Udacity Machine Learning Nanodegree

Capstone Project

Food Shazam

Author:
Van Nhat Huy Phan

Date:
Febuary, 2019

Final Report

UDACITY

Contents

1 Definition

1.1 Project Overview

Many deep learning techniques have been used everyday to better our lives. From recognizing our friends and families automatically in photos for tagging (Google Photos), to understanding our voice commands to perform basic tasks for us (Alexa, Google Home). Imaging processing and natural languages processing are two big branches of deep learning. However, deep learning projects require a computer to run and deploy because it is computationally expensive. It is generally hard for scientist and researchers to prototype and deploy their work in deep learning personal devices like smartphones.

There are many tools has been made by big companies to make deep learning more accessible on smartphones. Here are the most popular solutions[10]:

- TensorFlow Lite (Google): An extension of the ubiquitous TensorFlow, TensorFlow Lite is a framework for translating models into more mobile-friendly versions. It focuses on low-latency, small model size, and fast execution. It's still very early in the development cycle though, so you might see mixed results.

- Core ML (Apple): Core ML is Apple's solution for deploying ML on Apple devices: it lets us design and develop ML models for Apple OS apps, and then package them into the app bundle. Core ML supports conversion from many of the popular frameworks like TensorFlow Lite and Caffe2, and recently got a major performance upgrade at Apple's 2018 WWDC.

- Caffe2Go (Facebook): Caffe2Go is based on the popular Caffe2 framework for developing deep learning models. It stems from Facebook's experience deploying ML models on mobile devices, and looks to be a promising solution whenever it's released.

In academia, there is a very interesting paper written by Ji Wang on the topic of "Deep Learning Towards Mobile Applications". In his paper, he provides an overview of the current challenges and representative achievements about pushing deep learning on mobile devices from three aspects: training

with mobile data, efficient inference on mobile devices, and applications of mobile deep learning. [12]

In this project, we want to solve a real life problem, and have the solution available on a smartphone, a device we always carry with us. To solve these kind of problems, there 2 two main approaches: use the smartphones as a sending/receiving devices and put the trained model on a computer, or put everything on the smartphones including the trained model. We will approach this problem using the latter method.

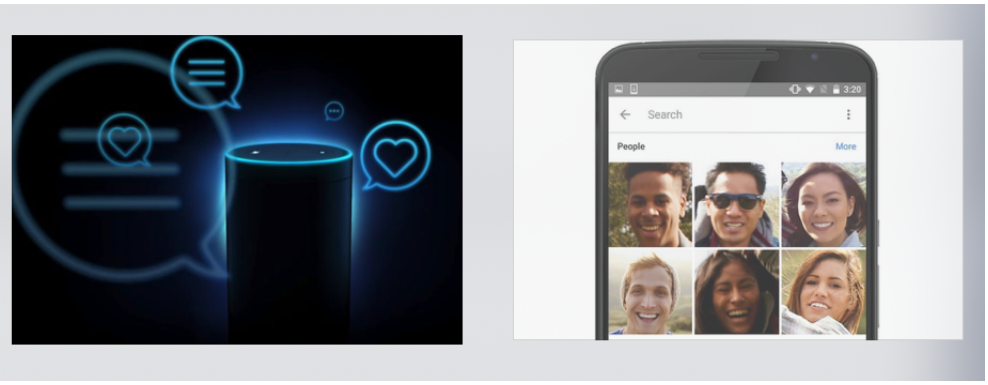

Figure 1: Deep Learning Applications [3]

1.2 Problem Statement

When we hear a song and want to know its title and artist, we can use Shazam or SoundHound to find out. These apps record the sound input by the smartphone, send it to the server, the receive the result from the server. However, when it comes to food, there is no widely known mobile solution to recognize the dish.

We will try to solve this problem by creating an app on iOS that allows us to recognize the dish just by pointing the camera at it. The app will take a picture of the dish, compare it will the trained model of 101 dishes, output the label with the highest score. This is a classification task and could be accomplished by:

- Find a reasonable size food data set with clear labels.
- Train a deep learning model on a computer to classify different dishes.

- Convert the trained model to a compatible format that can run on a iPhone.
- Create a simple app that can take pictures and display the result.

In future development the app should help user to identify the dish and see relevant information about the dish like nutrition and recipe. The difficult part of this problem is that it quite difficult to put a deep learning model on a mobile phone which has little computing power.

1.3 Metrics

Since this is a classification task and the classes are not skewed. Hence accuracy and log-loss are enough to evaluate the model:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total predictions made}} \tag{1}$$

$$LogLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij}) \tag{2}$$

where,
y_{ij} indicates whether sample i belongs to class j or not.
p_{ij}indicates the probability of sample i belonging to class j.

2 Analysis

2.1 Data Exploration

We will use a publicly accessible Food-101 dataset [1]. This dataset contains 101 food categories, with 101 thousands images. There are 1000 images for each class. On purpose, the training images were not cleaned, and thus still contain some amount of noise. This comes mostly in the form of intense colors and sometimes wrong labels. All images were re scaled to have a maximum side length of 512 pixels.

Figure 2: Overview of the Food-101 Dataset

2.2 Exploratory Visualization

These are all 101 categories of the Food-101 dataset. The dataset consist of 101 folders, each folder contains all images for a particular dish. In each folder there are also many images with false color, intense color, black and white, and also there are many images that do not belong to the group at all. This can be beneficial because it gives us a reasonable amount of noise a real-world model might have to face.

Also the total size of the dataset is 5GB, which is very large. When we split that data in to 80-20 for training and validation, we end up with 4GB of images for training the model. This will make the training part of the task much more challenging and hence demand a multi GPU setup. Normal laptops might take days to train the model. The cheapeast option is to rent an Amazon EC2 instance. However, that amount of data is necessary to give the model a better performance than traditional machine learning algorithms

```
food-101  food-101.tar.gz
ubuntu@ip-172-31-46-160:~/data/101food$ cd food-101/
ubuntu@ip-172-31-46-160:~/data/101food/food-101$ ls
images  license_agreement.txt  meta  README.txt
ubuntu@ip-172-31-46-160:~/data/101food/food-101$ cd images/
ubuntu@ip-172-31-46-160:~/data/101food/food-101/images$ ls
apple_pie            crab_cakes              gyoza                 poutine
baby_back_ribs       creme_brulee            hamburger             prime_rib
baklava              croque_madame           hot_and_sour_soup     pulled_pork_sandwich
beef_carpaccio       cup_cakes               hot_dog               ramen
beef_tartare         deviled_eggs            huevos_rancheros      ravioli
beet_salad           donuts                  hummus                red_velvet_cake
beignets             dumplings               ice_cream             risotto
bibimbap             edamame                 lasagna               samosa
bread_pudding        eggs_benedict           lobster_bisque        sashimi
breakfast_burrito    escargots               lobster_roll_sandwich scallops
bruschetta           falafel                 macaroni_and_cheese   seaweed_salad
caesar_salad         filet_mignon            macarons              shrimp_and_grits
cannoli              fish_and_chips          miso_soup             spaghetti_bolognese
caprese_salad        foie_gras               mussels               spaghetti_carbonara
carrot_cake          french_fries            nachos                spring_rolls
ceviche              french_onion_soup       omelette              steak
cheesecake           french_toast            onion_rings           strawberry_shortcake
cheese_plate         fried_calamari          oysters               sushi
chicken_curry        fried_rice              pad_thai              tacos
chicken_quesadilla   frozen_yogurt           paella                takoyaki
chicken_wings        garlic_bread            pancakes              tiramisu
chocolate_cake       gnocchi                 panna_cotta           tuna_tartare
chocolate_mousse     greek_salad             peking_duck           waffles
churros              grilled_cheese_sandwich pho
clam_chowder         grilled_salmon          pizza
club_sandwich        guacamole               pork_chop
ubuntu@ip-172-31-46-160:~/data/101food/food-101/images$
```

Figure 3: All categories Food-101 Dataset

Figure 4: Noise in the Apple Pie label

2.3 Algorithms and Techniques

We will use a Deep Convolutional Neural Networks called AlexNet as core of this project [7] to classify images. AlexNet is arguable the most popular neural networks architecture when it comes to image classification because AlexNet performs surprisingly well in the ImageNet Large Scale Visual Recognition Challenge. It achieved top-1 and top-5 error rates of 37.5 percent and 17.0 percent which is considerably better than the previous state-of-the-art.

The reason we use AlexNet instead of building a neural network from the ground up like in the Dog-Classification because it performs significantly better and is optimized for a multi GPU system. The NVIDIA Deep Learning GPU Training System even has a pre-trained for AlexNet which is very useful because we have to train 101 000 images.

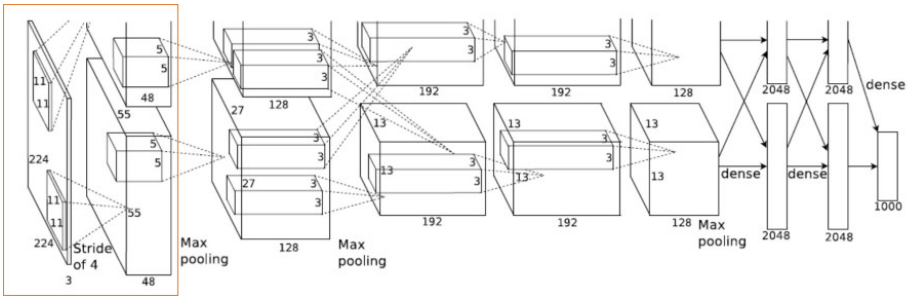

Figure 5: Architecture of AlexNet

AlexNet has a total of 8 layers: 5 convolutional and 3 fully connected layers. There are also pooling and activation in between reduces computation and controls over fitting. Our input to AlexNet is 256*256*3 matrix (size of image) and output is a vector of length 101 (total labels to dataset).

The 1st layer of AlexNet is a convolutional layer. Out data input size is 224 x 224 x3. The 1st layer does not perform a direct convolution but instead use a General Matrix Multiply, Fast Fourier Transform and Winograd to performs the task. The output fed to the 2nd layer is a of size 55 x 55 x 96, but this split into 2 parts of each GPU so 55 x 55 x 48.

The 2nd layer Max Pooling then an convolution layer. It uses the maximum value from each of a cluster of neurons at the prior layer. The purpose of a max pooling layer is down-sample an input representation (hidden-layer output matrix), reducing its dimensionality and allowing for assumptions to

be made about features contained in the sub-regions binned. Layer 2 output is 27 x 27 x 256 matrix split across 2 GPUs – So 27 x 27 x 128 for each GPU.

The 3rd, 4th, and 5th layer follow a similar structure.

The 6th layer is, however, different. It is a fully connected layer with drop out layers to reduce overfitting. Without dropout, our network exhibits substantial overfitting. Dropout roughly doubles the number of iterations required to converge.

The weights in AlexNet are updated based on the following rule:

$$\begin{aligned} v_{i+1} &:= 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i} \\ w_{i+1} &:= w_i + v_{i+1} \end{aligned}$$

where i is the iteration index, v is the momentum variable, ϵ is the learning rate, and δL/δW is the average over the i-th batch Di of the derivative of the objective with respect to w, evaluated at wi.

AlexNet is extremely useful because it can take advantage of a multiple GPUs setup. This is a break down of how AlexNet works with multiple GPUs [4]:

- Copy convolution layers into different GPUs;
- Distribute the fully connected layers into different GPUs.
- Feed one batch of training data into convolutional layers for every GPU (Data Parallel).
- Feed the results of convolutional layers into the distributed fully connected layers batch by batch (Model Parallel)/
- When the last step is done for every GPU. Backpropogate gradients batch by batch and synchronize the weights of the convolutional layers.

When we have the trained model, we could convert this model to a iOS-compatible format using Apple CoreML [6] tool. Then finally make a simple app with a UIView and a UILabel to display the picture and the result.

Figure 6: How CoreML works

2.4 Benchmark

We can use the accuracy of the several models the author of the Food-101 paper as our benchmark [2].

Model	Accuracy
Random Forest	50.76%
Bag-of-Words Histogram	38.83%
Improved Fisher Vectors	49.40%

We think our Deep Learning Model using AlexNet can easily beat that. The reason being traditional model like Random Forrest and SVM do not scale very well with big data set. As the dataset gets bigger the performance on these traditional algorithms stay the same. On the other hand, deep learning proves that it scales suprisingly well with big dataset.

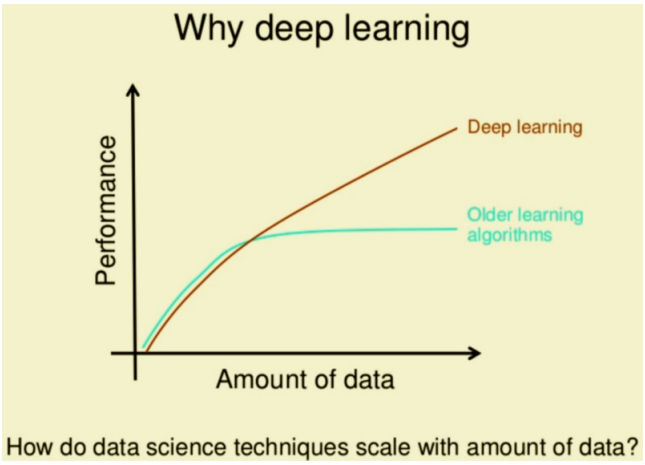

Figure 7: Deep Learning vs Traditional Machine Learning [8]

3 Methodology

3.1 Data Preprocessing

The provided dataset does not need to be pre-process. The reason is that training data is not perfect and has some noise. This is intentional by the developer of the dataset. We think it make senses and we should not clean the data because the advance computer vision algorithm must be able to deal with weakly labeled data. The noise comes in the form of wrong labels and wrong color.

Besides, with the help of NVIDIA Deep Learning GPU Training System we can take images and feed directly to the first layer of AlexNet. These 2 reasons simplify the need to preprocess the data

3.2 Implementation

Since training a convolutional neural networks to classify 101,000 images is a very computational expensive task, we must rent an Amazon EC2 instance to train the network. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers [5].

Since Amazon EC2 is widely popular among deep learning developers,

there are many pre-configured instances of Amazon EC2, we will use one called "big-vision-digits" with NVIDIA DIGITS already set up to simplify the task [9].

Next, we will download the Food-101 data set to the instance. We will resize the images from their original size to 256*256 to reduce the training time. Then we will specify the percentage of the dataset use for validation which is 25 percent.

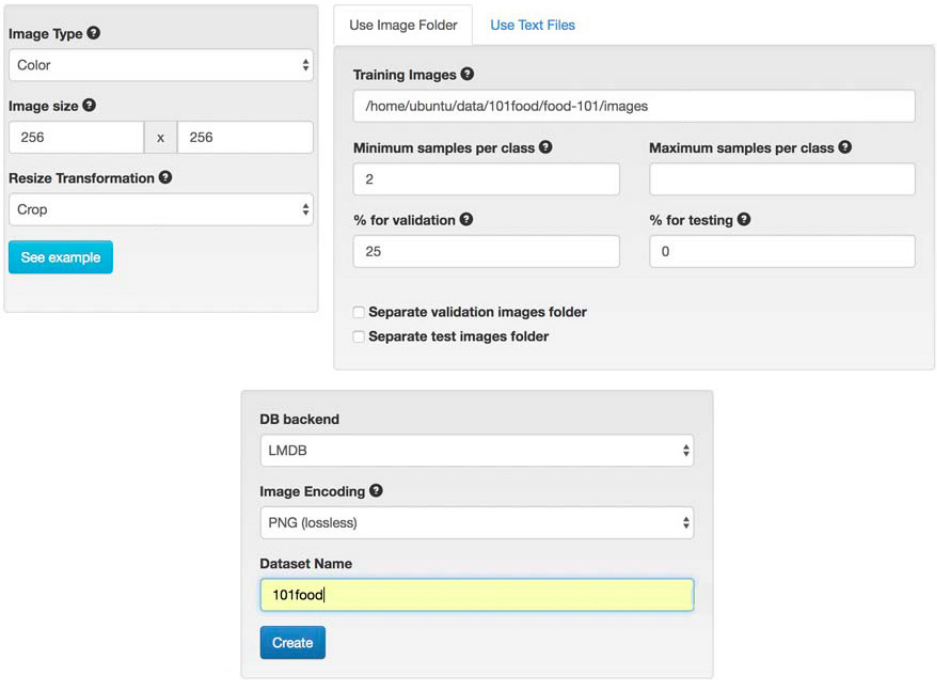

Figure 8: Cropping and Splitting

Now that everything has been set up. we are ready to train the model. This is computational expensive task, even for an Amazon EC2 instance it takes more than 4 hours to train the data.

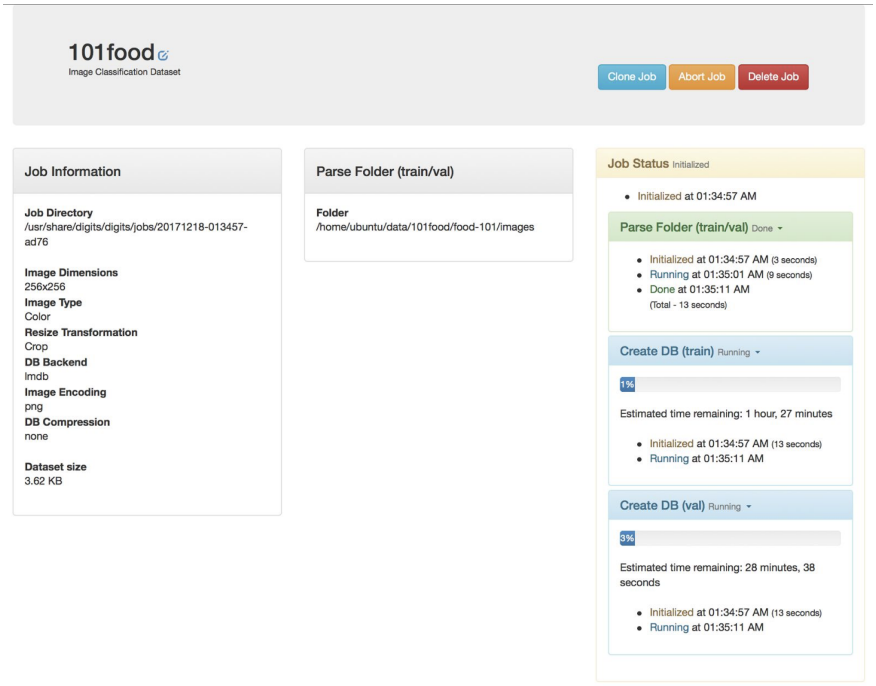

Figure 9: Training

Next I download the model, converted it to a CoreML model to use in the iOS App. The app is pretty straight forward. It has an image view in the middle, a button to chose image, and a text label to display the result. [11]

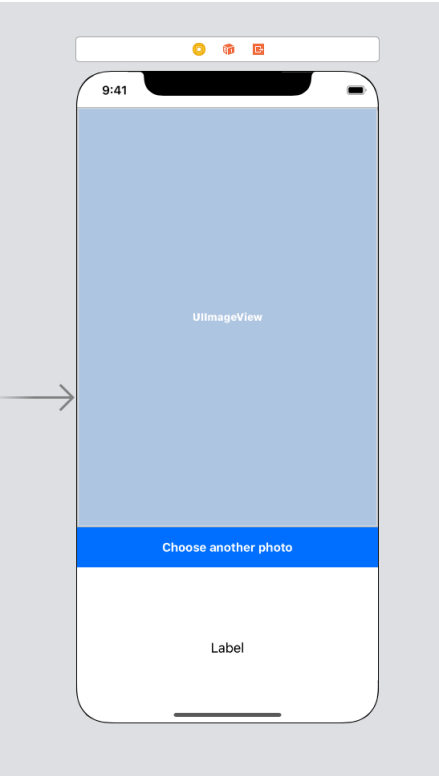

Figure 10: The final app layout

3.3 Refinement

The initial accuracy was about 60 percent. Givien the black-box property of this project there is many parameters we could tweak. However, this time I decided to not crop the images from it original size. I also change the encoding from JPEG to PNG (lossless). The final result comes out with the accuracy of 64 percent. Not a significant improvement but it is still better than the original model.

4 Results

4.1 Model Evaluation and Validation

We test the the model with a images it has not seen. The results are very good. The model correctly predict the label.

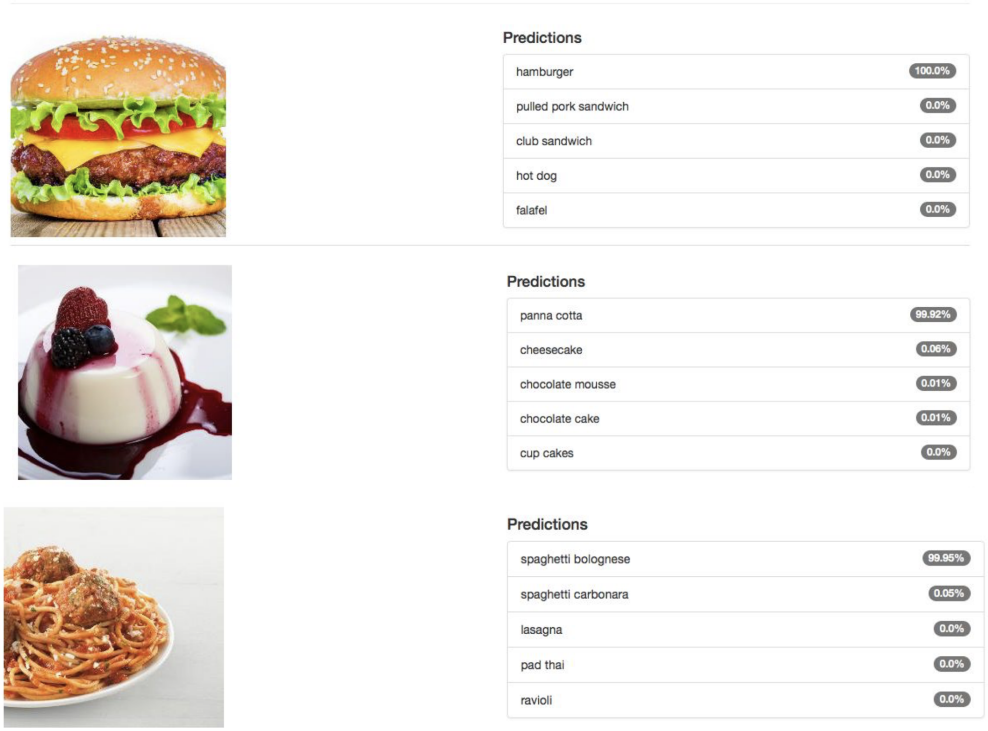

Figure 11: Testing with a single image feed from computer

When testing with a image that's is not in the 101 labels, the model gives the results it thinks that are the closest.

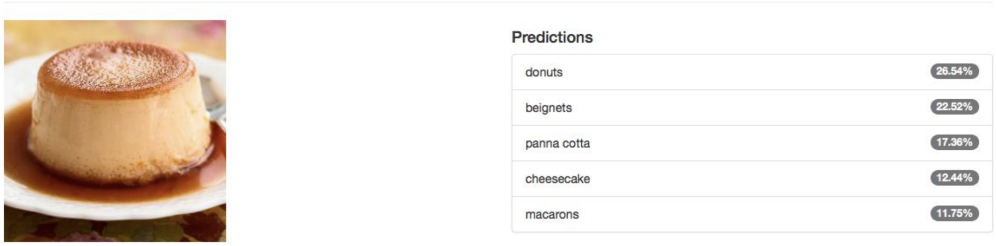

Figure 12: Testing with dish not in the 101 labels set

The overall accuracy of the model is 64 percent. This is acceptable and align with our expectation.

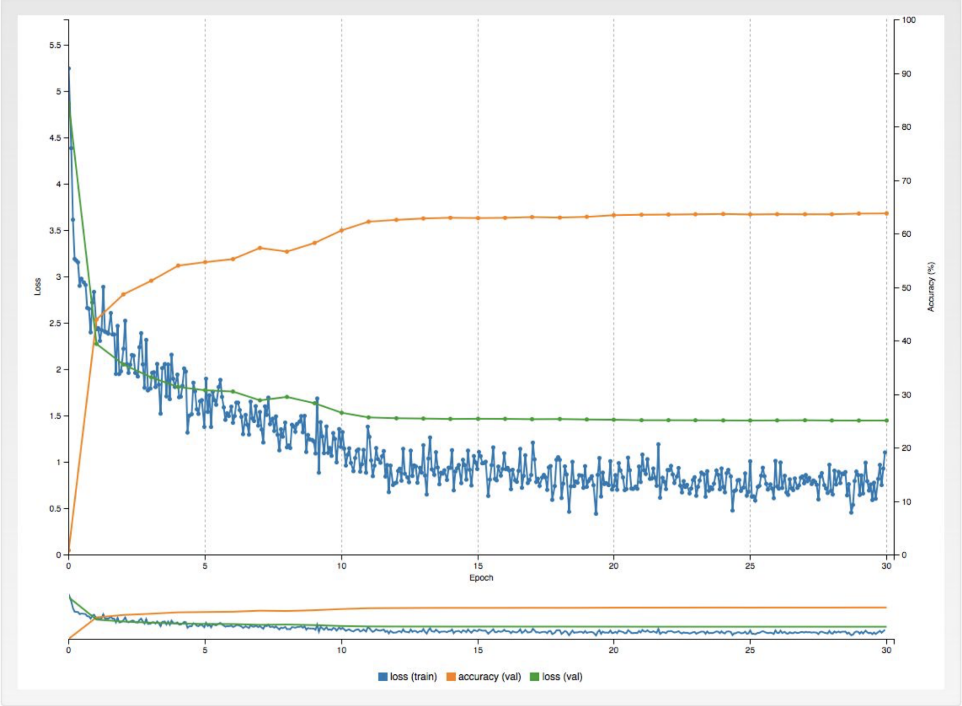

Figure 13: Accuracy and LogLoss of the model

We could test the model real-world performance by setting up this experiment:

- Pick 10 random dishes in the database.

- For each dish, find 10 photos on Google.
- Download the photos, and test on a real iPhone.
- Record the result and repeat.

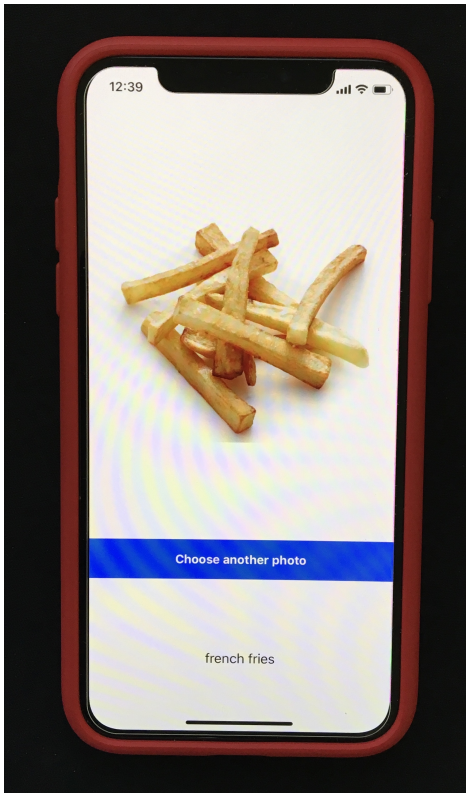

Figure 14: Test on iOS Device

Dish	Accuracy
Hamburger	90%
Pizza	70%
Hot Dog	80%
Chicken Wings	60%
French Fries	100%
Chocolate Cake	80%
Donut	90%
Spaghetti Bolognese	70%
Caesar Salad	80%
Chicken Wings	60%

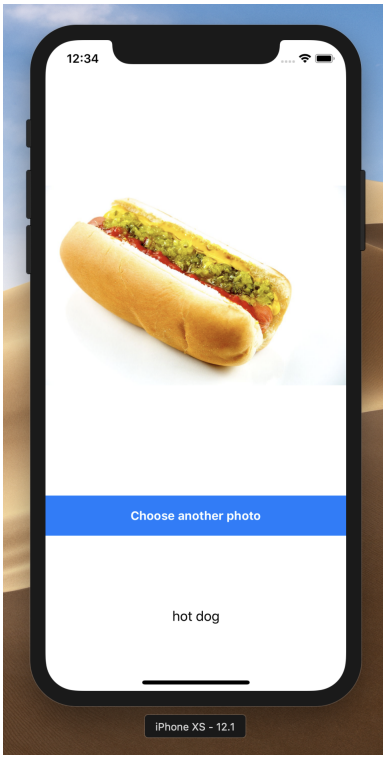

Figure 15: Final App

4.2 Justification

The final accuracy of the model is 64 percent and is much better than the benchmark model which is only 50 percent. The results and the solution are significant enough to have solved the problem posed in the project. The model perform better in real world (when deployed in an iPhone) in my opinion. The reason because the size of image fed to the model is much larger about 2048*2048 pixels and the color is accurate.

5 Conclusion

5.1 Free-Form Visualization

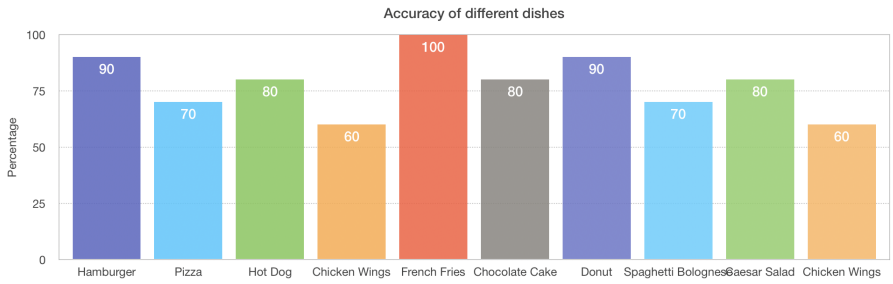

Figure 16: Real world accuracy visualization

5.2 Reflection

This is a classic classification problem. However it is computationally expensive to train a model to classify hundred of dishes. Even though I tried to use to most advanced methods and rented a powerful computer, it still took me 4 hours to train.

The app performs really well in real life situations. However, there are some limits. The app right now can only classify about 100 dished, that's not enough. To publish this app to the App Store I think the app should at least be able to recognize more than 1000 dishes. That requires a larger training set which usually not free and much harder to train.

I wasn't able to tweak many parameters when I used the AlexNet convolutional network to train my model to better the accuracy. I tried to tweak

the learning rate and discover that the greater the learning rate the lesser time it takes to train the model, but the accuracy is poorer.

5.3 Improvement

To improve the accuracy I think the biggest limitation is the dataset. With bigger dataset and bigger images I think we can improve the accuracy significantly. However, this comes as a much greater training time. We can also improve the accuracy by using better algorithm like:

- GoogLeNet
- VGGNet
- GoogLeNet
- Resnet

Year	CNN	Developed By	Error rates	No. of parameters
1998	LeNet	Yann LeCun et al		60 thousand
2012	AlexNet	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	15.3%	60 million
2013	ZFNet	Matthew Zeiler, Rob Fergus	14.8%	
2014	GoogLeNet	Google	6.67%	4 million
2014	VGGNet	Simonyan, Zisserman	7.3%	138 million
2015	ResNet	Kaiming He	3.6%	

Figure 17: Comparision between different CNN Architecture

The app performs really well in real life situations. However, there are some limits. The app right now can only classify about 100 dished, that's

not enough. To publish this app to the App Store I think the app should at least be able to recognize more than 10,000 dishes around the world. That kind of dataset is not available right now to the public. However I'm quite happy with the current result with 64 percent accuracy.

References

[1] Lukas Bossard. *Food-101 – Mining Discriminative Components with Random Forests*. URL: https://www.vision.ee.ethz.ch/datasets_extra/food-101/.

[2] Lukas Bossard. *Food-101 – Mining Discriminative Components with Random Forests*. URL: https://www.vision.ee.ethz.ch/datasets_extra/food-101/static/bossard_eccv14_food-101.pdf.

[3] Alan Boyle. *Amazon is widening the repertoire for Alexa AI voice assistant*. URL: https://www.geekwire.com/2018/getting-know-amazon-widening-repertoire-alexa-ai-voice-assistant/.

[4] Hao Gao. *A Walk-through of AlexNet*. URL: https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637.

[5] Amazon Inc. *Amazon Elastic Compute Cloud (Amazon EC2)*. URL: https://aws.amazon.com/ec2/.

[6] Apple Inc. *Core ML - Integrate machine learning models into your app*. URL: https://developer.apple.com/documentation/coreml.

[7] Alex Krizhevsky. *ImageNet Classification with Deep Convolutional Neural Networks*. URL: http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf.

[8] Sambit Mahapatra. *Why Deep Learning over Traditional Machine Learning?* URL: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063.

[9] Satya Mallick. *Deep Learning using NVIDIA DIGITS 3 on EC2*. URL: https://www.youtube.com/watch?v=QZaAcl_F9R0.

[10] Jesus Rodriguez. *What's New in Deep Learning Research: Mobile Deep Learning with Google MnasNet*. URL: https://blog.algorithmia.com/machine-learning-and-mobile-deploying-models-on-the-edge/.

[11] Audrey Tam. *Core ML and Vision: Machine Learning in iOS 11 Tutorial*. URL: https://www.raywenderlich.com/577-core-ml-and-vision-machine-learning-in-ios-11-tutorial0.

[12] Ji Wang. *Deep Learning Towards Mobile Applications*. URL: https://arxiv.org/pdf/1809.03559.pdf.