A picture containing graphical user interface

Description automatically generated **Configuration Manual**

MSc Research Proposal

Research in Computing

Akshen Doke

Student ID: x18191592

School of Computing

National College of Ireland

Supervisor: - Prof. Rashmi Gupta

A picture containing text, clipart

Description automatically generated

**National College of Ireland**

**MSc Project Configuration Manual**

**School of Computing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name:** | Akshen Doke | | |
| **Student ID:** | x18191592 | | |
| **Programme:** | MSc in Data Analytics | **Year:** | 2019-2020 |
| **Module:** | Research Project | | |
| **Supervisor:** | Prof. Rashmi Gupta | | |
| **Submission Due Date:** | 17th December 2020. | | |
| **Project Title:** | Image Classification: Using Ensemble method to classify covid19, normal and pneumonia x-ray images. | | |
| **Word Count:**1335 | **Page Count** 20 | | |
|  |  | | |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

|  |  |
| --- | --- |
| **Signature:** | ……………………………………………………………………………………………………………… |
| **Date:** | 17h December 2020 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

|  |  |
| --- | --- |
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

|  |  |
| --- | --- |
| **Office Use Only** | |
| Signature: |  |
| Date: |  |
| Penalty Applied (if applicable): |  |

1. **Hardware and Software Requirements**

For this project, all compute intensive tasks like modelling, data visualization and prediction was done on a cloud service called Google Colab which was accessed using a MacBook Air. Only the data downloaded from data source was converted from jpeg to png and was renamed on local device (MacBook Air) using bash program before uploading it to the cloud.

**Cloud Setup Option**

|  |  |
| --- | --- |
| Processor | page13image60933248page13image38216000  On-demand |
| Graphic Card | page13image60947072page13image60947456  TPU and GPU option available  page13image60948032page13image60937856 |
| RAM | Min 8Gb-Max 32GB |
| HDD | 12GB free space |

Bash scripts for data format changing and renaming.

Graphical user interface, text, application

Description automatically generated

**FIGURE 1: Bash Scripts**

1. **Google Collaboratory (Colab) Setup**

Since this research was carried out using Google Colab’s Cloud infrastructure, we need to first upload our dataset to Google drive which can be connected to our notebook (code platform of colab) were we are going to code and use the data.

Graphical user interface, application

Description automatically generated

**FIGURE 2: Google Drive**

We create three folders, one in which we are going to store our training data, second our test data and third for the models on which the training is going to be happening.

As mentioned in (Google, n.d.) Google Colab is an Infrastructure and Software as a Service free to use provided by Google for tasks related to machine learning, data analytics and artificial intelligence in python and its related libraries.

List of libraries and packages used

* Python 3.6.9
* Keras 2.4.0
* Matplotlib
* os
* tensorflow
* sklearn, numpy

To mount the drive to our notebook we use the code given below

Graphical user interface, application

Description automatically generated

**FIGURE 3: G-Drive mounting**

After our drive is mounted successfully we can set paths for our train and test files, also import the required libraries and functions for our project.

Text

Description automatically generated

**FIGURE 4: importing required libraries and functions**

To get maximum speed and utilization of our notebook we change our runtime to GPU from None, this will make our program execution faster while we train and run our predictions on the dataset.

Graphical user interface, application

Description automatically generated

**FIGURE 5: Setting Notebook runtime to GPU**

**3. Data Preparation and Visualization**

Next, we set paths for our train and test datasets in the respect variable names.

Text

Description automatically generated

**FIGURE 6: Setting path to variables**

Now we need to calculate the overall count of each set of images and represent it visually for that we use python based library called matplotlib

Text

Description automatically generated

**FIGURE 7: Counting datasets and plotting**

The output of Data spread which we get is

Chart, pie chart

Description automatically generated

**FIGURE 8: Pie plot of Dataset**

As we can see the count of covid is relatively low, in order to balance this out we would be using data augmentation techniques while training our model.

1. **Implementation of Models**

Since we are going to make use of ensemble methods for prediction, we would be training around 7 models using which we would be performing the ensemble based prediction.

For the first 5 models, we would be using transfer learning methodology via which a previously trained/optimized model on a large dataset can be inherited and reutilized on other datasets, the advantage of using such a method is that since these models are trained and optimized on large and complex datasets, their architecture can quickly adapt to most of the image datasets and reduce the huge time overhead of creating a convolutional neural network from scratch.

Keras package has numerous such models which can be inherited via transfer learning and reused.

**Image Augmentation and rescaling**

Certain methods would be common throughout the model training process like image rescaling and augmentation which is shown below

Text

Description automatically generated

**FIGURE 9: Data Augmentation**

**Common Packages and libraries**

**Text

Description automatically generated**

**FIGURE 10: Other imports**

* 1. DenseNet201

Below is the code for implementation of DenseNet201 model which we import from keras package and train our dataset on.

Text

Description automatically generated

**FIGURE 11: Building the DenseNet Model**

Once the model is build and compiled, we begin the training process, we can optimize the parameters while training our model in order to get better output.

Text

Description automatically generated

**FIGURE 12: Training and saving the densenet model.**

We also save and download the model which we will be using later on for our ensemble of models. Here on, same steps would be repeated for all the models mentioned below.

* 1. VGG 16

Text

Description automatically generated

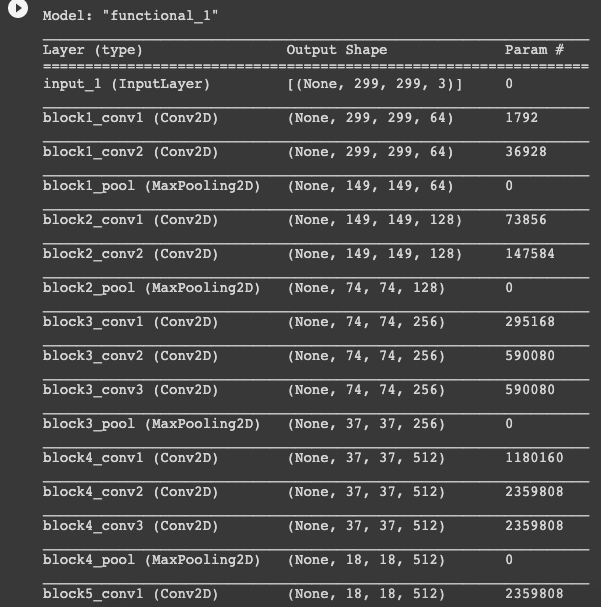
**FIGURE 13: Building the VGG 16 Model**

Text

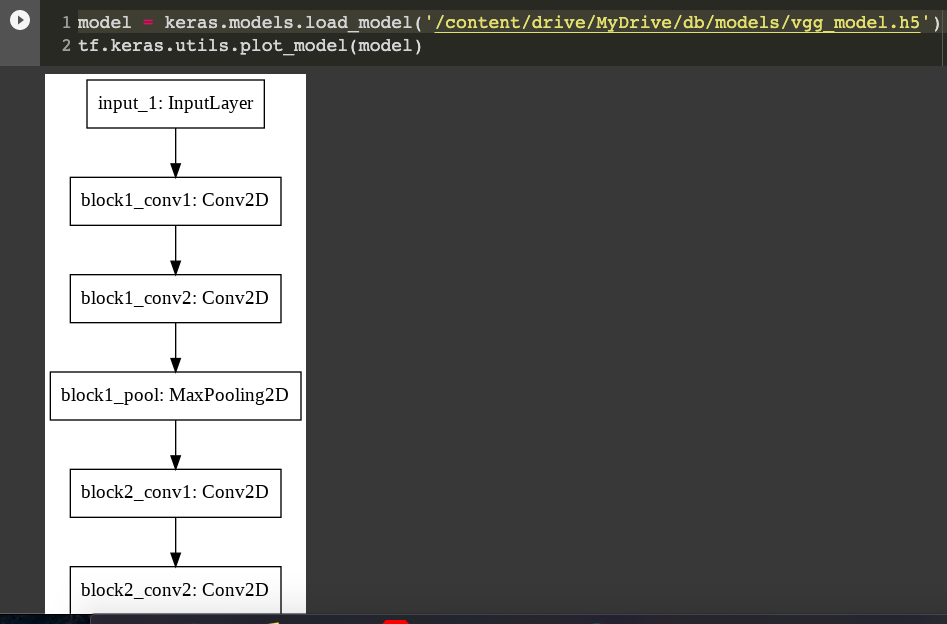
Description automatically generated

F**IGURE 14: Training and Saving Model**

Some other features worth mentioning which can help us improve the performance and accuracy of our models is that we can take a peek in to the model architecture by using a built in method called `model.summary()` which summarizes the architecture of the model in our case VGG16 in a textual format and another function which gives a plot of our layer stack is `tf.keras.utils.plot\_model(model)` output of both functions is given below.



**FIGURE 15: Summary of VGG 16**



**FIGURE 16: Architecture Plot for VGG16 Model**

Also, we have another technique to see the output of the prediction layers by plotting a heatmap around the input image. This technique is called “Grad-CAM”

And the code and output for it is given below

Text

Description automatically generated

**FIGURE 17: GRAD-CAM settings**

Text

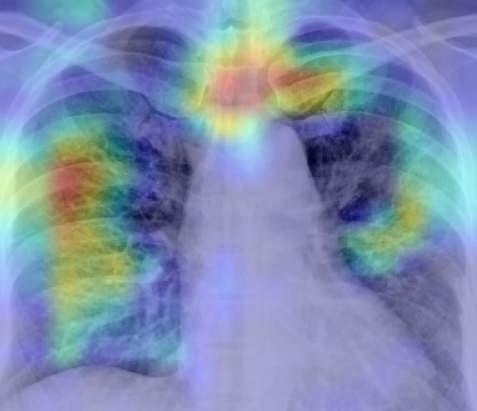
Description automatically generated

**FIGURE 18: GRAD-CAM algorithm implementation**

Graphical user interface, text

Description automatically generated

**FIGURE 19: GRAD-CAM HeatMap**



**FIGURE 20:HeatMap Over Image**

This technique can be applied on individual models but can’t be implemented on the overall output of the ensemble networks which we are going to create.

* 1. NasNet

Text

Description automatically generated

**FIGURE 21: Building NASNET Model**

Text

Description automatically generated

**FIGURE 22: Training and saving Nasnet**

* 1. Xception

Text

Description automatically generated

**FIGURE 23: Building Xception Model**

Text

Description automatically generated

**FIGURE 24: Compiling and Training Xception Model**

* 1. Resnet

Text

Description automatically generated

**FIGURE 25: Building a Resnet**

Text

Description automatically generated

**FIGURE 26: Training of Resnet Model**

* 1. MyModel

In case of this model, we create it from scratch and train it on our data, the performance of this model was close to 90 % similar to our other models but since it is only trained on our dataset, the overall performance in comparison to other models might differ when other datasets are taken into consideration.

Text

Description automatically generated

**FIGURE 27: Building the custom model**

Text

Description automatically generated

**FIGURE 28: Training the Model**

* 1. AlexNet

Text

Description automatically generated

Text

Description automatically generated

**FIGURE 29: Building Alexnet Model from scratch**

Text

Description automatically generated

**FIGURE 30: Training Alexnet Model**

1. Implementation and Evaluation of Ensemble Networks.

Ensemble is a collection of the above mentioned models, the input image is given to each model and output of each is stored in a list and the majority is regarded as the final outcome for that input Image. Here we implement two techniques of ensemble networks first one is based on voting and second one is based on weighted voting.

Each model created above has its own function within which we import the trained model for that type and pass on our data to it which then returns output for the same.

**Example: Resnet Function**

Text

Description automatically generated

**FIGURE 31: Function for Resnet loading and prediction**

We call all our defined functions and save their output in respective variables.

Text

Description automatically generated

**FIGURE 32: Calling models**

Then we merge them in a list and for each input we calculate the prediction based on voting and weighted voting algorithm.

Note: We’ve passed the complete directory of our test data instead of a single image in order to evaluate the ensembles properly.

Text

Description automatically generated

**FIGURE 33: Creating Ensembles of Model**

**Evaluation**

For our ensemble based on voting we get the following metrics

Text

Description automatically generated

**FIGURE 34: Evaluation Metrics for Voting Based Ensemble Network**

For ensemble based on weight increment, we get the following output

Text

Description automatically generated

**FIGURE 35: Metrics for Weighted Voting Based Ensemble Network**

All the code mentioned in the screenshots above are provided with the report for this project.

# Works Cited

Google, n.d. *Collaboratory : - Frequently Asked Questions.* [Online]   
Available at: https://research.google.com/colaboratory/faq.html  
[Accessed 8 December 2020].

Chollet, F., 2020. *Grad-CAM class activation visualization.* [Online]   
Available at: https://keras.io/examples/vision/grad\_cam/  
[Accessed 10 December 2020].

Kumar, D. V., 2020. *Hands-on Guide To Implementing AlexNet With Keras For Multi-Class Image Classification.* [Online]   
Available at: https://analyticsindiamag.com/hands-on-guide-to-implementing-alexnet-with-keras-for-multi-class-image-classification/  
[Accessed 10 November 2020].