1 Model documentation and write-up

1. **Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.**

   I'm a master student in Skoltech and Higher School of Economics universities in Russia, Moscow. My major is Data Science. I'll be graduated in June 2019 and my thesis is about computer vision methods in defectoscopy.

2. **High level summary of your approach: what did you do and why?**

   I use LSTM for processing the data of the time series. LSTM is known as a good feature extractor for sequential data. And it can be applied to the sequences with different lengths.

   Then I take the output of LSTM together with the process' metadata and send them in a small neural network with 4 dense layers.

   The last layer's output is a single number because we have a regression task here.

   I optimize directly the competition metric.

   In order to prevent overfitting I augment the initial data to enlarge the training dataset.

   Finally, to get my best result I use test time augmentation.

   This approach allows us to have a single model instead of a separate model for each process. Because we can process time series with different lengths. And we don't create any SARIMA-like model to make a predict for each process separately.

3. **Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.**

   - This code does train and test time augmentation.

     Here the list of features is enlarged with our time series crops (sequence, sequence_1, sequence_2, sequence_3 are crops of time series up to the ends of different stages) while metadata and targets for these crops are the same.

     The second augmentation is the addition of Gaussian noise to different crops of our time series.

     Augmentations allow us to train relatively deep neural networks and have more stable results. Different crops allow us to make predictions using only the part of the full time series as required by the structure of the test set.

```
if self.name == 'train' or \
(self.name == 'test' and self.evaluate_on_train) or \
(self.name == 'test' and self.config['tta']):

    self.augment_with_part_crops(features, names, process_id,
    sequence_1, sequence_2, sequence_3,
    meta, metas, target, targets)
    self.augment_with_gaussian_noise(features, names, process_id,
    sequence, sequence_1, sequence_2, sequence_3,
    meta, metas, target, targets)
```

   - This is the forward pass for my model. Here I take the last hidden state of the LSTM and concatenate it with the metadata for the process. Then all together propagate through the linear layers. Here I use the variable input_lengths in order combine time series of different lengths in one batch. This allows me to train my model much faster.

```
def forward(self, input):
    input, input_lengths, meta = input

    lstm_output_2, _ = self.lstm_2(input.float())

    input_lengths = input_lengths.to(device=self.config['device']) - 1
    lstm_output_last_2 = lstm_output_2[range(input_lengths.shape[0]), input_lengths, :]

    lstm_output_last = torch.cat((lstm_output_last_2, meta.squeeze(dim=1)), dim=1)

    output = self.linear_1(lstm_output_last)
    output = self.linear_2(output)
    output = self.linear_3(output)
    output = self.linear_4(output)

    output = output.squeeze(dim=1)

    return output
```

   - This is the loss function for my model. I optimize the competition metric directly because this gives me better results comaparing with the standard regression loss functions. I use Adam optimizer.

```
def mape_loss(y_input, y_target, config):
    threshold = 290000.0 * 0.000001
    threshold = threshold * torch.ones(y_target.shape).to(device=torch.device(config['device']))

    for_max = torch.cat((torch.abs(y_target).unsqueeze(0), threshold.unsqueeze(0)), dim=0)
    max_tensor, _ = torch.max(for_max, dim=0)
    loss = (torch.abs(y_target - y_input) / max_tensor).mean()
    return loss
```

4. **What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?**

   Things didn't work:

   - I tried additional augmentation of the training data with crops of time series starting from the beginning with random lengths.
   - I tried the averaging of the results of different models with approximately the same level of quality.
   - I tried to create a second level model on the top of the outputs of the models with approximately the same level of quality.
   - I tried more complex architectures with GRUs or combination of LSTM and GRU, different number of layers, different dimensions, even bidirectional RNNs, tried to pass not only the last hidden state of the RNN further but last two hidden states
   - I tried to remove boolean time series features.
   - I tried optimization of MSE and Huber losses.
   - I tried a lot of different learning rates, optimizers, learning rates schedules, batch sizes, weight decays, a total number of epochs.
   - I tried different approaches to the data normalization.
   - I tried using differences instead of the original time series.

5. **Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?**

   No, all tools (libraries, packages) I've used are in my code.

6. **How did you evaluate the performance of the model other than the provided metric, if at all?**

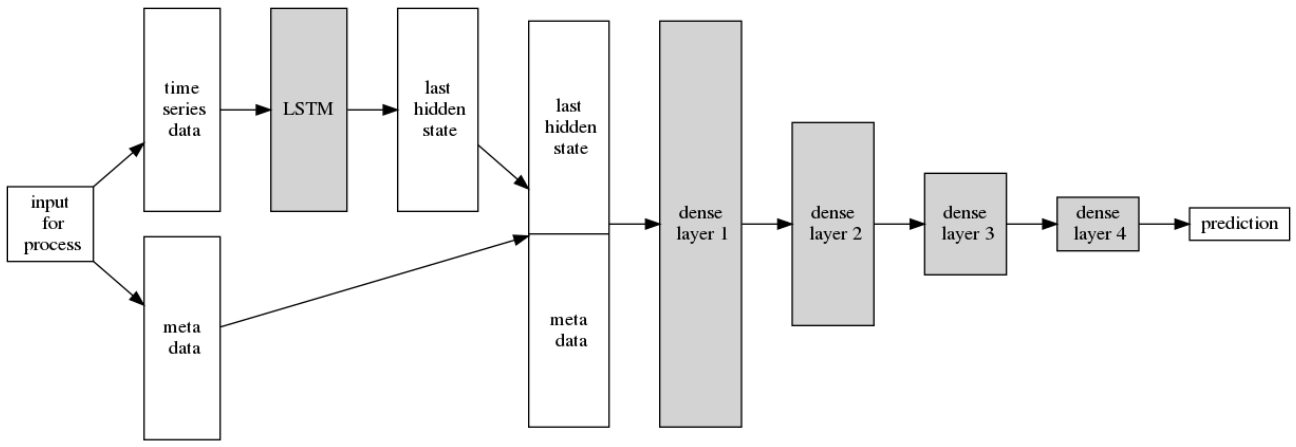
   I've tried to optimize MSE and Huber loss functions, but direct optimization of the competition metric is the best.

7. **Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?**

   My laptop configuration is 32 Gb of CPU memory and 6 Gb of GPU memory. As described in README.md file my code can be executed on CPU or on GPU (you should just put the correspondent parameter in the configuration file).

	Run on GPU much faster	Run on CPU
Inference only, batch size = 1	0.5 Gb GPU memory 5 Gb CPU memory	3 Gb CPU memory
Training with 5 folds, batch size = 64	6Gb GPU memory 8 Gb CPU memory	7.5 Gb CPU memory

8. **Do you have any useful charts, graphs, or visualizations from the process?**



9. **If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?**

   I would try to train separate models for only pre-rinse phase data, for the data up to the caustic phase, for the data up to the intermediate phase, for the data up to the acid phase and apply the correspondent model to each part of the test set. This will allow me to analyze how many phases we actually need to predict the final turbidity good enough.

   I would try to use more aggregated features from time series with an analysis of feature importance. Because I didn't try all of them during the competition.

   I would try to use skip connection between dense layers.

   I think that the type of product which was initially in the object can help. For example rinse process for milk or juice can be very different because milk has more fat and it's harder to clean the object. This product type can be given as a categorical variable for each process.