# Using Satellite Imagery to Improve Soil Quality Prediction

## Intro

The following study is motivated by the following paper: <https://www.nature.com/articles/s41598-021-85639-y>. Later in this text I refer to this paper as just “the paper”.

The aim of the current study is to test weather inclusion of actual satellite images can improve the prediction power of ML model compared to existing approach of using only tabular satellite data (as described in the paper). The reasons why this might be interesting are:

* Images could include information that is not available in tabular data but has impact on soil quality. In fact, it might even include information that is not obvious for domain experts but plays a role for soil quality (new knowledge generation)
* Changes over time are easier to capture on images as in tabular data. So, using images might be promising for tracking and prediction of soil quality evolution over time (e.g., what happens to the soil, if one would plant a particular crop). It is a task for the future, but it is the future we want to create.

## Target Data

Target data (or variable) is the soil property we want to predict. Currently, I focus only on one soil property: organic carbon (OC). This is a very important soil quality factors, s., e.g., <https://en.wikipedia.org/wiki/Soil_carbon>

The data I use are chemical test data collected all over the Africa. I took the data from <https://gitlab.com/openlandmap/compiled-ess-point-data-sets>. The authors of this data set are (partially) the same as the authors of the paper. But in the paper, the authors use also data from other sources. Considered data set includes measurements of soil properties for almost 190,000 unique locations all over the world. Note that each location might have several measurements (e.g., at different depths). To create this data set, the authors have combined and harmonized data from ca. 40 various sources/studies. So, these data already include considerable expert knowledge that was needed to harmonize data.

In the current study, I consider only OC measurements from African continent. There are 55,885 measurements at 14,186 unique locations. As mentioned above, some of these locations have measurements of OC at various depths.

The data for OC are given as g of OC/kg of soil. So, the value of OC can be theoretically between 1 and 1,000. However, OC above 120-180 is very rare and is associated with so-called organic soil or histosol that is not suitable for cultivation, s., <https://en.wikipedia.org/wiki/Histosol> and <https://www.researchgate.net/figure/2-Distribution-of-Histosols-in-Africa-at-continental-scale-after-Koohafkan-et-al_fig3_323624025>. In addition, the data set includes only 193 measurements with OC>120 (0.34% of all data). So, histosol is not enough represented in the data set and can be barely predicted. Also, we cannot exclude that at least part of these data points are anomalies (wrong/wrongly preprocessed measurements). Because of it and because histosols are not used for agriculture, I exclude these data points for the very first study.

As the authors of the paper point out, the ability of a model to predict soil properties (e.g., OC) at lower scale with high precision is more critical as ability to make predictions at higher scales with high precision. I.e., the difference between OC=1 and OC=2 might be critical. The difference between OC=100 and OC=101 is not critical at all. Therefore, it is better to use (natural) logarithm of OC as target variable as OC value by itself. To be also able to predict OC=0, we make the following transformation (exactly as it was done in the paper):

## Predictor Data

Predictor data (or variables) are the data that are used as input for an ML algorithm. A models learns how to map them on target variables.

In the paper, the authors use following predictors:

* Predictors that describe measurement itself, i.e., location coordinates of the measurement, depth of the measurement, etc.
* Various predictors from satellites or data calculated from raw data from satellites, like climatic data, temperature data, data on structure of terrain etc., but also raw data from satellites like the raw data from Sentinel-2 satellite. In the paper, each measurement location is associated with *a number* that represents a particular value of the considered predictor exactly at this location. So, the authors use satellite data, but not satellite images.

As one can see, the authors of the paper use only tabular data. Not every of these data is easy to get, so, we will focus only on tabular predictors that I could access (relatively) quick. These are:

* Predictors that describe measurement itself:
  + Depth of the measurement
  + Longitude
  + Latitude
* (Tabular) Predictors from satellites (source: API of OpenLandMap)
  + Monthly precipitation (12 predictors)
  + Monthly snowfall probability (12 predictors)
  + Monthly median of daytime temperature (12 predictors)
  + Monthly standard deviation of daytime temperature (12 predictors)
  + Monthly difference between daytime and nighttime temperature (12 predictors)
  + Monthly and annually Fraction of Absorbed Photosynthetically Active Radiation (FAPAR) (13 predictors)
  + Various data on terrain structure, like slope profile curvature, Multiresolution Index of Valley Bottom Flatness etc. (altogether 17 predictors)
  + USGS global landform and lithographical classification (2 predictors)
* (Tabular) raw data from all 13 spectral bands of Sentinel-2-satellite (13 predictors) (source: API of Sentinel Hub)

Altogether 108 (tabular) predictors.

In addition, I have tried to incorporate actual satellite images from Sentinel-2 as predictors. These images use red, green, and blue bands of Sentinel-2 and can be interpreted as “regular” photos. I use 20m resolution (i.e., one pixel corresponds to 20m x 20m square). Each image has 224 x 224 pixels. It means that each image represents an area of roughly 4.5 x 4.5 km2. These data also come from Sentinel Hub.

## General Methodology

I have created two models. One uses only tabular predictors, another tabular predictors and actual images. While creating these models, I use a consistent methodology, described below. Consistent methodology is required to be able to compare two models and measure the effect of adding images as predictors. Otherwise, any improvements/degradation in accuracy might also come from a different methodology.

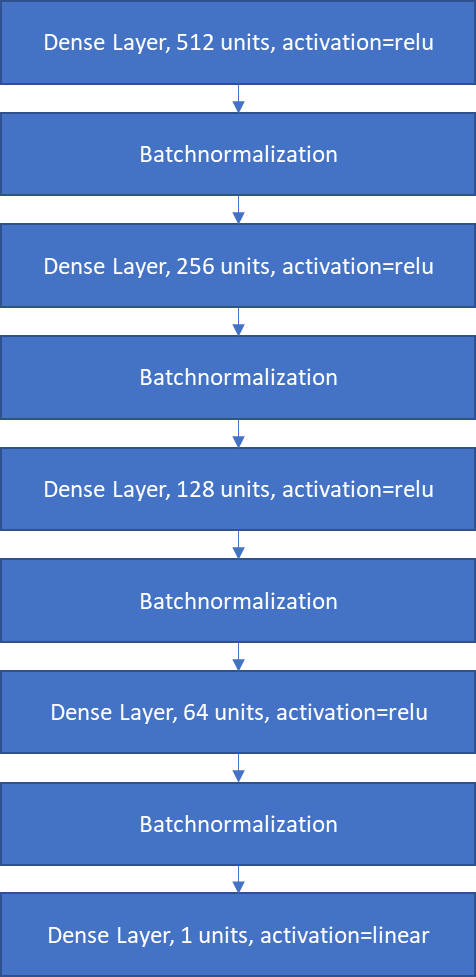
* Data are randomly split into three blocks:
  1. Training data (60% of all data): are used to train the model
  2. Validation data (20% of all data): are evaluated during the training but are not used for training. Observing evolution of optimization metric on this data set allows prevent overfitting.
  3. Test data (20% of all data): Data never seen by an algorithm. This data is used to evaluate the quality of the algorithm. In this report, I mention metrics for only test data.
* Currently, I use only Neural Networks
* I optimize towards minimum of mean squared error (mse)
* I use Adam optimizer with learning rate = 0.0001 (for simplicity, other features of Adam optimizer are defaults)
* I use batches of size = 32
* I use Early stopping on validation data set with patience=30 (for tabular input) and 10 (for model that uses images. Otherwise, calculation is too long). I.e., if the mse on validation data set does not improve for 30 (10) epochs, the training process is stopped and the model coefficients that produced the best mse for validation data set are restored.
* The major quality metric I report is root mean squared error (rmse). I use it because it was also used in the paper
* In addition, one could also redefine the problem as classification problem (instead of regression). As a kind of natural classes one can use rounded ln(oc+1). I.e.,
  1. Class 0 corresponds to oc between 0 and 0.65
  2. Class 1 corresponds to oc between 0.65 and 3.5
  3. Class 2 corresponds to oc between 3.5 and 11.2
  4. Class 3 corresponds to oc between 11.2 and 32.1
  5. Class 4 corresponds to oc between 32.1 and 89.0
  6. Class 5 corresponds to oc between 89.0 and 120

The metric here would be the number of data points with correctly predicted classes divided by the number of data points (accuracy). Note that I tried to build classification models, but in my experiments, they produced not so good results as regression models (accuracy was used to compare the models). The reason might be that classification models do not recognize natural order of classes. Nonetheless, I still report also the accuracy as metric to provide a better feeling on quality of the model.

* After a model is trained and rmse and accuracy on test data set are calculated, we also need to know how much we can trust these results. The reason for potential mistrust is that even though the splitting of data into training/validation/test data sets is made randomly, a particular choice of training data might have impact on the accuracy metrics. It means that applied on completely new data, the model would produce predictions of different accuracy. To take this effect into account, I repeat the random splitting 10 times (with different seeds). Then the model is trained from scratch without changing any of hyperparameters (the only exception is the number of epochs since they are controlled by early stopping). The resulting rmse and accuracy are combined with the original metrics. So, we have 11 measurements for rmse and accuracy that have some particular distribution. This distribution is much better indicator of model quality as only one measurements of rmse/accuracy. This procedure is basically equivalent to nested 2-fold cross validation.

## Model that uses only tabular predictors

The model has the following structure:

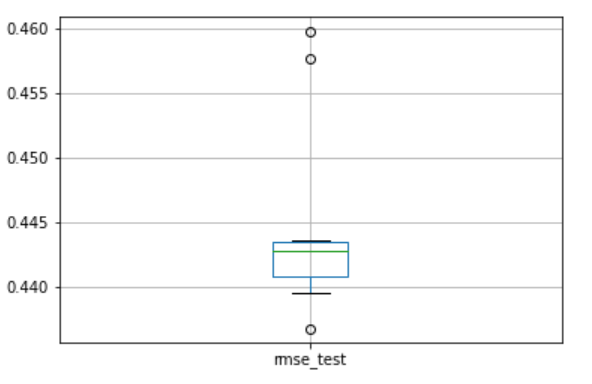
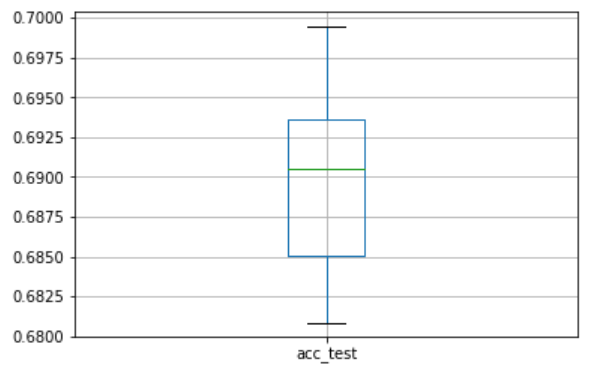


I use Batch Normalization to accelerate training. It has also a moderate regularization effect. In my experiments, I didn’t see a need for application of further regularization techniques.

The results of this model are not really impressive. The rmse equals **0.440** on test data and is quite large compared to **0.369** reported in the paper. However, one should keep in mind that in this study I used much less data points as the authors of the paper (ca. 56.000 vs. ca. 122.000). Incorporating further data for target variable might considerably increase quality of the model. In any case, the aim of this study is not to beat the paper but check weather usage of satellite images can improve precision of model predictions.

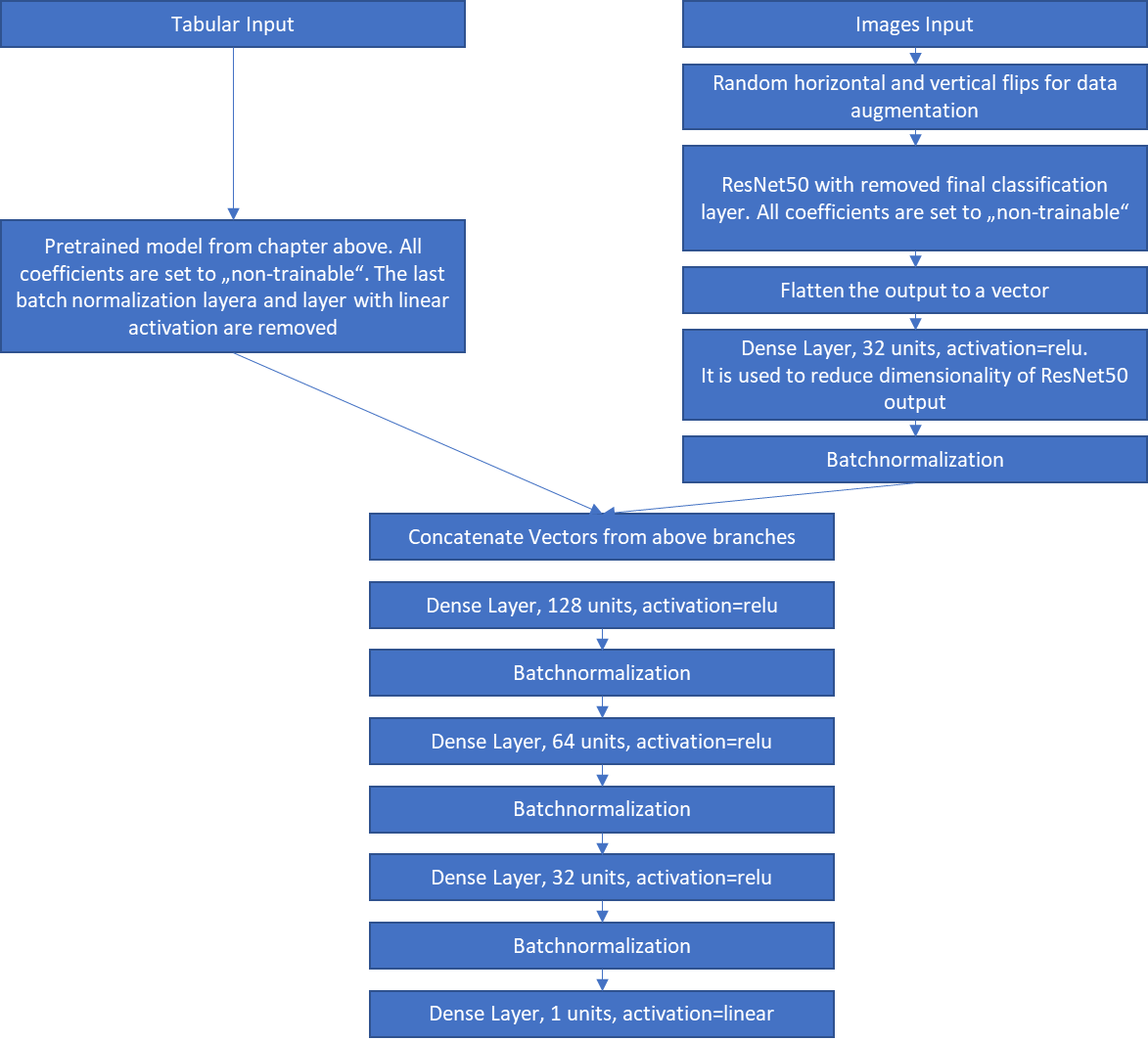
Rounding predictions and true values for target variable and treating them as classes leads to accuracy = **69.94%** on the test data.

Repeating the training with different splits of training/validation/test leads to following distributions on test data:

## Model that uses tabular predictors and images

The model has the following structure:



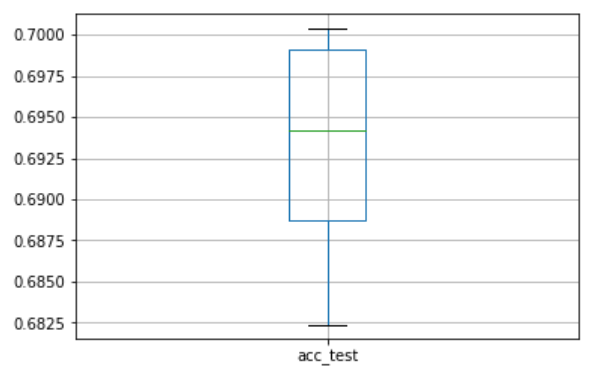
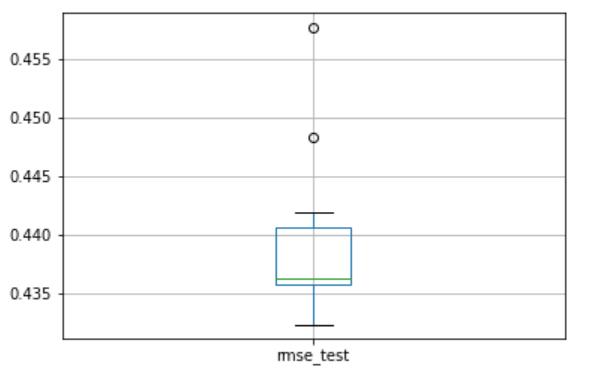
Note that the model has two branches. The first one is pretrained model from the previous section. But the last batch normalization layer and the layer for final prediction (that with linear activation function) are removed. So, the output of this branch is a 64-dimensional vector. All the coefficients of this branch are not trainable.

The second branch is the famous ResNet50 model pretrained on ImageNet data set. Only the last classification layer is removed. I do not re-train ResNet50 and keep all coefficients not trainable. After reduction of dimensionality, the output of this branch is connected to the output of the first branch and is fed into the next model.

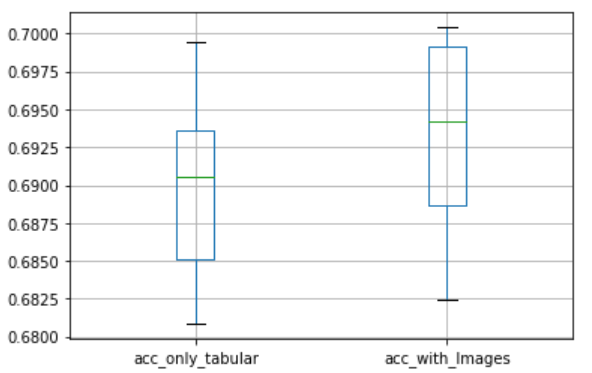
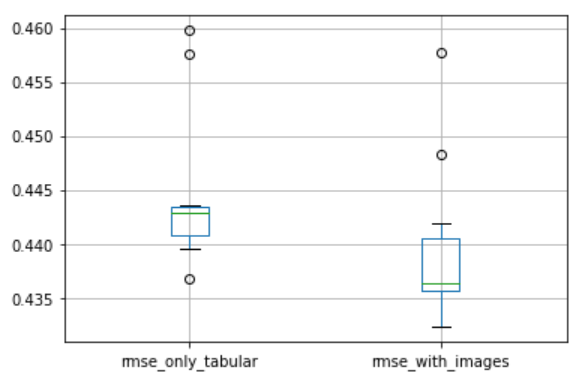
The results are rmse = **0.436** and accuracy = **70.04%.** So, almost no improvement.

Here I also use different train/valid/test splits for cross validation. One must be very cautious here. Since the branch for tabular data is pre-trained on the same data and since the weights of this branch are not trainable, using a not suitable train/valid/test split might lead to data leakage. So, for each run of cross validation, the loaded model for tabular branch must be the model that was pretrained on exactly the same train/valid/test split as in the current cross validation run. Otherwise, the data that originally were used in training of the model used in tabular branch might be part of the current validation/test data set and vice versa. So, in this case, the tabular branch would include the information outside of the training data set (data leakage) and would create too optimistic results (In fact, I did this mistake and observed results too good to be true).

If everything is done correctly, repeating the training with different splits of training/validation/test leads to following distributions on test data:



Compared to the model without images:

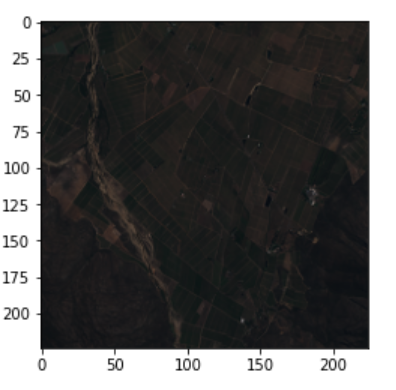
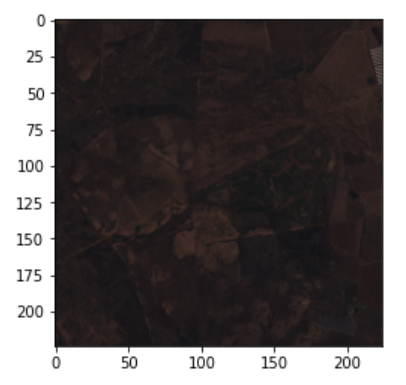


The improvement is there but only marginal a one. The reason for that is probably the bad image quality, s. next chapter.

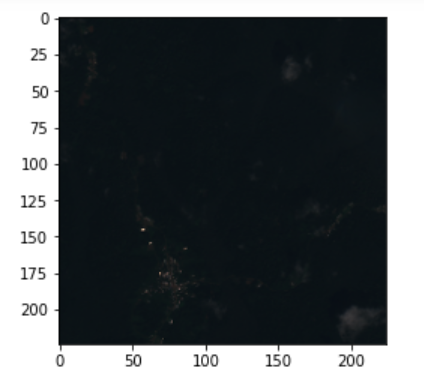
## Activities that might increase accuracy

* I’m currently using less than a half of target data points that were used in the paper. Taking more points into account can increase the accuracy of both models. However, this might be a non-trivial task since it will require data harmonization. The later might require some sort of domain expert knowledge (currently I do not know how much). The paper provides basically no information on how the data were harmonized. Weather we can repeat the same/similar harmonization process without external help is a question.
* Current images I use are of quite bad resolution (20m per pixel). These images can provide only high-level information on the surrounding terrain of the measured location. High-detail information (like, e.g., currently planet crop) is not available at the current scale but is required to be able to predict evolution of the soil quality parameters. Higher resolution might also increase the difference in accuracy level between above two models
* **Current images have often bad quality**: there are images with a lot of clouds and images created by night. Sometime both.

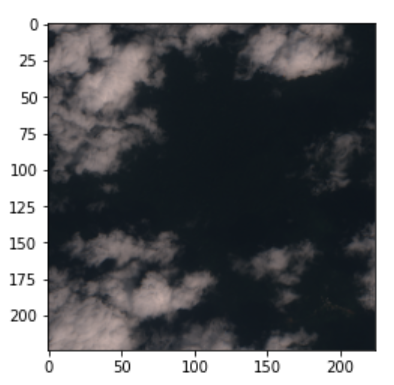
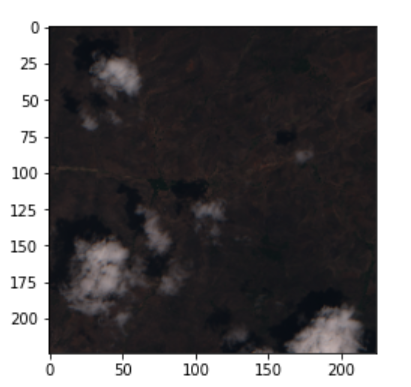
Good examples:



Bad examples:

Ein Bild, das Text, Monitor, schwarz, Screenshot enthält.

Automatisch generierte Beschreibung

The fact that there are clouds on images is very strange, because while doing API calls to Sentinel Hub I say: “take an image from the last two month with lowest cloud coverage”. It looks like some areas are very cloudy. Maybe we should consider more than 2 months. I have now idea how to control whether day- or nighttime images are downloaded. I have to read documentation more carefully. In any case, the problem seems to be quite big: from 10 images I randomly looked at, 4 were trash.

Of course, the quality of images has a large impact on the results. We should pay more attention on data preparation. But it will be probably not an easy task since we will have to look at thousands of images manually and try to identify all reasons for bad quality of images and try to find corresponding solution. Here we might have to acquire some better understanding of satellite imagery in general.

* There are also some potential improvements on the model side:
  1. Instead of having a pretrained branch for tabular data, we might allow the model to train/retrain the weights in this branch. However, it will require more computing power/time for training
  2. ResNet50 was trained on images from internet. It was not trained to work with satellite images. I’m quite sure that it leads to a bad interpretation of our input images by ResNet50. There are two ways how we can try to solve this problem:
     1. Make some of the ResNet50 layers trainable. However, this will dramatically increase training time. I’m already working on RTX3080 with 16GB. It is one of the best GPUs for ML available for laptops. With my setup the training is progressing very slowly (for cross validation of the model with images I spent ca. 18 hours). We will either need a lot of time or very strong cloud computing power.
     2. BigEarthNet (<https://git.tu-berlin.de/rsim/BigEarthNet-S2_19-classes_models> or <https://www.tensorflow.org/datasets/catalog/bigearthnet>) has pretrained ResNet50 (and other) CNN architectures on satellite images. So, their version of ResNet50 should be more suitable for us. However, the implementation seems to be not trivial.

In addition, one might test other pretrained architectures for images (e.g., other ResNet architectures, VGG-family of models etc.)

## Conclusion

Currently, both our models are far from perfection. Both of them will benefit from more measurements of target variable. These are already existing but will need harmonization efforts.

I was not able to answer the question “can inclusion of actual satellite images improve prediction accuracy?” The reason for it is the bad quality of current images I have.

We will need to work mainly in two directions:

* Understand and prepare images better
* Include more data points on the target variable

In the short term, i.e., for MVP Web-App for application for the grant from WFP and ESA, I’ll use the simple model since it is faster and needs less calls to Sentinel Hub API (that has restrictions on the number/complexity of Calls if one uses these services for free). But we still should say that we are working on a technology that incorporates satellite imagery to improve predictions on soil quality measures and to enable prediction of evolution of these measures.