iPre: A Cross-platform Mobile App for Image Classification

Dade Sheng
Department of Electrical and
Computer Engineering
University of Toronto
Email: dade.sheng@mail.utoronto.ca

***Abstract*—Image Classification has been studied and developed for years. Machine learning models continue to show improvements on classification results. To explore more practical usage of these work, the paper describes the development of a cross-platform mobile app, called iPre, which runs in Android phones and web browsers. Part I describes the background, and introduces iPre, which follows a web development pattern and uses services from Amazon Web Services. Part II describes the structure of the app, from frontend to backend, with server support, Android development, and classification model. Part III demonstrates the app on an Android device. Part IV demonstrates the app on a Chrome browser. Part V concludes the work.**

I. Introduction

Our brains make vision seem easy. It doesn't take any effort for humans to tell apart a lion and a jaguar, read a sign, or recognize a human's face. But these are actually hard problems to solve with a computer: they only seem easy because our brains are incredibly good at understanding images.

In the last few years the field of machine learning has made tremendous progress on addressing these difficult problems. In particular, it's been found that a kind of model called a deep convolutional neural network can achieve reasonable performance on hard visual recognition tasks – matching or exceeding human performance in some domains. Basically, a model needs to be trained using existing datasets, after which the model is used to predict new inputs. However, the training process can take months to be complete for quite large training datasets. For predicting an image on a mobile device, it's inappropriate to include training process and let users wait months for results. So, it would be a good idea to take advantage of pre-trained neural networks.

Researchers have demonstrated steady progress in computer vision by validating their work against ImageNet – an academic benchmark for computer vision. Successive models continue to show improvements, each time achieving a new state-of-the-art result: QuocNet, AlexNet, Inception (GoogLeNet), BN-Inception-v2. At the meantime, Python libraries for numerical computation in the field of machine learning have been developed, such as TensorFlow and Theano. Keras is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. There are many trained image classification models for Keras, such as VGG16 [1], VGG19, ResNet50 [2], and Inception v3 [3].

iPre, is a mobile app for image classification using Keras library with VGG16 model. It behaves like a photo album. After registering an account, users can log in and upload their images. All the uploaded images would appear at main page. Clicking one of them, users would see predicted classifications of that image, presented in a pie chart, a bar chart and a probability table. All the images and their predictions are stored permanently, so users can always log back in and browse their images.

A. Web Based Application

The app is designed by web technologies to make it more flexible. Mobile apps are dependent on platforms. It requires different technologies to design an app in IOS and Android, or other mobile platforms. In some circumstances, a web application stands out because it's running in web browsers of both mobile and PC devices. Therefore, iPre was firstly developed as a web application, and then was migrated to an Android app. The development of Android side is rather straightforward. A web browser was embedded in the Android app to render contents fetched from remote server. In a similar way, it shouldn't take much effort to develop an IOS version, either on iPhone or iPad.

B. Amazon Web Services

Amazon Web Services (AWS), a subsidiary of Amazon.com, offers a suite of cloud computing services that make up an on-demand computing platform. These services operate from 16 geographical regions across the world. They include Amazon Elastic Compute Cloud, also known as "EC2", and Amazon Simple Storage Service, also known as "S3".

1) S3: Amazon S3 (Simple Storage Service) is a web service offered by Amazon Web Services. Amazon S3 provides storage through web services interfaces. [4] The app stores all user images on S3. Once a user uploaded an image, the image would be temporarily cached on server for classification, and permanently stored in S3. In this way, it provides a stable storage and resource availability. Later on, the app can display images by direct links provided by S3.

2) EC2: Amazon Elastic Compute Cloud (EC2) forms a central part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), by allowing users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. [5] The app uses EC2 instances as backend server, which stores user data and image keys, uploads images to S3, and classify images with Keras library.

II. Structure

A. Frontend Design

For frontend development, it involves web technologies, such as HTML, CSS, and JAVASCRIPT. In order to better fit in mobile devices, responsive web design is employed. Responsive web design (RWD) is an approach to web design aimed at allowing desktop webpages to be viewed in response to the size of the screen or web browser one is viewing with. [6] To achieve better user interface and responsiveness, the app uses a front-end library, called Bootstrap. Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. [7] Following the responsive design pattern, three main pages were designed:

1) Login Page: It contains two parts. First part is registration, where new users can create new accounts. User data would be permanently stored on EC2 instance database. After registration, users can log in with their usernames and passwords. JAVASCRIPT is written here to validate user inputs, to make sure all fields are not empty and passwords are matching certain criterion.

2) Main Page: After successful log-in, users would be taken to this log-in page, where they can view all the images they uploaded and predicted in an album. Also, they can upload new images for classifications. Once new images are uploaded, they would appear in album. There is also an option for logging out accounts. After clicking on a specific image, users would be taken to view pages.

3) View Page: Here show the classification results of an image. It contains four parts. The first part is the image itself. The remaining parts are different representations of classification results, with each of them having five values, which are organized based on classification probabilities. These includes a pie chart, a bar chart, and a table.

B. Backend Design

1) Flask: The app is supported by Flask. Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. [8] Once a flask app is running on server, it receives requests from clients. It handles all incoming requests, and return data as requested. It works closely with local database which contains user data.

2) Celery: The app takes approximately 7 seconds to classify an image. It would not be user-friendly if it takes too long to respond. The app takes an asynchronous approach. Once an image is uploaded, it starts an asynchronous task using Celery, which executes in the background. Celery is an open source asynchronous task queue or job queue which is based on distributed message passing. While it supports scheduling, its focus is on operations in real time. [9]

C. Server Support

an AWS EC2 instance is serving as a server for the app. The instance type is c4.large. C4 instances are the latest generation of Compute-optimized instances, featuring the highest performing processors and the lowest price/compute performance in EC2. It has 2 virtual central processing units and 3.75GB memory. Several packages are installed and configured:

1) Redis-server: Redis is an open source Database, in-memory data structure store. It communicates with Celery for background task execution.

2) Supervisor: Supervisor is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems. It deals with the management of backend Flask app.

3) MySQL: MySQL is an open-source relational database management system. The app uses a MySQL database to store user information and image keys on S3.

D. Android Development

With Android Studio as IDE, the Android side of development is straightforward. The app's minimum SDK version is 15 (Android 4.0). Running with full-screen mode, the app is embedded with a WebView, which is standard component in Android to render web contents. With JAVASCRIPT enabled and several functions modified, the web app works more native on Android platform.

E. Classification Model

As mentioned above, the app uses Keras library with VGG16 model for image classification. To classify an image, firstly, VGG16 model is loaded with weights pre-trained on ImageNet. Secondly, it loads the image with target size 224 by 224. Thirdly, image data is converted to an array and been normalized. Then the image data is passed into classification model and been classified. Finally, the app decodes the model outputs as different categories and gives top five 'winners' based on probabilities from high to low.

III. Android Demonstration

To better demonstrate the usage of iPre, this part walks through the functionalities from a user's perspective. It's been installed on Nexus 6, an Android device with API 25.

A. Log In

Figure 1 shows the log-in page, which allows the user register a new account or log in with an existing username. For demo purpose, a user named 'ece1780' has been created. Several images were already uploaded under this account.

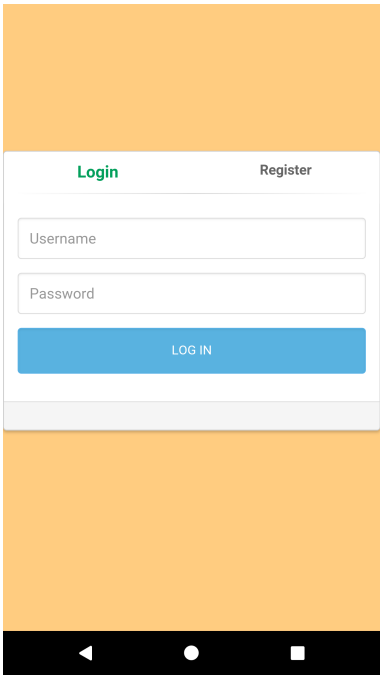

Fig. 1. Log in.

B. Main Page

Figure 2 shows the main page, including an photo album, which lists all the images of the user 'ece1780'. The user can scroll and browse the images. There is a collapse button at top right corner. After clicking it, the user can choose to upload a new image, or log out current account.

Fig. 2. Main page.

C. Classification

If we choose an image from album in main page, iPre would lead us to classification page. Figure 3 and figure 4 shows us an example (bottom right corner image from album). There are three presentations of classification results, including a pie chart, a bar chart, and a table. Each presentation shows us five values of probability.

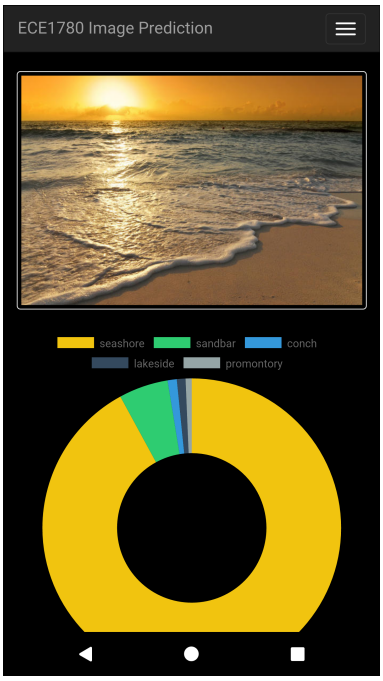

Fig. 3. View page.

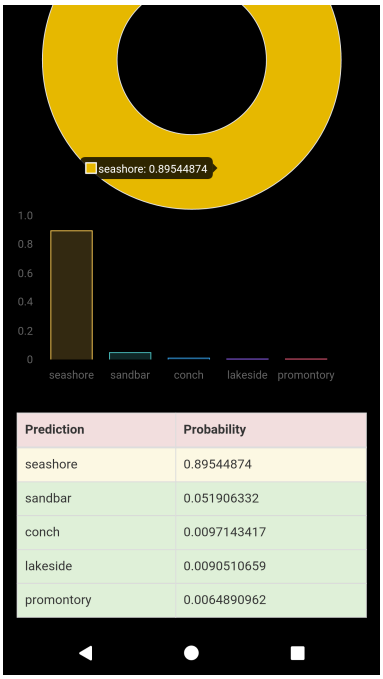

Fig. 4. View page.

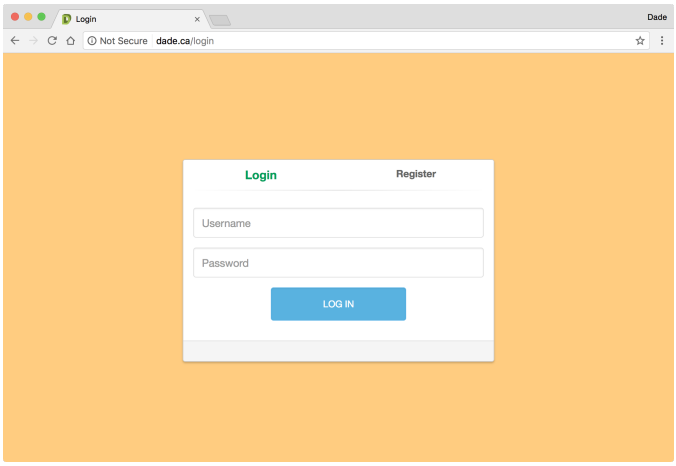

Fig. 5. Log in.

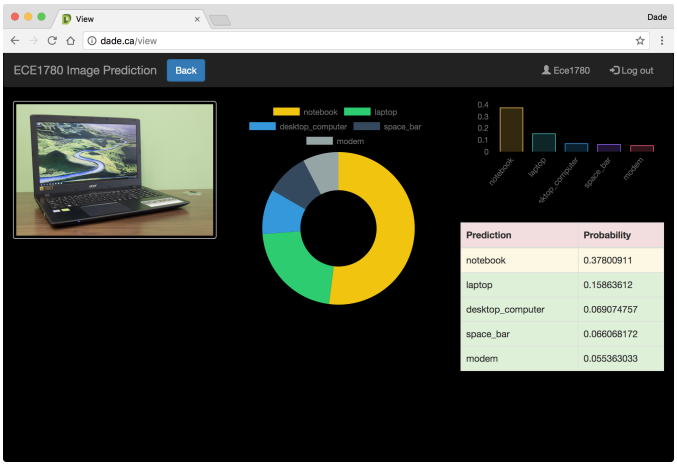

Fig. 7. View page.

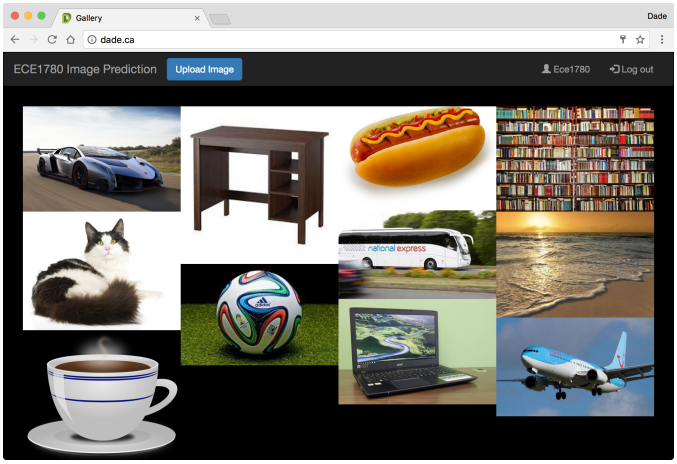

Fig. 6. Main page.

IV. Browser Demonstration

Since it's built through a web development approach, it doesn't need to be confined to any specific platform. Figure 5, 6 and 7 show a web version of the app, which runs on a Chrome browser.

V. Conclusion

The work is related to Image Classification problem, which is the task of assigning an input image one label from a fixed set of categories. This is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. Taking advantage of pre-trained neural networks and high-level libraries, iPre classifies images uploaded to user's albums. It is platform-independent. Following a web development pattern, iPre was designed to be more flexible. It can be run on any browsers, or migrated to native mobile platforms, such as Android, IOS, and Windows Phone. It's simple to use and user-friendly. It improves user interface by different classification result interpretations, including two charts and a table. The app is currently running on an EC2 instance of type c4.large. It takes approximately 7 seconds to classify an image right now. Classification time depends on server performance. The better the server performs, the sooner it classifies. As listed in Table I, classification time depends on EC2 instance types.

TABLE I
PERFORMANCE ON DIFFERENCE EC2 INSTANCE TYPES.

Model	vCPU	Mem (GiB)	Time (second)
t2.micro	1	1	50.42
t2.small	1	2	35.87
m3.medium	1	3.75	13.25
c4.large	2	3.75	7.61
c4.4xlarge	16	30	6.12
c3.8xlarge	32	60	6.07

References

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
[2] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.
[3] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.
[4] Wikipedia contributors. "S3." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 4 Mar. 2017. Web. 30 Mar. 2017.
[5] Wikipedia contributors. "EC2." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 28 Apr. 2015. Web. 30 Mar. 2017.
[6] Wikipedia contributors. "Responsive web design." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 27 Mar. 2017. Web. 30 Mar. 2017.
[7] Wikipedia contributors. "Bootstrap (front-end framework)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 23 Feb. 2017. Web. 30 Mar. 2017.
[8] Wikipedia contributors. "Flask (web framework)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 20 Jan. 2017. Web. 30 Mar. 2017.
[9] Wikipedia contributors. "Celery (software)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 22 Feb. 2017. Web. 30 Mar. 2017.
[10] Chatfield, Ken, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Return of the devil in the details: Delving deep into convolutional nets." arXiv preprint arXiv:1405.3531 (2014).