ECE 1780
Design and Implementation of Chinese Recipe Finder Android Application

1. Introduction

The Chinese Recipe Finder is an android application that can identify Chinese dishes using either image classification or speech recognition, and display their cooking instructions to the user. It is designed for food lovers or chefs who are interested in Chinese cuisine.

The application mainly has three functions: User can navigate through a menu of all 10 available recipes and select recipe to see details; User can open mobile camera from the application, real-time images of the dish are sent to convolutional neural network model which predicts the name of the dish, the cooking instruction of the identified dish is displayed on the screen; User can speak the dish name to the application, Houndify API is used for speech recognition of the dish name and the recipe of the identified dish is displayed on the screen. These will be explained in detail in the following section.

2. Android Design and Implementation

Android platform was chosen for this application because I found it more developer friendly from my experience in mobile development. In term of programming language, Java is easier to learn and more commonly used than objective-c.

Figure 1. Simple Layout diagram of Chinese Recipe Finder

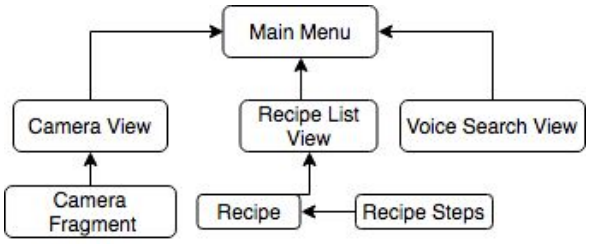

Figure 1 shows a simplified layout diagram of the application. The user can choose either camera view, voice search view or recipe list view from the main menu, which are used for the image classification, voice recognition and recipe list browsing respectively.

a. Implementation of Recipe Menu

For this function, three application layouts are implemented: recipe list view displays the list of available recipes, recipe steps view is wrapped into recipe view to display the cooking steps of a single recipe. Recipe list is implemented using List<Item> data structure, each item represents a recipe. Java tutorial on building Android ListView with ListAdapter [1] demonstrates the steps to build a list. Moreover, a JSON file is created to store the title and head image name of all recipes. Each recipe also has one corresponding JSON file which contains cooking step image names and text. All JSON files and images are stored as assets. These files are then loaded in Java classes to be displayed in application layouts. All cooking instructions and images were collected from the Internet.

b. Convolutional Neural Network Model for Image Classification

Convolutional Neural Network (CNN) model was used for image classification of this application. It was chosen for the reason of being very effective in image recognition and classification [2]. It allows image processing to be computationally manageable using similarity filtering [3]. The guide on TensorFlow layers [4] posted on TensorFlow official websites very helpful for learning coding implementation of a CNN model.

i. Model Implementation

For this application, a convolutional neural network based image classifier is built using TensorFlow in Python. The model mainly contains four different layers: an input layer, convolutional layer, pooling layer and dense layer.

To start with, 10 categories of training and testing images are collected from Google Images. These correspond to 10 different Chinese dishes. Each category contains 500 images in RGB format (channel of 3). 90% are used as training data and the rest 10% are test data. These images are loaded in Python code using Python Imaging Library and resized to 128x128 pixels, then passed to the convolutional layers.

Next, image feature maps are produced in the 2-dimensional convolution layers using 2 by 2 filters. These filters basically slide over an image input and do mathematical computation which represents the features of the image. In addition, Max pooling operations are done on the feature maps. It extracts the largest element from the feature map within a 2 by 2 window, strides of 2, thus making the input representations smaller but keeps the most signification information about them. For this classification model, I use 4 convolutional and pooling layers since dish images are complex (contain ingredients of multiple colors and shapes) and I want the model to learn more parameters of the input images.

Finally, the feature maps are flatten using reshape function and passed to a dense layer with 1024 neurons and ReLu activation which perform feature classification, then the prediction values are returned using the final logits layer which is a dense layer with 10 neurons (one for each dish category). ReLu is used to introduce non-linearity in the model since convolution itself is a linear operation. Softmax activation is applied to the prediction probabilities so that the sum of the probabilities is 1.

Table 1. number of filters of each convolution layer

Number of convolution filter	1st layer	2nd layer	3rd layer	4th layer
	32	64	128	256

ii. Model Training and Evaluation

To train the model, a batch size of 80 and epoch number of 50 are applied. It is trained using Tensorflow session and saved as checkpoint (.ckpt) files. For measuring the correctness of model predictions, I used softmax cross entropy loss function to calculate cross-entropy with one-hot labeled prediction values. Table 2 shows the accuracy and loss at the final epoch.

Table 2. Model Evaluation at epoch 50

Avg. Training Accuracy	Avg. Epoch Loss	Avg. Test Accuracy
100%	0.000479	53.12%

Notice that the test accuracy is low compared to the training accuracy, there exists overfitting in the model. Adding more training data and the use of dropout regularization may help reducing overfitting in the model. On the other hand, low test accuracy may imply what the model learns from training data does not accurately represent the test data. We could verify whether test data are similar to the training data. There may have images that are unclear or do not represent the dishes well.

One key finding related to accuracy is that the convolutional neural network model does not reject unknown classes and tends to assign them to its own classes. Classifying images of unknown classes (I tested with images of tables) using the Chinese Recipe Finder will show dish names to the user. To avoid this situation in future development, we could add probability threshold and reject test images as unknown if their probabilities are under the threshold.

Furthermore, it is found that prediction results of certain dish categories are better than the others. This could happen when the amounts of training data are not balanced between classes. However, this is not the case for our model since we use exactly 500 images for each category. A possible reason could be that images of dishes can vary significantly even if they are in the same class since people can use different ingredients (different colors and shapes) based on their taste and location. It is harder to extract common features in this case.

iii. Model Integration with Android

To integrate the convolutional neural network classifier with Android application, the model is first converted from .ckpt format to .pb format, then converted to Tensorflow lite file format (.tflite) which is a lightweight solution for mobile with low latency and a small binary size using Toco[5]. Android development tutorial[6] posted on TensorFlow official is very useful in demonstrating how to send the real-time image captured from mobile camera to Tensorflow lite file.

The Tensorflow lite file is included in the android application and loaded using Tensorflow Lite Interpreter android API inside the classifier class. When the user chooses from the application to identify dishes using the mobile camera, camera fragment view is wrapped into camera view and displays real-time camera view on the screen. The camera fragment Java class will initialize the image classifier, opens the mobile camera and keeps sending real-time images to the classifier as Bitmap. Images are converted from Bitmap to byte buffer inside the classifier Java class, then passed to the Tensorflow lite Interpreter. This Interpreter returns the probability array computed from the CNN model. A low pass filter is applied on the probability array to filter out high-frequency probabilities so that we could detect changes in probabilities. Finally, a priority queue is used to sort the probability array and the category with the highest probability is displayed on the screen. A popup saying whether the user is looking for the identified dish name will show on the screen when the probability is higher than 80%. If the dish name is confirmed, the screen will jump to recipe view that displays the classified dish's cooking instructions. Otherwise, the classifier will be ran 10 more times then check the probability again.

c. Speech Recognition with Houndify

Houndify is a public platform, developed by SoundHound, to add voice-enabled interfaces that allows conversational interaction with application user using internet connection [7]. I choose this platform because it is recommended by Prof. Aarabi, and it includes fast and large-scale speech recognition.

When the user taps on voice search option from the main menu, voice search view is displayed on the screen and a microphone button is available for the user to press on and start voice recognition. Once the user taps on the microphone button, the application starts recording user's voice and pass it as audio byte stream source to Houndify voice search function which does voice to text translation in real time. This translation is displayed on the voice search layout. Once the translation is done, like image classification activity, a popup will be displayed on the screen asking whether the user is looking for the translated dish name if the name exists in application's recipe list. The screen will jump to the identified recipe view if the user confirms. An error message will show in case the translated text doesn't correspond to any recipe in the application. The Houndify developer guide [7] is a useful reference source for developing the voice recognition interface.

It is observed that the Houndify speech to text translation is very fast and accurate as long as the user speaks properly and doesn't pause between words for a long time. In addition, it has great scale of vocabulary including words that are transliterated into English from other languages. For instance, 'Mapo Tofu' is an English translation of one of my Chinese dish category and it is well recognized by Houndify.

3. Conclusion

In conclusion, the implementation of this application helps me significantly in developing android development skills and improving my understanding in deep learning.

Reference

[1] *Android ListView with ListAdapter Example | Java Tutorial Network*. (2018). *Javatutorial.net*. Retrieved 8 April 2018, from https://javatutorial.net/android-listview-with-listadapter-example

[2] *An Intuitive Explanation of Convolutional Neural Networks. (2017, May 29). Retrieved April 8, 2018, from https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/*

[3] *KDnuggets. (n.d.). Retrieved April 8, 2018, from https://www.kdnuggets.com/2017/08/convolutional-neural-networks-image-recognition.html*

[4] *A Guide to TF Layers: Building a Convolutional Neural Network | TensorFlow. (n.d.). Retrieved April 8, 2018, from https://www.tensorflow.org/tutorials/layers*

[5] *Introduction to TensorFlow Lite | TensorFlow. (n.d.). Retrieved April 8, 2018, from https://www.tensorflow.org/mobile/tflite/*

[6] *Building TensorFlow on Android | TensorFlow. (n.d.). Retrieved April 8, 2018, from https://www.tensorflow.org/mobile/android_build*

[7] *Houndify by SoundHound Inc. (n.d.). Retrieved April 8, 2018, from https://www.houndify.com/docs*