Kyle Stanford


# Complex Machine Learning Models and Keras

Convolution Neural Networks (CNN) and Recurrent Neural Networks (RNN) were investigated for ClimateWins. I estimate that the RNN models may perform better due to their advantage in being able to handle temporal data. Although, temporal data is technically being removed from the dataset to ensure both models can run properly. Thus, it is unclear if either model has an edge over the other.

## Part 1 – Convolution Neural Network (CNN)

To start testing and adjusting the model, I began with a low number of hidden layers and a random activation type. I figured I would keep the number of epochs and batch size consistent to not drag out this process too long:

```
# Define hyperparamters at the top for easy adjustments
epochs = 30
batch_size = 16
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='softmax')) # options include: softmax, sigmoid, tanh, or relu
```

*Figure 1 - Attempt 1 (activation = softmax, layers = 4, accuracy ≈ 11.5%, loss increased exponentially, 9 stations recognised)*

```
180/180 ━━━━━━━━━━━━━━━━━━━━ 0s 939us/step
```

| Pred / True | BELGRADE | BUDAPEST | DEBILT | HEATHROW | KASSEL | LJUBLJANA | MADRID | \ | MUNCHENB | STOCKHOLM |
|---|---|---|---|---|---|---|---|---|---|---|
| BASEL | 4 | 1129 | 15 | 7 | 366 | 5 | 1546 | | 303 | 307 |
| BELGRADE | 0 | 749 | 0 | 0 | 8 | 0 | 333 | | 2 | 0 |
| BUDAPEST | 0 | 121 | 0 | 0 | 0 | 0 | 92 | | 1 | 0 |
| DEBILT | 0 | 37 | 0 | 0 | 2 | 0 | 43 | | 0 | 0 |
| DUSSELDORF | 0 | 9 | 0 | 0 | 0 | 0 | 19 | | 1 | 0 |
| HEATHROW | 0 | 17 | 0 | 0 | 1 | 0 | 62 | | 2 | 0 |
| KASSEL | 0 | 7 | 0 | 0 | 0 | 0 | 3 | | 1 | 0 |
| LJUBLJANA | 0 | 21 | 0 | 0 | 0 | 0 | 38 | | 2 | 0 |
| MAASTRICHT | 0 | 2 | 0 | 0 | 2 | 0 | 5 | | 0 | 0 |
| MADRID | 0 | 71 | 0 | 0 | 1 | 0 | 378 | | 8 | 0 |
| MUNCHENB | 0 | 3 | 0 | 0 | 0 | 0 | 4 | | 1 | 0 |
| OSLO | 0 | 4 | 0 | 0 | 0 | 0 | 1 | | 0 | 0 |
| STOCKHOLM | 0 | 3 | 0 | 0 | 0 | 0 | 1 | | 0 | 0 |
| VALENTIA | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | 0 | 0 |

*Figure 2 – First Confusion Matrix*

**Adjustments & Observations:**

- The first **softmax** model recognised 9/15 stations but showed low accuracy (≈11.5%) and the loss function increased exponentially.
- Increasing the number of hidden layers meant the model could recognise all 15 stations, but the loss function was always increasing, and the accuracy tends to hover around 12%
    - 8, 16, 32, 64, and 128 layers were trialled. No correlation between accuracy and number of hidden layers.
    - Model would sometimes fail to recognise all 15 stations (stayed above 9)
- Changing the activation type to **"sigmoid"** lead to very similar accuracy scores, loss function increases, and recognising less stations
    - Increasing the number of hidden layers for the "sigmoid activation led to insignificant changes (difference of ≈1% in accuracy and only 1 station being recognised)
- Changing the activation type to **"tanh"** resulted in the model fully stabilising after epoch 4, with a loss of 22.9, accuracy of 25.5%, and 5 stations recognised
    - On another attempt, the same hyperparameters ran completely differently.
    - Sometimes the model is able to get its accuracy up to over 50% with 16 or 32 layers, although the same hyperparameters would perform poorly when rerun.
    - Consistently, 5 to 8 stations were recognised and the loss plateaus at 22 to 28
- Changing the activation type to **"relu"** gives an accuracy of 64.4% and a "nan" for the loss function. Perhaps this is an error.
    - Changing the number of hidden layers did not change anything.
    - Only 1 station was recognised

**Summary of CNN:**

It appears the softmax activation is best for recognising the highest number of stations the most consistently. It seems to do this best when more hidden layers are present (32 and higher), however its accuracy is low, and the model does not converge. Other activation types like tanh and relu are more capable of increasing accuracy, but the model doesn't recognise most stations (which seems counterintuitive).

Overall, the CNN model struggled to perform well under any circumstance. There also appears to be a somewhat random nature to the model's performance. The following images show the final version of the CNN model using softmax and 128 hidden layers. In this case, it recognised all 15 stations but did not do this consistently.

```python
# Define hyperparamters at the top for easy adjustments
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 include: softmax, sigmoid, tanh, or relu
```

```
Epoch 1/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0961 - loss: 21371.2832
Epoch 2/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1055 - loss: 222759.6719
Epoch 3/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1130 - loss: 698795.0000
Epoch 4/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1156 - loss: 1546897.2500
Epoch 5/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1160 - loss: 2711850.7500
Epoch 6/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1168 - loss: 4201885.5000
Epoch 7/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1141 - loss: 6211947.5000
Epoch 8/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1199 - loss: 8517501.0000
Epoch 9/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1170 - loss: 11720648.0000
Epoch 10/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1139 - loss: 15284291.0000
Epoch 11/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1136 - loss: 19545926.0000
Epoch 12/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1153 - loss: 24304422.0000
Epoch 13/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1196 - loss: 29894926.0000
Epoch 14/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1156 - loss: 36135524.0000
Epoch 15/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1189 - loss: 43256648.0000
Epoch 16/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1148 - loss: 50803336.0000
Epoch 17/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1168 - loss: 59605676.0000
Epoch 18/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1179 - loss: 68991824.0000
Epoch 19/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1175 - loss: 79372208.0000
Epoch 20/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1201 - loss: 90216208.0000
Epoch 21/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1186 - loss: 102886408.0000
Epoch 22/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1182 - loss: 116042184.0000
Epoch 23/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1174 - loss: 130297632.0000
Epoch 24/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1196 - loss: 146544896.0000
Epoch 25/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1183 - loss: 163559088.0000
Epoch 26/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1165 - loss: 182005840.0000
Epoch 27/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1154 - loss: 200482208.0000
Epoch 28/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1171 - loss: 222266496.0000
Epoch 29/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1172 - loss: 244349808.0000
Epoch 30/30
1076/1076 - 1s - 1ms/step - accuracy: 0.1175 - loss: 266731696.0000
```

180/180 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step

| Pred<br>True | BASEL | BELGRADE | BUDAPEST | DEBILT | DUSSELDORF | HEATHROW | KASSEL | LJUBLJANA | MAASTRICHT | MADRID | MUNCHENB | OSLO | SONNBLICK | STOCKHOLM | VALENTIA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASEL | 322 | 147 | 348 | 667 | 63 | 54 | 9 | 84 | 165 | 1627 | 38 | 45 | 8 | 72 | 33 |
| BELGRADE | 44 | 28 | 190 | 84 | 9 | 5 | 0 | 1 | 79 | 638 | 0 | 4 | 0 | 10 | 0 |
| BUDAPEST | 2 | 4 | 22 | 30 | 2 | 2 | 0 | 1 | 7 | 140 | 0 | 1 | 0 | 3 | 0 |
| DEBILT | 1 | 0 | 9 | 20 | 0 | 0 | 0 | 0 | 5 | 44 | 0 | 0 | 0 | 3 | 0 |
| DUSSELDORF | 0 | 0 | 2 | 11 | 0 | 1 | 0 | 0 | 2 | 12 | 0 | 0 | 0 | 1 | 0 |
| HEATHROW | 0 | 0 | 1 | 24 | 0 | 1 | 0 | 0 | 3 | 50 | 0 | 2 | 0 | 1 | 0 |
| KASSEL | 0 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 1 | 4 | 0 | 0 | 0 | 1 | 0 |
| LJUBLJANA | 1 | 0 | 12 | 4 | 0 | 1 | 0 | 3 | 1 | 39 | 0 | 0 | 0 | 0 | 0 |
| MAASTRICHT | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 5 | 0 | 0 | 0 | 0 | 0 |
| MADRID | 11 | 1 | 18 | 91 | 1 | 21 | 0 | 8 | 12 | 271 | 0 | 8 | 0 | 16 | 0 |
| MUNCHENB | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 5 | 0 | 1 | 0 | 0 | 0 |
| OSLO | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 0 |
| STOCKHOLM | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| VALENTIA | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Part 2 – Convolution Neural Network (CNN)

Just like the CNN model, I have run this model using various hyperparameters and activation types. I started with the sigmoid activation and 4 hidden layers:

```
from tensorflow.keras.layers import LSTM

# Define hyperparamters at the top for easy adjustments
epochs = 30
batch_size = 16
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(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='sigmoid')) # options include: softmax, sigmoid, or tanh (DON'T USE RELU HERE)
```

*Figure 3 – Hyperparameters for first iteration of RNN model (activation = sigmoid, layers = 4)*

```
180/180 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Pred         BASEL  BELGRADE  MADRID
True
BASEL         1471         0    2211
BELGRADE       993         0      99
BUDAPEST       207         0       7
DEBILT          82         0       0
DUSSELDORF      27         0       2
HEATHROW        70         0      12
KASSEL          11         0       0
LJUBLJANA       48         0      13
MAASTRICHT       4         0       5
MADRID         167         1     290
MUNCHENB         7         0       1
OSLO             4         0       1
STOCKHOLM        4         0       0
VALENTIA         0         0       1
```

*Figure 4 – Confusion Matrix for first iteration of RNN model (only 3 stations recognised)*

### Adjustments & Observations:

- First model with **"sigmoid"** activation showed a decreasing accuracy (18.4% → 6.8%) and increasing loss (9.1 → 12.9) over 30 epochs. Only 3 stations were recognised.
  - ➢ Rerunning the model without changes led to similar results
  - ➢ Rerunning the model with 8 hidden layers started promising for the first 5 epochs, but beyond this accuracy begins to drop and loss increases (2 stations recognised)
  - ➢ Increasing the hidden layers further worsened the accuracy and loss whilst only recognising 2 stations
- Changing the activation type to **"softmax"** lead to similar results, recognising only 1 or 2 stations, finishing the final epoch with a low accuracy of ≈6% and loss of ≈11

    - Increasing the number of hidden layers seems to generally increase the number of stations recognised (up to 6 stations at 64 hidden layers), but the accuracy and loss remain poor.
- Changing the activation type to **“tanh”** led to the most interesting results:
    - Occasionally, at 4 layers, the model would return a loss of “nan”. This may suggest the network is too shallow, causing unstable gradients. More hidden layers fixed this issue.
    - The number of stations recognised did not seem to correlate with the number of hidden layers. 4 layers and 32 layers recognised 7 stations.
    - For 8 hidden layers or more, accuracy fluctuated widely, in some cases ranging between less than 1% and above 30% over 30 epochs. (May suggest the model needs epochs/time to converge)
        - However, ***64 layers*** seems to be most consistent at improving towards the final epoch, finishing with a higher accuracy score overall (***up to 11.6%***).
    - Other than with 4 hidden layers, the loss function was consistent across all models, fluctuating without any clear trend.
- Adding a Convolution and MaxPooling layer did not seem to change the results in any significant way. In fact, the accuracy here was lower than without the layers.

**Summary of RNN:**

Tanh activation seems best for recognising the highest number of stations (up to 7), although there doesn’t seem to be any obvious correlation between this and the number of hidden layers. Also, the accuracy, whilst low, seemed greater by the final epoch more consistently with more hidden layers. However, accuracy always fluctuated quite greatly during the 30 epochs. It is likely this model requires either a higher number of epochs to converge, a lower dropout rate, or both.

The sigmoid and softmax activation methods, whilst having a lower loss function consistently, recognised less stations and never showed improvements in their accuracy under every scenario. This may also be a failure caused by the dropout rate, epochs, or some other cause.

**Final Summary:**

Overall, the RNN model performed worse than the CNN model in many ways, especially in recognising all stations. Moreover, the time taken for the RNN model to run was far greater than the CNN model. However, both models may be improved by changing different hyperparameters like the number of epochs and dropout.

## Final model and confusion matrix:

```
from tensorflow.keras.layers import LSTM

# Define hyperparamters at the top for easy adjustments
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')) # options include: softmax, sigmoid, or tanh (DON'T USE RELU HERE)
```

```
Epoch 1/30
1076/1076 - 4s - 4ms/step - accuracy: 0.0245 - loss: 24.3459
Epoch 2/30
1076/1076 - 2s - 2ms/step - accuracy: 0.1211 - loss: 23.6739
Epoch 3/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0783 - loss: 23.8056
Epoch 4/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0175 - loss: 24.6209
Epoch 5/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0168 - loss: 25.3151
Epoch 6/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0232 - loss: 24.8667
Epoch 7/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0249 - loss: 24.8033
Epoch 8/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0286 - loss: 24.7598
Epoch 9/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0320 - loss: 24.6778
Epoch 10/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0841 - loss: 24.9842
Epoch 11/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0846 - loss: 24.6049
Epoch 12/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0941 - loss: 24.7990
Epoch 13/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0452 - loss: 24.4368
Epoch 14/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0351 - loss: 24.5168
Epoch 15/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0359 - loss: 24.4963
Epoch 16/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0151 - loss: 24.1834
Epoch 17/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0070 - loss: 24.5315
Epoch 18/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0078 - loss: 24.7838
Epoch 19/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0100 - loss: 24.6592
Epoch 20/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0135 - loss: 24.3823
Epoch 21/30
1076/1076 - 3s - 2ms/step - accuracy: 0.0153 - loss: 24.6034
Epoch 22/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0172 - loss: 24.5619
Epoch 23/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0192 - loss: 24.4344
Epoch 24/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0346 - loss: 24.8454
Epoch 25/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0327 - loss: 24.5512
Epoch 26/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0374 - loss: 24.5035
Epoch 27/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0447 - loss: 24.2939
Epoch 28/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0567 - loss: 24.4860
Epoch 29/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0385 - loss: 24.4483
Epoch 30/30
1076/1076 - 2s - 2ms/step - accuracy: 0.0310 - loss: 24.5407
```

```
180/180 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Pred        BUDAPEST  SONNBLICK
True
BASEL           3681          1
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
```

**Note:** This is the final model experimented with but not what I would consider a good "launching point" for ClimateWins.