CMPT 412

Assignment 6 Report (4 Late Days Used)

Maoshun Shang

301280479

# Introduction

This assignment mainly focuses on the technique studied in lectures, including basics about ConvNet, standardization, dropout, and deep learning etc. The programming language used is python, which copes with TensorFlow and TensorPack for easier neuron network training and validation.

# Part 3: Training

## 3.1 TensorFlow Installation

## Installation

The environment is set up on my personal Windows 10 (64bits) laptop. The following commands are my installation procedures:

1. Download and install python 3.6.3, then add python and pip to %PATH
2. Install TensorFlow and TensorPack
   1. PS > pip install --upgrade pip
   2. PS > pip install tensorflow
   3. PS > pip install --upgrade tensorflow==1.15.0
   4. pip install --pre "tensorflow==1.15.0" (1)
   5. PS > pip install --upgrade tensorpack==0.8.9
3. Install required packages
   1. PS > pip install opencv-python

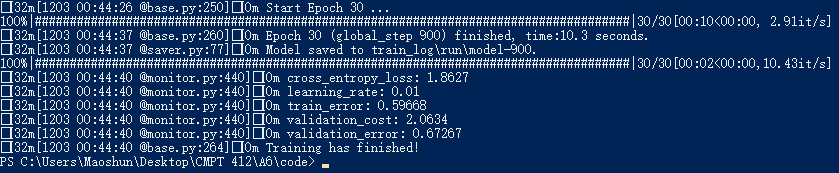
(1) According to the TensorFlow document, for 1.15 release, CPU and GPU support are included in a single package

### Run

The following command is for running the code

PS > python run.py --task 1 --gpu 1

The followings are the output (partial) for the command above:



It achieves an accuracy at around 33.8%.

## 3.2 Training from Scratch

### 3.2.1 Add Improvements

The following features are added into the code

* run.py
  + Standardization
    - Subtracting the mean of images intensities then divide by standard deviation
  + Augmentation (imgaug package added)
    - Resize
    - Random crop
    - Flip (horizontal)
    - Transpose
* your\_model.py
  + Dropout regularization
    - Added at the end of the original network architecture
  + More layers
    - Added three more layers, including dropout and dense layers, between the initial max pooling layer and the initial fully connected layer. (see your\_model.py for detail)

The followings are the best test error:

### 

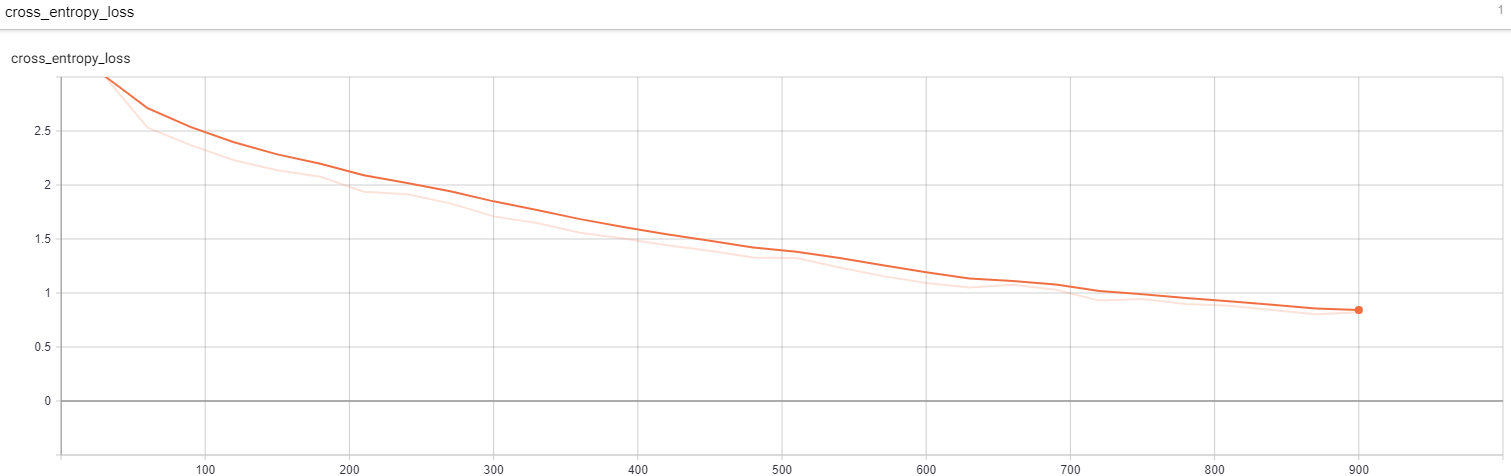
It achieved test accuracy at around 54.6%

### 3.2.2 Visualize by TensorBoard

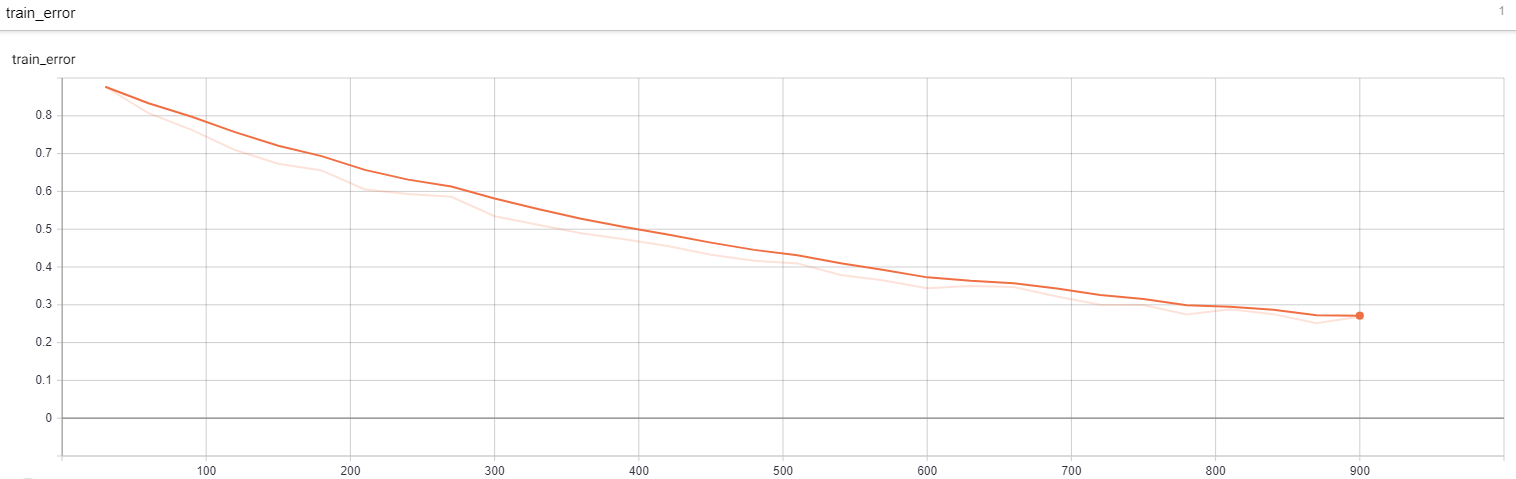
Running the following command

PS > tensorboard --logdir=train\_log/run

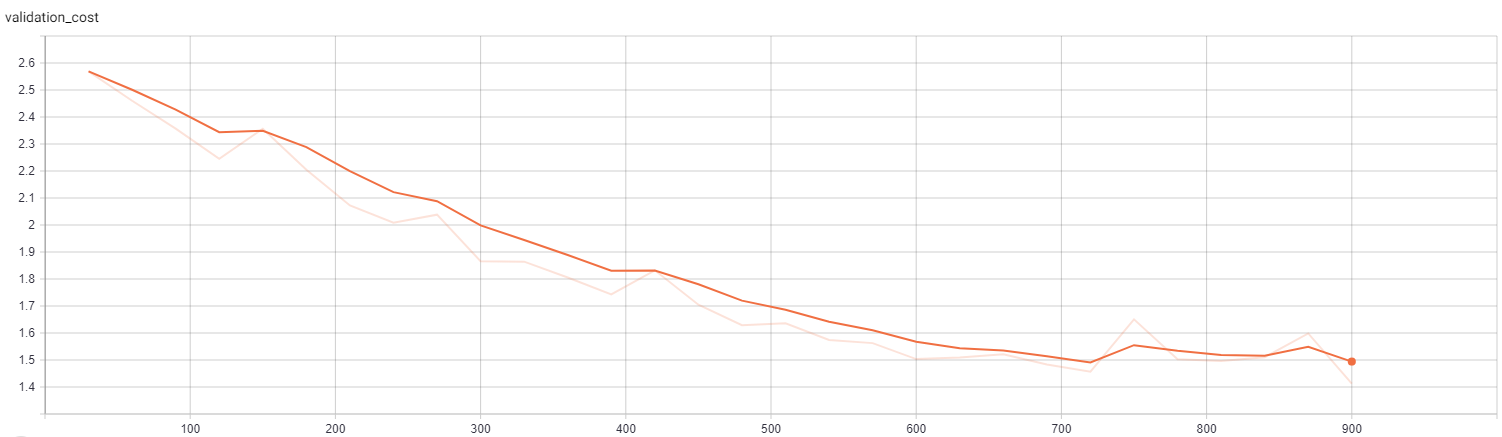
By navigating to <http://localhost:6006/>, we see the following graphs



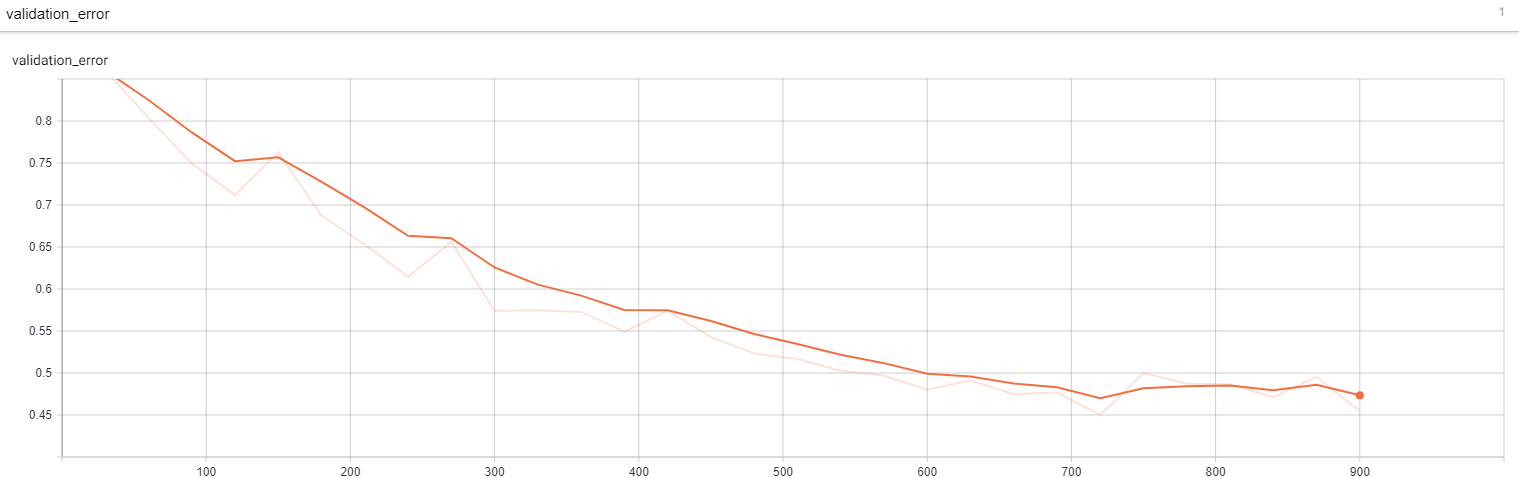
Cross Entropy Loss



Train Error



Validation Cost



Validation Error

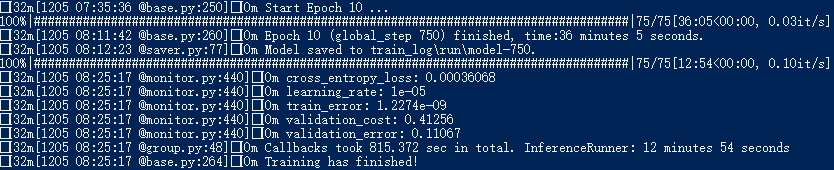
The training error is monotonically decreasing while the validation error is not as low as training error. In addition, it shows a few fluctuations. All the phenomenon above might indicate an overfitting.

## 3.3 Fine-tuning VGG

Adjustments:

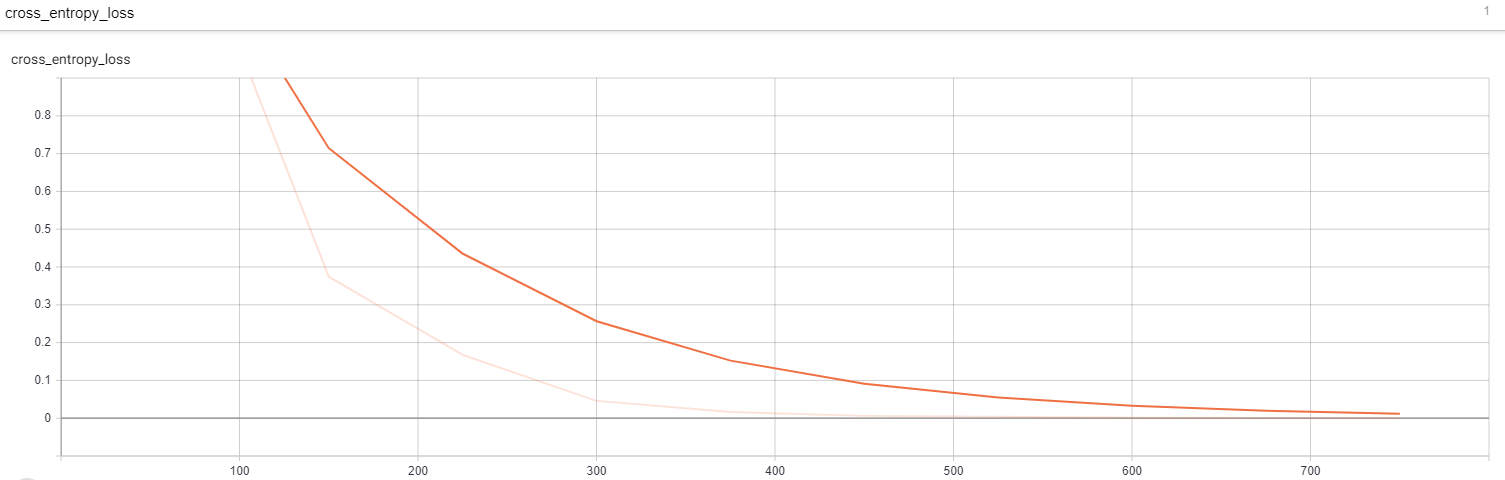
|  |  |  |
| --- | --- | --- |
|  | Original | Adjusted |
| num\_epochs | 30 | 10 |
| batch\_size | 50 | 20 |
| learning\_rate | 0.01 | 0.00001 |

The result of hyperparameter turning is shown below:

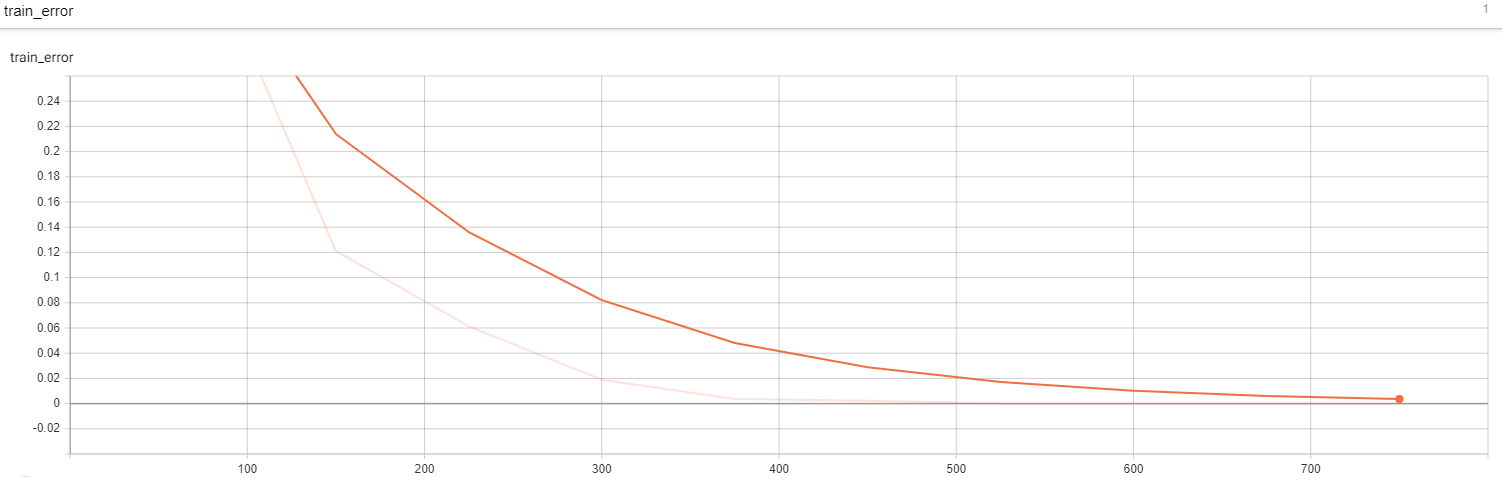


The test accuracy is at around 89%.

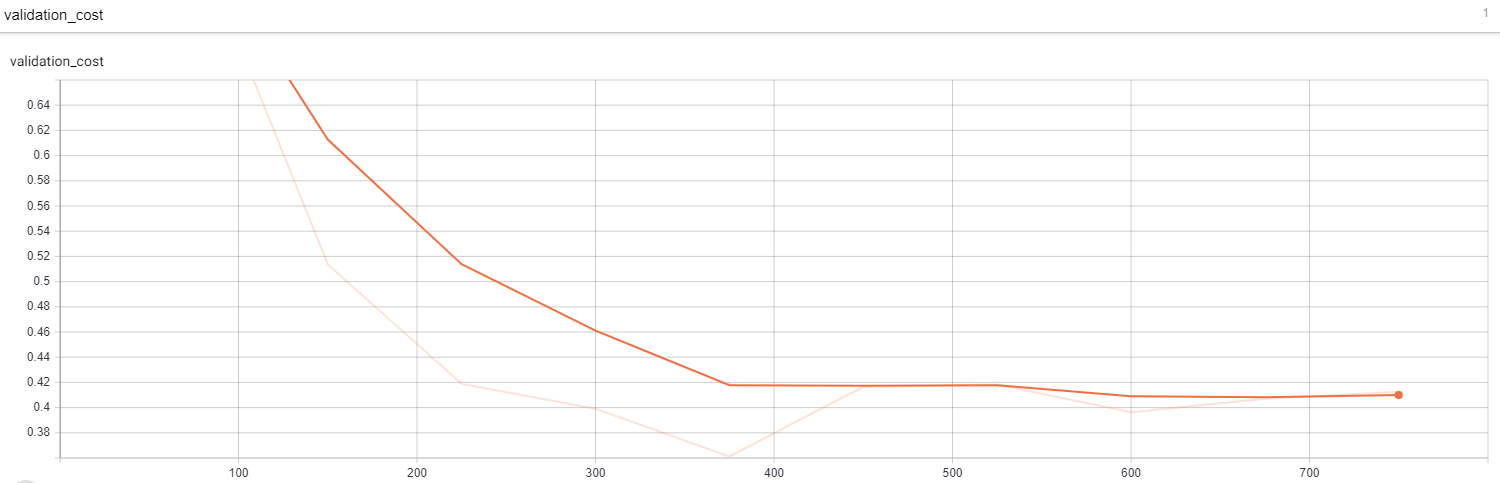
The below are the screenshots from TensorBoard:



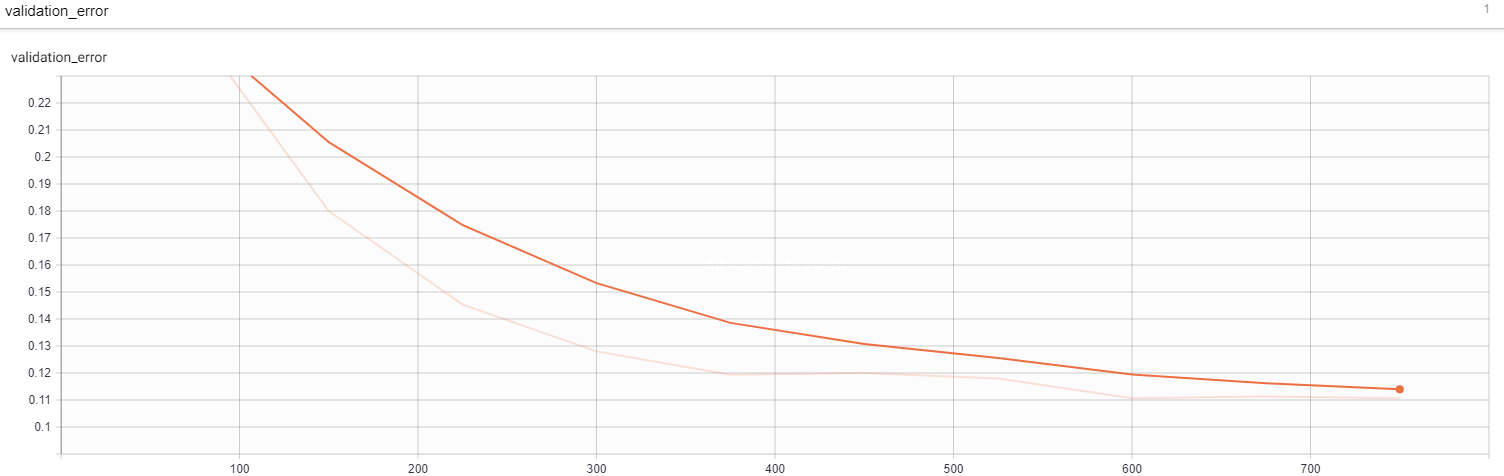
Cross Entropy Loss



Train Error



Validation Cost



Validation Error

VGG16 model is significantly different from the previous model. Both training and validation error shows a satisfying trend, which decreases until a very low level.

Note:

run “pip install numpy==1.16.2” if the system prompts exception of ValueError.

Reduce the batch\_size if the system prompts “TensorFlow allocation memory” exception.