

# Project 3 - Convolutional Neural Network

CPSC 585 Neural Networks  
Professor Kenytt Avery

William Dalessi, [wdalessi@csu.fullerton.edu](mailto:wdalessi@csu.fullerton.edu)  
Ken Hsu, [hsu.kenchen@csu.fullerton.edu](mailto:hsu.kenchen@csu.fullerton.edu)  
Paul Do, [khoado@csu.fullerton.edu](mailto:khoado@csu.fullerton.edu)

Due: 2020-05-06

## Overview of Submission

For this project, we wanted to have the following features:

- the ability to run more than one models by avoiding hard coding the actual network
- the ability to declaratively specify a network to be built for our classification task
- the ability to run these models without having to go back into the python code
- the ability to generate different varieties of network with a program
- the ability to manually specify another network without having to modify the code
- the ability to test different configuration with varying hyperparameters
- the ability to run the same network multiple times
- the ability to store the result/accuracy of the classification in a file for further analysis

## Model.csv

An independent model can be created using a text editor or generated by a program. This model file describes how the main notebook can create a model. Each model can be specified in a file (example: model\_ref.csv):

The top line (excluding the comments) are the hyperparameters that can specify the different parameters that can be specified. The comments describe the values and are self explanatory. One value, the processCount specifies how many times the model is run **only if** the computed accuracy exceeds a threshold of 90%.

**code:**

```
# Hyperparameters  
  
# processCount, epochCount, batchSize, loss, optimizer  
  
3,1000,2000,categorical_crossentropy,RMSprop
```

The layer description will describe the different layers. Allowable values (not complete) are conv2d, dense, dropout, flatten. In the first layer is conv2d,relu,32,3,3,l1,0.0002 which means that the **conv2d** layer is used with a **relu** activation function with a filter count of **32** and a kernel size of **(3, 3)**. The kernel regularizer is **l1** with a value of **0.0002**.

**code:**

```
# Layer Description  
  
conv2d,relu,32,3,3,l1,0.0002  
  
conv2d,relu,64,3,3,,  
  
maxpooling2d,2,2  
  
dropout,0.25
```

## Execution

During execution of the main notebook, the program will load all model files in **./models** directory and execute each one. Once the model is executed and the score is computed, the score is saved in a file **./models/results/scoreboard.csv**. The accuracy scores are sorted with the best at the top. The model may be run multiple times but once they are done, they are

stored at **./models/archive**. The model might encounter an exception. Those are stored at **./models/archive/bad**.

## Notebook

The submission includes two notebooks:

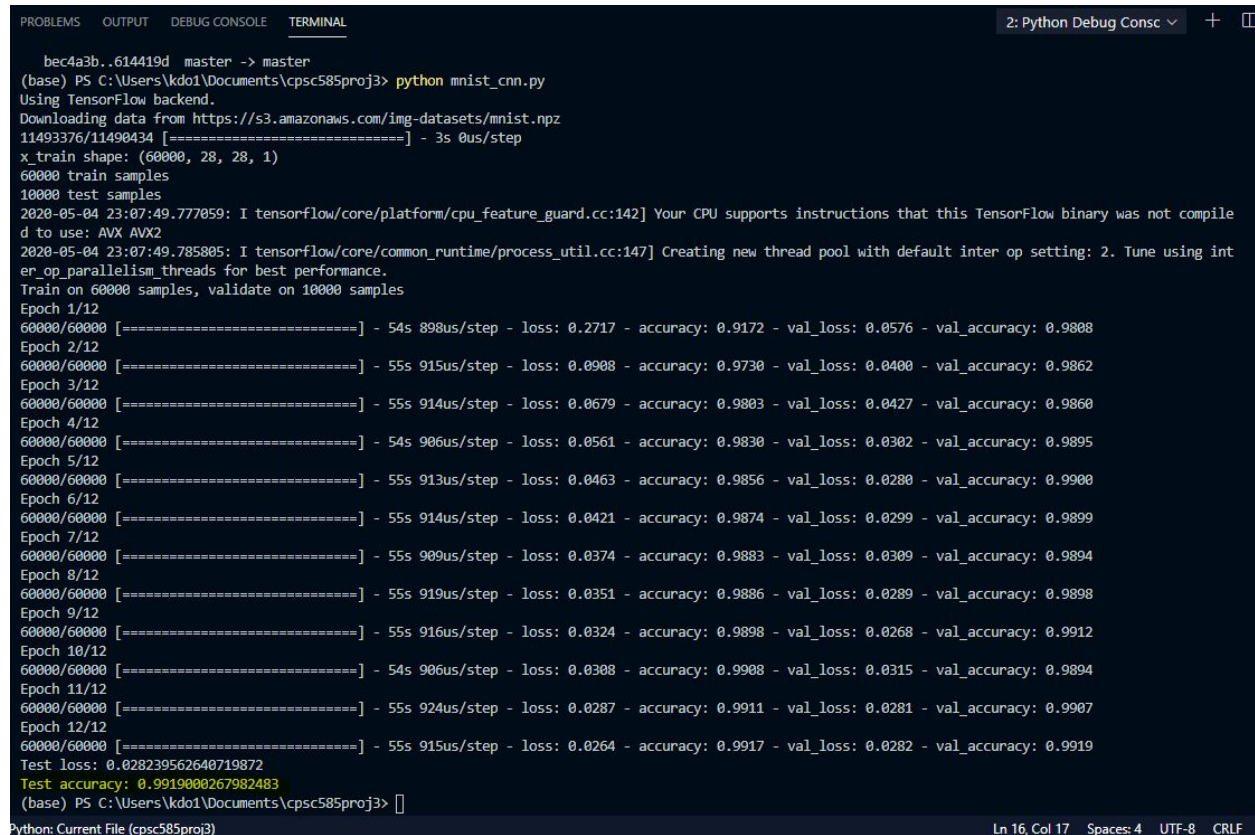
**project3-2.ipynb** - this notebook will run **all** models and is the main program.

**generate\_model.ipynb** - this notebook can be used to generate additional models for analysis.

# Questions

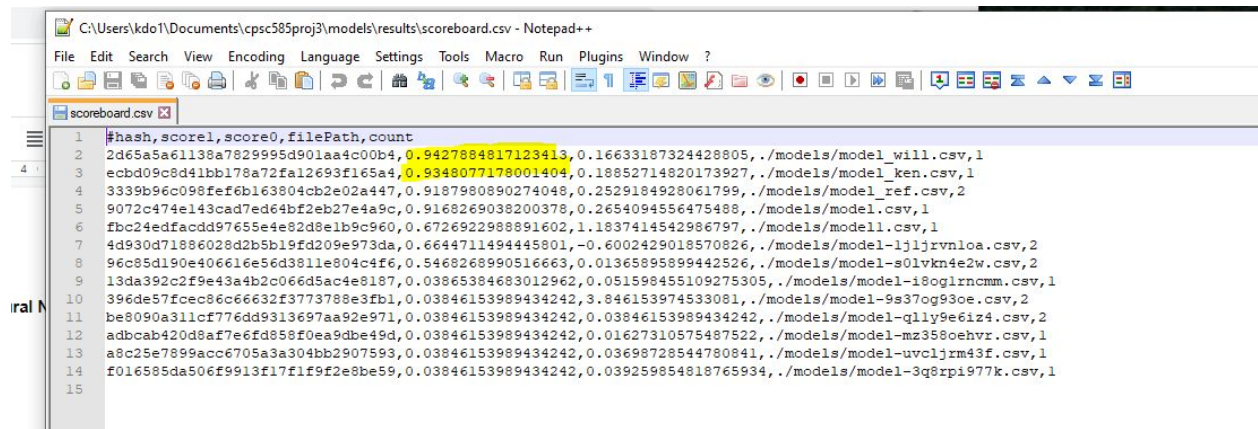
How does the accuracy compare to its performance on MNIST?

We ran the script `mnist_cnn.py`. We were able to get **0.99190** which is extremely good.



```
bec4a3b..614419d master -> master
(base) PS C:\Users\kdo1\Documents\cpsc585proj3> python mnist_cnn.py
Using TensorFlow backend.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [=====] - 3s 0us/step
x train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
2020-05-04 23:07:49.777059: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2
2020-05-04 23:07:49.785805: I tensorflow/core/common_runtime/process_util.cc:147] Creating new thread pool with default inter op setting: 2. Tune using inter_op_parallelism_threads for best performance.
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [=====] - 54s 898us/step - loss: 0.2717 - accuracy: 0.9172 - val_loss: 0.0576 - val_accuracy: 0.9888
Epoch 2/12
60000/60000 [=====] - 55s 915us/step - loss: 0.0908 - accuracy: 0.9730 - val_loss: 0.0400 - val_accuracy: 0.9862
Epoch 3/12
60000/60000 [=====] - 55s 914us/step - loss: 0.0679 - accuracy: 0.9803 - val_loss: 0.0427 - val_accuracy: 0.9860
Epoch 4/12
60000/60000 [=====] - 54s 906us/step - loss: 0.0561 - accuracy: 0.9830 - val_loss: 0.0302 - val_accuracy: 0.9895
Epoch 5/12
60000/60000 [=====] - 55s 913us/step - loss: 0.0463 - accuracy: 0.9856 - val_loss: 0.0280 - val_accuracy: 0.9900
Epoch 6/12
60000/60000 [=====] - 55s 914us/step - loss: 0.0421 - accuracy: 0.9874 - val_loss: 0.0299 - val_accuracy: 0.9899
Epoch 7/12
60000/60000 [=====] - 55s 909us/step - loss: 0.0374 - accuracy: 0.9883 - val_loss: 0.0309 - val_accuracy: 0.9894
Epoch 8/12
60000/60000 [=====] - 55s 919us/step - loss: 0.0351 - accuracy: 0.9886 - val_loss: 0.0289 - val_accuracy: 0.9898
Epoch 9/12
60000/60000 [=====] - 55s 916us/step - loss: 0.0324 - accuracy: 0.9898 - val_loss: 0.0268 - val_accuracy: 0.9912
Epoch 10/12
60000/60000 [=====] - 54s 906us/step - loss: 0.0308 - accuracy: 0.9908 - val_loss: 0.0315 - val_accuracy: 0.9894
Epoch 11/12
60000/60000 [=====] - 55s 924us/step - loss: 0.0287 - accuracy: 0.9911 - val_loss: 0.0281 - val_accuracy: 0.9907
Epoch 12/12
60000/60000 [=====] - 55s 915us/step - loss: 0.0264 - accuracy: 0.9917 - val_loss: 0.0282 - val_accuracy: 0.9919
Test loss: 0.028239562640719872
Test accuracy: 0.9919000267982483
(base) PS C:\Users\kdo1\Documents\cpsc585proj3>
```

We were able to train several models with very good results. Using the application developed by the team, we compiled the top results and stored in the file at `models\results\scoreboard.csv`. Looking at the current result, the top two results were: **.94279** and **.93480**. However, they don't compare to `mnist_cnn` result which was fantastic.



```
C:\Users\kdo1\Documents\cpsc585proj3\models\results\scoreboard.csv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
scoreboard.csv
1 #hash,score1,score0,filePath,count
2 2d65a5a61138a7829995d901aa4c00b4,0.9427864817123413,0.16633187324428805,./models/model_will.csv,1
3 ecbbd09c8d41bb178a72fa12693f165a4,0.9348077178001404,0.18852714820173927,./models/model_ken.csv,1
4 3339b96c098fef6b163804cb2e02a447,0.9187980890274048,0.2529184928061799,./models/model_ref.csv,2
5 9072c474e143cad7ed64bf2eb27e4a9c,0.9168269038200378,0.2654094556475488,./models/model.csv,1
6 fbc24edfacdd97655e4e82d8eb9c960,0.6726922988891602,1.1837414542986797,./models/model1.csv,1
7 4d930d71886028d2b5b19fd209e973da,0.6644711494445801,-0.6002429018570826,./models/model-1jljrvnloa.csv,2
8 96c85d190e406616e56d3811e804c4f6,0.5468268990516663,0.01365895899442526,./models/model-s0lvkn4e2w.csv,2
9 13da392c2f9e43a4b2c066d5ac4e8187,0.03865384683012962,0.051598455109275305,./models/model-i8oglrncmm.csv,1
10 396de57fcec86c66632f3773788e3fb1,0.03846153989434242,3.846153974533081,./models/model-9s37og93oe.csv,2
11 be8090a311cf776dd9313697aa92e971,0.03846153989434242,0.03846153989434242,./models/model-q1ly9e6iz4.csv,2
12 adbcab420d8af7e6fd858f0ea9dbe49d,0.03846153989434242,0.01627310575487522,./models/model-mz358oehvr.csv,1
13 a8c25e7899acc6705a3a304bb2907593,0.03846153989434242,0.03698728544780841,./models/model-uvcljrm43f.csv,1
14 f016585da506f9913f17f1f9f2e8be59,0.03846153989434242,0.039259854818765934,./models/model-3q8rp1977k.csv,1
15
```

## How does the accuracy compare to the MLP you trained in Project 2?

In our project 2, we were able to get our accuracy to around **91.4%**. This was not as good as our CNN results.

```
Epoch 48/1000
104000/104000 [=====] - 2s 15us/sample - loss: 0.2187 - accuracy: 0.9230 - val_loss: 0.2630 - val_accuracy: 0.9161
Epoch 49/1000
104000/104000 [=====] - 1s 14us/sample - loss: 0.2162 - accuracy: 0.9239 - val_loss: 0.2632 - val_accuracy: 0.9165
Epoch 50/1000
104000/104000 [=====] - 2s 15us/sample - loss: 0.2140 - accuracy: 0.9240 - val_loss: 0.2672 - val_accuracy: 0.9151
Epoch 51/1000
104000/104000 [=====] - 1s 13us/sample - loss: 0.2118 - accuracy: 0.9245 - val_loss: 0.2690 - val_accuracy: 0.9146
Epoch 52/1000
104000/104000 [=====] - 1s 13us/sample - loss: 0.2093 - accuracy: 0.9261 - val_loss: 0.2674 - val_accuracy: 0.9149
Epoch 53/1000
104000/104000 [=====] - 1s 14us/sample - loss: 0.2071 - accuracy: 0.9265 - val_loss: 0.2656 - val_accuracy: 0.9159
Test loss: 0.2708200138642524
Test accuracy: 0.9149038
```