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))

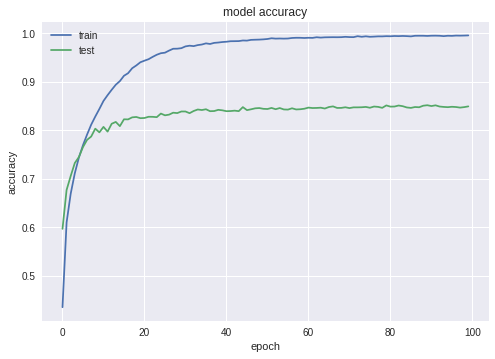
model.add(Dense(1024, activation='relu', kernel\_constraint=maxnorm(3)))

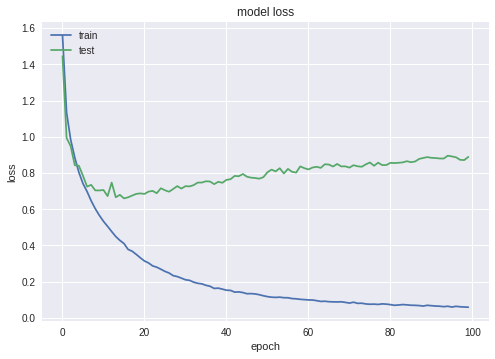
model.add(Dropout(0.2))

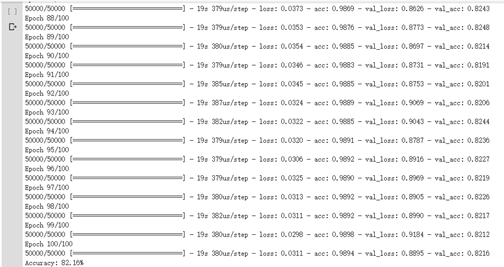
model.add(Dense(512, activation='relu', kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.2))

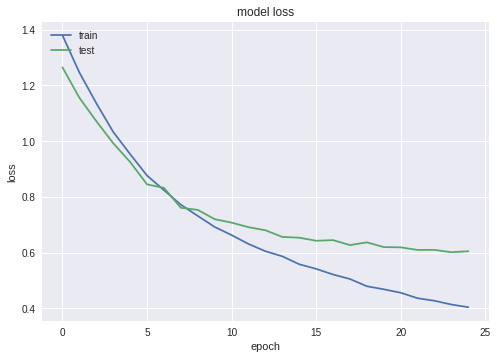
model.add(Dense(num\_classes, activation='softmax'))

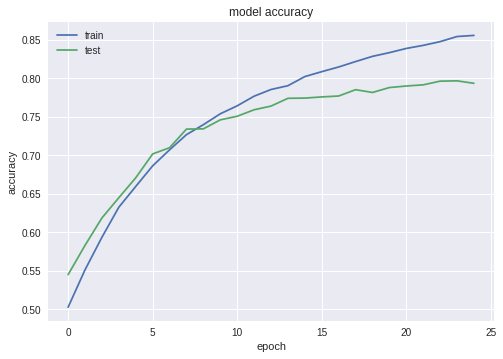


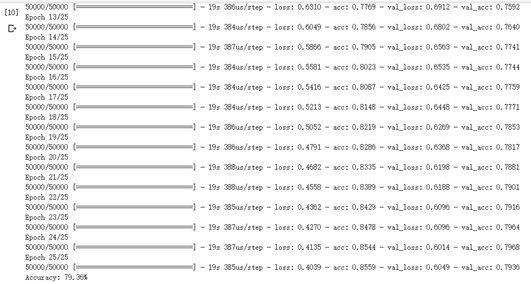




Accuracy is about 82.16% But it’s get overfitting quickly.







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))

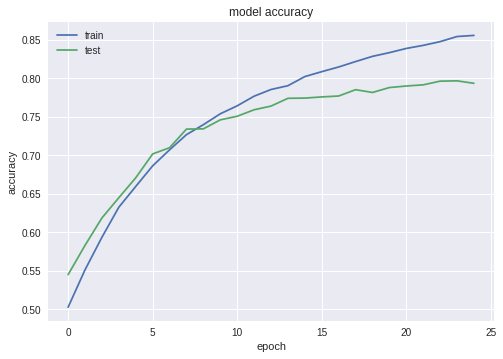
model.add(Dense(1024, activation=' tanh, kernel\_constraint=maxnorm(3)))

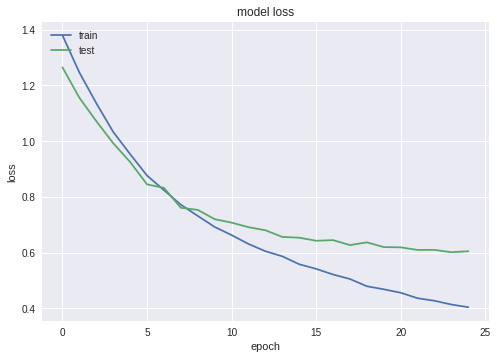
model.add(Dropout(0.2))

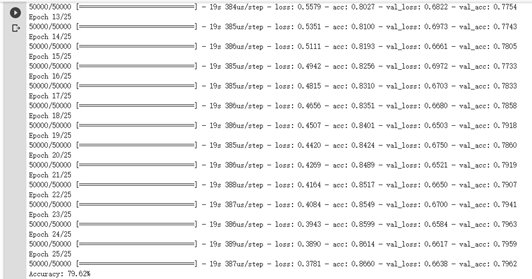
model.add(Dense(512, activation=' tanh, kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))







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))

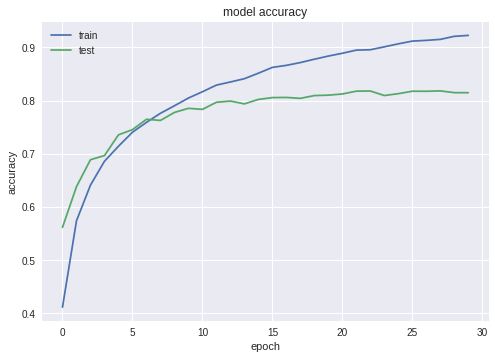
model.add(Dense(1024, activation='elu', kernel\_constraint=maxnorm(3)))

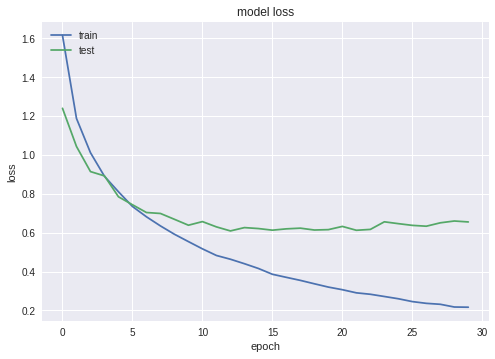
model.add(Dropout(0.2))

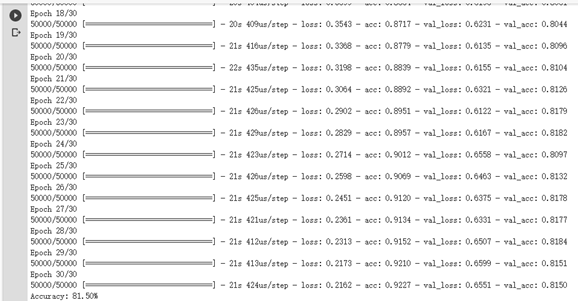
model.add(Dense(512, activation='elu', kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))







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)))

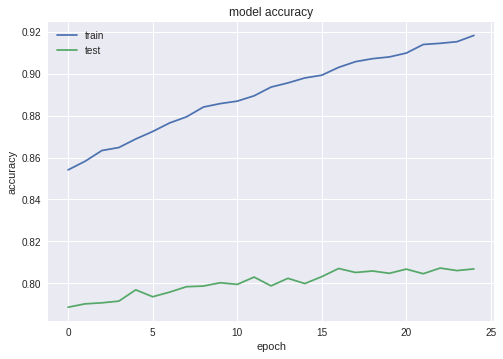
model.add(Dropout(0.2))

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.