University of Liège

Big Data - Review 4

Renewable Energy Production Forecast

Yann Claes
Gaspard Lambrechts
François Rozet

April 9, 2020

1 Photovoltaic production

This section will describe the improvements that were made to both current models and discuss their future use.

1.1 Panel-wise model

As mentioned in the previous review, the model built for the Sart-Tilman is now complete, and we don't plan on improving it even more since it was a startup model and the goal is to predict the energy production at the scale of the city/province of Liège.

For that purpose, we refreshed the latter model to make it more modular in order to be ready (to some extent) to receive as input photovoltaic enumeration data. That is, we moved the Sart-Tilman model onto a file named `sart-tilman.py` and implemented in the file `solar.py` a model that should be combined with our photovoltaic panel detector.

To combine our photovoltaic panel detector with our panel-wise model, we plan (hopefully) to use the outputs of the detector. It should yield, at least (for each detected panel/set of panels) :

- Its corresponding latitude
- Its corresponding longitude
- Its corresponding area

To get more reliable results, we observed on the Sart-Tilman that computing the incidence angle was quite a big deal, thus we would need to compute that angle. Nevertheless, this angle requires knowledge about the tilt and surface azimuth of the panel. The second one could be easily computed (at first sight); however the tilt of each panel seems quite hard to deduce.

With all these information, we should be able to compute the power output of each PV panel as follows :

$$P = \eta I A \frac{\cos(\theta)}{\sin(\psi)}$$

where η is the efficiency, I is the forecast irradiance, A is the panel area (output of the detector) and ψ is the solar altitude (function of the latitude and longitude of the panel).

One can notice that, in addition to the tilt and surface azimuth, another unknown is the panel efficiency. To deal with all these uncertainties, we could potentially try and use a probabilistic model on top of this physical model (using `PyStan` for instance), if it seems appropriate.

A sample execution was implemented in `solar.py`, with user-coded coordinates and parameters, showing the kind of inputs the program expects and plotting the corresponding energy production for the specified forecast period. For now, the irradiance data that is used comes from the CAMS radiation service, which doesn't provide forecast data, but rather historical data (up to two days ago). Later, a forecast API will be used, but we are still looking for a (free) reliable one.

1.2 Provincial model

This model is concerned with the prediction of the energy production at the scale of the province of Liège. As explained during the previous review, it uses radiation data (from the CAMS radiation service, providing global horizontal irradiance data at ground level) in Liège to produce an estimation of the provincial output power as follows :

$$P = \eta I A$$

where η corresponds to the efficiency, I the irradiance data and A is a rough measure of the PV area we have in the province of Liège. The latter is the product of the installed power in the province of Liège (kWPeak) and of some measure of average panel area per kWPeak in Belgium (m^2/kWPeak). Thus, all parameters are, in some sense, arbitrarily fixed in advance, which leads to the example results of Figure 1, for the 21st of February 2019.

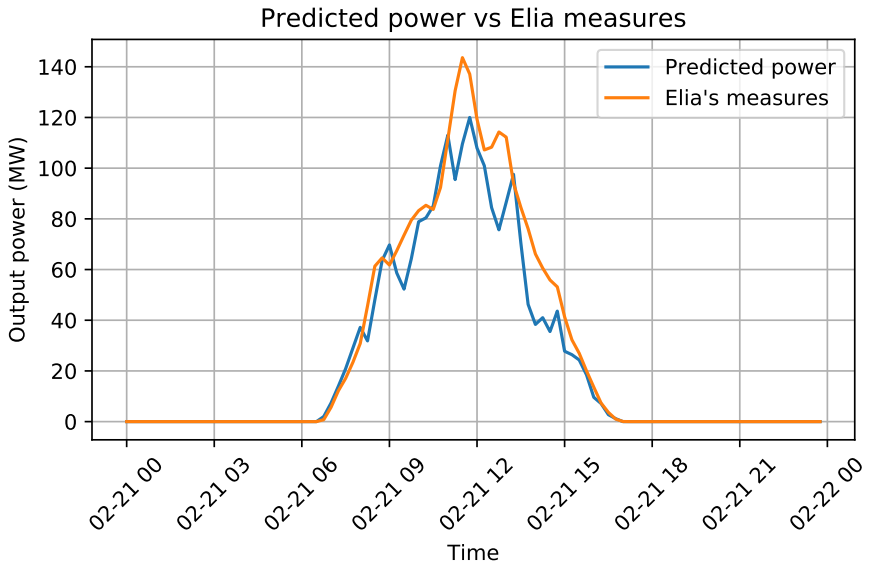

Figure 1 – Naive physical model example.

As one can see, the model isn't too bad, but the fact of setting the parameters to some fixed value is too restrictive. For that purpose, we implemented a probabilistic model in PyStan in order to define uncertainties on:

- η
- The installed power
- The peak area
- The irradiance

and, therefore, to define uncertainty on the resulting output model. At the same time, we fit our parameters to Elia's measures (they will in the end be represented by posterior distributions) and we also predict the output power for a given forecast period.

For example, we can fit our parameters on a period ranging from the 15th of February 2019 to the 23rd (excluded) and predict for the next day (*i.e.* 23rd of February).

We can now compare the forecast output power that we would get with our *fixed-parameterized* model with the one we get with PyStan. This comparison can be seen on Figure 2.

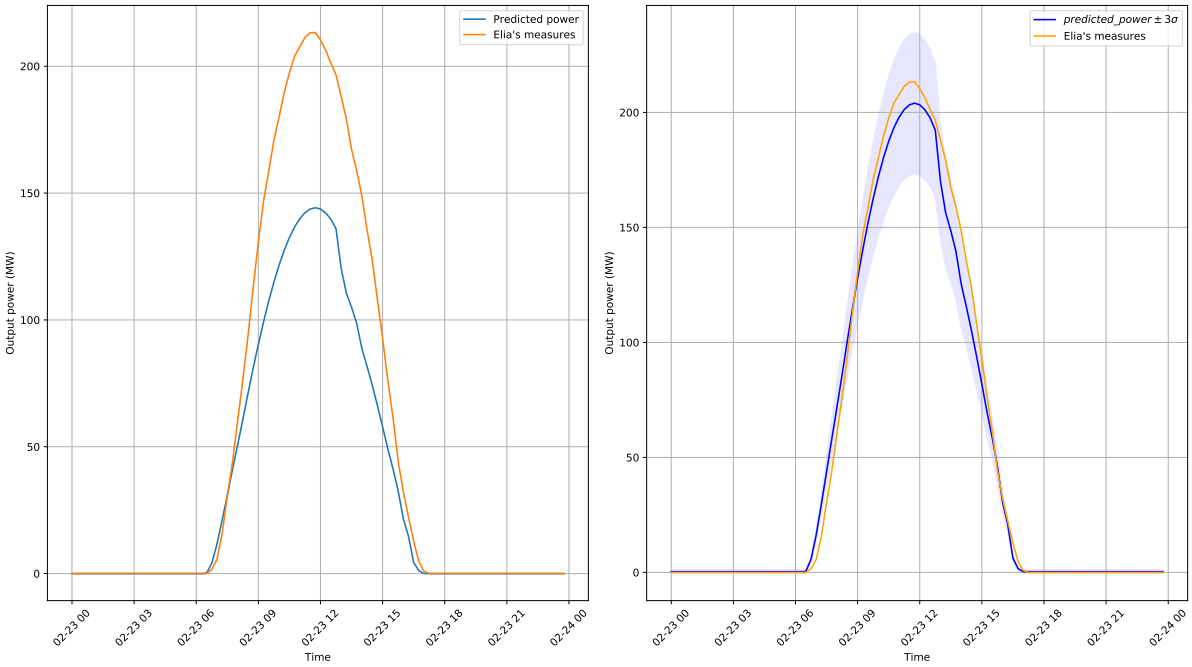

Figure 2 – Comparison between prior and posterior predictive models.

We can clearly see that the fitted model performs better at predicting the output power than the naive one. Note that the predictive model is defined as a normal distribution centered around $\mu = \eta I A$ with a standard deviation $\sigma = 100\,\text{kW}$ (where η, I and A are sampled by the model[1]).

To get more insight about this, we can compute the mean square error, as well as the root mean square error, which is a nice metric if we want to estimate the standard deviation of our residuals (and therefore, estimate the quality of the prediction) and which is also commonly used for assessing forecasts[1]. To have a more representative metric, we decided to not take into account night observations. We get the results of Table 1 (still for this sample example), using the mean posterior model as reference for our probabilistic model.

	MSE	RMSE
Naive	2211.522	47.027
Posterior	126.439	11.244

Table 1 – MSE and RMSE for the naive and posterior models.

Although these results are obviously not representative of all cases we could have (in terms of shape for Elia's measures), the main point is to notice that, by introducing a probabilistic model, we managed to reduce greatly the errors we make on our predictions.

[1]See `province.ipynb` for full details.

By contrast, we can look at the errors Elia makes on their prediction values. This can be seen on Figure 3.

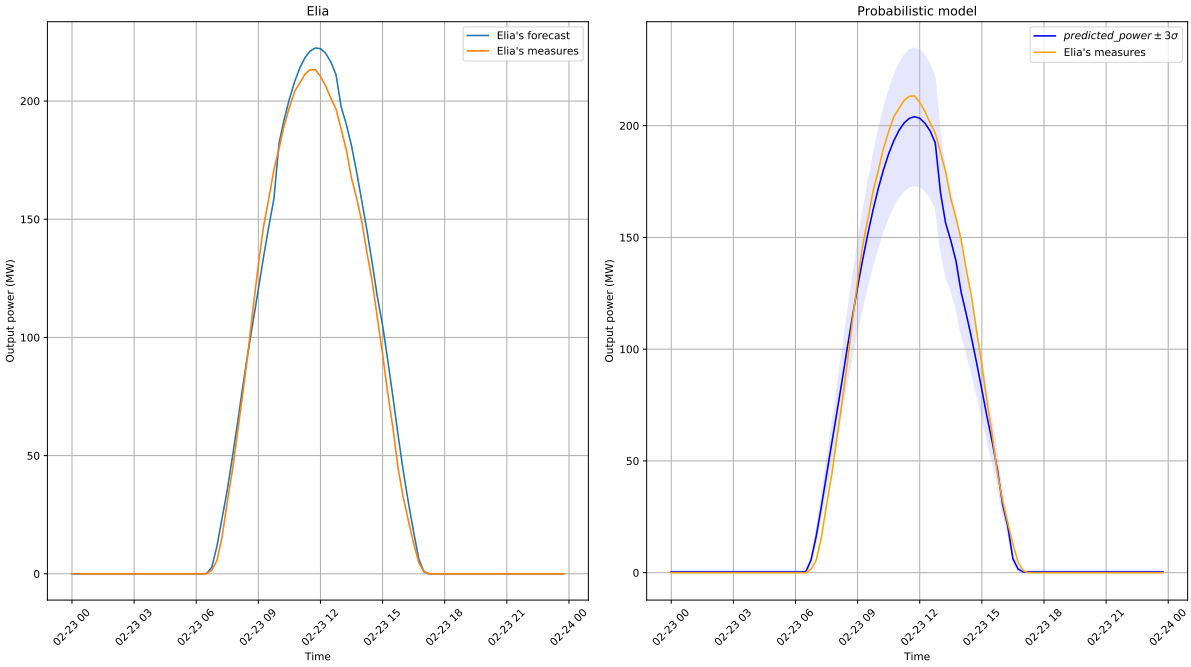

Figure 3 – Comparison between Elia's forecast and the posterior model.

As one could expect, it seems that Elia performs slightly better than our probabilistic model. This could be explained by the fact that Elia probably does not (or does not only) rely on a simple physical model as we do. This is also reflected in Table 2.

	MSE	RMSE
Elia	84.988	9.219
Posterior	126.439	11.244

Table 2 – MSE and RMSE for Elia's predictions and the posterior model.

Nevertheless, we shouldn't draw hasty conclusions. Indeed, doing the same experiment on the next month had the effect of pushing up the parameters too much (w.r.t. the naive predictions), leading to a higher overestimation of the power forecast, as can be seen on Figure 4, while Elia's forecast is still on point, as can be observed on Figure 5.

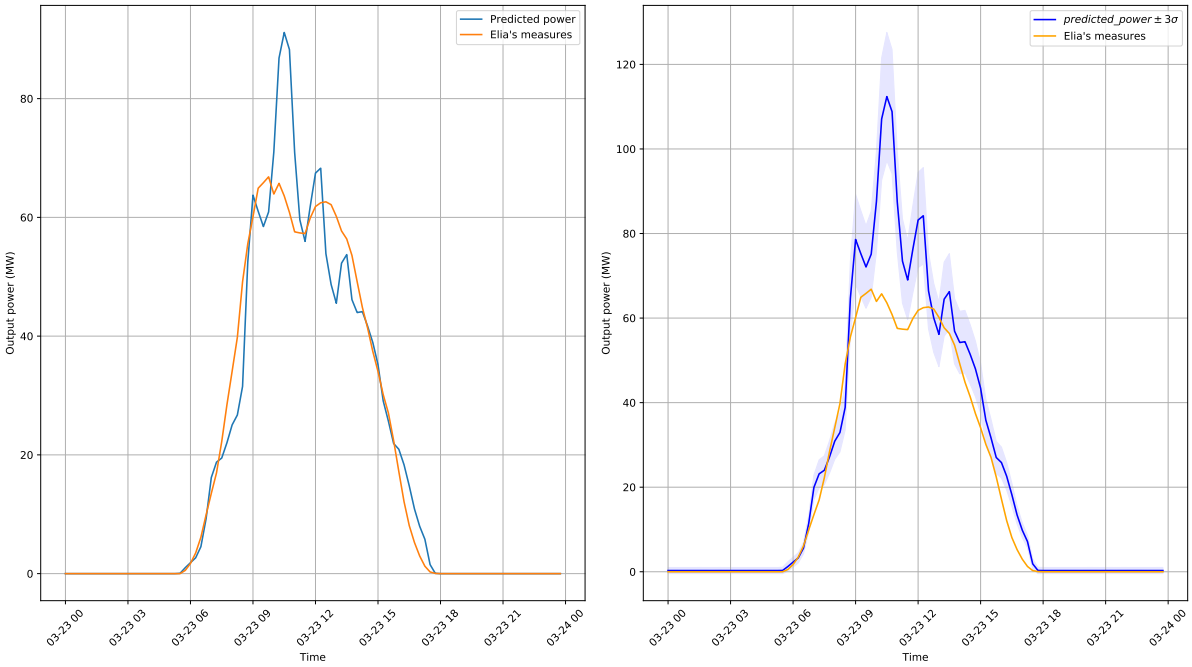

Figure 4 – Same example for March 2019 (naive vs posterior model).

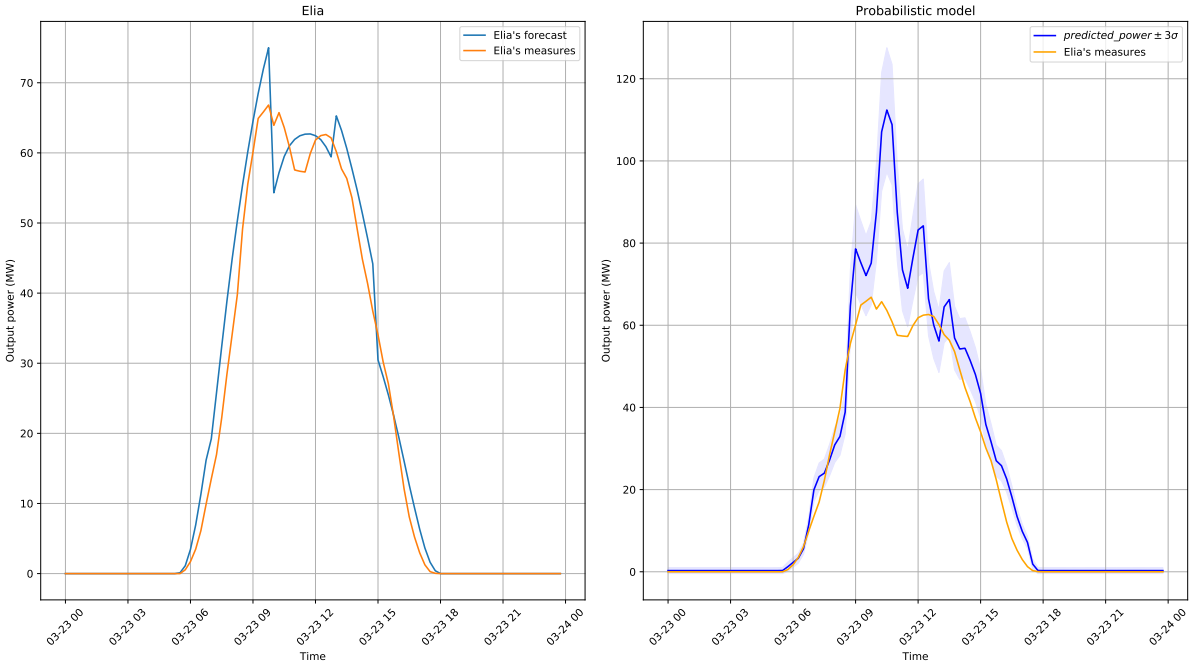

Figure 5 – Same example for March 2019 (Elia vs posterior model).

1.3 Irradiance data

For now, no free and reliable forecast API have been found to retrieve irradiance data in Liège. Indeed, the only one we have so far is Climacell, but as was already said, the measures they provide seem to be quite underestimating the true irradiance that is observed. We thus only use the data coming from the CAMS radiation service, that only provides historical data.

A potential alternative could be HelioClim-3 [2], but no free automatic access can be obtained, and once again, there seems to be unreliable measurements: they tend to highly overestimate the true irradiance (since they are expressed in $\mathrm{W\,h\,m^{-2}}$, a conversion is needed to get $\mathrm{W\,m^{-2}}$).

1.4 Next objectives

In addition to trying to find a reliable irradiance forecast API, it would be nice to find an API that combines both historical data as well as forecast data. Indeed, since the irradiance plays a crucial role in the produced power in our models, fitting parameters on some irradiance data coming from source 1 and predicting the output power using irradiance data coming from source 2 will definitely be more biased than if we use irradiance data coming from the same source.

What remains to be done to complete the analysis of such a posterior model is to compare the latter with simple baseline models (moving average of the output power over the last D days, etc.) as well as to evaluate the posterior model over a representative period (and eventually compare with what Elia obtains over such a period).

2 Photovoltaic panels enumeration

As a quick reminder, we have already obtained a suitable dataset for training : a large collection of *publicly available* [link] high resolution aerial imagery, with over 19 000 human annotated PV panels locations.

Our goal is now to design and train a neural network able to *detect* photovoltaic (or solar) panels in satellite images, hence the name *Automatic Detection Of Photovoltaic Panels Through Remote Sensing* or ADOPPTRS[2].

Note. What we discovered recently is that this collection was released with the paper "Automatic Detection of Solar Photovoltaic Arrays in High Resolution Aerial Imagery" [3] from Duke University from which our sub-project name is inspired. Interestingly, this paper didn't use neural networks for detection but Random Forests instead.

2.1 DeepSolar

The first model we looked after was the one used and designed for the DeepSolar [4] project conducted at Stanford University as it is one of the most reviewed. DeepSolar's goal is very similar to ours, *i.e.* detecting all PV panels in the United States using satellite imagery. However, the model used by DeepSolar is fairly complicated. It incorporates both image classification (Google Inception V3 [5]) and semantic segmentation in a single convolutional neural network.

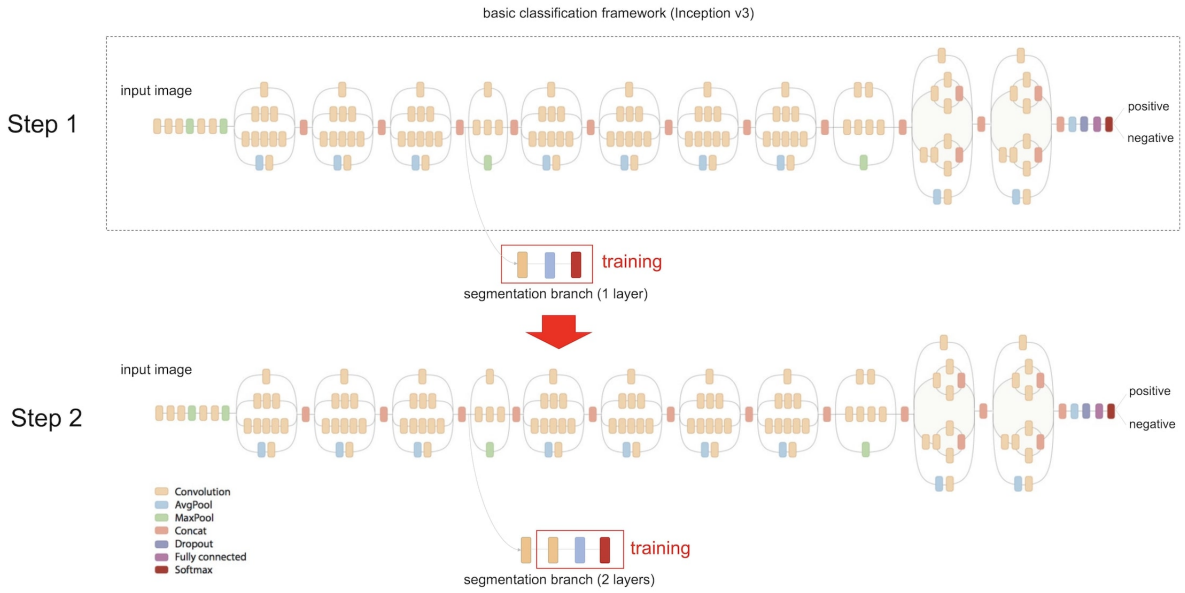

Figure 6 – DeepSolar's network representation. [4]

The classification branch is used to localize the solar panels and the segmentation one to estimate their sizes. Only if an image is classified as *positive* (containing a PV panel), the segmentation branch is executed. The advantage of this method is that the segmentation

[2]This part of the project is conducted in parallel with our project of *Deep Learning*. Therefore, we have created another GitHub repository especially for this purpose : `https://github.com/francois-rozet/adopptrs`.

does *not* need another forward pass. Unfortunately, we believe that reproducing such network is not within our capabilities, at least for now.

Furthermore, the reason why DeepSolar was first built as a classification network was because the researcher(s) didn't have *segmentation* masks as training set. In fact they only had images with *labels* indicating the presence or absence of panels. Therefore they had to solve a problem of *semi-supervised* segmentation.

Conversely, we have segmentation masks (cf. previous review) and therefore we can train directly a segmentation model.

2.2 U-Net

One of the most famous segmentation model is U-Net [6]. Initially designed for biomedical image segmentation, it actually works very well for a lot of applications.

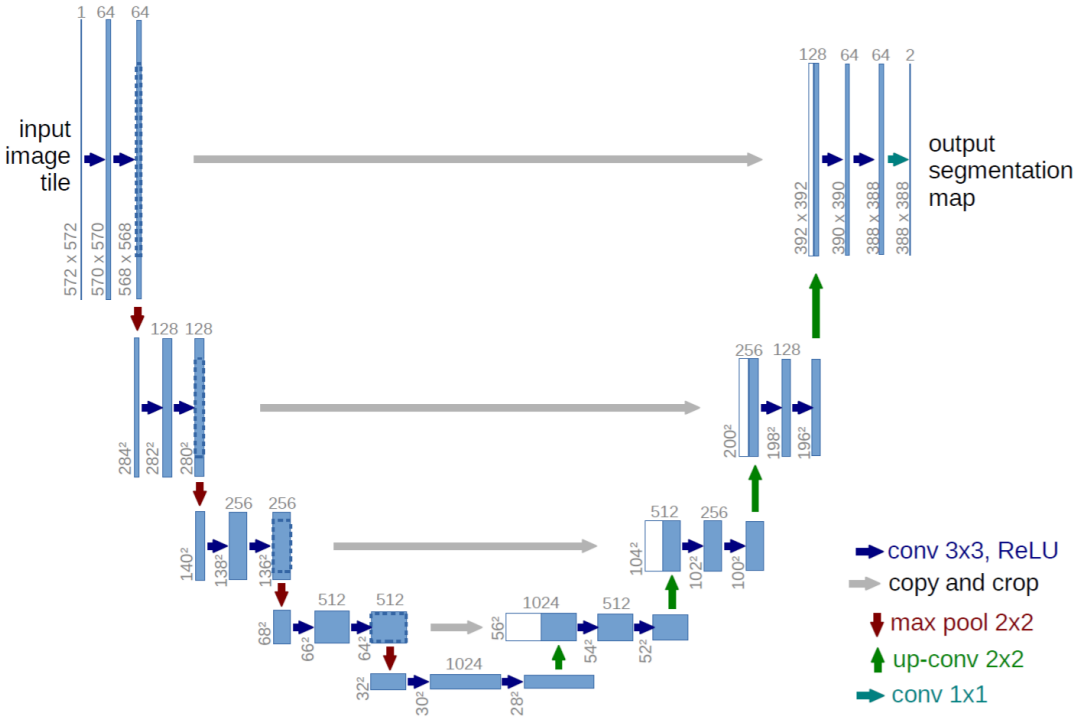

Figure 7 – U-net architecture. [6]

We have implemented and trained this model using `PyTorch`. We first tried with a shortened version of U-Net but it didn't worked very well. We are now using the full architecture.

2.3 Training

We divided our dataset in 3 subsets : a training set (75 %), a validation set (12.5 %) and a testing set (12.5 %). The first one will be used for training our models, the second one for selecting the best one (and train it again) and the third one will be used to evaluate our final model (at the very end of our journey).

2.3.1 Loss function

One of the most used loss function for segmentation (and classification) is the *cross entropy loss* function.

$$CrossEntropy(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{c=1}^{C} y_c \log \hat{y}_c$$

where $\hat{\boldsymbol{y}} \in [0,1]^C$ is the model probability distribution prediction among the C classes and $y \in \{0,1\}^C$ is the ground truth. For a binary classification problem as we face, this reduces to

$$BinaryCrossEntropy(\hat{y}, y) = -y \log \hat{y} - (1-y) \log(1-\hat{y})$$

We tried using this function (implemented in `PyTorch` as `BCELoss`) but the results were quite poor. Indeed, most of our predictions were simply pitch black. We think this is due to the huge class imbalance[3] in the dataset.

We therefore searched for a "loss function for imbalanced segmentation" and a lot of forums and websites mentioned the *dice coefficient* and *intersection over union*. Conversely to the cross entropy, these metrics are not applied pixel-wise but image-wise (or batch-wise).

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad Dice(A,B) = \frac{2\,|A \cap B|}{|A| + |B|}$$

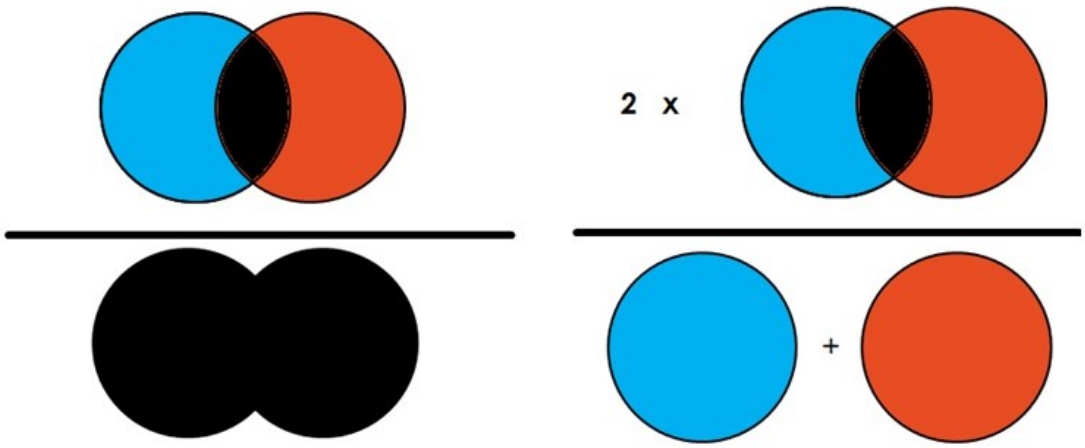

Figure 8 – Illustration of Intersection over Union (left) and Dice Coefficient (right). [7]

Both metrics are very similar but the Dice Coefficient is actually a little bit easier to implement which is why we chose to continue with it.

Note. To transform these metrics into loss function we have to consider $1 - Dice$ and $1 - IoU$.

2.4 Data augmentation

As we have mentioned in the previous review, we wished to increase the size of our dataset trough data augmentation. Our approach is to randomly apply some transformation(s)

[3] We haven't tried class weighting yet.

to the training set *while training*. Therefore, at each epoch, the training set is slightly different and becomes much harder to overfit on.

The implemented transformations are :

- Rotations : 90°, 180°, 270°
- Flipping horizontally or vertically
- Brightness alteration
- Saturation alteration
- Contrast alteration

Figure 9 – Example of some image transformations. The first column is the original.

2.5 Results

For the moment, our best model (obtained after 32 epochs in 16 hours[4]) achieves an average Dice Loss of 26.5 % on the validation set (and 16.91 % on the training set). Pay attention that it is *not* a measure of accuracy ! What should be more informative is the *confusion matrix* :

		Truth	
		0	1
Prediction	0	3.236×10^5	1.054×10^3
	1	3.096×10^2	2.690×10^3

Table 3 – Average confusion matrix on the validation set.

As one can see in Table 3, the majority of our predictions are (rightfully) negative, which emphasises the class imbalance we talked earlier. As a consequence our model reaches an (irrelevant) accuracy of 99.58 %. Instead, we should consider the *specificity*/*precision* and *sensitivity*/*recall* metrics :

$$precision = \frac{TP}{TP + FP} = 89.69\,\%$$
$$recall = \frac{TP}{TP + FN} = 71.92\,\%$$

This precision score means that our model rarely classifies something else as a photovoltaic panel but this recall score means that it often fails to recognize a photovoltaic panel. Yet for a first model this isn't terrible at all !

Some evaluation samples are provided in the appendix.

2.6 Next objectives

Even though our first model isn't that bad, we still want to improve it and to try a few other models before applying it to our initial problem.

Also, at each epoch, we evaluated our model on the validation set and saved a few statistical measures about the losses. We could try to analyse convergence through these measures.

Finally, there is still to do a few post-processing to convert the obtained predictions into photovoltaic panel locations.

[4]We actually did way more training phases than this, but we stumbled upon quite a few problems with the GPU clusters.

3 Wind production

3.1 Classification of wind forecasting problems

- Very-short-term or immediate-short-term (a few hour ahead)
- Short-term (few hours to several days)
- Long-term (multiple days ahead)

3.2 Classical approaches in Wind Power Prediction

From *Sweeny et. al, 2019* [8] and *Messner et. al, 2020* [9], we have identified the following approaches for wind power forecasting.

1 **Physical modeling**: based on Numerical Weather Prediction (NWP), the model aim at outputting the power produced by one power plant using the power plant characteristics (wind turbine power curves, surface roughness and obstacles, terrain wake, hub heights, and even the wind farms wake effect on themselves is taken into account in recent works). These approaches are said to be very computationally expensive, and quite hard to apply on very large scale situation as in our case where we have 64 power plants. However, what we showed in review 2 corresponds to the strategy for physical modelling described in the second paper, with a simpler model.

2 **Statistical modeling**

   a On the very short term: *time series based forecasting*: the NWP data is not used, only the power time series is used to make the prediction. This approach is said to be hard to beat on 4 to 6 hours ahead predictions (very short term).

   b On the longer term: *post processing of NPW*: the temporal aspect is not considered. The power predictions are based on NWP. Roughly, this approach aim at mimicking the physical model when some information are missing such as the surface roughness and terrain wake effect for each power plant.

3 **Hybrid method**: uses *post processing of NPW* **and** *of physical modeling.*

For now, the most promising and adapted approach seems too be the *post processing of NPW*, since:

- Our physical model was not good (but could be improved) and we have a big number of wind farms,
- We have a good amount of data,
- We address the one-day ahead forecasting task,
- The results of the simple statistical model of review 3 were promising.

However, the possible improvements on the physical model are considered below, as a hybrid method could perform even better than the statistical modeling.

3.3 Physical model

The physical model that we built for review 2 was constituted of a power curve for each wind turbine. It took as input the wind at each power plant location.

This model could be refined by using the *log wind profile* (semi-empirical relationship) to model the wind at hub height :

$$u(z_2) = u(z_1)\frac{\ln(z_2/z_0)}{\ln(z_1/z_0)}$$

where z_2 is the height of interest (hub height), z_1 is the height at which the wind speed is measured (typically 10 m/s), and z_0 is the roughness length that measures the roughness of a surface on the wind flow. This value ranges between 0.005 and 0.1 for onshore wind farm terrains. This parameter would have to be estimated.

It could also be refined by taking into account the wind direction and model the self wake effect due to alignment of wind and wind turbines (the exact location of each wind turbine in Wallonia is known). This would also imply several unknown parameters to estimate.

3.4 Statistical model - Weather to Power

We chose to name the statistical model a *weather to power* model (case 2.b in the classification of section 3.2) in order to distinguish from the time series modeling.

The statistical model built at the last review was a *wind to power* model where the forecast of wind was considered at a single location. This time, we have built a new training set that is constituted of the following inputs (NPW measured at 15 different locations) :

`windSpeed`	`windGust`	`windBearing`	`temperature`	`humidity`	`pressure`	`airDensity`
m/s	m/s	m/s	K	-	Pa	kg/m^3

This columns are thus repeated 15 times (one time at each location where the weather forecasts are made), they are thus 105 in total. The forecast locations have been chosen by hand, looking at the map of all wind turbines in Wallonia. The limitation of 15 locations is due to the fact that we use a free weather API. But later, we will certainly retrieve these 7 weather data-points at each power plant.

On the figure 10, we can see the locations of each power plant and in red the location of the weather measures.

The last weather variable `airDensity` has been computed from the other ones, as it has been considered useful as feature of the regression problem. The thermodynamics relations used are available in [10].

3.4.1 Metrics

Typical metric for wind power forecasting are:

- sRMSE: standardized Root Mean Squared Error.
- sMAE: standardized Mean Absolute Error.

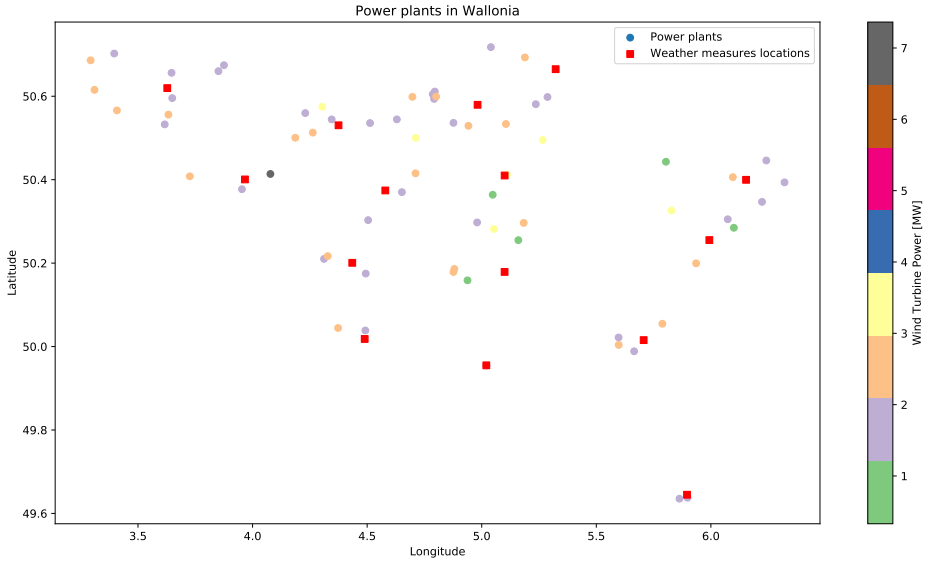

Figure 10 – Power Plants and weather measures in Wallonia

State-of-the-art in both metrics is around 5 % for the 12 hour ahead task, 8 to 10 % for the 24 hour ahead task, as can be verified in *Croonenbroack and Dahl, 2014* [11] and *Xiaochen Wang et. al, 2011* [12].

If the cost of an error is directly proportional to the error, it is justified to use the MAE to assess a model, that is why we choose to use sMAE as metric.

Furthermore, as we designed our model such that it is able to predict confidence intervals, we have to use a proper metric to assess these predicted intervals. We have chosen to use the following metric, defined in *Meinshausen, 2006* [13] :

$$L_\alpha(y, q) = \begin{cases} \alpha \left| y - q \right| & \text{if } y > q \\ (1 - \alpha) \left| y - q \right| & \text{if } y \leq q \end{cases}$$

where α denotes the percentile and q the predicted quantile.

3.4.2 Protocol

For now, two protocols have been used to choose and assess our models. Both protocols used the data between January 2019 and February 2020.

- Protocol 1: model selection and training on the 13 first months and testing on February 2020.
- Protocol 2: model selection and training on a subset randomly drawn from thirteen fourteenth of the data, testing on the rest.

3.4.3 Models and results

As the data has no temporal variable such that we do not have the need to extrapolate, we focused on tree-based methods as it seemed to perform better than other methods. The two considered models are

- Extra Trees regressor (tree bagging method)

- Gradient Boosting regressor (tree boosting method)

The hyperparameters of these models have been determined by cross-validation on the train set. The best obtained scores are listed in the table 4.

MAE	Extra Trees	Gradient Boosting
Protocol 1	36.35	36.93
Protocol 2	25.37	28.18

Table 4 – Train set CV scores

The results on the test set are listed in the table 5. Theses scores include the MAE and sMAE, together with the scores of the 10 % are 90 % quantiles using the loss defined earlier.

Protocol	Method	MAE	sMAE	MQL10	MQL90
Protocol 1	Extra Trees	42.38	6.00%	63.06	61.99
	Gradient Boosting	43.08	6.10%	58.91	43.96
Protocol 2	Extra Trees	28.13	4.03%	46.63	50.32
	Gradient Boosting	31.11	4.46%	48.32	47.91

Table 5 – Test set scores

These results are quite good, as is illustrated on figures 11 and 12 (protocol 1 only).

We can notice that the sMAE obtained are very good because they are based on the weather measurement instead of the day-ahead weather prediction. Once the model will be used to make forecast based on weather prediction, the error on the NWP will increase the current sMAE. If we can assume a MAE of 2.4 m/s (*El-Fouly et. al, 2008* [14]) for the wind measurement, the sMAE should be bounded by 15 % in the worst case, given the power curves derivatives. First results computed on only 1 week, with day-ahead NWP instead of measurement have shown a MAE of 68.19 for Extra Trees and 70.26 for Gradient Boosting.

Then, it can be noticed that the Extra Trees and Gradient Boosting compete in term of MAE, even if the Extra Trees are slightly better. However, by looking at the metrics MLQ10 and MLQ90, the Gradient Boosting method seems better on the quantile regression task. Furthermore, when we look at the figures 11 and 12, we can see that the quantiles built by the Gradient Boosting method are much tighter, which does not seem to be reflected in the quantile loss.

These observations encouraged us to prefer the Gradient Boosting method in our future forecasting refinements.

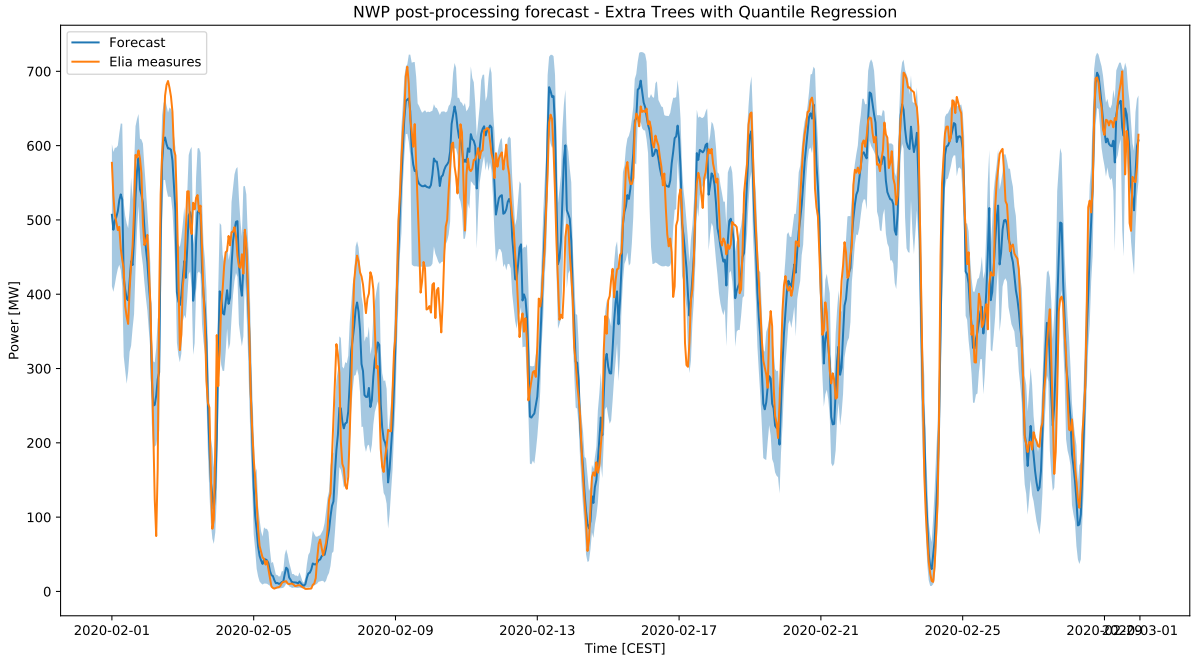

Figure 11 – Wind power forecasting - Extra Trees

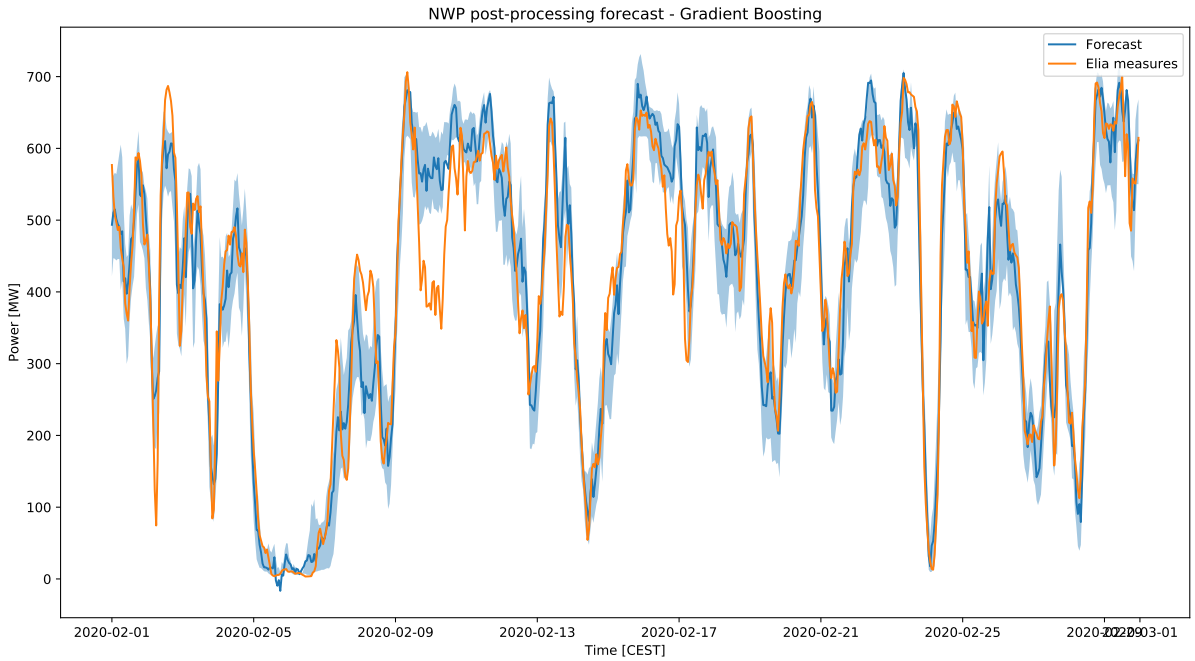

Figure 12 – Wind power forecasting - Gradient Boosting

3.5 Next objectives

- Including 3 new variables in the training set in order to be able to train on older data without degrading the performance :
  - One-day-before power measurement (because there is a very strong correlation between the wind and the wind 24 hours before)

  - Total wind power installed in Wallonia over time.
  - Elia's monitored power over time (the total nominal power on which Elia takes its measurements: currently around 900 MW out of the 1036 MW of Wallonia)

- Testing a Multi Layer Perceptron as learning method
- Having a look at the feature importances (some variables could be physically irrelevant) and considering feature selection

References

[1] Alexis Bocquet et al. "Assessment of probabilistic PV production forecasts performance in an operational context." In: *6th Solar Integration Workshop - International Workshop on Integration of Solar Power into Power Systems*. (2016).

[2] *HelioClim-3 database*. 2020. URL: `http://www.soda-pro.com/web-services/radiation/helioclim-3-real-time-and-forecast`.

[3] Jordan M Malof, Kyle Bradbury, Leslie M Collins, and Richard G Newell. "Automatic detection of solar photovoltaic arrays in high resolution aerial imagery". In: *Applied energy* 183 (2016), pp. 229–240.

[4] Jiafan Yu, Zhecheng Wang, Arun Majumdar, and Ram Rajagopal. "DeepSolar: A machine learning framework to efficiently construct a solar deployment database in the United States". In: *Joule* 2.12 (2018), pp. 2605–2617.

[5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[7] Ekin Tiu. *Towards data science*. 2019. URL: `https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2`.

[8] Conor Sweeney, Ricardo J Bessa, Jethro Browell, and Pierre Pinson. "The future of forecasting for renewable energy". In: *Wiley Interdisciplinary Reviews: Energy and Environment* (2019), e365.

[9] Jakob W Messner, Pierre Pinson, Jethro Browell, Mathias B Bjerregård, and Irene Schicker. "Evaluation of wind power forecasts—An up-to-date view". In: *Wind Energy* (2020).

[10] Wikipedia. *Density of Air - Humid Air*. 2020. URL: `https://en.wikipedia.org/wiki/Density_of_air#Humid_air`.

[11] Carsten Croonenbroeck and Christian Møller Dahl. "Accurate medium-term wind power forecasting in a censored classification framework". In: *Energy* 73 (2014), pp. 221–232.

[12] Xiaochen Wang, Peng Guo, and Xiaobin Huang. "A review of wind power forecasting models". In: *Energy procedia* 12 (2011), pp. 770–778.

[13] Nicolai Meinshausen. "Quantile regression forests". In: *Journal of Machine Learning Research* 7.Jun (2006), pp. 983–999.

[14] Tarek HM El-Fouly, Ehab F El-Saadany, and Magdy MA Salama. "One day ahead prediction of wind speed and direction". In: *IEEE transactions on energy conversion* 23.1 (2008), pp. 191–201.

A U-Net evaluation samples

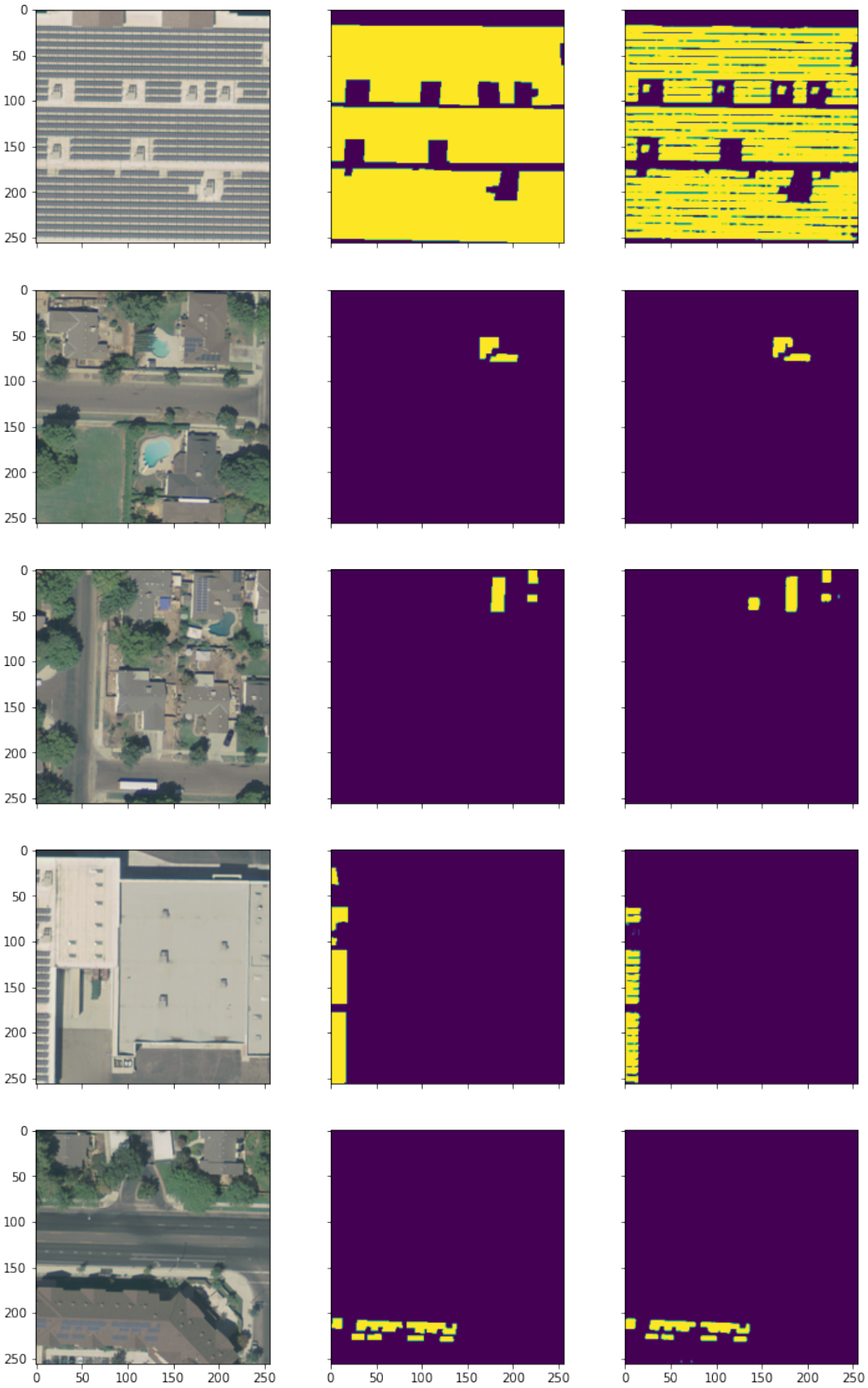

Figure 13 – Random sample 1

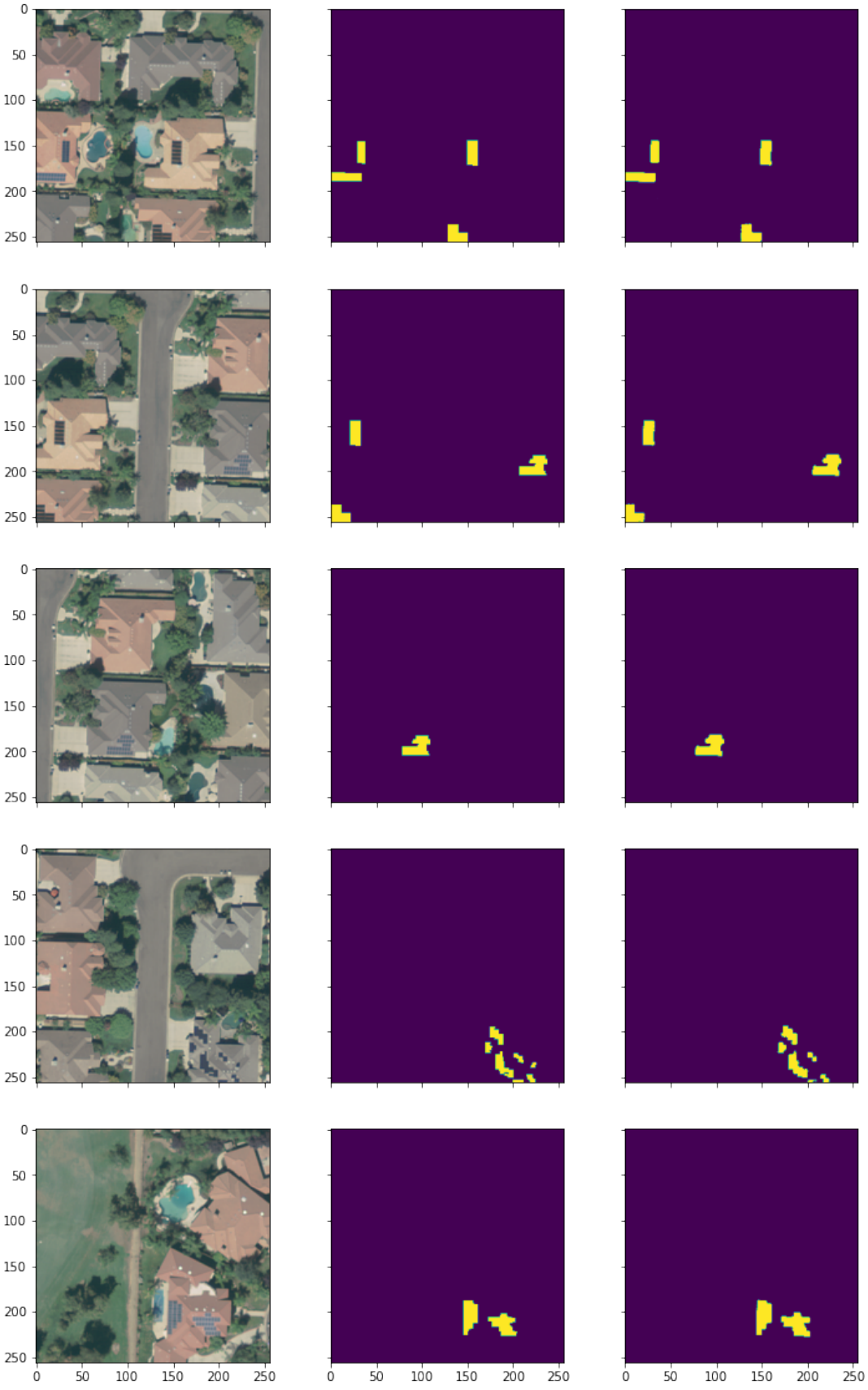

Figure 14 – Random sample 2

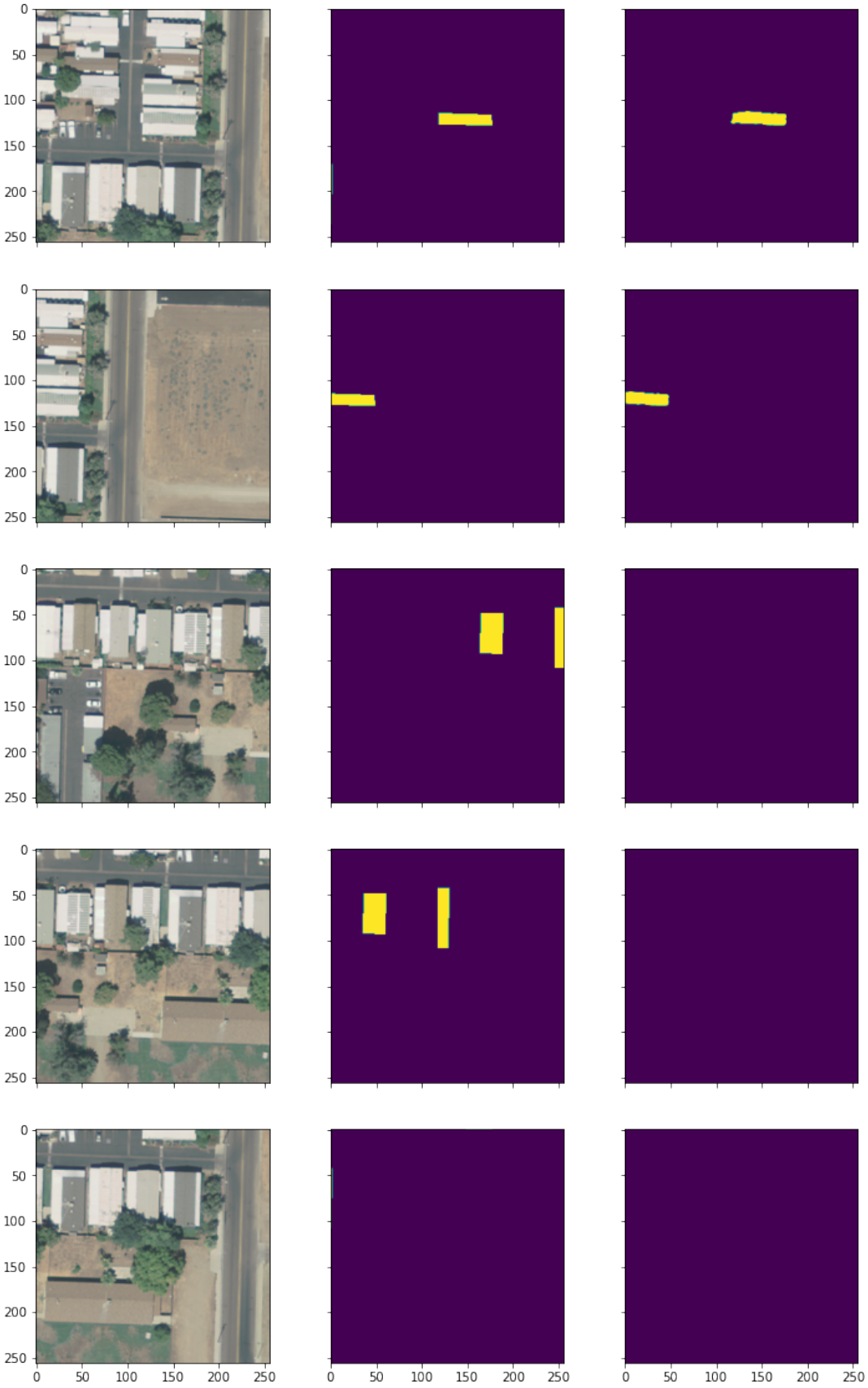

Figure 15 – Abnormal panel colors.

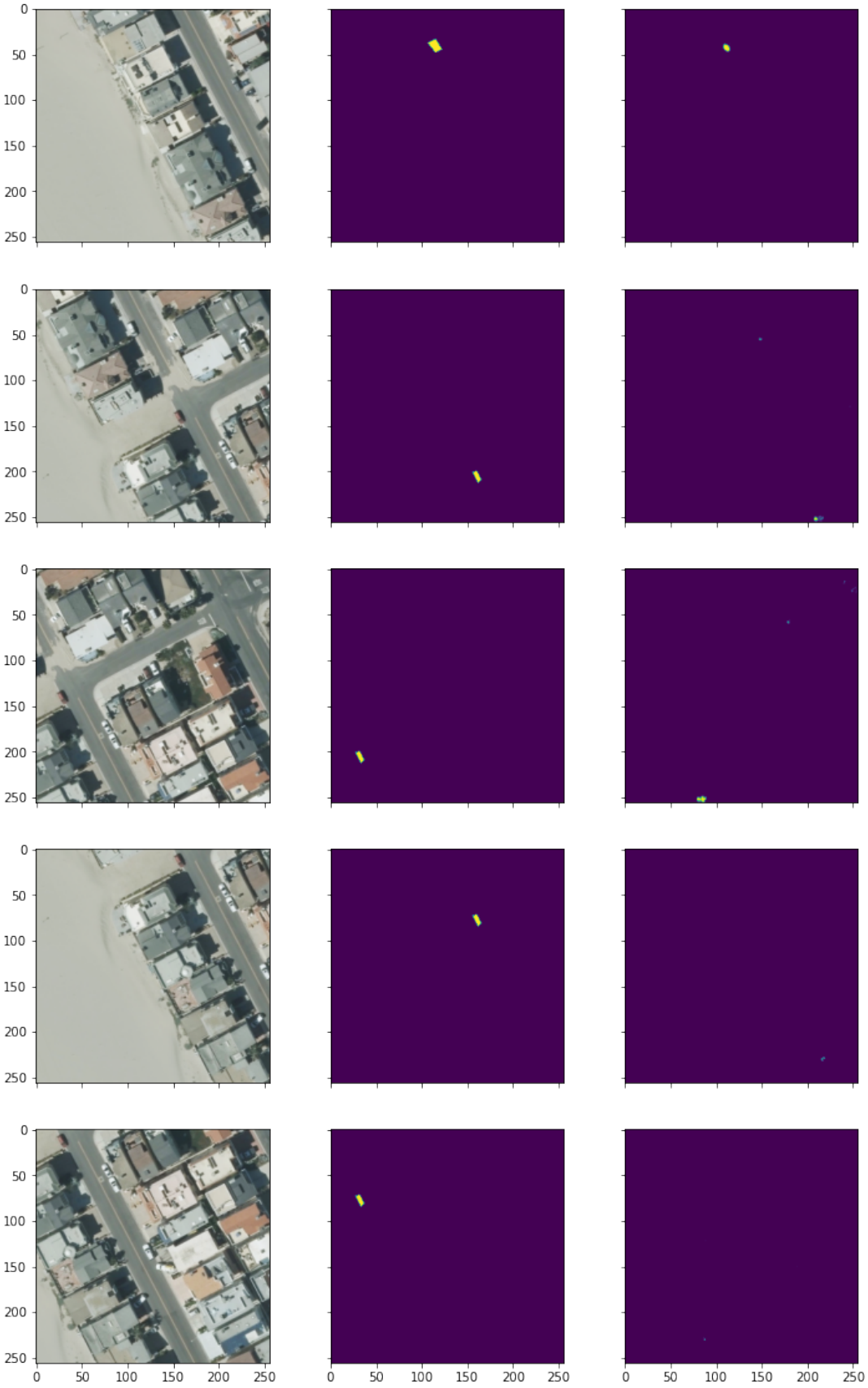

Figure 16 – Abnormal panel sizes.

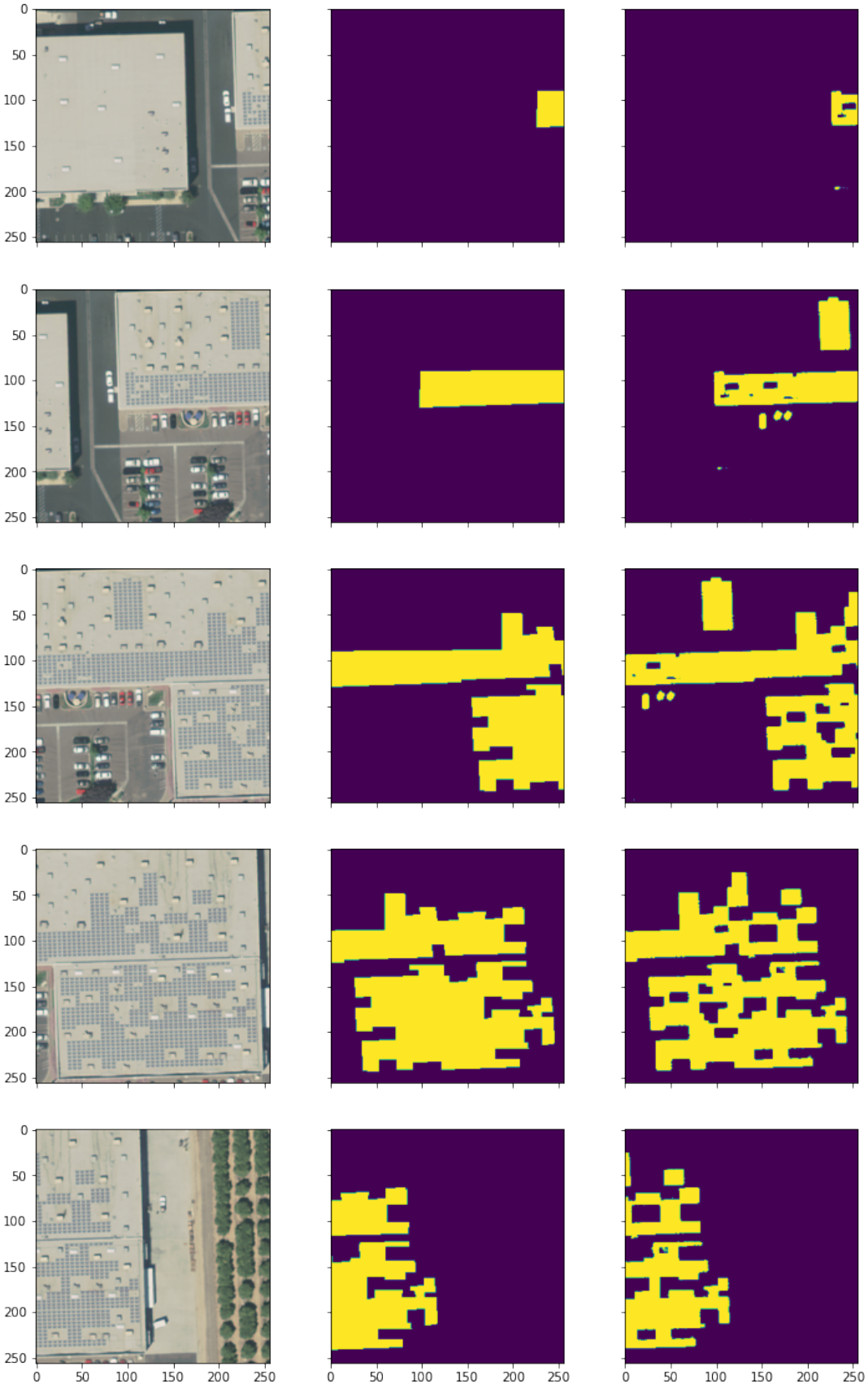

Figure 17 – Predictions better than annotations.

Note. Such annotations could be one of the causes of the relatively bad recall of our model.