Classifying Cat and Dog Audio using CNN architecture

## Domain Background

Classification of sound and noise sources has long involved analysis of waveform and Fourier Transform of sound signals. Techniques like Single Value Decomposition and traditional machine learning methodologies such as SVM (eg. Musical instruments: <https://dl.acm.org/citation.cfm?id=2351793>; Content based classification: <http://www.cbsr.ia.ac.cn/publications/Stan/ACM-03.pdf>) have been widely used to construct decision rules.

In light of advances in deep learning, pattern recognition has been made easier with the application of convolutional neural network which greatly enhances the spatial invariance of models compared to traditional methods. By adopting a similar approach to that of image recognition, CNN is expected to be able to extract patterns (harmonics) out of audio signals and classify sources.

One interesting application of this is classification of dog and cat sounds. To human audience, the two sound sources have distinct difference (tone, length etc.) which are not necessarily easy for traditional techniques such as SVM to handle (spatial/sequential invariance etc.). However, it is expected that a CNN architecture with appropriate depth can provide a model that is more accurate.

## Problem Statement

The project aims to create a light weight classifier for dog barks and cat “meows” that is capable of achieving reasonable accuracy (>80%) by applying neural network on sound data.

## Datasets and Inputs

The data used in this project is “Audio Cats and Dogs” dataset from Kaggle: (<https://www.kaggle.com/mmoreaux/audio-cats-and-dogs/data>)

The data specs are as follow:

* Length: 2 to 10+ seconds per clip
* Train/test split: recommended sets of train and test sound clips are specified
* Number of clips:
  + Train: 64 (dog) and 115 (cat)
  + Test: 49 (dog) and cat (cat)
* Total clip length:
  + Train: 942s (cat) and 317s (dog)
  + Test: 382s (cat) and 281s (dog)
* Format: WAV

## Solution Statement

The proposed solution is that of a convolutional neural network which takes the Fourier Transformed wav data as input and outputs a binary value to denote whether the sample is dog or cat. The CNN will consist of three convolutional layers each with its own activation. Apart from the main section where convolutional layers are chained, each layer will have its own fully connected layer to output an array which will be concatenated at the end of the convolutional chain for another dense connection to compute the output. Dropouts and BatchNorm will be used for regulation.

The architecture is inspired by the skip layers in ResNet : <https://arxiv.org/abs/1512.03385>

## Benchmark Model

To show the suitability of neural networks in sound recognition compared to traditional machine learning methods, an SVM model trained on the same dataset will be used to benchmark the performance of the CNN built.

## Evaluation Metrics

The specified validation set (49 dog clips and 49 cat clips) will be used for benchmark the model. Since the number of clips are balanced, a simple accuracy score would be appropriate for the binary classification problem.

## Project Design

The expected workflow of the project is as follows:

1. Collect data from Kaggle
2. Prepare tools for preprocessing of data which includes:
   1. Reading wav data into time series using scipy.io.wavfile
   2. Concatenating all wav data into one dataset separately for training and testing, cats and dogs
   3. Generate spectrogram data (harmonics) using consecutive Fourier Transform (window=('tukey', 0.25), available again in scipy
   4. Create tool to process any wav file of arbitrary length into segments of spectrogram data of fix window size for batch training and evaluation
3. Perform explorative data analysis explore the differences in harmonics of cat and dog sounds
4. Make a train (and validation) data generator for training which randomly generates audio data of fix window size from the concatenated data
5. Prepare SVM model for benchmarking using data generator
6. Design appropriate CNN architecture with tunable hyperparameters
7. Use grid search method to select the best model using data generator
8. Train CNN model with selected parameters
9. Create mechanism to save, read model and pass new, unseen wav recoding to model for classification
10. Compare model with benchmark model