Guess A Sketch: Applying Machine Learning To Predict The Object User Sketches

Mo Alnuaimat, Abrhaley Aynialem, Subhechha Bista
Machine Learning - CS582
Maharishi University of Management
March 2017

INTRODUCTION

Object recognition is an important application in the field of machine learning and computer vision. As a human, we intuitively know that pictures have a conceptual structure.We recognize the idea of an object no matter what surface it is on or which direction it is faced to. We don't have to relearn the idea of the object for all possible surfaces it can appear on or all possible directions it can be viewed from. But it is hard for artificial neural networks to determine that moving the object around in the picture does not make it something different. With the evolution of machine learning throughout the years, convolutional neural networks can be used as one of the good approaches to solving these object recognition related problems.

Guess A Sketch is a system designed for predicting objects the users are trying to sketch. The basic idea here is to provide users with a canvas to draw the sketches in and try to predict what object the sketch resembles. Based on the prediction the system gives feedback to the user of what it thinks the object is. As the user modifies the sketch and submits the sketch to the system, it updates the prediction. The user is also able to give positive or negative feedback to the system to tell if the prediction was correct or not. Positive feedback is when the system prediction is correct. If not, they can tell the system what object they are trying to draw. This feedback mechanism increases the confidence of the system in case of positive feedback or learns from it in the case of negative feedback.

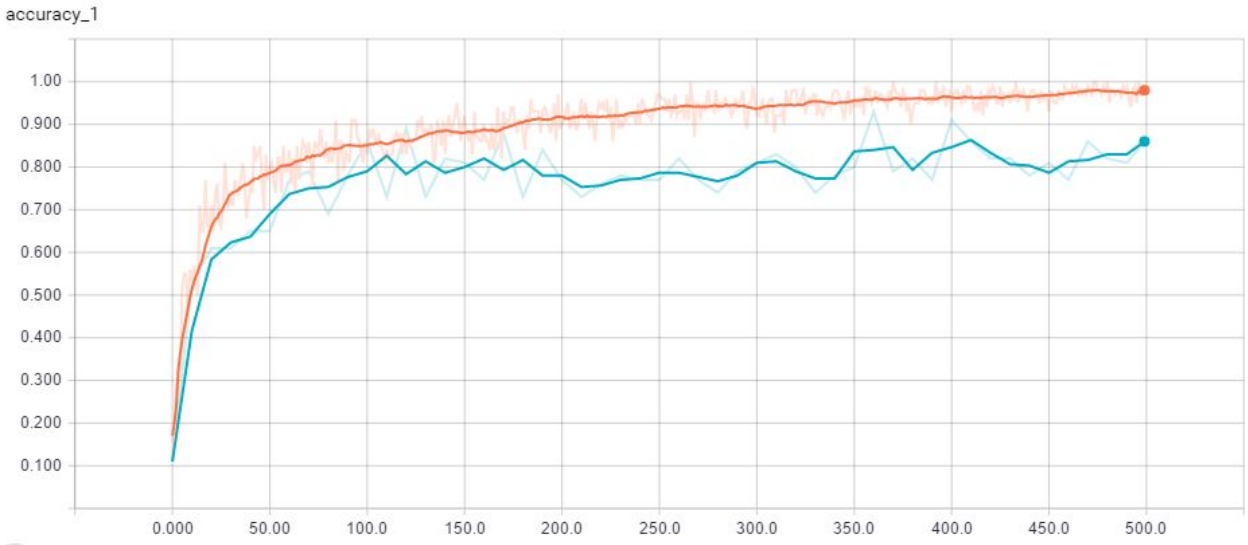
System main prediction component was built using Tensorflow inception engine, a convolutional neural network engine created by google.

DATA SETS

We created an archive of 12 different categories of common objects like line, square, cat, house. Under each of these categories initially, we had around 30 images. Following the supervised learning workflow, these images (input) along with their labels (targeted output) were fed to the system. The training phase analyzes all the images and creates bottleneck files for each of these images. We put aside 10% of these images to be used for testing. Once trained we test the classifier with these test dataset.

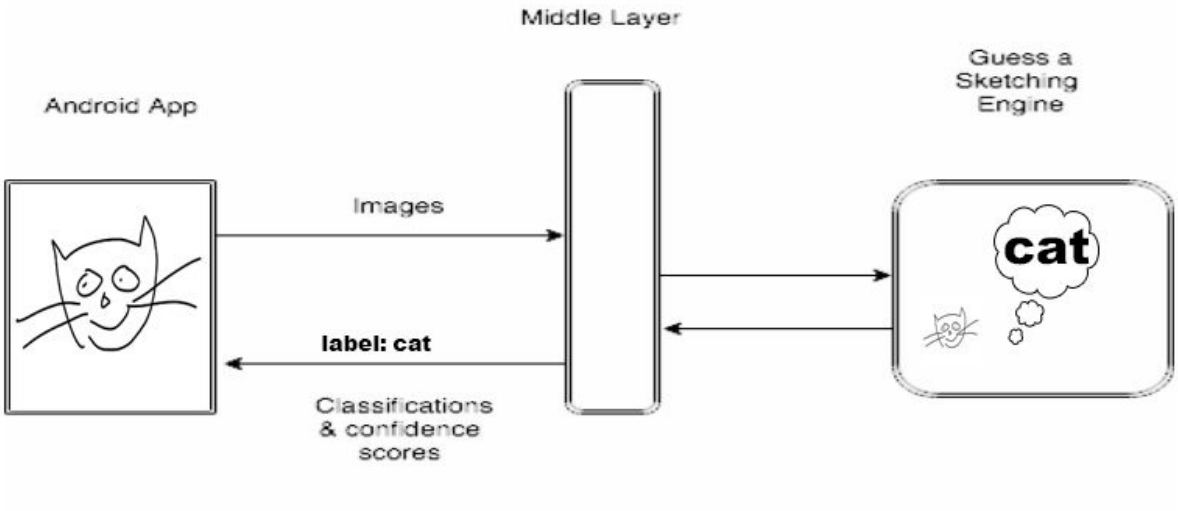
```
2017-03-21 13:39:43.484922: Step 470: Train accuracy = 100.0%
2017-03-21 13:39:43.485006: Step 470: Cross entropy = 0.317017
2017-03-21 13:39:43.676009: Step 470: Validation accuracy = 86.0% (N=100)
2017-03-21 13:39:45.089407: Step 480: Train accuracy = 100.0%
2017-03-21 13:39:45.089471: Step 480: Cross entropy = 0.308524
2017-03-21 13:39:45.182555: Step 480: Validation accuracy = 82.0% (N=100)
2017-03-21 13:39:46.076821: Step 490: Train accuracy = 99.0%
2017-03-21 13:39:46.076879: Step 490: Cross entropy = 0.324888
2017-03-21 13:39:46.167963: Step 490: Validation accuracy = 81.0% (N=100)
2017-03-21 13:39:46.971571: Step 499: Train accuracy = 98.0%
2017-03-21 13:39:46.971652: Step 499: Cross entropy = 0.307822
2017-03-21 13:39:47.070294: Step 499: Validation accuracy = 86.0% (N=100)
Final test accuracy = 83.3% (N=36)
```

While training, the classifier in each epoch checks the training accuracy, cross-entropy and validation accuracy. Training accuracy is the percentage of training batch that was labeled with the category. Validation accuracy shows the percentage of correctly labeled images. Cross-entropy in each epoch is when some of the images of the dataset are reused by the classifier for both training and validation. The final test accuracy after completion of training was 83.3 for the dataset we used.



IMPLEMENTATION

We created 3 layers of interaction, following the SOA (Service Oriented Architecture), We chose this approach because it would help us separate different components of the system as well as easily scale the system up if the number of users grows.

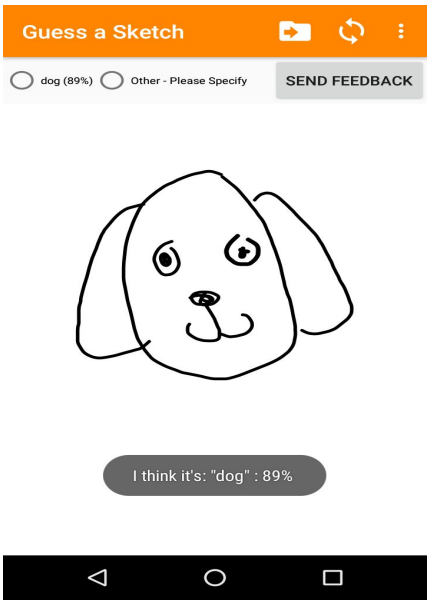


Component diagram - interaction between different layers of the system

As you can see in the top diagram, We had 3 layers, the first one is the Guess a Sketch Android App, we created an android app with a drawing canvas based on Valerio Bozzolan's open source project called Acrylic Paint (https://github.com/valerio-bozzolan/AcrylicPaint) it is GNU license. The Android app was modified such that it doesn't save sketches locally, but instead, send them over the internet in a POST HTTP request to the middle layer. Which in turn would return a JSON format response with labels it predicts would match this sketch.

Guess a Sketch Android app would then display those predictions to the user in a list of radio buttons, There also would be an option to suggest a different label. If the predictions provided by the Middle layer were correct and the user selects one of them, then that would be a positive feedback for the system, we simply guessed it right, this positive feedback would be used later on to retrain the engine and increase the confidence of this prediction.

If the user didn't find the label they were expecting, then they can choose the "Other (Please specify)" button, which would by then asks them to enter a label for the sketch the drew. This is called "Negative feedback", this Label along with the sketch would be sent to the middle layer and would be used later on to retrain the system and introduce a new class (label) if number of images submitted by users matches a certain threshold (30 images, for now, it's a requirement by tensorflow), In order to help faster builds for the Android app as well as better dependency handling we used a software project management tool and build management called Gradle (https://gradle.org/)



Guess a Sketch Android App -
Along with the predicted labels

The middle layer is Java based and uses Jersey Java library (https://jersey.java.net/) to provide a RESTFul web service for the system. It's deployed on Apache Tomcat (http://tomcat.apache.org/) Application server and built using Maven software project management (https://maven.apache.org/), Maven helps to build project much faster and provides dependencies resolving feature. Middle layer also uses a special logic to sanitize illegal input sent by users when asked about labels as this might cause a security threat.

Functionalities of Middle layers:

1. Guess a sketch, in this API call, the android app would send the image binary data, middle layer would forward this binary data along with a specially crafted HTTP POST request to the Guess-Sketch Engine, This engine would reply with a JSON string containing a list of predictions matching that sketch. Middle layer would then pass this JSON string back to the Android app.

2. Positive Feedback: Android app sends the binary data of the image along with the label predicted by the Guess a sketch engine earlier, this image data along with that label would be kept inside a positive feedback folder and would be used later to retrain the system. Retraining would happen in this case by copying over all new images from the positive feedback folder to their appropriate locations inside the dataset folder and run the retraining script.

3. Negative Feedback: Android app sends the binary data of the image along with the label entered by the user, this image data along with that label would be kept inside a

negative feedback folder and would be used later to retrain the system once the number of images reaches a threshold (30 images in this case). Retraining would happen copying over this label-named new folder along with images inside it to the dataset folder and run the retraining script.

Guess A Sketch Engine is python based FastCGI script that uses TensorFlow (https://www.tensorflow.org/) inception engine to predict images sent by the middle layer and responds back in a JSON format string.

Tensorflow initially comes untrained, we trained that system by drawing sketches for 12 different classes (refer to Dataset section above), once trained the system generates a bottleneck files, labels file and a .pb file which acts like a knowledge database that tensorflow engine can use later on to make predictions.

Guess a Sketch Engine was deployed as a FastCGI script, using Apache FastCGI Module (https://httpd.apache.org/mod_fcgid/) the benefit of this approach is that we will always have parameterized number of script instances loaded in memory and waiting to serve requests. We also deploy this on Apache web server (https://httpd.apache.org/)

It also used several python libraries in order to achieve the functionality needed:

- Flipflop (https://pypi.python.org/pypi/flipflop) : FastCGI wrapper for WSGI applications
- Werkzeug (http://werkzeug.pocoo.org/) : WSGI utility library for Python that simplifies handling of HTTP Requests/Response and supports multipart/form-data uploads.
- Tensorflow : The tensorflow inception engine python interface
- JSON : to produce output in JSON format

Full source code of the project can be found on GitHub on this link (https://github.com/nuaimat/GuessSketch)

FEATURES

- Guess sketches drew by users
- A feedback mechanism (positive and negative) that is used to either increase the confidence of the system or retrain the guess sketch engine.
- Classification is open to users. We do not limit the classes of objects they can draw.
- Since retraining process is accumulative it does not take much time to retain the system.
- Service Oriented Architecture gives us the ability to scale up the system for a larger number of users.

CONCLUSION AND FUTURE WORK

Convolutional Neural Network is a really interesting technology. It can learn easily and gives a very good prediction. Convolutional neural networks usually require a large amount of training data in order to avoid overfitting. A common technique is to train the network on a larger data set from a related domain. Once the network parameters have converged, an additional training step is performed using the in-domain data to fine-tune the network weights. This allows convolutional networks to be successfully applied to problems with small training sets. Image recognition and dealing with convolutional neural network was made easier by TensorFlow.

We can also provide a web interface and an IOS app. This application can be turned into a game like solution for kids to teach not only drawing but also words in different languages. It can be useful for graphic designers to quickly tag their drawings. Furthermore, the main goal in future is to improve the performance of the system. This can be achieved by configuring Tensorflow to use GPU processing power to predict the image. instead of CPU and using a server with more resources, Or use Google Cloud Tensorflow engine (https://cloud.google.com/ml-engine/)

REFERENCES

- **https://www.tensorflow.org/get_started/get_started**
- **https://www.slideshare.net/JenAman/large-scale-deep-learning-with-tensorflow**
- **https://www.slideshare.net/tw_dsconf/tensorflow-tutorial**
- **http://www.python.org/about/gettingstarted/**
- **https://www.electricmonk.nl/docs/apache_fastcgi_python/apache_fastcgi_python.html**
- **http://stackoverflow.com/questions/20322528/uploading-images-to-server-android**
- **https://jersey.java.net/documentation/latest/getting-started.html**
- **https://www.ibm.com/developerworks/library/os-eclipse-tomcat/**