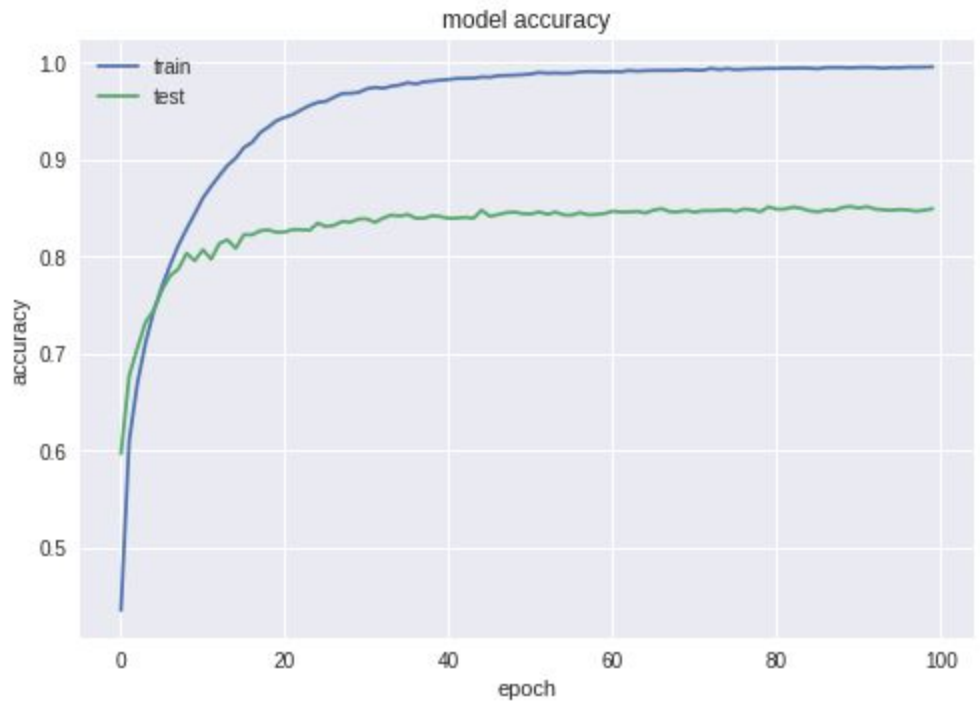
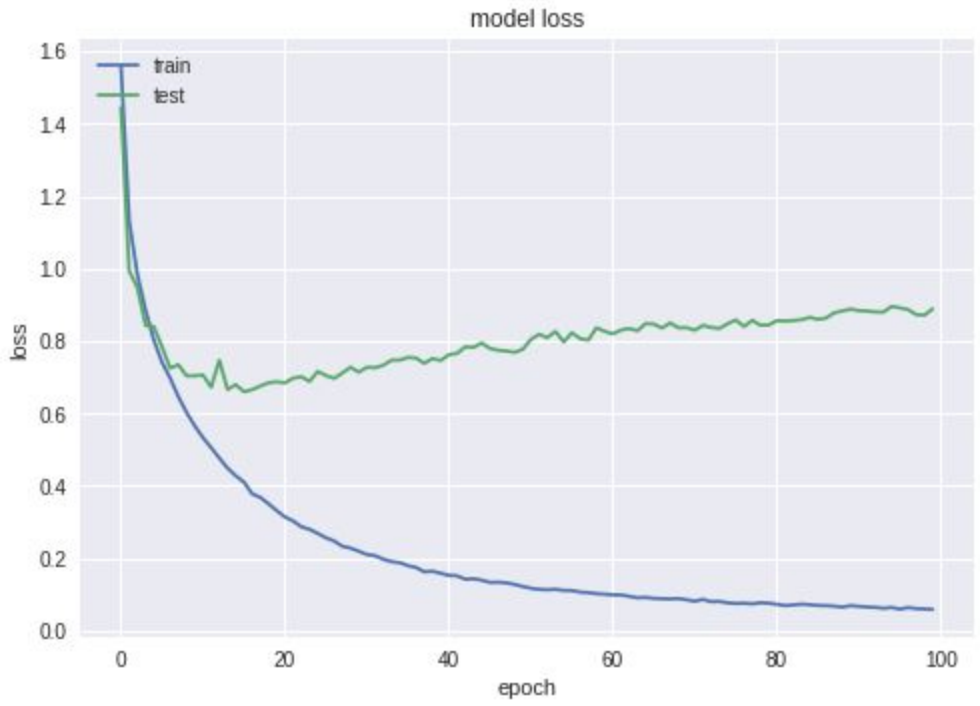
INFO 7374 Spring 2019
Assignment 1
Group 6

Part 2

Activation uses relu:

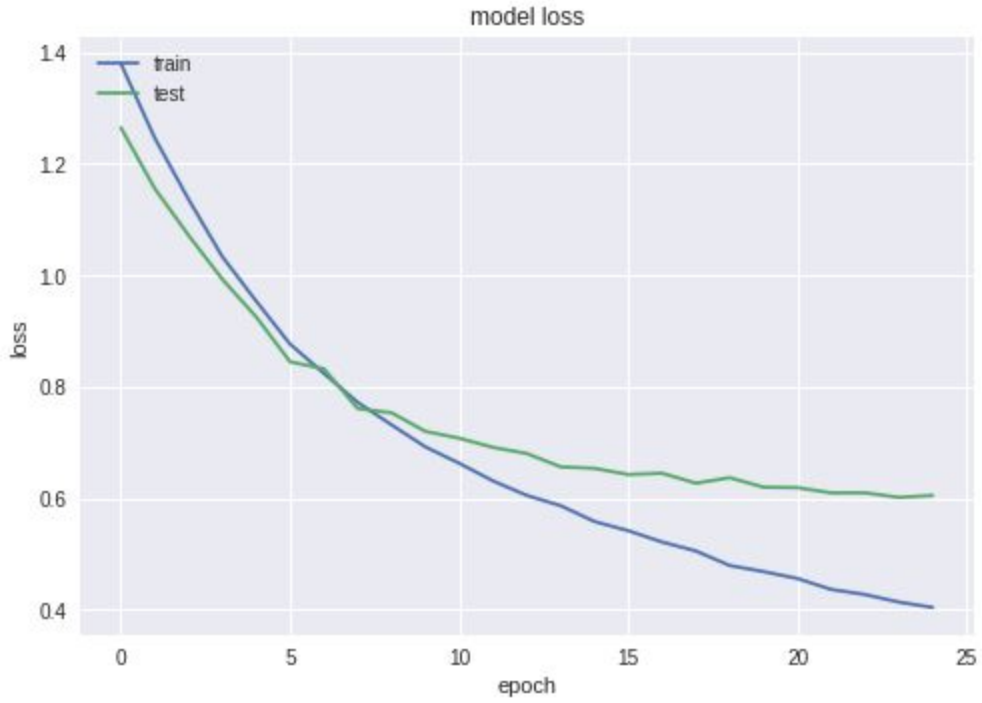
```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
```

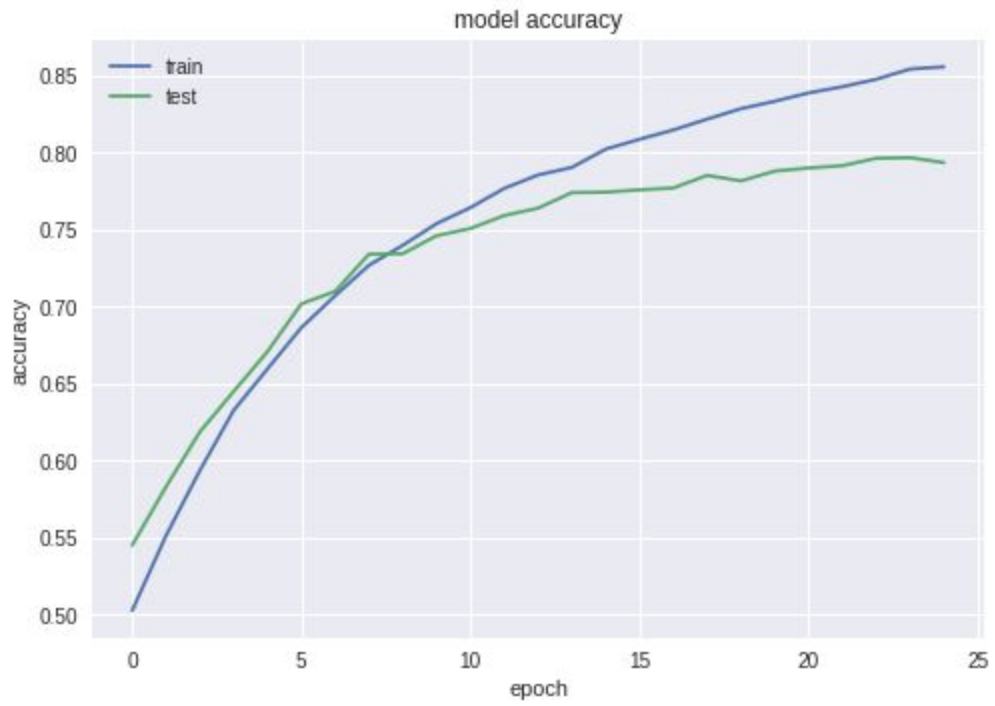

model accuracy
train
test
accuracy
epoch
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100


model loss
train
test
loss
epoch
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

```
50000/50000 [==============================] - 19s 379us/step - loss: 0.0373 - acc: 0.9869 - val_loss: 0.8626 - val_acc: 0.8243
Epoch 88/100
50000/50000 [==============================] - 19s 379us/step - loss: 0.0353 - acc: 0.9876 - val_loss: 0.8773 - val_acc: 0.8248
Epoch 89/100
50000/50000 [==============================] - 19s 380us/step - loss: 0.0354 - acc: 0.9885 - val_loss: 0.8697 - val_acc: 0.8214
Epoch 90/100
50000/50000 [==============================] - 19s 379us/step - loss: 0.0346 - acc: 0.9883 - val_loss: 0.8731 - val_acc: 0.8191
Epoch 91/100
50000/50000 [==============================] - 19s 385us/step - loss: 0.0345 - acc: 0.9885 - val_loss: 0.8753 - val_acc: 0.8201
Epoch 92/100
50000/50000 [==============================] - 19s 387us/step - loss: 0.0324 - acc: 0.9889 - val_loss: 0.9069 - val_acc: 0.8206
Epoch 93/100
50000/50000 [==============================] - 19s 382us/step - loss: 0.0322 - acc: 0.9885 - val_loss: 0.9043 - val_acc: 0.8244
Epoch 94/100
50000/50000 [==============================] - 19s 379us/step - loss: 0.0320 - acc: 0.9891 - val_loss: 0.8787 - val_acc: 0.8236
Epoch 95/100
50000/50000 [==============================] - 19s 379us/step - loss: 0.0306 - acc: 0.9892 - val_loss: 0.8916 - val_acc: 0.8227
Epoch 96/100
50000/50000 [==============================] - 19s 379us/step - loss: 0.0325 - acc: 0.9890 - val_loss: 0.8969 - val_acc: 0.8219
Epoch 97/100
50000/50000 [==============================] - 19s 380us/step - loss: 0.0313 - acc: 0.9892 - val_loss: 0.8905 - val_acc: 0.8220
Epoch 98/100
50000/50000 [==============================] - 19s 382us/step - loss: 0.0311 - acc: 0.9892 - val_loss: 0.8990 - val_acc: 0.8217
Epoch 99/100
50000/50000 [==============================] - 19s 380us/step - loss: 0.0298 - acc: 0.9898 - val_loss: 0.9184 - val_acc: 0.8212
Epoch 100/100
50000/50000 [==============================] - 19s 380us/step - loss: 0.0311 - acc: 0.9894 - val_loss: 0.8895 - val_acc: 0.8216
Accuracy: 82.16%
```

Accuracy is about 82.16% But it’s get overfitting quickly.





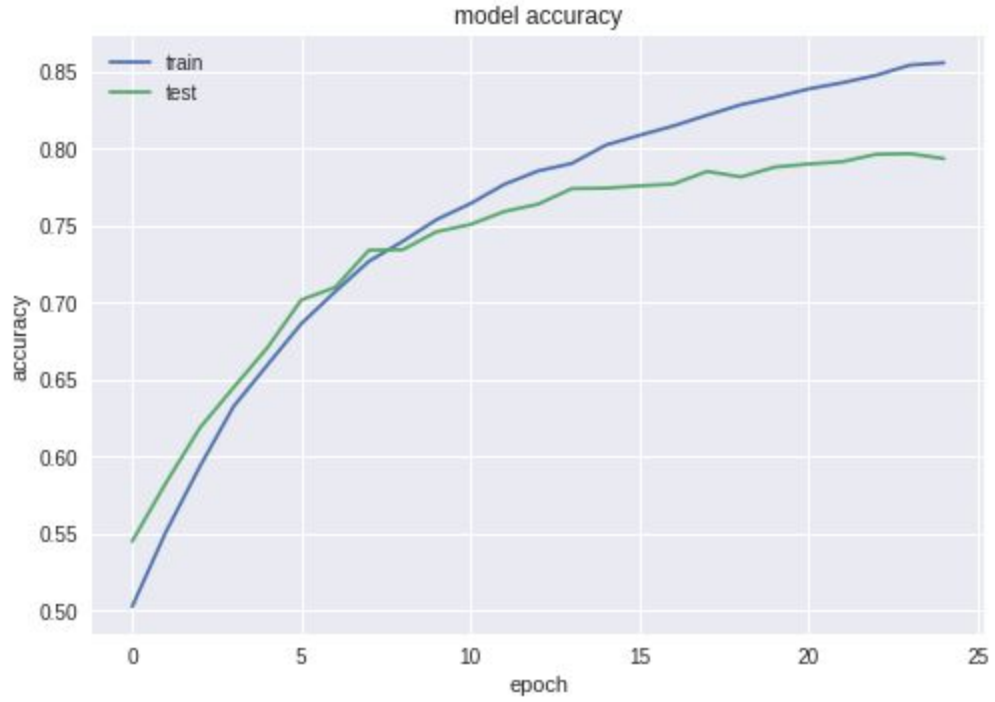

```
[10] 50000/50000 [==============================] - 19s 386us/step - loss: 0.6310 - acc: 0.7769 - val_loss: 0.6912 - val_acc: 0.7592
     Epoch 13/25
     50000/50000 [==============================] - 19s 384us/step - loss: 0.6049 - acc: 0.7856 - val_loss: 0.6802 - val_acc: 0.7640
     Epoch 14/25
     50000/50000 [==============================] - 19s 387us/step - loss: 0.5866 - acc: 0.7905 - val_loss: 0.6563 - val_acc: 0.7741
     Epoch 15/25
     50000/50000 [==============================] - 19s 384us/step - loss: 0.5581 - acc: 0.8023 - val_loss: 0.6535 - val_acc: 0.7744
     Epoch 16/25
     50000/50000 [==============================] - 19s 384us/step - loss: 0.5416 - acc: 0.8087 - val_loss: 0.6425 - val_acc: 0.7759
     Epoch 17/25
     50000/50000 [==============================] - 19s 384us/step - loss: 0.5213 - acc: 0.8148 - val_loss: 0.6448 - val_acc: 0.7771
     Epoch 18/25
     50000/50000 [==============================] - 19s 386us/step - loss: 0.5052 - acc: 0.8219 - val_loss: 0.6269 - val_acc: 0.7853
     Epoch 19/25
     50000/50000 [==============================] - 19s 386us/step - loss: 0.4791 - acc: 0.8286 - val_loss: 0.6368 - val_acc: 0.7817
     Epoch 20/25
     50000/50000 [==============================] - 19s 388us/step - loss: 0.4682 - acc: 0.8335 - val_loss: 0.6198 - val_acc: 0.7881
     Epoch 21/25
     50000/50000 [==============================] - 19s 388us/step - loss: 0.4558 - acc: 0.8389 - val_loss: 0.6188 - val_acc: 0.7901
     Epoch 22/25
     50000/50000 [==============================] - 19s 385us/step - loss: 0.4362 - acc: 0.8429 - val_loss: 0.6096 - val_acc: 0.7916
     Epoch 23/25
     50000/50000 [==============================] - 19s 387us/step - loss: 0.4270 - acc: 0.8478 - val_loss: 0.6096 - val_acc: 0.7964
     Epoch 24/25
     50000/50000 [==============================] - 19s 387us/step - loss: 0.4135 - acc: 0.8544 - val_loss: 0.6014 - val_acc: 0.7968
     Epoch 25/25
     50000/50000 [==============================] - 19s 385us/step - loss: 0.4039 - acc: 0.8559 - val_loss: 0.6049 - val_acc: 0.7936
     Accuracy: 79.36%
```

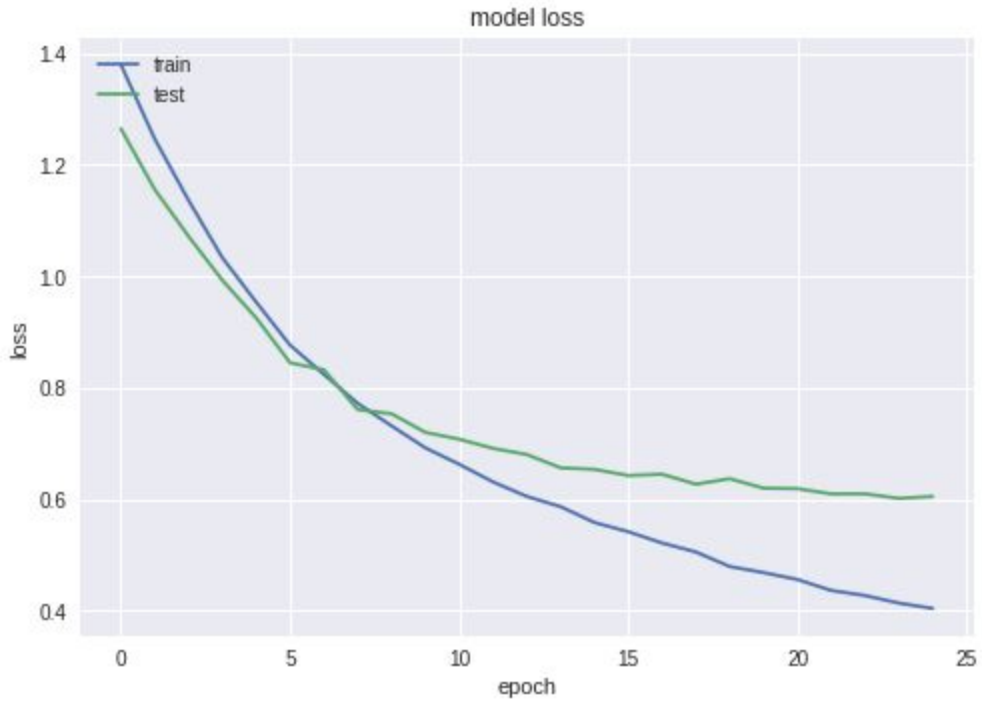
Accuracy is about 79.36%.

Activation uses tanh:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='tanh', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation=' tanh, padding='same'))
```

```
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation=' tanh, padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation=' tanh, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation=' tanh, padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation=' tanh, kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation=' tanh, kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
```





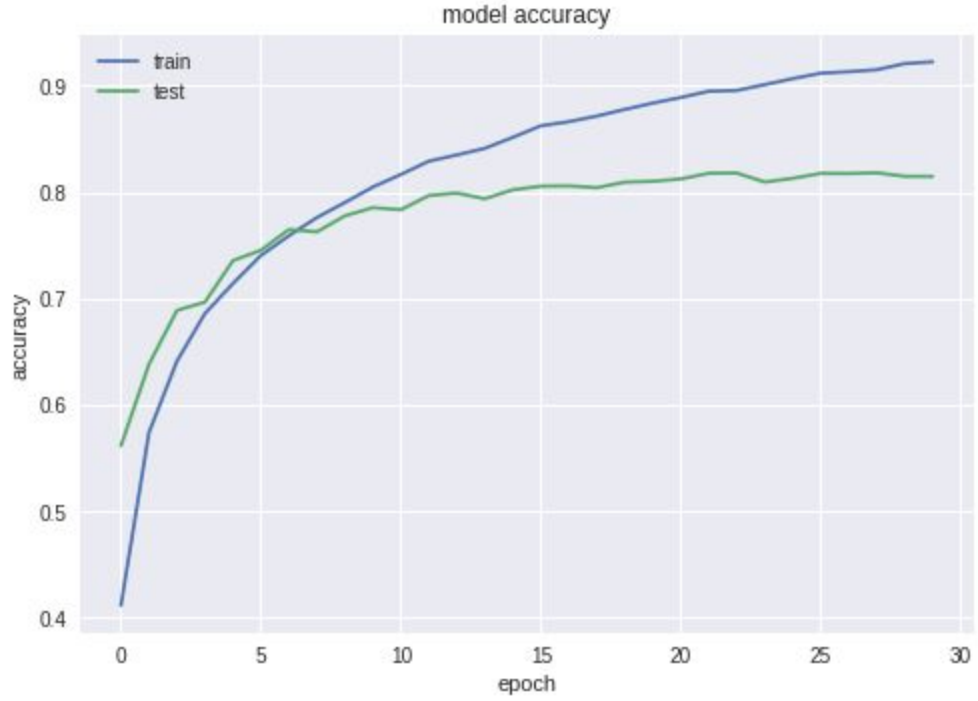

```
50000/50000 [==============================] - 19s 384us/step - loss: 0.5579 - acc: 0.8027 - val_loss: 0.6822 - val_acc: 0.7754
Epoch 13/25
50000/50000 [==============================] - 19s 385us/step - loss: 0.5351 - acc: 0.8100 - val_loss: 0.6973 - val_acc: 0.7743
Epoch 14/25
50000/50000 [==============================] - 19s 386us/step - loss: 0.5111 - acc: 0.8193 - val_loss: 0.6661 - val_acc: 0.7805
Epoch 15/25
50000/50000 [==============================] - 19s 385us/step - loss: 0.4942 - acc: 0.8256 - val_loss: 0.6972 - val_acc: 0.7733
Epoch 16/25
50000/50000 [==============================] - 19s 385us/step - loss: 0.4815 - acc: 0.8310 - val_loss: 0.6703 - val_acc: 0.7833
Epoch 17/25
50000/50000 [==============================] - 19s 386us/step - loss: 0.4656 - acc: 0.8351 - val_loss: 0.6680 - val_acc: 0.7858
Epoch 18/25
50000/50000 [==============================] - 19s 386us/step - loss: 0.4507 - acc: 0.8401 - val_loss: 0.6503 - val_acc: 0.7918
Epoch 19/25
50000/50000 [==============================] - 19s 385us/step - loss: 0.4420 - acc: 0.8424 - val_loss: 0.6750 - val_acc: 0.7860
Epoch 20/25
50000/50000 [==============================] - 19s 386us/step - loss: 0.4269 - acc: 0.8489 - val_loss: 0.6521 - val_acc: 0.7919
Epoch 21/25
50000/50000 [==============================] - 19s 388us/step - loss: 0.4164 - acc: 0.8517 - val_loss: 0.6650 - val_acc: 0.7907
Epoch 22/25
50000/50000 [==============================] - 19s 387us/step - loss: 0.4084 - acc: 0.8549 - val_loss: 0.6700 - val_acc: 0.7941
Epoch 23/25
50000/50000 [==============================] - 19s 386us/step - loss: 0.3943 - acc: 0.8599 - val_loss: 0.6584 - val_acc: 0.7963
Epoch 24/25
50000/50000 [==============================] - 19s 389us/step - loss: 0.3890 - acc: 0.8614 - val_loss: 0.6617 - val_acc: 0.7959
Epoch 25/25
50000/50000 [==============================] - 19s 387us/step - loss: 0.3781 - acc: 0.8660 - val_loss: 0.6638 - val_acc: 0.7962
Accuracy: 79.62%
```

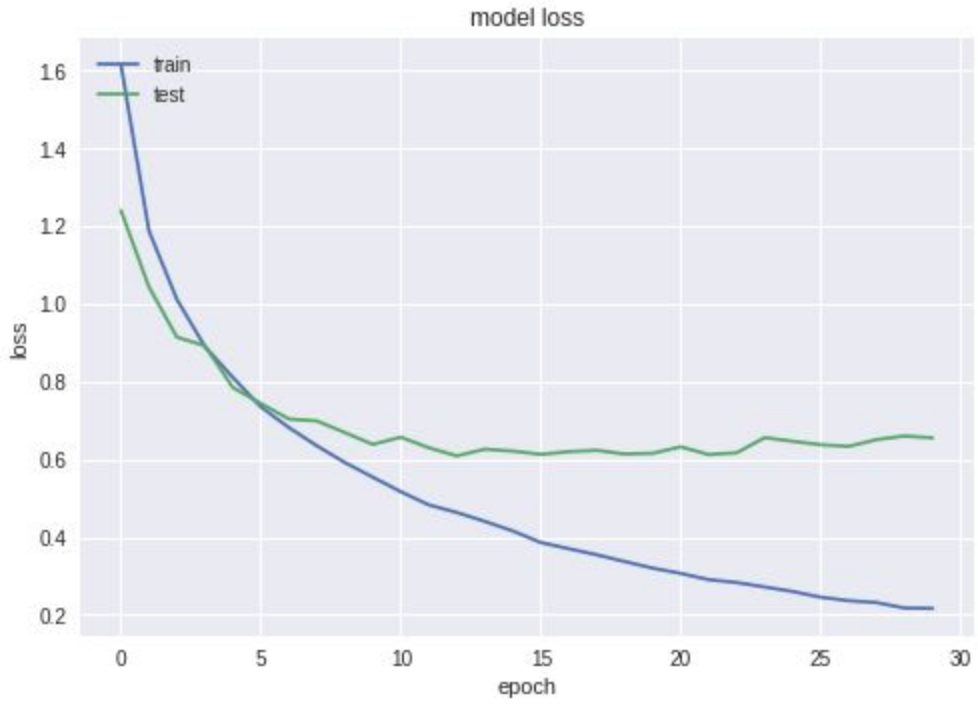
Accuracy is about 79.62%

Activation uses elu:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='elu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='elu', padding='same'))
```

```
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='elu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='elu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='elu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='elu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='elu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
```






```
Epoch 18/30
50000/50000 [==============================] - 20s 409us/step - loss: 0.3543 - acc: 0.8717 - val_loss: 0.6231 - val_acc: 0.8044
Epoch 19/30
50000/50000 [==============================] - 21s 416us/step - loss: 0.3368 - acc: 0.8779 - val_loss: 0.6135 - val_acc: 0.8096
Epoch 20/30
50000/50000 [==============================] - 22s 435us/step - loss: 0.3198 - acc: 0.8839 - val_loss: 0.6155 - val_acc: 0.8104
Epoch 21/30
50000/50000 [==============================] - 21s 425us/step - loss: 0.3064 - acc: 0.8892 - val_loss: 0.6321 - val_acc: 0.8126
Epoch 22/30
50000/50000 [==============================] - 21s 426us/step - loss: 0.2902 - acc: 0.8951 - val_loss: 0.6122 - val_acc: 0.8179
Epoch 23/30
50000/50000 [==============================] - 21s 429us/step - loss: 0.2829 - acc: 0.8957 - val_loss: 0.6167 - val_acc: 0.8182
Epoch 24/30
50000/50000 [==============================] - 21s 423us/step - loss: 0.2714 - acc: 0.9012 - val_loss: 0.6558 - val_acc: 0.8097
Epoch 25/30
50000/50000 [==============================] - 21s 426us/step - loss: 0.2598 - acc: 0.9069 - val_loss: 0.6463 - val_acc: 0.8132
Epoch 26/30
50000/50000 [==============================] - 21s 425us/step - loss: 0.2451 - acc: 0.9120 - val_loss: 0.6375 - val_acc: 0.8178
Epoch 27/30
50000/50000 [==============================] - 21s 421us/step - loss: 0.2361 - acc: 0.9134 - val_loss: 0.6331 - val_acc: 0.8177
Epoch 28/30
50000/50000 [==============================] - 21s 412us/step - loss: 0.2313 - acc: 0.9152 - val_loss: 0.6507 - val_acc: 0.8184
Epoch 29/30
50000/50000 [==============================] - 21s 413us/step - loss: 0.2173 - acc: 0.9210 - val_loss: 0.6599 - val_acc: 0.8151
Epoch 30/30
50000/50000 [==============================] - 21s 424us/step - loss: 0.2162 - acc: 0.9227 - val_loss: 0.6551 - val_acc: 0.8150
Accuracy: 81.50%
```

Accuracy is about 81.50%.

We tried several experiments to find the best model. The best model is activation with elu.

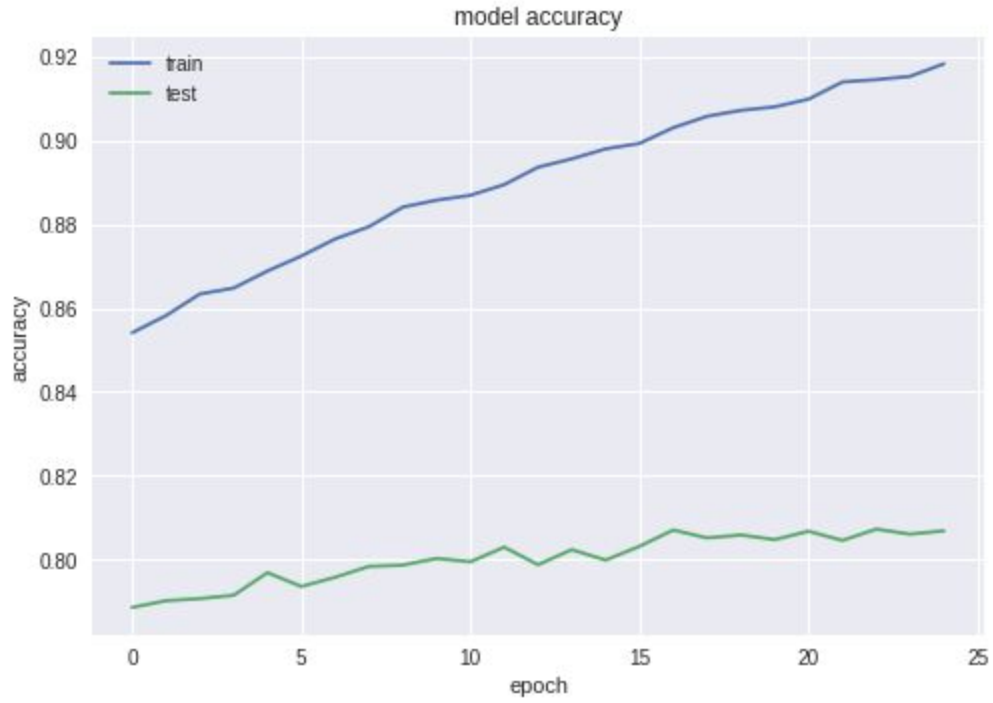
Sigmoid

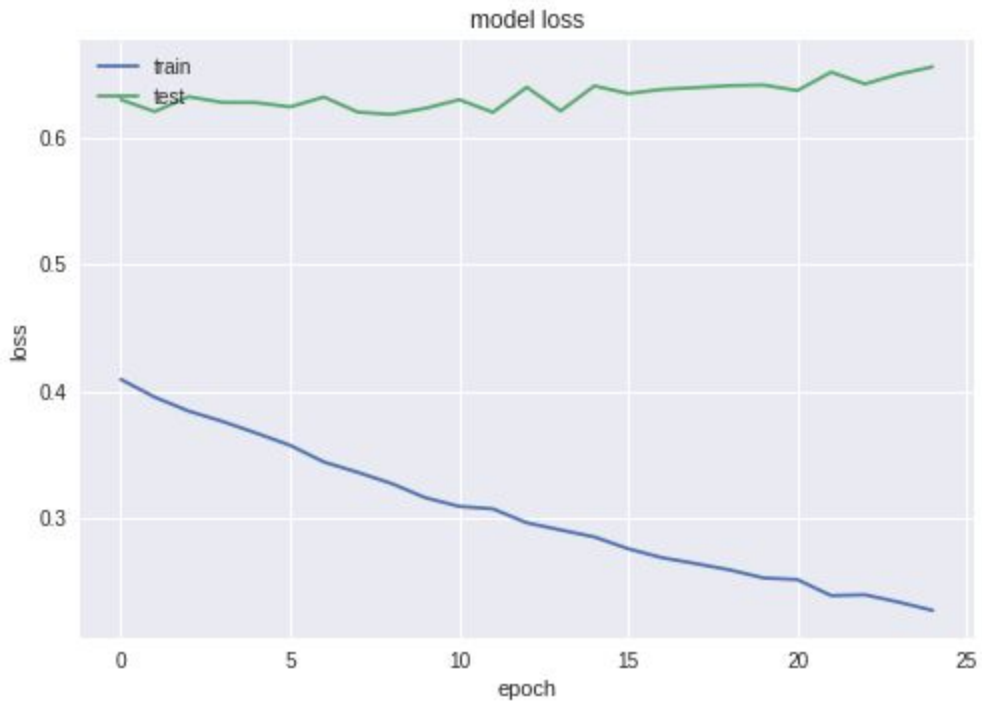
We tried training the model with sigmoid activation, and shown below is the set up of the model

```
# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), activation='sigmoid', padding='same'))
model.add(Dropout(0.2))
```

```
model.add(Conv2D(32, (3, 3), activation='sigmoid', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='sigmoid', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='sigmoid', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='sigmoid', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='sigmoid', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='sigmoid', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
```

Here is the preview of some of the results that we got





The results above simply indicate that the model did not learn anything, and that using sigmoid activation in the subsequent levels of the model is not sufficient for the model to learn this problem mainly because of the vanishing gradient problem

Findings

1. Training with CNN can be very time-consuming especial when we try each parameter other functions one by one.
2. By controlling the dropout rate, we can control the learning speed of our model, which can be helpful when the neural network model is a little under-fitting.
3. Reading related articles is really important for us. Deep learning is somehow based on experience on models. When we familiar with those model, we can save plenty of time to choose, build and train our model.