Analysis

Table of Contents

Introduction

Machine Learning is an application of AI that allows a computer to learn certain behavior and characteristics of a system without the behavior being explicitly programmed. Machine learning trains itself on a set of sample data. It then builds up a mathematical model to fit the data as best as possible.

Neural Networks are a machine learning technique inspired by interconnected biological neurons. They try to replicate data patterns the same way the human brain learns.

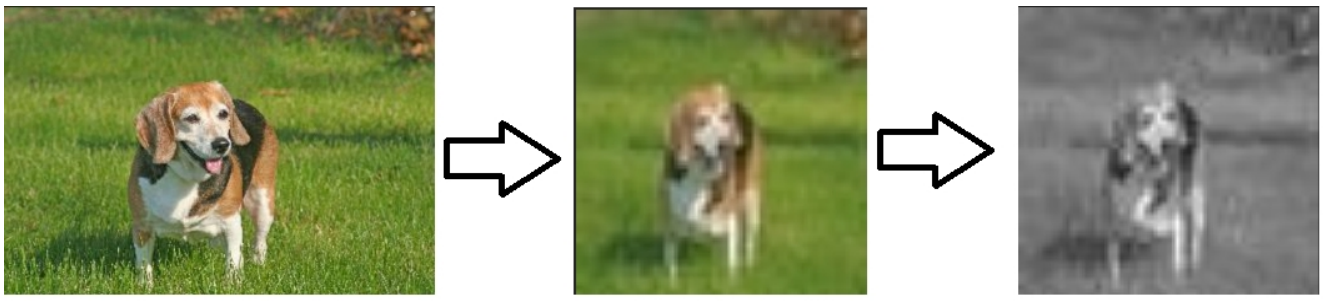
Problem Statement and Objective

The objective of this project was to solve the ‘Cats VS Dogs’ dataset published by Microsoft to Kaggle. Since this was an image classification task, Convolutional Neural Network architecture was selected. We aimed to get at least ~80% classification accuracy on highly pixelated images.

Methodology

Data Pre-Processing

The original dataset when obtained from Kaggle contains ~12,000 images of cats and dogs each of varying resolutions. In an idealistic case, we would operate directly on these high resolution images. But to account for resource constraints, all images were resized to 50px*50px monochrome images. This greatly reduces computational load while also requiring us to use a two dimensional convolution operation. The downside is that some subtle features that differentiate Cats and Dogs (fine lines and spots) are greatly diminished/absent.



350*233px(original) 50*50px(resized) 50*50px(bw)

Once all the images were downscaled to an appropriate resolution, The images were all stored in a numpy file along with the class label. This was effectively the final dataset. All

images of Dogs were assigned a label ‘1’, while cats got ‘0’. The final numpy file was 80MB compared to the original 860MB downloaded from Kaggle.

Choice of Network

For choosing the type of network, two choices were available. One was the highly inelegant method of using a standard feed-forward neural net. Such a net would have 50*50 input neurons. While such a network performs exceptionally on a simpler dataset like MNIST, it is extremely unsuitable for a complex case such as Cats v Dogs. The appropriate type of Neural Network for image classification tasks is a Convolutional Neural Network, which is what we have implemented.

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers and then followed by one or more fully connected layers as in a standard neural net. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features.

Each convolutional layer performs a sort of ‘feature abstraction’. This is repeated across multiple layers till recognizable objects (faces, objects, etc.) have been obtained. These features are sent as inputs to a feed-forward section for the final decision making to occur.

Tools and dataset

OpenCV and PIL were used for all image manipulation tasks. Pytorch was used as our neural net framework. It was chosen over Tensorflow for its better python implementation. Tensorboard was used for network evaluation and keeping track of training progress.

The dataset was from Kaggle

Implementation

The actual implementation of the neural network is largely trial and error based. Great freedom is available in the selection of network shape, activation functions, optimizer function, and learning rate.

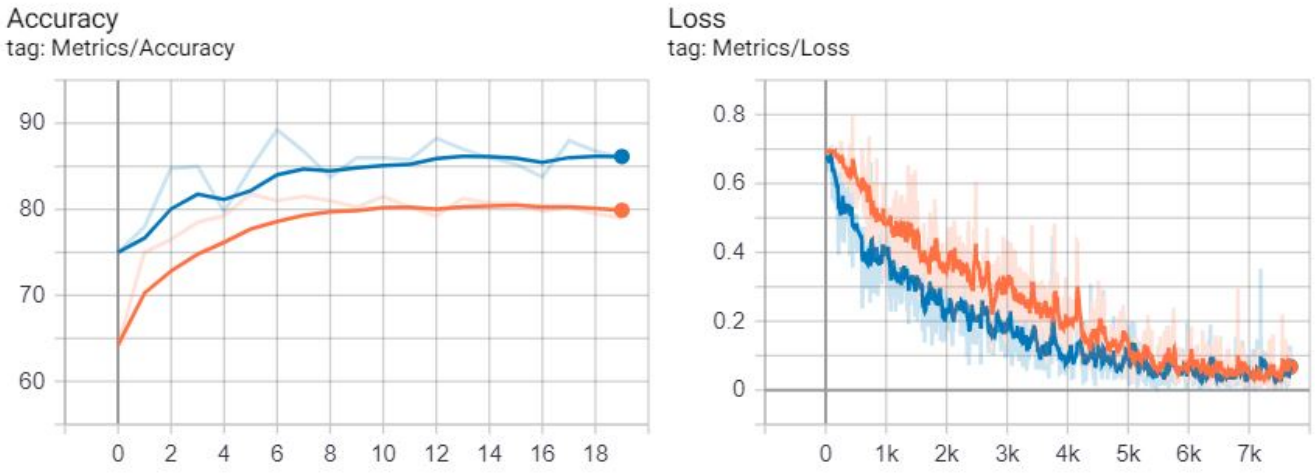
Network Shape

The network shape we chose used 7 convolution layers. The layer 1 takes a 1*50*50 input, where 1 represents the number of input channels (which is 1 because the image is monochrome). This passes through each convolutional layer which causes the number of

available features to increase exponentially. The final conv. layer provides 600 output features.

The output of some layers are max-pooled to stay within our compute constraints. This does of course cause a minor hit to our network accuracy. All outputs are also batch normalized to keep outputs ϵ (0,1).

An increasing number of convolutional layers does improve accuracy, but eventually it reaches a point of diminishing returns. For comparison, we pitted a less complex network against the ideal one and the accuracy difference is easily seen.

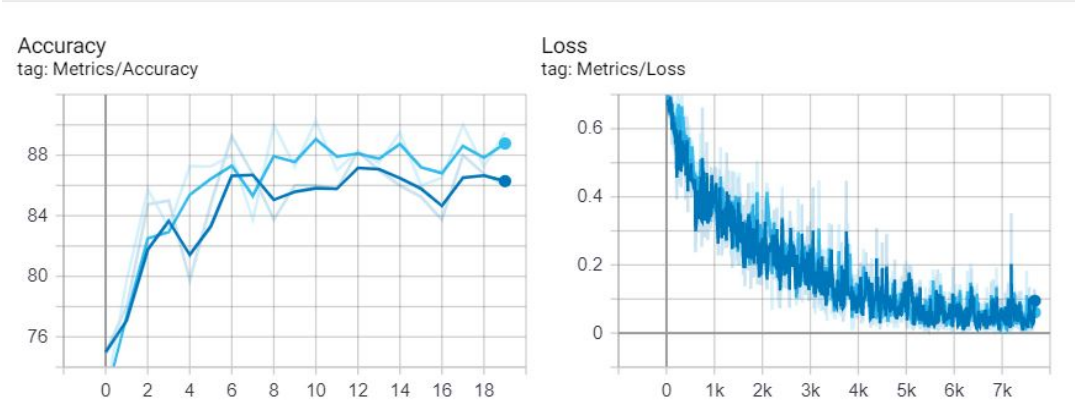


Blue → Best Net Orange → Simpler Net

It is observed that Blue(7 layers) easily outclasses Orange(5 layers) in terms of validation accuracy and net loss.

Activation functions

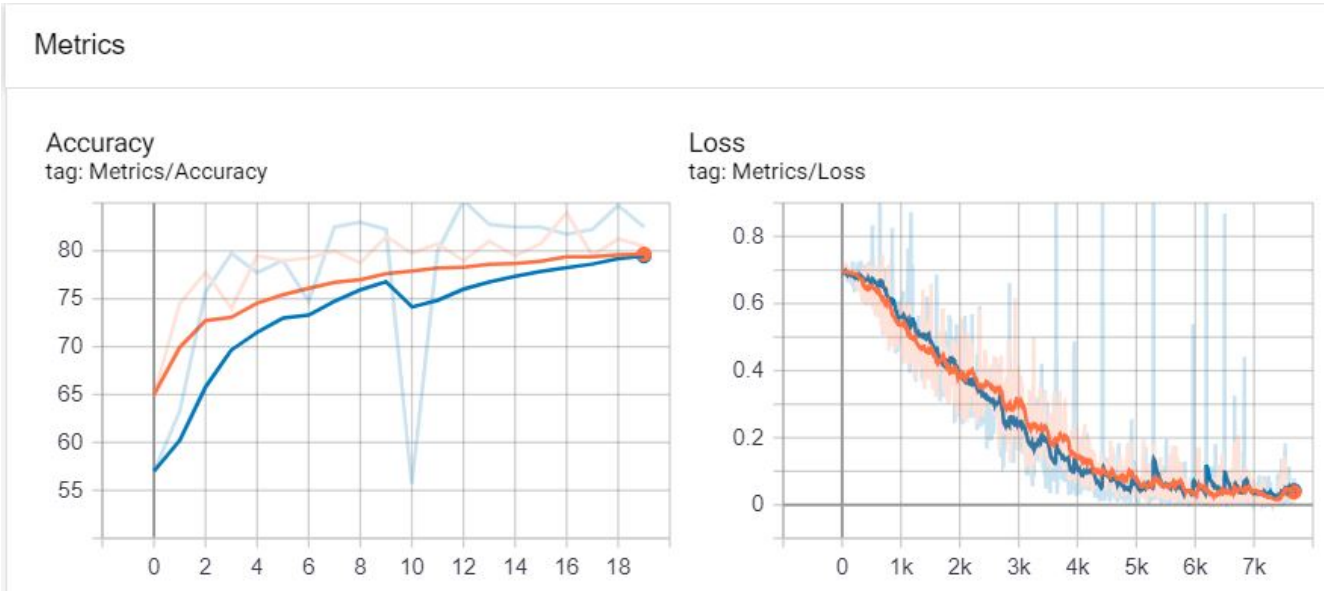
It is generally recommended to use Leaky Relu or plain Relu for layer activation. Leaky Relu is better because it prevents neurons with weak outputs from completely dying out.



Light Blue → Leaky Relu Dark Blue → Relu

Optimizer function

One of the newest ‘discoveries’ in the field of ML is the use of Adam as an optimizer. It is an adaptive algorithm specially catered for use in deep neural networks. The creators of Adam describe it as an amalgamation of SGD(stochastic gradient descent) and RMSProp. We used Adam in our network and a comparison with RMSProp can be seen below.



Adam → Orange RMSProp → Blue

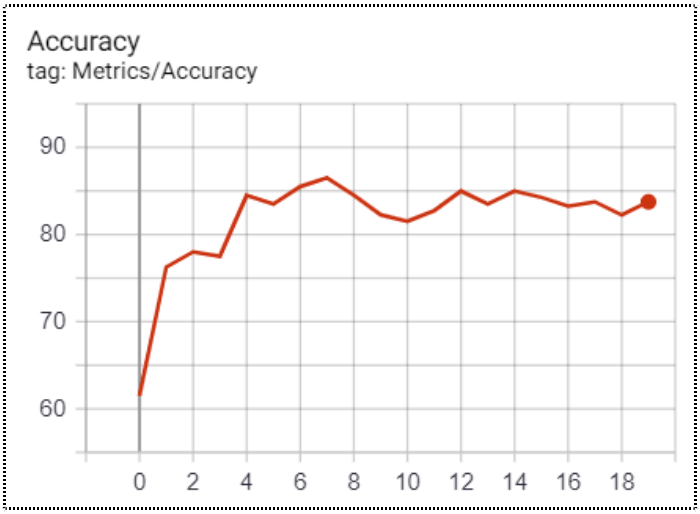
Learning Rate

The learning rate determines the weight change applied by the optimizer. Too low an LR results in being trapped in the error hypersurface's local minimas, while a high one causes oscillations in hypersurface canyons. We found an LR of 0.002 to be suitable.

Red → 0.002LR Orange → 0.001LR Blue → 0.005LR

Results and Analysis

Through the implementation on the above mentioned techniques, successful classification of images of Cats and Dogs was performed.



The above graph represents accuracy vs epoch. Accuracy testing was done on 400 isolated, randomly selected images.

A peak accuracy of 87% is obtained at epoch #7 after which performance slightly deteriorates. Such a graph is useful as it lets us deduce when to end training the network. Post epoch #7, the network shows symptoms of data over-fitting. The network has effectively started memorizing data instead of generalized learning This causes accuracy to decrease.

Over multiple runs the average accuracy hovered around the 88% mark, with some runs showing up to a 92% validation accuracy.

Conclusion and Future Scope

Through this project it can be concluded that Convolutional Neural Networks are an extremely powerful tool that can be used for classification tasks. The existence of powerful frameworks like Pytorch and Tensorflow make it simple for easily applying high level mathematics to daily life problems.

With an average 88% accuracy, we consider our network to be a success. Anything above 80% is considered to be a success in ML circles.

Going forward, we wish to operate on images of higher resolutions. 50*50px resolution is our weakest link at the moment. By shifting to higher resolutions, our accuracy values are bound to increase greatly. We also wish to add a web scraper to automatically collect images and build the dataset without human intervention. More data is the best cure to network over-fitting.