TensorFlow Report
--Machine Learning Models for Segment Objects Detection

by Bowen Qin

1 Introduction

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Companies like Airbnb, china Mobile, Coca Cola, GE Healthcare, Google, Intel, Lenovo, PayPal, Twitter, WPS Office and so on are using TensorFlow to optimize their products.

Developed and maintained by Google Brain, Google's artificial intelligence team, TensorFlow has multiple projects including TensorFlow Hub, TensorFlow Lite, and TensorFlow Research Cloud, as well as various Application Programming Interfaces. Since 2015 from November 9th, TensorFlow became an open source under the Apache 2.0 open source license.

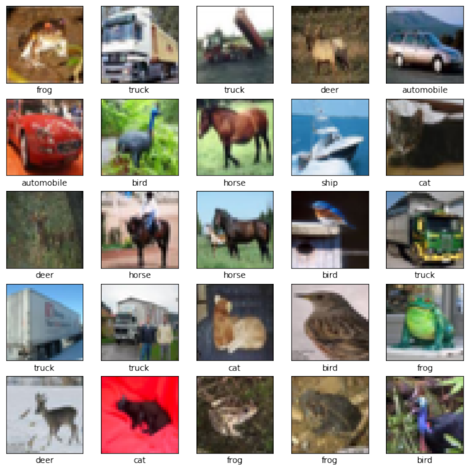
TensorFlow provides a collection of workflows to develop and train models, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use. Currently it supports C, Python, JavaScript, C++, Java, Go, and Swift. Other support language including C#, Haskell, Julia, Ruby, Rust, and Scala are still under development.

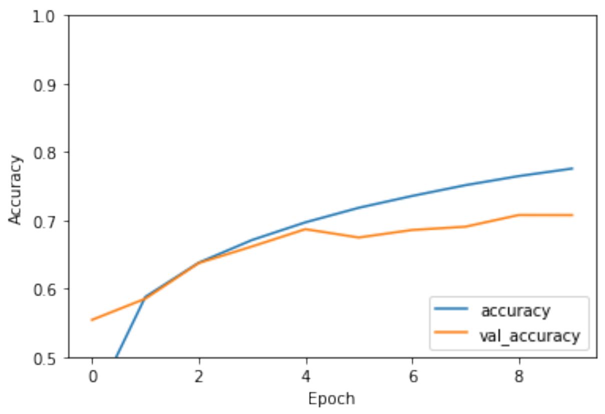
In the following paragraphs, we mainly focus on some image detection aspects in TensorFlow.

2 Analysis of results

2.1 Convolutional Neural Network (CNN)

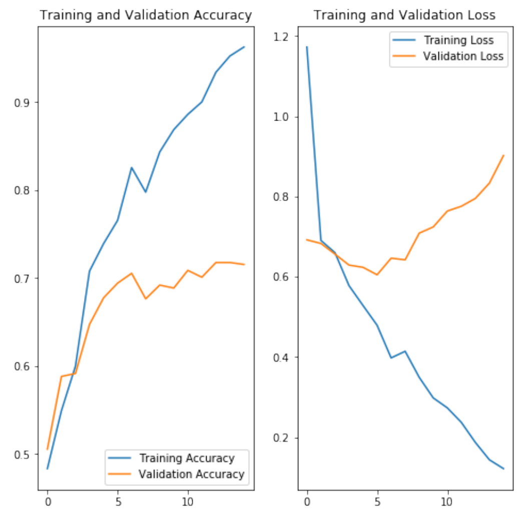
Users can use Convolutional Neural Network (CNN) to classify CIFAR images in TensorFlow.. The CIFAR10 dataset contains 60,000 color images in 10 classes, with 6,000 images in each class. The dataset is divided into 50,000 training images and 10,000 testing images. The classes are mutually exclusive and there is no overlap between them. For simple CNN, users can achieve a test accuracy of over 70%.



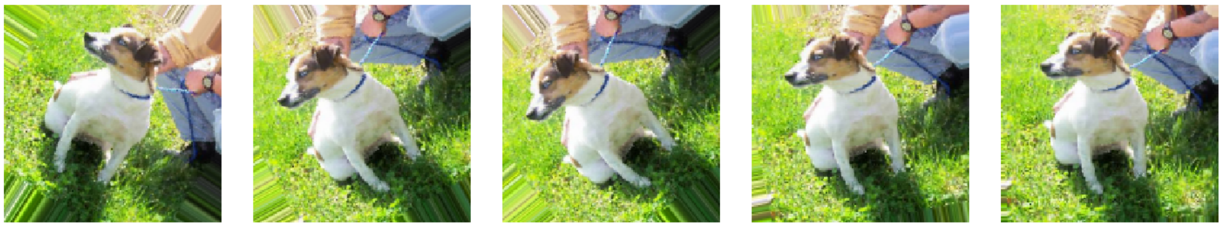


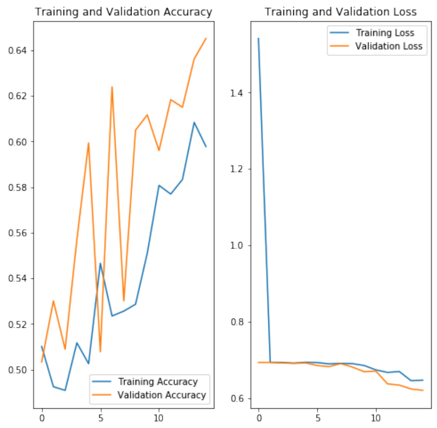
2.2 Image Classification

Users can use TensorFlow to classify cats or dogs from images. It builds an image classifier using a tf.keras.Sequential model and load data using tf.keras.preprocessing.image.ImageDataGenerator. By examining and understanding data, building an input pipeline, building the model, training the model, testing the model, improving the model and repeating the process, users can realize basic image classification.



As you can see from the plots, training accuracy and validation accuracy are off by large margin and the model has achieved only around 70% accuracy on the validation set. To fight overfitting in the training process, users can use data augmentation(Randomly rotate the image, Apply zoom augmentation, etc.) and add dropout to our model.



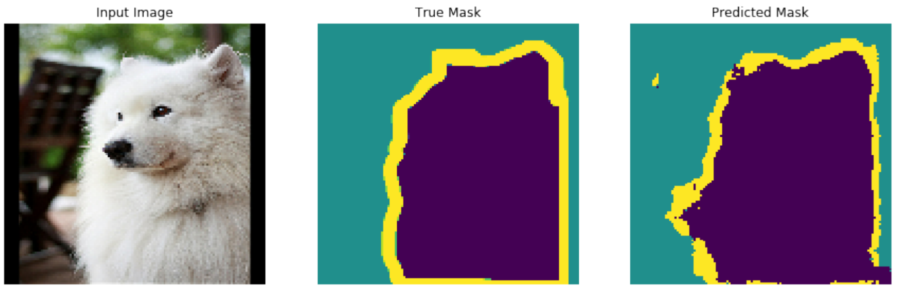


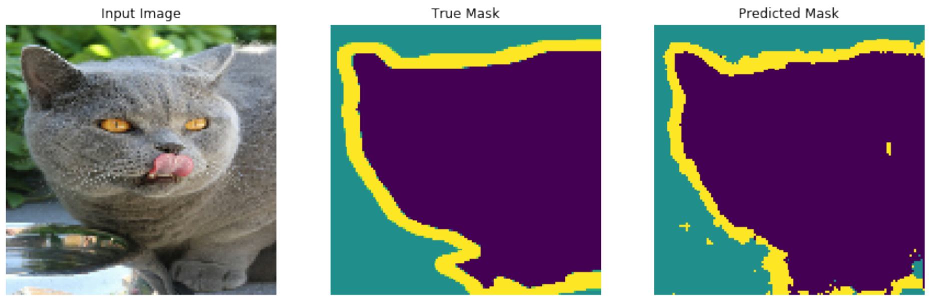
As you can see from the plots, there is significantly less overfitting than before. The accuracy should go up after training the model for more epochs.

2.3 Image Segmentation

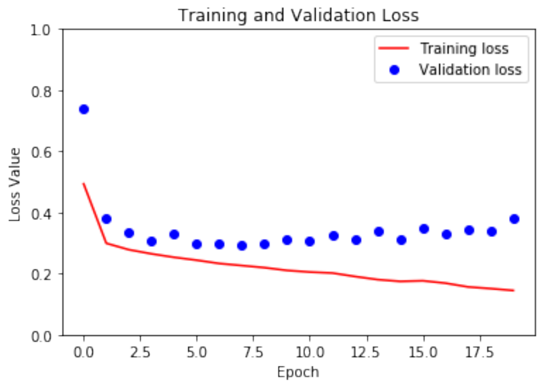
Image segmentation is to train a neural network to output a pixel-wise mask of the image. This helps in understanding the image at a much lower level, i.e., the pixel level. Image segmentation has many applications in medical imaging, self-driving cars and satellite imaging to name a few.

The dataset is the Oxford-IIIT Pet Dataset. It consists of images, their corresponding labels, and pixel-wise masks. The masks are basically labels for each pixel. Each pixel is given one of three categories :Class 1 : Pixel belonging to the pet; Class 2 : Pixel bordering the pet; Class 3 : None of the above/ Surrounding pixel.





The model being used here is a modified U-Net. A U-Net consists of an encoder (downsampler) and decoder (upsampler). In-order to learn robust features, and reduce the number of trainable parameters, a pretrained model can be used as the encoder. Thus, the encoder for this task will be a pretrained MobileNetV2 model, whose intermediate outputs will be used, and the decoder will be the upsample block already implemented in TensorFlow.



As you can see from the plots, the training loss will drop after increasing the Epoch value.

2.4 Pros and cons

Pros:

- TensorFlow support multiple programming languages, such as Python, C++, Java, R, and Go.
- Easy to use: There are many great tutorials on the official web and Internet.
- TensorFlow has better computational graph visualizations.
- Can work on a variety of platforms, even mobile and distributed platforms.
- It is supported by all cloud services (AWS, Google, and Azure).
- Keras, a high-level neural network API, has been integrated with TensorFlow.
- Have very good community support.
- TensorFlow is more than just a software library. It is a suite of software including TensorFlow, TensorBoard and TensorServing.

Cons:

- Frequent releases. TensorFlow upgrades quickly.
- Lower computation speed. TensorFlow focus on the production environment rather than the performance.

3 Recommendations

First of all, TensorFlow helps users easily build models. Users can build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging. Whether users are beginners or experts, they all can find proper tutorials on the official TensorFlow web and self-learn the techniques.

Secondly, TensorFlow can be used for robust ML production anywhere. Users can easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language they use.

Moreover, TensorFlow offers powerful experimentation for research. Via a simple and flexible architecture, users are able to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

4 Conclusions

TensorFlow is a powerful open source platform for machine learning. Users can quickly get to know TensorFlow via the tutorials on the web no matter they are beginners or experts in ML. As an open source in deep learning library, TensorFlow allows deep neural network computing to be deployed on any number of CPUs or GPUs on a server, PC or mobile device, using only one TensorFlow API. TensorFlow can automatically derivate, open source, support multiple CPU/GPUs, have pre-trained models, and support common NN architectures such as recurrent neural networks (RNN) and convolutional neural networks (CNN).) and Deep Trusted Network (DBN).

5 References

[1] https://www.tensorflow.org
[2] https://www.tensorflow.org/tutorials/images/cnn
[3] https://www.tensorflow.org/tutorials/images/classification
[4] https://www.tensorflow.org/tutorials/images/segmentation
[5] https://modelzoo.co/framework/tensorflow