Model documentation and write-up

1. Who are you (mini-bio) and what do you do professionally?

I'm Machine Learning Researcher specialised in Computer Vision and Deep Learning. I very much enjoy facing new challenges, so I spend part of my spare time participating in Machine Learning and Data Science competitions.

2. What motivated you to compete in this challenge?

The topic and the dataset.

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

My approach is based on two layer pipeline, where the first layer are 14 CNN based regression models and the second one a GBM model adding extra features to the first layer predictions and past predictions.

First layer models.
The first layer is trained over a stratified 4 k-folds split scheme.

- A1: *CNN(vgg11) + RNN(gru); SeqLen=8*
Base CNN model: VGG-11
Sequence model: GRU
Instance input: RGB image made by copying raw 1-channel image 3 times.
Sequence input: 8 RGB consecutive images:
[3x img_{idx}], [3x Img_{idx-1}], …, [3x img_{idx-7}]

- A2: *CNN(vgg11) + RNN(gru); SeqLen=12*
Base CNN model: VGG-11
Sequence model: GRU
Instance input: RGB image made by copying raw 1-channel image 3 times.
Sequence input: 12 RGB consecutive images:
[3x img_{idx}], [3x Img_{idx-1}], …, [3x img_{idx-7}]

- A3: *CNN(vgg11) + RNN(gru); SeqLen=24*
Base CNN model: VGG-11
Sequence model: GRU
Instance input: RGB image made by concatenating 3 consecutive 1-ch images.
Sequence input: 8 RGB consecutive 3-concatenated 1-ch images:
[img_{idx}, $img_{idx-1,}$ img_{idx-2}], [img_{idx-3}, $img_{idx-4,}$ img_{idx-5}], …, [img_{idx-21}, $img_{idx-22,}$ img_{idx-23}]

- A4: *CNN(vgg11) + Transformer Encoder; SeqLen=24; Gap=0.5 h*
Base CNN model: VGG-11
Sequence model: Transformer encoder
Instance input: RGB image made by concatenating 3 1-ch images spaced 0.5 hours.
Sequence input: 8 RGB spaced 1.5 hours of 3-concatenated 1-ch images:
[img_{t}, $img_{t-0.5,}$ $img_{t-1.0}$], [$img_{t-1.5}$, $img_{t-2.0,}$ $img_{t-2.5}$], …, [$img_{t-10.5}$, $img_{t-11.0,}$ $img_{t-11.5}$]

Sample of concatenating 3 consecutive images (note the colours)

- A5: *CNN(vgg11) + Transformer Encoder; SeqLen=24; Gap=1.0 h*
Base CNN model: VGG-11
Sequence model: GRU
Instance input: RGB image made by concatenating 3 1-ch images spaced 1.0 hours.
Sequence input: 8 RGB spaced 3.0 hours of 3-concatenated 1-ch images:
[img_{t}, $img_{t-1.0,}$ $img_{t-2.0}$], [$img_{t-3.0}$, $img_{t-4.0,}$ $img_{t-5.0}$], …, [$img_{t-21.0}$, $img_{t-22.0,}$ $img_{t-23.0}$]

- A6: *CNN(vgg11) + RNN(gru); SeqLen=24; Gap=1.0; predict ALL*
Base CNN model: VGG-11
Sequence model: Transformer encoder
Instance input: RGB image made by concatenating 3 1-ch images spaced 1.0 hours.
Sequence input: 8 RGB spaced 3.0 hours of 3-concatenated 1-ch images:
[img_{t}, $img_{t-1.0,}$ $img_{t-2.0}$], [$img_{t-3.0}$, $img_{t-4.0,}$ $img_{t-5.0}$], …, [$img_{t-21.0}$, $img_{t-22.0,}$ $img_{t-23.0}$]
Output: In training time, predict all the outputs of the sequence.

- A7: *CNN(vgg11) + Transformer Encoder; SeqLen=24; Gap=1.0 h; predict ALL*
Base CNN model: VGG-11
Sequence model: GRU
Instance input: RGB image made by concatenating 3 1-ch images spaced 1.0 hours.
Sequence input: 8 RGB spaced 3.0 hours of 3-concatenated 1-ch images:
[img_{t}, $img_{t-1.0,}$ $img_{t-2.0}$], [$img_{t-3.0}$, $img_{t-4.0,}$ $img_{t-5.0}$], …, [$img_{t-21.0}$, $img_{t-22.0,}$ $img_{t-23.0}$]
Output: In training time, predict all the outputs of the sequence.

- B1: *CNN(*resnext101_64x4d*)*
Base CNN model: resnext101_64x4d
Instance input: RGB image made by copying raw 1-channel image 3 times:
[3x img_{idx}]

- B2: *CNN(*se_resnet152*)*
Base CNN model: se_resnet152
Instance input: RGB image made by copying raw 1-channel image 3 times:
[3x img_{idx}]

- B3: *CNN(*se_resnext101_32x4d*)*
Base CNN model: se_resnext101_32x4d
Instance input: RGB image made by copying raw 1-channel image 3 times:
[3x img_{idx}]

- C1: *CNN(*resnext101_64x4d); Channels=3, Gap=1.0 h; SmoothL1 beta=4
Base CNN model: resnext101_64x4d
Instance input: RGB image made by concatenating 3 1-ch images spaced 1.0 hours:
[img_{t}, $img_{t-1.0,}$ $img_{t-2.0}$]
Loss function: smoothL1(beta=4) instead of RMSELoss

- C2: *CNN(*resnext101_64x4d); Channels=3, Gap=1.0 h; Diff. Aug.
Base CNN model: resnext101_64x4d
Instance input: RGB image made by concatenating 3 1-ch images spaced 1.0 hours:
[img_t, $img_{t-1.0,}$ $img_{t-2.0}$]
Augmentation: RandomRotation + Resize(224) instead of
RandomRotation + CenterCrop(366) + Resize(224)

- C3: *CNN(*resnext101_64x4d); Channels=7, Gap=1.0 h; Diff. Aug.
Base CNN model: resnext101_64x4d
Instance input: RGB image made by concatenating 7 1-ch images spaced 1.0 hours:
[img_t, $img_{t-1.0,}$ $img_{t-2.0,}$ $img_{t-3.0,}$ $img_{t-4.0,}$ $img_{t-5.0,}$ $img_{t-6.0}$]
Augmentation: RandomCircleMask

- D1: *CNN(*vgg16); Channels=7, Gap=1.0 h; Diff. Aug.
Base CNN model: VGG-16
Instance input: RGB image made by representing Optical Flow vectors from a
sequence of 5 images spaced 1.0 hours.

The next table shows cross validation scores for individual models:

	Desc.	**CV RMSE**
A1	CNN(vgg11) + RNN(gru); SeqLen=8	7.994
A2	CNN(vgg11) + RNN(gru); SeqLen=12	7.726
A3	CNN(vgg11) + RNN(gru); SeqLen=24	7.441
A4	CNN(vgg11) + TRANSFORMERS; SeqLen=24, Gap=0.5 h	7.612
A5	CNN(vgg11) + TRANSFORMERS; SeqLen=24, Gap=1.0 h	7.516
A6	**CNN(vgg11) + RNN(gru); SeqLen=24, Gap=1.0; Predict ALL**	**7.231**
A7	CNN(vgg11) + TRANSFORMERS; SeqLen=24, Gap=1.0 h; Predict ALL	7.408
B1	CNN(resnext101_64x4d)	8.878
B2	CNN(se_resnet152)	9.162
B3	CNN(se_resnext101_32x4d)	9.072
C1	CNN(se_resnext101_32x4d); Channels=3, Gap=1.0 h; SmoothL1 beta=4	8.482
C2	CNN(se_resnext101_32x4d); Channels=3, Gap=1.0 h; Diff. Aug.	8.476
C3	CNN(se_resnext101_32x4d); Channels=7, Gap=1.0 h; Diff. Aug.	8.700
D1	CNN(vgg16); OpticaFlow vectors; Images=5, Gap=0.5 h	12.260

Second level model - Lightgbm

This model is used to predict wind speed based on 1st level models predictions and other features:

- 1st level predictions at time *t*
- 1st level predictions at time *t-1, t-2, t-4, t-6, t-8, t-10, t-12, t-16, t-20, t-24*
- Extra features: *ocean*, *storm_duration*

At prediction time, the model substitute all past predictions for actual wind speed from the training data, taking advantage of the known past velocities.

The final solution performance is (LocalCV|PublicLB|PrivateLB):

6.4961 | 6.8392 | 6.4561

The next table shows cross validation scores for 2nd level model depending on the number of 1st level models used:

Nb of models	Model ID	Desc.	CV RMSE	%
1	**A6**	**CNN(vgg11) + RNN(gru); SeqLen=24, Gap=1.0; Predict ALL**	7.0394	92.69%
2	**A2**	**CNN(vgg11) + RNN(gru); SeqLen=12**	6.8175	95.71%
3	**B1**	**CNN(resnext101_64x4d)**	6.7065	97.29%
4	**A4**	**CNN(vgg11) + TRANSFORMERS; SeqLen=24, Gap=0.5 h**	**6.6537**	**98.06%**
5	C1	CNN(se_resnext101_32x4d); Channels=3, Gap=1.0 h; SmoothL1 beta=4	6.6149	98.64%
6	A5	CNN(vgg11) + TRANSFORMERS; SeqLen=24, Gap=1.0 h	6.5850	99.08%
7	B2	CNN(se_resnet152)	6.5707	99.30%
8	A1	CNN(vgg11) + RNN(gru); SeqLen=8	6.5602	99.46%
9	C3	CNN(se_resnext101_32x4d); Channels=7, Gap=1.0 h; Diff. Aug.	6.5500	99.61%
10	B3	CNN(se_resnext101_32x4d)	6.5399	99.77%
11	A7	CNN(vgg11) + TRANSFORMERS; SeqLen=24, Gap=1.0 h; Predict ALL	6.5360	99.83%
12	A3	CNN(vgg11) + RNN(gru); SeqLen=24	6.5296	99.92%
13	C2	CNN(se_resnext101_32x4d); Channels=3, Gap=1.0 h; Diff. Aug.	6.5248	100.00%
14	D1	CNN(vgg16); OpticaFlow vectors; Images=5, Gap=0.5 h	6.5247 (*)	100.00%

(*) Final model model score is 6.4961 after averaging 4 of the model's predictions. See code for more details.

Using just 4 models we can achieve 98% of the score when using all 14 models.

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

4.1 Parse blocks of images

Many consecutive images for the same storm has the same or very similar target or wind speed. To accelerate training, instead of feeding the model with all the images, I sampled random images within a “block”.
A “block” is a group of consecutive images for the same storm that the range between the minimum wind speed and the maximum is less or equal to a specific value (5 for the models I trained).
This “trick” reduce dramatically the training times while performance is almost as good as using all the images per epoch. I think that as the validation (and test) set contains some storms that are also in training, some over fitting to the training set is beneficial.

```
  def _get_blocks(self):
    """
    Function that groups data for each storm in blocks of specific size.
    For each storm id, blocks are built as follow:
      - Create first block with id=0 and add first image.
      - Next image will belong to the same block if the difference between max and
min wind_speed
      within the block would be less or equal to `max_block_size`. If not, new
block is created
      with id += 1.
      -
    Return a pandas DataFrameGroupBy object.
    """
    result = []
    for storm_id, gr in self.df[self.df.train].groupby('storm_id'):
      tmp_df = gr.sort_values('relative_time', ascending=True)
      block_id, block_min, block_max = 0, None, None
      for image_idx, image_id, wind_speed in zip(tmp_df.index, tmp_df.image_id,
tmp_df.wind_speed):
        if block_min is None or wind_speed < block_min:
          new_block_min = wind_speed
        if block_max is None or wind_speed > block_max:
          new_block_max = wind_speed
        new_block_size = new_block_max - new_block_min
        if new_block_size > self.max_block_size:
          block_id += 1
          new_block_min = wind_speed
          new_block_max = wind_speed
        block_min = new_block_min
        block_max = new_block_max
        result.append([image_idx, storm_id, image_id, wind_speed, block_id])
    return pd.DataFrame(result, columns=['image_idx', 'storm_id', 'image_id',
'wind_speed', 'block_id']). \
      groupby(['storm_id', 'block_id'])
```

4.2 Average some models in the 2nd level model

Usually I just concatenate predictions from 1st level models to build the 2nd level data set, but in this case, aggregating some of the models and averaging their predictions improved a little bit the final score from 6.524 to 6.496

```
MODELS_LIST = ['L1A_A2', 'L1A_B1', 'L1A_C1',
        'L1A_B2', 'L1A_A1', 'L1A_C3', 'L1A_B3',
        'L1A_A3', 'L1A_C2', 'L1A_D1']
MAIN_PRED_LIST = ['L1A_A6', 'L1A_A4', 'L1A_A5', 'L1A_A7', ]
```

4.3 Using actual wind velocities when they are known

This little trick assumes that in a real application we would have actual values of wind velocity for timestamps in the past, so we can substitute predicted values of wind velocity for actual ones, i.e. for timestamp t, all 1st level predictions for t-1, t-2, …, t-24 are substituted by the actual values. It improved a little bit the final score from 6.522 to 6.496

```
# Update validation L1 with actual known wind_speed values
VALID_train_L1_pred_df = train_L1_pred_df.copy()
for column in VALID_train_L1_pred_df.columns:
  VALID_train_L1_pred_df.loc[train_field_ids_i, column] = dset_df.loc[
    train_field_ids_i, 'wind_speed']
```

5. Please provide the machine specs and time you used to run your model.

- CPU (model): Intel Core i7-5960X
- GPU (model or N/A): Nivida RTX 2080Ti 11GB
- Memory (GB): 64 GB
- OS: Linux Mint 18.3 (Ubuntu 16.04 based)
- Train duration: 223 hours
- Inference duration: 6 hours

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

- Optical Flow: Even if it is in the final solution (D1), its contribution to the final model is minimal, if any. I would have removed this model.
- 3D-CNN models performed worst than LSTM and didn't contributed to the final ensemble.
- Trainig models from scratch didn't seem to improve the results, so all my models are fine tuned from Imagenet pre trained weights.

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

To come up with the cross validation splits, I explored the distribution of the data using Jupyter notebooks and matplotlib.

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

I expended some time splitting the data to do cross validation, trying to replicate the test set distribution (what I know about it).

Test set has new storms and storms that exist also in the training data but at later stage. When splitting the training set in 4 folds to perform cross validation, I did it in a way that each fold contains same proportion of “new” storms than the test set. For the remaining storms, in both validation set and training set for each fold, I split each of the storms in two, at a random point with same distribution than the test set, and validate with the later split. I also ensure that ocean, storm duration and wind velocity were evenly distributed.

Other than that, I only used RMSE.

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

Because of memory overload during training, I run one script per fold for training, but removing ‘--folds’ from the training command train all the folds in the same execution.

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

I already included some images and tables above.

11. 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 was a bit surprised that VGG11 performed that good. I would expend some time looking for better architecture for this specific domain.

Even if the centred images for the same storm are very useful for CNN models, I have the intuition that data about the displacement of the centre of the storm might have some correlation with the target. Maybe coordinates of the centre of the image would be enough.