**Machine Learning Engineer Nanodegree**

**Capstone Proposal**

Vinit Kumar  
July 11, 2018

**Proposal**

*(approx. 2-3 pages)*

**Domain Background**

*(approx. 1-2 paragraphs)*

In this section, provide brief details on the background information of the domain from which the project is proposed. Historical information relevant to the project should be included. It should be clear how or why a problem in the domain can or should be solved. Related academic research should be appropriately cited in this section, including why that research is relevant. Additionally, a discussion of your personal motivation for investigating a particular problem in the domain is encouraged but not required.

**Problem Statement**

*(approx. 1 paragraph)*

In this section, clearly describe the problem that is to be solved. The problem described should be well defined and should have at least one relevant potential solution. Additionally, describe the problem thoroughly such that it is clear that the problem is quantifiable (the problem can be expressed in mathematical or logical terms) , measurable (the problem can be measured by some metric and clearly observed), and replicable (the problem can be reproduced and occurs more than once).

The problem here is to use Supervised Learning to train a computer to distinguish between images of dogs and cats. We have a dataset of 25000 images for cats and dogs with labels to accomplish this. Once the computer is trained, we have a test dataset of 12,500 mixed images for which a label (cat or dog) needs to be provided. The success of the project will depend on our accuracy in classifying cat or dog images in the test dataset.

This problem is a classic case of a computer vision challenge. The images are not centered and there are often other objects on the image. Due to variation in scale, rotation and noise between images, my approach would be to train a deep convolution neural network (CNN) implemented using TensorFlow to teach the computer to analyze various aspects of the image. My goal is to start with a simple classic implementation of CNN with several layers of convolution and max pooling. And gradually improve on this basic model by adding dropout, regularization and learning rate decay as well as optimizing hyper parameters and training parameters.

One challenge that I will likely encounter is processing power. I don’t have a GPU computer and I am not sure if my Mac with CPU will have enough processing power. If I do need access to GPU, then I will leverage Amazon Web Services, Elastic Cloud 2 GPU for training the model. This will be new for me and another learning experience. The challenge here is to approach the problem in a computationally efficient manner

**Datasets and Inputs**

*(approx. 2-3 paragraphs)*

In this section, the dataset(s) and/or input(s) being considered for the project should be thoroughly described, such as how they relate to the problem and why they should be used. Information such as how the dataset or input is (was) obtained, and the characteristics of the dataset or input, should be included with relevant references and citations as necessary It should be clear how the dataset(s) or input(s) will be used in the project and whether their use is appropriate given the context of the problem.

For this project we have two datasets:

1. Train dataset of 25,000 images with labels provided for each image
2. Test dataset of 12,500 images. No labels provided. After the model has been trained, the test set will be scored on the best model and the output submitted to Kaggle. Kaggle will provide a log loss estimate on the submission

My plan is to divide the train dataset into 3 different sets – 1.Ttraining dataset of 20,000 images that will be used to train the model, 2. Validation dataset of 4,000 images for testing the performance of the CNN and 3. A test set of 1000 images that will be used to do a final evaluation before submission. The goal of creating this test set is that as the CNN is repeatedly trained and tested on validation set, it will eventually “see” the validation set. This test set will be a more objective evaluation of the accuracy of CNN on an out of sample. I will be using two metrics for measuring the success of the project:

1. Log Loss function – The goal of the convolution optimizer will be to minimum the cross entropy loss which is the same as log loss. The selected model would be the one with the lowest log loss on the validation and test sets. This is also the metric that will be used by Kaggle to rank contestants in this competition. The calculation of log loss is below:

Log loss will be computed on training and validation sets before submission. The Log loss for testing dataset will be provided by Kaggle.

2. Accuracy Score – Defined as % labels correctly classified when comparing model prediction vs actual. This metric will be computed on the training dataset and the validation set which is the population for which we have labels for the images. This is the secondary metric that will be computed and shared but the select of the optimal model will be based on minimizing log loss. It is expected that the model with the lowest log loss will also have one of the best accuracy score.

**Solution Statement**

*(approx. 1 paragraph)*

In this section, clearly describe a solution to the problem. The solution should be applicable to the project domain and appropriate for the dataset(s) or input(s) given. Additionally, describe the solution thoroughly such that it is clear that the solution is quantifiable (the solution can be expressed in mathematical or logical terms) , measurable (the solution can be measured by some metric and clearly observed), and replicable (the solution can be reproduced and occurs more than once).

Given the nature of this problem – images with difference in scale, rotation and noise, I think a deep convoluted neural network is the best algorithm to choose for this problem. Convolutional neural networks are biologically inspired variants of multilayer perceptrons (MLP), designed to emulate the behaviour of a visual cortex. These models mitigate the challenges posed by the MLP architecture by exploiting the strong spatially local correlation present in natural images Also all neurons in a given convolutional layer share the same weights and detect exactly the same thing. Replicating units in this way allows for features to be detected regardless of their position in the visual field, thus constituting the property of translation invariance. Also sharing weights between neurons makes it computationally efficient.

Here is the approach I have in mind:

1. Start with a basic convolution neural network structure – A Convolution Layer with RELU activation function followed by a max pool layer. Multiple such convolution layers will be connected. Finally have one fully connected layer before the output layer. Stochastic gradient descent will be used to optimize this model.

Input Layer -> Conv Layer 1 with RELU -> Max Pool Layer 1 -> Conv Layer 2 with RELU -> Max Pool Layer 2 -> Fully Connected Layer with RELU -> Output Layer

Add regularization to the basic convolution neural network defined in step 1 above to solve for the problem of over fitting. The regularization techniques I will use are dropout and L2 regularization. I would also like to explore learning rate decay to further tune stochastic gradient descent optimization

2. Once a basic convolution neural network is set up, then I will move to the important task of tuning it to the dataset to improve on the results. The following will be done to further tune the basic model:

a. Change in layer structure of the neural network – Adding more convolution and fully connected layers, replacing max pool layer with a convolution layer etc.

b. Tuning the no of neurons in the all the different layers

c. Optimization of regularization techniques – L2 regularization on just fully connected layer vs all layers

d. Selecting the best optimization function – SGD, Adam etc

e. Tuning the learning rate in the optimization function

f. Image augmentation

g. Increasing no of training epochs

h. Using color images instead of grayscale images

**Benchmark Model**

*(approximately 1-2 paragraphs)*

In this section, provide the details for a benchmark model or result that relates to the domain, problem statement, and intended solution. Ideally, the benchmark model or result contextualizes existing methods or known information in the domain and problem given, which could then be objectively compared to the solution. Describe how the benchmark model or result is measurable (can be measured by some metric and clearly observed) with thorough detail.

**Evaluation Metrics**

*(approx. 1-2 paragraphs)*

In this section, propose at least one evaluation metric that can be used to quantify the performance of both the benchmark model and the solution model. The evaluation metric(s) you propose should be appropriate given the context of the data, the problem statement, and the intended solution. Describe how the evaluation metric(s) are derived and provide an example of their mathematical representations (if applicable). Complex evaluation metrics should be clearly defined and quantifiable (can be expressed in mathematical or logical terms).

**Project Design**

*(approx. 1 page)*

In this final section, summarize a theoretical workflow for approaching a solution given the problem. Provide thorough discussion for what strategies you may consider employing, what analysis of the data might be required before being used, or which algorithms will be considered for your implementation. The workflow and discussion that you provide should align with the qualities of the previous sections. Additionally, you are encouraged to include small visualizations, pseudocode, or diagrams to aid in describing the project design, but it is not required. The discussion should clearly outline your intended workflow of the capstone project.

1. Data Visualization: Visual representation of data to find the degree of correlations between predictors and target variable and find out correlated predictors. Additionally, we can see ranges and visible patterns of the predictors and target variable. b.
2. Data Preprocessing: Scaling and Normalization operations on data and splitting the data in training, validation and testing sets.
3. c. Feature Engineering: Finding relevant features, engineer new features using methods like PCA if feasible.
4. d. Model Selection: Experiment with various algorithms to find out the best algorithm for this use case.
5. Model Tuning: Fine tune the selected algorithm to increase performance without overfitting.
6. Testing: Test the model on testing dataset.