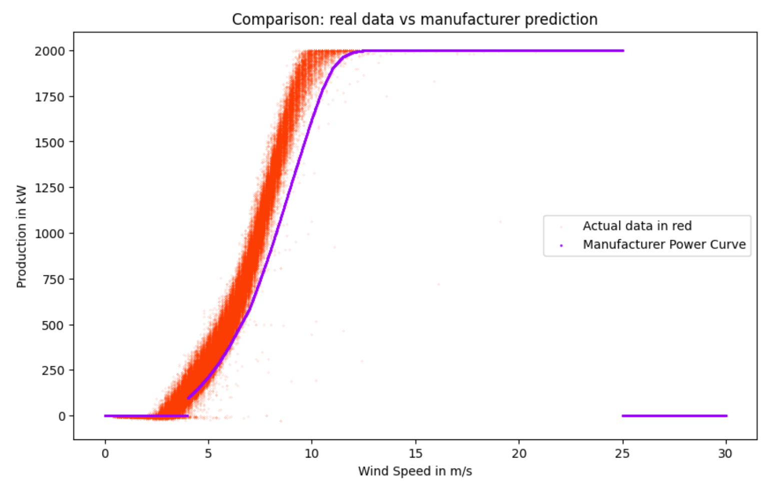
Improving energy production expectations of wind energy systems using deep learning

Paul Wecker
943726

BHT Berliner Hochschule für Technik

Task

- Assessing a wind energy system's production performance requires knowledge about how much energy (kW) we can expect for given conditions (e.g. wind speed and temperature)
- Most manufacturers provide a „power curve" (MPV), informing us on how the relationship between wind speed (m/s) and energy production:
- However this power curve isn't always reliable or even available
- Also, the power curve neglects factors which have an impact on energy production, e.g. outside temperature

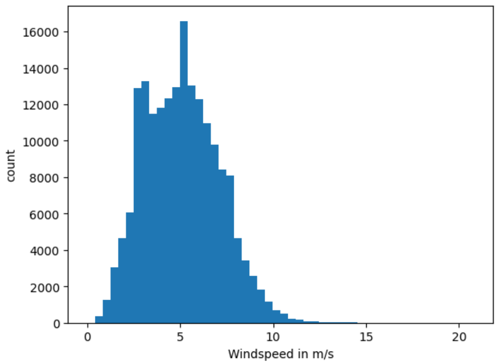


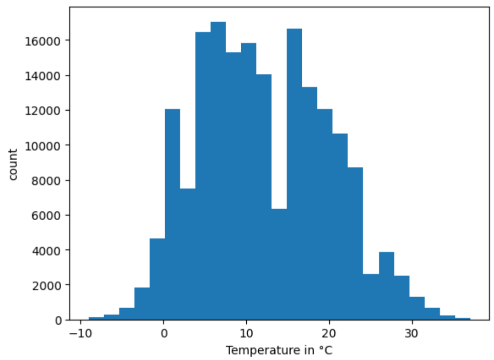
→ Task: Build neural network outperforming the manufacturer's power curve

Data Set

- Dataset comprises standardized operational data of a single wind energy system over five years on 10-minute granularity
- Periods of inspections, repairs, faults etc are excluded
- data represents „normal operating behaviour“
- Features: wind_speed, temperature; target: power

timestamp	power	wind_speed	temperature
2018-11-30 08:40:00+01:00	1411.400024	8.3	1.0
2018-11-30 08:50:00+01:00	1201.000000	7.7	1.0
2018-11-30 09:00:00+01:00	1136.000000	7.5	1.0
2018-11-30 09:10:00+01:00	1351.099976	8.2	1.0






```
Dataset with size: 184702
Mean of windspeed: 5.1 m/s
Std of windspeed: 2.01
Mean of temperature: 12.14 °C
Std of temperature: 7.8
Mean of output power: 434.83 in kW
Std of output power: 474.95
```

Model

- I experimented with MLPs and ended up with:
  - using batchnorm
  - using 2 – 4 layers
  - with 8 – 32 neurons
  - Relus
  - Sigmoid due to its similarity to typical power curves

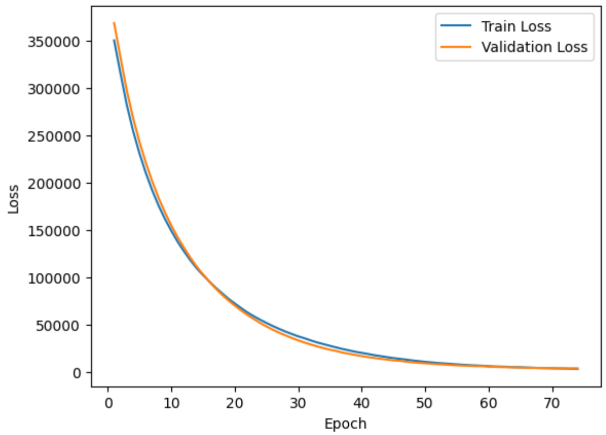
```python
class PowerPredictionModel(nn.Module):
    def __init__(self, input_dim):
        super(PowerPredictionModel, self).__init__()
        self.batch_norm = nn.BatchNorm1d(input_dim)
        self.fc1 = nn.Linear(input_dim, 12)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(12, 24)
        self.relu2 = nn.ReLU()
        self.fc3 = nn.Linear(24, 12)
        self.relu3 = nn.ReLU()
        self.fc4 = nn.Linear(12, 12)
        self.sig = nn.Sigmoid()
        self.fc5 = nn.Linear(12, 1)  # Output layer with 1 neuron (regression)
```

- Nevergrad was used to find a good learning rate:

```python
# set logarithmic search space for learning rate search
instrumentation = ng.p.Instrumentation(learn_rate=ng.p.Log(lower=0.0001, upper=.1))
optimizer = ng.optimizers.NGOpt(parametrization=instrumentation,
                                budget=20) # try out 20 learning rates

# start search & retrieve recommendation
recommender = optimizer.minimize(create_train_model)
recommended_learning_rate = recommender.value[1]['learn_rate']
print(f"Recommended value for learning rate: f{recommended_learning_rate}")
```

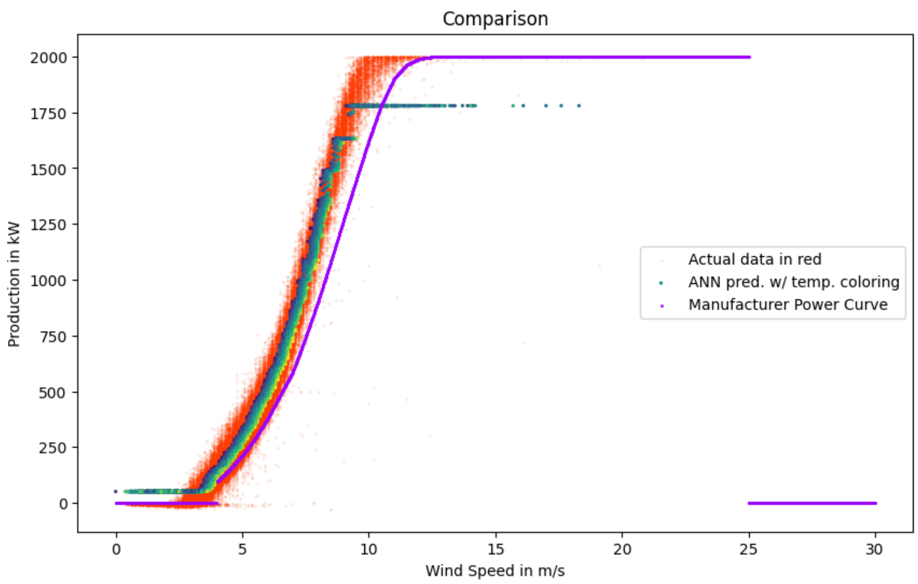
Results

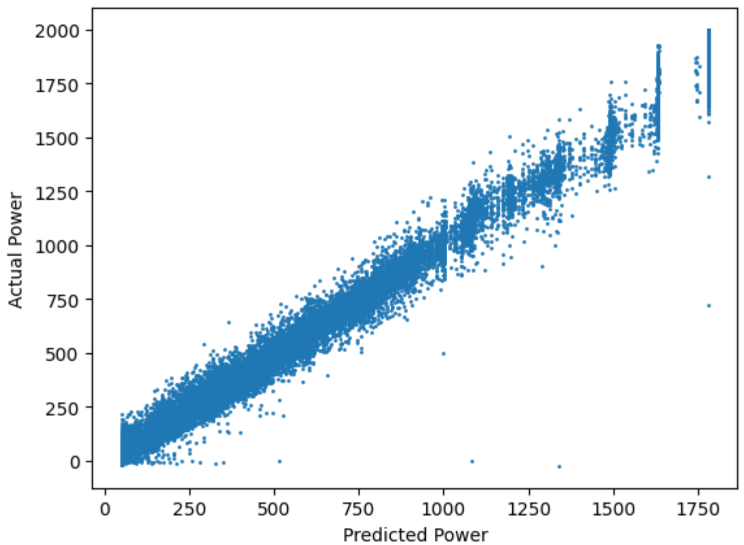


- ANN could learn power curve and outperformed MPV in terms of MSE:

```
Test Loss Neural Network MSE: 3781.520751953125
Manufacturer Prediction MSE:  26595.038436376733
```

- However, the upper wind speeds were modelled poorly on first sight:





Discussion

- This project serves as proof of concept of using neural networks as replacement of the MPC
- ANNs may replace manufacturer power curves all together
- other regression methods may be tested too
- Balancing data in terms of wind speeds could improve performance for higher wind speeds (as these are modelled poorly)
- Modelling power curves may be done for multiple systems of the same type in a tranfer learning fashion
- Looking at the difference between actual production data and ANN predictions may be informative on abnormal operational behaviour
- More architectures need to be tested to achieve optimized training performance
- Nevergrad may be used for other parameters, e.g. batch size