National College of Ireland

Phishing Detection Using Convolutional Neural Network and ADADELTA

Configuration Manual

MSc Internship
MSc Cyber Security

Tejas Umakant Phade
Student ID: 18195709

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Tejas Umakant Phade
Student ID:	18195709
Programme:	MSc Cyber Security
Year:	2019-2020
Module:	MSc Internship
Supervisor:	Niall Heffernan
Submission Due Date:	17-08-2020
Project Title:	Phishing Detection Using Convolutional Neural Network and ADADELTA - Configuration Manual
Word Count:	1176
Page Count:	13

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.

I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature:	Tejas Umakant Phade
Date:	14th August 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):	

Phishing Detection Using Convolutional Neural Network and ADADELTA

Tejas Umakant Phade
18195709

1 Introduction

This paper presents the configuration manual demonstrating the walkthrough of implementation stages involved in our research "Phishing Detection Using Convolutional Neural Network and ADADELTA". The aim of this study is to develop a solution for detection of Phishing login pages. Building the artefact consisted of combining two techniques 'Convolutional Neural Network' and 'ADADELTA'. In Section 2 of the configuration manual, the hardware and software project specifications are elaborated. The Data Collection steps are explained in Section 3 followed by Training and Deployment in Section 4.

2 System Specification

Since the artefact is an extension of *Google-Chrome*, hardware configuration to train and test the platform is important. This solution has been developed on the following pillars:

2.1 Hardware Configuration

The artefact has been developed on a laptop having hardware configuration described in the Figure 1:

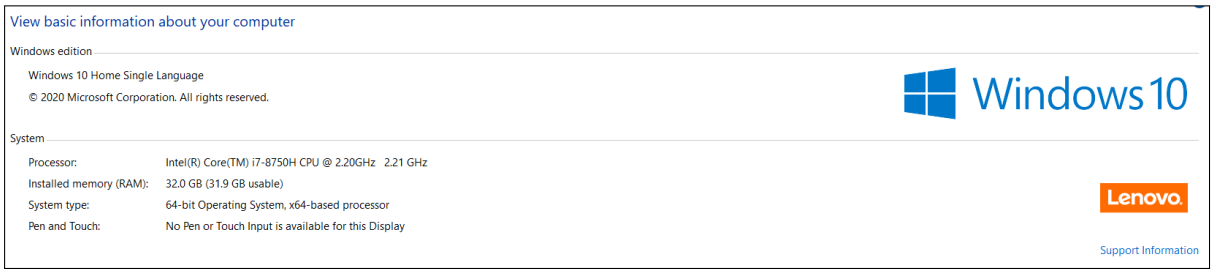

Figure 1: Hardware Configuration

2.2 Software Configuration

The developed solution has been developed using ConvNetJs [1] which has a dependency of NodeJs [2]. The implementation is discussed in this section:

2.2.1 NodeJs

The NodeJs is a JavaScript run-time environment built on the foundation of Chrome's V8 JavaScript engine. In this research, NodeJs version 12.18.2 has been used. It supports various API's (Application Programming Interface) which can be leveraged for the benefit of developed solution. It also has ES6 Features i.e. ECMAScript. The installation of NodeJs is described below:

1. Firstly the installer needs to be downloaded from NodeJs Official Website and select the desired version of operating system. In this project as we're working on 64 Bit Windows machine, the respective installer is downloaded and has been installed. It has been explained in Figure 2:

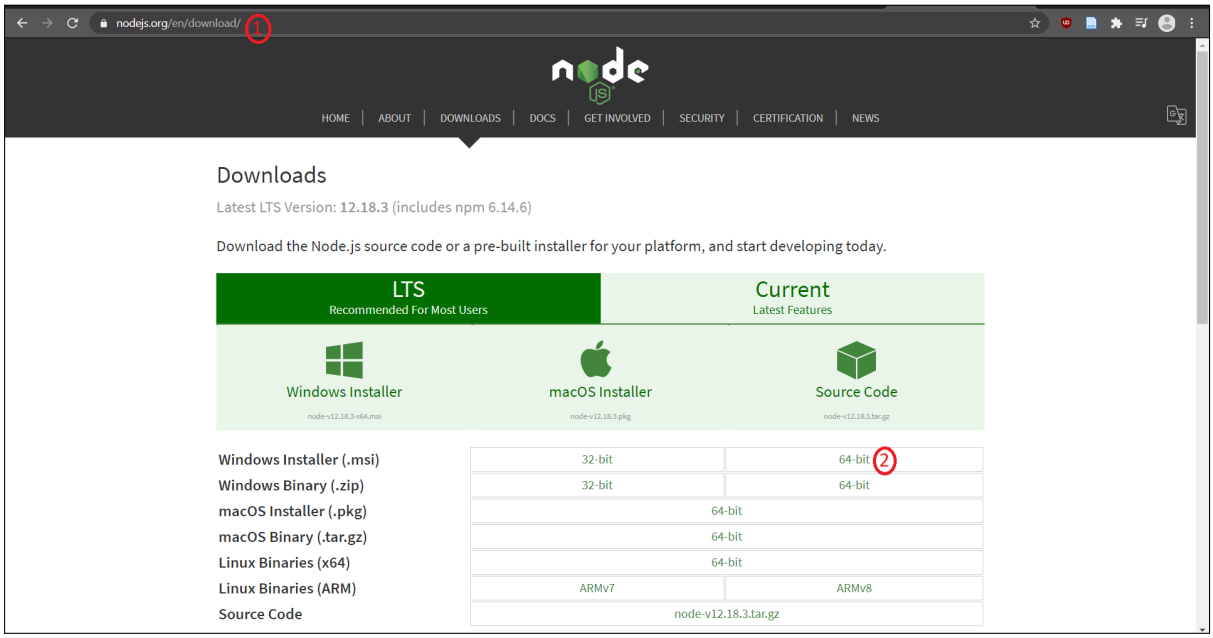

Figure 2: Website of NodeJs

2. Install the NodeJs as per standard format and Click on **Finish**. Ideally the content should be installed in 'C' drive and should not be interrupted while installing.

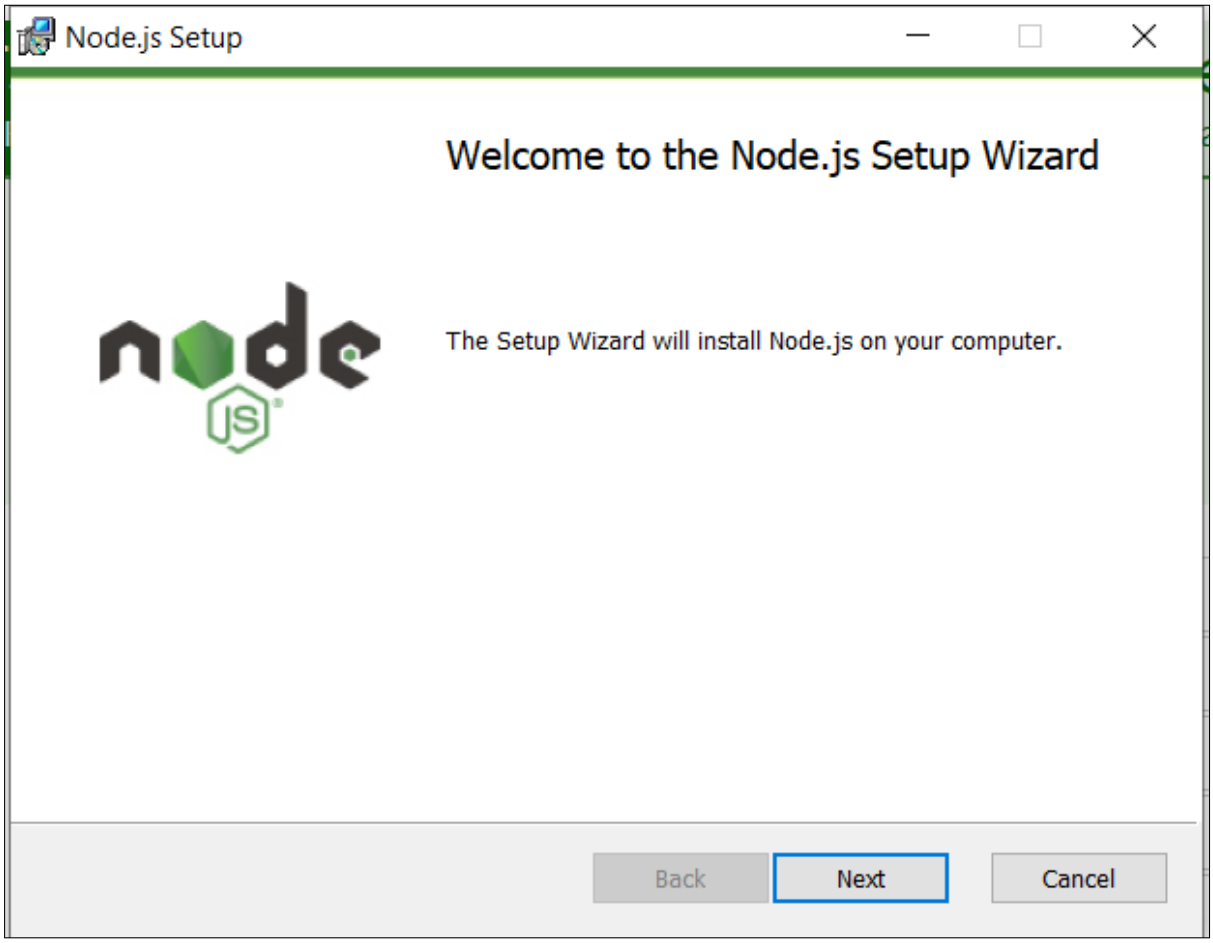

Figure 3: NodeJs Installer

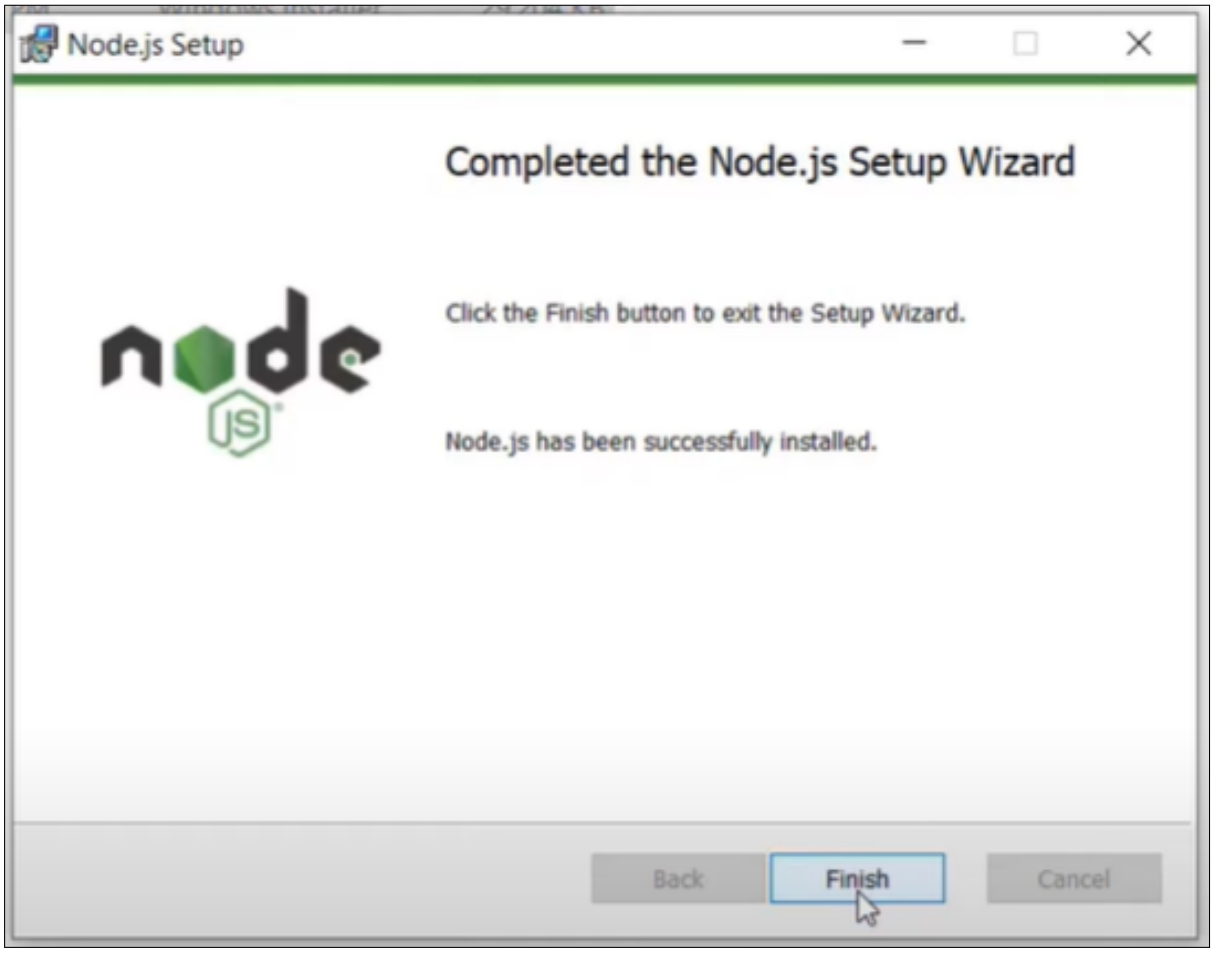

Figure 4: NodeJs Installation Finished

3. After installation of NodeJs is complete, verifying of the version can be done in Command Prompt with the command **node -v**

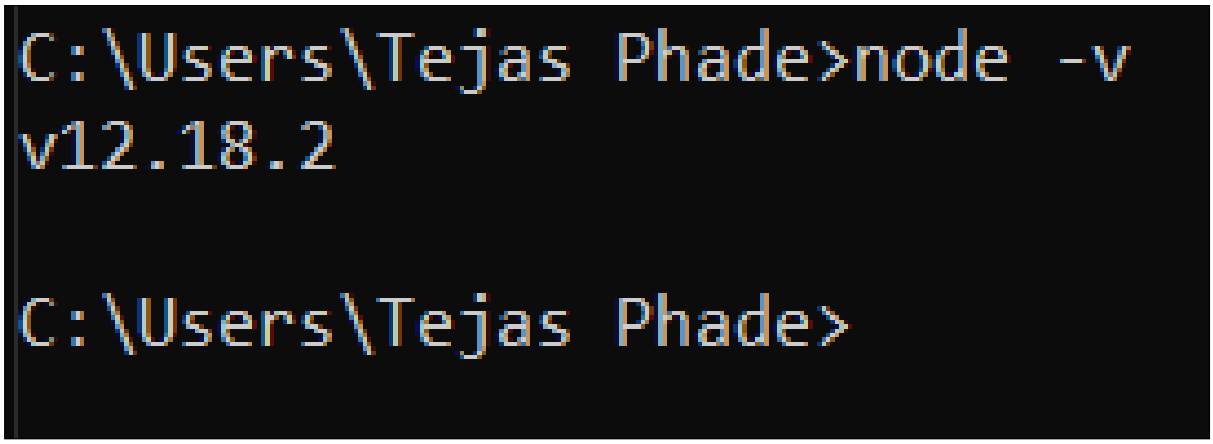

Figure 5: NodeJs Version Check

2.2.2 Visual Studio Code

The program has been developed using Visual Studio Code [3]. The VSCode provides multiple plugins and flexibility to the code and hence it has been used to develop the artefact.

3 Importing/Uploading The Extension

The developed solution is an extension of *Google-Chrome* and it needs to be uploaded to the browser before training or taking any samples. The steps to import an extension into the browser are listed below:

1. After opening *Google-Chrome* click on the three dots and and moving the courser to 'More tools..' will give us the 'Extension' menu. Since it does not have any shortcuts on keyboard, this is the only way to open it. The details are showed in below Figure:

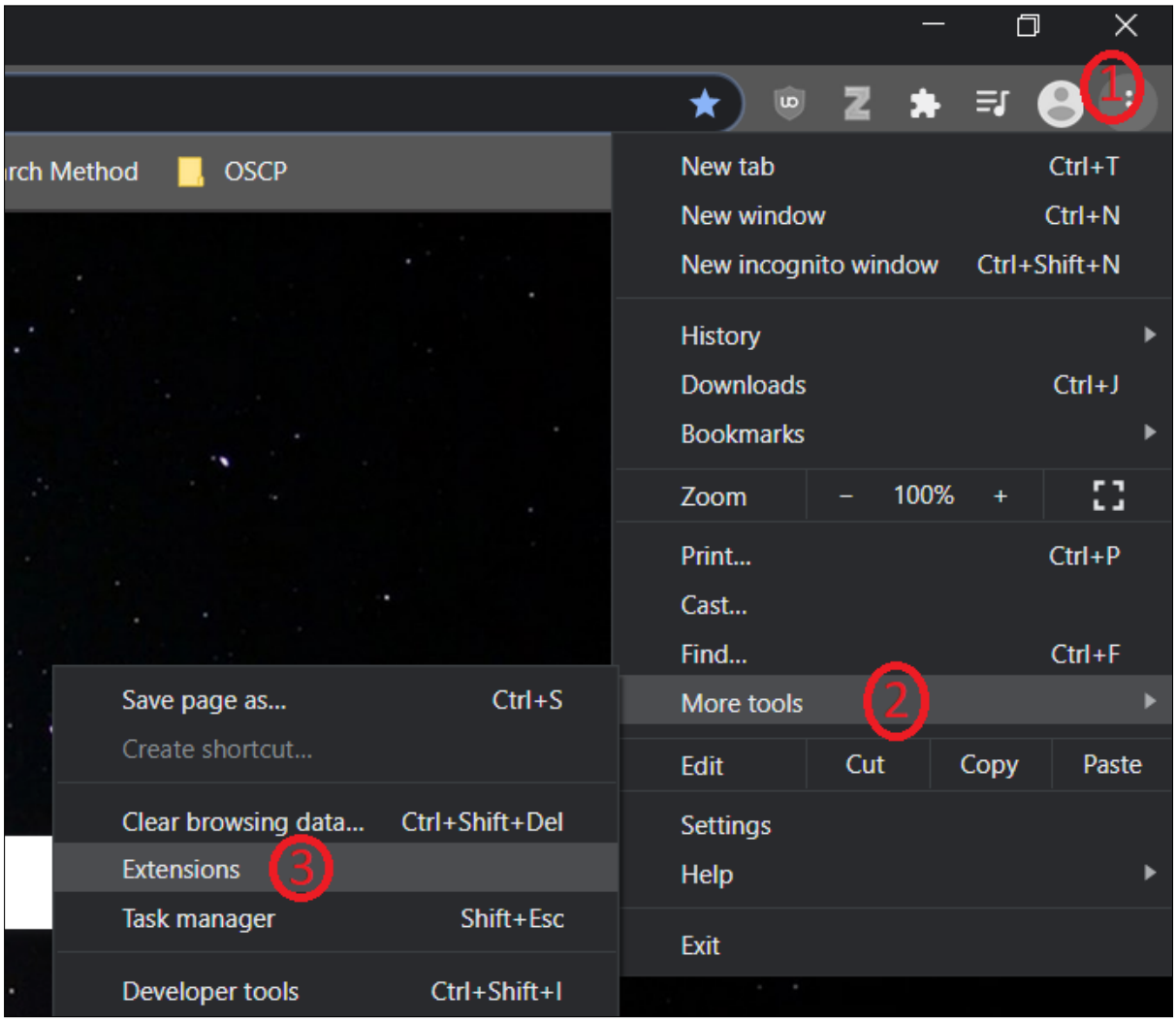

Figure 6: Extension Menu

2. Once the extension menu is opened, you will be greeted with the existing extension in the browser. Now, since the extension is new and not published into the Extension-Store, there is a need to enable 'Developer Option' which can be seen in upper-right corner of the screen, illustrated in the Figure 7.

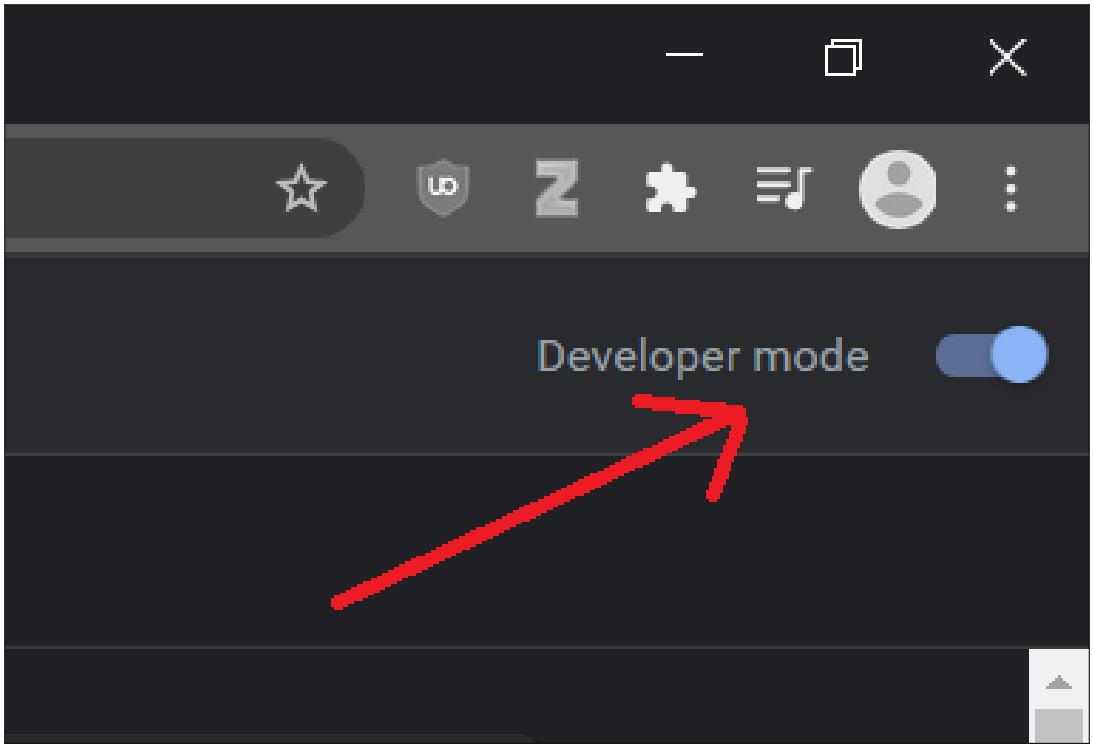

Figure 7: Developer Mode

3. The developer mode will enable us to upload any extension which is not in the store. Clicking on 'Load unpacked' will open up a window from where a selection of extension is possible.

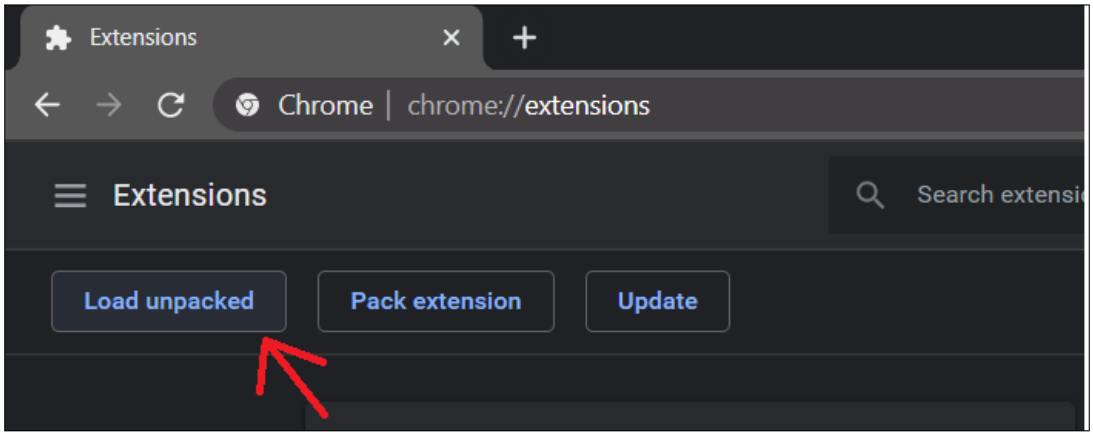

Figure 8: Import/Load Option

4. Now, once the selection window is open it is required to select the extension file.

In this case we've named it 'chrome-ext' which has all the essential files to run the extension.

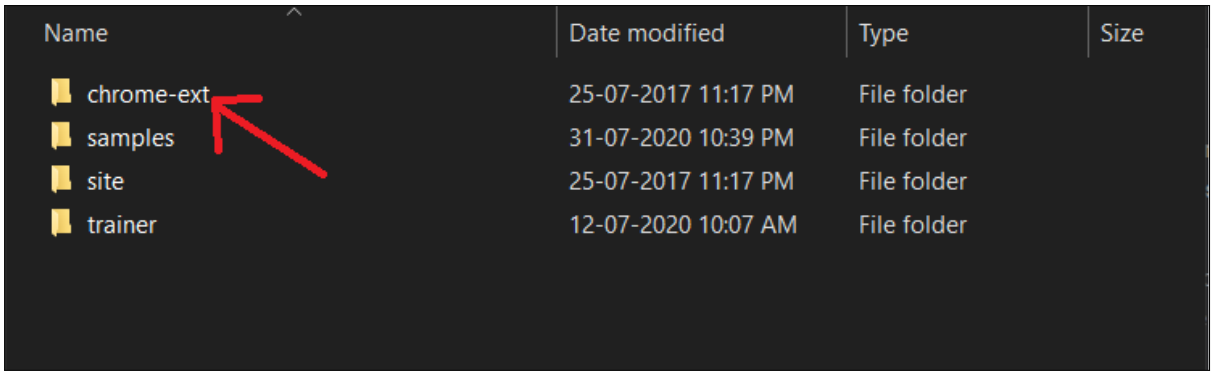

Figure 9: Extension Folder

After selection of the file, we can the extension in the 'Extension' menu and can modify some software aspect of it. The added extension is shown in figure below.

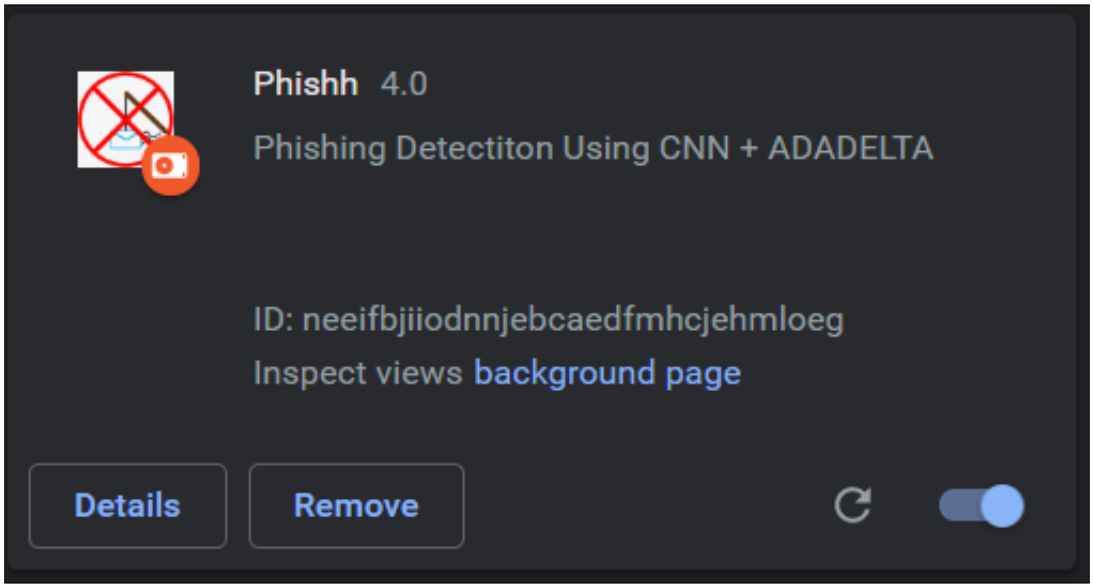

Figure 10: Final Added Extension View

4 Data Collection

The Convolutional Neural Network is trained using Images as the sample-set. These images are manually sampled with the help of the extension. No screenshot of the page can work as while training the algorithm, it is needed to have the specific resolution and grading which is possible only if the image is grabbed with the help of extension. Here are the steps to grab a sample of a login webpage.

1. Visit the desired login webpage. Here, we're taking *www.google.com* as an example.

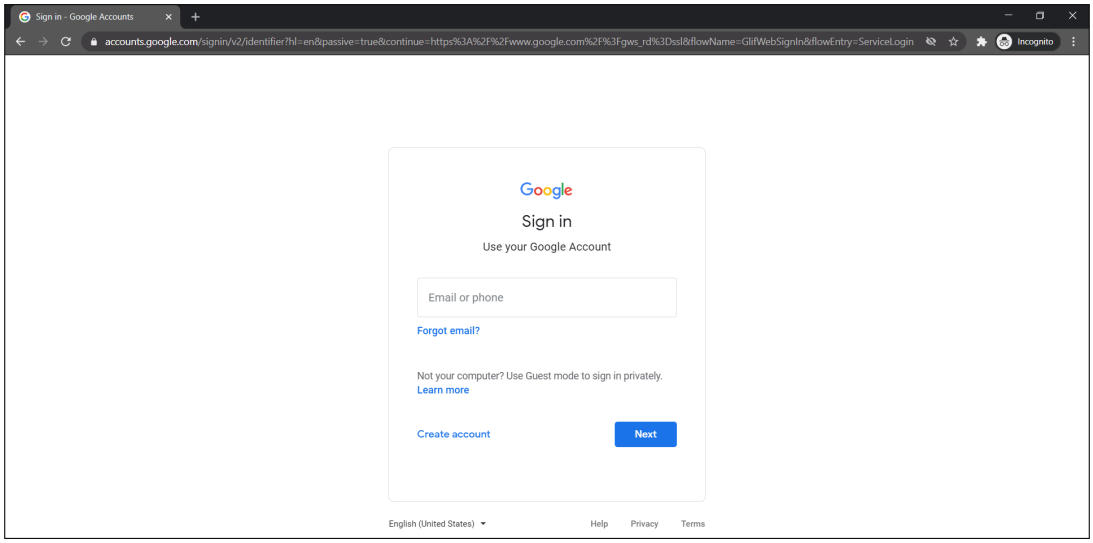

Figure 11: Google Login Webpage

2. After reaching the desired webpage, click on the extension icon and it will run the analysis. This might take couple of seconds.

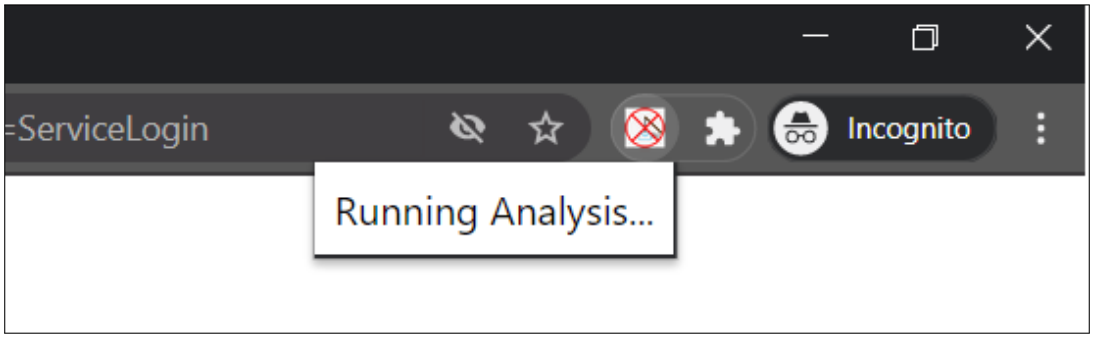

Figure 12: Running The Analysis

3. After analysis is complete, if the image is matched into the database, it will show the matching website. If it is new and does not know the particular website, then it will show that as well.

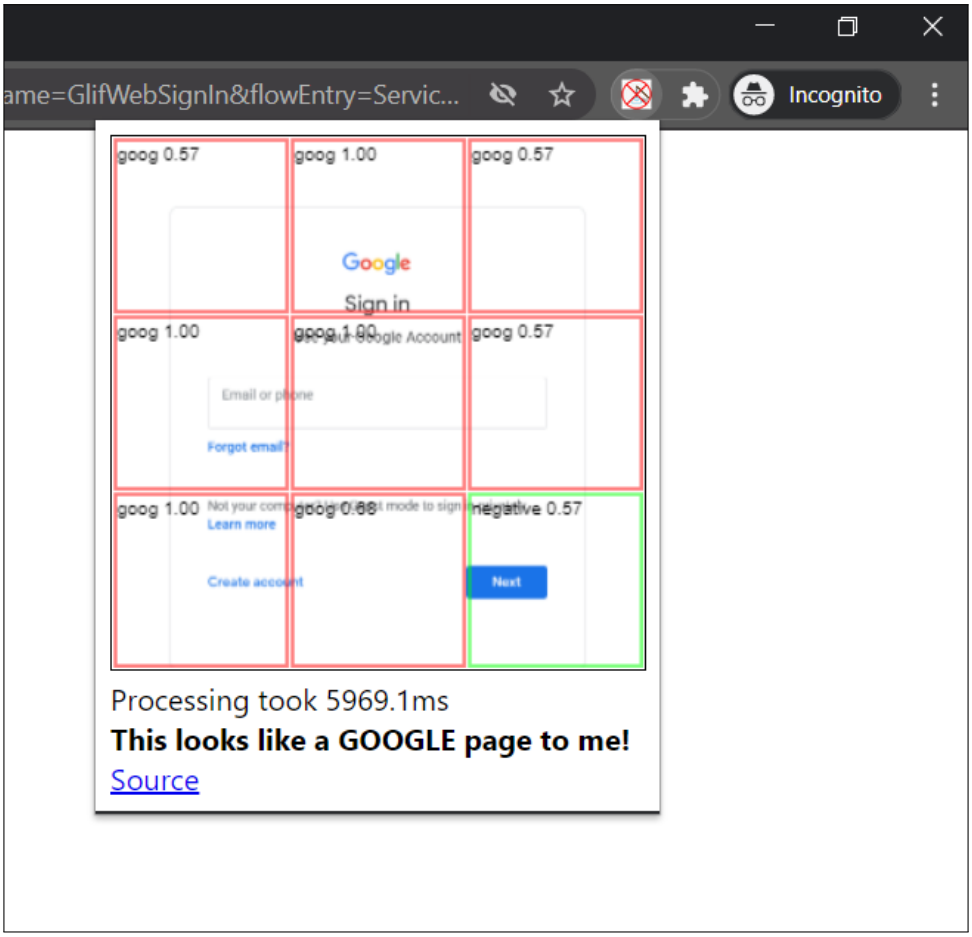

Figure 13: Complete analysis on Known Website

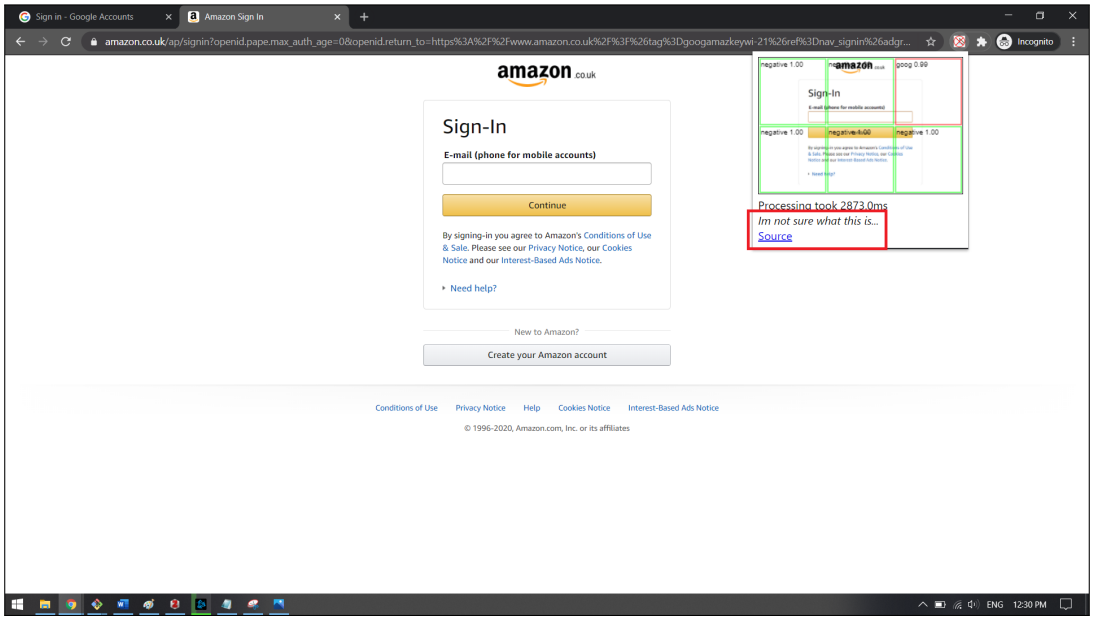

Figure 14: Complete analysis on Unknown Website

4. So, if you want to improve the accuracy of current webpage or introduce any new website, saving the image in the respective folder is vital. It is shown in the below Figure.

   Firstly, click on the link Source and it will pop-up the image. Right-clicking on the image denoted by '1' will give us a drop down menu. From there, select *'Save Image as..'* and save it in the respective folder, let's assume we're taking example of Google here then the image will be saved in folder named Google.

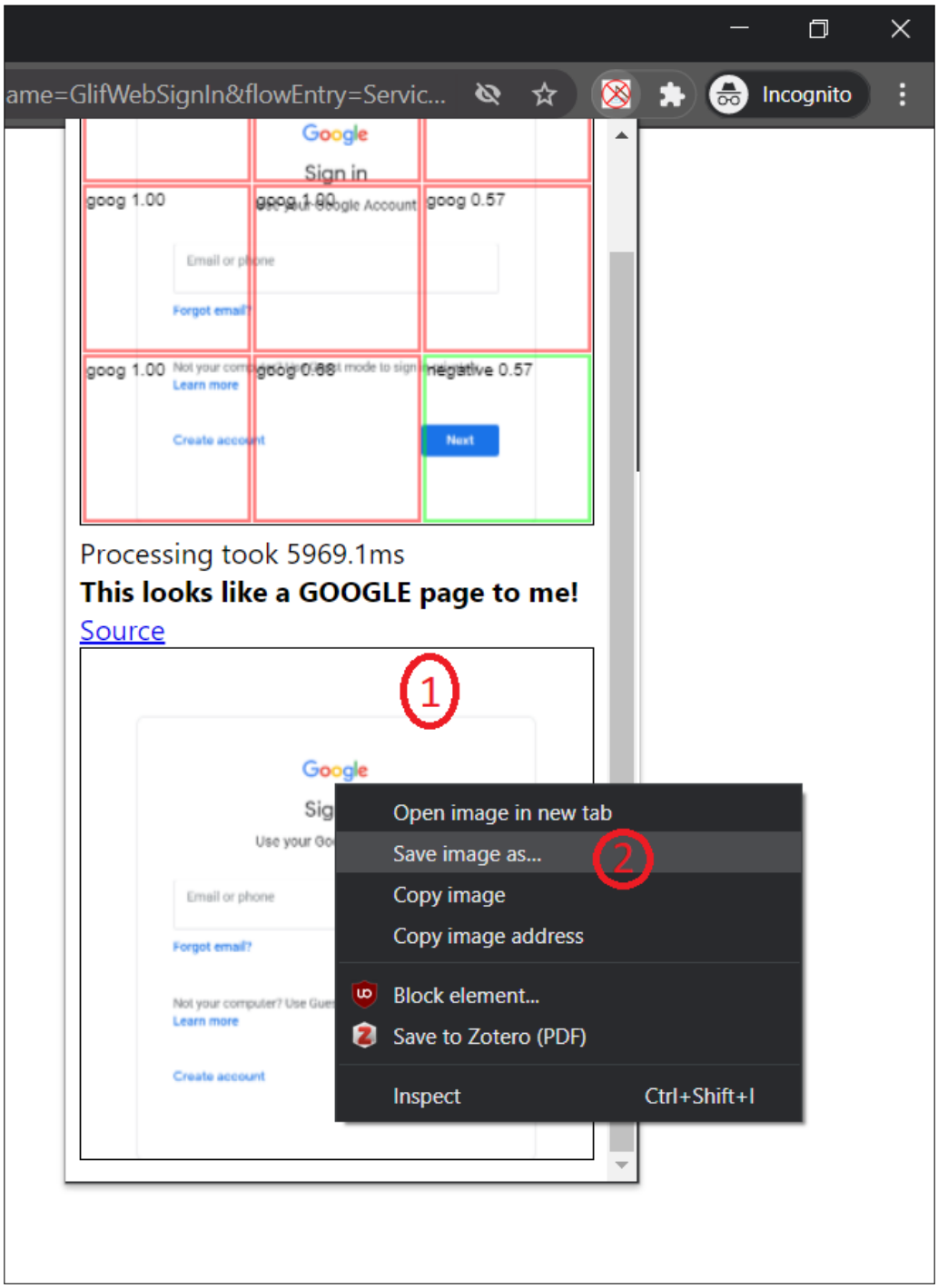

Figure 15: Save Image to respective Folder

5. There is no limit to how many images of the webpage you save. Here is the sample-set to train google webpage.

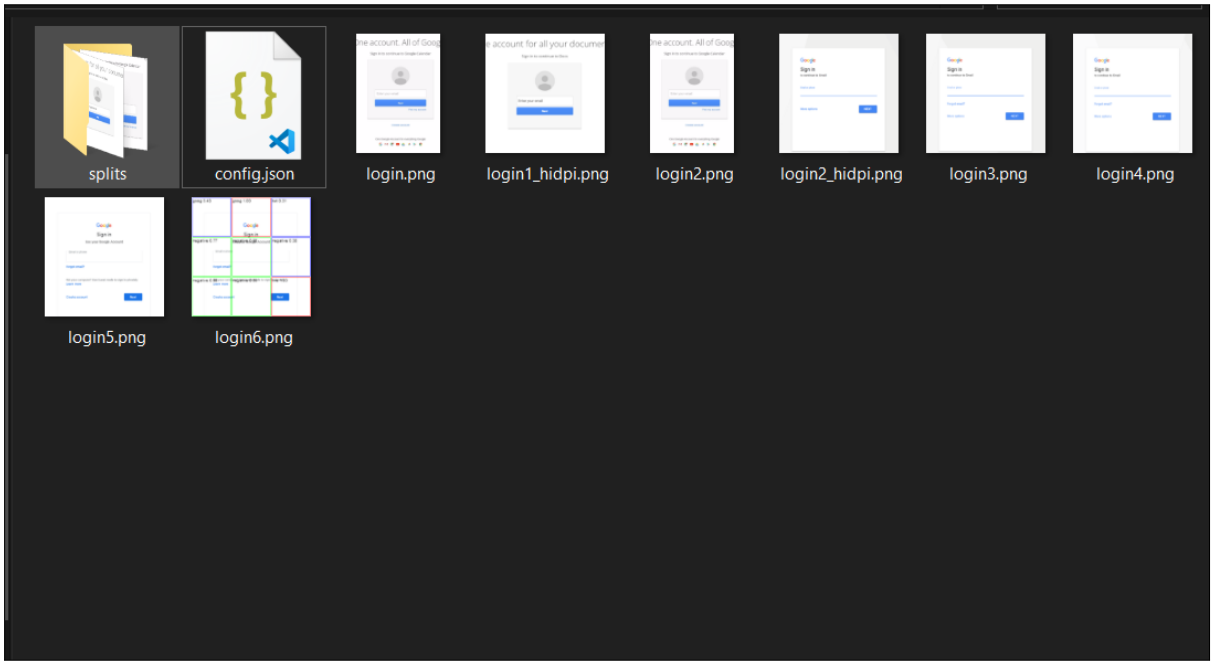

Figure 16: Collection of sample

5 Training of Convolutional Neural Network

After the completion of Extension Import and Data Collection comes the step to train the network. The steps to training the Neural Network are as follows:

1. Before training the network, it is important to specify the ID for the given domain and different domains to which the image sample belongs to. This will be coded into each folder of samples and file name will be 'config.json'. In this example, no matter if you sign in to G-Mail or YouTube it will always redirect the page to 'accounts.google.com' so we will only add one domain here as shown in below figure.

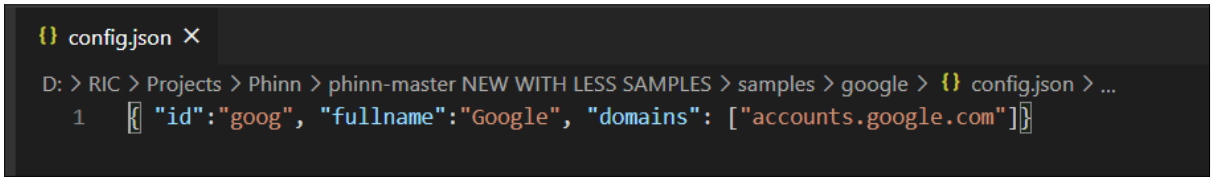

Figure 17: Config.Json File

2. Now there are several bash scripts which invoke specific functions utilizing each aspect of the extension. To train the neural network, only one script is needed, 'train_network.sh'.

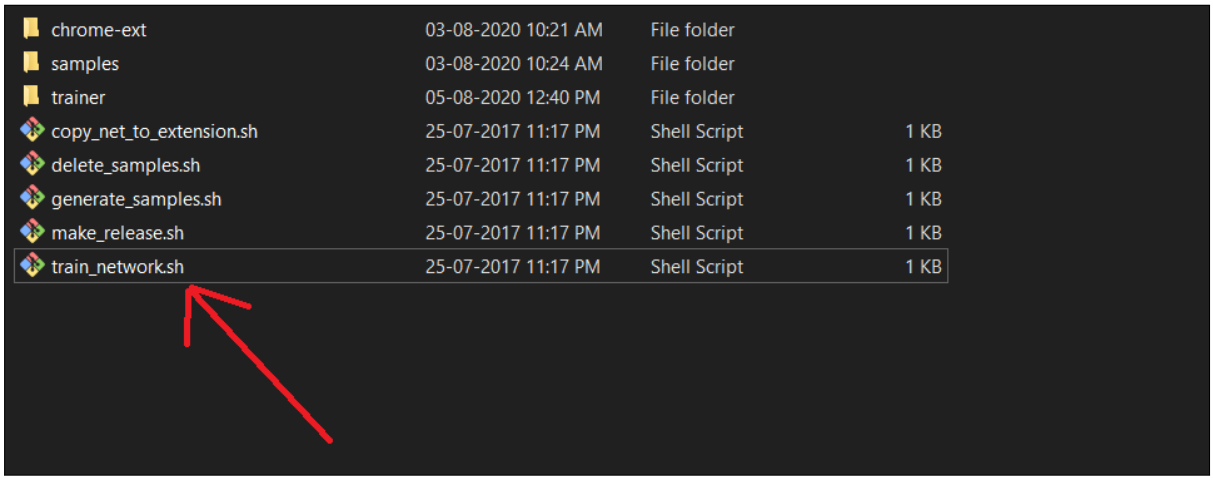

Figure 18: Train_Network.Json file

3. Clicking on this will invoke the training scripts and neural network will be trained. The time taken to train will depend on the base machine configuration and image sample numbers. Here, we're only added two samples just for reference:

```
Tejas Phade@LAPTOP-7J9JE8GR MINGW64 /d/RIC/Projects/TESTING
$ ./train_network.sh
Initializing sample labels.
{
  id: 'box',
  fullname: 'Dropbox',
  domains: [ 'dropbox.com', 'www.dropbox.com' ]
}
{ id: 'goog', fullname: 'Google', domains: [ 'accounts.google.com' ]
{ id: 'negative' }

Sample classes identified: ["dropbox","google","negative"]
Loading all samples...
Done! Loaded 75 samples.
Created new network.
Starting training. This will take a while...

Performing validation pass...
Validation accuracy: G=0.398 V=0.398
Training accuracy: G=0.4225 V=0.49127
Forward time per sample: 265ms
Backprop time per sample: 311ms
Classification loss: 1.45885
L2 Weight decay loss: 0.00061
Negative sample ratio: 0.15
Samples seen: 100

Current label accuracy:
    dropbox 0.04
     google 0.80
   negative 0.35
```

Figure 19: Training the network with two sample

4. Training will stop once it reaches the desired accuracy level. For reference accuracy of 95% is set so the training stopped at that stage shown in the below Figure.

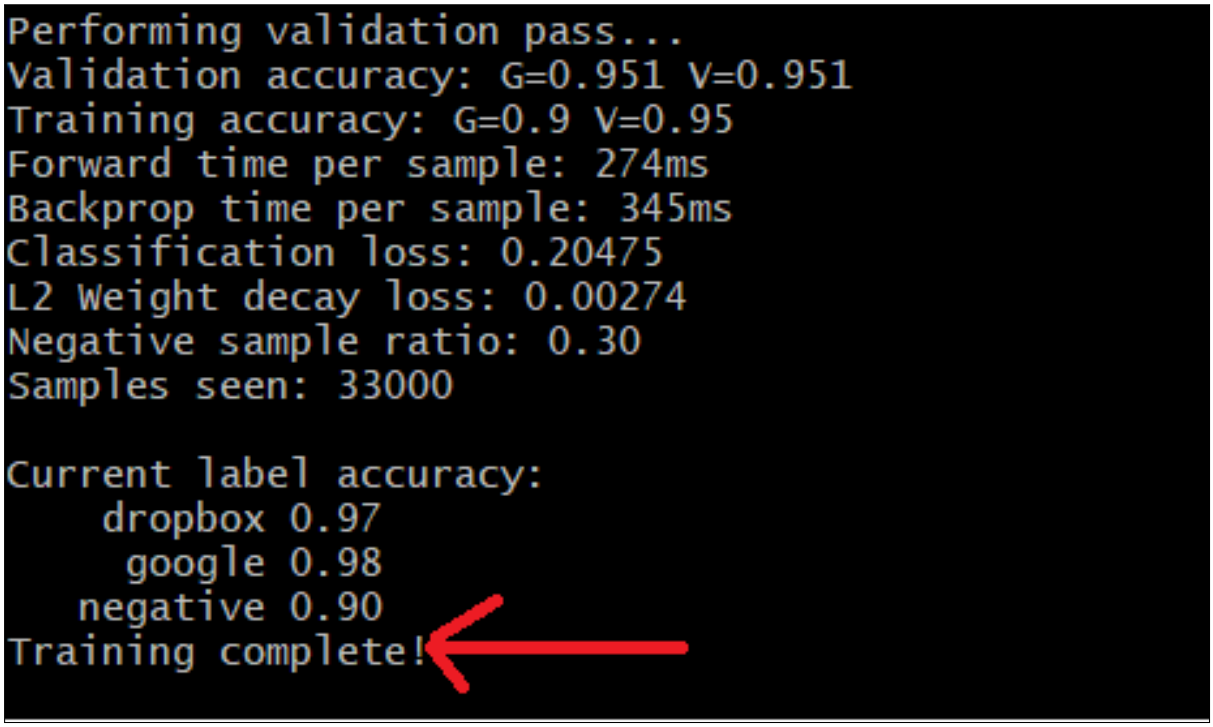

Figure 20: Training Complete

6 Conclusion

Aim of the research was to Detect Phishing Webpages with Convolutional Neural Network and it has been achieved by following the aforementioned steps.

References

[1] "ConvNetJS: Deep Learning in your browser." [Online]. Available: https://cs.stanford.edu/people/karpathy/convnetjs/index.html

[2] Node.js, "Node.js," library Catalog: nodejs.org. [Online]. Available: https://nodejs.org/en/

[3] "Visual Studio Code - Code Editing. Redefined," library Catalog: code.visualstudio.com. [Online]. Available: https://code.visualstudio.com/