# Exercises5

November 27, 2020

# 1 Exercise set 5

## 1.1 Answered to all questions (1-5).

# 2 Question 1

Number of parameters in:

**First layer:**

49152 inputs connected to each neuron in first layer. Therefore 49152*100 = 4915200 learnable weights. Also each neuron in first layer has a bias value. Therefore total number of parameters in this layer is 4915200+100 = 4915300

**Second layer:**

100 inputs connected to each neuron in second layer. Therefore 100*100 = 10000 learnable weights. Also each neuron in second layer has a bias value. Therefore total number of parameters in this layer is 10000+100 = 10100

**Third layer:**

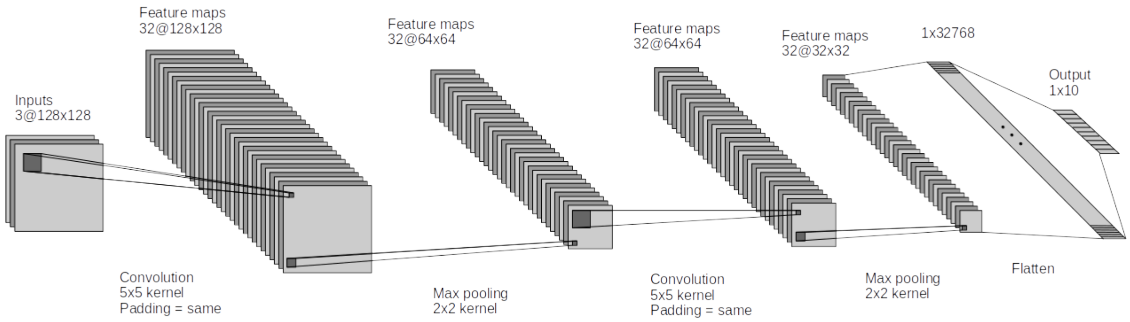
100 inputs connected to each neuron in third layer. Therefore 100*10 = 1000 learnable weights. Also each neuron in third layer has a bias value. Therefore total number of parameters in this layer is 1000+10 = 1010

**Total number of parameters in network**

4915300 + 10100 + 1010 = 4926410

# 3 Question 2

a)




b)

Number of parameters in:

**First convolutional layer:**

Each convolution kernel has $5 * 5$ parameters and we have 3 input layers. Therefore a single convolutional kernel "cube" has a $5 * 5 * 3* = 75$ learnable weight parameters. Also, you need to learn unique one for each output feature map. Therefore $75*32 = 2400$ learnable weight parameters. Last thing is to add bias values for each output feature map, which increases the total number of parameters to $2400 + 32 = 2432$

**Pooling layer:**

Pooling layer takes no parameters. It just shrinks the size of each feature map.

**Second convolutional layer:**

Each convolution kernel has $5 * 5$ parameters and we have 32 input layers. Therefore a single convolutional kernel "cube" has a $5 * 5 * 32* = 800$ learnable weight parameters. Also, you need to learn unique one for each output feature map. Therefore $800 * 32 = 25600$ learnable weight parameters. Last thing is to add bias values for each output feature map, which increases the total number of parameters to $25600 + 32 = 25632$

**Pooling layer:**

Takes no parameters.

**Flatten**

No learnable parameters. Just reduces the dimensionality of the input.

**Output layer**

Each neuron in the input is connected to each neuron in the output with learnable weight parameter. Therefore $32768*10 = 327680$ learnable weight parameters. Also each neuron in the output layer has a learnable bias value. Therefore learnable parameters in this layer adds up to $327680+10 = 327690$

**Total number of parameters in the network**

$2432 + 25632 + 327690 = 355754$

c)

On first layer, the convolution window slides over all possible locations in the 3 input channels and mutliplies the corresponding pixel values intersecting with the window. This procedure has to be done for each output feature map. In numbers, convolution window has a size of $5x5$ so 25 multiplications is done in single location. Also, convolution window of given size can move to 124 differernt positions horizontally, and 124 differernt positions vertically in the feature map of size of $128x128$. Therefore, $5x5$ window has $124 * 124 = 15376$ possible locations in the single input channel, so $15376 * 25 = 384400$ multiplications. We have 3 input channels, so multiplications needed to calculate values for a single output channel equals to $3 * 384400 = 1153200$. Finally, we have 32 output channels, so total number of scalar multiplications on the first convolutional layer equals to $32 * 1153200 = 36902400$.

# 4 Question 3

[1]:
```
import os
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# Load the data
os.chdir('/home/tuomas/Python/DATA.ML.200/Ex5')
trainX = np.load('X_train.npy')
trainY = np.load('y_train.npy')
testX = np.load('X_test.npy')
testY = np.load('y_test.npy')
# Vectorize
trainX_vec = trainX.reshape(-1, trainX.shape[1]*trainX.shape[2])
testX_vec = testX.reshape(-1, testX.shape[1]*testX.shape[2])
```

[2]:
```
models = [RandomForestClassifier(n_estimators=10,n_jobs=-1),
          RandomForestClassifier(n_estimators=50,n_jobs=-1),
          RandomForestClassifier(n_estimators=100,n_jobs=-1)]

names = ['RF10','RF50','RF100']
accuracy = []
for i in range(len(models)):
    print('Training {}...'.format(names[i]))
    models[i].fit(trainX_vec, trainY)
    print('Testing {}...'.format(names[i]))
    predY = models[i].predict(testX_vec)
    print('Evaluating {}...'.format(names[i]))
    accuracy.append(accuracy_score(testY, predY))
```

```
Training RF10…
Testing RF10…
Evaluating RF10…
Training RF50…
Testing RF50…
Evaluating RF50…
```

```
Training RF100…
Testing RF100…
Evaluating RF100…
```

[3]:
```python
# a, b & c
for i in range(len(accuracy)):
    print('{} accuracy = {}'.format(names[i], accuracy[i]))
```

```
RF10 accuracy = 0.5153333333333333
RF50 accuracy = 0.5833333333333334
RF100 accuracy = 0.5946666666666667
```

# 5 Question 4

[3]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import␣
↪Conv2D,BatchNormalization,Dropout,Flatten,Dense,MaxPool2D
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
# Add dummy dimensions & categorize labels
trainX = trainX[..., np.newaxis]
trainY_cat = to_categorical(trainY, 15)
testX = testX[..., np.newaxis]
testY_cat = to_categorical(testY, 15)
```

[3]:
```python
# Build the CNN
layers=[Conv2D(32, kernel_size=5, activation='relu', input_shape=(40,501,1)),
        MaxPool2D(pool_size=(2,2)),
        Conv2D(32, kernel_size=5, activation='relu', padding='same'),
        MaxPool2D(pool_size=(2,2)),
        Flatten(),
        Dense(15, activation='softmax')]

model = Sequential(layers)
model.
↪compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

[5]:
```python
# Train the CNN
callback = EarlyStopping(monitor='val_accuracy', patience=10,
                         restore_best_weights=True, mode="max")

epochs = 50
batch_size = 64
history = model.fit(trainX, trainY_cat,
                    validation_data=(testX, testY_cat),
                    batch_size=64,
                    epochs=50,
```

```
                        use_multiprocessing=True,
                        callbacks=[callback],
                        verbose=1
                        )
```

```
Epoch 1/50
71/71 [==============================] - 6s 84ms/step - loss: 3.1328 - accuracy:
0.1016 - val_loss: 2.3994 - val_accuracy: 0.1580
Epoch 2/50
71/71 [==============================] - 3s 39ms/step - loss: 2.5945 - accuracy:
0.1187 - val_loss: 2.6976 - val_accuracy: 0.1493
Epoch 3/50
71/71 [==============================] - 3s 39ms/step - loss: 2.5573 - accuracy:
0.1102 - val_loss: 2.3555 - val_accuracy: 0.1533
Epoch 4/50
71/71 [==============================] - 3s 39ms/step - loss: 2.2633 - accuracy:
0.1660 - val_loss: 2.1841 - val_accuracy: 0.1700
Epoch 5/50
71/71 [==============================] - 3s 40ms/step - loss: 2.1549 - accuracy:
0.2078 - val_loss: 2.1832 - val_accuracy: 0.1653
Epoch 6/50
71/71 [==============================] - 3s 39ms/step - loss: 2.1116 - accuracy:
0.2249 - val_loss: 2.1855 - val_accuracy: 0.1907
Epoch 7/50
71/71 [==============================] - 3s 39ms/step - loss: 2.0582 - accuracy:
0.2589 - val_loss: 2.1002 - val_accuracy: 0.2693
Epoch 8/50
71/71 [==============================] - 3s 39ms/step - loss: 2.0374 - accuracy:
0.2771 - val_loss: 2.0809 - val_accuracy: 0.2820
Epoch 9/50
71/71 [==============================] - 3s 39ms/step - loss: 2.0016 - accuracy:
0.2993 - val_loss: 2.1304 - val_accuracy: 0.3167
Epoch 10/50
71/71 [==============================] - 3s 39ms/step - loss: 2.1852 - accuracy:
0.2664 - val_loss: 2.1242 - val_accuracy: 0.2660
Epoch 11/50
71/71 [==============================] - 3s 39ms/step - loss: 2.0052 - accuracy:
0.3136 - val_loss: 2.0605 - val_accuracy: 0.2933
Epoch 12/50
71/71 [==============================] - 3s 39ms/step - loss: 1.9471 - accuracy:
0.3451 - val_loss: 2.0276 - val_accuracy: 0.2907
Epoch 13/50
71/71 [==============================] - 3s 39ms/step - loss: 1.9099 - accuracy:
0.3300 - val_loss: 2.0049 - val_accuracy: 0.3160
Epoch 14/50
71/71 [==============================] - 3s 39ms/step - loss: 1.8808 - accuracy:
0.3436 - val_loss: 1.9618 - val_accuracy: 0.2900
```

```
Epoch 15/50
71/71 [==============================] - 3s 39ms/step - loss: 1.8345 - accuracy:
0.3664 - val_loss: 1.9271 - val_accuracy: 0.3447
Epoch 16/50
71/71 [==============================] - 3s 39ms/step - loss: 1.8350 - accuracy:
0.3696 - val_loss: 1.9270 - val_accuracy: 0.3080
Epoch 17/50
71/71 [==============================] - 3s 39ms/step - loss: 1.7908 - accuracy:
0.3793 - val_loss: 1.8866 - val_accuracy: 0.3633
Epoch 18/50
71/71 [==============================] - 3s 39ms/step - loss: 1.7455 - accuracy:
0.3951 - val_loss: 2.0116 - val_accuracy: 0.3353
Epoch 19/50
71/71 [==============================] - 3s 39ms/step - loss: 1.7645 - accuracy:
0.3924 - val_loss: 1.8413 - val_accuracy: 0.3307
Epoch 20/50
71/71 [==============================] - 3s 39ms/step - loss: 1.6786 - accuracy:
0.4218 - val_loss: 1.9011 - val_accuracy: 0.3053
Epoch 21/50
71/71 [==============================] - 3s 40ms/step - loss: 1.6249 - accuracy:
0.4358 - val_loss: 1.8084 - val_accuracy: 0.3480
Epoch 22/50
71/71 [==============================] - 3s 39ms/step - loss: 1.6020 - accuracy:
0.4480 - val_loss: 1.7787 - val_accuracy: 0.3727
Epoch 23/50
71/71 [==============================] - 3s 39ms/step - loss: 1.5198 - accuracy:
0.4749 - val_loss: 1.8750 - val_accuracy: 0.3540
Epoch 24/50
71/71 [==============================] - 3s 39ms/step - loss: 1.5056 - accuracy:
0.4731 - val_loss: 1.7611 - val_accuracy: 0.4113
Epoch 25/50
71/71 [==============================] - 3s 39ms/step - loss: 1.4192 - accuracy:
0.5093 - val_loss: 1.7532 - val_accuracy: 0.3853
Epoch 26/50
71/71 [==============================] - 3s 39ms/step - loss: 1.3364 - accuracy:
0.5427 - val_loss: 1.6745 - val_accuracy: 0.4140
Epoch 27/50
71/71 [==============================] - 3s 39ms/step - loss: 1.2526 - accuracy:
0.5618 - val_loss: 1.7301 - val_accuracy: 0.4360
Epoch 28/50
71/71 [==============================] - 3s 39ms/step - loss: 1.1888 - accuracy:
0.5929 - val_loss: 1.7342 - val_accuracy: 0.4327
Epoch 29/50
71/71 [==============================] - 3s 40ms/step - loss: 1.0604 - accuracy:
0.6353 - val_loss: 1.7289 - val_accuracy: 0.4327
Epoch 30/50
71/71 [==============================] - 3s 39ms/step - loss: 1.0344 - accuracy:
0.6482 - val_loss: 1.6827 - val_accuracy: 0.4387
```

```
Epoch 31/50
71/71 [==============================] - 3s 39ms/step - loss: 0.9364 - accuracy:
0.6827 - val_loss: 1.8933 - val_accuracy: 0.4373
Epoch 32/50
71/71 [==============================] - 3s 39ms/step - loss: 0.9105 - accuracy:
0.6918 - val_loss: 1.9479 - val_accuracy: 0.4493
Epoch 33/50
71/71 [==============================] - 3s 39ms/step - loss: 0.8713 - accuracy:
0.7082 - val_loss: 1.8138 - val_accuracy: 0.4127
Epoch 34/50
71/71 [==============================] - 3s 39ms/step - loss: 0.8252 - accuracy:
0.7238 - val_loss: 1.8386 - val_accuracy: 0.4347
Epoch 35/50
71/71 [==============================] - 3s 39ms/step - loss: 0.7992 - accuracy:
0.7267 - val_loss: 1.8279 - val_accuracy: 0.4400
Epoch 36/50
71/71 [==============================] - 3s 39ms/step - loss: 0.7525 - accuracy:
0.7518 - val_loss: 1.9589 - val_accuracy: 0.4440
Epoch 37/50
71/71 [==============================] - 3s 39ms/step - loss: 0.7385 - accuracy:
0.7560 - val_loss: 1.9793 - val_accuracy: 0.4453
Epoch 38/50
71/71 [==============================] - 3s 39ms/step - loss: 0.6907 - accuracy:
0.7680 - val_loss: 1.9418 - val_accuracy: 0.4380
Epoch 39/50
71/71 [==============================] - 3s 39ms/step - loss: 0.6850 - accuracy:
0.7756 - val_loss: 1.9826 - val_accuracy: 0.4353
Epoch 40/50
71/71 [==============================] - 3s 39ms/step - loss: 0.6669 - accuracy:
0.7758 - val_loss: 2.1422 - val_accuracy: 0.4433
Epoch 41/50
71/71 [==============================] - 3s 39ms/step - loss: 0.6305 - accuracy:
0.7911 - val_loss: 2.3152 - val_accuracy: 0.4287
Epoch 42/50
71/71 [==============================] - 3s 39ms/step - loss: 0.6184 - accuracy:
0.7973 - val_loss: 2.1647 - val_accuracy: 0.4380
```

[9]:
```
# Evaluate the CNN
loss, acc = model.evaluate(testX, testY_cat, verbose=0)
print("CNN Accuracy using model.evaluate: {}".format(acc))
```

```
CNN Accuracy using model.evaluate: 0.4493333399295807
```

## 6 Question 5

[10]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Flatten, Dense
from tensorflow.keras.utils import to_categorical
# Process the data
os.chdir('/home/tuomas/Python/DATA.ML.200/Ex5')

trainX = np.load('X_train.npy')
trainX = trainX.reshape(-1,501,40)
trainY = np.load('y_train.npy')
trainY_cat = to_categorical(trainY, 15)

testX = np.load('X_test.npy')
testX = testX.reshape(-1,501,40)
testY = np.load('y_test.npy')
testY_cat = to_categorical(testY, 15)
```

[11]:
```
# Build the RNN
model = Sequential()

model.add(LSTM(units=32, return_sequences=True,␣
 ↪kernel_initializer='he_uniform', input_shape=(501,40)))
model.add(Flatten())
model.add(Dense(15, activation='softmax'))
model.
 ↪compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

[12]:
```
# Train the RNN
epochs = 50
batch_size = 32#16
history = model.fit(trainX, trainY_cat,
                    batch_size=batch_size,
                    epochs=epochs,
                    use_multiprocessing=True,
                    validation_data=(testX, testY_cat),
                    #callbacks=[callback],
                    verbose=1
                    )
```

```
Epoch 1/50
141/141 [==============================] - 3s 21ms/step - loss: 2.3836 -
accuracy: 0.2862 - val_loss: 1.8761 - val_accuracy: 0.3053
Epoch 2/50
141/141 [==============================] - 3s 19ms/step - loss: 1.5554 -
accuracy: 0.4891 - val_loss: 1.7439 - val_accuracy: 0.4233
Epoch 3/50
```

```
141/141 [==============================] - 3s 19ms/step - loss: 1.3068 -
accuracy: 0.5631 - val_loss: 1.6719 - val_accuracy: 0.4273
Epoch 4/50
141/141 [==============================] - 3s 19ms/step - loss: 1.1263 -
accuracy: 0.6360 - val_loss: 1.5927 - val_accuracy: 0.4820
Epoch 5/50
141/141 [==============================] - 3s 19ms/step - loss: 1.0088 -
accuracy: 0.6780 - val_loss: 1.5417 - val_accuracy: 0.5067
Epoch 6/50
141/141 [==============================] - 3s 19ms/step - loss: 0.8867 -
accuracy: 0.7149 - val_loss: 1.5727 - val_accuracy: 0.5127
Epoch 7/50
141/141 [==============================] - 3s 19ms/step - loss: 0.6927 -
accuracy: 0.7856 - val_loss: 1.4374 - val_accuracy: 0.5353
Epoch 8/50
141/141 [==============================] - 3s 19ms/step - loss: 0.6204 -
accuracy: 0.8124 - val_loss: 1.4539 - val_accuracy: 0.5260
Epoch 9/50
141/141 [==============================] - 3s 19ms/step - loss: 0.5757 -
accuracy: 0.8338 - val_loss: 1.5268 - val_accuracy: 0.5127
Epoch 10/50
141/141 [==============================] - 3s 19ms/step - loss: 0.4818 -
accuracy: 0.8622 - val_loss: 1.5199 - val_accuracy: 0.5267
Epoch 11/50
141/141 [==============================] - 3s 19ms/step - loss: 0.4292 -
accuracy: 0.8829 - val_loss: 1.5569 - val_accuracy: 0.5107
Epoch 12/50
141/141 [==============================] - 3s 19ms/step - loss: 0.3917 -
accuracy: 0.8960 - val_loss: 1.5875 - val_accuracy: 0.5760
Epoch 13/50
141/141 [==============================] - 3s 19ms/step - loss: 0.3532 -
accuracy: 0.9082 - val_loss: 1.4395 - val_accuracy: 0.5407
Epoch 14/50
141/141 [==============================] - 3s 19ms/step - loss: 0.2814 -
accuracy: 0.9336 - val_loss: 1.4081 - val_accuracy: 0.5407
Epoch 15/50
141/141 [==============================] - 3s 19ms/step - loss: 0.2515 -
accuracy: 0.9429 - val_loss: 1.4088 - val_accuracy: 0.5627
Epoch 16/50
141/141 [==============================] - 3s 19ms/step - loss: 0.2143 -
accuracy: 0.9560 - val_loss: 1.6449 - val_accuracy: 0.4980
Epoch 17/50
141/141 [==============================] - 3s 19ms/step - loss: 0.2092 -
accuracy: 0.9542 - val_loss: 1.7211 - val_accuracy: 0.5093
Epoch 18/50
141/141 [==============================] - 3s 19ms/step - loss: 0.1800 -
accuracy: 0.9653 - val_loss: 1.5291 - val_accuracy: 0.5507
Epoch 19/50
```

```
141/141 [==============================] - 3s 19ms/step - loss: 0.1456 -
accuracy: 0.9747 - val_loss: 1.4634 - val_accuracy: 0.5707
Epoch 20/50
141/141 [==============================] - 3s 19ms/step - loss: 0.1239 -
accuracy: 0.9822 - val_loss: 1.5205 - val_accuracy: 0.5947
Epoch 21/50
141/141 [==============================] - 3s 20ms/step - loss: 0.1047 -
accuracy: 0.9871 - val_loss: 1.5115 - val_accuracy: 0.5780
Epoch 22/50
141/141 [==============================] - 3s 20ms/step - loss: 0.0717 -
accuracy: 0.9951 - val_loss: 1.6406 - val_accuracy: 0.5387
Epoch 23/50
141/141 [==============================] - 3s 20ms/step - loss: 0.0963 -
accuracy: 0.9880 - val_loss: 1.5859 - val_accuracy: 0.5380
Epoch 24/50
141/141 [==============================] - 3s 20ms/step - loss: 0.0976 -
accuracy: 0.9867 - val_loss: 1.6052 - val_accuracy: 0.5633
Epoch 25/50
141/141 [==============================] - 3s 20ms/step - loss: 0.0727 -
accuracy: 0.9933 - val_loss: 1.7144 - val_accuracy: 0.5313
Epoch 26/50
141/141 [==============================] - 3s 20ms/step - loss: 0.0721 -
accuracy: 0.9902 - val_loss: 1.6216 - val_accuracy: 0.5687
Epoch 27/50
141/141 [==============================] - 3s 20ms/step - loss: 0.0706 -
accuracy: 0.9900 - val_loss: 1.6315 - val_accuracy: 0.5480
Epoch 28/50
141/141 [==============================] - 3s 20ms/step - loss: 0.0479 -
accuracy: 0.9987 - val_loss: 1.6432 - val_accuracy: 0.5800
Epoch 29/50
141/141 [==============================] - 3s 20ms/step - loss: 0.0769 -
accuracy: 0.9876 - val_loss: 1.6604 - val_accuracy: 0.5887
Epoch 30/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0328 -
accuracy: 0.9987 - val_loss: 1.5897 - val_accuracy: 0.5853
Epoch 31/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0216 -
accuracy: 0.9998 - val_loss: 1.6626 - val_accuracy: 0.5647
Epoch 32/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0974 -
accuracy: 0.9820 - val_loss: 1.7337 - val_accuracy: 0.5700
Epoch 33/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0284 -
accuracy: 0.9982 - val_loss: 1.7067 - val_accuracy: 0.5760
Epoch 34/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0160 -
accuracy: 1.0000 - val_loss: 1.7495 - val_accuracy: 0.5633
Epoch 35/50
```

```
141/141 [==============================] - 3s 19ms/step - loss: 0.0134 -
accuracy: 1.0000 - val_loss: 1.6221 - val_accuracy: 0.5980
Epoch 36/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0144 -
accuracy: 1.0000 - val_loss: 1.6117 - val_accuracy: 0.5880
Epoch 37/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0111 -
accuracy: 1.0000 - val_loss: 1.8765 - val_accuracy: 0.5720
Epoch 38/50
141/141 [==============================] - 3s 19ms/step - loss: 0.1942 -
accuracy: 0.9431 - val_loss: 1.6838 - val_accuracy: 0.5440
Epoch 39/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0425 -
accuracy: 0.9964 - val_loss: 1.6887 - val_accuracy: 0.5793
Epoch 40/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0141 -
accuracy: 0.9998 - val_loss: 1.7261 - val_accuracy: 0.5873
Epoch 41/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0104 -
accuracy: 1.0000 - val_loss: 1.7461 - val_accuracy: 0.5707
Epoch 42/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0078 -
accuracy: 1.0000 - val_loss: 1.7313 - val_accuracy: 0.5967
Epoch 43/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0066 -
accuracy: 1.0000 - val_loss: 1.7412 - val_accuracy: 0.5773
Epoch 44/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0060 -
accuracy: 1.0000 - val_loss: 1.7814 - val_accuracy: 0.5860
Epoch 45/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0053 -
accuracy: 1.0000 - val_loss: 1.7709 - val_accuracy: 0.5873
Epoch 46/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0056 -
accuracy: 1.0000 - val_loss: 1.7482 - val_accuracy: 0.5947
Epoch 47/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0048 -
accuracy: 1.0000 - val_loss: 1.7739 - val_accuracy: 0.5853
Epoch 48/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0053 -
accuracy: 1.0000 - val_loss: 1.7828 - val_accuracy: 0.5933
Epoch 49/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0042 -
accuracy: 1.0000 - val_loss: 1.7949 - val_accuracy: 0.5867
Epoch 50/50
141/141 [==============================] - 3s 19ms/step - loss: 0.0040 -
accuracy: 1.0000 - val_loss: 1.7664 - val_accuracy: 0.5893
```

[13]:
```python
# Evaluate the RNN
loss, acc = model.evaluate(testX, testY_cat, verbose=2)
print("Accuracy : {}".format(acc))
```

```
47/47 - 0s - loss: 1.7664 - accuracy: 0.5893
Accuracy : 0.5893333554267883
```

[ ]:
```python

```