Assignment 2. Intro to Machine Learning

Tsukanov Nikita, DS-02

Task 1: Exploratory Data Analysis (EDA) and Preprocessing

1.1 Selected Columns (from README):

- `DK_solar_generation_actual` — Actual solar generation in Denmark (MW)
- `DK_load_actual_entsoe_transparency` — Total electricity load in Denmark (MW)
- `DK_wind_generation_actual` — Actual wind generation in Denmark (MW)

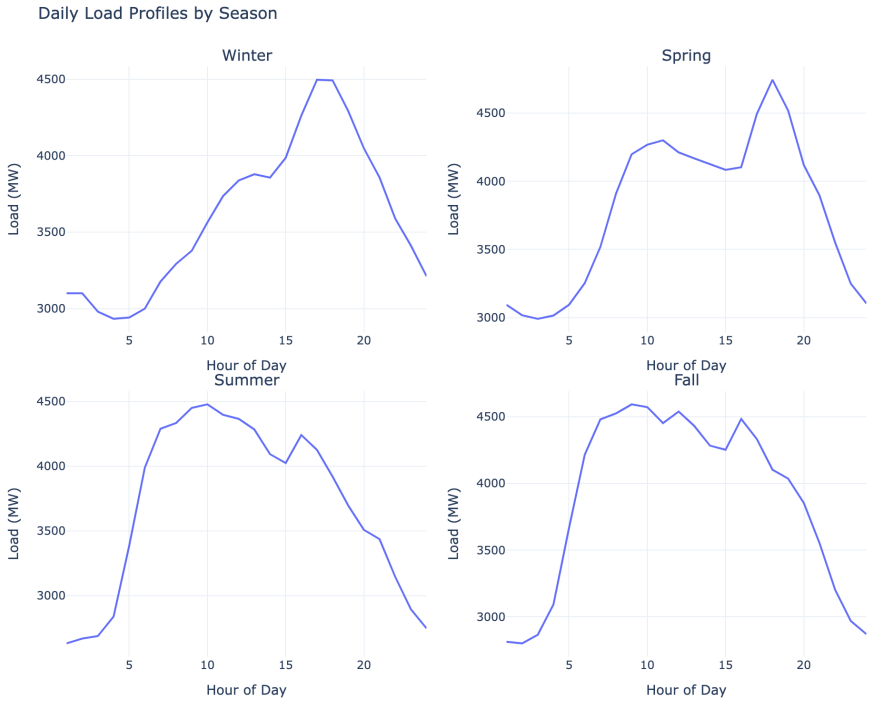
1.2 Data Inspection:

- Dataset shape: **(50400, 4)**
- Missing values were imputed using **time-based interpolation**, which is suitable for energy data with regular sampling intervals.
- Remaining gaps (e.g., at the start/end) were filled using **forward/backward filling** to preserve data continuity without discarding valuable sequences.
- The data was grouped daily. Days with incomplete hourly records were removed to maintain fixed input lengths.
- Resulting samples were labeled by meteorological seasons (Spring, Summer, Fall, Winter).

1.3 Season Distribution:

```
Spring: 552 samples
Summer: 552 samples
Winter: 511 samples
Fall:   485 samples
```

1.4 Brief Analysis:



1. Spring days exhibit the highest average electricity load.
2. Peak loads frequently occur around midday, suggesting a consistent daily usage pattern.

Task 2: Baseline MLP (Fully Connected Network)

All plots for train and validation loss and accuracy score are in the notebook

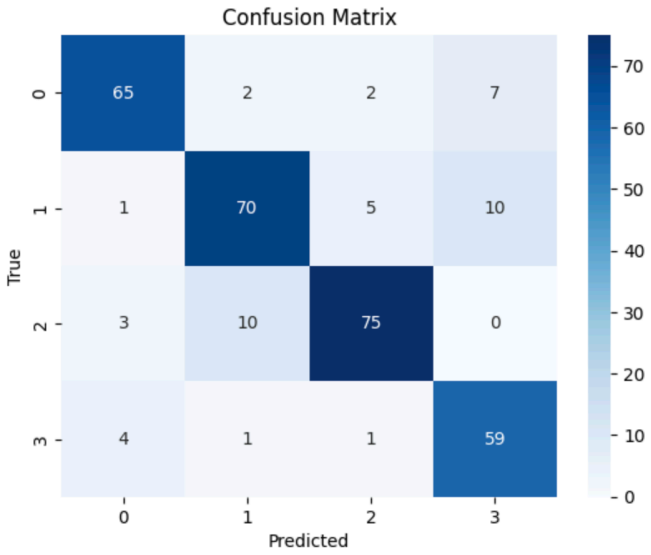
Model Architecture:

- **Input Layer:** 72 features (24 hourly values for each of load, solar, and wind).
- **Hidden Layer:** One dense layer with 128 units and ReLU activation.
- **Output Layer:** A dense layer with 4 units (season classes), followed by `CrossEntropyLoss`.

Hyperparameters:

- **Epochs:** 65, selected by evaluating training and validation performance trends.
- **Batch Size:** 128, balancing learning stability and memory efficiency.

Accuracy on test set – 0.853



Task 3: 1D CNN on Raw Time-Series

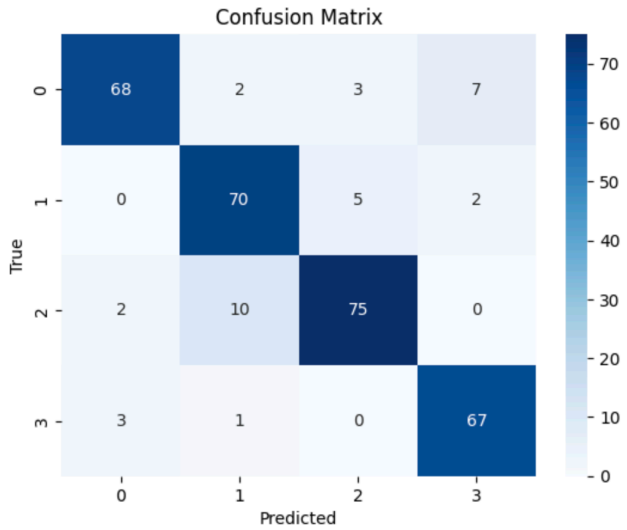
Model Architecture:

- **Input Shape:** `(3 channels × 24 time steps)` corresponding to load, solar, and wind.
- **Conv1D Layers:**
  - Conv1D-64 → ReLU → MaxPool1D
  - Conv1D-128 → ReLU → MaxPool1D
  - Conv1D-256 → ReLU → MaxPool1D
- **Dropout Layers:** Regularization to reduce overfitting.
- **Fully Connected Layer:** Dense layer with `256×3` units.
- **Output Layer:** Softmax activation via `CrossEntropyLoss`.

Hyperparameters:

- **Epochs:** 100, selected by evaluating training and validation performance trends.
- **Batch Size:** 64

Accuracy on test set – 0.888



Task 4: 2D CNN on Transformed Data

2D Transform Justification:

The time series for each energy type (load, solar, wind) is reshaped from 1D (24,) into 2D (6×4) "images". These are stacked across channels to form a tensor of shape `(3, 6, 4)`. This transformation is inspired by techniques used in **time-series image encoding**, which enable CNNs to capture spatial dependencies and interactions between variables over time.

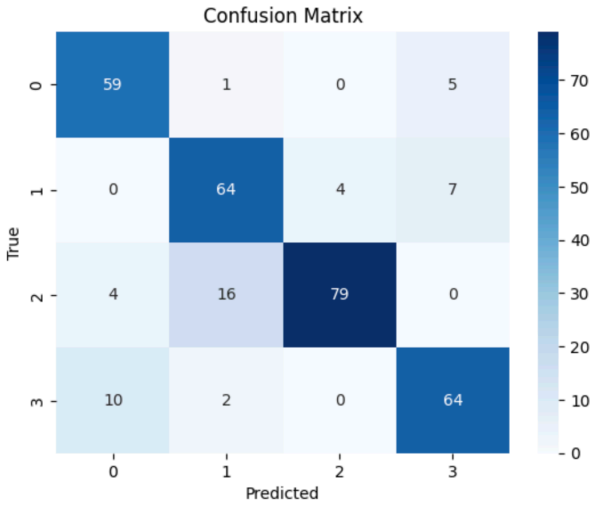
Model Architecture:

- **Input Shape:** `(3, 6, 4)`
- **Conv2D Layers:**
  - Conv2D-64 → ReLU
  - Conv2D-128 → ReLU
  - MaxPooling (2×2)
- **Dropout Layer:** To reduce overfitting.
- **Fully Connected Layers:**
  - Dense 768 → 64 → 4 output classes.

Hyperparameters:

- **Epochs:** 60
- **Batch Size:** 128

Accuracy on test set – 0.866



Handling Data Splits & Leakage

To **prevent seasonal data leakage**, the dataset is split chronologically rather than randomly. This strategy mimics real-world forecasting, ensuring that the model only learns from past seasons and is evaluated on future ones. Random splits would artificially mix seasonal patterns and inflate performance.

Results

CNN 1D outperformed all the other models. I think that it introduces useful **local connectivity and weight sharing**, becoming ideal for detecting **temporal patterns** in structured time series. The MLP, by contrast, requires more data to generalize well and lacks any understanding of temporal order or proximity between inputs. The 2D CNN transforms the sequence into a pseudo-image format (3 channels × 6×4 grid). While this introduces some spatial structure, it may distort or weaken the true **temporal alignment** between features. The model can still extract useful correlations but might not learn time-specific dependencies as efficiently as the 1D CNN.