TASK 1

First, I chose which features to use in my dataset. The original data includes information from many European countries, but since I am only working with Denmark, I kept only the columns related to Denmark. To do this, I selected all columns that start with “DK”, and also kept the utc_timestamp column to track the time.

Next, based on the README file, I selected three important features: total energy load, wind power generation, and solar power generation. These columns are:

1.'DK_load_actual_entsoe_transparency',
2.'DK_wind_generation_actual',
3.'DK_solar_generation_actual'

I also used the utc_timestamp column to determine the season (spring, summer, autumn, or winter) for each day.

Then, I checked for missing values in the data:

	missing_data_rows	mising_data_percentage
utc_timestamp	0	0.000000
DK_load_actual_entsoe_transparency	2	0.003968
DK_wind_generation_actual	2	0.003968
DK_solar_generation_actual	11	0.021825

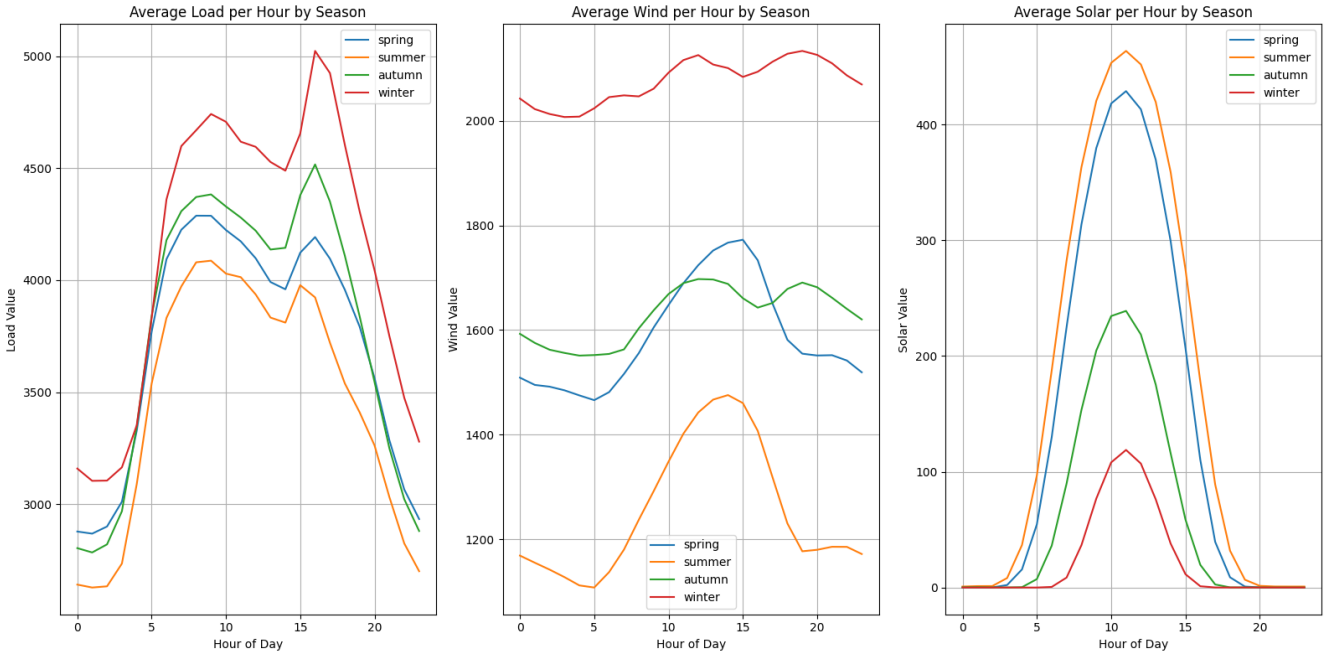
Because the amount of missing data was very small, I decided to simply remove those rows. This has little to no effect on the model’s performance.

After cleaning the data, I converted the timestamps to seasons and checked how many samples I had for each season. The distribution was fairly balanced, which is good for training a model.

I then split the data into training, validation, and test sets. I used stratified sampling by season, so that each set contains a similar number of samples for each season. This helps the model learn more equally.

I also used a StandardScaler to scale the values in each set. This makes sure that the model trains better and faster.

To better understand the data, I made line plots showing the average 24-hour pattern for each season. I made one plot each for load, wind, and solar power.



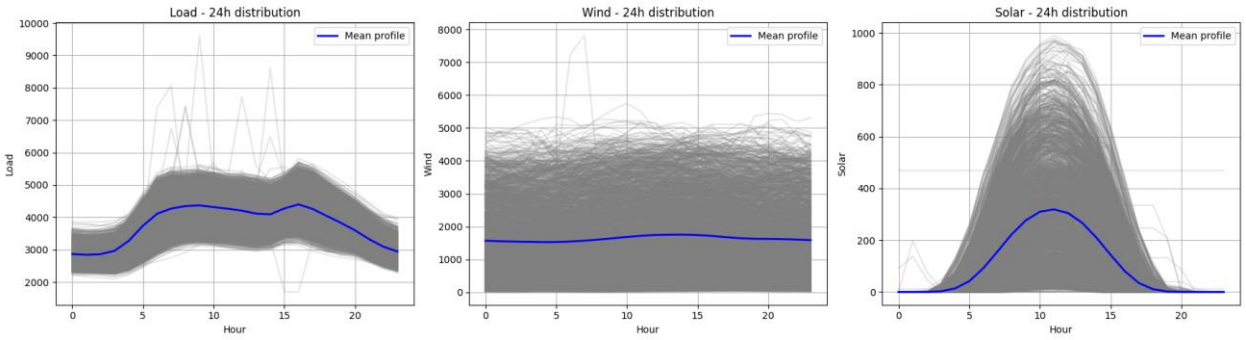
On all graphs, spring and autumn are located between summer and winter because they are transition seasons without any sharp differences. This is especially noticeable on the second graph, where the average wind is almost the same, which can cause some difficulties in determining these two seasons.

On the first graph, I see the dependence of average load per hour in all seasons. The lowest consumption is in summer and the highest is in winter. This is logical because in summer you do not need to spend money on heating, and you do not need to use lamplight as much as in winter, since daylight in summer in the northern hemisphere is much longer than in winter.

On the second graph, I see that winter is much windier than summer.

On the third graph, I see that in summer and spring much more solar energy is produced than in winter and autumn. This happens because of the length of daylight. In winter and autumn, the sun's angle is lower, so sunlight passes through more atmosphere, losing energy.

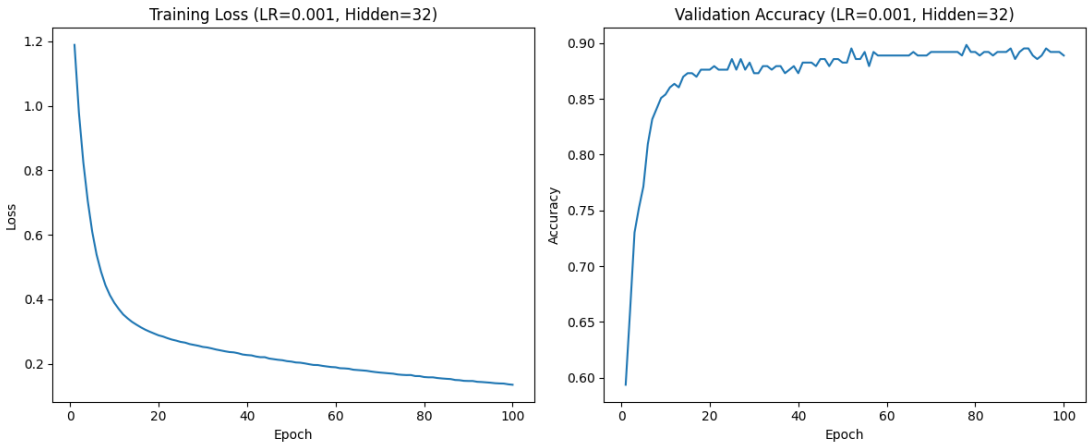
All things considered, it will be more difficult to separate spring from autumn than the other seasons.



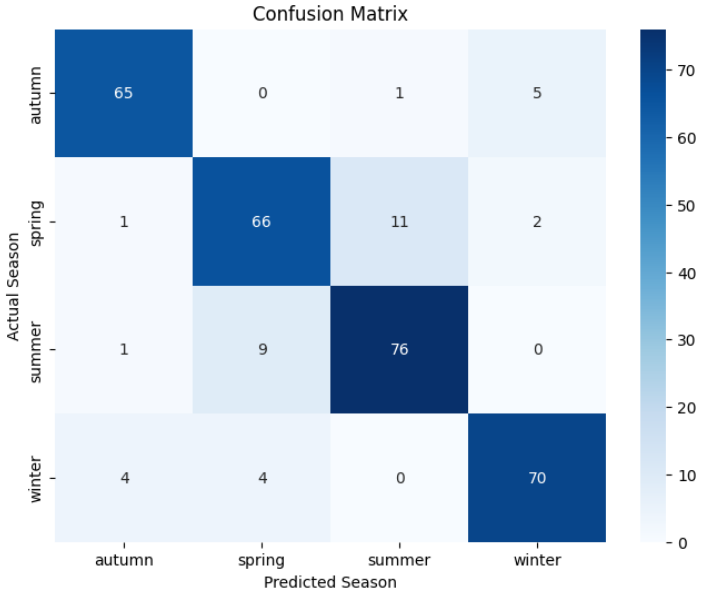
On this graph we see the distribution of the features, it helps us to use best Scaler. It is described more detailed in notebook.

TASK 2

The easiest way to define best hyperparameters in model is just to make grid-search and choose the model with best hyperparameters.



It is graphs which show training loss and validation accuracy. I experimented a little bit with num of epochs and I see that 100 was the best.

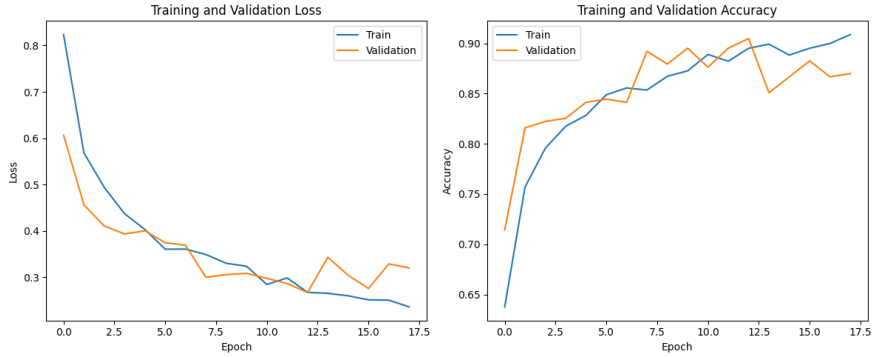


On this confusion matrix we see that the most difficult task to the model is to define differences with summer and spring.

TASK 3

Same situation here, I also used grid search to find best hyperparameters.

The first block converts the 3 input channels into 32 feature maps using a Conv1d layer with ReLU activation.To prevent overfitting, dropout is applied after these layers before the flattened feature map is processed through fully connected layers that reduce the features from 384 to 128 with ReLU activation, and finally map the representation to 4 target classes.

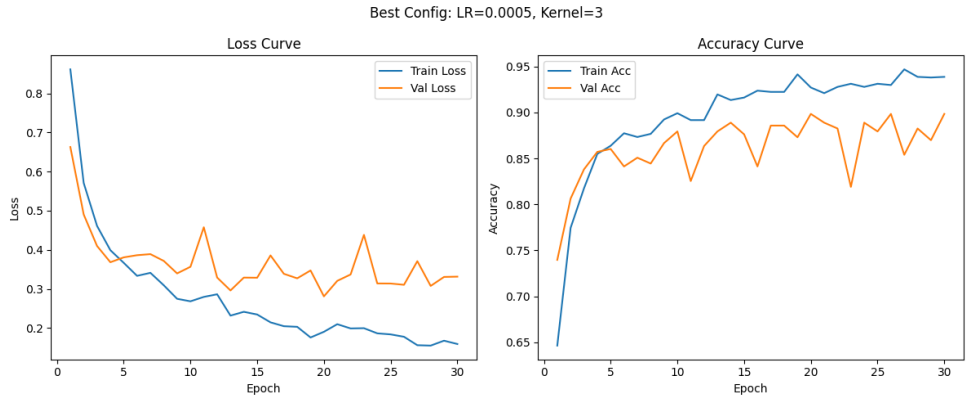


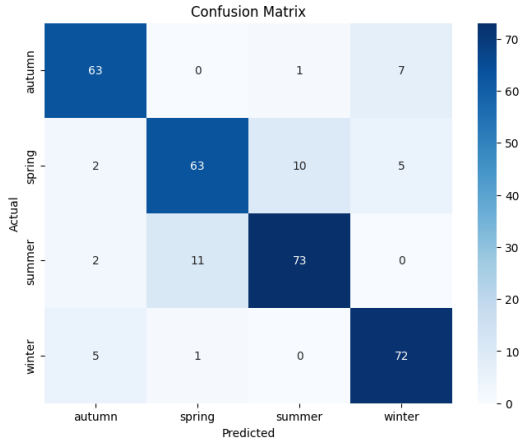
Here the graph of validation. I will not attach confusion matrix here because it is close to the previous task. You can see it in notebook.

TASK 4

Same situation here, I also used grid search to find best hyperparameters.

The decision to reshape the 24-hour time series into a 6×4 matrix (Option B) is driven by these key advantages. First, this approach preserves local temporal relationships by grouping consecutive hours into rows and columns, enabling the CNN to detect meaningful short-term patterns—such as differences between morning and evening energy consumption—without distorting the inherent structure of the data. Second, the transformation ensures compatibility with standard 2D convolutional layers (nn.Conv2d), which are optimized for grid-like inputs. By reformatting the data into a 4D tensor (batch, channels, height=6, width=4), the model aligns with PyTorch's expected input dimensions, as noted in the official documentation: "*Input: (N, C, H, W) […] where N is the batch size, C is the number of channels, H and W are the height and width of the input data*" (PyTorch Conv2d Docs https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html). Also, the method is computationally efficient, requiring no additional preprocessing overhead compared to more complex techniques like Gramian Angular Fields (Option A).





Conclusion:

Accuracies:

Task2 `0.8889`

Task3 `0.8857`

Task4 `0.8603`

All the models have almost same accuracies. I think that is because I used grid search which gave me a possibility to use best parameters everywhere and have good models.