Achievement 2.2: Complex Machine Learning Models and Keras Part 1

Olufemi Ige (01/26/2026)

1. **Keras Layered Model CNN (Convolution Neural Network)**
    a. I chose this model because CNN model provides a solid starting point, with softmax emerging as the most effective activation function for full weather-station recognition, despite ReLU achieving higher raw accuracy but exhibiting instability. The CNN is the best choice for this task because it matches the nature of the data, delivers stronger predictive performance, and is more efficient than RNN/LSTM for this use case. It performed better than the RNN/LSTM.
    b. The CNN model was selected as the initial approach because it trains efficiently and handles larger datasets well. Although CNNs are more commonly applied to spatial or image-based data, this model was tested to evaluate how well it could perform on time-series-like weather data, before comparing results with an RNN/LSTM model.
    c. The CNN was trained and evaluated multiple times using different hyperparameter configurations, as shown in the code and training outputs:
        - **Epochs tested**: 10, 15, and 30
        - **Batch sizes tested:** 4, 16, and 32
        - **Hidden layer sizes:** 2, 4, 16, 32, 128, and 256
        - **Activation functions tested:** ReLU, tanh, sigmoid, and softmax
    d. Activation function performance observations**:**
        - **ReLU**
            - Achieved the highest accuracy (64.4%)
              Loss increased to NaN, indicating overfitting and unstable training
        - **Tanh**
            - Produced a lower but stable accuracy (59.1%)
            - Loss stabilized at 26.79, suggesting better convergence
        - **Softmax and Sigmoid**
            - Showed lower accuracy overall
            - Were the only activations that correctly identified all 15 weather stations
            - Softmax performed best when:
                - Epochs = 30
                - Batch size = 16
                - Hidden layer size = 32
    e. As the hidden layer size increased and batch size decreased, training time increased significantly. In several runs, the model failed to converge, with accuracy remaining around 12% and loss increasing rapidly.
    f. While ReLU and tanh showed promising accuracy values, their inability to correctly classify all weather stations limits their suitability for this task.
    g. Softmax is currently the most appropriate activation function, as it enables full station recognition while maintaining acceptable accuracy and stable loss behavior.
    h. Overall, the CNN model demonstrates a strong baseline performance. With additional hyperparameter tuning and optimization, it shows potential to be further improved to better meet ClimateWins' classification needs.

Key trials & observations

• **Trial 1** (Epochs:**30,** batch size:**16**, hidden layer size:**32**): training accuracy stayed low (~0.10 –0.13) while loss grew very large, indicating an unstable objective / scaling issue in the setup.

• **Trial 2** (Epochs:**30,** batch size:**16**, hidden layer size:**128**): training stabilized around ~0.19 accuracy with loss ~20.42, but the confusion matrix shows strong confusion between stations.

• **Trial 3** (Epochs:**15,** batch size:**4**, hidden layer size:**4**): accuracy jumped to ~0.64, but loss became NaN, and evaluation shows the model predicting almost everything as a single station.

Trial 1 — CNN (unstable / very large loss)
Hyperparameters: Epochs:**30,** batch size:**16**, hidden layer size:**32**

[91]:
```
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(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
```

[98]:
```
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=2)
```

```
Epoch 1/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0998 - loss: 5821.7676
Epoch 2/30
1076/1076 - 3s - 2ms/step - accuracy: 0.1184 - loss: 58830.9180
Epoch 3/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1315 - loss: 207284.7344
Epoch 4/30
1076/1076 - 3s - 2ms/step - accuracy: 0.1323 - loss: 434324.7188
Epoch 5/30
1076/1076 - 3s - 2ms/step - accuracy: 0.1317 - loss: 786619.5625
Epoch 6/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1256 - loss: 1221632.5000
Epoch 7/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1264 - loss: 1785125.3750
Epoch 8/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1250 - loss: 2486798.7500
Epoch 9/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1250 - loss: 3290574.5000
Epoch 10/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1253 - loss: 4274683.0000
Epoch 11/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1240 - loss: 5469417.5000
Epoch 12/30
1076/1076 - 2s - 2ms/step - accuracy: 0.1230 - loss: 6801944.0000
Epoch 13/30
1076/1076 - 5s - 5ms/step - accuracy: 0.1202 - loss: 8285614.0000
Epoch 14/30
1076/1076 - 3s - 2ms/step - accuracy: 0.1169 - loss: 10028371.0000
Epoch 15/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1214 - loss: 11972306.0000
Epoch 16/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1228 - loss: 14073440.0000
Epoch 17/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1211 - loss: 16492661.0000
Epoch 18/30
1076/1076 - 3s - 2ms/step - accuracy: 0.1177 - loss: 18858558.0000
Epoch 19/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1151 - loss: 21929778.0000
Epoch 20/30
1076/1076 - 2s - 2ms/step - accuracy: 0.1196 - loss: 24905074.0000
Epoch 21/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1180 - loss: 28037724.0000
Epoch 22/30
1076/1076 - 2s - 2ms/step - accuracy: 0.1203 - loss: 31872742.0000
Epoch 23/30
Epoch 22/30
1076/1076 - 2s - 2ms/step - accuracy: 0.1203 - loss: 31872742.0000
Epoch 23/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1186 - loss: 35631620.0000
Epoch 24/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1171 - loss: 40001484.0000
Epoch 25/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1170 - loss: 44299540.0000
Epoch 26/30
1076/1076 - 3s - 2ms/step - accuracy: 0.1172 - loss: 48963760.0000
Epoch 27/30
1076/1076 - 3s - 2ms/step - accuracy: 0.1193 - loss: 54258112.0000
Epoch 28/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1213 - loss: 59864840.0000
Epoch 29/30
1076/1076 - 5s - 5ms/step - accuracy: 0.1177 - loss: 65832376.0000
Epoch 30/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1185 - loss: 72368880.0000
```

[98]:
```
<keras.src.callbacks.history.History at 0x2571fbb3b90>
```

Confusion Matrix

```
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

Pred	BASEL	BELGRADE	BUDAPEST	DEBILT	DUSSELDORF	HEATHROW	KASSEL
True							
BASEL	4	2143	5	88	53	7	78
BELGRADE	0	934	0	1	0	0	6
BUDAPEST	0	166	0	0	0	0	0
DEBILT	0	55	0	0	0	0	1
DUSSELDORF	0	16	0	0	0	0	0
HEATHROW	0	33	0	4	0	0	1
KASSEL	0	8	0	0	0	0	1
LJUBLJANA	0	44	0	0	0	0	0
MAASTRICHT	0	9	0	0	0	0	0
MADRID	0	182	0	19	0	0	2
MUNCHENB	0	7	0	0	0	0	0
OSLO	0	1	0	0	0	0	0
STOCKHOLM	0	3	0	0	0	0	0
VALENTIA	0	0	0	0	0	0	0

Pred	LJUBLJANA	MAASTRICHT	MADRID	MUNCHENB	OSLO	SONNBLICK
True						
BASEL	88	12	722	21	315	4
BELGRADE	0	0	37	0	101	0
BUDAPEST	0	0	11	0	32	0
DEBILT	0	0	5	0	17	0
DUSSELDORF	0	0	3	0	7	0
HEATHROW	0	0	17	0	22	0
KASSEL	0	0	0	0	2	0
LJUBLJANA	0	0	13	0	2	0
MAASTRICHT	0	0	0	0	0	0
MADRID	0	0	171	0	57	0
MUNCHENB	0	0	0	0	1	0
OSLO	0	0	0	0	4	0
STOCKHOLM	0	0	0	0	1	0
VALENTIA	0	0	0	0	1	0

Pred	STOCKHOLM	VALENTIA
True		
BASEL	110	32
BELGRADE	13	0
BUDAPEST	5	0
DEBILT	4	0
DUSSELDORF	3	0
HEATHROW	5	0
KASSEL	0	0
LJUBLJANA	2	0
MAASTRICHT	0	0
MADRID	27	0
MUNCHENB	0	0
OSLO	0	0
STOCKHOLM	0	0
VALENTIA	0	0

Trial 2 — CNN (stable but low separability)

Hyperparameters: Epochs:**30,** batch size:**16**, hidden layer size:**128**

[121]:
```
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='tanh')) # Options: sigmoid, tanh, softmax, relu
```

[127]:
```
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=2)
```

```
Epoch 1/30
1076/1076 - 7s - 6ms/step - accuracy: 0.2050 - loss: 20.4631
Epoch 2/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1997 - loss: 20.4248
Epoch 3/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1954 - loss: 20.4182
Epoch 4/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1932 - loss: 20.4079
Epoch 5/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1929 - loss: 20.4126
Epoch 6/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1921 - loss: 20.4182
Epoch 7/30
1076/1076 - 6s - 5ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 8/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 9/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 10/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 11/30
1076/1076 - 5s - 4ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 12/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 13/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 14/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 15/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 16/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 17/30
1076/1076 - 6s - 5ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 18/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 19/30
```


```
Epoch 19/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 20/30
1076/1076 - 5s - 5ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 21/30
1076/1076 - 3s - 3ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 22/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 23/30
1076/1076 - 4s - 3ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 24/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 25/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 26/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 27/30
1076/1076 - 6s - 6ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 28/30
1076/1076 - 6s - 6ms/step - accuracy: 0.1914 - loss: 20.4210
Epoch 29/30
1076/1076 - 4s - 4ms/step - accuracy: 0.1914 - loss: 20.4211
Epoch 30/30
1076/1076 - 5s - 5ms/step - accuracy: 0.1914 - loss: 20.4210
```


[127]:
```
<keras.src.callbacks.history.History at 0x257289bc5c0>
```

Confusion Matrix

[131]:
```
# Evaluate

print(confusion_matrix(y_test, model.predict(X_test)))
```

```
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
Pred        BASEL  BUDAPEST  HEATHROW  MAASTRICHT  MUNCHENB  SONNBLICK  \
True
BASEL        1036        20       493          52      1537         27
BELGRADE      512         0       149          49       376          0
BUDAPEST       77         0        44          11        79          0
DEBILT         58         0        13           1         9          0
DUSSELDORF     14         0        10           0         3          0
HEATHROW       21         0        25           4        28          0
KASSEL          5         0         3           0         3          0
LJUBLJANA       6         0         8           3        43          0
MAASTRICHT      0         0         1           0         8          0
MADRID         49         0        93          38       271          2
MUNCHENB        0         0         0           0         7          0
OSLO            0         0         2           0         1          0
STOCKHOLM       0         0         2           0         1          0
VALENTIA        0         0         0           0         1          0

Pred        VALENTIA
True
BASEL            517
BELGRADE           6
BUDAPEST           3
DEBILT             1
DUSSELDORF         2
HEATHROW           4
KASSEL             0
LJUBLJANA          1
MAASTRICHT         0
MADRID             5
MUNCHENB           1
OSLO               2
STOCKHOLM          1
VALENTIA           0
```

Loss has improved, but accuracy remains low, so let's try adjusting the layers.

Trial 3 — CNN (NaN loss + majority-class collapse)
Hyperparameters: Epochs:**15,** batch size:**4**, hidden layer size:**4**

[160]:
```
epochs = 15
batch_size = 4
n_hidden = 4

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='relu')) # Options: sigmoid, tanh, softmax, relu
```

```
[166]: model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbo
```

```
Epoch 1/15
4303/4303 - 14s - 3ms/step - accuracy: 0.0705 - loss: 27.4544
Epoch 2/15
4303/4303 - 13s - 3ms/step - accuracy: 0.4043 - loss: nan
Epoch 3/15
4303/4303 - 13s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 4/15
4303/4303 - 15s - 4ms/step - accuracy: 0.6440 - loss: nan
Epoch 5/15
4303/4303 - 14s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 6/15
4303/4303 - 15s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 7/15
4303/4303 - 13s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 8/15
4303/4303 - 15s - 4ms/step - accuracy: 0.6440 - loss: nan
Epoch 9/15
4303/4303 - 12s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 10/15
4303/4303 - 11s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 11/15
4303/4303 - 13s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 12/15
4303/4303 - 13s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 13/15
4303/4303 - 13s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 14/15
4303/4303 - 11s - 3ms/step - accuracy: 0.6440 - loss: nan
Epoch 15/15
4303/4303 - 12s - 3ms/step - accuracy: 0.6440 - loss: nan
```

```
[166]: <keras.src.callbacks.history.History at 0x2572bccbd40>
```

Confusion Matrix

```
[174]: # Evaluate

print(confusion_matrix(y_test, model.predict(X_test)))
```

```
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
Pred        BASEL
True
BASEL        3682
BELGRADE     1092
BUDAPEST      214
DEBILT         82
DUSSELDORF     29
HEATHROW       82
KASSEL         11
LJUBLJANA      61
MAASTRICHT      9
MADRID        458
MUNCHENB        8
OSLO            5
STOCKHOLM       4
VALENTIA        1
```

2. RNN and LSTM Models

Notebook: Achievement 2.2. RNN and LSTM

- **Goal:** compare three sequential model trials aimed at classifying weather-station data using recurrent and convolutional neural network architectures. Each trial varied one primary hyperparameter while keeping the number of epochs constant. The goal was to evaluate model stability, accuracy, and class-prediction behavior.
- **Approach:** experimented with different hidden sizes and architectures; one configuration combines Conv1D feature extraction with an LSTM layer.

Key trials & observations

• **Trial 1**-**Baseline LSTM Model**: (Epochs:**30,** batch size:**16**, hidden layer size:**128**): model struggled to learn meaningful patterns (very low accuracy (~0.016 → ~0.047) and spread predictions across a small subset of classes.
• **Trial 2 LSTM with Modified Batch Size / Hidden Units (Low Accuracy)**: (Epochs:**30,** batch size:**16**, hidden layer size:**64**): Accuracy improves and plateaus around ~0.368. Loss stabilizes around ~18.57 (no explosion).
• **Trial 3 Hybrid Conv1D Model (Low Accuracy, No Exploding Loss):** (Epochs:**30,** batch size:**16**, hidden layer size:**4**): improved performance with accuracy reaching ~0.37 and a stable loss (~18.57), although the confusion matrix indicates persistent class overlap (many stations still confused)

Trial 1 — RNN/LSTM (low accuracy)
Hyperparameters: Epochs:**30,** batch size:**16**, hidden layer size:**128**
Keras Layout:

```
[69]: 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='tanh')) # Options: sigmoid, tanh, softmax, relu
```

Model Output:

[75]:
```
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=2)
```

```
Epoch 1/30
1076/1076 - 5s - 4ms/step - accuracy: 0.0162 - loss: 24.7950
Epoch 2/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0302 - loss: 25.0228
Epoch 3/30
1076/1076 - 4s - 3ms/step - accuracy: 0.0379 - loss: 25.0705
Epoch 4/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0406 - loss: 25.0780
Epoch 5/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0424 - loss: 25.0789
Epoch 6/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0428 - loss: 25.0761
Epoch 7/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0444 - loss: 25.0780
Epoch 8/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0458 - loss: 25.0817
Epoch 9/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0461 - loss: 25.0836
Epoch 10/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0470 - loss: 25.0845
Epoch 11/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0845
Epoch 12/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 13/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 14/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 15/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 16/30
```

```
Epoch 15/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 16/30
1076/1076 - 3s - 2ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 17/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 18/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 19/30
1076/1076 - 4s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 20/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 21/30
1076/1076 - 4s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 22/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 23/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 24/30
1076/1076 - 4s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 25/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 26/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 27/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 28/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 29/30
1076/1076 - 3s - 3ms/step - accuracy: 0.0471 - loss: 25.0855
Epoch 30/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0471 - loss: 25.0855
```

Confusion Matrix

Pred	BELGRADE	BUDAPEST	DUSSELDORF	LJUBLJANA	MAASTRICHT	OSLO	\
True							
BASEL	791	169	163	1891	583	1	
BELGRADE	212	201	99	142	387	2	
BUDAPEST	49	14	19	24	107	0	
DEBILT	42	4	13	4	19	0	
DUSSELDORF	10	0	3	3	13	0	
HEATHROW	17	1	14	17	30	0	
KASSEL	5	0	1	0	5	0	
LJUBLJANA	1	1	5	17	32	1	
MAASTRICHT	0	0	1	1	6	0	
MADRID	18	2	28	292	100	0	
MUNCHENB	1	0	0	0	6	0	
OSLO	0	0	1	0	4	0	
STOCKHOLM	0	0	1	0	3	0	
VALENTIA	0	0	0	1	0	0	

Pred	SONNBLICK
True	
BASEL	84
BELGRADE	49
BUDAPEST	1
DEBILT	0
DUSSELDORF	0
HEATHROW	3
KASSEL	0
LJUBLJANA	4
MAASTRICHT	1
MADRID	18
MUNCHENB	1
OSLO	0
STOCKHOLM	0
VALENTIA	0

Trial 2 — RNN/LSTM with Modified Batch Size / Hidden Units (Low Accuracy, no exploding loss)

Hyperparameters: Epochs:**30,** batch size:**16**, hidden layer size:**64**

Keras Layout:

[83]:
```
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='tanh')) # Options: sigmoid, tanh, softmax, relu
```

Model Output:

[89]:
```
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verb
```

```
Epoch 1/30
1076/1076 - 5s - 4ms/step - accuracy: 0.0830 - loss: 26.0653
Epoch 2/30
1076/1076 - 3s - 3ms/step - accuracy: 0.2484 - loss: 27.9989
Epoch 3/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3414 - loss: 28.0278
Epoch 4/30
1076/1076 - 4s - 4ms/step - accuracy: 0.3150 - loss: 28.0184
Epoch 5/30
1076/1076 - 3s - 3ms/step - accuracy: 0.2653 - loss: 23.3297
Epoch 6/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 7/30
1076/1076 - 4s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 8/30
1076/1076 - 4s - 4ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 9/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 10/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 11/30
1076/1076 - 4s - 4ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 12/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 13/30
1076/1076 - 4s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 14/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 15/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 16/30
```

```
Epoch 15/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 16/30
1076/1076 - 4s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 17/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 18/30
1076/1076 - 4s - 4ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 19/30
1076/1076 - 2s - 2ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 20/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 21/30
1076/1076 - 4s - 4ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 22/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 23/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 24/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 25/30
1076/1076 - 3s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 26/30
1076/1076 - 4s - 3ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 27/30
1076/1076 - 4s - 4ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 28/30
1076/1076 - 4s - 4ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 29/30
1076/1076 - 5s - 5ms/step - accuracy: 0.3681 - loss: 18.5725
Epoch 30/30
1076/1076 - 4s - 4ms/step - accuracy: 0.3681 - loss: 18.5725
```

Confusion Matrix

```
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
180/180 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
```

Pred	BASEL	DUSSELDORF	HEATHROW	MAASTRICHT	MUNCHENB	SONNBLICK
True						
BASEL	2109	973	7	73	454	2
BELGRADE	977	109	0	1	3	0
BUDAPEST	206	8	0	0	0	0
DEBILT	82	0	0	0	0	0
DUSSELDORF	29	0	0	0	0	0
HEATHROW	81	1	0	0	0	0
KASSEL	11	0	0	0	0	0
LJUBLJANA	60	1	0	0	0	0
MAASTRICHT	8	1	0	0	0	0
MADRID	285	167	2	2	2	0
MUNCHENB	8	0	0	0	0	0
OSLO	5	0	0	0	0	0
STOCKHOLM	4	0	0	0	0	0
VALENTIA	1	0	0	0	0	0

Pred	VALENTIA
True	
BASEL	64
BELGRADE	2
BUDAPEST	0
DEBILT	0
DUSSELDORF	0
HEATHROW	0
KASSEL	0
LJUBLJANA	0
MAASTRICHT	0
MADRID	0
MUNCHENB	0
OSLO	0
STOCKHOLM	0
VALENTIA	0

Trial 3 — Hybrid Conv1D + LSTM (best of the tested runs)
Hyperparameters: Epochs:**30,** batch size:**16**, hidden layer size:6**4**

[96]:
```
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
```

Model Output:

[102]:
```
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=2)
```

```
Epoch 1/30
1076/1076 - 5s - 5ms/step - accuracy: 0.6251 - loss: 12311.9707
Epoch 2/30
1076/1076 - 3s - 2ms/step - accuracy: 0.6432 - loss: 123929.5859
Epoch 3/30
1076/1076 - 3s - 2ms/step - accuracy: 0.6434 - loss: 409433.0938
Epoch 4/30
1076/1076 - 3s - 3ms/step - accuracy: 0.6435 - loss: 888315.8125
Epoch 5/30
1076/1076 - 4s - 4ms/step - accuracy: 0.6434 - loss: 1597337.8750
Epoch 6/30
1076/1076 - 3s - 3ms/step - accuracy: 0.6435 - loss: 2536331.7500
Epoch 7/30
1076/1076 - 5s - 5ms/step - accuracy: 0.6436 - loss: 3636220.5000
Epoch 8/30
1076/1076 - 4s - 4ms/step - accuracy: 0.6436 - loss: 5137453.5000
Epoch 9/30
1076/1076 - 4s - 4ms/step - accuracy: 0.6436 - loss: 6783524.5000
Epoch 10/30
1076/1076 - 4s - 4ms/step - accuracy: 0.6436 - loss: 8738813.0000
Epoch 11/30
1076/1076 - 4s - 4ms/step - accuracy: 0.6437 - loss: 11094854.0000
Epoch 12/30
1076/1076 - 4s - 4ms/step - accuracy: 0.6437 - loss: 13736234.0000
Epoch 13/30
1076/1076 - 8s - 8ms/step - accuracy: 0.6437 - loss: 16854642.0000
Epoch 14/30
1076/1076 - 6s - 5ms/step - accuracy: 0.6437 - loss: 20314708.0000
Epoch 15/30
1076/1076 - 4s - 4ms/step - accuracy: 0.6437 - loss: 23862800.0000
Epoch 16/30
```

```
Epoch 16/30
1076/1076 - 6s - 6ms/step - accuracy: 0.6437 - loss: 28302738.0000
Epoch 17/30
1076/1076 - 9s - 9ms/step - accuracy: 0.6437 - loss: 33084254.0000
Epoch 18/30
1076/1076 - 8s - 8ms/step - accuracy: 0.6437 - loss: 38313800.0000
Epoch 19/30
1076/1076 - 5s - 5ms/step - accuracy: 0.6437 - loss: 44372360.0000
Epoch 20/30
1076/1076 - 4s - 4ms/step - accuracy: 0.6437 - loss: 50400308.0000
Epoch 21/30
1076/1076 - 4s - 3ms/step - accuracy: 0.6437 - loss: 57680480.0000
Epoch 22/30
1076/1076 - 7s - 6ms/step - accuracy: 0.6437 - loss: 65293548.0000
Epoch 23/30
1076/1076 - 6s - 5ms/step - accuracy: 0.6437 - loss: 73515352.0000
Epoch 24/30
1076/1076 - 3s - 3ms/step - accuracy: 0.6437 - loss: 81908000.0000
Epoch 25/30
1076/1076 - 9s - 8ms/step - accuracy: 0.6437 - loss: 91745256.0000
Epoch 26/30
1076/1076 - 8s - 8ms/step - accuracy: 0.6437 - loss: 102083264.0000
Epoch 27/30
1076/1076 - 5s - 4ms/step - accuracy: 0.6437 - loss: 113276944.0000
Epoch 28/30
1076/1076 - 5s - 5ms/step - accuracy: 0.6437 - loss: 123897720.0000
Epoch 29/30
1076/1076 - 5s - 5ms/step - accuracy: 0.6437 - loss: 136805040.0000
Epoch 30/30
1076/1076 - 5s - 5ms/step - accuracy: 0.6437 - loss: 149282400.0000
```

[102]:
```
<keras.src.callbacks.history.History at 0x15290e0dbe0>
```

Confusion Matrix

[106]:
```
# Evaluate

print(confusion_matrix(y_test, model.predict(X_test)))
```

```
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Pred        BASEL  VALENTIA
True
BASEL        3680         2
BELGRADE     1092         0
BUDAPEST      214         0
DEBILT         82         0
DUSSELDORF     29         0
HEATHROW       82         0
KASSEL         11         0
LJUBLJANA      61         0
MAASTRICHT      9         0
MADRID        458         0
MUNCHENB        8         0
OSLO            5         0
STOCKHOLM       4         0
VALENTIA        1         0
```