# Abstract

Machine Learning and Deep Learning Algorithms were investigated in terms of their ability to perform a supervised binary image classification task involving the Kaggle Dogs vs Cats dataset. Machine Learning algorithms struggled to achieve above 60% training accuracy. Though the CNNs tended to overfit, the inclusion of regularisation via dropouts reduced this effect and the optimal deep learning algorithm developed using Convolutional Neural Networks achieved a training accuracy of 96% and a validation accuracy using unlabelled images of 94%.

In a straight comparison the optimal CNN model had an AUC of 94% compared to 51% for kNN and 58% for Naive Bayes when tested using unseen data.

# Acknowledgements

I would like to thank my supervisor Abhishek Kaushik for his support and advice, particularly for stressing the importance of viewing the project as an opportunity to learn. I would also like to thank the module leader Dr Shazia A Afzal for her support, direction, and encouragement.

I would also like to thank the lecturers and staff at DBS for their insights, encouragement, advice,

and kind support as I was progressing along this educational journey.

Finally, I would like to thank my fellow students within the module for their engagement, friendship, and support.

# Contents

[Abstract 1](#_bookmark0)

[Acknowledgements. 2](#_bookmark1)

[Contents. 3](#_bookmark2)

1. [Introduction 5](#_bookmark3)
   1. [Aims of the project 5](#_bookmark3)
   2. [Scope of the project 5](#_bookmark3)
   3. [Approach Taken 5](#_bookmark3)
2. [Background 6](#_bookmark4)
   1. [Parametric vs. Non-Parametric Models. 6](#_bookmark4)
   2. [Linear vs. non-linear. 6](#_bookmark4)
   3. [Bias vs. Variance. 7](#_bookmark5)
   4. [Models Developed 7](#_bookmark5)
      1. [Naive Bayes 7](#_bookmark5)
      2. [Support Vector Machines 7](#_bookmark5)
      3. [kNN 8](#_bookmark6)
      4. [CNN 8](#_bookmark6)
   5. [Benefits of automated image classification. 10](#_bookmark7)
3. [Requirements Specification and Design 10](#_bookmark7)
   1. [Project / Business requirements 10](#_bookmark7)
   2. [Information Requirements 10](#_bookmark7)
   3. [Tools and Techniques. 10](#_bookmark7)
   4. [Top-level Design 11](#_bookmark8)
      1. [Image data 11](#_bookmark8)
      2. [Image processing 11](#_bookmark8)
4. [Implementation 11](#_bookmark8)
   1. [Naive Bayes 11](#_bookmark8)
   2. [Support Vector Machines 11](#_bookmark8)
   3. [kNN 11](#_bookmark8)
   4. [CNN 11](#_bookmark8)
5. [Testing and Results 12](#_bookmark9)
   1. [Naive Bayes 12](#_bookmark9)
      1. [Greyscale Dataset 12](#_bookmark9)
      2. [Edge Features Dataset 14](#_bookmark10)
      3. [Thresholded Image Features Dataset 16](#_bookmark11)
      4. [Combined Features Dataset 17](#_bookmark12)
   2. [kNN 20](#_bookmark13)
      1. [Greyscale Dataset 20](#_bookmark13)
      2. [Edge Features Dataset 21](#_bookmark14)
      3. [Thresholded Features Dataset 21](#_bookmark14)
      4. [Combined Features Dataset 22](#_bookmark15)
   3. [SVM 24](#_bookmark16)
      1. [Greyscale Dataset 24](#_bookmark16)
      2. [Edge Features Dataset 25](#_bookmark17)
      3. [Thresholded Features Dataset 25](#_bookmark17)
      4. [Thresholded Image Features Dataset 26](#_bookmark18)
   4. [CNN 26](#_bookmark18)
   5. [Comparison data. 29](#_bookmark19)
      1. [Greyscale comparison 30](#_bookmark20)

[5.5.1 Edge Features comparison 31](#_bookmark21)

* 1. [Discussion 31](#_bookmark21)
     1. [Datasets 31](#_bookmark21)
     2. [Models 32](#_bookmark22)

1. [Conclusions and Future work 32](#_bookmark22)
2. [References / Bibliography 33](#_bookmark23)

# Introduction

Image recognition is a process that human beings take for granted and one where they generally achieve high accuracy quickly and efficiently. Until relatively recently computer based approaches, largely involving machine learning (ML), were seen as inadequate by comparison.

There are numerous issues that cause problems for computer based image recognition, as all objects of the same class will not look the same, be the same size, be viewed under the same illumination or from the same angle, or be fully visible (Rafay, 2020).

Websites often use CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) image recognition tests to prevent malicious automated brute force attacks. A CAPTCHA is a recognition task believed to be trivial for humans but difficult for machines. One such 12 image cat and dog CAPTCHA dataset called ASIRRA (Animal Species Image Recognition for Restricting Access) was investigated in 2007. The researchers, (Elson, Douceur, Howell and Saul, 2007) felt that “barring a major advance in machine vision, we expect computers will have no better than a 1/54,000 chance of solving it.”

Since 2007, ML approaches to image recognition have been overtaken by deep learning models involving Convolutional Neural Networks (CNN) based on non-linear or piecewise linear approaches.

This applied project involves investigating the development and implementation of algorithms and models to provide binary classification of images of cats and dogs using both ML and deep learning models.

## Aims of the project

The main aim of the project is to apply machine learning and deep learning techniques to develop and compare supervised binary classification models tasked with successfully classifying unseen photos of cats and dogs.

## Scope of the project

The project will be achieved by investigating the use of machine learning and deep learning algorithm techniques, including Naive Bayes, Support Vector Machines (SVM), kNN and Convolutional Neural Networks.

## Approach Taken

Python code based on Pillow, Scikit Learn, and Tensorflow 2 was used to pre-process and load the images and then develop, train, test, and validate a range of binary classifiers.

A kaggle dataset of photos of cats and dogs was used (Dogs vs. Cats | Kaggle, 2013). Four different feature sets were prepared:

* + - image greyscale values as features
    - image edge features extracted using Canny algorithm
    - thresholded representation of images using Otsu’s thresholding
    - combination of image texture features
      * hu moments
        + a set of 7 numbers that provide shape matching
      * haralick features
        + analysis of textural features of an image
      * histogram of oriented gradients
        + intensity gradients and edge directions as a means of object classification

Four different supervised binary classification algorithms were used to implement the models:

SVM, kNN, Naive Bayes, and CNN.

Once trained the models were validated using a set of unseen images.

# Background

## Parametric vs. Non-Parametric Models

Machine learning algorithms are based on mathematical approaches that seek to use training data to identify a function (f) that maps input variables (X) to output variables (Y), such that Y = f(X) (Brownlee, 2020a).

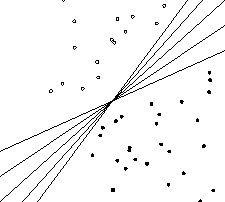
Parametric algorithms, such as Naive Bayes, use a fixed set of parameters to develop the algorithm. Parametric algorithms are easy to understand and interpret, and train quickly. However they are not suited to complex systems and are limited by the form of the specified function.

Non-parametric algorithms such as kNN, decision trees, and SVM, identify the mapping relationship from the training data. The number of nearest neighbours, nodes in a decision tree or support vectors in SVM can be varied. Non-parametric algorithms are flexible as they don’t rely on prior knowledge, but require more data, are slow to train, and prone to overfitting.

Parametric vs. non-parametric approaches can be seen as a trade-off between computational cost and accuracy.

## Linear vs. non-linear

Where binary classification is involved a linear classifier can decide class membership by comparing a linear combination of the features to a threshold. In a system with multiple dimensions the linear classifier is a hyperplane and if it perfectly separates the two classes (Manning, Raghavan, and Schütze, 2008).



**Fig 2.1 There is an infinite number of hyperplanes that separate two linearly separable classes. From Manning, Raghavan, and Schütze, 2008.**

If the true relationship between features/variables is non-linear then linear models will not be particularly effective.

CNNs have been a primary image classification algorithm for almost 2 decades and use a deep learning approach that explicitly assumes that images are the input being processed (Torres, 2018). A principal advantage of CNNs is that they do not require manual feature extraction to work (Kandel and Castelli, 2020). However they can be prone to overfitting especially if there is insufficient data.

## Bias vs. Variance

The goal is to identify a model that accurately captures the detail of the training data but also generalises well on unseen data. This results in a trade off between variance error and bias error.

Bias error is where the algorithm fails to incorporate important relationships between the features and the labels and can result in underfitting. (Brownlee, 2020b)

Variance error is where the algorithm captures the relationships in the training data too well, resulting in overfitting. Variance is high if a different result is seen with different test data.

Linear models often have high bias and low variance whereas non-linear models often have

high variance and low bias.

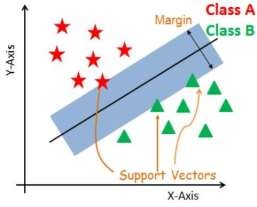
## Models Developed

1. Naive Bayes

With Naive Bayes, each pair of features being classified is assumed to be independent of the other. The model is a linear parametric model as it assumes a Gaussian distribution for the data. The approach is derived from statistics and probability theory. With Naive Bayes it is easy and fast to predict the class.

1. Support Vector Machines

This ML classifier attempts to identify the hyperplane that best separates the classes using the training data. The support vectors are the points closest to the hyperplane (Navlani, 2019).



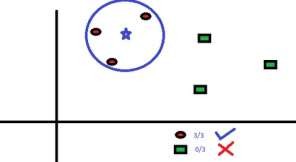
**Fig 2.2 Two classes separated by a hyperplane. The support vectors are the points closest to the hyperplane. The larger the margin the better. From Navlani, 2019.**

Any new points are either inside or outside the boundary established and are classified accordingly.

Large datasets can require a high training time.

* + 1. kNN

kNN algorithms are easy to interpret and train quickly. A kNN algorithm is a lazy learner in that it memorises the dataset. A model isn’t generalised but rather each new example is compared with existing examples in the training dataset.



**Fig 2.3 with k = 3 the three nearest neighbours to the blue star determine its class. From Srivastava, 2018.**

The choice of k is critical to the success of the model and they can take a long time to classify

unseen data.

* + 1. CNN

A neural network has layers of nodes which map inputs to outputs. The inputs are multiplied by the weights in that node, with the results summed to give the summed activation. An activation function is used to transform this to an output (Brownlee, 2019).

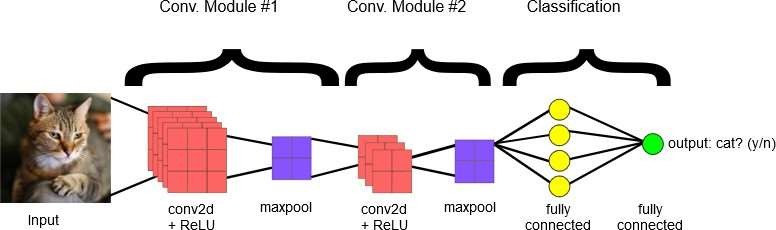
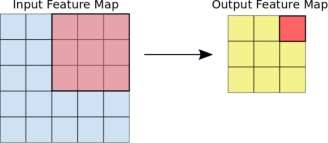


Fig 2.3 **Basic structure of an image classification CNN. The CNN shown here contains two convolution modules (convolution + ReLU + pooling) for feature extraction, and two fully connected layers for classification. Other CNNs may contain larger or smaller numbers of convolutional modules, and greater or fewer fully connected layers. From *ML Practicum: Image Classification | Machine Learning Practica, 2020.***

### Convolution

A convolution extracts tiles of the input image (feature map), and applies filters to them to compute new features (ML Practicum: Image Classification | Machine Learning Practica, 2020). See Fig 2.4. An input that is 96x96 pixels that has 32 filter applied will result in an

output from that Convolutional layer that is 96\*96\*32, where padding is applied (Ruizendaal, 2017). A Conv2d was chosen as the images have 2 dimensions.



**Fig 2.4 A (3,3) kernel size tile performs a calculation on each 3\*3 set of pixels in a 5\* 5 feature map, resulting in a 3\*3 output. Padding (with blank rows/columns) can be used to bring that output back up to the same size as the original feature map. From ML Practicum: Image Classification | Machine Learning Practica, 2020.**

### Activation

Activation introduces non-linearity into the system (ML Practicum: Image Classification | Machine Learning Practica, 2020).

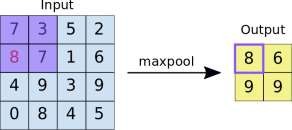
ReLU activation (**rectified linear activation unit**) implements a piecewise linear rectifier

function (Brownlee, 2019) and:

* provides mostly linear activation
* is easy to train and implement
* overall provides better performance than purely nonlinear activation such as sigmoid or hyperbolic tangent

### Pooling

This process allows the model to extract meaningful features such as textures, edges, and shapes from the image (ML Practicum: Image Classification | Machine Learning Practica, 2020).



**Fig 2.5 A maxpool (2\*2) filter identifies the maximum value in each section of the input. The stride determines the distance between each section where the filter is applied. A stride of 2 means that all non-overlapping 2\*2 sections will be filtered with only the maximum value of each input section preserved. From ML Practicum: Image Classification | Machine Learning Practica, 2020.**

### Fully connected / dense layers

Where every node is connected to every node in the next layer the layers are fully connected or dense. These layers perform the classification based on the features extracted by the convolution layers.

### Batch Size

Batch size is one of the many hyperparameters that must be tuned to optimise performance of a CNN in image classification (Kandel and Castelli, 2020). A high batch size can result in the network taking too long to reach convergence, where there is no more gain in accuracy, whereas a low batch size can prevent the network reaching acceptable performance (Kandel and Castelli, 2020).

### Loss

Loss is an error function that measures how far the model’s predictions are from their labels.

## 2.5 Benefits of automated image classification

People are capturing and sharing a vast amount of image data, with the advent of smart phones and social media. Image classification can help with image organisation and storage, provide businesses with product discoverability, provide automated medical image analysis, and assist with quality control in manufacturing (Maruti Techlabs, 2020).

# Requirements Specification and Design

## Project / Business requirements

The business requirement is to develop models capable of successfully classifying images of cats and dogs.

## Information Requirements

The data involved are in the form of images. There are no missing data, outlier issues, or invalid data. The photos utilised are freely available from the Kaggle Dogs vs. Cats competition from 2013.

## Tools and Techniques

The following equipment, tools and technology was used:

* + - Images sourced from:

o <https://www.kaggle.com/c/dogs-vs-cats>

* + - Python 3.8, Tensorflow 2, and libraries such as Pillow & SciKit Learn
    - Jupyter Lab 2.2.8
    - Personal Laptop
    - Excel & Word

A deep learning approach is the main technique that is being implemented. However, neural networks are difficult to understand and it is not easy to explain how they achieve the results. They are in many respects a ‘black box’ with high computational requirements and a potential for overfitting.

Other approaches where the mechanics of the model are easier to understand include kNN models. However, though these can train quite quickly they can take a long time to test, and require the entire training set to be saved in memory.

## Top-level Design

1. Image data

The image data involves 12,500 images each of cats and dogs which are all labelled. There is

another set of 12,500 unlabelled images of cats and dogs for validation.

1. Image processing

Data preparation was implemented using python. See Appendix 1 for examples of the image processing involved.

* + the 25,000 training images were converted from RGB to greyscale
  + the files were cropped to a square based on their minimum dimension and then resized to 96 x 96 pixels
  + each image file was flattened and converted to a dataframe which was exported to a csv file such that each row represented a single image
  + Four training sets were created and saved:
    - the native greyscale images saved in a csv file as above
    - as with greyscale images with additional threshold function applied before flattening
    - edge features extracted from each greyscale image, and saved as above
    - combined feature/texture detail identified for each image, and saved as

above based on a concatenation of

* + - * hu moments
      * haralick features
      * Histogram of Oriented Gradients
  + each training set was saved as a csv file to be imported for modelling

# Implementation

Jupyter Lab was used to develop implement and test the python code. A lot of the code for the models is available in pre-packaged functions within libraries such as SciKit Learn and Tensorflow 2.

## Naive Bayes

A Gaussian Naive Bayes ML classifier was developed using the SciKit Learn library. With Naive Bayes there are strong independence assumptions and little tuning possible.

## Support Vector Machines

A linear SVM classifier was developed using the SciKit Learn library.

* 1. kNN

A k Nearest Neighbours classification ML algorithm was implemented using the SciKit Learn library. The main parameter to be tuned is the value of k.

* 1. CNN

A deep learning CNN algorithm was the main technique investigated for implementation. The suggestion from the literature is that the linear models will underfit and the kNN will struggle with the large volume of features. However, CNNs are hard to understand or explain. Central to the complexity is the number of parameters and the range of options for each and how they can be tuned.

The base model was implemented using Tensorflow 2, based on a few core options:

* + - Convolution using Conv2D layers with a (3,3) kernel, as the images have 2

dimensions.

* + - ReLU activation was chosen.
    - Pooling was provided by a maxpool (2,2)
    - Softmax activation with 2 classifications was used to provide the output
    - A batch size of 50 was implemented with 15 Epochs
    - SparsecategoricalCrossentropy was used as the loss function as the categories are

simply 0 and 1 as opposed to say [1 , 0] and [0 , 1].

# Testing and Results

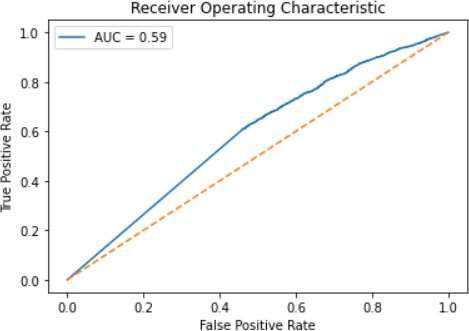
## 1. Naive Bayes

Naive Bayes was implemented and applied to all 4 datasets.

A 70:30 training test split was used to train each model and the resulting model s were then applied to the unseen validation datasets derived from 185 images of cats and dogs.

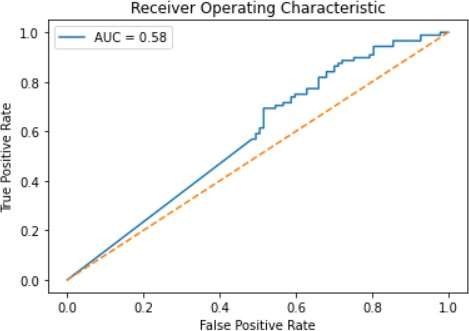
1. Greyscale Dataset

The AUC for the model was 0.59

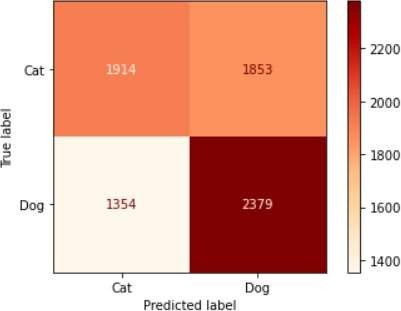


**Fig 5.1 AUC for the Naive Bayes applied to the greyscale training dataset**

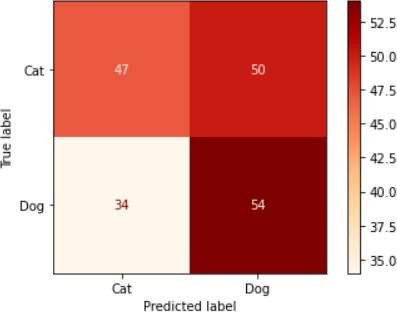
The AUC for the unseen data was 0.58.



**Fig 5.2 AUC for the Naive Bayes applied to the greyscale unseen validation set**



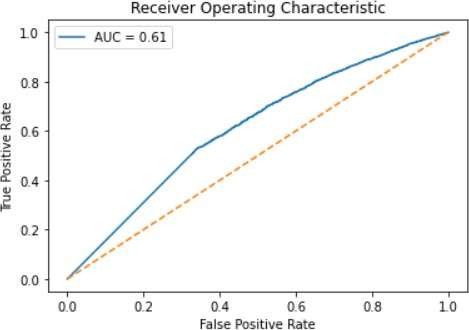
**Fig 5.3 Confusion Matrix for Naive Bayes applied to greyscale training dataset**



**Fig 5.4 Confusion Matrix for Naive Bayes applied to greyscale unseen validation dataset**

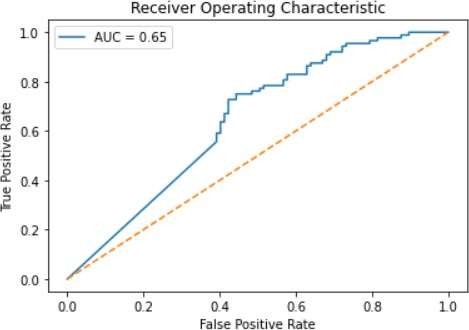
* + 1. Edge Features Dataset

The AUC for the model was 0.61

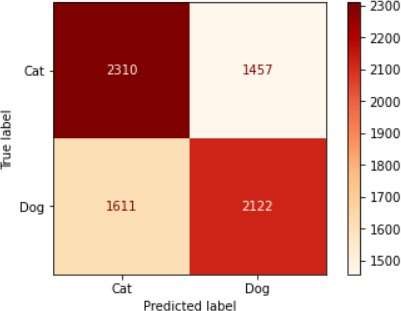


**Fig 5.5 AUC for Naive Bayes applied to the edge features training dataset**

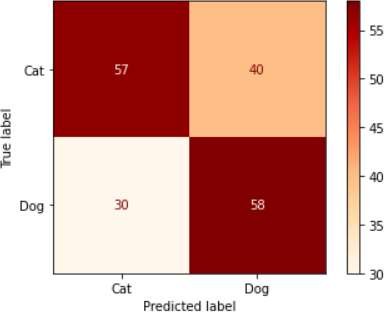
The AUC for the unseen data was 0.65.



**Fig 5.6 AUC for Naive Bayes applied to the edge features unseen validation dataset**



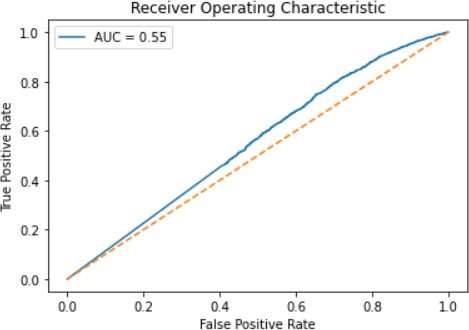
**Fig 5.7 Confusion Matrix for Naive Bayes applied to edge features training dataset**



**Fig 5.8 Confusion Matrix for Naive Bayes applied to edge features unseen validation dataset**

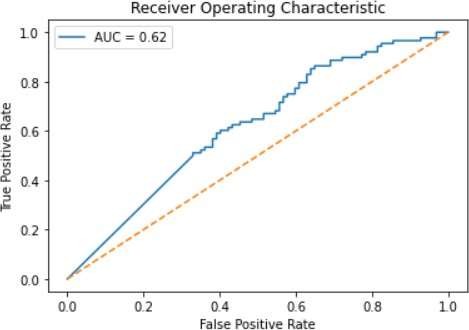
* + 1. Thresholded Image Features Dataset

The AUC for the model was 0.55

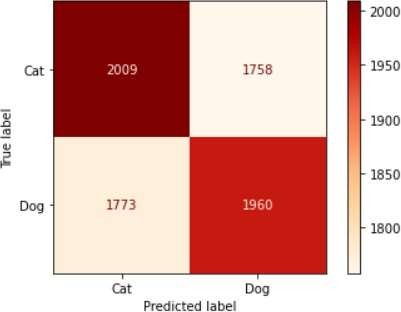


**Fig 5.9 AUC for Naive Bayes applied to the Thresholded features training dataset**

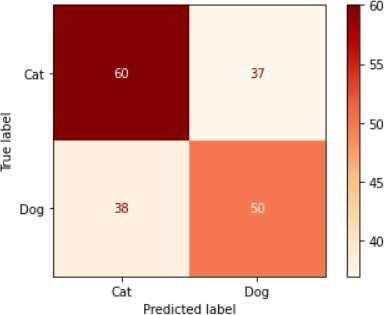
The AUC for the unseen data was 0.62.



**Fig 5.10 Confusion Matrix for Naive Bayes applied to thresholded image features unseen validation dataset**



**Fig 5.11 Confusion Matrix for Naive Bayes applied to thresholded features training dataset**

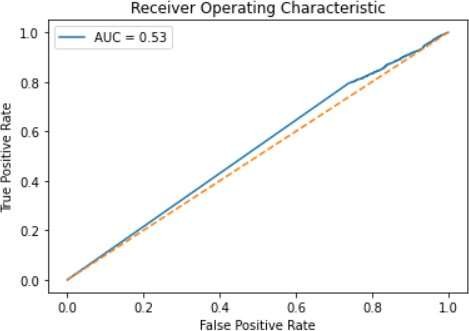


**Fig 5.12 Confusion Matrix for Naive Bayes applied to thresholded image features unseen validation**

**dataset**

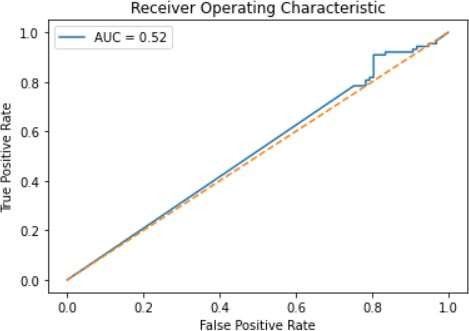
* + 1. Combined Features Dataset

The AUC for the model was 0.53

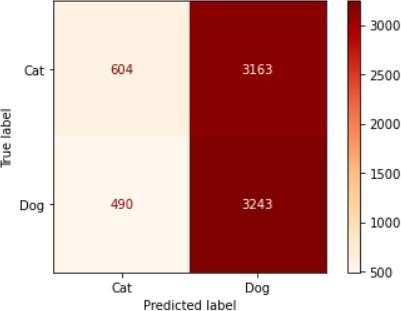


**Fig 5.13 AUC for Naive Bayes applied to the Combined features training dataset**

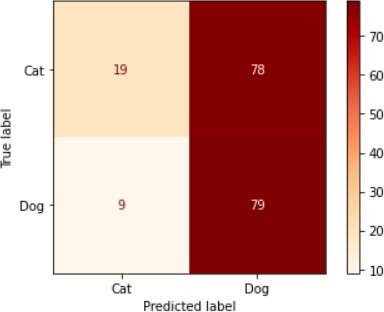
The AUC for the unseen data was 0.52.



**Fig 5.14 AUC for the Naive Bayes applied to the Combined features unseen dataset**



**Fig 5.15 Confusion Matrix for Naive Bayes applied to Combined features training dataset**



**Fig 5.16 Confusion Matrix for Naive Bayes applied to Combined features training dataset**

## kNN

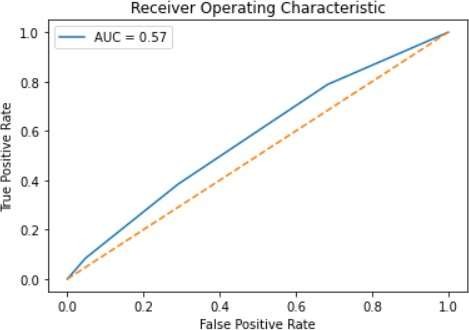
kNN was implemented and applied to 3 of the datasets.

A 70:30 training test split was used to train each model and the resulting model s were then applied to the unseen validation datasets derived from 185 images of cats and dogs.

An initial value for k=3 was used but k =9 was also briefly investifated.

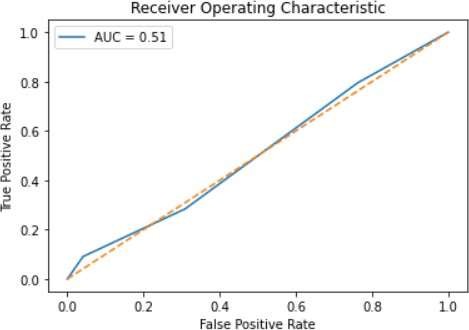
1. GreyscaleDataset

The AUC for the model was 0.57

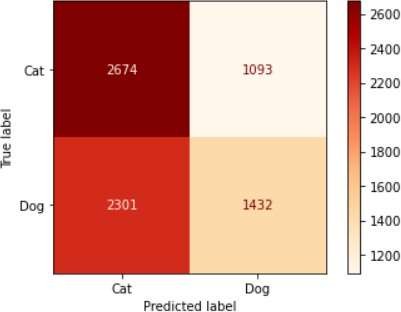


**Fig 5.17 AUC for kNN applied to the greyscale training dataset**

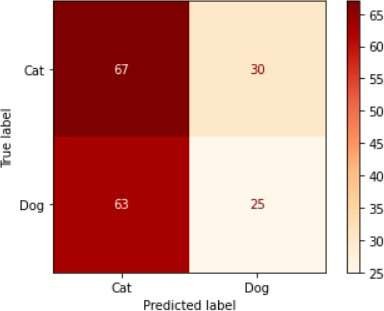
The AUC for the unseen data was 0.51.



**Fig 5.18 AUC for kNN applied to the greyscale unseen validation set**



**Fig 5.19 Confusion Matrix for kNN applied to greyscale training dataset**



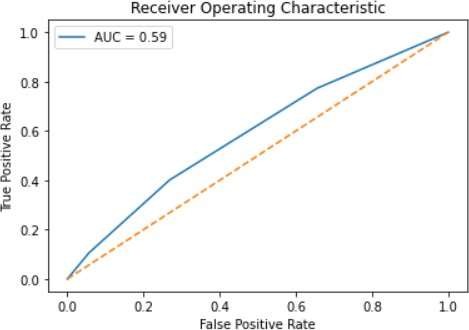
**Fig 5.20 Confusion Matrix for kNN applied to greyscale unseen validation dataset**

* + 1. Edge Features Dataset

kNN was not tested on the Edge Features dataset dues to time constraints and the long classification times that were being seen.

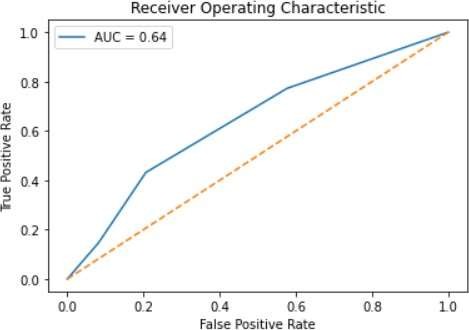
* + 1. Thresholded Features Dataset

The AUC for the model was 0.59.



**Fig 5.21 AUC for kNN applied to the thresholded image features training dataset**

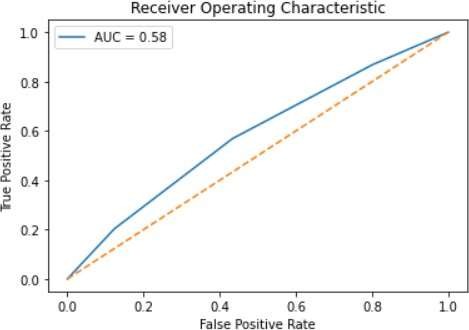
The AUC for the unseen data was 0.64.



**Fig 5.22 AUC for kNN applied to the thresholded image features unseen validation dataset**

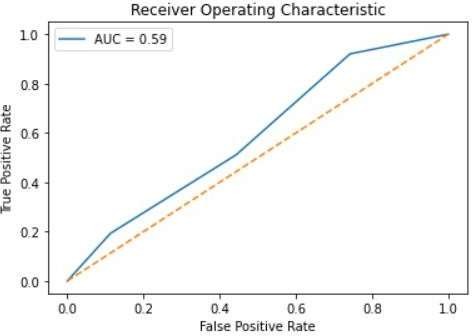
* + 1. Combined Features Dataset

The AUC for the model was 0.58

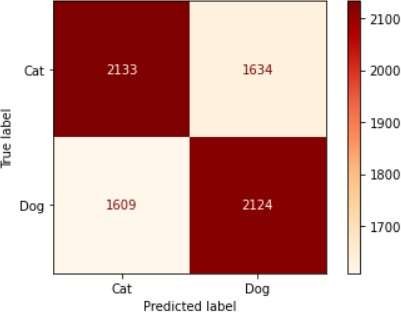


**Fig 5.23 Confusion Matrix for kNN applied to combined features training dataset**

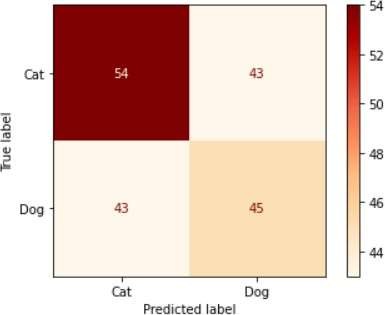
The AUC for the unseen data was 0.59



**Fig 5.24 Confusion Matrix for kNN applied to combined features unseen validation dataset**



**Fig 5.25 Confusion Matrix for kNN applied to combined features training dataset**



**Fig 5.26 Confusion Matrix for kNN applied to combined features unseen validation dataset**

Additional testing was attempted using k = 9, but there was no significant difference seen.

## 3. SVM

SVM was implemented and applied to all 2 datasets.

A 70:30 training test split was used to train each model and the resulting model s were then applied to the unseen validation datasets derived from 185 images of cats and dogs.

1. GreyscaleDataset

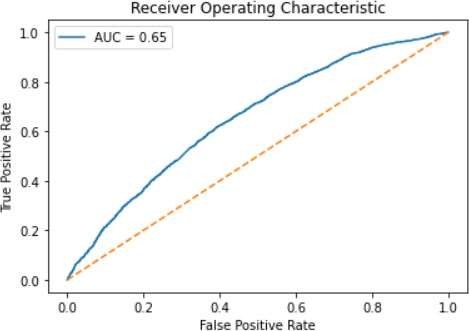
The model failed to complete after 7 days of processing on my system and eventually crashed. Another run using the greyscale dataset wasn’t attempted.

1. Edge Features Dataset

SVM was not tested on the Edge Features dataset .

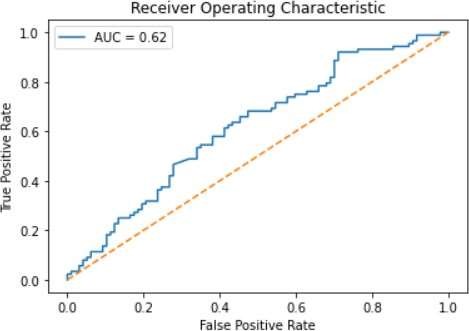
1. Thresholded Features Dataset

The AUC for the model was 0.65.



**Fig 5.27 AUC for SVM applied to the combined features training dataset**

The AUC for the unseen data was 0.62.



**Fig 5.28 AUC for SVM applied to the combined features unseen validation dataset**

5.3.4 Thresholded Image Features Dataset

SVM was not tested on the Thresholded Features dataset.

## CNN

CNN was implemented and applied to the Greyscale Dataset and the Edge Features dataset

only.

The design was initially implemented with 6 blocks of Conv2d/ ReLU and Maxpool layers and 4 dense layers.

A second ‘reduced’ model with 4 blocks of Conv2d/ ReLU and Maxpool layers and three dense layers was also investigated.

The reduced model was tested using batch sizes of 25, 50 and 75.

Additional tuning improve performance saw the addition of batch normalisation and two different implementations of dropouts.

Due to the time taken tuning the model for the greyscale dataset, the model was not applied

to the thresholded image features dataset or the combined features dataset.

Each model was trained using the entire 25,000 images and then validated using the unseen data.

The best model was applied along with kNN and Naive Bayes using traditional 70: 30 training

test split and validation using unseen data, for comparison. This model was also applied to the edge features dataset. A summary of the CNN models used is in Table 5.1.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model Name** | **Dataset used** | **Training volume** | **CNN model** | **Epochs** | **Batch Size** | **Batch Normalisati on** | **Dropouts 1** | **Dropouts 2** |
| **15F50** | Greyscale | 25,000 | Full | 15 | 50 | N | N | N |
| **15R50** | Greyscale | 25,000 | Reduced | 15 | 50 | N | N | N |
| **15R25** | Greyscale | 25,000 | Reduced | 15 | 25 | N | N | N |
| **15R75** | Greyscale | 25,000 | Reduced | 15 | 75 | N | N | N |
| **10R50** | Greyscale | 25,000 | Reduced | 10 | 50 | N | N | N |
| **10R50BN** | Greyscale | 25,000 | Reduced | 10 | 50 | Y | N | N |
| **10R50BND1** | Greyscale | 25,000 | Reduced | 10 | 50 | Y | Y | N |
| **10R50BND2** | **Greyscale** | **25,000** | **Reduced** | **10** | **50** | **Y** | **N** | **Y** |
| **Edge** | **Edge**  **Features** | **17,500** | **Reduced** | **10** | **50** | **Y** | **N** | **Y** |
| **Grey** | **Greyscale** | **17,500** | **Reduced** | **10** | **50** | **Y** | **N** | **Y** |

**Table 5.1 summary of CNN models developed**

The best CNN model developed had the following characteristics:

Training Loss for CNN models

0.8

15F50

10R50BN

15R50

10R50BND1

15R25

10R50BND2

15R75

10R50

Edge Features Grey

0.7

0.6

0.5

0.4

0.3

0.2

0.1

0

1

2

3

4

5

6

7

8

9

10

11 12 13 14 15 16

Epochs

* 10 Epochs
* Batch size of 50
* Batch Normalisation
* Dropouts of 0.5 after the first two dense layers
* ‘Reduced’ model layer blocks

**Fig 5.29 Training loss for the CNN models.**

* The full model (heavy green line) saw the highest loss. All other lines are the reduced model
* The introduction of Batch Normalisation improved the loss (10R50BN)
* The only difference between Grey (25,000) and 10R50BND2 (17,500) was the

amount of data that was used

* The straight blue line is when a batch size of 25 was used.
* The thin green line represents where a batch size of 75 was used.
* The models using batch normalisation gave good performance with 10 epochs A similar picture is evident when examining the training accuracy shown in Fig 5.30.

Training Accuracy for CNN Models

1

0.9

0.8

0.7

0.6

0.5

15F50

10R50BN

15R50

10R50BND1

15R25

10R50BND2

15R75 10R50

Edge Features Grey

0.4

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Epochs

**Fig 5.29 Training accuracy for the CNN models.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Training Accuracy** | **Unseen Data Accuracy** | **Difference** |
| **15F50** | 0.93 | 0.82 | 0.11 |
| **15R50** | 0.99 | 0.87 | 0.12 |
| **15R25** | 0.5 | 0.52 | -0.02 |
| **15R75** | 0.99 | 0.84 | 0.15 |
| **10R50** | 0.96 | 0.87 | 0.09 |
| **10R50BN** | 0.98 | 0.84 | 0.14 |
| **10R50BND1** | 0.97 | 0.88 | 0.09 |
| **10R50BND2** | **0.96** | **0.94** | **0.02** |
| **Edge** | 0.93 | 0.82 | 0.11 |
| **Grey** | 0.96 | 0.89 | 0.07 |

**Table 5.2 Comparison of Training and validation accuracies for the CNN models**

Training and validation accuracy are in Table 5.2, and additional performance metrics are in Table 5.3. The Model 10R50BND2 performs best across all metrics.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Class** | **Precision** | **Recall** | **f1-score** |
| **15F50** | Cat | 0.86 | 0.78 | 0.82 |
|  | Dog | 0.78 | 0.86 | 0.82 |
| **15R50** | Cat | 0.9 | 0.85 | 0.87 |
|  | Dog | 0.84 | 0.9 | 0.87 |
| **15R25** | Cat | 0.52 | 1 | 0.69 |
|  | Dog | 0 | 0 | 0 |
| **15R75** | Cat | 0.9 | 0.85 | 0.87 |
|  | Dog | 0.84 | 0.9 | 0.87 |
| **10R50** | Cat | 0.95 | 0.79 | 0.87 |
|  | Dog | 0.81 | 0.95 | 0.88 |
| **10R50BN** | **Cat** | 0.78 | 0.96 | 0.86 |
|  | **Dog** | 0.94 | 0.7 | 0.81 |
| **10R50BND1** | **Cat** | 0.86 | 0.91 | 0.88 |
|  | **Dog** | 0.89 | 0.84 | 0.87 |
| **10R50BND2** | **Cat** | **0.94** | **0.94** | **0.94** |
|  | **Dog** | **0.93** | **0.93** | **0.93** |
| **Edge** | **Cat** | 0.85 | 0.78 | 0.82 |
|  | **Dog** | 0.78 | 0.85 | 0.82 |
| **Grey** | Cat | 0.9 | 0.89 | 0.89 |
|  | Dog | 0.88 | 0.89 | 0.88 |

**Table 5.3 Additional metrics for the CNN models**



**Fig 5.30 Sample output for a CNN model. The image bordered in red is incorrectly classified. A full view of the output of**

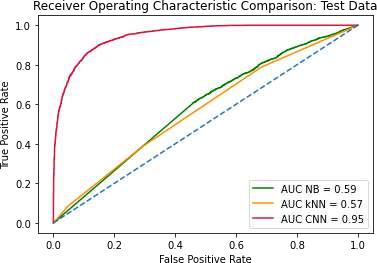
**the best CNN model is in Appendix 3.**

## Comparison data

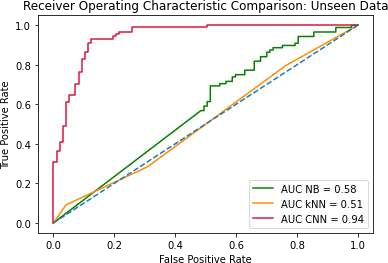
The following graphs show comparison data for three models applied to the greyscale dataset and the edged features dataset.

The models used a 70:30 training test split on the 25,000 records.

* + 1. Greyscale comparison

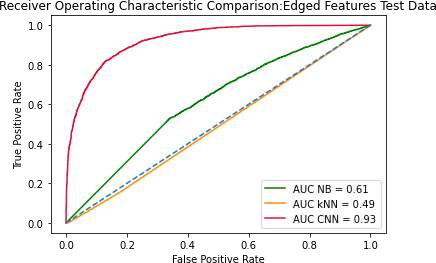


**Fig 5.30: AUC values for three models applied to the greyscale training data**

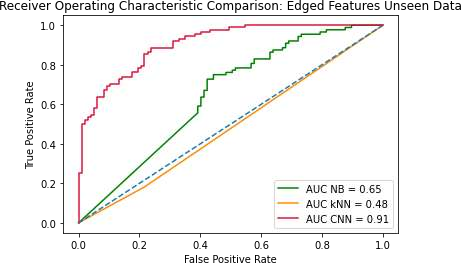


**Fig 5.31: AUC values for three models applied to the greyscale validation data**

The CNN model has significantly better AUC than either kNN or NB.

5.5.1 Edge Features comparison

**Fig 5.32: AUC values for three models applied to the edge features training data**



**Fig 5.32: AUC values for three models applied to the edge features training data**

## 6. Discussion

1. Datasets

The greyscale dataset was the best representation of the images. The edge features datasets was useful bur inferior to the greyscale. I probably didn’t investigate the thresholded data sufficiently, due to time constraints.

The combined features dataset probably used the wrong features. Hu moments and Haralick deal with shape and texture and would not likely be good at differentiating small dogs and cats. However the HOG approach was likely the reason that the data saw some success with SVM. However, as can be seen in Appendix 1 , there is a lot of detail lost when comparing a HOG representation of the original image and the scaled down version.

1. Models

kNN:

* + with k = 3 the results were poor. k= 9 was tried once but the results were no better
  + the nature of the algorithm suggests it isn’t suited to image classification, as there are too many features. Additional work tuning for the optimal value of k might have yielded better results, in combination with feature reduction techniques

Naive Bayes:

* + this is a quick and easy algorithm but clearly underfits and isn’t flexible enough SVM:
  + SVM, and my laptop, struggled to process the image data
  + results with the smaller combined features dataset were reasonable and a better set of image features might be worth investigating
  + a non-linear SVM might be worth investigating CNN:

The CNN model achieved the best results. Early models showed pronounced overfitting as evidenced by Table 5.2. Adding Batch Normalisation improved the loss/accuracy and including dropout regularisation helped reduce the overfitting. This highlights the importance of tuning the hyperparameters.

The final model showed excellent results. It is interesting to note the difference seen when

the dataset was reduced from 25,000 to 17,500.

I would like to have applied to model to a larger number of images.

The CNN model with a batch size of 25 labelled everything as a cat and each image received identical probability scores. If I had implemented this model first with no prior awareness of CNN models I might have concluded that it wasn’t appropriate.

However it is clear that the CNN model was very successful.

# Conclusions and Future work

This applied project successfully developed a model with a training accuracy of 96% and a validation accuracy of 94% using a relatively straightforward CNN architecture.

My interim report involved image colourisation but I switched to image classification on the

advice of my supervisor.

The project demonstrated that deep learning is superior to ML when binary classification of images is involved.

Next step might involve using a larger training dataset validated using a larger number of images as well as investigating Generative Adversarial Networks (GAN) / generative deep learning.

# References / Bibliography

Brownlee, J., 2019. *A Gentle Introduction To The Rectified Linear Unit (Relu)*. [online] Machine Learning Mastery. Available at: [https://machinelearningmastery.com/rectified-linear-activation-](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/) [function-for-deep-learning-neural-networks/](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/) [Accessed 24 August 2020].

Brownlee, J., 2020a. *Parametric And Nonparametric Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at: [https://machinelearningmastery.com/parametric-and-](https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/) [nonparametric-machine-learning-algorithms/](https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/) [Accessed 9 September 2020].

Brownlee, J., 2020b. *Gentle Introduction To The Bias-Variance Trade-Off In Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/gentle- introduction-to-the-bias-variance-trade-off-in-machine-learning/ [Accessed 4 September 2020].

Dataturks, 2018. *Understanding Svms’: For Image Classification*. [online] Medium. Available at:

[<https:](https://medium.com/%40dataturks/understanding-svms-for-image-classification-cf4f01232700)/[/medium.com/@dataturks/understanding-svms-for-image-classification-cf4f01232700](https://medium.com/%40dataturks/understanding-svms-for-image-classification-cf4f01232700)> [Accessed 16 August 2020].

Elson, J., Douceur, J., Howell, J. and Saul, J., 2007. *Asirra: A CAPTCHA That Exploits Interest-Aligned Manual Image Categorization*. [online] Microsoft Research. Available at: [https://www.microsoft.com/en-us/research/publication/asirra-a-captcha-that-exploits-interest-](https://www.microsoft.com/en-us/research/publication/asirra-a-captcha-that-exploits-interest-aligned-manual-image-categorization) [aligned-manual-image-categorization](https://www.microsoft.com/en-us/research/publication/asirra-a-captcha-that-exploits-interest-aligned-manual-image-categorization) [Accessed 7 September 2020].

Google Developers. 2020. *ML Practicum: Image Classification | Machine Learning Practica*. [online] Available at: https://developers.google.com/machine-learning/practica/image- classification/convolutional-neural-networks [Accessed 5 August 2020].

Kaggle.com. 2013. *Dogs Vs. Cats | Kaggle*. [online] Available at: https://[www.kaggle.com/c/dogs-vs-](http://www.kaggle.com/c/dogs-vs-)

cats [Accessed 25 September 2020].

Kandel, I. and Castelli, M., 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*.

Manning, C.D., Raghavan, D., & Schütze, H. (2008) *Introduction to Information Retrieval.* Available at: <https://nlp.stanford.edu/IR-book/html/htmledition/linear-versus-nonlinear-classifiers-1.html> [Accessed 3rd September 2020]

Maruti Techlabs. 2020. *What Is The Working Of Image Recognition And How It Is Used?*. [online] Available at: https://marutitech.com/working-image-recognition> [Accessed 11 August 2020].

Navlani, A., 2019. *(Tutorial) Support Vector Machines (SVM) In Scikit-Learn*. [online] DataCamp Community. Available at: [https://www.datacamp.com/community/tutorials/svm-classification-scikit-](https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python) [learn-python](https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python) [Accessed 14 August 2020].

Rafay, A., 2020. *Main Challenges In Image Classification*. [online] Medium. Available at: https://towardsdatascience.com/main-challenges-in-image-classification-ba24dc78b558 [Accessed 5

September 2020].

Ruizendaal, R., 2017. *Deep Learning #3: More On Cnns & Handling Overfitting*. [online] Medium. Available at: https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting- 2bd5d99abe5d [Accessed 12 August 2020].

Srivastava, T., 2018. *K Nearest Neighbor | KNN Algorithm | KNN In Python& R*. [online] Analytics Vidhya. Available at: [https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-](https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/) [algorithm-clustering/](https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/) [Accessed 18 August 2020].

Torres, J., (2018) *Convolutional Neural Networks for Beginners*. Available at: [https://towardsdatascience.com/convolutional-neural-networks-for-beginners-practical-guide-with-](https://towardsdatascience.com/convolutional-neural-networks-for-beginners-practical-guide-with-python-and-keras-dc688ea90dca) [python-and-keras-dc688ea90dca](https://towardsdatascience.com/convolutional-neural-networks-for-beginners-practical-guide-with-python-and-keras-dc688ea90dca) [Accessed 10th June 2020].

GOOGLE COLAB:

https://colab.research.google.com/drive/1VZOwI\_66v-Wa9-tMXYKE8KjwljQMvWc1

**GITHUB :**

https://github.com/Pragati01234/Dog-vs-cat/commit/1cd1f381739787c7c6904f88ae5a7cf618f0c462

# Appendices

## Appendix 1: Processing of Input image files

This appendix demonstrates the application of the image pre-processing using a sample image. Original image:



500 x 374 RGB 24 bit image 12.1 kB



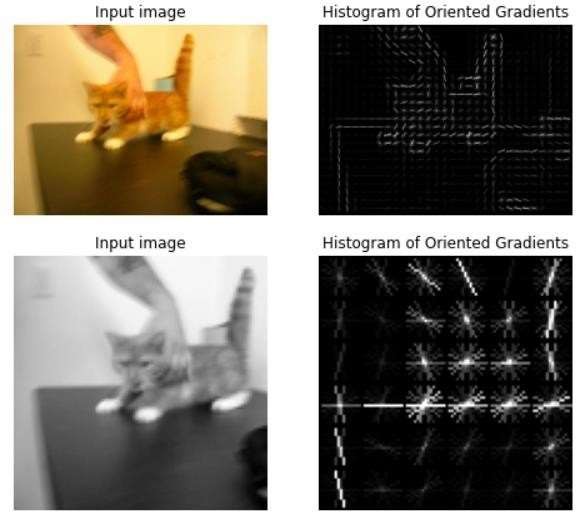
96 x 96 greyscale 8 bit image 1.43kB



96 x 96 edges extracted 24 bit image 3.31kB



96 x 96 thresholded 24 bit image 2.28 kB



Comparison of Histogram of Gradients output for the original colour image and the greyscale image. A lot of detail was lost when the image was converted to 96x96 greyscale.

## Appendix 2: CNN Classification output

This file shows the classification output achieved using the best CNN model: 10R50BND2

10 Epochs of the reduced model with a batch size of 50, using Batch Normalisation and using the second implementation of dropouts.

Please use zoom function to see the detail.

## Appendix 4: Flowchart of project timeline

