# Cats and Dogs

*Convolutional Neural Network Image Classification*

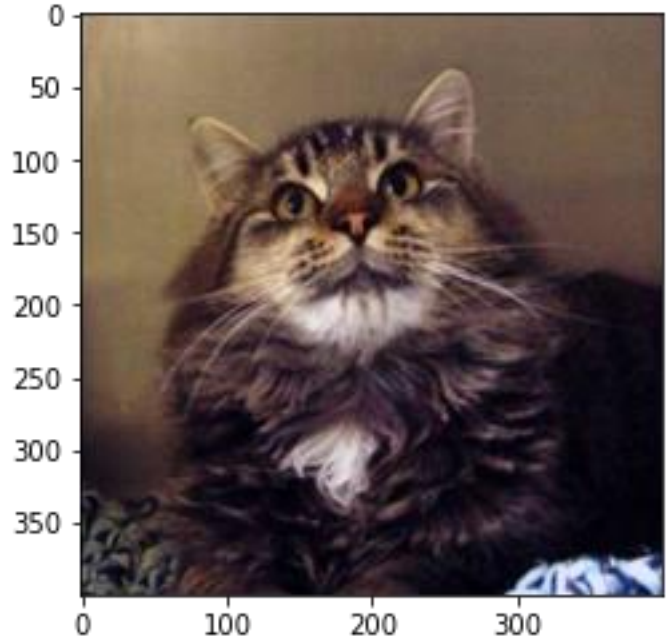
## Main Objective of Analysis

The analysis is centered around the binary classification of cat and dog images utilizing the convolutional neural network (CNN). Unlike dense neural networks, CNN layers act as filters of varying size and depth to attenuate the number of parameters and reduce the total computation time. This is ideal for classifying images, which have the dimensions of length, width, and channel. While the CNN models in this current analysis differentiates between images of cats and dogs, with the right architecture, fine tuning, and robust data, a CNN model architecture that performs well with cats and dogs can also work with the binary classification of other kinds of images. The high-level deep learning library Keras was used for this analysis.

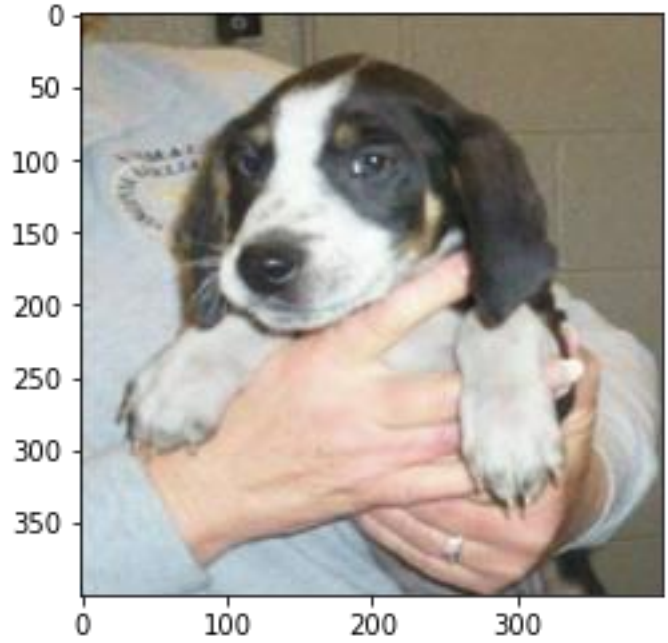
## Description of Dataset

The dataset is from a collection of images of cats and dogs in JPEG format. The train set includes 4000 images of cats and 4000 images of dogs. The validation set includes 1000 images of cats and 1000 images of dogs. The following are sample images from the dataset:

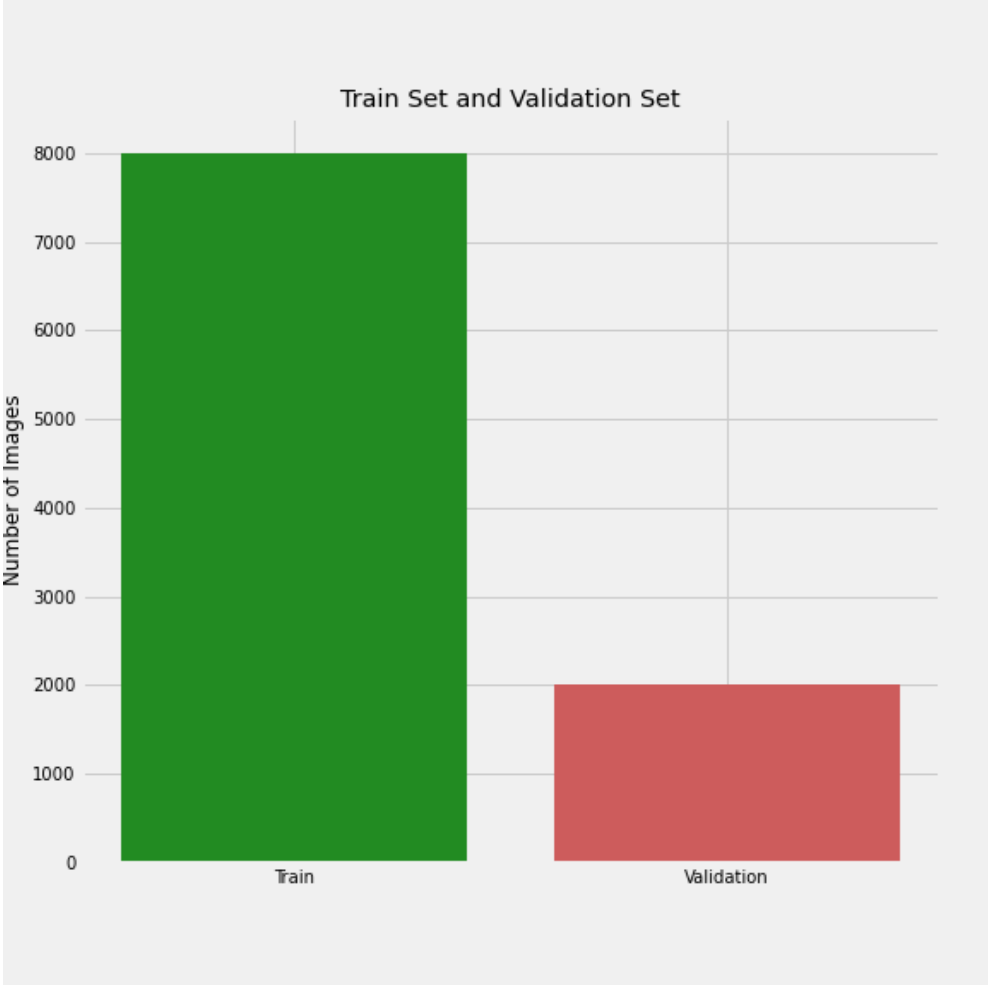
```
Image label: 0
This is a cat
```




```
Image label: 1
This is a dog
```



For the size of the dataset, the proportion of train set versus validation set seems appropriate, with the validation set 20% the total number of images.




**Data Preprocessing**

For both the train and validation sets, the images were converted to arrays and then resized to 100x100 in RGB format with three color channels. The cat images were labeled with 0 while the dog images were labeled with 1. The shape of the train set was `(8000, 2)` and the shape of the validation set was `(2000, 2)`. Then, the arrays for the images were split from the labels and the shapes were as follows: `x_train (8000, 100, 100, 3)`, `y_train (8000,)`, `x_test (2000, 100, 100, 3)`, `y_test (2000,)`. The arrays were reshaped in this way because such dimensions are required as input for a Keras model. Then, `y_train` and `y_test` were converted into one hot encoded categories.

**Convolutional Neural Network Models**

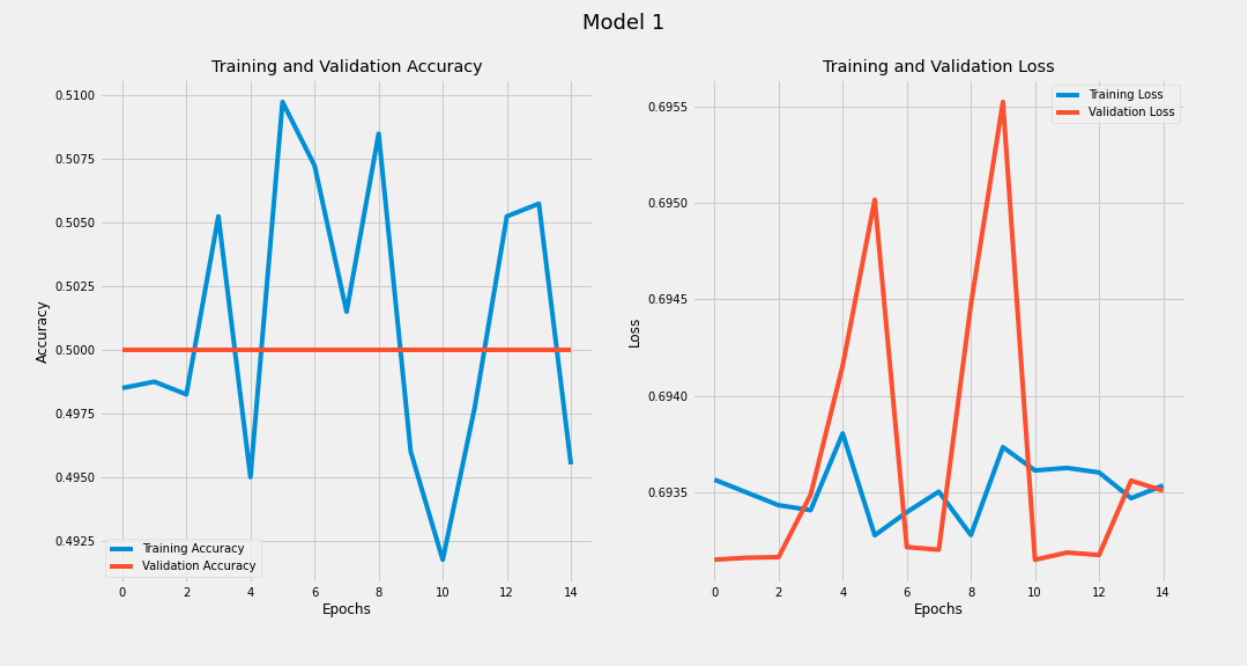
Three CNN models were created with Model 1 being relatively shallow and Model 2 and Model 3 deeper and based on the same architecture with different tuning. They were all trained for 15 epochs and while accuracy and loss scores varied during the course of training, Model 1 and Model 2 had similar metrics in the end while the scores for Model 3 differed considerably.

Model 1

The training data was first inputted into a convolutional layer with a depth of 32 and a kernel size of (5,5) and a stride of (2,2), and then the output was passed into a ReLU activation layer. This was followed by a second convolutional layer identical to the first one, and then another ReLU activation layer. Then, max pooling with a pool size of (2,2) and dropout of 0.25. Finally, the output was flattened and passed to a dense layer of 512, a ReLU activation layer, a dropout layer of 0.5, another dense layer of 2, the number of classes, and then finally to a softmax activation layer. Categorical crossentropy was used as the loss function and Adam with a learning rate of 0.01 was used as the optimizer. The following is a summary of Model 1:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 50, 50, 32)        2432
_________________________________________________________________
activation (Activation)      (None, 50, 50, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 23, 23, 32)        25632
_________________________________________________________________
activation_1 (Activation)    (None, 23, 23, 32)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 11, 11, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 11, 11, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 3872)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1982976
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
_________________________________________________________________
activation_3 (Activation)    (None, 2)                 0
=================================================================
Total params: 2,012,066
Trainable params: 2,012,066
Non-trainable params: 0
_________________________________________________________________
```

Line graphs for Model 1 comparing accuracy and loss for train and validation sets over 15 epochs:



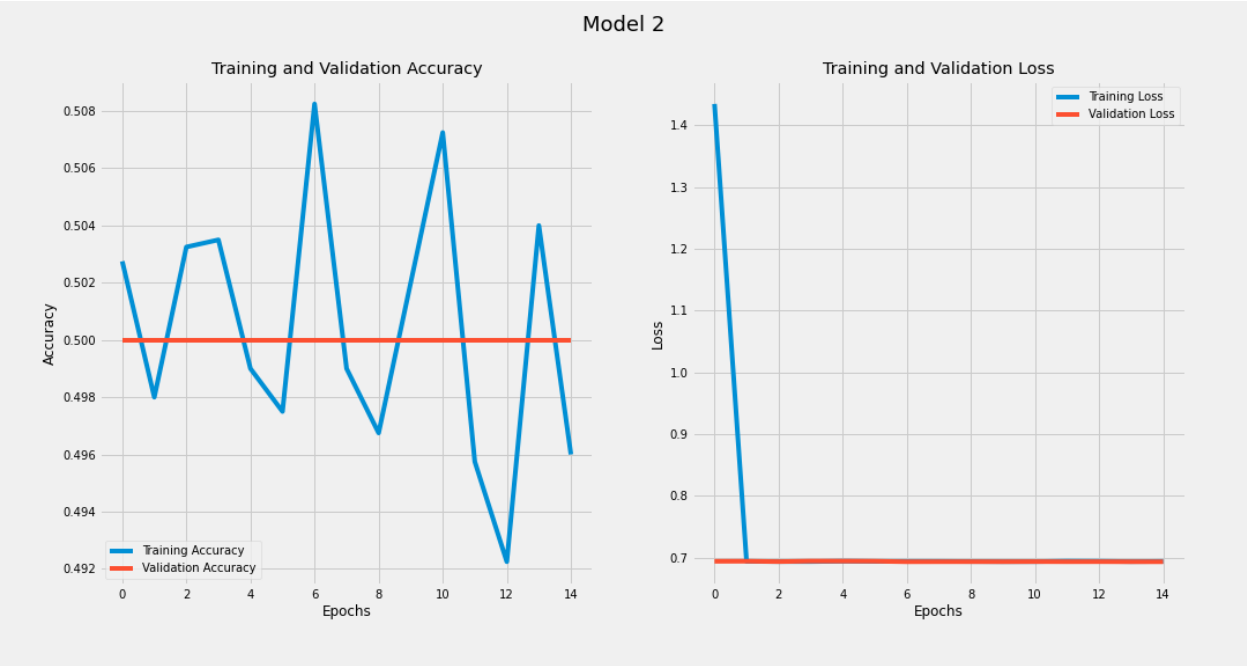

## Model 2

The training data was first inputted into a convolutional layer with a depth of 32 and a kernel size of (3,3) and a stride of (1,1), and then the output was passed into a ReLU activation layer. This was followed by a second convolutional layer identical to the first one, and then another ReLU activation layer. Then, max pooling with a pool size of (2,2) and dropout of 0.25. The output was then passed into a convolutional layer with a depth of 64 and a kernel size of (3,3) and then into a ReLU activation layer. Next, the output was passed into an identical convolutional layer and a ReLU activation layer as previously along with max pooling with a pool size of (2,2) and dropout of 0.25. Finally, the data was flattened and passed to a dense layer of 512, a ReLU activation layer, a dropout layer of 0.5, another dense layer of 2, the number of classes, and then finally to a softmax activation layer. Categorical crossentropy was used as the loss function and Adam with a learning rate of 0.01 was used as the optimizer. The following is a summary of Model 2:

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 100, 100, 32)      896
_________________________________________________________________
activation_28 (Activation)   (None, 100, 100, 32)      0
```

```
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 98, 98, 32)        9248
_________________________________________________________________
activation_29 (Activation)   (None, 98, 98, 32)        0
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 49, 49, 32)        0
_________________________________________________________________
dropout_14 (Dropout)         (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 49, 49, 64)        18496
_________________________________________________________________
activation_30 (Activation)   (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 47, 47, 64)        36928
_________________________________________________________________
activation_31 (Activation)   (None, 47, 47, 64)        0
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 23, 23, 64)        0
_________________________________________________________________
dropout_15 (Dropout)         (None, 23, 23, 64)        0
_________________________________________________________________
flatten_5 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_10 (Dense)             (None, 512)               17334784
_________________________________________________________________
activation_32 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_16 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 1026
_________________________________________________________________
activation_33 (Activation)   (None, 2)                 0
=================================================================
Total params: 17,401,378
Trainable params: 17,401,378
Non-trainable params: 0
_________________________________________________________________
```

Line graphs for Model 2 comparing accuracy and loss for train and validation sets over 15 epochs:



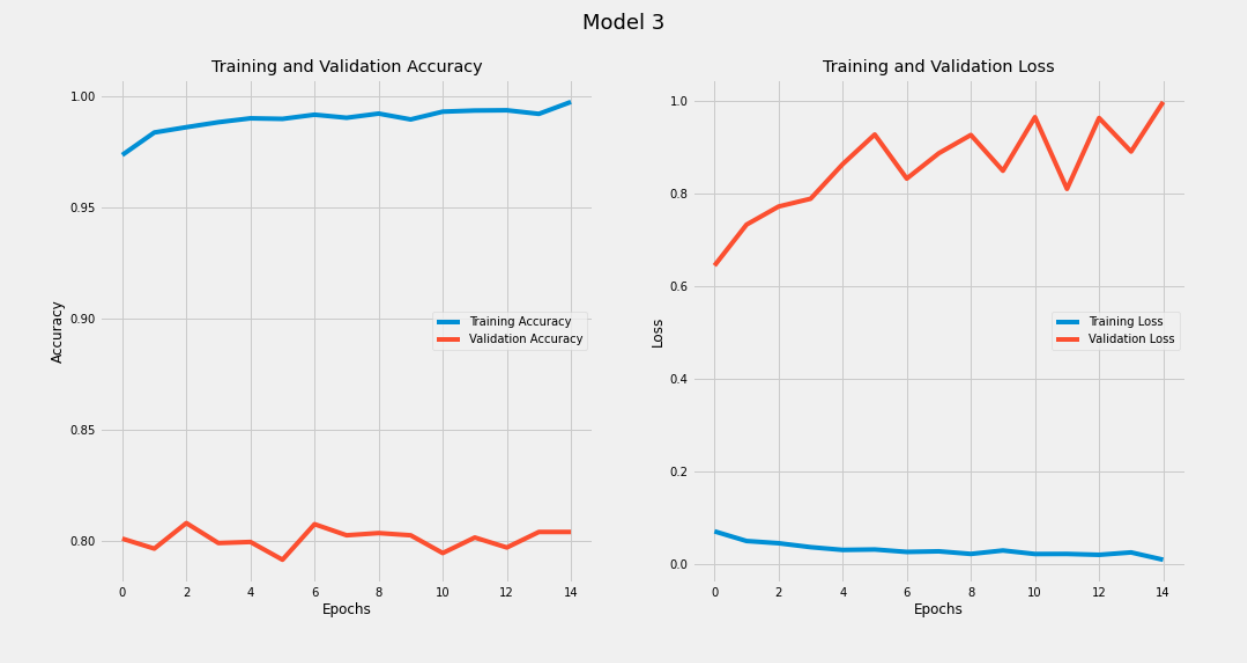

## Model 3

The model architecture was based on Model 2 with a couple of adjustments in tuning. Rather than using kernels with size (3,3), kernels with size (5,5) were used. Also, in the compiling step, Adam was used as the optimizer as well, but the learning rate was changed from 0.01 to 0.0001. The following is a summary of Model 3:

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 100, 100, 32)      2432
_________________________________________________________________
activation_22 (Activation)   (None, 100, 100, 32)      0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 96, 96, 32)        25632
_________________________________________________________________
activation_23 (Activation)   (None, 96, 96, 32)        0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 48, 48, 32)        0
_________________________________________________________________
dropout_11 (Dropout)         (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 48, 48, 64)        51264
_________________________________________________________________
```

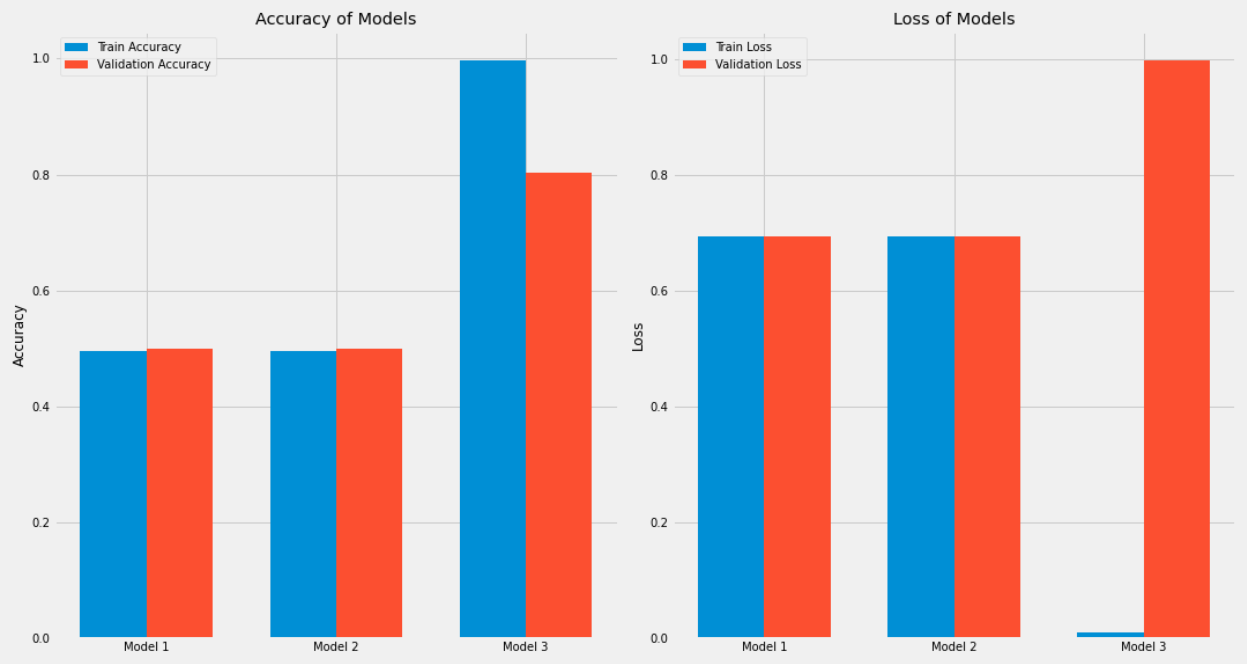
```
activation_24 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)            (None, 44, 44, 64)        102464
_________________________________________________________________
activation_25 (Activation)    (None, 44, 44, 64)        0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2  (None, 22, 22, 64)        0
_________________________________________________________________
dropout_12 (Dropout)          (None, 22, 22, 64)        0
_________________________________________________________________
flatten_4 (Flatten)           (None, 30976)             0
_________________________________________________________________
dense_8 (Dense)               (None, 512)               15860224
_________________________________________________________________
activation_26 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_13 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_9 (Dense)               (None, 2)                 1026
_________________________________________________________________
activation_27 (Activation)    (None, 2)                 0
=================================================================
Total params: 16,043,042
Trainable params: 16,043,042
Non-trainable params: 0
_________________________________________________________________
```

Line graphs for Model 3 comparing accuracy and loss for train and validation sets over 15 epochs:



A summary table of the train accuracy, validation accuracy, train loss, and validation loss for the models after 15 epochs as well as bar charts visually comparing the results:

| Model | Model 1 | Model 2 | Model 3 |
|---|---|---|---|
| **Train Accuracy** | 0.495500 | 0.496000 | 0.997500 |
| **Validation Accuracy** | 0.500000 | 0.500000 | 0.804000 |
| **Train Loss** | 0.693533 | 0.693719 | 0.010391 |
| **Validation Loss** | 0.693505 | 0.693182 | 0.997631 |




## Key Findings and Insights

The final scores for train accuracy, validation accuracy, train loss, and validation loss after 15 epochs present interesting observations. Model 1 was a more shallow network than Model 2, but the end results were nearly identical. The train accuracy fluctuated greatly for both, but the validation accuracy remained constant, around 0.50, which means the models would perform no better than chance when classifying images of cats and dogs. The behavior for loss was different in Model 1 and Model 2 over the course of 15 epochs. The validation loss for Model 1 had very big fluctuations over training, and the train loss was relatively more stable. For Model 2, the validation loss was constant at around 0.69, and the train loss started high and quickly dropped to around 0.69. In contrast, Model 3 had significantly higher train and validation accuracy scores with a validation accuracy around 0.80 and was seemingly the better model

compared to Model 1 and Model 2. Model 2 and Model 3 had nearly identical architectures, but were tuned differently and perhaps this is an explanation for Model 3 to perform much better in accuracy. The learning rate for the Adam optimizer was significantly smaller in Model 3. Also, the kernel size was larger in Model 3, which can account for the decreased number of parameters. However, Model 3 had high validation loss, which continued to increase with more training. In addition, because the training accuracy was considerably higher than the validation accuracy, overfitting in Model 3 was a problematic issue, perhaps due to the smaller learning rate of the optimizer. As far as choosing the best model out of the three, because Model 3 had the highest validation accuracy score and serves as a possible candidate, but due to the large amount of overfitting, it may not generalize well to new images, so improvements must be made.

**Next Steps**

Out of the three CNN models tested, because of the superior accuracy scores, Model 3 should be further pursued and improved upon. This would most likely entail fine tuning the model and perhaps adding additional layers in the network. The smaller learning rate in the Adam optimizer seemed to increase the accuracy scores, but overfitting is apparent and experimenting with larger learning rates may prove to be beneficial. Other types of optimizers such as RMSprop and stochastic gradient descent can be used as well. The image data could be reshaped into a different size, and as far as the data goes, a larger dataset could be used to expand the train set and in addition, the images could be augmented so the model can adjust for factors such as orientation and distortion.