Giada Griso
September 3, 2024
Machine Learning Course
Exercise 2.2

Complex Machine Learning Models and Keras – Part 1

1. CNN (Convolution Neural Network) model

I decided to start with a CNN model because, although known to be better in the spatial and visual realms, it can run quickly and handle a larger amount of data.

I have run the model using different parameters and activation type.

```
epochs = 30
batch_size = 16
n_hidden = 64

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='sigmoid')) # Options: sigmoid, tanh, softmax, relu
```

- The smaller the batch size, the longer it takes to get the output for each epoch. 16 seems to be a good number.
- With “tanh” activation and 64 hidden layers, the loss gets stable at 31,63 and the accuracy increases up to 51,6%. The model can only recognise 5 out of 15 stations.
- With the same parameters, “sigmoid” can only recognise 1 weather station. The accuracy is around 12% and the loss grows exponentially.
- Softmax activation gives very similar results in terms of accuracy and loss than “sigmoid” with this number of hidden layers, but it can regognise 13 out of 15 stations.
- With the same parameters, “relu” activation gives stable accuracy (64,4%) and loss (‘nan’) after the first epoch. The model can recognise only 1 station though.

→ So far “softmax” activation seems to make the model recognise the highest number of stations. Let’s see what happens when we increase the number of hidden layers up to 128:

```
epochs = 30
batch_size = 16
n_hidden = 128

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='softmax')) # Options: sigmoid, tanh, softmax, relu
```

- The higher the number of layers, the longer it takes to get the output for each epoch.
- The accuracy slightly decreases from 14% to 11%, while the loss grows exponentially after each epoch.
- The model can recognise 13 out of 15 stations.
- When the number of layers is increased up to 256, the model can recognise all 15 weather stations. The accuracy stays low: 12,3%.

```
epochs = 30
batch_size = 32
n_hidden = 256

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='softmax')) # Options: sigmoid, tanh, softmax, relu
```

```
Epoch 1/30
538/538 - 2s - 4ms/step - accuracy: 0.0918 - loss: 4257.5908
Epoch 2/30
538/538 - 1s - 2ms/step - accuracy: 0.1213 - loss: 42290.4258
Epoch 3/30
538/538 - 1s - 2ms/step - accuracy: 0.1281 - loss: 153432.8438
Epoch 4/30
538/538 - 1s - 2ms/step - accuracy: 0.1345 - loss: 335217.5938
Epoch 5/30
538/538 - 1s - 2ms/step - accuracy: 0.1353 - loss: 637794.0000
Epoch 6/30
538/538 - 1s - 2ms/step - accuracy: 0.1421 - loss: 1031150.1250
Epoch 7/30
538/538 - 1s - 2ms/step - accuracy: 0.1409 - loss: 1569855.6250
Epoch 8/30
538/538 - 2s - 3ms/step - accuracy: 0.1373 - loss: 2190171.7500
Epoch 9/30
538/538 - 1s - 3ms/step - accuracy: 0.1389 - loss: 2903206.0000
Epoch 10/30
538/538 - 2s - 4ms/step - accuracy: 0.1404 - loss: 3745978.7500
Epoch 11/30
538/538 - 1s - 2ms/step - accuracy: 0.1391 - loss: 4753507.5000
Epoch 12/30
538/538 - 1s - 2ms/step - accuracy: 0.1372 - loss: 5916840.0000
Epoch 13/30
538/538 - 1s - 2ms/step - accuracy: 0.1366 - loss: 7022730.0000
Epoch 14/30
538/538 - 1s - 2ms/step - accuracy: 0.1346 - loss: 8415674.0000
Epoch 15/30
538/538 - 1s - 2ms/step - accuracy: 0.1358 - loss: 10039291.0000
```

```
Epoch 16/30
538/538 - 1s - 2ms/step - accuracy: 0.1353 - loss: 11701336.0000
Epoch 17/30
538/538 - 1s - 2ms/step - accuracy: 0.1327 - loss: 13513610.0000
Epoch 18/30
538/538 - 2s - 3ms/step - accuracy: 0.1313 - loss: 15582827.0000
Epoch 19/30
538/538 - 2s - 3ms/step - accuracy: 0.1327 - loss: 17631964.0000
Epoch 20/30
538/538 - 2s - 3ms/step - accuracy: 0.1297 - loss: 19806970.0000
Epoch 21/30
538/538 - 2s - 3ms/step - accuracy: 0.1313 - loss: 22273332.0000
Epoch 22/30
538/538 - 1s - 2ms/step - accuracy: 0.1260 - loss: 24858384.0000
Epoch 23/30
538/538 - 3s - 5ms/step - accuracy: 0.1243 - loss: 27716028.0000
Epoch 24/30
538/538 - 1s - 2ms/step - accuracy: 0.1238 - loss: 30569924.0000
Epoch 25/30
538/538 - 3s - 5ms/step - accuracy: 0.1213 - loss: 33929096.0000
Epoch 26/30
538/538 - 2s - 5ms/step - accuracy: 0.1220 - loss: 37048356.0000
Epoch 27/30
538/538 - 1s - 2ms/step - accuracy: 0.1197 - loss: 40416144.0000
Epoch 28/30
538/538 - 1s - 2ms/step - accuracy: 0.1224 - loss: 44066972.0000
Epoch 29/30
538/538 - 1s - 2ms/step - accuracy: 0.1231 - loss: 47778544.0000
Epoch 30/30
538/538 - 1s - 2ms/step - accuracy: 0.1228 - loss: 51947708.0000
```

180/180 ━━━━━━━━━━━━━━━━━━━━ **1s** 3ms/step

Pred / True	BASEL	BELGRADE	BUDAPEST	DEBILT	DUSSELDORF	HEATHROW	KASSEL \
BASEL	2	68	864	78	248	61	200
BELGRADE	0	89	77	4	78	1	6
BUDAPEST	0	15	9	3	21	0	0
DEBILT	0	2	0	0	14	0	0
DUSSELDORF	0	0	1	0	5	0	0
HEATHROW	0	6	2	3	13	0	1
KASSEL	0	1	0	0	1	0	0
LJUBLJANA	0	4	0	1	8	0	0
MAASTRICHT	0	0	0	2	2	0	1
MADRID	0	28	50	10	35	4	4
MUNCHENB	0	1	0	1	1	0	0
OSLO	0	2	0	0	0	0	0
STOCKHOLM	0	2	0	0	1	0	0
VALENTIA	0	0	0	0	0	0	0

Pred / True	LJUBLJANA	MAASTRICHT	MADRID	MUNCHENB	OSLO	SONNBLICK \
BASEL	1183	1	303	413	115	6
BELGRADE	642	0	20	142	9	0
BUDAPEST	120	0	9	18	3	0
DEBILT	49	0	2	11	0	0
DUSSELDORF	16	0	1	4	2	0
HEATHROW	36	0	8	9	1	0
KASSEL	7	0	0	0	1	0
LJUBLJANA	34	0	8	1	2	0
MAASTRICHT	2	0	0	2	0	0
MADRID	134	0	139	28	11	0
MUNCHENB	3	0	0	1	1	0
OSLO	2	0	0	1	0	0
STOCKHOLM	0	0	0	0	0	0
VALENTIA	0	0	0	0	0	0

Pred / True	STOCKHOLM	VALENTIA
BASEL	103	37
BELGRADE	24	0
BUDAPEST	16	0
DEBILT	4	0
DUSSELDORF	0	0
HEATHROW	3	0
KASSEL	1	0
LJUBLJANA	3	0
MAASTRICHT	0	0
MADRID	15	0
MUNCHENB	0	0
OSLO	0	0
STOCKHOLM	1	0
VALENTIA	1	0

2. RNN (Recurrent Neural Network) / LSTM model

Again I have run the model using different parameters and activation types.

```
epochs = 30
batch_size = 16
n_hidden = 32

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='sigmoid')) # Don't use relu here!
```

- It takes longer to run the LSTM model in comparison to the CNN.
- With these parameters, "sigmoid" provides a very low accuracy (5%) and a loss of 20,20. The model is able to recognise 2 out of 15 stations.
- With "tanh" activation, the model takes even longer to run, and it gets an accuracy of 4,37% and a loss of 23,95. It can recognise 6 out of 15 stations.
- With "softmax" activation, the model is even slower, and it gets an accuracy of 5,29% and a loss of 17,00. It can recognise 5 out of 15 stations.

→ So far "tanh" activation seems to make the model recognise the highest number of stations. Let's see what happens when we increase the number of hidden layers up to 64:

```
epochs = 30
batch_size = 16
n_hidden = 64

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='tanh')) # Don't use relu here!
```

- The accuracy increases up to 7,62% with a loss of 24.56. The model can recognise 5 out of 15 stations.

→ Let's see what happens when Conv1D and MaxPooling layers are added:

```
epochs = 30
batch_size = 16
n_hidden = 64

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(MaxPooling1D())
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='tanh')) # Don't use relu here!
```

```
Epoch 15/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1417 - loss: 24.1993
Epoch 16/30
1076/1076 - 5s - 4ms/step - accuracy: 0.0420 - loss: 24.6426
Epoch 17/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0454 - loss: 24.5047
Epoch 18/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0280 - loss: 24.5542
Epoch 19/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1158 - loss: 24.7097
Epoch 20/30
1076/1076 - 5s - 5ms/step - accuracy: 0.1292 - loss: 24.3296
Epoch 21/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0662 - loss: 24.3983
Epoch 22/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0372 - loss: 24.5850
Epoch 23/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0289 - loss: 24.6177
Epoch 24/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0542 - loss: 24.6508
Epoch 25/30
1076/1076 - 5s - 5ms/step - accuracy: 0.1007 - loss: 24.4422
Epoch 26/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0795 - loss: 23.9892
Epoch 27/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0389 - loss: 24.4042
Epoch 28/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0560 - loss: 24.1619
Epoch 29/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0672 - loss: 24.2516
Epoch 30/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0996 - loss: 24.9123

180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
Pred        LJUBLJANA  OSLO  STOCKHOLM
True
BASEL              12  3656         14
BELGRADE            0  1092          0
BUDAPEST            0   213          1
DEBILT              0    82          0
DUSSELDORF          0    29          0
HEATHROW            0    82          0
KASSEL              0    11          0
LJUBLJANA           0    61          0
MAASTRICHT          0     9          0
MADRID              0   458          0
MUNCHENB            0     8          0
OSLO                0     5          0
STOCKHOLM           0     4          0
VALENTIA            0     1          0
```

- The accuracy improves to 9,96% while the loss remains similar (24,91). The model can now recognise 3 out of 15 stations.
- When the number of hidden layers increases to 128, the accuracy reaches 10,2% and the model can recognise only 1 station. Adding more hidden layers doesn't seem to improve the model effectively.

In conclusion, we can state that by employing a RNN model:

- It takes longer to run the model compared to CNN.
- Tanh activation gets better results.
- The accuracy rate improves when adding Conv1D and MaxPooling layers, but in general it's lower than the level achieved by CNN.
- The level of losses is much lower than CNN.
- The model doesn't get to recognise all 15 weather stations.