1. A pdf or .docx named “dl\_problem\_your\_name” file containing briefing about your methodology, your analysis of the outputs, training and test loss graphs, and any figures you see relevant.

“dl\_problem\_your\_name” report file is an important part of the grading,

make sure to reflect on your choices and analysis briefly.

The document should not be more than 5 pages by any means, one page could be good enough.

Think of it as a document to communicate your implementation (without the code), and why it works, to a fellow deep learning engineer.

#### Evaluation Criteria:

1. **Evaluation Metric** (similar to eval.py but not identical to it).
2. **Report: Briefness & Clarity –** Can we make the same conclusions you made, fast ?
3. **Problem formulation** (make sure you explain it well in the report file, including tokenization, loss function, architecture ..etc)
4. **Code readability:** This is necessary for us to be able to evaluate your code. Do not submit all your code in one notebook. You are a software engineer, structure your code in folders, files, and classes when needed.
5. **Code Correctness:** Your logic should be sane, assumptions you make should be explained in the report.
6. **Results readability:** Can we visually and logically validate that your results are ok ? Provide some output examples, maybe also provide examples where your model failed and your reflection on them.
7. **Percentage of original code**: While not essential, it is important to be able to see your coding quality and skills.
8. **Bonus:** Deep Learning and coding best practices: shuffling data, test data, manual seed to allow for experiment replication, type hinting ..etc.

[make sure to reflect on your choices and analysis briefly]

**Report: Briefness & Clarity –** Can we make the same conclusions you made, fast ?

**Code Correctness:** Your logic should be sane, assumptions you make should be

explained in the report.

**Results readability:** Can we visually and logically validate that your results are ok ? Provide some output examples, maybe also provide examples where your model failed and your reflection on them.

Architecture

The model is a series of linear layers composed of different nodes to eventually output

8 digits (batch\_size, output\_size) and combine them into real date as the input is shape

of (batch\_size, 10 - features).

Each layer followed by a batch normalization, dropout and finally activation function.

(batch\_size, 10) => 32 node => 128 node => 512 node => 64 node => 32 node

=> 8 node => (batch\_size, 8)

The model is trained with nn.L1Loss function as it gives higher loss magnitude and

penalty than nn.MSELoss function since we need to be accurate as possible. Here

we’ve used Adam optimizer along with momentum, adaptive learning rate and

L2 regularization.

Input features: ['day\_scaled', 'day\_sin', 'day\_cos', 'month\_scaled', 'month\_sin',

'month\_cos', 'leap\_year', 'decade\_scaled', 'decade\_sin', 'decade\_cos'].

Target: [‘date\_int’].

EDA and tokenization:

Using pandas to read and parse the data, firstly trying to infer date and then into numpy

array 2d shape as target. Going to parse input conditions into a side pandas dataframe

then encode into number, scale them with scikit learn scaler and calculate sine and

cosine on them (since as a pair it’s hard to repeat ending as unique pairs).

Loss

training and test loss graphs

Accuracy

your analysis of the outputs