CMPE- 258 Deep Learning

Deep Learning CNN Cat & Dog Classifier w/ VGG16

Presented to

Dr. Nooshin Nabizadeh Ardakani

By

William Carrasco | SJSU ID: 015288325

GitHub Repo: <https://github.com/wilcarrasco/CPME-258-Project>

Contents

[Introduction 3](#_Toc89266919)

[Motivation 3](#_Toc89266920)

[Objective 3](#_Toc89266921)

[System Design & Implementation details 3](#_Toc89266922)

[Algorithms / Methods selected 3](#_Toc89266923)

[Technologies and tools used 3](#_Toc89266924)

[Model design/architecture/data flow 3](#_Toc89266925)

[Experiments / Proof of concept evaluation 4](#_Toc89266926)

[Dataset 4](#_Toc89266927)

[Preprocessing 4](#_Toc89266928)

[Implementation and Evaluation 4](#_Toc89266929)

[Model 5](#_Toc89266930)

[Results 6](#_Toc89266931)

[Discussion & Conclusions 7](#_Toc89266932)

[Decisions made 7](#_Toc89266933)

[Difficulties faced 7](#_Toc89266934)

[Things that worked 7](#_Toc89266935)

[Things that didn’t work well 7](#_Toc89266936)

[Conclusion 7](#_Toc89266937)

[References 7](#_Toc89266938)

# Introduction

Deep learning is a field in the artificial intelligence / machine learning domain that has gain a resurgence due to the availability of large datasets and better computational power. Convolutional neural networks are a class of neural networks that are commonly used to analyze visual imagery. This project serves as an attempt to gain a deeper understanding of how to apply a CNN classifier to process and train a model with high accuracy.

## Motivation

The motivation for this project was to become familiar with TensorFlow and the Keras library and utilize these Deep Learning tools to successfully train a model to recognize different images by category. This hands-on approach to learning and applying techniques learned in this course and researched elsewhere will hopefully give a better understanding of training deep neural networks.

## Objective

The object for this project was to apply a convolutional neural network using VGG16 to correctly classify images of dogs and cats with a high degree of certainty as opposed to other methods.

# System Design & Implementation details

## Algorithms / Methods selected

* CNN w/ VGG16
* CNN with three convolutional layers and a single neural network (5 total)

## Technologies and tools used

* Python 3.x
* Jupyter Notebooks
* VSCode, Keras, Pandas, Sklearn, Matplotlib

## 

## Model design/architecture/data flow



# Experiments / Proof of concept evaluation

## Dataset

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

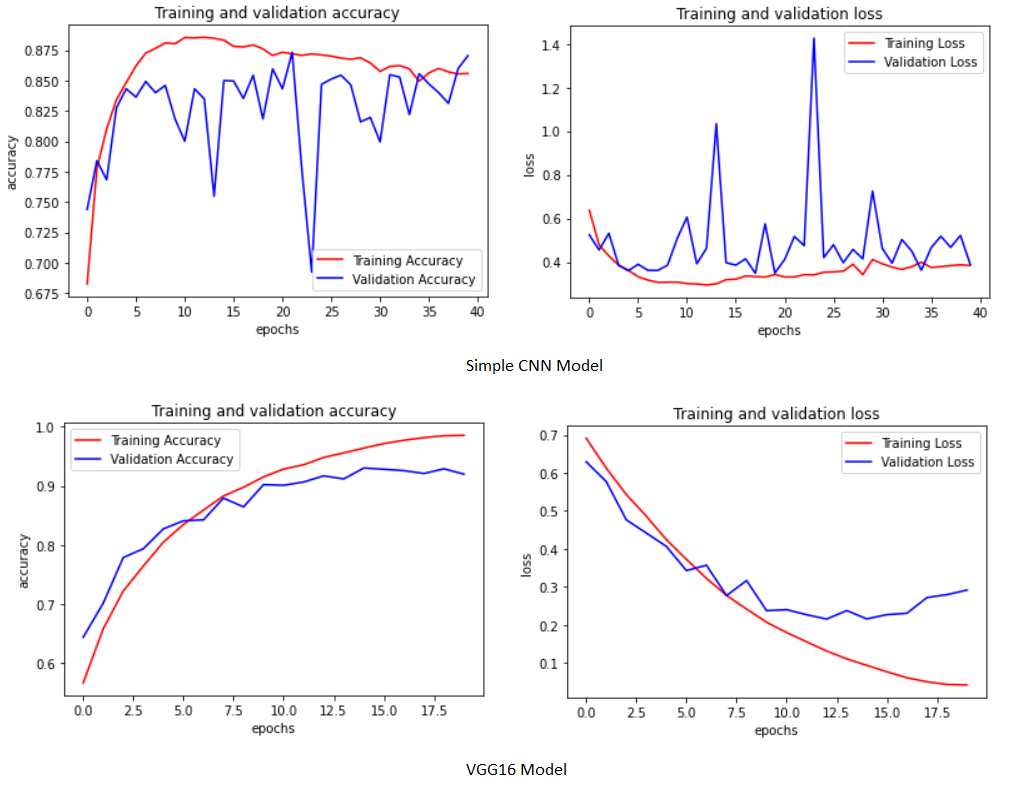
The dataset is a Kaggle competition dataset consisting of different images of dogs and cats. In total, there are 25,000 labeled images with half labeled as dogs and the other half labeled as cats. The dataset also included an unlabeled set of images, 12,500 in total, that was used to test the model. I also included images of my own pet and some random images from the internet to test.

## Preprocessing

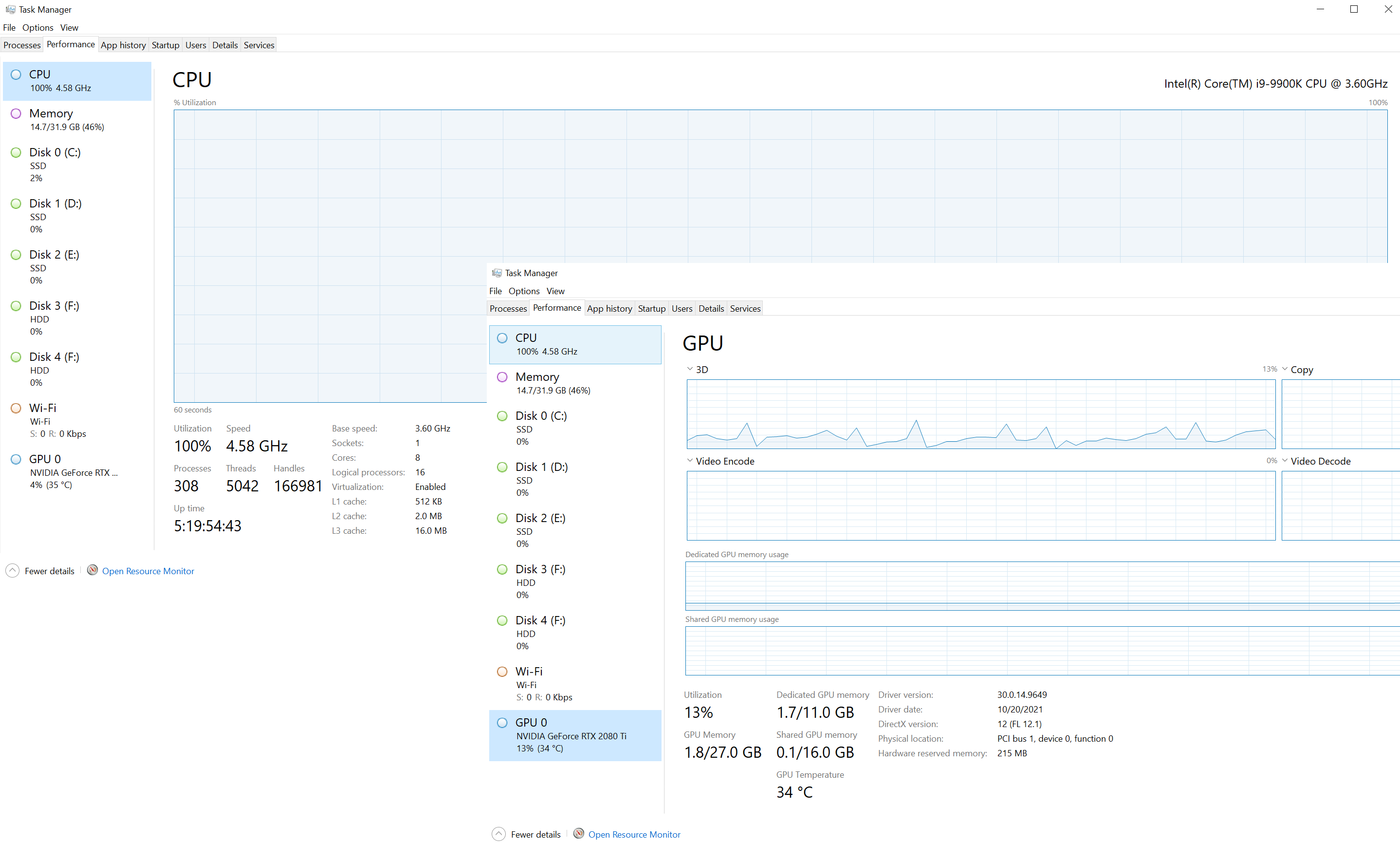
The images were resized to 150 x 150 to aid in faster training. No other image modifications were made. The train set was split into a 90-10% split to be used as the training and validation set respectively. No other regularizations were applied to the images.

## Implementation and Evaluation

I decided on a categorical output instead of a binary classifier, with two output nodes in the final layer, each corresponding to either cat or dog. This was done to get a better sense as to how well the prediction was between cat and dog. The last layer gave a probability of each category type using a Softmax activation. Two different model architectures where selected, a simpler CNN with a total of 5 layers and a modified VGG16 CNN model. Classification was done by selecting the highest predicted value between the two classes, using the array index as the label: 0 – Cat | 1 – Dog. The more complex model took in excess of twelve hours to train while the simple model took about an hour. Interesting, both models performed well, achieving high accuracy ratings but the VGG was superior in training and was a lot smoother during the process:



Processing resources where in high demand during the training phase as noted below:



## Model

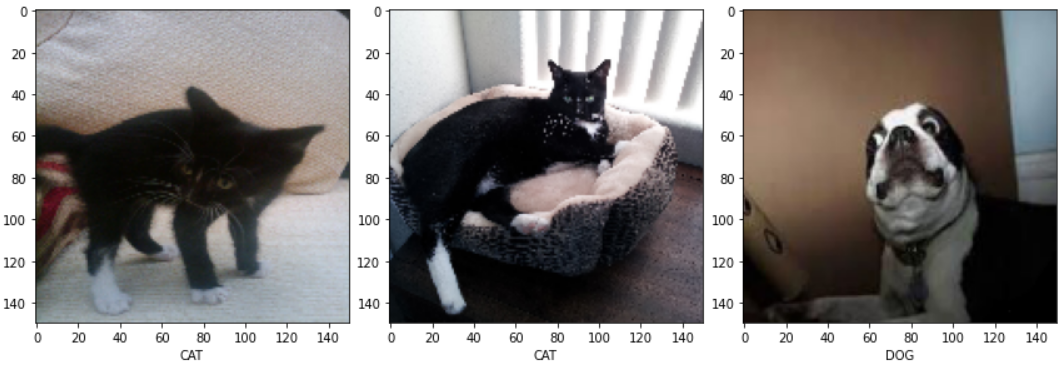
The VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisser from the University of Oxford. The model contains sixteen different weights that can be tuned. Instead of a large set of hyperparameters, focus is placed on multiple convolutional networks with 3 x 3 filters with a stride of one. It is widely used due to it’s ease and ability to process images. Additionally, a simpler model consisting of 3 convolutional neural networks with a single fully connected was employed. This simple model also using a 3 x 3 filter and the performance was compared to give a better idea of how a better model would perform.



## Results

The final VGG16 like model performed better than the simpler CNN Since at the cost of making more time to train. With the supplied test data set, the VGG16 got a score of over 90% in predictions while the simpler model. The simple CNN model scored around 85% accuracy and had a harder time with images that had some occlusions or multiple features. Most of the test focused on the VGG16 model since this was the focus of this project:  
  


The VGG16 model was also ran against random images from the internet and pictures of my cat, again with great results:



Once the model was trained, classification of images was relatively quick. With additional parameter tunning, the VGG16 model accuracy could be improved. Stochastic gradient descent was chosen over other optimizers since it provided the best results.

# Discussion & Conclusions

## Decisions made

I decided to decrease the size of the images to 150 x 150 instead of 224 x 244 in the interest of reducing the time required to train. Additional images were introduced to the model to verify my results and reaffirm that unseen images would be classified correctly.

## Difficulties faced

The main difficulty was time required to train the more complex VGG16 model. Tunning or changing hyperparameters was very expensive to validate. Space requirements to store the dataset since images are huge in size compared to numeric datasets. Lack TensorFlow training caused a lot of time spent learning that framework.

## Things that worked

The VGG16 model worked great, TensorFlow is an easy API to utilize for deep learning tasks. Many libraries existed, including an already built VGG16, that mode development easier. I still opted to create my neural network from scratch for learning purposes.

## Things that didn’t work well

Adam, while fast, this not preform well with the VGG16 model. More knowledge in hyperparameter tunning experience would have been helpful.

## Conclusion

It was interesting to learn about different models used for visual classification and to see an implemented model correctly classify images. Clearly, VGG16 produced great results in my classification model and seeing the train and test graph showed this. I was surprised by how long the model took to train and gained a better understanding of this fact. Overall, it was a interesting project and the hands on approach definitely helped reinforce deep learning concepts.

# References

<https://arxiv.org/pdf/1409.156.pdf>

<https://neurohive.io/en/popular-networks/vgg16>

<https://medium.com/@mygreatlearning/what-is-vgg16-introduction-to-vgg16-f2d63849f615>

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

<https://neurohive.io/en/popular-networks/vgg16/>

GitHub Repo: <https://github.com/wilcarrasco/CPME-258-Project>