Final Year Project Literature Review

Individual Project

a literature review authored by

Wasim Ramzan (u1970064)

and supervised by

Ilias Tachmazidis

A literature review presented for the degree of
Computer Science BSc(Hons)

School of Computing and Engineering
The University of Huddersfield

03 May 2022

Contents

Abstract

This project explores the use of computer vision and how it is used within the modern society. Furthermore, the study researches approaches used today to make use of computer vision which is a subset of machine learning, then applying this to datasets for classification. Vast amounts of people use technologies derived from computer vision in their daily lives but are unaware on the techniques used to reach the end result. For example, facial recognition is something that has become part of many of our lives. We see them at airports, on phones and many other places. Also, what other technologies may arise from computer vision that we can make use of today? I believe there is yet a lot to explore and will use computer vision to research on what architectures and neural networks perform the best given different situations. This is what my project aims to answer by providing an insight on the underlying concepts of certain technologies derived from computer vision. Then I will apply this knowledge towards creating four image classification models using two different datasets and determining which performs the best given the task at hand. I aim to make comparisons between different architectures and neural networks.

Chapter 1

Introduction

1.1 Motivation

Technology has been evolving at a rapid rate and it is slightly difficult to imagine what we will see in the future. We have cars that drive on their own, space exploration and new 5G providing faster speeds. Most companies already understand the importance of certain technologies as they are able to automate impractical and time consuming tasks. Computer vision is an area that we see a lot but is yet to be explored even further.

The important characteristic of computer vision is to allow computers to visualise digital images and videos which we can then perform tasks on. Computer vision is what allows self-driving cars to perform their tasks, which car manufactures such as Tesla, BMW and many others use. Facial recognition used for security also makes use of computer vision through various techniques. The computer vision market is only going to grow in the next years.
"Analytics Insights predicts the computer vision market to reach US$27.02 Billion by 2028 at a CAGR of 7.8% from 2021 to 2028". (Aratrika Dutta, 2021)

As mentioned previously, the ability to automate is vital in modern times. For example, airports that use facial recognition allows them to process individuals a lot faster and accurately as opposed to doing it manually. It is equally important to ensure that the system in place is highly accurate. "According to CSIS data and NIST's studies, FRT algorithms' accuracy can reach up to 99.97%, matching iris scanners." This figure shows facial recognition being incredibly accurate, but not perfect. However, it is still suitable to use in various places. (Recfaces, 2020)

To provide an insight on the use of computer vision I will create a Pokemon classifier as an example using the Pokemon dataset (Zhang, n.d.). There are many more useful real life uses but this would be a good example to explain the techniques. Although there may not be a real-life use case for this, it can have many benefits for example being able to classify images can lead to a faster response time. To analyse this further I will be creating different models using various approaches to determine which one will be the best for this use case. I will also be using a secondary example and repeat my processes to show that this can be done easily again with a different dataset. My second dataset will be determining whether images are a car or truck using the car or truck dataset (Holbrook and Cook, n.d.). Creating Neural Networks is one popular approach which attempts to replicate how the brain operates with neurons. Connections between the neurons may strengthen as we are trained and learn more things. Similarly, machine learning models adopt this principle into creating connections to strengthen the training process of a model.

1.2 Problem Statement

My project idea is to use different deep learning approaches for comparison on their accuracy and performance.

1.2.1 Image Classification

Image classification has evolved greatly over the past years. "The first artificial neural network was invented in 1958 by psychologist Frank Rosenblatt. Called Perceptron, it was intended to model how the human brain processed visual data and learned to recognize objects." (Kay, 2001) Many years later other neural networks such as CNN and RNN were invented. How do these actually work and can be applied to classification?

Classifying Pokemon characters can have many limitations. A poor model and dataset can lead to a low accuracy. Another reason may be Pokemon's that look similar are being classified into the incorrect category. What can we do to extract features from the images in our dataset to avoid these limitations? Despite these barriers, through the use of neural networks and deep learning techniques we are able to classify them correctly to a high degree. I will be exploring all stages into creating this model to resolve the issues mentioned. As well as this, there are different neural networks and architectures which can be confusing at first. By creating models using the different approaches I will hopefully be able to conclude which is the best for different scenarios and why. To do this, I will research in depth the processes they use and comparing many factors.

1.3 Approach

There has been a lot of research done within computer vision. I will use my knowledge to research this area and make my own model which allows for classification. The main areas I will be focusing on are:

- Create Classifiers; Pokemon classification and Car/Truck datasets
- Comparing Neural Networks and Support Vector Machines
- Accuracy Analysis and limitations of the different approaches

I will attempt to create four models for two different datasets depending on how successful I am and time left during my project. Pokemon character classification is the one I will be focusing on first before proceeding to my second dataset.

1.4 Aims

During my project, I will cover my concept to a certain extent. Continuously throughout, I will be answering these questions through completing my projects:

- **Q1: Can my datasets be classified accurately using various models?**
- **Q1.1: What are the different use cases for neural networks and why?**
- **Q1.2: What are the limitations of neural networks and architectures when put through different scenarios?**
- **Q2: Which machine learning approach performs the best and why?**

Chapter 2

Literature Review/Related Work/Similar Products

Through researching my project ideas I have begun to understand what will be required from the technical side. Also, I have learnt the essential components to be cautious about during development of my project. This section, I will be covering these essential components as well as the research I have done which improved my understanding of the subject. Furthermore, I will be discussing similar products to what I am aiming for.

When beginning development I will start with creating simple models then improving them as I go further. This will enhance my knowledge about deep learning especially on developing classification models. I have developed familiarity with a library called TensorFlow which allows for creating models. Within classification one of the main elements is to have a reliable dataset. Hence, I will be using supervised learning to complete this model as my data will be labelled. My dataset will consist of classes and within the classes they will contain their respective images. There are a variety of ways I can get my dataset. For instance, I could scrape images myself or find a dataset online - although I will be using a dataset online. To train my neural network and support vector machines, I can go through approaches such as augmenting my data to ensure I can reach a high accuracy.

Creating classification models has evolved greatly to modern times where we can make an accurate classifier with only few lines of code. Here is an example of a bird classifier I have previously created. You can see majority of the testing set has been correctly classified and this was done in less than 20 lines of code. I will do something similar where my datasets are able to be categorised with minimal error rate - although it will be more complex than the bird one I previous created.

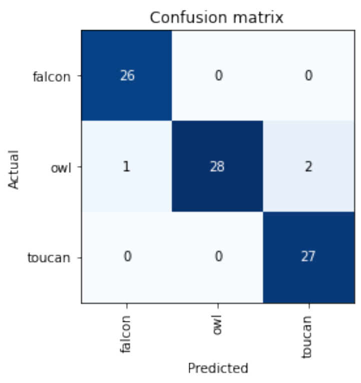


Figure 2.1: Bird Classifier Example Confusion Matrix

Once I have developed a neural network and I am pleased with the results, I will explore other approaches.

A study also goes through the procedure on comparing different models using various datasets. In this case, they used the known MNIST dataset, Network Graphs and CalTech 101 Silhouettes image datasets along with numeric datasets. They have used CNN and SVM/NN for performance comparison. In this study it is shown that NN/SVM models slightly perform better than using a sole CNN. They have combined a neural network along with the support vector machine and as a result it performs better than using a CNN alone. "On the MNIST dataset, every NN/SVM model performed better than their corresponding CNN-only model. The difference, while small, (on the order of 0.01%) does show that these NN features do enable higher performance from an SVM as compared to using either a CNN or SVM alone." (Notley, S & Magon-Ismail, M, 2018)

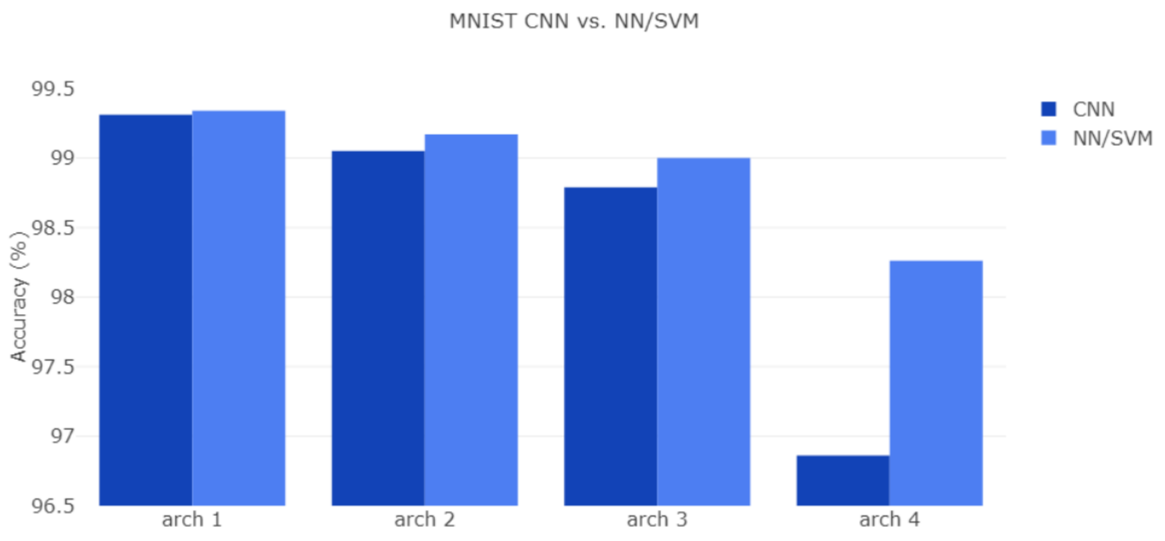


Figure 2.2: MNIST CNN vs NN/SVM
(Notley, S & Magon-Ismail, M, 2018)

I can expect similar results if I were to take this approach. Combining SVM with a neural network we can expect the results to be better than using a single approach. From using this new knowledge my prediction would be using an SVM along with a neural network would be better than using CNN on its own. This combination can increase the performance compared to other approaches. Despite this, all datasets are different and depending on the approach I take - the result can vary. There are other disadvantages of using particular approaches and these will be explained during my analysis. For my implementation I will not be combining different approaches as it would be better to compare what they solely offer.

On the topic of approaches taken, the flow chart below is quite strong as it goes through the typical steps of developing a classification model. Although many other factors may effect the accuracy of this approach. These include how I augment the images, what split I use and the layers involved in creating the model. So, I will proceed with caution and adjust certain parameters until I am confident it is good enough. Depending on the number of classes of my dataset the binary classification may not be applicable as shown in the flow chart. For example, my Pokemon dataset has 150 classes therefore I will be creating a multi-class classification model.

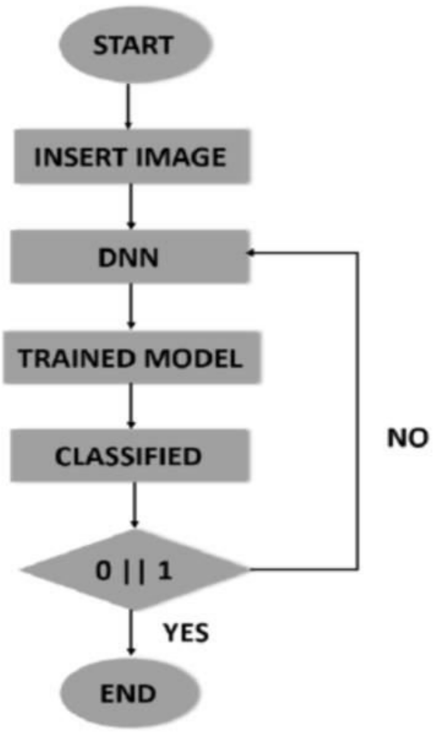


Figure 2.3: Flow Chart Classifier
(Kothair J.B, 2018)

As mentioned previously, there are many use cases for computer vision. These can be applied to real life scenarios providing a wide range of benefits including automation. My initial idea of classifying Pokemon's can be used in the TV industry to sort images etc. Or more advanced technologies can assist people whom are partially blind by automatically giving names of objects.

The flow chart shown in Figure 2.3. is a very high level one as it does not show any underlying concepts but rather an overview of the whole process. Another article which is similarly comparing classification also reaches the conclusion that SVM performs better than CNN. The dataset used this time is a airborne hyperspectral dataset called the Chikusei Dataset. The article outlines in depth the concept of the actual methods themselves and defining how they operate.

SVM is one classification model that I will consider applying to my datasets. "The strategy of this classifier is to find an optimal separating hyperplane with the maximum margin between the classes by focusing on the training samples located at the edge of the class distribution." (Hasan et al., 2019) We can visualise the mapping of data points to separate them into different classes. With a large class and dataset this can be highly memory intensive with SVM's therefore I may have to reduce the size of my dataset. Again, this will be explained later on.

The process for CNN is also explained. Features of images are extracted (pixels) through convolutional layers. These features are processed into a single vector and passed onto the next layer. Gradually as we go through more layers, the neural network is able to analyse more advanced features. The final layer will determine which class is closest by scoring them by confidence scores. Feature extraction is potentially the most vital of this and is done by a kernel or filter which are essential matrixes that perform the dot product on a region of pixels. My models will be explained in-depth during the implementation stage.

A more sophisticated example of a CNN and SVM is outlined below. This will be very similar to when I create the models.

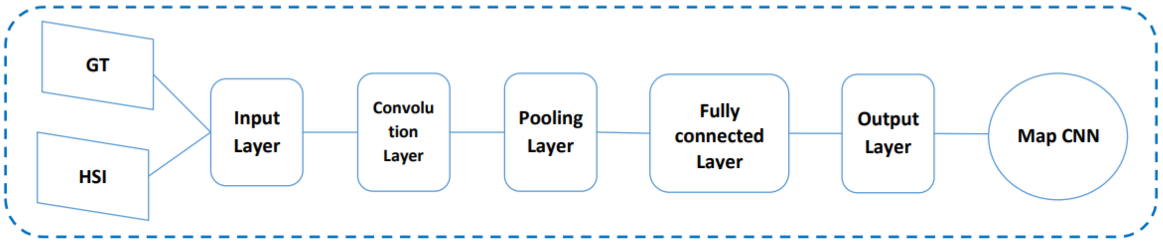


Figure 2.4: CNN Flow Chart #2
(Hasan et al., 2019)

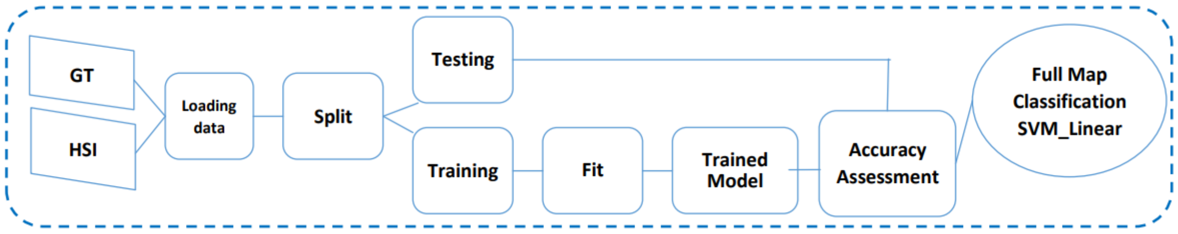


Figure 2.5: SVM Flow Chart
(Hasan et al., 2019)

Chapter 3

Used Technologies

There are various languages, libraries and architectures to assist with creating machine learning models. The language I will be using is Python as it is what I am familiar with and is a great language for creating ML models. I will mainly be using Google Colab whilst in development as I can use a GPU on the cloud. This is a preferred approach as I don't have to rely on my own computers performance. Google Colab provides NVIDIA Tesla K80 GPU which is much better than the one I currently have. All this is for free!
I will be using various technologies throughout but here are some of the main ones that I will be using:

- TensorFlow: is a library that provides us with machine learning algorithms so we do not have to implement them ourselves. It will assist greatly in the creation of neural networks.
- Keras: contains deep learning models and is built on top of TensorFlow. This will primarily be used to import my layers and models such as ResNet. This neural network library is what will allow me to actually create the models structure
- Numpy: library to work with arrays which will hold data for our images. Main use for this is to be able to work easily with numpy arrays to understand the shapes/dimensions of our images or access them if needed. Provides further benefits than a normal Python list and is supposedly "50x faster than traditional Python lists". (w3schools, n.d.)
- Matplotlib: a library providing the ability for data visualisation. To evaluate how my models perform I can create a line graph of the accuracy and measure them against the loss. Visualisation is a great way to understand the performance of my models as opposed to numbers - graphs are an improvement for quick understanding.
- Pandas: data analysis library. Will be used for my SVM where image data is store in a two dimensional pandas dataframe structure.
- Sklearn: Provides a wide range of uses especially to import my SVC (support vector classification). Also for creating test splits and confusion matrixes for both SVM/ANN to view classification accuracies. There are also many further use cases for statistical modelling not just classification.
- ImageNet: large-scale database containing visual images for object recognition. Most likely be loaded as weights so not much configuration needed apart from declaration.
- Flask: A Python framework aimed for web development. It provides good compatibility with my models which are also created using Python. This will be used to create a simple website interface for classifying user images.

These are some of the essential components that I will be using and developing familiarity with over the course of my project. Being able to efficiently use these technologies will allow me to create models quickly rather than having to implement the algorithms/neural networks myself.

Chapter 4

Planning & Methodologies

4.1 Software Methodologies

4.1.1 Agile

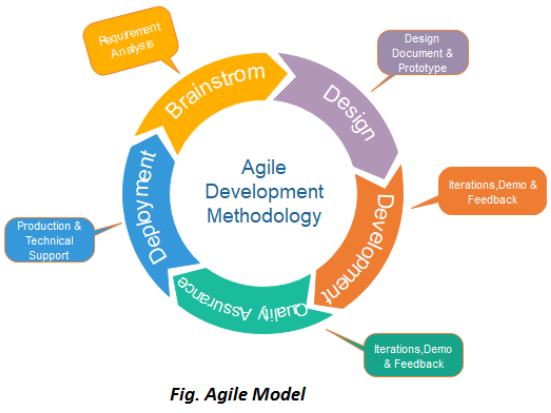
The agile methodology is commonly used within software development as it provides an improved methodology compared to others such as the waterfall model. It was derived due to software developers requiring the ability to continuously improve on their projects. Although agile is used mainly in the software industry - it does exist in other fields as well.

Agile consists of four core values and 12 principles called the "agile manifesto". There are various agile life cycles such as Scrum and Kanban which are derived from agile. Scrum takes an interval approach called sprints where development is done rather quickly. Kanban helps visualise their work and monitor progress of the project.

The agile life cycle I can take can be seen in the image below which consists of 7 phases. Advantage of this approach is that I am able to test concurrently with my development enabling me to have constant feedback on how my models are performing.

Quick overview of my approach:

- Plan; gaining an overview of my goals and how I will reach creating my models. Also the tools that I may use can be in the planning stage
- Design; using this stage to plan my dataset and the deep learning techniques that I will be using. Also, how I will pre-process my dataset etc.
- Develop; beginning creation of my models
- Test; testing the performance of my models and gaining feedback on them
- Deploy; so that I will always have a working model
- Review; determining whether my models can be improved and loop back to design stage to understand how I can proceed with this
- Launch; once my model has been created and I have met my performance goals then I am able to finalise the whole project



Requirement Analysis
Brainstrom
Design Document & Prototype
Design
Agile Development Methodology
Iterations,Demo & Feedback
Development
Production & Technical Support
Deployment
Quality Assurance
Iterations,Demo & Feedback
Fig. Agile Model

Figure 4.1: Agile Methodology
(Javatpoint, n.d.)

4.1.2 Prototype Model

This approach goes through creating a prototype rather than a fully featured application. Once created, the prototype is presented to clients to make improvements based on their feedback. I have decided not to take this approach as it is unclear on how I will become aware that I can begin real development. Also, it is more of a lengthier approach whereas I can get started creating models and make changes during development if necessary. To summarise, it is not catered to my needs as creating a prototype of a machine learning model is uncommon and rather difficult to abstract.

4.1.3 Waterfall Model

The waterfall model allows for working sequentially where I can finish one task and proceed to the next. My tasks will be separated in different sections where I can focus on each one individually. The model below shows the 6 stages which are quite similar to the agile stages. However, you are unable to go back to previous stages which is a huge disadvantage for my project. For example, during testing if I were not getting good accuracy for my models I am unable to go back to the previous stage, preferably system design. Hence, agile is more flexible and suitable for my project.

Another benefit that agile has over waterfall is that I will have a working product earlier on during the project. Furthermore, this results in being able to improve on that product (my model) as much times needed.

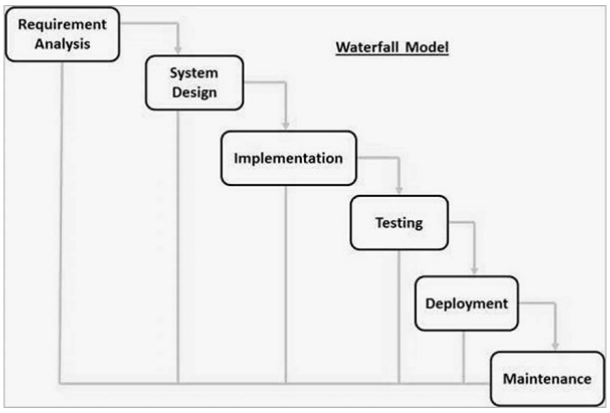


Figure 4.2: Waterfall Model
(tutorialspoint, n.d.-b)

4.1.4 V-Shaped Model

The V-Model takes a sequential approach and is quite similar to the waterfall model. It is also known as the Verification and Validation model due to its approach where testing is paired up with a development phase.

Verification is the process of reviewing the product and consists of several design phases such as:

- Requirement Analysis: the first phase where product and user requirements are made aware of. For my idea this will consist of what I hope to achieve when creating the models. Also, putting myself in the users shoes to understand what they require in terms of performance and reliability of my classification models.
- System Design: once I am aware of the requirements I can begin my designing how to meet those goals by understanding the software requirements. These may include the libraries, languages and frameworks that I plan on using. Since training models has many system requirements I will need to be aware of the memory requirements and whether I need to use a GPU for faster training times.
- Architectural Design: Breaking the system design by separating the functionality into different modules to analyse them individually. I can separate the different models and find the individual requirements for each of them. For example, CNN and SVM are different and I wont need to use the same libraries for both of them. TensorFlow will be used in CNN and not SVM for instance.
- Module Design: "the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD)".(tutorialspoint, n.d.-a) By including the logic within the modules I will have a detailed solution to meet my requirements.

After the coding phase we can begin the testing phase. Testing phases are implemented so the design phases can be reviewed and corrected if needed. Again there are different validation phases outlined below:

- Unit Testing: This will verify the unit tests created within the module design are working and any bugs that arise can be fixed
- Integration Testing: This is performed on Architectural Design where the modules are integrated and tested. Moreover, this verifies the communication between the modules and they are compatible with each other.
- System Testing: Testing the application as a whole with its functional and non-functional requirements. This can be done when I finish creation of my models and develop a simple website to test them.
- User Acceptance Testing: Testing how the system performs in a user environment to determine it is ready for production.

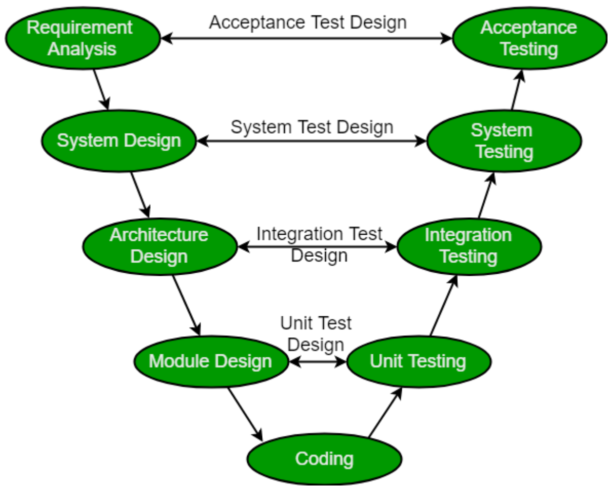


Figure 4.3: V-Model
(GeeksforGeeks, 2022)

4.2 Gantt Chart

Here is the Gantt Chart which will guide me to completing my project. It allows me to work in a systematical manner where future tasks will benefit from previous ones. During the start of my project it will mainly consist of planning and learning technologies that I will be using during the project. Becoming familiar with libraries such as TensorFlow and Scikit learn will be essential as I am required to know how they operate. Also, this will enable me to create models as I have designed them. There is overlapping between machine learning basics and literature review as these are two tasks that do not completely rely on each other and I can do both at the same time without effecting the other (for the most part).

Once I have acquired these skills, I can then proceed to implementing my ideas. Pokemon classification along with another dataset. For each dataset I will create four different models consisting of various methods. As mentioned these are CNN, ResNet, SVM and ANN. Once these tasks are fully implemented to my capability, I can test them properly once again to receive confirmation they work as planned. Upon creation of each model I am required to do data analysis where I create graphs to show the accuracy increasing etc. It is preferable to do this before I begin my report. At least one model from each dataset will be use-able on the notebook or website - upload images for classification.

Finally, I can then finalise both my report and presentation. Note that this is just a guideline, therefore I will be flexible depending on events that may occur during the project.

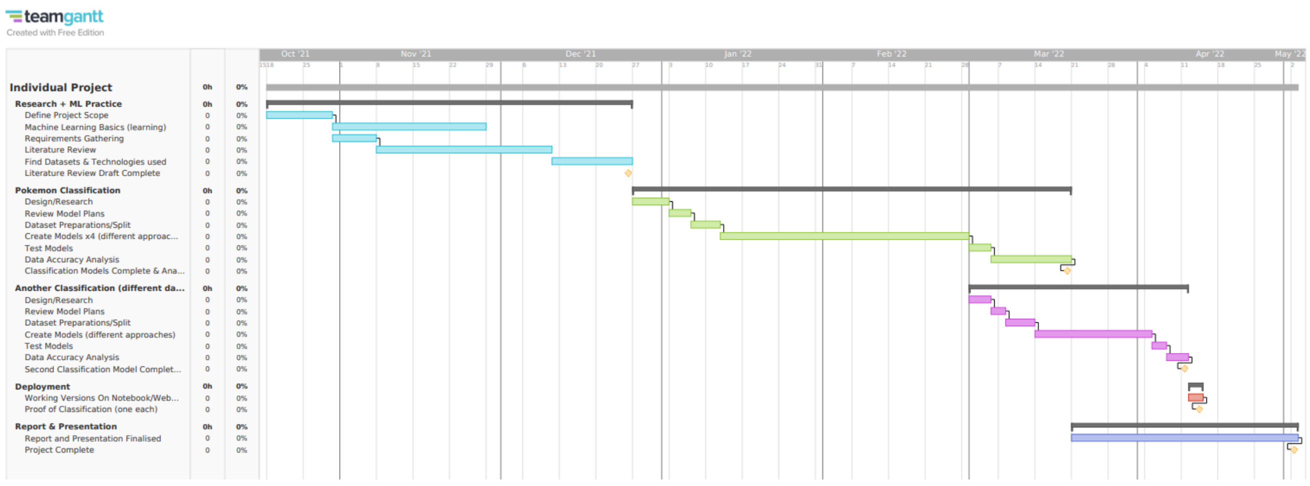
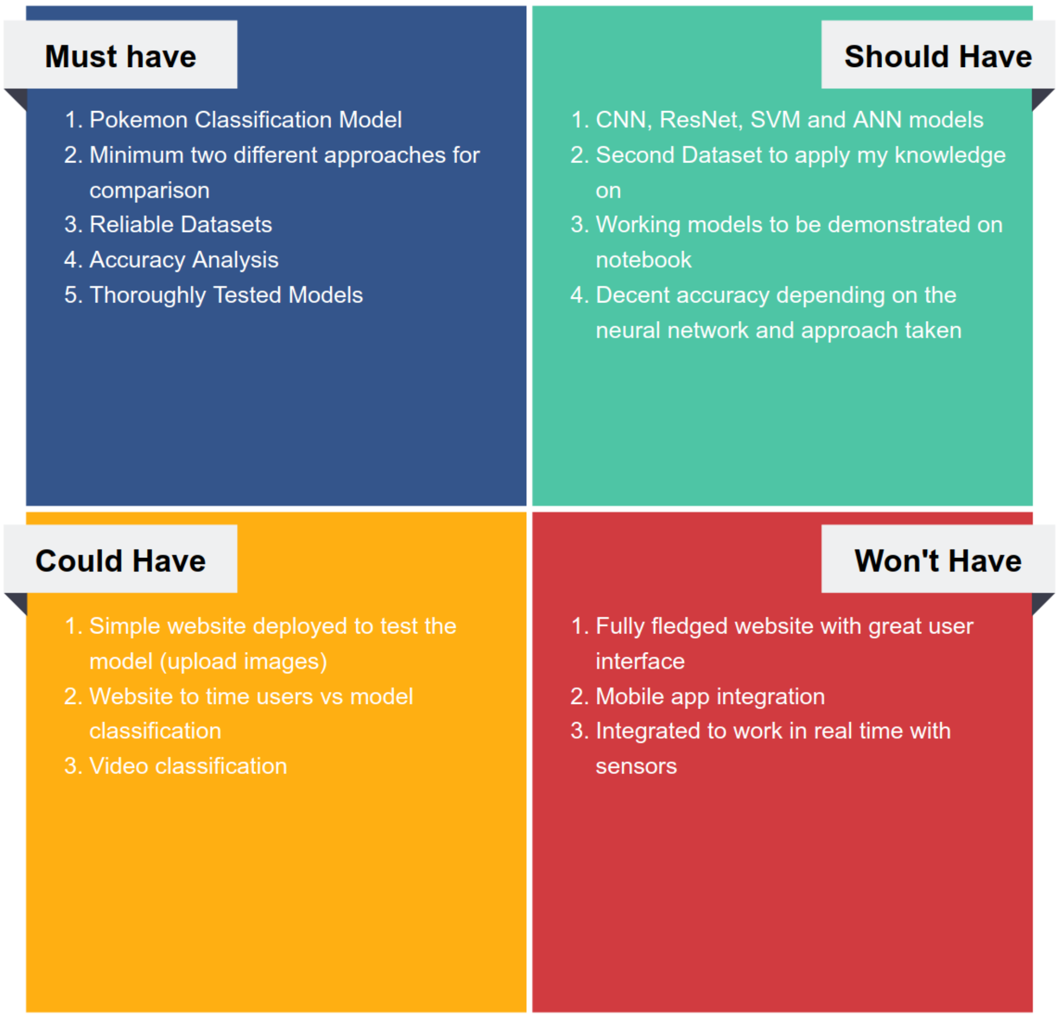


Figure 4.4: Project Gantt Chart

4.3 MoSCoW Analysis

The MoSCoW model below outlines the areas I will be prioritising during my project. It is essential that before my deadline I have met my main goals which are expressed within the **"Must Have"** section. I definitely require a classification model in this case using my Pokemon dataset, as without this I will have no analysis or product to work with. To reach this, I also need a reliable dataset and two different approaches minimum for comparison which is the main goal of my project. Alternatively I **should have** four different approaches and potentially a second dataset to apply the knowledge I have gained from completing the first one. Working models on the notebook so that I can demonstrate the model working. Accuracy must be good for at least some approaches for example CNN and ResNet should have good accuracy for my Pokemon dataset.

I **could have** a website rather than just demonstrating it on the notebook. Deploying it using flask or other python compatible frameworks. Video classification could be my second dataset to show neural networks being able to work with not just images. These aren't vital for my project and are optional. Next, the **won't have** contains all the features I am certain are not required. Having a fully fledged working website with all the features a typical site would have is out of scope for my project. Hence, there is no need for me to spend time on this. Creating a mobile app would also be pointless as I am focused on analysis deep learning approaches. Real-time sensors such as using a raspberry pi or other equipment is too complex for my project therefore are not needed.


Must have

1. Pokemon Classification Model
2. Minimum two different approaches for comparison
3. Reliable Datasets
4. Accuracy Analysis
5. Thoroughly Tested Models

Should Have

1. CNN, ResNet, SVM and ANN models
2. Second Dataset to apply my knowledge on
3. Working models to be demonstrated on notebook
4. Decent accuracy depending on the neural network and approach taken

Could Have

1. Simple website deployed to test the model (upload images)
2. Website to time users vs model classification
3. Video classification

Won't Have

1. Fully fledged website with great user interface
2. Mobile app integration
3. Integrated to work in real time with sensors

Figure 4.5: MoSCoW Analysis Chart

Chapter 5

PLESI

5.1 Professional Issues

Within the computing industry there are professional bodies promoting rules and legislation's that must be followed to prevent any legal issues arising. These professional bodies set codes of conduct which are guidelines of what is expected when working within the computing industry to ensure professionalism. Moreover, the code of conduct may consist of data protection, computer misuse, as well as security, integrity and privacy which must be maintained. The 'BCS, The Chartered Institute for IT' is an example of a professional body.

Within my project I will have to maintain professionalism when conducting my tasks. Currently, I am using reliable datasets which are able for free use, so no copyright infractions will arise. Despite this, if I were to for example decide to create my own dataset using personal information and details of people then I will need to ensure security is strong to prevent any unauthorised access to the data. If my system was to be implemented in real life then I would definitely take caution without how I maintain data being use for deep learning tasks. Privacy will mostly apply towards the Car/Truck dataset as license plates or individuals within their cars may need to be kept private.

5.2 Legal Issues

A facial recognition system is an example of one which can cause many legal issues. For instance, surveillance is something that should be kept secure at all times and only viewed by those permitted. Facial recognition is used world wide, schools, airports as well as police are some examples of those that use such system. Storing images of people in databases - if there were to be a leak this can lead to legal issues to the company. As a result, they must ensure security is a focal point to their storage systems.

As my primary idea is classifying Pokemon there are not much privacy issues that might arise. Ensuring the images that I use are copyright free is one mentioned previously. Also, if I were to create a device that conducts Pokemon classification then I need to ensure the system is accurate as this can cause some issues within production. Lets create a scenario where Pokemon classification was used to separate Pokemon toys into different batches depending on their features. If my model is inaccurate this could be done incorrectly resulting in customers receiving the incorrect product and raising potential legal issues. A more applicable example to legal issues would be for my car or truck dataset as there are real life use cases for such system. Since the arrival of AI within many of our lives there has been great concern for the privacy of individuals. Tracking vehicles of people and what the type of car they are driving could be seen as a breach of privacy.

5.3 Ethical Issues

As mentioned previously, inaccurate models can lead to legal issues. An article discusses the ethical issue of fair-washing. This is the event which promotes the "false perception that a machine learning model respects some ethical values". (Aıvodji, U., Arai, H., Fortineau, O., Gambs, S., Hara, S., & Tapp, A, 2019) A developing company are required to explain the internal logic of the models and how they operate which could be a complex task itself. This could lead to companies using the product believing the model is fair and just when in reality it is not. The article also notes real life circumstances where incorrect decisions has caused issues such as incorrect medical diagnosis which is a huge issue to an individuals health.

To prevent this from arising users need ways to make sure that the models are reasonable and from a reliable developing company who provides fair descriptions of how the model operates internally to prevent issues such as fair-washing from occurring.

Service-level agreements is a contract which outlines the service expected from a customer by the supplier. It may not just be customers but also external suppliers who require the use of such system. If I were to outsource my classification models to these individuals I must take careful considerations when creating my SLA so that it is an accurate definition of what I can offer. Since, my models most likely wont be 100% accurate this can be specified within the SLA so that the consumers are aware. Breaching an SLA may give a company a bad reputation and if it the contract is legally binding then legal implications can occur. SLA's are not always legally binding but as I claimed before it could lead to a bad reputation and losses.

5.4 Social Issues

Footage of people must be kept private as leaks can lead to social issues such as online discrimination and harassment. Leaked footage is something that occurs on a daily basis with it being able to spread quickly on social media. Celebrities for example are known to be a victim of this. This greatly applies to my car/trucks dataset as I may be required to keep individuals vehicles private as individuals may be affected if images of them are leaked. To prevent this we can blur their license plate if it was being used in a legal case for instance.

An article states that "increased surveillance puts vulnerable groups under a microscope". Similar to my project it claims technologies such as facial recognition concerns vulnerable groups and communities. (Negrón, W.N., 2017)

Note: See **Appendix A** for the completed Ethical Form which outlines any issues **during** my project. For image classification there were not any risks or hazards. There were however some considerations needed on the participants of my survey and as to whether they are able to anonymously give their responses (which I provide).

Chapter 6

Project Design & Implementation

My project consists of creating four different models using two datasets. The process for both datasets are quite similar with little modifications during the pre-processing phase and during creation of the model. For this reason, I will mainly be describing the approach I took towards the Pokemon dataset then briefly mention the Car/Trucks dataset. In addition, I will be explaining the process in which I determined how the models were going to be created whilst explaining how they operate. Then I will describe the important aspects of their implementation.

From a high level the models I will be discussing have a similar approach. They have an input which is the images pixels from the dataset itself, the nodes in between (hidden layers) with connections between them and finally the output which determines which class they belong to in this case. The output will be dependent on how the previous operations have been conducted. For example, weak connections between the node may lead to poor accuracy and categorising them wrongly. There are much more processes occurring but it is best I explain them independently as the models handle the three tasks (input, hidden layers and output) differently.

For my Pokemon dataset it consists of 150 classes although for memory intensive models this will be reduced (ANN and SVM). Also, the car trucks dataset number of images will be severely reduced for these models. This is due to memory overload causing the program to crash or long training times.

6.1 Pokemon Classification

6.1.1 Artificial Neural Network (ANN)

Artificial Neural Network Overview

The design of an Artificial Neural Network is no different to what I mentioned earlier. We take an input, process it through the hidden layers resulting in an output. Neural Network and ANN coincide with each-other although the term neural network is used in the biological sense of the nervous system whereas ANN is used within the deep learning field. "Multilayer Perceptron also known as Artificial Neural Networks consists of more than one perception which is grouped together to form a multiple layer neural network". On the other hand, a "simple form of Neural Network and consists of a single layer where all the mathematical computations are performed". (Deepanshi, 2021) My ANN has 2 hidden layers so it is usually referred to as a Deep ANN. Within the layers mathematical functions are performed on the input to understand the data better. As a result, ANN are also known as feed forward networks as the connections between the layers "adaptively change the information received from layer to layer through a series of transformations". (techopedia, 2021) So hopefully, as we go from layer to layer the model is learning more than the previous ones.

ANN operates similarly to how our brain does. If we humans are doing a task and realise we have made a mistake then we will go back are correct it. We also learn from that mistake to prevent it from occurring again. Similarly, if ANN realises a mistake during training then it will go back and change the way it thinks. This process is known as backpropagation. ANN uses weights to learn and if it gets something wrong then the "ANN goes back and changes the weights depending on the accuracy calculated by a cost function". (Meel, n.d.) Essentially, backpropagation calculates the gradient for us and we are able to adjust the weights based on the cost function. Going through the implementation and explaining the functions used will better explain how ANN operates.

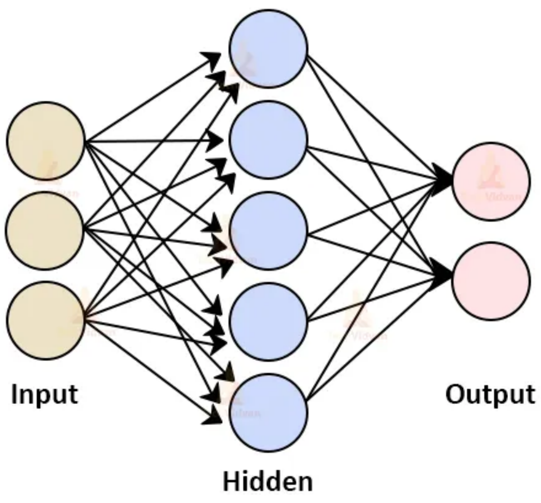


Figure 6.1: ANN Architecture
(Gupta, 2021)

Pokemon ANN Implementation

The first step to implementation is pre-processing the images so they are suitable for the ANN model. For my images I have used five classes as more would have a detrimental impact on the performance and storage requirements. The pokemons I have chosen are Bulbasaur, Charizard, Charmander, Pikachu and Gengar. Amongst these images there are approximately 238 images. These images come in various sizes therefore they have been processed to a size of 100x100. To process these images within an ANN we need to convert them to one-dimensional. As a result, the number of parameters significantly increase. I now have a total of 30000 trainable parameters.

Normalisation is necessary before feeding our input to the neural network as our features scale differently and can impact the accuracy. "If you do not normalize your inputs between (0,1) or (-1,1) you could not equally distribute importance of each input, thus naturally large values become dominant according to less values during ANN training". (Simsek, 2014) In my case I normalised them between 0 and 1 so all my features are treated somewhat fairly. I simply divide by 255.0 so that the pixel range is converted between 0 and 1. We also create an array of indexed labels to later match with the class name.

Next we create our training and validation split. For my models I used a 80-20 split to prevent over-fitting from occurring. It gives us enough data to train with then fit to unseen data.

Now for the most important part I create a Neural Network with the input size of 30000 as specified before. I decided to have two layers consisting of 500 and 250 neurons respectively. The output is 5 as I am classifying it into 5 different classes. The base neural network class and methods can be seen within **Appendix B** which was derived from a combination two sources (Hansen, 2020 & Ahmad, 2020)

Activation's are required for non-linearity when there is not a "direct relationship between an independent variable and a dependent variable".(Hayes, 2021) There are no relationships between the class label and its image inputs therefore if we were to plot them against each other there will not be a straight line but rather a curved one. Hence, to model this non-linear relationship we require activation's when our points show a curved line for accurate results - as our data is not linear so linear regression cannot be used. My ANN makes use of the tanh activation as it speeds up learning by centering the hidden layer outputs closer to zero. "Convergence is usually faster if the average of each input variable over the training set is close to zero".(Amor, 2020) This is just to optimise my results as much as possible as well as making it easier for future layers, and as we know ANN can take a long time to train!

Backpropagation Calculations Explained

How exactly does calculating and optimizing the weights work? Initially we have our forward pass which traverses from the first layer to the last. At the last layer we calculate the loss function from the values received by the output. The forward pass is depicted below note that the activation value is calculated after performing the activation function to the neuron value. In my ANN I use the tanh activation as it has a max derivative up to 1 so we can update the bias and weights more efficiently compared to sigmoid for my ANN.

$$
\begin{aligned}
&x = a^{(1)} && \textit{Input layer} \\
&z^{(2)} = W^{(1)}x + b^{(1)} && \textit{neuron value at Hidden}_1\textit{ layer} \\
&a^{(2)} = f(z^{(2)}) && \textit{activation value at Hidden}_1\textit{ layer} \\
&z^{(3)} = W^{(2)}a^{(2)} + b^{(2)} && \textit{neuron value at Hidden}_2\textit{ layer} \\
&a^{(3)} = f(z^{(3)}) && \textit{activation value at Hidden}_2\textit{ layer} \\
&s = W^{(3)}a^{(3)} && \textit{Output layer}
\end{aligned}
$$

Figure 6.2: ANN Forward Pass
(Kostadinov, 2019)

We then take the value of s against the output y calculating the cost function. Backpropagation then performs the chain rule from the end back to the input. Using gradient descent we update the weights by calculating the gradient of the error function to minimize it. Once the weights are updated we repeat the forward pass using the new weights.

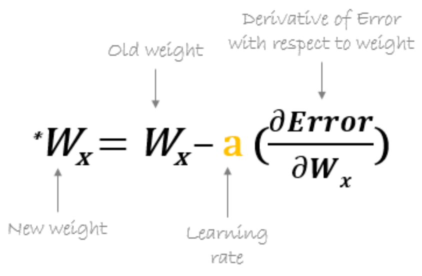

Derivative of Error with respect to weight
Old weight
${}^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$
New weight
Learning rate

Figure 6.3: ANN Update Weights
(hmkcode, 2019)

ANN Analysis

I trained my model for a total of 150 epochs. Epochs are the cycles in which the neural network has been trained with the data in one full cycle. My ANN received a high training accuracy of 99% but 41% on the validation set (unseen data). The loss function decreases as seen by figure . This is a clear sign of over fitting as it performs poorly on the unseen data whilst extremely well on the training data. Figure 6.5. shows the training set classifying most images into the correct category index.

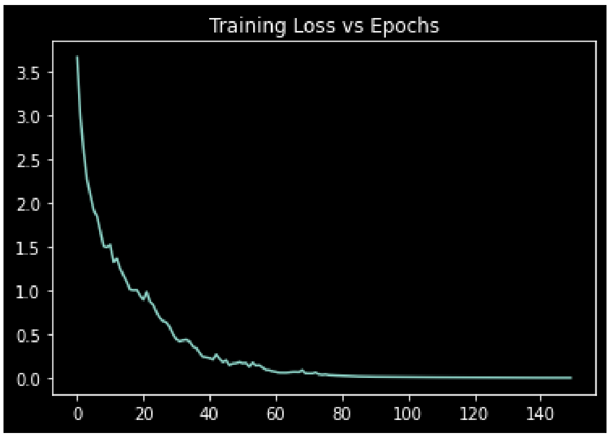


Figure 6.4: Training Loss vs Epochs

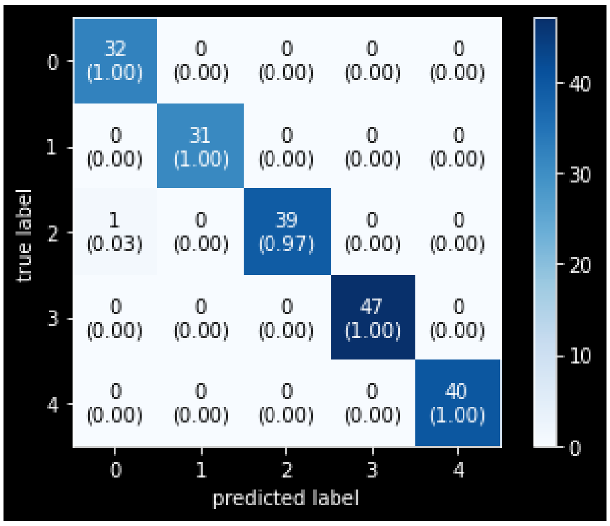
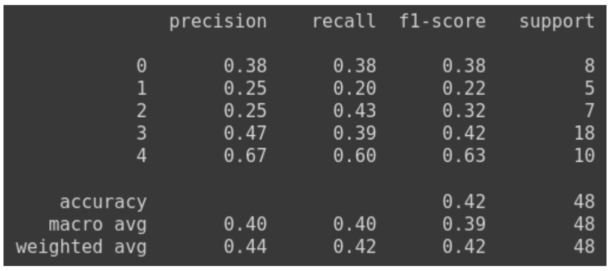


Figure 6.5: Training Confusion Matrix

A 42% accuracy is not ideal for a classifier and the confusion matrix is incorrectly classifying many of the images.



```
              precision    recall  f1-score   support

           0       0.38      0.38      0.38         8
           1       0.25      0.20      0.22         5
           2       0.25      0.43      0.32         7
           3       0.47      0.39      0.42        18
           4       0.67      0.60      0.63        10

    accuracy                           0.42        48
   macro avg       0.40      0.40      0.39        48
weighted avg       0.44      0.42      0.42        48
```

Figure 6.6: Validation Metrics

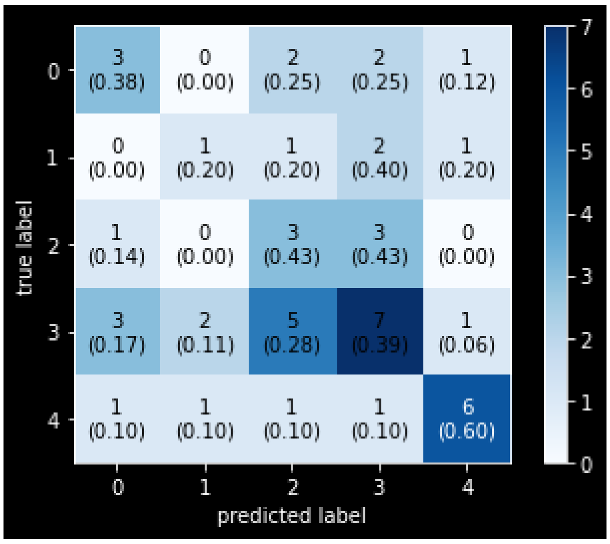


Figure 6.7: Validation Confusion Matrix

6.1.2 Convolutional Neural Network (CNN)

Convolutional Neural Network Overview

CNN is one of the most popular deep learning model when it comes to image classification. The simple neural network is based on the human brain whereas CNN is based on the visual cortex system which allows us to process visual information.

CNNs consist of convolutional layers which "extract features to be used for image classification, with early convolutional layers in the network extracting low-level features (e.g.,edges) and later layers extracting more-complex semantic features (e.g., car headlights)". These convolutional layers are what differs it from other neural networks as it is responsible for feature extraction.

Lets imagine my Pokemon image, During the convolutional layer we begin by selecting a region of the image called a filter or kernel. A filter is just matrix for example 7x7 or 3x3. Within these filters mathematical computation is performed (dot product) and we receive an output called the feature map. This layer is simply responsible for extracting patterns from the image. There are two other important layers.

Pooling Layer is responsible for taking the feature map received from the convolutional layer and reduce the dimension so that it is not computationally costly. The benefit is that "they help to reduce complexity, improve efficiency, and limit risk of over-fitting". (Wang et al., 2020)

The fully connected layer is one of the last layers and is responsible for connecting to nodes from the previous layers. The effect this has is we are able perform classification and determine the output.

CNN Implementation

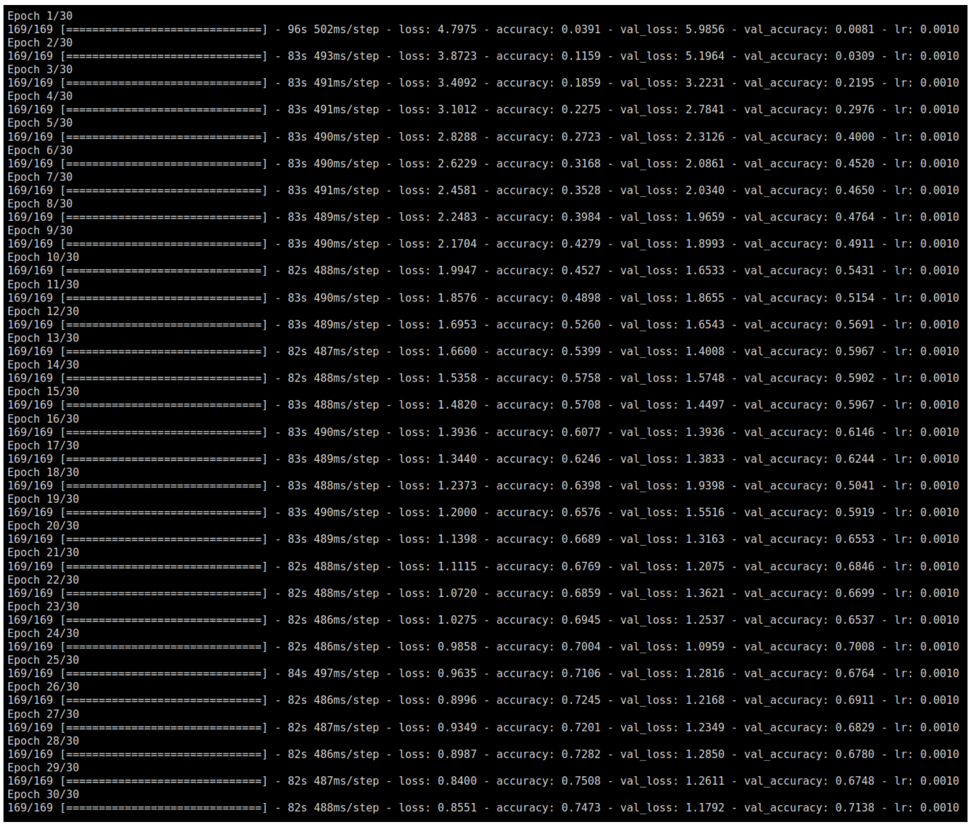
CNN uses less storage than ANN as there is no need to convert to one-dimensional during pre-processing. For this reason we are able to classify all 150 classes. Also, my implementation makes use of TensorFlow and Keras to provide functions to create a CNN.

I have split my dataset using an 80-10-10 split for the train, validation and test set. As we are handling more images 80% will still be suitable for training on. After splitting, the images in the training set are gone through data augmentation. Data augmentation is modify the images by scaling, rotating and shifting. Furthermore, it is not always possible that our images to be perfect therefore by modifying them we can put them through different scenarios. An example is below showing the augmentation of a single Pokemon. Also, during pre-processing we perform other tasks such as specifying the directories of the sets so Keras can access them. In addition the batch size is specified which updates the model after the declared samples are processed.
We then proceed to creating the CNN model. All the layers mentioned previously are declared here. Using the Keras library provided by TensorFlow this is done very easily by just declaring the model as Sequential and adding layers. My model looks quite complex but I have specified multiple convolutional, max-pooling layers. Most of these have been explained but there are some layers I have not discussed. See **Appendix C** for the CNN model structure.

- Batch-Normalisation; normalisation of the layers to speed up training
- Dropout; a regulariser to reduce over-fitting by dropping neurons
- Flatten; converting to 1 dimensional to be inputted in the dense layer
- Dense; basic neural network layer where outputs from one layer and all its neurons outputs are sent to the next

- Relu; a common activation function and estimates decision boundaries that are complex. Vanishing gradients are not an issue with this activation (gradients becoming smaller and harder to predict)
- Softmax Activation; activation function used within multi-class classifiers, " converts a vector of numbers into a vector of probabilities" (Brownlee, 2020)so we can find which class has the highest probability/prediction

After compiling and fitting the model with 30 epochs:



```
Epoch 1/30
169/169 [==============================] - 96s 502ms/step - loss: 4.7975 - accuracy: 0.0391 - val_loss: 5.9856 - val_accuracy: 0.0081 - lr: 0.0010
Epoch 2/30
169/169 [==============================] - 83s 493ms/step - loss: 3.8723 - accuracy: 0.1159 - val_loss: 5.1964 - val_accuracy: 0.0309 - lr: 0.0010
Epoch 3/30
169/169 [==============================] - 83s 491ms/step - loss: 3.4092 - accuracy: 0.1859 - val_loss: 3.2231 - val_accuracy: 0.2195 - lr: 0.0010
Epoch 4/30
169/169 [==============================] - 83s 491ms/step - loss: 3.1012 - accuracy: 0.2275 - val_loss: 2.7841 - val_accuracy: 0.2976 - lr: 0.0010
Epoch 5/30
169/169 [==============================] - 83s 490ms/step - loss: 2.8288 - accuracy: 0.2723 - val_loss: 2.3126 - val_accuracy: 0.4000 - lr: 0.0010
Epoch 6/30
169/169 [==============================] - 83s 490ms/step - loss: 2.6229 - accuracy: 0.3168 - val_loss: 2.0861 - val_accuracy: 0.4520 - lr: 0.0010
Epoch 7/30
169/169 [==============================] - 83s 491ms/step - loss: 2.4581 - accuracy: 0.3528 - val_loss: 2.0340 - val_accuracy: 0.4650 - lr: 0.0010
Epoch 8/30
169/169 [==============================] - 83s 489ms/step - loss: 2.2483 - accuracy: 0.3984 - val_loss: 1.9659 - val_accuracy: 0.4764 - lr: 0.0010
Epoch 9/30
169/169 [==============================] - 83s 490ms/step - loss: 2.1704 - accuracy: 0.4279 - val_loss: 1.8993 - val_accuracy: 0.4911 - lr: 0.0010
Epoch 10/30
169/169 [==============================] - 82s 488ms/step - loss: 1.9947 - accuracy: 0.4527 - val_loss: 1.6533 - val_accuracy: 0.5431 - lr: 0.0010
Epoch 11/30
169/169 [==============================] - 83s 490ms/step - loss: 1.8576 - accuracy: 0.4898 - val_loss: 1.8655 - val_accuracy: 0.5154 - lr: 0.0010
Epoch 12/30
169/169 [==============================] - 83s 489ms/step - loss: 1.6953 - accuracy: 0.5260 - val_loss: 1.6543 - val_accuracy: 0.5691 - lr: 0.0010
Epoch 13/30
169/169 [==============================] - 82s 487ms/step - loss: 1.6600 - accuracy: 0.5399 - val_loss: 1.4008 - val_accuracy: 0.5967 - lr: 0.0010
Epoch 14/30
169/169 [==============================] - 82s 488ms/step - loss: 1.5358 - accuracy: 0.5758 - val_loss: 1.5748 - val_accuracy: 0.5902 - lr: 0.0010
Epoch 15/30
169/169 [==============================] - 83s 488ms/step - loss: 1.4820 - accuracy: 0.5708 - val_loss: 1.4497 - val_accuracy: 0.5967 - lr: 0.0010
Epoch 16/30
169/169 [==============================] - 83s 490ms/step - loss: 1.3936 - accuracy: 0.6077 - val_loss: 1.3936 - val_accuracy: 0.6146 - lr: 0.0010
Epoch 17/30
169/169 [==============================] - 83s 489ms/step - loss: 1.3440 - accuracy: 0.6246 - val_loss: 1.3833 - val_accuracy: 0.6244 - lr: 0.0010
Epoch 18/30
169/169 [==============================] - 83s 488ms/step - loss: 1.2373 - accuracy: 0.6398 - val_loss: 1.9398 - val_accuracy: 0.5041 - lr: 0.0010
Epoch 19/30
169/169 [==============================] - 83s 490ms/step - loss: 1.2000 - accuracy: 0.6576 - val_loss: 1.5516 - val_accuracy: 0.5919 - lr: 0.0010
Epoch 20/30
169/169 [==============================] - 83s 489ms/step - loss: 1.1398 - accuracy: 0.6689 - val_loss: 1.3163 - val_accuracy: 0.6553 - lr: 0.0010
Epoch 21/30
169/169 [==============================] - 82s 488ms/step - loss: 1.1115 - accuracy: 0.6769 - val_loss: 1.2075 - val_accuracy: 0.6846 - lr: 0.0010
Epoch 22/30
169/169 [==============================] - 82s 488ms/step - loss: 1.0720 - accuracy: 0.6859 - val_loss: 1.3621 - val_accuracy: 0.6699 - lr: 0.0010
Epoch 23/30
169/169 [==============================] - 82s 486ms/step - loss: 1.0275 - accuracy: 0.6945 - val_loss: 1.2537 - val_accuracy: 0.6537 - lr: 0.0010
Epoch 24/30
169/169 [==============================] - 82s 486ms/step - loss: 0.9858 - accuracy: 0.7004 - val_loss: 1.0959 - val_accuracy: 0.7008 - lr: 0.0010
Epoch 25/30
169/169 [==============================] - 84s 497ms/step - loss: 0.9635 - accuracy: 0.7106 - val_loss: 1.2816 - val_accuracy: 0.6764 - lr: 0.0010
Epoch 26/30
169/169 [==============================] - 82s 486ms/step - loss: 0.8996 - accuracy: 0.7245 - val_loss: 1.2168 - val_accuracy: 0.6911 - lr: 0.0010
Epoch 27/30
169/169 [==============================] - 82s 487ms/step - loss: 0.9349 - accuracy: 0.7201 - val_loss: 1.2349 - val_accuracy: 0.6829 - lr: 0.0010
Epoch 28/30
169/169 [==============================] - 82s 486ms/step - loss: 0.8987 - accuracy: 0.7282 - val_loss: 1.2850 - val_accuracy: 0.6780 - lr: 0.0010
Epoch 29/30
169/169 [==============================] - 82s 487ms/step - loss: 0.8400 - accuracy: 0.7508 - val_loss: 1.2611 - val_accuracy: 0.6748 - lr: 0.0010
Epoch 30/30
169/169 [==============================] - 82s 488ms/step - loss: 0.8551 - accuracy: 0.7473 - val_loss: 1.1792 - val_accuracy: 0.7138 - lr: 0.0010
```

Figure 6.8: Pokemon Classifier Epochs

CNN Analysis

We can clearly see the validation of accuracy being 71% whilst the training accuracy is 74%. Unlike with the ANN there are no clear signs of over-fitting. The loss also significantly decreases and our validation almost converges and potentially would converge if the number of epochs increases. This is approximately a 73% increase in the validation accuracy compared to ANN.

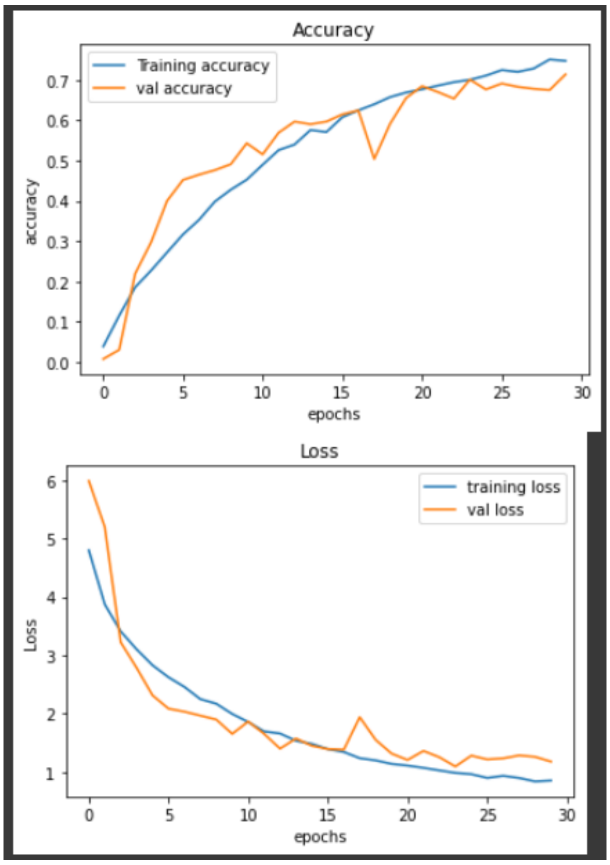


Figure 6.9: Pokemon Classifier Accuracy & Loss

6.1.3 Residual Network (ResNet)

ResNet50 Overview

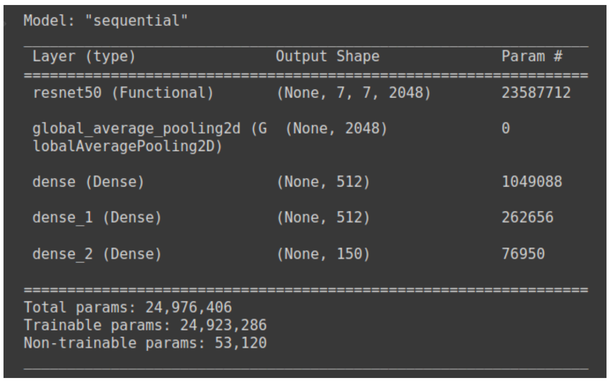
ResNet is a type of CNN architecture. As we are aware, layers in a neural network gradually work towards learning more about the image. Usually, it goes from edges to shapes to objects - level of complexity increases. ResNet comes in different number of layers the one I am using is ResNet-50 which is 50 layers deep and pretrained using Imagenet. Imagenet is a pretrained network using the Imagenet Dataset which consists of millions of images.

Researches found that an increase in layers did not necessarily result in better performance as some models with a small number of layers performed better. This is due to the gradient becoming smaller and makes learning become more difficult - a process called vanishing gradients. As a result, a process called skip connections was introduced. Skip connections "alleviate the issue of vanishing gradient by setting up an alternate shortcut for the gradient to pass through". The neural network can also learn identity functions "which ensures that the higher layer will perform at least as good as the lower layer, and not worse". (Boesch, n.d.) So, we can always rely on this process that the high layers are performing better providing a solution to the vanishing gradients discussed previously.

As I am using the pretrained ImageNet network I am doing a process called transfer learning. Transfer learning is when I use the pretrained network for a new task. In my case the Pokemon classification, so the ResNet50 model pretrained using ImageNet provides me with a starting point.

ResNet50 Implementation

My pre-processing is similar to the CNN where we create the splits and augmentation on my images. The only difference is loading ResNet50 with imagenet as its weights. The diagrams below shows what this looks like. We receive 23587712 parameters from the resnet architecture. Three further dense layers are added and we do not need conv2d like the CNN as these are already added to ResNet. Also uses the relu activation for the first two dense layers and softmax for the final dense layer.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 150)               76950

=================================================================
Total params: 24,976,406
Trainable params: 24,923,286
Non-trainable params: 53,120
_________________________________________________________________
```

Figure 6.10: Pokemon ResNet Parameters

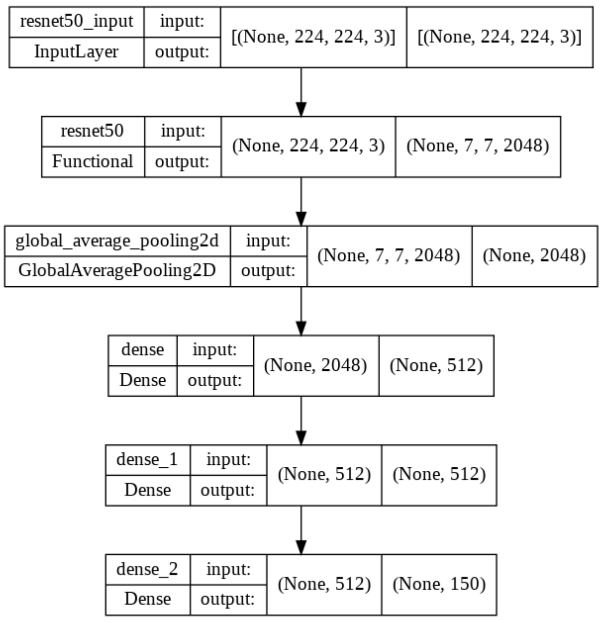
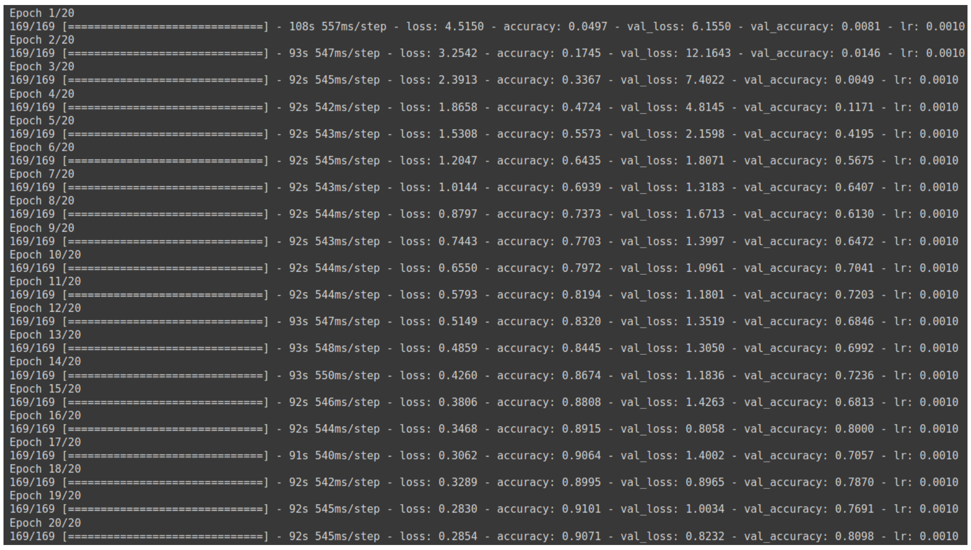


Figure 6.11: Pokemon ResNet Model Structure

ResNet50 Analysis

Training using only 20 epochs we receive the highest accuracy so far of 80% on our validation set! With a lower number of epochs this model managed to perform better than the CNN model I created previously. I decided not to increase the number of epochs as the accuracy began to slow down and I did not want the model to over-fit. The validation loss converges nicely with the training loss almost correlating with the pattern.



```
Epoch 1/20
169/169 [==============================] - 108s 557ms/step - loss: 4.5150 - accuracy: 0.0497 - val_loss: 6.1550 - val_accuracy: 0.0081 - lr: 0.0010
Epoch 2/20
169/169 [==============================] - 93s 547ms/step - loss: 3.2542 - accuracy: 0.1745 - val_loss: 12.1643 - val_accuracy: 0.0146 - lr: 0.0010
Epoch 3/20
169/169 [==============================] - 92s 545ms/step - loss: 2.3913 - accuracy: 0.3367 - val_loss: 7.4022 - val_accuracy: 0.0049 - lr: 0.0010
Epoch 4/20
169/169 [==============================] - 92s 542ms/step - loss: 1.8658 - accuracy: 0.4724 - val_loss: 4.8145 - val_accuracy: 0.1171 - lr: 0.0010
Epoch 5/20
169/169 [==============================] - 92s 543ms/step - loss: 1.5308 - accuracy: 0.5573 - val_loss: 2.1598 - val_accuracy: 0.4195 - lr: 0.0010
Epoch 6/20
169/169 [==============================] - 92s 545ms/step - loss: 1.2047 - accuracy: 0.6435 - val_loss: 1.8071 - val_accuracy: 0.5675 - lr: 0.0010
Epoch 7/20
169/169 [==============================] - 92s 543ms/step - loss: 1.0144 - accuracy: 0.6939 - val_loss: 1.3183 - val_accuracy: 0.6407 - lr: 0.0010
Epoch 8/20
169/169 [==============================] - 92s 544ms/step - loss: 0.8797 - accuracy: 0.7373 - val_loss: 1.6713 - val_accuracy: 0.6130 - lr: 0.0010
Epoch 9/20
169/169 [==============================] - 92s 543ms/step - loss: 0.7443 - accuracy: 0.7703 - val_loss: 1.3997 - val_accuracy: 0.6472 - lr: 0.0010
Epoch 10/20
169/169 [==============================] - 92s 544ms/step - loss: 0.6550 - accuracy: 0.7972 - val_loss: 1.0961 - val_accuracy: 0.7041 - lr: 0.0010
Epoch 11/20
169/169 [==============================] - 92s 544ms/step - loss: 0.5793 - accuracy: 0.8194 - val_loss: 1.1801 - val_accuracy: 0.7203 - lr: 0.0010
Epoch 12/20
169/169 [==============================] - 93s 547ms/step - loss: 0.5149 - accuracy: 0.8320 - val_loss: 1.3519 - val_accuracy: 0.6846 - lr: 0.0010
Epoch 13/20
169/169 [==============================] - 93s 548ms/step - loss: 0.4859 - accuracy: 0.8445 - val_loss: 1.3050 - val_accuracy: 0.6992 - lr: 0.0010
Epoch 14/20
169/169 [==============================] - 93s 550ms/step - loss: 0.4260 - accuracy: 0.8674 - val_loss: 1.1836 - val_accuracy: 0.7236 - lr: 0.0010
Epoch 15/20
169/169 [==============================] - 92s 546ms/step - loss: 0.3806 - accuracy: 0.8808 - val_loss: 1.4263 - val_accuracy: 0.6813 - lr: 0.0010
Epoch 16/20
169/169 [==============================] - 92s 544ms/step - loss: 0.3468 - accuracy: 0.8915 - val_loss: 0.8058 - val_accuracy: 0.8000 - lr: 0.0010
Epoch 17/20
169/169 [==============================] - 91s 540ms/step - loss: 0.3062 - accuracy: 0.9064 - val_loss: 1.4002 - val_accuracy: 0.7057 - lr: 0.0010
Epoch 18/20
169/169 [==============================] - 92s 542ms/step - loss: 0.3289 - accuracy: 0.8995 - val_loss: 0.8965 - val_accuracy: 0.7870 - lr: 0.0010
Epoch 19/20
169/169 [==============================] - 92s 545ms/step - loss: 0.2830 - accuracy: 0.9101 - val_loss: 1.0034 - val_accuracy: 0.7691 - lr: 0.0010
Epoch 20/20
169/169 [==============================] - 92s 545ms/step - loss: 0.2854 - accuracy: 0.9071 - val_loss: 0.8232 - val_accuracy: 0.8098 - lr: 0.0010
```

Figure 6.12: Pokemon ResNet Epochs

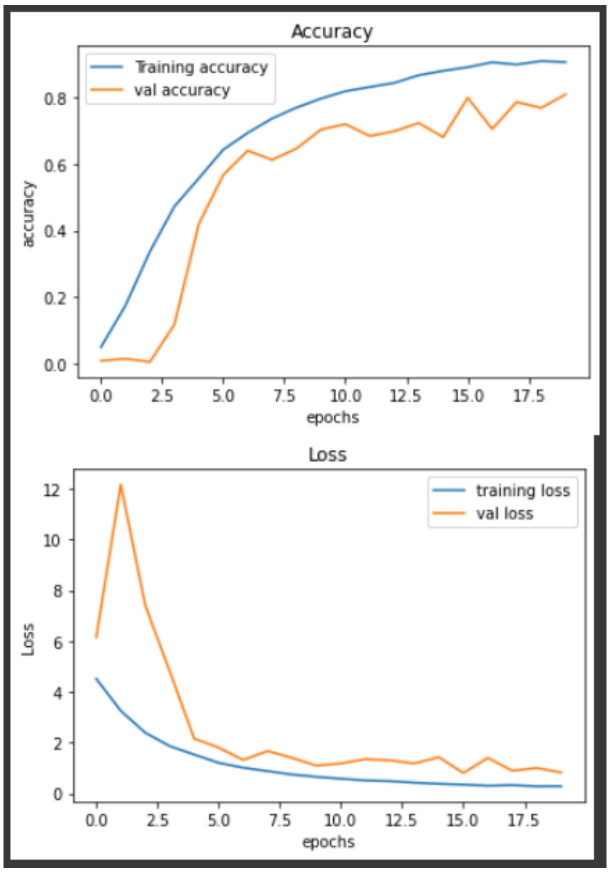


Figure 6.13: Pokemon ResNet Line Graphs

6.1.4 Support Vector Machine (SVM)

Support Vector Machine Overview

SVM's has a simple concept compared to the other models. It is a type of linear model which can be used for classification. The goal of this model is to separate classes by finding a hyperplane that best divides a dataset into their classes. For my Pokemon dataset we are required to have multiple hyper-planes which adds some complexity to the problem. The hyperplane is not random however is determined as the one which maximizes the margin between the classes. This creates a larger difference between them.

Figure 6.15 below shows an example where two classes are perfectly linearly separable by the hyperplane. For my dataset this will be unlikely as features between my images may have similarities therefore we can expect some overlapping.

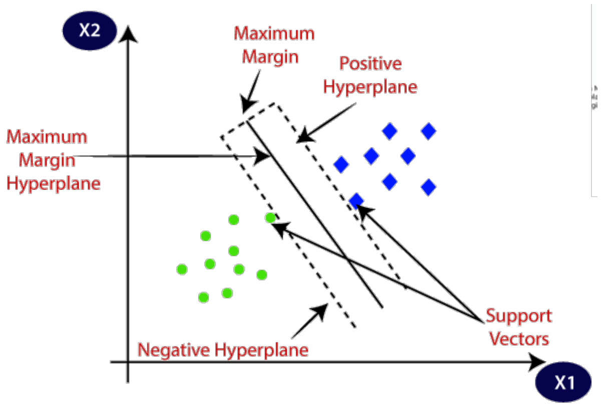


Figure 6.14: SVM Example Hyperplane
(Datasciencelovers, 2020)

Support Vector Machine Implementation

Pre-processing is quite simple as all I did was resize the images and flatten them which converts them to one-dimensional. I do the same with my class names and store these as a numpy array. The image below shows 20 data points plotted and an example on how they can be separated. This is not perfect as there is clear signs of overlapping but is one of the flaws of an SVM. Despite this, we can see some points collated together such as the Charizard points to the left. Rather than one line we have five to separate the five classes.

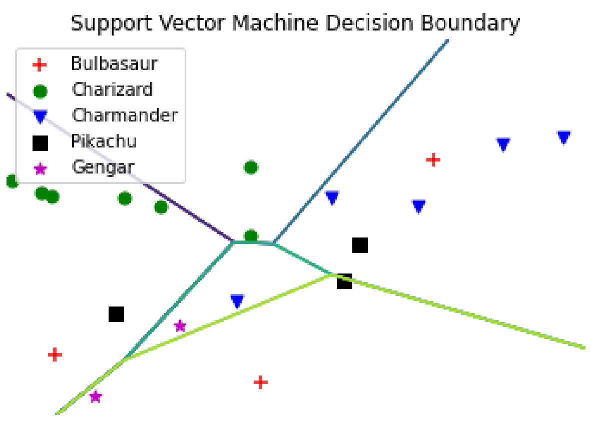
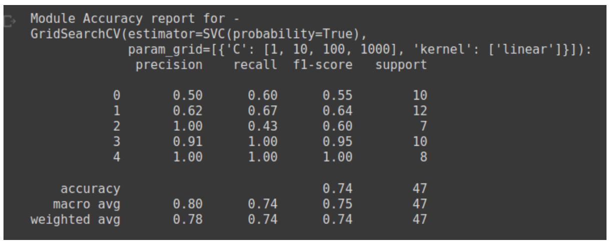


Figure 6.15: Pokemon SVM Hyperplane

Another vital implementation function is the kernel which I specified, I declared linear in this case. This is due to my linear data as seen in the image previously can be presented on a line graph. A disadvantage is that I have five classes which makes it rather complex to separate them and I cannot use a single hyperplane to do this either. Next we can create the split and fit the model against our validation set. There are better kernels such as RBF (radial basis function) which has the ability to map my dataset onto a higher dimension but these take a very long time to train!

SVM Analysis

Surprisingly, we receive a 74% accuracy. This is depicted below on the accuracy report as well as confusion matrix. This is quite a positive result compared to the ANN we created at the beginning.



```
Module Accuracy report for -
GridSearchCV(estimator=SVC(probability=True),
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel': ['linear']}]):
              precision    recall  f1-score   support

           0       0.50      0.60      0.55        10
           1       0.62      0.67      0.64        12
           2       1.00      0.43      0.60         7
           3       0.91      1.00      0.95        10
           4       1.00      1.00      1.00         8

    accuracy                           0.74        47
   macro avg       0.80      0.74      0.75        47
weighted avg       0.78      0.74      0.74        47
```

Figure 6.16: SVM Accuracy

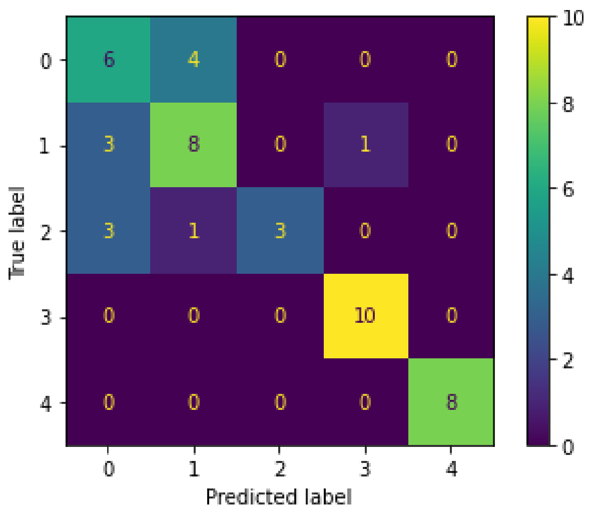
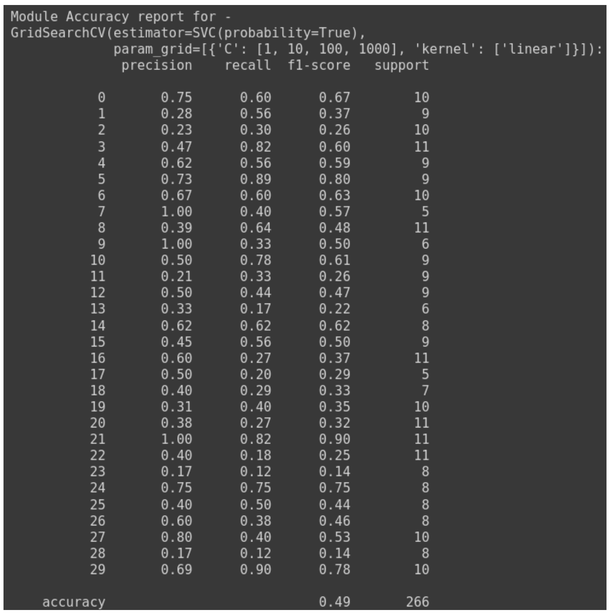


Figure 6.17: SVM Confusion Matrix

This result does not mean it is an improvement over the CNN as we had a higher accuracy. This is because I only used 5 classes as "at prediction time, SVM takes a linear combination of all support vectors. So, if there are a lot of support vectors, you need to store all of them".(Goyal, 2018) Again the memory requirements are too high and caused my program to either perform poorly or take a very long time to train. To put this in a picture, I also trained it with 30 classes and only received 49% accuracy similar to the ANN.



```
Module Accuracy report for -
GridSearchCV(estimator=SVC(probability=True),
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel': ['linear']}]):
              precision    recall  f1-score   support

           0       0.75      0.60      0.67        10
           1       0.28      0.56      0.37         9
           2       0.23      0.30      0.26        10
           3       0.47      0.82      0.60        11
           4       0.62      0.56      0.59         9
           5       0.73      0.89      0.80         9
           6       0.67      0.60      0.63        10
           7       1.00      0.40      0.57         5
           8       0.39      0.64      0.48        11
           9       1.00      0.33      0.50         6
          10       0.50      0.78      0.61         9
          11       0.21      0.33      0.26         9
          12       0.50      0.44      0.47         9
          13       0.33      0.17      0.22         6
          14       0.62      0.62      0.62         8
          15       0.45      0.56      0.50         9
          16       0.60      0.27      0.37        11
          17       0.50      0.20      0.29         5
          18       0.40      0.29      0.33         7
          19       0.31      0.40      0.35        10
          20       0.38      0.27      0.32        11
          21       1.00      0.82      0.90        11
          22       0.40      0.18      0.25        11
          23       0.17      0.12      0.14         8
          24       0.75      0.75      0.75         8
          25       0.40      0.50      0.44         8
          26       0.60      0.38      0.46         8
          27       0.80      0.40      0.53        10
          28       0.17      0.12      0.14         8
          29       0.69      0.90      0.78        10

    accuracy                           0.49       266
```

Figure 6.18: SVM 30 Classes Accuracy

6.2 Car Or Truck Classification

I have applied the same concepts above and created all four models for my Car/Truck dataset. This is a lot simpler as I now only have two classes. Hence, my problem changes from the previous categorical classification to binary classification as both my classes are 0 and 1. Most of the concepts explained previously are applied exactly the same but with few minor changes.

ANN

The only change to this is mainly with the input size. The dimensions of my image are now 128x128x3 (49152). Also, my output size changes to 2 as we are only classifying it into a car or truck. Aside from that everything remains almost the same and I receive 61% accuracy on the validation set whilst 82% on the training set which is quite impressive.

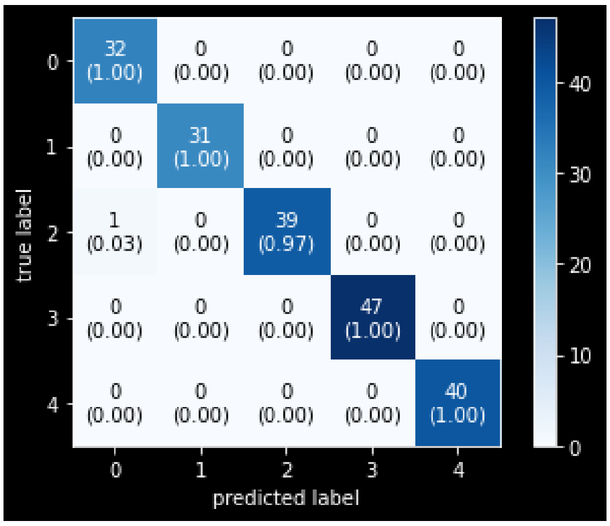
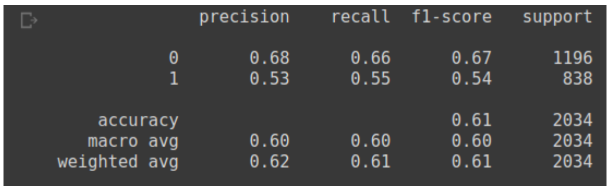


Figure 6.19: Car/Truck Training Confusion Matrix



```
              precision    recall  f1-score   support

           0       0.68      0.66      0.67      1196
           1       0.53      0.55      0.54       838

    accuracy                           0.61      2034
   macro avg       0.60      0.60      0.60      2034
weighted avg       0.62      0.61      0.61      2034
```

Figure 6.20: Car/Truck Validation Accuracy

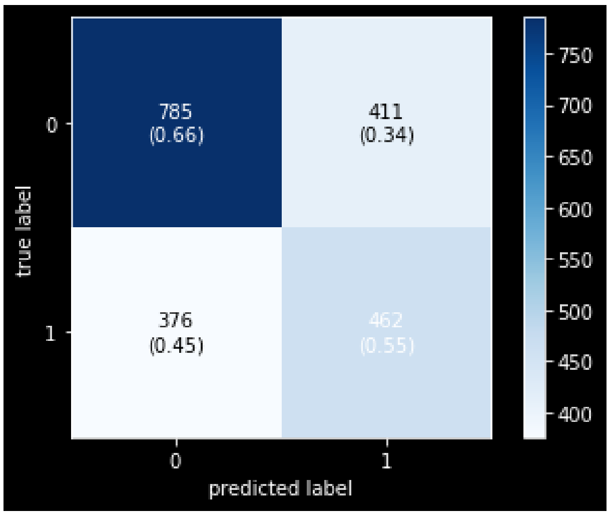


Figure 6.21: Car/Truck Validation Confusion Matrix

CNN

Aside from the input size changing everything else remained the same. We receive 83% accuracy on the validation set whilst 81% on the training. This converges very nicely as the validation set beats the training set. The only issue is that there are signs that the model is not learning that well due to the accuracy not being robust enough. We can see this on the line graphs where the accuracy/loss jumps around.

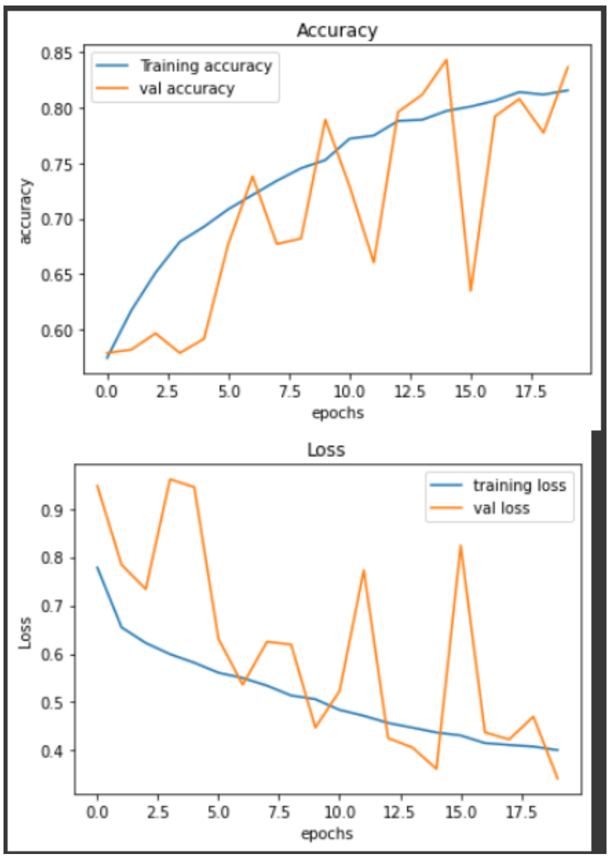
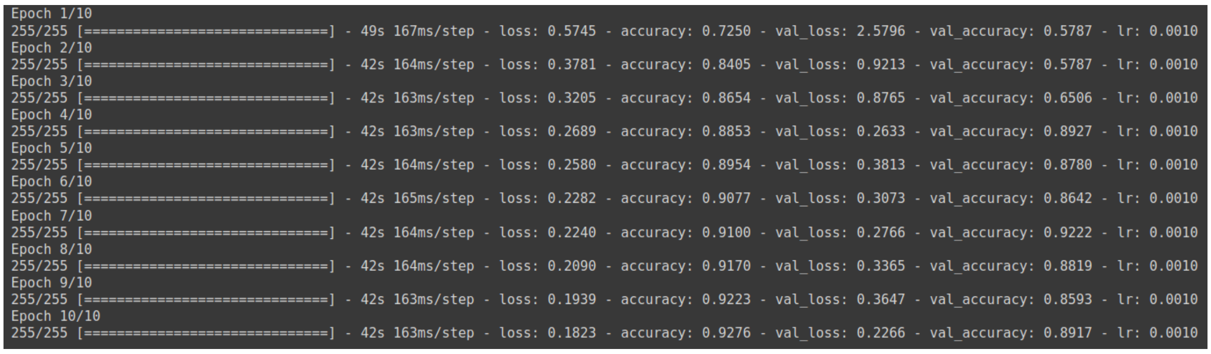


Figure 6.22: Car/Truck CNN Line graph

ResNet

For this I only ran 10 epochs as the accuracy begins to not change that much and it converges very quickly. I receive 89% accuracy after 10 epochs at one point it was 92% at 7 epochs. The line graphs looks alot better than the CNN and we can be confident that the model is learning very well.



```
Epoch 1/10
255/255 [==============================] - 49s 167ms/step - loss: 0.5745 - accuracy: 0.7250 - val_loss: 2.5796 - val_accuracy: 0.5787 - lr: 0.0010
Epoch 2/10
255/255 [==============================] - 42s 164ms/step - loss: 0.3781 - accuracy: 0.8405 - val_loss: 0.9213 - val_accuracy: 0.5787 - lr: 0.0010
Epoch 3/10
255/255 [==============================] - 42s 163ms/step - loss: 0.3205 - accuracy: 0.8654 - val_loss: 0.8765 - val_accuracy: 0.6506 - lr: 0.0010
Epoch 4/10
255/255 [==============================] - 42s 163ms/step - loss: 0.2689 - accuracy: 0.8853 - val_loss: 0.2633 - val_accuracy: 0.8927 - lr: 0.0010
Epoch 5/10
255/255 [==============================] - 42s 164ms/step - loss: 0.2580 - accuracy: 0.8954 - val_loss: 0.3813 - val_accuracy: 0.8780 - lr: 0.0010
Epoch 6/10
255/255 [==============================] - 42s 165ms/step - loss: 0.2282 - accuracy: 0.9077 - val_loss: 0.3073 - val_accuracy: 0.8642 - lr: 0.0010
Epoch 7/10
255/255 [==============================] - 42s 164ms/step - loss: 0.2240 - accuracy: 0.9100 - val_loss: 0.2766 - val_accuracy: 0.9222 - lr: 0.0010
Epoch 8/10
255/255 [==============================] - 42s 164ms/step - loss: 0.2090 - accuracy: 0.9170 - val_loss: 0.3365 - val_accuracy: 0.8819 - lr: 0.0010
Epoch 9/10
255/255 [==============================] - 42s 163ms/step - loss: 0.1939 - accuracy: 0.9223 - val_loss: 0.3647 - val_accuracy: 0.8593 - lr: 0.0010
Epoch 10/10
255/255 [==============================] - 42s 163ms/step - loss: 0.1823 - accuracy: 0.9276 - val_loss: 0.2266 - val_accuracy: 0.8917 - lr: 0.0010
```

Figure 6.23: Car/Truck ResNet Epochs

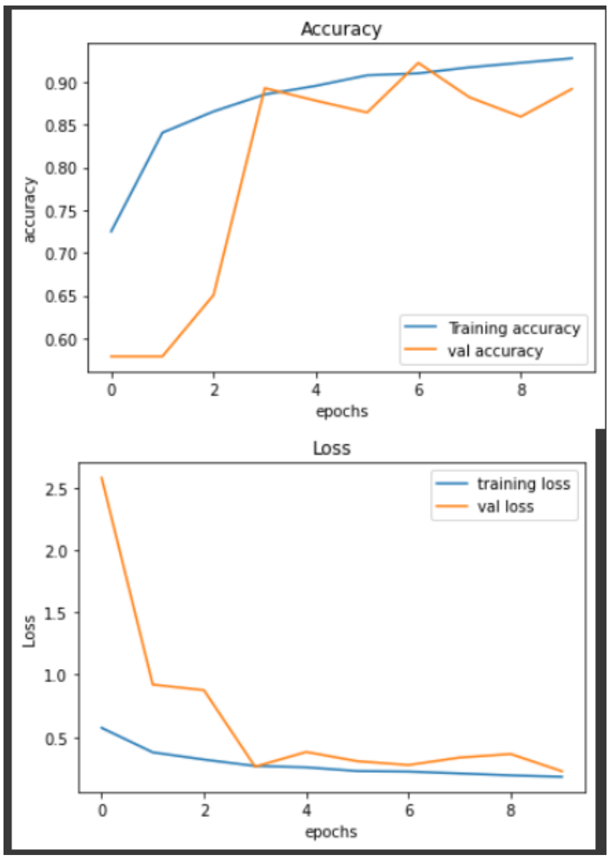


Figure 6.24: Car/Truck ResNet Line graph

SVM

Receiving 63% accuracy which is very similar to the ANN's accuracy. Despite this we can now separate our classes with a single hyperplane rather than 5 as we are only handling two classes. As you can see, this is linearly separable although there are some overlapping between the classes. Despite this majority of the car class points are on the left - some uncertainty for the truck class.

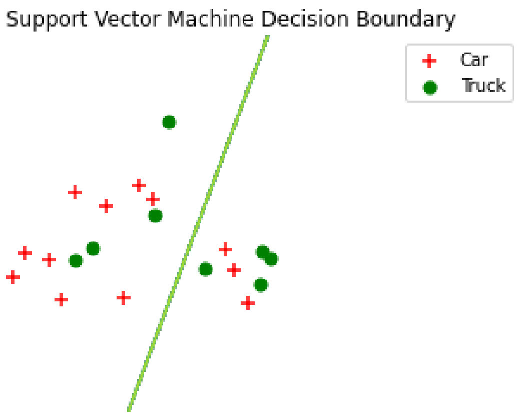


Figure 6.25: Car/Truck LinerSVM Hyperplane

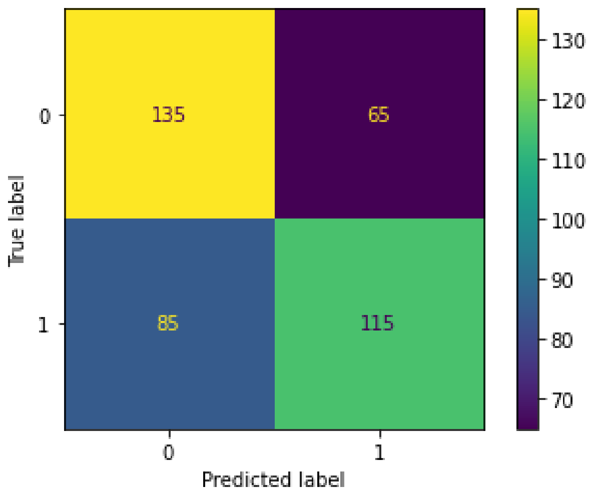


Figure 6.26: Car/Truck SVM confusion matrix

6.3 Summary

If we were to rank all the models from both datasets the leader-board would look like:

1. ResNet50
2. CNN
3. SVM
4. ANN

Plotting all the accuracies for both datasets using all four models is displayed in the bar chart below. The results are expected with ResNet50 being most powerful as we are training it with 50 layers resulting in better feature extraction. CNN is second with a reduced number of layers but still strong due to its convolutional layers etc. This is then followed by SVM and ANN which under-perform although the SVM beats CNN in the Pokemon dataset.

The SVM as mentioned performs better than the CNN it can be seem biased as we only trained the SVM with 5 classes. The decision boundary for the SVM did quite a good job at separating the images into their respective classes.

You may also be wondering why the SVM in the Pokemon dataset performed better than the car/trucks dataset which only had 2 classes. I believe this is because the images are more different within the Pokemon dataset as cars and trucks have features which are very similar. As a result, the points in the car/trucks dataset were not able to be separated and there was a lot of overlapping between the two classes.

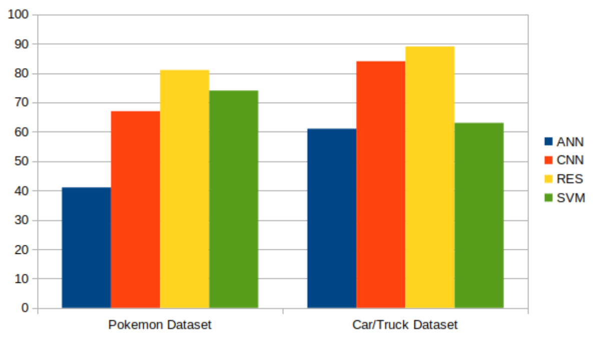


Figure 6.27: Models Accuracy Bar Chart

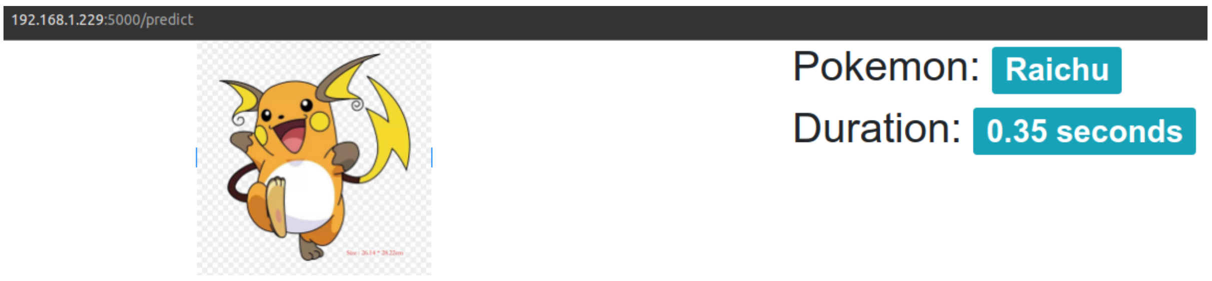
Chapter 7

Project Testing

Testing deep learning models differs from regular software development testing. Creating models is a complex task and we rely on many different functions/libraries therefore testing can be a very difficult task. Testing the functions can be done using techniques such as dependency injection however we can rely on libraries such as Tensorflow as they are widely used. The best way to test in my case is to evaluate how my models perform. In the previous section there are various methods to determine how our model is performing. These include using metrics to view the training and loss accuracy for each epoch. We can then plot these in a confusion matrix to see which classes are under-performing etc. We can even go further and find the individual images that are incorrectly classified!

Another way I tested my models especially ResNet is to extract the model and manually upload my own images to see if they ACTUALLY work. It crossed my mind that the accuracies could just be some made up number therefore if I manually test them I can actually see them classifying images they have never seen before correctly. This will make me more confident that my models are behaving as expected. To do this I extracted the model and created a simple Flask application where I am able to upload my own images and classify them. I receive the Class Name as well as how fast it classified the images.

As you can see it classifies Raichu which looks quite similar to Pikachu correctly. I have manually downloaded images and uploaded them a few times and so far they have classified correctly. Despite this, as the model is not 100% accurate there will be some that classify incorrectly.


192.168.1.229:5000/predict

Pokemon: Raichu

Duration: 0.35 seconds

Figure 7.1: Website Testing

Chapter 8

Project Evaluation

The goal of my project was to create four models using two different datasets. Fortunately, I have managed to achieve this goal as my models have been trained and the accuracy is visible outlining how each of them performed. The models can be saved and exported to create a website similar to the website created. Furthermore, others can also use my model and experiment with them if needed. All they require is to import the same libraries and have the dataset within their local drives etc. There may be some need to change filenames but all main structure is there.

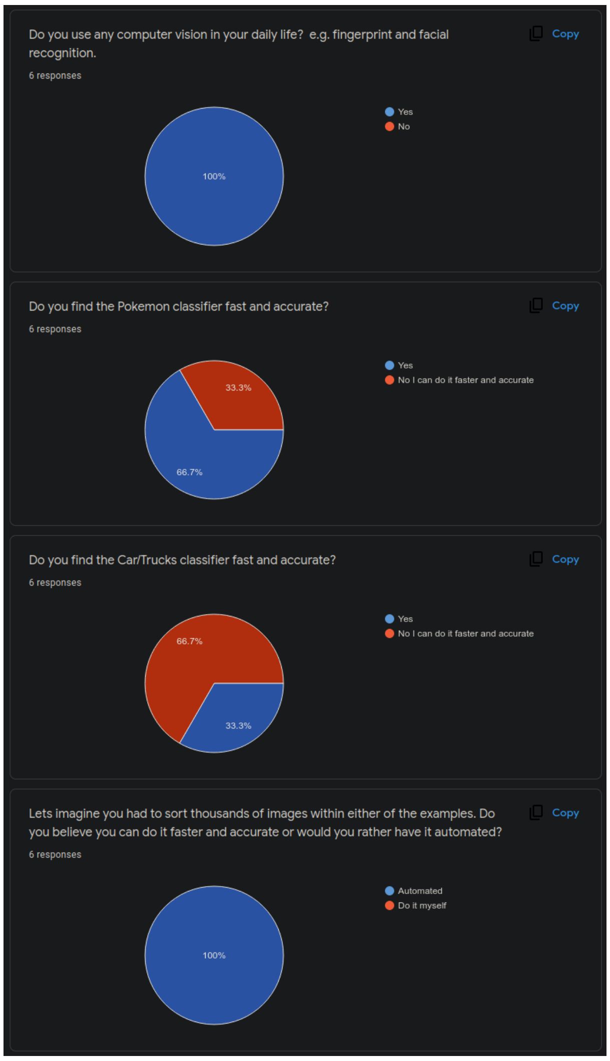
The website has been created as an addition to show the ResNet model is working and the accuracy is not just a made up number. The website does lack deployment as I ran into issues when deploying to heroku. These issues include overloading the memory quota provided as well as receiving the error "slug size is too large". This implies that my libraries such as Tensorflow or the model itself are taking up too much space. As for the memory quota error this occurs after classifying the image even if I manage to deploy the website. Despite this flaw, there are two alternatives to go around this. These include running the model within the notebook allowing users to upload their own images or run the Flask application locally. I have done both of these but prefer the Flask application for its ease of use. For deployment to work I may have been able to deploy it to a cloud service such as AWS but I believe this not to be necessary as it is out of scope for my project.

As for the models themselves there is much to be improved in terms of how fair I was with each of the models. For both datasets I attempted to keep many of the configurations the same, such as input size and train/validation/testing splits. The epochs vary to prevent over-fitting of some models and I believe this to still be fair as it shows how one model outperforms the other e.g. ResNet vs CNN.

Due to the high memory requirements of the SVM and ANN I had to severely reduce the class sizes to 5 for the Pokemon Dataset. This can be seen as biased but it was the only way to have them working to some extent. This issue continued with the Car/Trucks dataset but this time I reduced the number of images for those two models. To keep it fair as possible my SVM and ANN have a similar training/test split size. An alternative solution would be to find the number of images and classes that works across all models. In my opinion, this would limit the potential of the more powerful models CNN and ResNet preventing them from showing how capable they are.

For my final part of the project evaluation I created a small survey to ask users on their opinion of the classifier. Unfortunately, due to having to run this locally and time constraints I was only able to receive 6 responses. This is is no way an accurate representation of user opinions however it provides some insight on what people think of computer vision.

100% of the responses answered yes to whether they use computer vision. It is a big part of our lives and most likely people use some form of it within their phones. This question gave me an insight on the survey takers. Most people found the Pokemon classification fast and accurate. They maybe were not familiar with Pokemons but found the timings and correct classifications satisfactory. In addition the answers are reverse for the car/trucks classification which makes sense as most people will probably quickly determine if the vehicle is a car or truck. For my final question automation is the preferred approach as nobody would like to manually go through thousands of images to sort.


Do you use any computer vision in your daily life? e.g. fingerprint and facial recognition.
6 responses

Do you find the Pokemon classifier fast and accurate?
6 responses

Do you find the Car/Trucks classifier fast and accurate?
6 responses

Lets imagine you had to sort thousands of images within either of the examples. Do you believe you can do it faster and accurate or would you rather have it automated?
6 responses

Figure 8.1: Classification Survey

Overall, I am pleased with the results and given my research during the Literature Review the results I received were expected in terms of how each of the models performed. Testing the models and receiving feedback that they are working was an increase in confidence of my work. The accuracies I received from all the models are a good representation on the performance of each.

Chapter 9

Personal Evaluation

Overall, I am pleased with my performance on the project as a whole. I have created what I had planned to accomplish and created all four models using two datasets. This has also been presented in the form of a website. Along the duration of the project I have gathered many useful skills by understanding the technology behind deep learning and how it is used in the current society. Due to my literature review I managed to research critical aspects of my idea through understanding what others have done.

The planning stage was essential to my success as it provides me with a set approach and guideline to follow. I engaged with the special support facilities provided by Brightspace to determine which methodologies would be most useful for my project. Using the article provided within the software development methodology section I was able to research the different methodologies as well as view their advantages and disadvantages. The article was quite thorough as it provided 12 different methodologies with descriptions. (Oladele, 2020) Although I created a decent plan there were times where my performance failed to meet deadlines. For example, learning and creating some models took more time than expected. This was due to a number of reasons such as long training times and bugs that had to be fixed. I originally planned within the Gantt chart that I would have all my Pokemon classification models completed by the end of March however this went over towards the middle of April. Although this was not a huge issue it did cause concern for when I was going to finish my second dataset as I had not yet completed the first. Despite this, as soon as I finished my Pokemon models I was able to apply much of this towards the second dataset, but it did reduce the time I had to work on my report and presentation.

Further advice provided by the specialist support proved beneficial towards the early stages of my project. The "General advice on starting your FY Project" (O'Grady, n.d.) gave useful tips on what was expected of me during the project. It had an emphasis that I am working on a project and not just a product. So, rather than just completing the models and creating a simple website I had to treat it like a project from start to finish. Due to this I was able to structure my report in a way that covers all the concepts required within a project such as the literature review, design, implementation, evaluation of the project and product.

I believe that through the use of specialist support materials and applying this to my project I was able to complete it to a satisfactory standard. There were some concerns during the time scheduling of my project but I have learnt from these and will take into consideration issues that may arise during implementation.

Chapter 10

Conclusion

The evolution of deep learning is quite astonishing as we are now able to get more accurate estimates of real life events. Artificial Intelligence itself is a growing field with many of our tasks becoming automated which provides both advantages and disadvantages to our workforce. This report has gone through some of the approaches to image classification where we are able to recognise and differentiate images from each other. My report is only a fraction of what deep learning can do. There is also video classification and many other technologies that will continue to be part of our lives. It is uncertain what we may see in the future. Will human activities be detected and classified accurately in the future - detecting when criminal activity occurs? Can we have robots learn much more about how we humans operate to replace strenuous tasks? It is both an exciting but scary thought of what we could see in the future.

Bibliography

Ahmad, S. (2020). Getting very steep loss graph. https://discuss.codingblocks.com/t/getting-very-steep-loss-graph/67824/2

Aıvodji, U., Arai, H., Fortineau, O., Gambs, S., Hara, S., & Tapp, A. (2019). Fairwashing: The risk of rationalization. https://doi.org/10.48550/arXiv.1901.09749

Amor, E. (2020). Understanding non-linear activation functions in neural networks. https://medium.com/ml-cheat-sheet/understanding-non-linear-activation-functions-in-neural-networks-152f5e101eeb

Aratrika Dutta. (2021). Computer vision market predicted to reach $27.02 billion by 2028. https://www.analyticsinsight.net/the-computer-vision-market-is-predicted-to-reach-us27-02-billion-by-2028/

Boesch, G. (n.d.). Deep residual networks (resnet, resnet50) – guide in 2022. https://viso.ai/deep-learning/resnet-residual-neural-network/

Brownlee, J. (2020). Softmax activation function with python. https://machinelearningmastery.com/softmax-activation-function-with-python/

Datasciencelovers. (2020). Support vector machine-theory. http://www.datasciencelovers.com/tag/hyper-plane/

Deepanshi. (2021). Beginners guide to artificial neural network. https://www.analyticsvidhya.com/blog/2021/05/beginners-guide-to-artificial-neural-network/

GeeksforGeeks. (2022). Software engineering | sdlc v-model. https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/

Goyal, P. (2018). Why are svms memory-intensive? https://www.quora.com/Why-are-SVMs-memory-intensive

Gupta, S. (2021). Architecture of artificial neural network. https://blog.knoldus.com/architecture-of-artificial-neural-network/

Hansen, C. (2020). Neural networks from scratch. https://developer.ibm.com/articles/neural-networks-from-scratch/

Hasan, H., Shafri, H., & Habshi, M. (2019). A comparison between support vector machine (svm) and convolutional neural network (cnn) models for hyperspectral image classification. https://doi.org/10.1088/1755-1315/357/1/012035

Hayes, A. (2021). Nonlinearity. https://www.investopedia.com/terms/n/nonlinearity.asp

hmkcode. (2019). Backpropagation step by step. https://hmkcode.com/ai/backpropagation-step-by-step/

Holbrook, R., & Cook, A. (n.d.). Car or truck? https://www.kaggle.com/datasets/ryanholbrook/car-or-truck

Javatpoint. (n.d.). Agile model. https://www.javatpoint.com/software-engineering-agile-model

Kay, A. (2001). Artificial neural networks. https://www.computerworld.com/article/2591759/artificial-neural-networks

Kostadinov, S. (2019). Understanding backpropagation algorithm. https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

Kothair J.B. (2018). A case study of image classification based on deep learning using tensorflow, 3. https://www.researchgate.net/publication/345149811

Meel, V. (n.d.). Ann and cnn: Analyzing differences and similarities. https://viso.ai/deep-learning/ann-and-cnn-analyzing-differences-and-similarities/

Negrón, W.N. (2017). 10 tech issues that will impact social justice in 2017. https://www.fordfoundation.org/news-and-stories/stories/posts/10-tech-issues-that-will-impact-social-justice-in-2017/

Notley, S & Magon-Ismail, M. (2018). Examining the use of neural networks for feature extraction. https://doi.org/10.48550/arXiv.1805.02294

O'Grady, M. (n.d.). General advice on starting your fy project. https://brightspace.hud.ac.uk/d2l/le/content/190780/viewContent/1219465/View

Oladele, A. (2020). The best software development methodologies. https://medium.com/itwis/the-best-software-development-methodologies-part-1-933f37e15e1d

Recfaces. (2020). How accurate is facial recognition today? https://recfaces.com/articles/how-accurate-is-facial-recognition

Simsek, M. (2014). Why normalization is necessary in ann? https://www.researchgate.net/post/Why_normalization_is_necessary_in_ANN/54786a9bd4c11801398b4698/citation/download

techopedia. (2021). Artificial neural network (ann). https://www.techopedia.com/definition/5967/artificial-neural-network-ann

tutorialspoint. (n.d.-a). Sdlc - v-model. https://www.tutorialspoint.com/sdlc/sdlc_v_model

tutorialspoint. (n.d.-b). Sdlc - waterfall model [jpeg]. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

w3schools. (n.d.). Numpy introduction. https://www.w3schools.com/python/numpy/numpy_intro.asp

Wang, Z. J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., Kahng, M., & Polo Chau, D. H. (2020). Understanding backpropagation algorithm. https://doi.org/10.48550/arXiv.2004.15004

Zhang, L. (n.d.). 7,000 labeled pokemon. https://www.kaggle.com/datasets/lantian773030/pokemonclassification

Chapter 11

Appendices

11.1 Appendix A

UNIVERSITY OF HUDDERSFIELD
SCHOOL OF COMPUTING AND ENGINEERING

PROJECT ETHICAL REVIEW FORM

Applicable for all research, masters and undergraduate projects

Project Title:	Individual Assignment: Computer Vision for Image Classification
Student:	Wasim Ramzan (u1970064)
Course/Programme:	Computer Science (BSc)
Department:	Department of Computer Science, School of Computing and Engineering
Supervisor:	Ilias Tachmazidis
Project Start Date:	28/10/2021

ETHICAL REVIEW CHECKLIST

	Yes	No
1. Are there problems with any participant's right to remain anonymous?	☐	☒
2. Could a conflict of interest arise between a collaborating partner or funding source and the potential outcomes of the research, e.g. due to the need for confidentiality?	☐	☒
3. Will financial inducements be offered?	☐	☒
4. Will deception of participants be necessary during the research?	☐	☒
5. Does the research involve experimentation on any of the following?		
(i) animals?	☐	☒
(ii) animal tissues?	☐	☒
(iii)human tissues (including blood, fluid, skin, cell lines)?	☐	☒
6. Does the research involve participants who may be particularly vulnerable, e.g. children or adults with severe learning disabilities?	☐	☒
7. Could the research induce psychological stress or anxiety for the participants beyond that encountered in normal life?	☐	☒
8. Is it likely that the research will put any of the following at risk:		
(i) living creatures?	☐	☒
(ii) stakeholders (disregarding health and safety, which is covered by Q9)?	☐	☒
(iii)the environment?	☐	☒
(iv)the economy?	☐	☒
9. Having completed a health and safety risk assessment form and taken all reasonable practicable steps to minimise risk from the hazards identified, are the residual risks acceptable (Please attach a risk assessment form – available at the end of this document)	☒	☐

STATEMENT OF ETHICAL ISSUES AND ACTIONS

If the answer to any of the questions above is yes, or there are any other ethical issues that arise that are not covered by the checklist, then please give a summary of the ethical issues and the action that will be taken to address these in the box below. If you believe there to be no ethical issues, please enter "NONE".

NONE

STATEMENT BY THE STUDENT

I believe that the information I have given in this form on ethical issues is correct.

Signature: Wasim. B Date: 25/04/2022

AFFIRMATION BY THE SUPERVISOR

I have read this Ethical Review Checklist and I can confirm that, to the best of my understanding, the information presented by the student is correct and appropriate to allow an informed judgement on whether further ethical approval is required.

Signature: I Tachmazidis Date: 03/05/2022

SUPERVISOR RECOMMENDATION ON THE PROJECT'S ETHICAL STATUS

Having satisfied myself of the accuracy of the project ethical statement, I believe that the appropriate action is:

The project proceeds in its present form	
The project proposal needs further assessment by an Ethical Review Panel. The Supervisor will pass the form to the Ethical Review Panel Leader for consideration.	

RETENTION OF THIS FORM

- The Supervisor must retain a copy of this form until the project report/dissertation is produced.
- The student must include a copy of the form as an appendix in the report/dissertation.

OUTCOME OF THE ETHICAL REVIEW PANEL PROCESS, <u>WHERE REQUIRED</u>

	Tick One
1. Approved. The ethical issues have been adequately addressed and the project may commence.	☐
2. Approved subject to minor amendments. The required amendments are stated in the box below. The project may proceed once the form has been amended in line with the requirements and signed by the Supervisor in the box imediately below to confirm this.	☐

I confirm, as Supervisor, that the amendments required have been made:

Signature: ______________________________ Date: ____________________

3. Resubmit. The areas requiring further action are stated in the box below. The project may not proceed until the form has been resubmitted and approved.	☐
4. Reject. The reasons why it will not be possible to address the ethical issues adequately are stated in the box below.	☐

For any of the outcomes 2, 3 or 4 above, please provide a statement in the box below.

AFFIRMATION BY THE REVIEW PANEL LEADER

I approve the decision reached above by the review panel members:

Signature: ______________________________ Date: ____________________

THE UNIVERSITY OF HUDDERSFIELD: RISK ANALYSIS & MANAGEMENT

ACTIVITY: Image Classification			Name: Wasim Ramzan	
LOCATION: University of Huddersfield			Date: 25/04/2022	Review Date: 01/05/2022
Hazard(s) Identified	Details of Risk(s)	People at Risk	Risk management measures	Other comments
NONE				

11.2 Appendix B

```python
class NeuralNetwork:
  def __init__(self, input_size, layers, output_size):

    np.random.seed(0)

    params = {
        'W1':np.random.randn(input_size, layers[0]),
        'W2':np.random.randn(layers[0], layers[1]),
        'W3':np.random.randn(layers[1], output_size),
        'B1':np.zeros((1,layers[0])),
        'B2':np.zeros((1,layers[1])),
        'B3':np.zeros((1,output_size)),
    }

    self.params = params
    self.activation_outputs = None


  def forward(self, x):
    params = self.params

    W1 = params['W1']
    W2 = params['W2']
    W3 = params['W3']

    B1 = params['B1']
    B2 = params['B2']
    B3 = params['B3']

    z1 = np.dot(x,W1) + B1
    a1 = np.tanh(z1)

    z2 = np.dot(a1, W2) + B2
    a2 = np.tanh(z2)

    z3 = np.dot(a2, W3) + B3
    y_ = softmax(z3)

    self.activation_outputs = (a1,a2,y_)
    return y_

  def backward(self, x, y, learning_rate = 0.001):
    params = self.params

    W1 = params['W1']
    W2 = params['W2']
    W3 = params['W3']

    B1 = params['B1']
```

```
        B2 = params['B2']
        B3 = params['B3']

        m = x.shape[0]

        a1,a2,y_ = self.activation_outputs

        delta3 = y_ - y
        dW3 = np.dot(a2.T,delta3)
        dB3 = np.sum(delta3, axis = 0)/float(m)

        delta2 = (1-np.square(a2))*np.dot(delta3, W3.T)
        dW2 = np.dot(a1.T, delta2)
        dB2 = np.sum(delta2, axis = 0)/float(m)

        delta1 = (1-np.square(a1))*np.dot(delta2, W2.T)
        dW1 = np.dot(X.T, delta1)
        dB1 = np.sum(delta1, axis = 0)/float(m)


        params["W1"] -= learning_rate*dW1
        params['B1'] -= learning_rate*dB1

        params["W2"] -= learning_rate*dW2
        params['B2'] -= learning_rate*dB2

        params["W3"] -= learning_rate*dW3
        params['B3'] -= learning_rate*dB3

    def predict(self, x):
        y_out = self.forward(x)
        return np.argmax(y_out, axis = 1)

    def summary(self):
        params = self.params

        W1 = params['W1']
        W2 = params['W2']
        W3 = params['W3']

        a1,a2,y_ = self.activation_outputs

        print("W1 ",W1.shape)
        print("A1 ",a1.shape)

        print("W2 ",W2.shape)
        print("A2 ",a2.shape)

        print("W3 ",W3.shape)
        print("Y_ ",y_.shape)
```

11.3 Appendix C

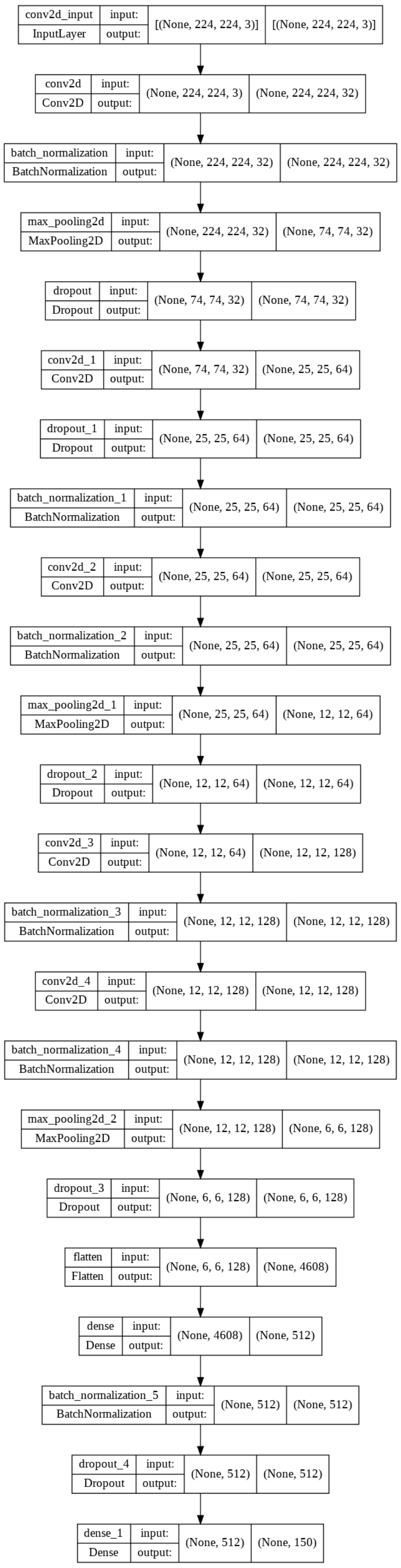

conv2d_input InputLayer input: output: [(None, 224, 224, 3)] [(None, 224, 224, 3)]
conv2d Conv2D input: output: (None, 224, 224, 3) (None, 224, 224, 32)
batch_normalization BatchNormalization input: output: (None, 224, 224, 32) (None, 224, 224, 32)
max_pooling2d MaxPooling2D input: output: (None, 224, 224, 32) (None, 74, 74, 32)
dropout Dropout input: output: (None, 74, 74, 32) (None, 74, 74, 32)
conv2d_1 Conv2D input: output: (None, 74, 74, 32) (None, 25, 25, 64)
dropout_1 Dropout input: output: (None, 25, 25, 64) (None, 25, 25, 64)
batch_normalization_1 BatchNormalization input: output: (None, 25, 25, 64) (None, 25, 25, 64)
conv2d_2 Conv2D input: output: (None, 25, 25, 64) (None, 25, 25, 64)
batch_normalization_2 BatchNormalization input: output: (None, 25, 25, 64) (None, 25, 25, 64)
max_pooling2d_1 MaxPooling2D input: output: (None, 25, 25, 64) (None, 12, 12, 64)
dropout_2 Dropout input: output: (None, 12, 12, 64) (None, 12, 12, 64)
conv2d_3 Conv2D input: output: (None, 12, 12, 64) (None, 12, 12, 128)
batch_normalization_3 BatchNormalization input: output: (None, 12, 12, 128) (None, 12, 12, 128)
conv2d_4 Conv2D input: output: (None, 12, 12, 128) (None, 12, 12, 128)
batch_normalization_4 BatchNormalization input: output: (None, 12, 12, 128) (None, 12, 12, 128)
max_pooling2d_2 MaxPooling2D input: output: (None, 12, 12, 128) (None, 6, 6, 128)
dropout_3 Dropout input: output: (None, 6, 6, 128) (None, 6, 6, 128)
flatten Flatten input: output: (None, 6, 6, 128) (None, 4608)
dense Dense input: output: (None, 4608) (None, 512)
batch_normalization_5 BatchNormalization input: output: (None, 512) (None, 512)
dropout_4 Dropout input: output: (None, 512) (None, 512)
dense_1 Dense input: output: (None, 512) (None, 150)

Figure 11.1: CNN Model Overview