**Research Project Report**

**Classifying the coat of cats using both categorical features and visual features**

Devon Hope – 101038344

Yichen Dou – 101003897

Prof. Alan Tsang

COMP 4107 B, Winter 2021

Apr 23, 2021

**1 Introduction**

**1.1 Research Question**

How neural networks help predict the coat of cats using both categorical features and visual features.

**1.2 Background**

Classification problems are a popular topic in areas of machine learning nowadays. The neural network is feed with several images and outputs a specific label or labels for classification. However, what if we have extra information about the labels? For example, if we want to predict the price of a house based on an image and numerical values such as the number of rooms, levels, and location, there is a problem that numerical, categorical, and image features are trained well on different neural network architectures respectively. In this research, we will make a neural network with two input layers, one is for categorical features, and another is for images when predicting the coat of cats.

**1.3 Datasets**

Cat Breeds Dataset from https://www.kaggle.com/ma7555/cat-breeds-dataset

It is a dataset which contains a csv data and a bunch of images. The csv data includes categorical features such as age, gender, breed, coat and size of a cat, where each instance belongs to a single cat with related images.

**1.4 Machine Learning Libraries Used In The Research**

Tensorflow: A well-known open-source library for machine learning.

Keras: An open-source library that provides extra features to tensorflow. It offers an interface user can access to make artificial neural networks with many toolkits.

Scikit-learn: A free machine learning library that offers many algorithms in data analysis.

**2 Methodology**

**2.1 Data Analysis**

All data formatting can be found in ‘Analysis.ipynb’.

**2.1.1 Missing Values**

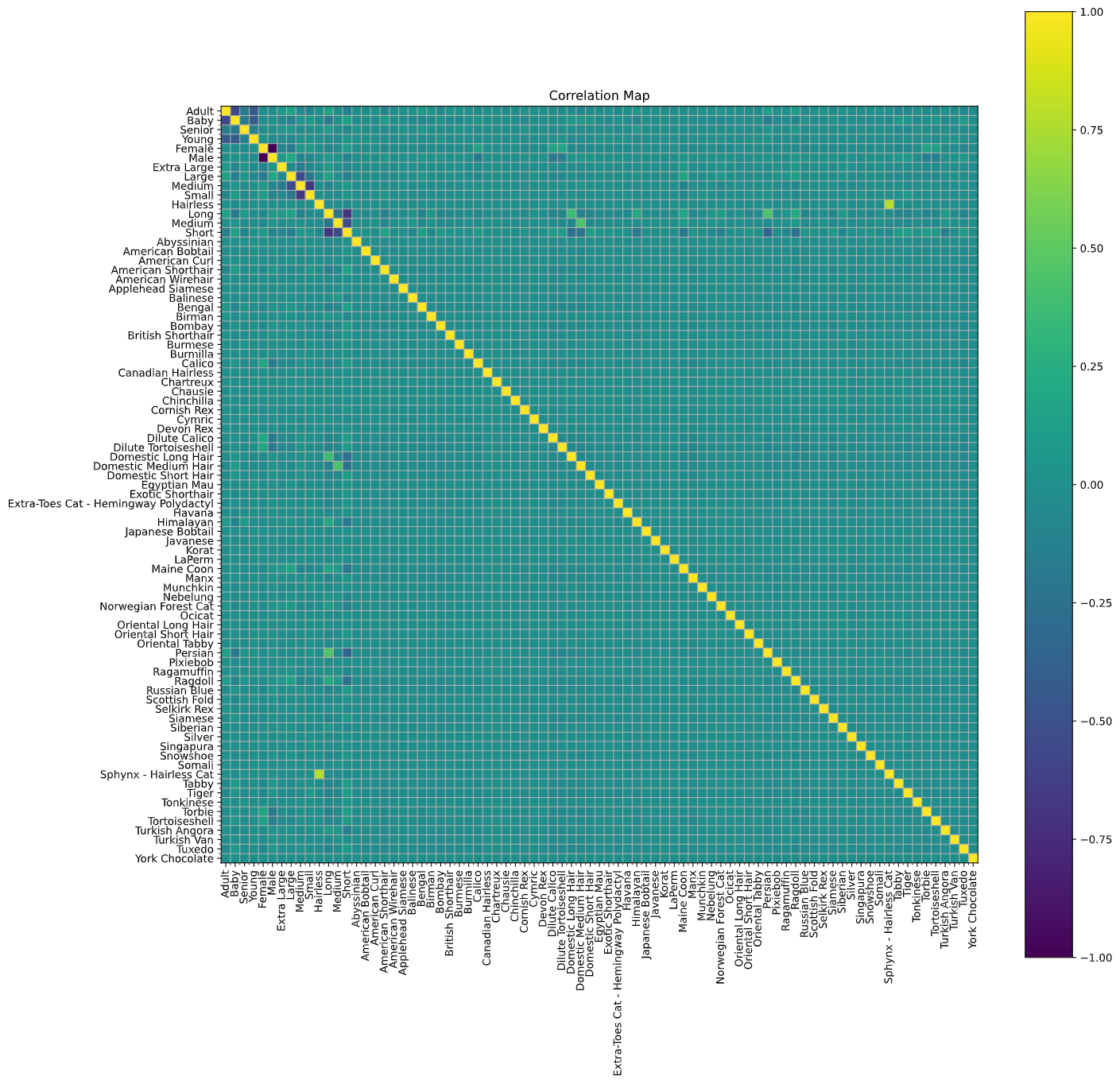
Several rows of cats were missing breeds, and the number of images per breed was severely uneven. The number of images per class type is important, if the number is uneven, or misrepresented, the produced results could be skewed and the accuracy of the results would be false.

**2.1.2 Correlation**

We used correlation graphs to determine which classes (i.e., coat, breed, age and gender) shared cats and for what reasons. It is commonly found that size and coat are related as well as gender and size, this makes sense on the surface as the larger a cat will appear has direct correlation with the size of its coat, as well as its gender.

**2.1.3 Uneven Categories**

Some classes contained very few categories, such as coat, where the total number of coat types was four, in most cases however, there were only three coats, thus the number of cats was limited to three types of coats. The same can be said with breeds, some breeds contained fewer than 500 image samples, our baseline for accurate testing, therefore they needed to be cut from the sample size, limiting our input again.



**2.2 Data Preprocessing**

**2.2.1 Image Resizing**

When we load images from paths stored in the data file, we need to make sure all images should have the same dimension. Moreover, for a neural network model can successfully predict the coat of cats, the network should be present enough detail of images, such that patterns like hair edges can be detected. After many trials on a standard convolutional neural network, we found that resolutions above 128 pixels of width and 128 pixels of height give us an optimal result at an acceptable loss.

**2.2.2 Remove Instances That Contain Null Values**

Since the features we will train for the neural network are all categorical features, unlike numerical features, we may not fill empty spaces with mean values of a feature in this dataset. It is better for us to drop instances that contain null values in this case.

**2.2.3 Remove Labels That Contain Few Instances**

In our data analysis, we investigated several instances in each category of the coat. Then we determined that there are only 60 instances in hairless cats, comparing to more than thousands of instances in other coat categories. With a lack of data, the neural network may not learn its associated features well, so instances about hairless cats are removed.

**2.2.4 One-hot Encoding**

Our labels were represented as text, but the neural network needs them to be encoded before it can understand. Since our research problem is about making a single-label and multi-class classification network, we use the one-hot encoding as our technique to encode the text of coats.

**2.3 Neural Network Model**

Model diagrams can be found in the Models folder under ‘final\_model\_1’ and ‘final\_model\_2’ for the complex structures.

**2.3.1 Basic Model Structure**

The first model that was implemented was the basic CNN structure using the RMSprop optimizer from Keras. This model produced inaccuracies and high loss values, but it also showed that the data was being skewed in our initial testing of the breed and coat classes. The preprocessing that had been done was for the total number of breeds and coat types, unfortunately we had to cut some of these breeds and coats as the number of images for specific categories in each class was less than desirable, leaving us with 26 breeds and 3 coats.

**2.3.2 Complex CNN Structure**

The complex deep learning CNN was made with dual input, from two separate models that are combined together. The first defines the density of each layer, and the second model defines the type of convolution with a series of convolutional layers, maxpooling layers, a flatten layer and a dropout layer. The dual input CNN produced the best accuracies and losses, nearing 70% accuracy in most tests, and losses below 0.7. There are two versions of this structure, the main difference is the optimizer, the first, model\_1, uses the standard gradient descent optimizer, or SGD, and the second uses the Adam optimizer. The first model measures loss with binary cross entropy and the second with categorical cross entropy, the difference is notable in the tuning that is needed to stop overfitting with the learning rate and decay for Adam optimizer.

**2.4 Training, Testing and Validation**

**2.4.1 Early Testing**

Initial testing began with the standard CNN at 20 epochs with a learning rate between 0.01 and 0.05 and the RMSprop optimizer. The results showed inconsistencies in our data formatting methods as well as the dataset itself. This was useful for developing the final combined neural network structure, if these errors in our data had not been found, the final network would not train as well, and the losses would be much higher. As you can see below, the first attempt at training with our standard CNN shows heavy overfitting.

****

Figure /Standard CNN

**2.4.2 Testing**

Testing took place with the final combined CNN, the more complex one, over several iterations of adjusting dropouts and layers to get the CNN to train properly. Testing the CNN encapsulates training and fitting properly for our desired accuracy and loss values. When testing the complex CNN’s, we encountered a few issues, overfitting, low accuracies or stagnant accuracies and high loss values. Over several iterations of the complex CNN model, we determined that testing the single input version of the complex CNN was not fitting or training properly with the breed class alone. When switching to the coat class, determining a cat by their coat value, we obtained more accurate results, but with fewer categories in this class, three, training the model would only be very inaccurate for real world use. This is where the idea of the dual input layer complex CNN came from, by training the network with more than one class type, we could identify a cat, their breed, age, gender and coat with more precision.

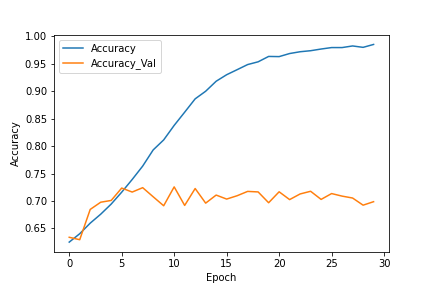


Figure /Single Input Complex CNN with coat class & untuned

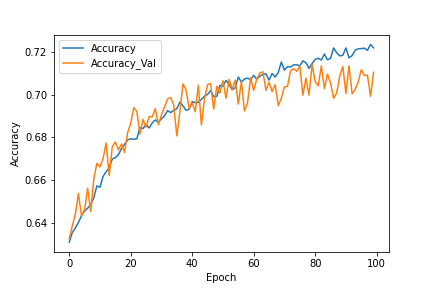
****

Figure / Single Input Complex CNN with Coat Class & tuning

**2.4.3 Training**

Training model\_1 of the complex dual input CNN’s resulted low but stagnant accuracies and low losses, it was clear that more tuning was needed to train the network properly. When training model\_2, of the complex dual input CNN’s, the accuracies were steadily increasing but would stagnate after they reached 70%. This showed us that the learning rate could be adjusted further and the dropout values of both CNN’s were too high. The follow up can be found in the next section, hyperparameter tuning. The range of epochs for training was limited by our combined compute power of our GPU’s, from 10, at testing, to 100 for training. Each step would approximately take up to a minute depending on the batch size and resolution of our formatted images.

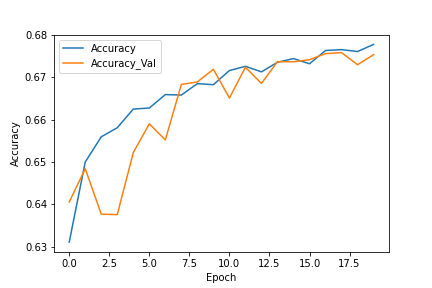


Figure / Model\_1 after 20 epochs

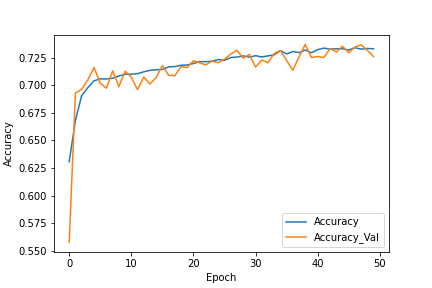
****

Figure / Model\_2 after 50 epochs

**2.5 Hyperparameter Tuning**

@TODO Testing different set-up for the net, including number of neurons, number of layers, learning rate, activation function etc.

**3 Results**

* Model\_2 after 50 epochs with a learning rate of 0.004

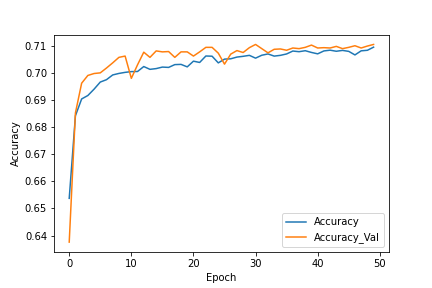


Figure / Model\_2 Accuracy

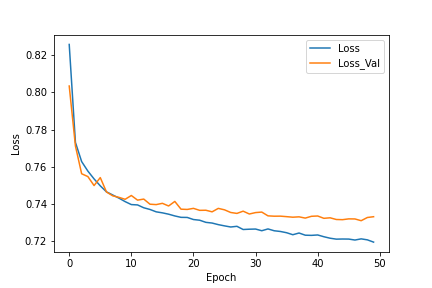


Figure / Model\_2 Loss

* Model\_2 after 50 epochs with a learning rate of 0.015

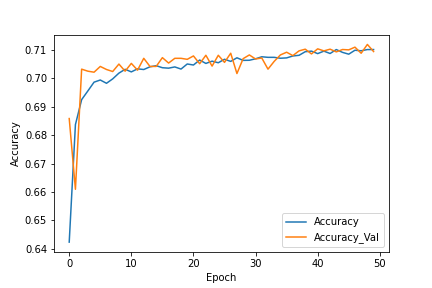


Figure / Model\_2 Accuracy

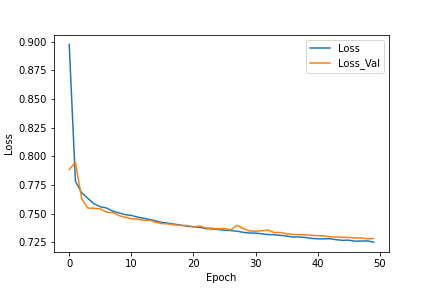


Figure / Model\_2 Loss

* Model\_2 after 50 epochs with learning rate of 0.005

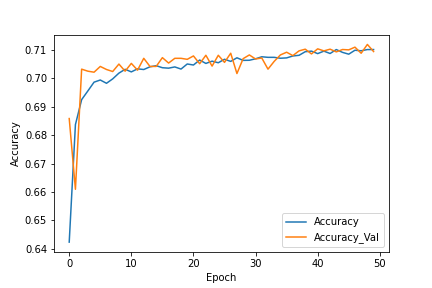


Figure / Model\_2 Accuracy

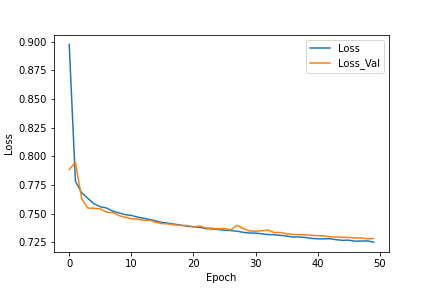


Figure / Model\_2 Loss

**4 Discussion**

@TODO Read from 3

**5 Conclusion**

@TODO Conclusion

**References**

APA Format