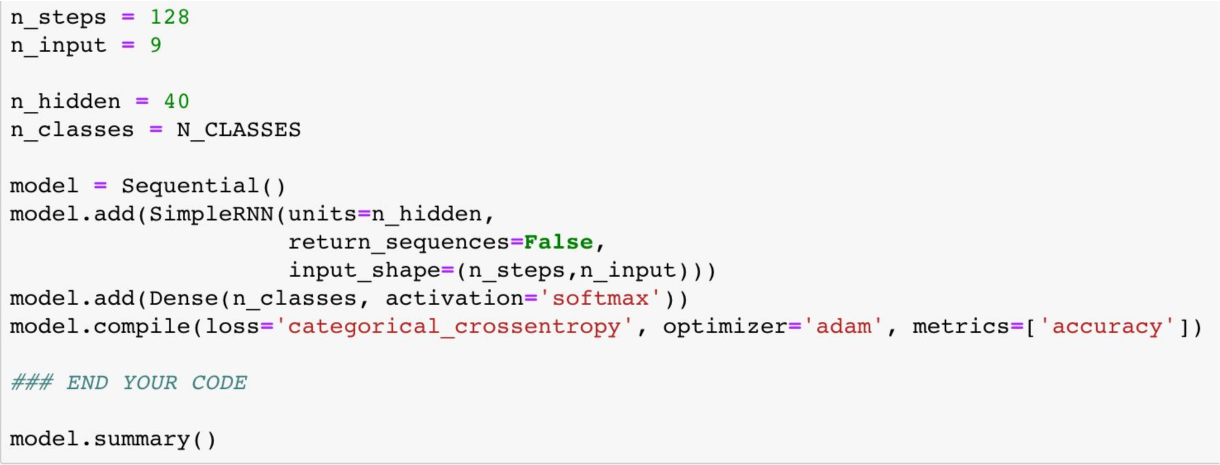
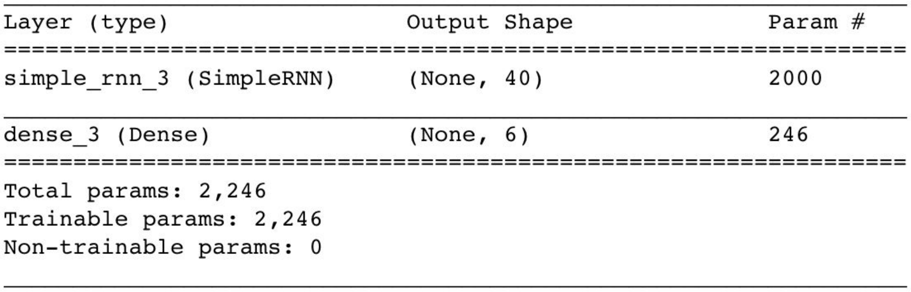
Ex 1

Using the following model:

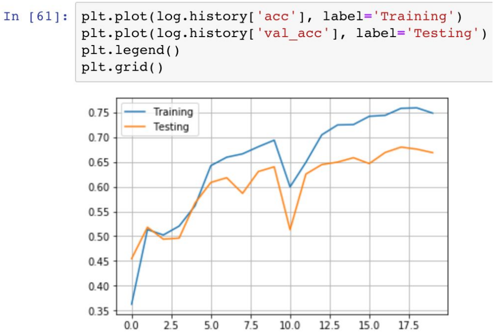


```
n_steps = 128
n_input = 9

n_hidden = 40
n_classes = N_CLASSES

model = Sequential()
model.add(SimpleRNN(units=n_hidden,
                    return_sequences=False,
                    input_shape=(n_steps,n_input)))
model.add(Dense(n_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

### END YOUR CODE

model.summary()
```

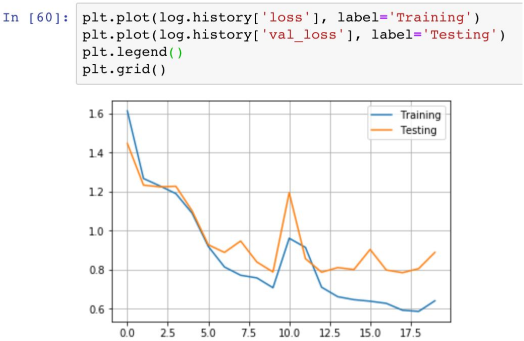


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_3 (SimpleRNN)     (None, 40)                2000
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 246
=================================================================
Total params: 2,246
Trainable params: 2,246
Non-trainable params: 0
_________________________________________________________________
```

Gives the following score:
Test score: 0.8889297847429048
Test accuracy: 0.6684764167151696



```
In [61]: plt.plot(log.history['acc'], label='Training')
         plt.plot(log.history['val_acc'], label='Testing')
         plt.legend()
         plt.grid()
```

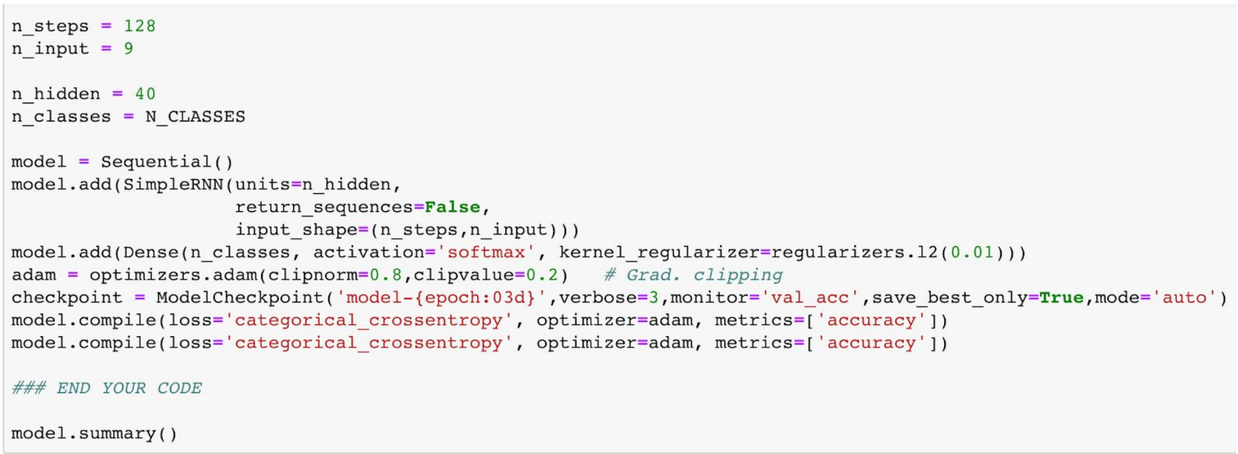


```
In [60]: plt.plot(log.history['loss'], label='Training')
         plt.plot(log.history['val_loss'], label='Testing')
         plt.legend()
         plt.grid()
```

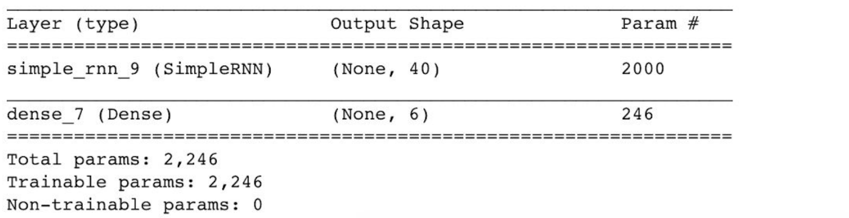
Confusion Matrix:

```
array([[106, 184, 143,  62,   1,   0],
       [ 61, 345,  45,  20,   0,   0],
       [100,  57, 223,  39,   1,   0],
       [  0,  23,   0, 416,  52,   0],
[  0,  22,   0, 137, 373,   0],
[  0,  28,   0,   2,   0, 507]])
```

Add Regularisation



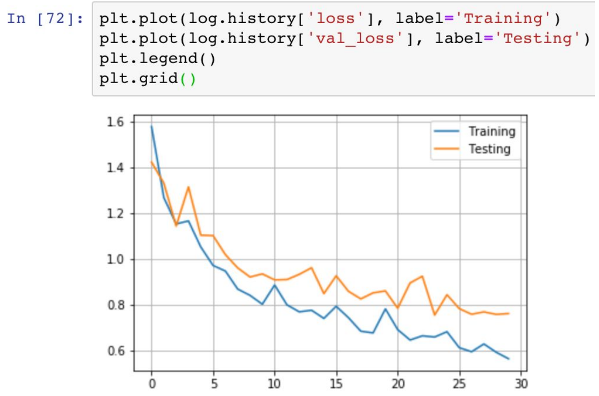
```
n_steps = 128
n_input = 9

n_hidden = 40
n_classes = N_CLASSES

model = Sequential()
model.add(SimpleRNN(units=n_hidden,
                    return_sequences=False,
                    input_shape=(n_steps,n_input)))
model.add(Dense(n_classes, activation='softmax', kernel_regularizer=regularizers.l2(0.01)))
adam = optimizers.adam(clipnorm=0.8,clipvalue=0.2)   # Grad. clipping
checkpoint = ModelCheckpoint('model-{epoch:03d}',verbose=3,monitor='val_acc',save_best_only=True,mode='auto')
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])

### END YOUR CODE

model.summary()
```



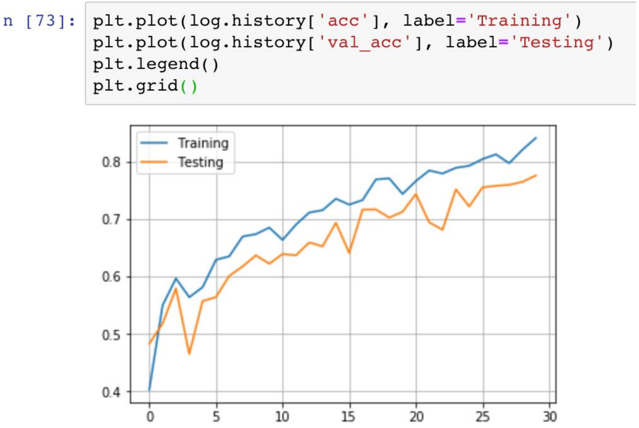
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
simple_rnn_9 (SimpleRNN)     (None, 40)                2000      
_________________________________________________________________
dense_7 (Dense)              (None, 6)                 246       
=================================================================
Total params: 2,246
Trainable params: 2,246
Non-trainable params: 0
```

Test score: 0.7603227934434449

Test accuracy: 0.7753647777603002



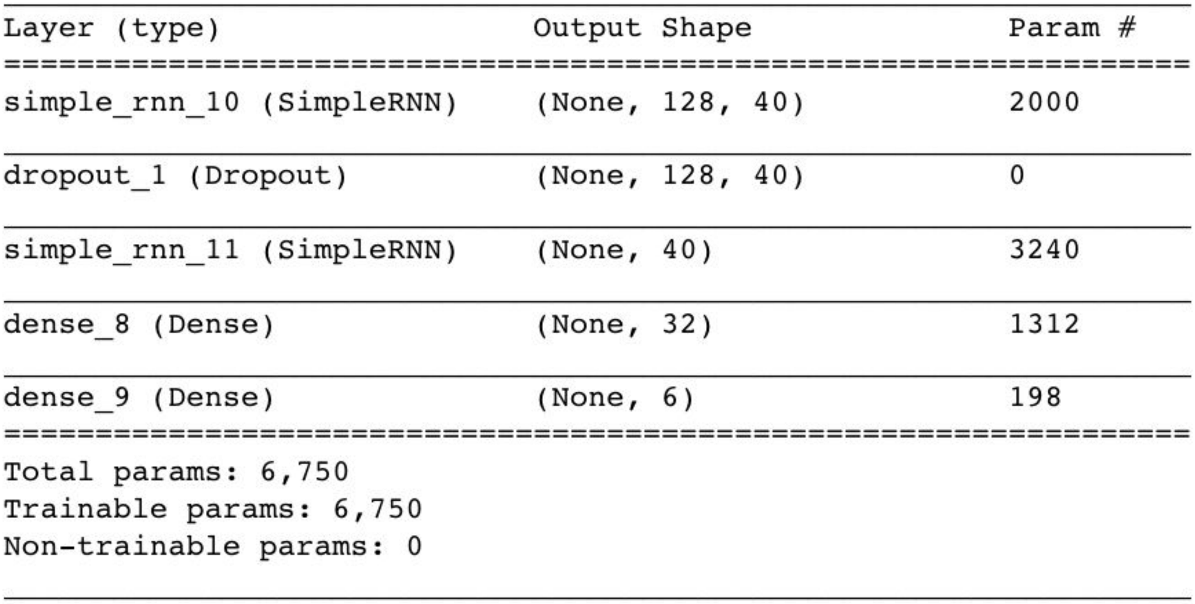
```
In [72]: plt.plot(log.history['loss'], label='Training')
         plt.plot(log.history['val_loss'], label='Testing')
         plt.legend()
         plt.grid()
```



```
n [73]: plt.plot(log.history['acc'], label='Training')
        plt.plot(log.history['val_acc'], label='Testing')
        plt.legend()
        plt.grid()
```

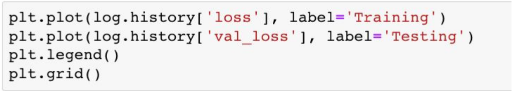
```
array([[414,   9,  50,  10,  13,   0],
       [119, 290,  45,   4,  13,   0],
       [137,   6, 276,   1,   0,   0],
       [  4,  14,   0, 341, 128,   4],
       [  3,   0,   0,  74, 455,   0],
       [  0,  27,   0,   1,   0, 509]])
```

Stacked Layers

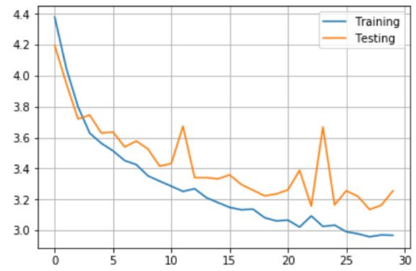


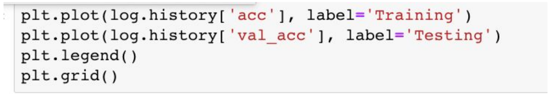
```
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_10 (SimpleRNN)    (None, 128, 40)           2000
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 40)           0
_________________________________________________________________
simple_rnn_11 (SimpleRNN)    (None, 40)                3240
_________________________________________________________________
dense_8 (Dense)              (None, 32)                1312
_________________________________________________________________
dense_9 (Dense)              (None, 6)                 198
=================================================================
Total params: 6,750
Trainable params: 6,750
Non-trainable params: 0
_________________________________________________________________
```

Test score: 1.070623964627476
Test accuracy: 0.8184594502884289

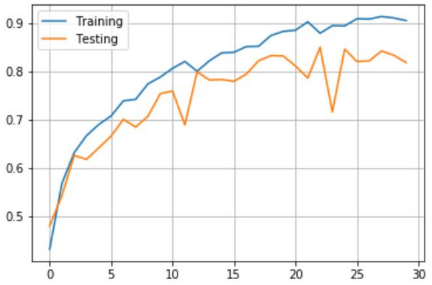


```
plt.plot(log.history['loss'], label='Training')
plt.plot(log.history['val_loss'], label='Testing')
plt.legend()
plt.grid()
```





```
plt.plot(log.history['acc'], label='Training')
plt.plot(log.history['val_acc'], label='Testing')
plt.legend()
plt.grid()
```



```
array([[468,   3,  24,   0,   0,   1],
       [106, 340,  25,   0,   0,   0],
       [ 70,   0, 350,   0,   0,   0],
       [  7,  19,   0, 409,  56,   0],
       [  1,  82,   0, 114, 335,   0],
       [  0,  26,   1,   0,   0, 510]])
```

Results

The best result was achieved with the stacking of the layer. This model also has the most parameters.