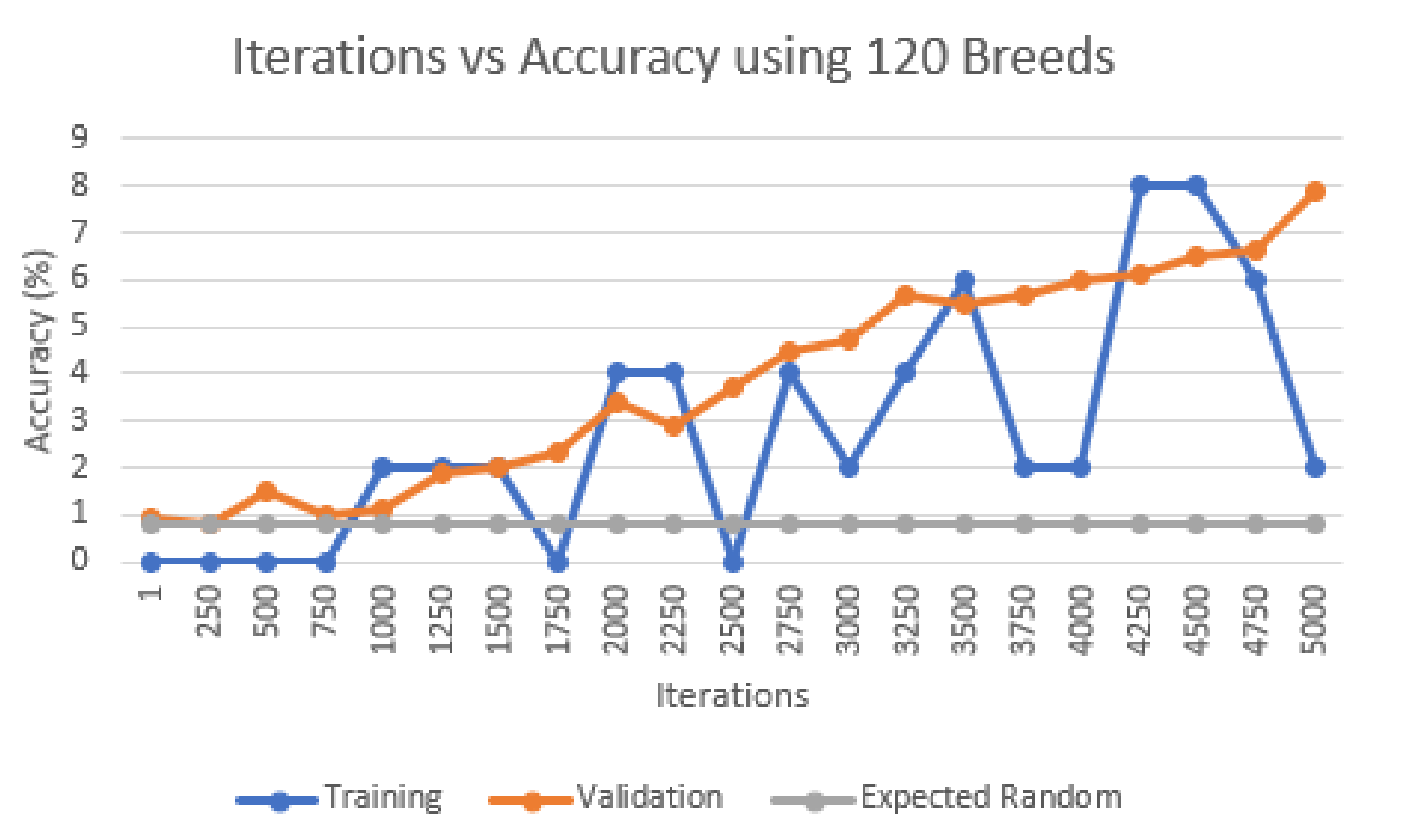


The topmost image depicts how different filters in a convolutional layer may affect the output. The picture below that demonstrates what actually happens in a convolutional layer. Essentially, a section is taken of the image matrix (bordered with solid red), and perform matrix multiplication with that and the filter, to produce the -7 in the right hand matrix. It then 'convolves' to the next chunk (bordered by dashed red), and repeats. The bottom most picture is an example of what occurs in the pooling section of the convolutional neural networks. Max pooling simply takes the largest value, while average pooling takes the average of all the values.

Results



The above graph is a test we conducted with 2250 iterations using only eight different dog breeds, and as you can see, the validation accuracy did not increase much over time. It started at 12.3%, as guessing 1 in 8 is a 12.5% chance. After 2250 iterations, it was at around 40%, which is much higher odds than randomly guessing.



The above graph is a test we conducted with 5000 iterations using all 120 dog breeds. As expected, the validation accuracy is around 0.9% at the beginning, as guessing 1 in 120 is a 0.8% chance. However, after 5000 iterations, it was at 7.9% validation accuracy, which is nearly ten times better than randomly guessing.

Note: Any large dips in the training percentage accuracy may be due to outliers, as a result of one test being done, rather than multiple tests averaged together.

Future Work

Currently, our convolutional neural network reaches 7.9% validation accuracy after 5000 iterations, which is about ten times better than randomly guessing, but it still is not very accurate. We intend to look into more options to increase our network's accuracy. One method is to rotate each training image, so that the small training dataset is now four times larger than before. Another method to increase the accuracy would be to utilize a pre-trained convolutional neural network, and apply its already trained filters to our data.

Literature cited

Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., Kudlur M., Levenberg J., Monga R., Moore S., Murray D., Steiner B., Tucker P., Vasudevan V., Warden P., Wicke M., Yu Y., & Zheng X. (2016). TensorFlow: A System for Large-Scale Machine Learning. Retrieved from Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, March 22, 2019.

Ciresan D., Meier U., Masci J., Gambardella L., & Schmidhuber J. (2011). Flexible, High Performance Convolutional Neural Networks for Image Classification. Retrieved from AAAI Publications, March 20, 2019.

Clément Farabet, Camille Couprie, Laurent Najman, Yann Lecun. (2013). Learning Hierarchical Features for Scene Labeling. (pp.1915 - 1929). Retrieved from IEEE Transactions on Pattern Analysis and Machine Intelligence, Institute of Electrical and Electronics Engineers, March 20, 2019.

Clevert D., Unterthiner T., & Hochreiter S. (2016). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). Retrieved from arVix.org, April 22, 2019.

Goldsborough P. (2016). A Tour of TensorFlow. Retrieved from arVix.org, March 22, 2019.

Krizhevsky A., Sutskever I., & Hinton G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Retrieved from Advances in Neural Information Processing Systems 25 (NIPS 2012), March 21, 2019.

Oquab M., Bottou L., Laptev I., & Sivic J. (2014). Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. Retrieved from IEEE Conference on Computer Vision and Pattern Recognition (CVPR), March 21, 2019.

Szegedy C., Toshev A., & Erhan D. (2013). Deep Neural Networks for Object Detection. Retrieved from Advances in Neural Information Processing Systems 26 (NIPS 2013), March 20, 2019.

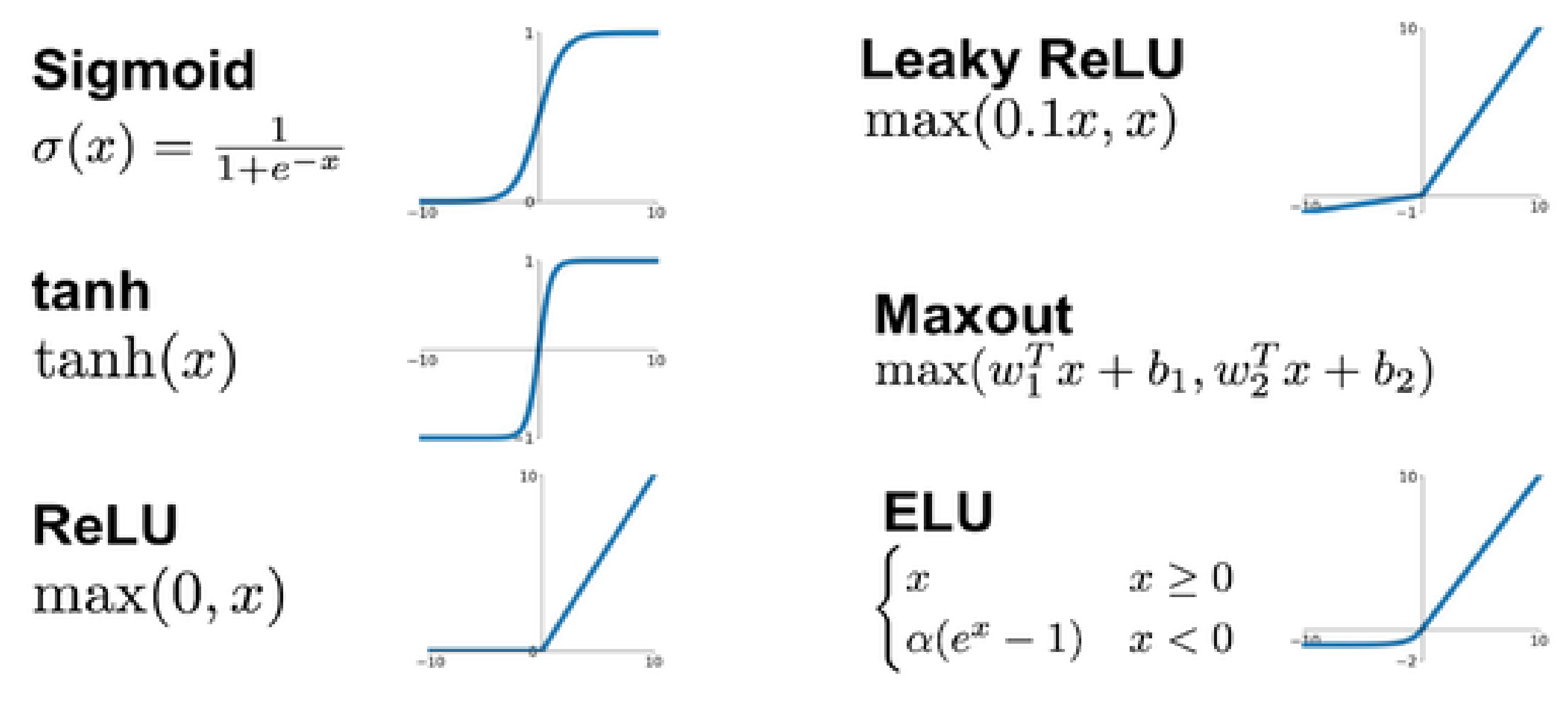
Acknowledgments

I would like to thank Dr. Art White for his guidance and helpful knowledge concerning the field of machine learning, which has helped me at several points in my research. We would also like to thank Dr. Jon Denning for walking with me through this process, and making sure that we're moving in the right direction.

Activation Function Alternatives

While we have used the standard *Rectified Linear Units* (ReLU) for our convolutional neural network, there are several alternatives that may be interesting to test, and compare their differences in performance. There are *Leaky ReLUs*, which allow a small, positive gradient when the unit is not active. There are *Parametric ReLUs*, which make the coefficient of leakage into a parameter that is learned along with the other neural network parameters. Finally, and possibly the most promising, are *Exponential Linear Units* (ELUs). ELUs try to make the mean activations closer to zero, which speeds up learning. It has been shown that ELUs can obtain higher classification accuracy than standard ReLUs, which makes it a promising future testing option.

We have also used *softmax* as the tool to assign a value from 0-1 for each classification possibility. There are a few other options depicted on the right. While most likely not worth using for this particular problem, *tanh* is useful for classifying between two classes, as it could be 1 or -1. *Sigmoid* is like softmax, but softmax gives each different class a value, and all those values add up to 1, while the values sigmoid gives do not necessarily need to add up to 1. Softmax is essentially a generalization of the sigmoid function, which is why we used it in our implementation.


Sigmoid
σ(x) = 1/(1+e^-x)
tanh
tanh(x)
ReLU
max(0, x)
Leaky ReLU
max(0.1x, x)
Maxout
max(w1^T x + b1, w2^T x + b2)
ELU
x, x ≥ 0
α(e^x − 1), x < 0