Flag Identification

Joseph Raskind
Computer Science
Binghamton University
jraskin3@binghamton.edu

Sheriff Samateh
Computer Science
Binghamton University
ssamate1@binghamton.edu

ABSTRACT

The Internet of Things (IoT) allows for computationally weak devices to offload difficult-to-run tasks onto servers far away from the device's physical location. This core idea of the IoT framework is absolutely crucial to the application of image identification which is why it pairs so well with the task of flag identification. Since device's do not need to be powerful to use IoT applications and are rather affordable it becomes easy to see how such devices can be used for educational purposes. One such purpose presented in this paper takes the form of flag identification. The aim of the flag identification application is to help users learn more about vexillology and the world around them.

CONCEPTS

•Internet of Things •Deep Learning •Cloud Computing

KEYWORDS

IoT, Flags, Image Recognition, Deep Learning

1 INTRODUCTION

Flags provide an interesting challenge to those ignorant of them because when a person sees one that they do not know the origin of it can prove to be rather difficult to accurately find out exactly what information is tied to a particular flag in the moment. For example, an individual (possibly a child) may look at the French flag and not recognize that the flag signifies the country of France. Furthermore, in trying to look up the flag they may try to search for a flag with red, white, and blue stripes and, in doing so, quickly find that there are many flags who fit that description. The purpose of the flag identification application is to rid people of that issue through the use of a convolutional neural network (CNN) algorithm performed on the cloud.

2 DESIGN

2.1 **Hardware**

2.1.1 Android Nokia One

Figure 1: Android Nokia One

The only piece of hardware necessary for this project is an android phone capable of running Android SDK 6.0 or higher.

2.2 **Software**

2.2.1 Amazon Web Server

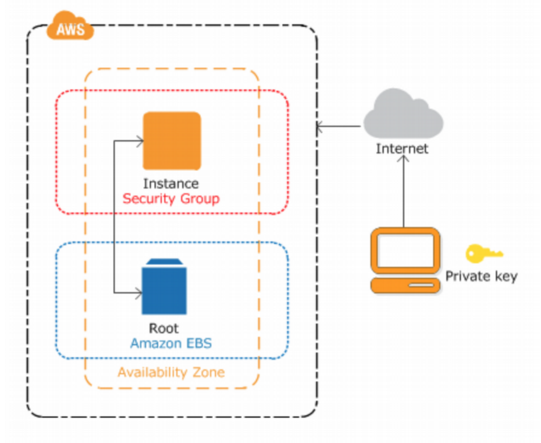

Figure 2: Amazon Web Service EC2 instance visualization

The Keras library is used to implement a CNN that can predict flags of various origins. At first, due to limited number of images available to us, we couldn't get an accuracy above fifty percent; however, after some research, we decided to use a ResNet50 pre-trained model. Additionally, we added a single dense layer to the network with a normalized exponential function which, in turn, increased our accuracy drastically. We them deployed this pre-trained model on an Amazon Web Service ec2 instance. We then used a python flask web server on the instance to handle user requests. In order to open the ec2 instance to the outside network we used nginx, while having all requests are proxied to the flask server. The flask server processes the image, forwards it to the pre-trained model, and sends back a response to the requester in a JSON format.

2.2.2 Android Application

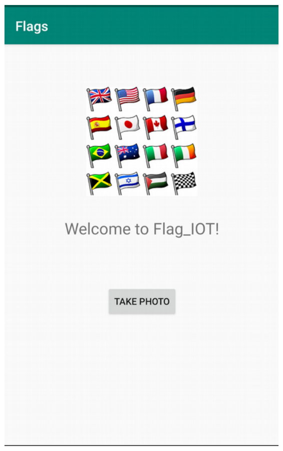

Figure 3: Flag Identification application home screen

The flag identification application was written in kotlin using the Android Studio IDE because of its user-friendly design and how simple it is to test changes to applications.

2.3 **Overall design**

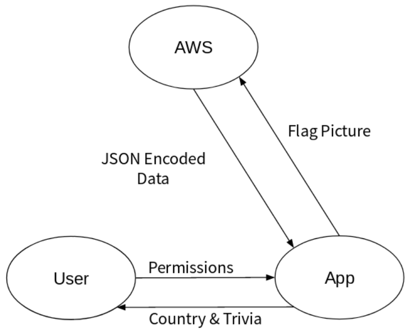

Figure 4: Design flow schema for flag identification

The design of the overall system is rather simple. The application and the server run independently of each other. The only interaction they have is when a user takes a picture and sends that picture in a post request to the server.

3. IMPLEMENTATION

Upon starting up the application the user will be asked for three separate permissions: camera permissions, write to external memory permissions, and read from external memory permissions. It is essential to the application that these permissions are given as the application must be able to take a picture, save that picture, and send that picture to the server. Once permissions are given, the user is free to take a picture of whatever they want. Once that image is taken, the user is then asked to confirm the picture they have taken is what they want to sent to the server for flag identification. The image is then sent to the server in a multipart/form-data request which is grabbed by the server and runs the CNN algortithm on the picture with the pre-trained flag model. The algorithm will then provide a confidence rating of all of the flag labels present in the model. If the highest confidence rating is below fifty percent an error message will be returned back to the application indicating that the image provided could not be used to identify the country; otherwise, trivia of the identified country will be sent.

4. EVALUATION

Once the model was properly trained with upwards of seventy images per flag we were able to maintain a confidence rating of over ninety percent meaning that when taking a clear picture of a recognized flag the algorithm managed to produce the correct output almost every time it was called.

5. CONCLUSION

The flag identification managed to perform to expectations by providing the correct label information almost every time the algorithm was run. Of course, it is difficult to tell if the application would prove effective in a case of wide adoption since the flags tested on were, of course, the flags used for training the model, therefore further testing and study would be required to dig deeper into that issue. It is undeniable that such an application could prove to be a rather engaging learning tool for those who are curious about flags and the different and interesting patterns that make them up.

REFERENCES

[1] https://aws.amazon.com/

[2] https://developer.android.com/