Machine Reading Spider id App/Project

<Rahul Dabbeta>

<19932154>

A report submitted for

INFO7016 Postgraduate Project A

in partial fulfillment of the requirements for the degree of

<Master of Data Science>

Supervisor: <Paul Hurley>

School of Computer, Data and Mathematical Sciences
Western Sydney University

<June 2022>

Machine Reading SpiderId App/Project

Rahul Dabbeta

June, 2022

Abstract

Is it venomous? One of the most common questions asked in Australia. Home to 170 species of snakes and 2000 species of spider. Physicist, Richmand Feymann once said knowing the scientific name of an organism tells you the name but nothing else. This report investigates the feasibility of using machine learning techniques, specifically neural networks, to augment the algorithms used at the upcoming spider identification app through image processing. Algorithms are being developed to extract information from the image pixels and use them to identify the specific image. we have focused on how neural networks can be a part of these algorithms. The specific kind of neural network that has been used in the project is a convolutional neural network. CNN:s are suitable for image recognition problems, or other problems where spatial relations are important, and the data can be represented as images. As this report shows, machine learning techniques could definitely be of use for identifying the spider species accurately and we will be investigating different types of methods we can use as well.

Contents

1. Introduction

There are nearly 50,000 species of spiders in the world, and identifying a spider species is primarily based on its characteristics. It is time consuming and subjective. It is primarily determined by spider classifications. There is a need to improve the accuracy of the identifications. The lack of funding and specific personnel for conducting this research to classify and describe the spiders' vast diversity is causing two major issues. There are numerous unidentified organisms as well. Second, because identifying some specific species requires attention to detail, the accuracy of identification for those will be low. Identification of the species will be difficult for some non-specialists, and the results will be disappointing. Acquiring proficiency in this will be difficult, and the long-term benefit for non-specialists will be minimal. Now, the solutions will serve to both alleviate time constraints and make species identification easier and more accurate for everyone.

The rapid development of deep learning over the last few decades has led to its use in image recognition and other fields, but identifying or recognising spider species by clicking on an image has yet to be reported. Deep learning and neural networks are gaining popularity and are now among the most popular topics in computer science. Deep learning has a promising field of application in computer vision and object detection. CNN application for image object detection and classification. In this project, we investigate various CNN models for identifying and classifying common spider species found in Australia.

1.1. Hypothesis/question

Our objective is to identify the species of the spider through classification and using deep learning concepts. Several questions arise to achieve this such as:

• Data used for classification
• Concepts used
• Tests to find out the accuracy and its parameters.

2. Literature Review

Many researchers have conducted thorough study and contributed numerous valuable ideas that have led to the development of image classification and neural networks. We will also attempt to create an application based on the model generated by all of these methods.

Machine Learning and Artificial Intelligence advancements are regarded as critical these days. Deep learning and neural networks, in particular, have received a lot of attention. Some potential discoveries occurred as a result of these in the fields of computer vision and picture classification. Convolutional neural networks are used for picture categorisation and object detection via deep learning. We investigate various approaches for classifying and identifying spider species found in Australia. We have made significant progress in detecting photos and comprehending their commonalities during the previous few decades. It contains useful resources for understanding the fundamentals of machine learning and computer vision, as stated by Rik Das and Komal kumari (Author) in Recent Trends and Techniques in Content Based Image Classification Paperback - September 13, 2017.

2.1. Image Classification

The task of teaching a computer to recognise photographs and classify them into one of the trained categories is known as image classification. To accomplish this, we must first educate the computer to recognise a spider before teaching it to recognise a new thing. The computer gets better at identifying spiders as it encounters more of them. This is known as supervised learning. We can finish this assignment by inputting labelled photographs; the computer will start recognising patterns in spider images that aren't present in others and will start constructing its own cognition.

2.1.1. Classification using Deep Learning

Classification based on Deep Learning. Because CNN is now the go-to model for any image-related task, we employ convolutional neural networks (CNN) for deep learning categorisation. It has also been successfully applied in recommender systems, natural language processing, and other fields. The fundamental advantage of CNN over its predecessors is that it finds significant traits automatically, without the need for human participation.

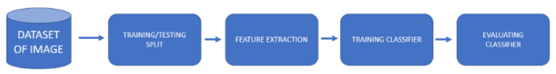
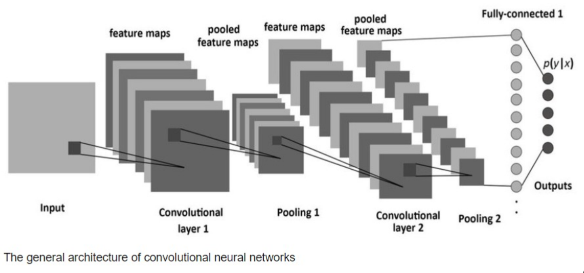


Figure 1: Image Classification

•Convolutional Neural Network

The authors of the research on Convolutional Neural Networks in Visual Computing, Raghav Venkatesan and Baoxin Li, define ConvNet as having a deep feed - forwards architecture and an exceptional capacity to generalise when compared to fully connected network layers. It is a biologically inspired concept for recognising hierarchical characteristics. CNN is prefered over other traditional models for several reasons. First, it employs the concept of weight sharing, which reduces the number of parameters that must be taught significantly, resulting in smooth training and no overfitting. Following that, the model's classification stage is fixed with feature extraction, both employing the learning process. Third, whereas some generic models in Artificial Neural Networks make it difficult to create large networks, CNN excels at it. Convolutional Neural Networks are frequently employed due to their exceptional performance in areas such as face recognition, picture classification, diabetic retinopathy, facial expression detection, and many more.



/Co

Figure 2: Convolutional Network

- Architecture of CNN

1.The input layer:

The input layer serves as a portal to the convolutional neural networks. This layer is where the inputs are routed. The dimensions of the input photos are supplied to the input layer as parameters. The matrix format has image dimensions of width x height x depth, where depth refers to colour channels such as RGB, grey, and so on. The input layer values that are seen by 2 are commonly used, which includes 32, 64, 128 and so on.

2.Convolution Layer:

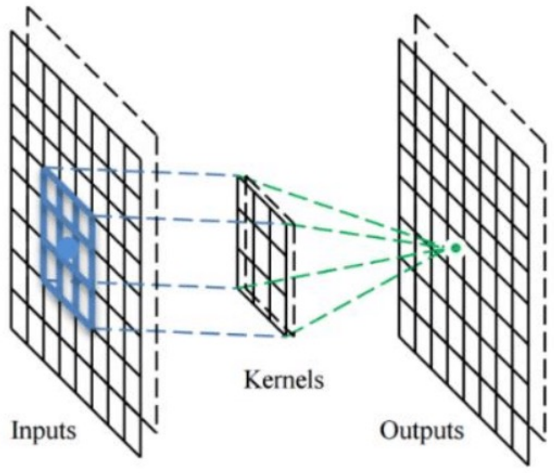
The primary goal of convolution layers is to extract features from image input. Convolution layers accept input images in chunks to reduce processing resources required for feature extraction. In neural networks, each layer's output serves as the input for the subsequent layers. The dimensions are filtered once the dot product is applied to the input fields. The multi integer inputs are turned into single valued integers in this procedure. The filter is then dragged over the next values of the matrix of the same image input to compute the dot product. This process is repeated until the entire image has been covered.

A few matrixes are employed in the extraction and detection of features from input images.

2.1 Filter: In convolution nets, filters are used to discover patterns like edges in images by detecting changes in intensity values in the images. In general, the high frequency section of an image is where the pixel's intensity chances at high values (Kumar, 2020). When the frequency is low, the pixel intensity is uniform. In most edge detection filters, the brightest pixels are on the left and the darkest pixels are on the right. Back propagation is used by the filters to learn throughout the training phase.

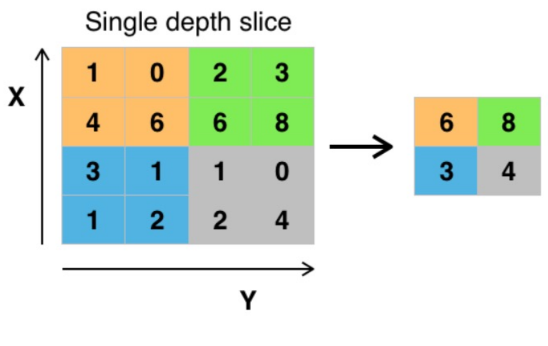
2.2 Padding: In convolution, padding is used to manage edge pixels. The amount of pixels required by the kernel to process the edge pixels is added around the perimeter of the image in this padding. One pixel is added outside for a 3 X 3 kernel. In general, edge pixels are ignored in many problems, although this results in information loss.

2.3 Kernels: Convolution layers contain filters known as convolution kernels. Filters are integer matrices that are applied to a collection of input pixels. The pixel is multiplied by the kernel value, and the outputs are combined to generate a single value that looks like a pixel.



2.4 Strides: Convolution layers, in general, use stride 2 rather than stride 1 where the kernel glides over two pixels at a time.

3.Pooling layer: In convolution neural nets, the pooling layer is utilised to minimise spatial dimension by combining multiple functions. This reduction in dimensions reduces computation power and controls model overfitting, which is the primary goal of convolution nets. Maxpooling, average pooling, and L2 regularisation are the various functions employed in the reductions, with maxpooling being the most widely utilised.



The figure above (Convolutional Neural Networks (CNN, or ConvNets), 2017) depicts a max-pooling function with a 2 x 2 filter and a 2stride.

4. The ReLU layer:

Rectified Linear Unit (ReLU) is a nonlinear activation function used in convolution networks that preserves values between 0 and 1, and converts values less than 0 to zeros.

5.Normalisation in batches:

Batch normalisation is one of the most significant layers in convolutional neural networks. It aids in the stabilisation of the network's output in predictions. Regularization is followed by batch normalisation to reduce overfitting and speed up training. Within the purview of the activation layer, normalisation is the process of subtracting the mean of the batch's activation function and dividing it by the standard deviation. This process is required because the activations of some neurones will remain high even after normalising the inputs, causing behavioural difficulties and reducing network stability.

6.Fully Connected Layer:

Every neurone in the layer is coupled to every neurone in the subsequent layers in fully connected layers. The softmax function is employed as an activation function in this layer to activate neurones and classify picture features into distinct groups based on the labels in the training data.

7.Learning Transfer:

Transfer learning is a deep learning technique in which a model that has been trained to solve a task can be used to solve similar problems. Transfer learning problems are frequently utilised to solve predictive modelling assignments. A model trained on massive amounts of data is employed for these jobs.

The pre-trained model can be downloaded directly and utilised for feature extraction for transfer learning. When using these pre-trained models, the model's output is supplied as input to a classifier model. Instead of implementing the entire model, individual model layers can be used in the new model by freezing the weights to prevent the weights from updating while the new model is being trained. VGG, GoogLeNet, and Residual Network are prominent transfer learning models.

2.1.2. Classification using Support Vector Machine

SVM can also be used to classify photos. Deep learning is more appropriate for our project, however we will explore it more to gain a better understanding of which classifier is ideal for our project.

The paper's authors, Images as Data in Social Science Research Described by Nora Webb Williams, Andreu Casas, and John D. Wilkerson SVM image classification is accomplished by generating a Bag of words (BoW) in Python. A histogram of visual word occurrences that represent an image is generated by the process. These histograms are used to train a classifier for image categories.

Preparing a Dataset: A Dataset folder is made up of a certain collection of sub-folders. Each subfolder is named after the category and contains only photographs from that category. Creating a Bag of Features (Words): By interpreting image features as words, the bag of words model is applied to image classification. The following steps are required to represent an image using this approach.

1.Feature Representation: Each image is abstracted by numerous local patches after feature detection. Approaches for representing patches as numerical vectors are dealt with in feature representation methods. These vectors are referred to as feature descriptors. To some extent, a decent descriptor should be able to handle intensity, rotation, scale, and affine variations. Scale Invariant Feature Transform is a well-known descriptor (SIFT). Each patch is converted to a 128-dimensional vector by SIFT. Each image is now a collection of vectors of the same dimension (128 for SIFT). The key stages of generating the set of image features are as follows (using SIFT).

- scale-space extream detection.
- keypoint localization
- Orientaton assignment
- keypoint descriptor

2.2. Building an application using the models

There are few applications for detecting spider species. We use quantization to turn the learned model into a tf light model. It shrinks the model's size so that it takes up less space.

We will deploy them into Google Cloud Platform and develop Google Cloud Functions after converting them to tf light models. These features are useful for our mobile application. It produces react native, a hybrid mobile app development framework.

2.2.1. Quantization

The process of decreasing the precision of weights, biases, and activations such that they consume less memory is known as quantization.

In other words, quantization is the process of converting a neural network, which typically employs 32-bit floats to store parameters, to a smaller representation, such as 8-bit integers.

Going from 32-bit to 8-bit, for example, would result in a fourfold reduction in model size, so one clear benefit of quantization is a huge reduction in memory.

2.2.2. React Native js

React Native is a JavaScript framework for creating natively rendered mobile apps for iOS and Android. It's built on React, Facebook's JavaScript toolkit for creating user interfaces, but instead of being aimed at browsers, it's aimed at mobile platforms. In other words, web developers can now create mobile applications that look and feel fully "native," all while using the familiar and beloved JavaScript library. Furthermore, because most of the code you create can be shared across platforms, React Native makes it simple to develop for both Android and iOS at the same time.

3. Models and software used

3.1. Software

1.TensorFlow: 1.TensorFlow: TF is utilised in some of Google's commercial products, like as their speech recognition API and Gmail, but it is also used for research. The framework can be used as a backend in C++ and as a frontend in Python. The numerous built-in functionalities are one of the benefits of utilising TF. We're utilising high-level API Estimators that can be adjusted if necessary. We created our own estimator with a specific model for this job. The TF built-in functions tf.layers and tf.losses are used in this model to define the structure and loss function of our neural network.

2.Loss function: In the tf.losses module, we employed the softmax cross entropy function. Weights, scope, reduction, and smoothing are among the loss function parameters. Weights can be employed to steer the network towards a specific class, for example, to combat overfitting. The optimisation method is also a network parameter.

3.2. Preprocessing of data and models

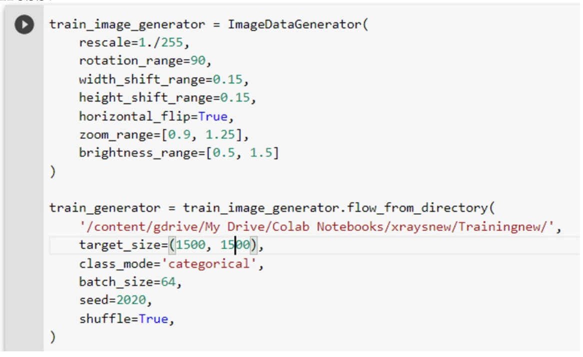
3.2.1. preprocessing the data

We partition the dataset into training, testing, and validation datasets after loading it. The model is trained using training data. When verifying the model, validating data is used. Finally, the testing dataset is used as unseen data for the model during model testing.

Following data setup, the data is pre-processed and readied for feeding the model. We adjust the photographs in pre-processing because they were captured by different devices and have varied dimensions. This inconsistency in the dimensions can lead the model to behave strangely. To avoid this, rescale or resize the photos to equivalent dimensions. High-end dimensions necessitate more computational power when processing photos and modelling. The image size is lowered to reduce computing power.

3.2.2. Data Augmentation

Data argumentation is conducted on the photos after they have been resized using the pre-defined libraries in Keras. ImageDataGenerator. Data argumentation is a deep learning strategy that aids in expanding the number of images used for training models without recording new data. Data argumentation enables practitioners to employ a diverse set of parameters. The parameters employed in this thesis are rescaling, rotation range, width shift, height shift, horizontal flip, zoom, and brightness.



```
train_image_generator = ImageDataGenerator(
    rescale=1./255,
    rotation_range=90,
    width_shift_range=0.15,
    height_shift_range=0.15,
    horizontal_flip=True,
    zoom_range=[0.9, 1.25],
    brightness_range=[0.5, 1.5]
)

train_generator = train_image_generator.flow_from_directory(
    '/content/gdrive/My Drive/Colab Notebooks/xraysnew/Trainingnew/',
    target_size=(1500, 1500),
    class_mode='categorical',
    batch_size=64,
    seed=2020,
    shuffle=True,
)
```

Figure 3: Training image generator

Image Data Generator's arguments are as follows:

Rescale: In general, rescale has a default value of None or 0 which signifies 'no scaling.' Apart from no scaling, the data must be multiplied with a value. After applying several modifications, the data is multiplied by 225 in the code above (Brownlee, 2019).

Rotation range: Rotation accepts the numbers that should be used to rotate the image clockwise from 0 to 360o. The image is rotated 90 degrees in this case.

Width shift: This causes the image to be shifted by the supplied value along the width dimension. In this case, the photos' width dimensions are displaced by 0.15 units.

Height shift: This causes the image to be shifted by the supplied value along the height dimension. In this case, the photos' height measurements are displaced by 0.15 units.

Horizontal flip: This flips the image horizontally. This parameter has two possible values: 'True' or 'False.' The passed value in this case is 'True.'

Zoom range: The zooming nature of the image can be controlled by the zoom range option. Zoom range uses the 'Zoom in' and 'Zoom out' settings. Zoom in and Zoom out values for this problem are 0.9 and 1.25, respectively.

Brightness range: This parameter is used to argue the image brightness levels. The brightness range is determined by two inputs: the minimum and maximum brightness ranges. Values less than 1.0 darken the image, whereas values greater than 1.0 brighten it. The lowest and maximum brightness parameters for this study are 0.5 and 1.5.

The following are the 'flow from directory' arguments:

Directory: To load data, a string with the link to the target directory where the data is located is used. The link to the Google Drive is supplied in the above-mentioned code to retrieve the material from the target documentary.

Classes: Argument classes are used to define label names. If the data is partitioned into subdirectories with the name of the category as the directory name, then the data classes can be accessed directly by the names of those sub-directories. If the data is not divided into subdirectories, it must be organised class by class. To be more specific, this condition is divided into three categories: Covid19, normal, and pneumonia. Now, all of the photographs of covid19 appear first, followed by normal images, and finally, images of pneumonia.

Class mode: This option specifies the type of label that the generator use. Categorical is the default value for class mode. The labels utilised include category, binary, sparse, and none. The number of classes in the problem can be calculated based on the type of classes. Categorical has several classes. There are two types of binary mode. Sparse returns a single 1D integer (Image Data Generators in Keras, 2019).

Batch size: The integer values in the argument batch size. The value represents the number of photos in a batch. The batch size for this challenge is 64, which means that each batch has 64 photos for quick processing.

The target size is a tuple of integers containing the image's height and width, to which the original image size is reduced. The height X width values are set to 256 x 256 by default. Because the original size of the photographs are roughly 2000 $\times$ 2000, the dimensions of height x breadth are decreased to 1500 x 1500 for this challenge. Too much decrease in dimensions may result in information loss.

The Shuffle argument uses the Boolean values 'True' and 'False'. The value is set to 'False' by default. For this issue, the value is 'true,' which permits the shuffling of data pictures; otherwise, the shuffle is disabled.

3.3. models

The image processing model utilised in this study is a multi-classification model that classifies images into numerous classes labelled spider species.

Convolution layers, maxpooling layers, dropout layers, batch normalisation layer, input layer, and output layer comprise the model's overall architecture. The mode's architecture is shown below, and it includes information about the type of layer utilised, the shape of output each layer has, and the total learnable parameters a layer has.

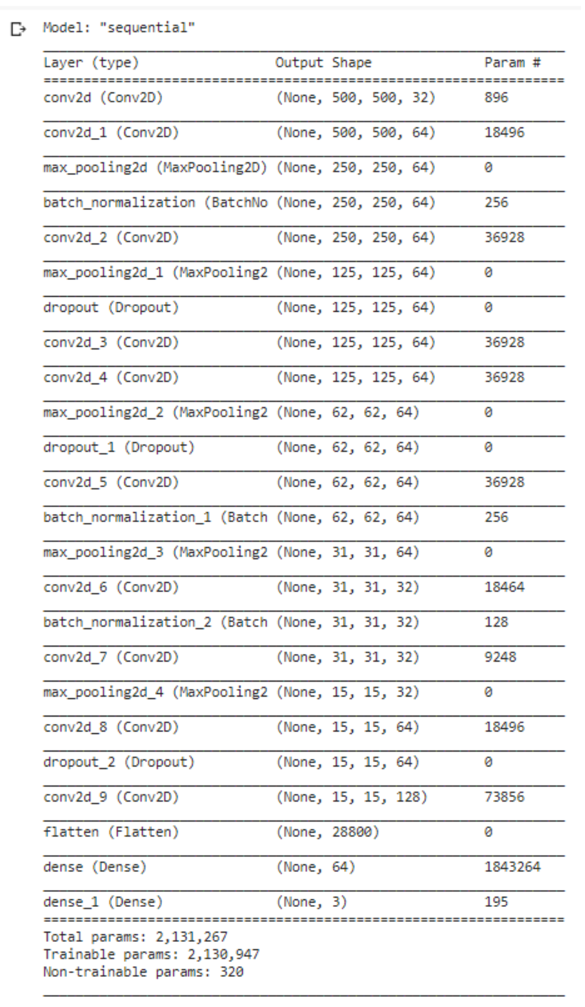
By adding the bias to the product of input and output, the number of learning parameters may be computed.

The input layer for the model is the input shape. The pixel size is lowered, and the colours utilised in the images can be any number. The layer's activation size is now calculated by multiplying the dimensions of the inputs.

The input layer accepts image input with the goal size, the number of filters applied, and the shape of the inputs. There are no learnable parameters in the input layers. Following the input layer are convolution layers, maxpooling layers, and dropouts.

(number of filters * (shape of filter width*height of filter width*filters in previous layer+1) can be used to get the parameter for the convolution layers. Because the pooling, drop out, and flatten layers do not have learnable parameters, the total number of parameters is zero.

The network's final layers are the dense layers, which contain the output layer. The number of neurones in the output layer is determined by the classification labels; in this case, the classification is between many labels, namely spider species; thus, the neurones' values are activated using the Soft max activation function.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 500, 500, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 500, 500, 64)      18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 250, 250, 64)      0
_________________________________________________________________
batch_normalization (BatchNo (None, 250, 250, 64)      256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 250, 250, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 125, 125, 64)      0
_________________________________________________________________
dropout (Dropout)            (None, 125, 125, 64)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 125, 125, 64)      36928
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 125, 125, 64)      36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 62, 62, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 62, 62, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 62, 62, 64)        36928
_________________________________________________________________
batch_normalization_1 (Batch (None, 62, 62, 64)        256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 31, 31, 32)        18464
_________________________________________________________________
batch_normalization_2 (Batch (None, 31, 31, 32)        128
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 31, 31, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 15, 15, 64)        18496
_________________________________________________________________
dropout_2 (Dropout)          (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 15, 15, 128)       73856
_________________________________________________________________
flatten (Flatten)            (None, 28800)             0
_________________________________________________________________
dense (Dense)                (None, 64)                1843264
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 2,131,267
Trainable params: 2,130,947
Non-trainable params: 320
_________________________________________________________________
```

Figure 4: Example of model output

Fittiing to the model: The model is trained utilising a 'categorical crossentropy' loss function and a 'Adam' optimiser to calculate accuracy. The models are trained across a large number of epochs.

3.3.1. Environment setup

We will utilise Google Colab to model the networks and train the deep learning models, which operates as a virtual environment by providing virtual GPUs, TPUs, and memory. To collaborate with the deep learning

Because training and processing a model need significant computational capacities, there is a requirement for high-end architecture that aids in quick processing.

Python was used to create the models, which included importing pre-defined libraries from the Tensorflow and Keras packages. Tensorflow has a large number of libraries that can be utilised at various stages of model development and training. The implementation of models and activation functions has been made simple with the help of tenosrflow and other sub-libraries by simply calling the methods rather than manually writing.

4. Gathering the data

Data sets are essential when working with Artificial Intelligence problems that require training and testing models to effectively forecast results. On the internet, there are no particular data collections of spider photos. In this project, I plan to aggregate datasets from public domains. Kaggle is one of the places where the majority of the datasets can be found. I am gathering data that is maintained and published by several writers, including high-quality photos and labels. Also, to boost the amount of photos, integrate the different datasets into one. One of the data sets I'm using is from Kaggle. YIKES! Spiders are classified into 15 species. There are two columns

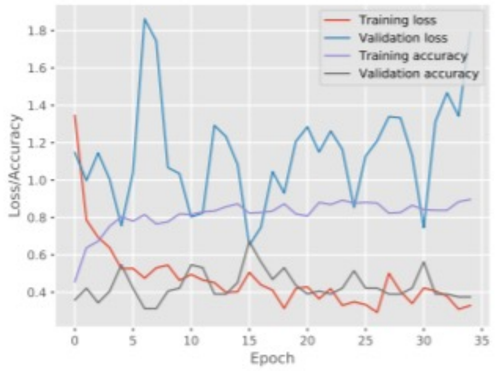
The data set's fields are as follows:

- class id
- spider species

We later divide them into training , testing and validating datasets.

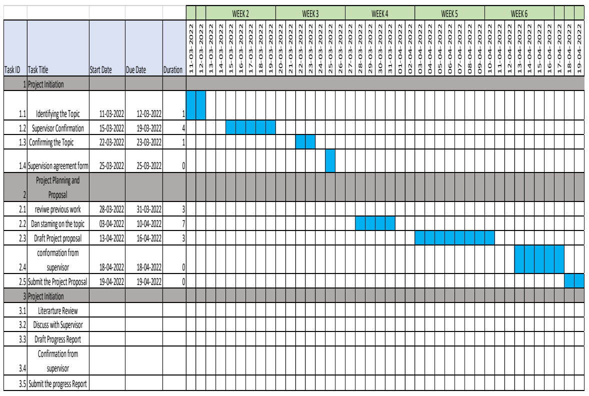
5. Accuracy test for a trained model

A few performance classification matrixes values must be examined after constructing the models and evaluating their performance. In this project, we should think about precision and recall when classifying the labels'spider species.' We can determine whether model has more accuracy and precision by retrieving information from the table. If the model is properly trained, the convolution neural network model maxpooling layers will achieve higher accuracy.



Model can be further developed by adding more layers of Convnets, dropping high frequency neurons and following the techniques to control overfitting of the model.

Program of work



							WEEK 2							WEEK 3							WEEK 4							WEEK 5							WEEK 6									
Task ID	Task Title	Start Date	Due Date	Duration	11-03-2022	12-03-2022	13-03-2022	14-03-2022	15-03-2022	16-03-2022	17-03-2022	18-03-2022	19-03-2022	20-03-2022	21-03-2022	22-03-2022	23-03-2022	24-03-2022	25-03-2022	26-03-2022	27-03-2022	28-03-2022	29-03-2022	30-03-2022	31-03-2022	01-04-2022	02-04-2022	03-04-2022	04-04-2022	05-04-2022	06-04-2022	07-04-2022	08-04-2022	09-04-2022	10-04-2022	11-04-2022	12-04-2022	13-04-2022	14-04-2022	15-04-2022	16-04-2022	17-04-2022	18-04-2022	19-04-2022
1	Project Initiation																																											
1.1	Identifying the Topic	11-03-2022	12-03-2022	1	🟦	🟦																																						
1.2	Supervisor Confirmation	15-03-2022	19-03-2022	4					🟦	🟦	🟦	🟦	🟦																															
1.3	Confirming the Topic	22-03-2022	23-03-2022	1												🟦	🟦																											
1.4	Supervision agreement form	25-03-2022	25-03-2022	0															🟦																									
2	Project Planning and Proposal																																											
2.1	reviwe previous work	28-03-2022	31-03-2022	3																																								
2.2	Dan staming on the topic	03-04-2022	10-04-2022	7																		🟦	🟦	🟦	🟦																			
2.3	Draft Project proposal	13-04-2022	16-04-2022	3																								🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦									
2.4	conformation from supervisor	18-04-2022	18-04-2022	0																																		🟦	🟦	🟦	🟦	🟦		
2.5	Submit the Project Proposal	19-04-2022	19-04-2022	0																																							🟦	🟦
3	Project Initiation																																											
3.1	Literarture Review																																											
3.2	Discuss with Supervisor																																											
3.3	Draft Progress Report																																											
3.4	Confirmation from supervisor																																											
3.5	Submit the progress Report																																											

Figure 5: Research Plan

Acknowledgements

I would like to take time and express my gratitude and appreciation to all who gave me this wonderful platform to complete the report. A special thanks to my mentor, Mr. Paul Hurley of the school of Computing, Engineering and Mathematics at the Western Sydney University for continuous suggestions and encouragement throughout this unit and especially in writing this report. I would like to thank my parents without which none of this would have been possible for me. I am also thankful to my friends who have been continuously assisting me throughout my time in Sydney and been with me during thick and thin.

Reference

1. Lindsay Stuart, 2018. [online] Available at: ¡ https://medium.com/noun-project/identifying-australian-spiders-with-icons-f03897f33ab6¿

2. Alison DonnellanJuly. [online] 7th, 2020 Available at: ¡ https://algorithm.data61.csiro.au/this-ai-powered-platform-is-a-shazam-for-insect-and-snake-identification/¿

3. G. M. Nicholson, R. Walsh, M. J. Little, and M. I. Tyler, 'Characterisation of the effects of robustoxin, the lethal neurotoxin from the Sydney funnel-web spider Atrax robustus, on sodium channel activation and inactivation,' Pflügers Archiv European Journal of Physiology, vol. 436, no. 1, pp. 117–126, 1998