Can Google Street View be a Reliable Source of Deep Learning Data?

This blog is part of the reproduction project for the needs of the course Deep Learning (CS4240), in the Technical University of Delft (TUD) for the academic year 2023-2024.

The contributors of the project are:

Jelle Vogel / JelleVogel/ J.Vogel-1@student.tudelft.nl / 4459911
Task : Data Extraction

Aleksander Seremak / aseremak / a.k.seremak@student.tudelft.nl / 6075401
Task: Data Filtering

Niek Schattenberg / nschattenberg / N.E.Schattenberg@student.tudelft.nl / 5121930
Task: Model Training

Alexandros Theocharous / atheocharous / a.theocharous@student.tudelft.nl / 5930901
Task :Result Assessment and Comparison / Blog Writing

Deep learning is constantly praised by academics for its ability to achieve great performance and flexibility on problems that normal machine learning approaches could only dream of [1]. On the flip side, for those numbers to be achieved, deep learning methods require an incredible amount of proper data, which sometimes if not outright impossible are extremely hard to be obtained by traditional methods [2]. Existing data sets may be insufficient and gathering and annotating new entries can be deemed extremely time consuming and expensive.

For this particular reason researchers are exhibiting interest in creating pipelines that can effectively and reliably utilize existing open-source software, like Google Street View, to create with low effort and monetary resources, streams of data for deep learning problems. On those grounds Jordi Laguarta Soler , Thomas Friedel and Sherrie Wang published the paper: **Combining Deep Learning and Street View Imagery to Map Smallholder Crop Types** [3], where they propose *"a cost-effective automated deep learning pipeline to generate crop type references and remote sensing crop type maps with minimal manual labelling"*. According to the results, this approach shows that, in Thailand, the resulting country-wide map of rice, cassava, maize, and sugarcane achieves an accuracy of 93%. Our goal is to see how a pipeline like this can generalize on similar deep learning problems that also suffer from data insufficiency and explore the possible strengths, limitations and steps that require specific attention when following that approach. The problem we tested the method on was the classification of the health of trees in Delft.

An overview of the pipeline the original paper used can be seen in figure below:

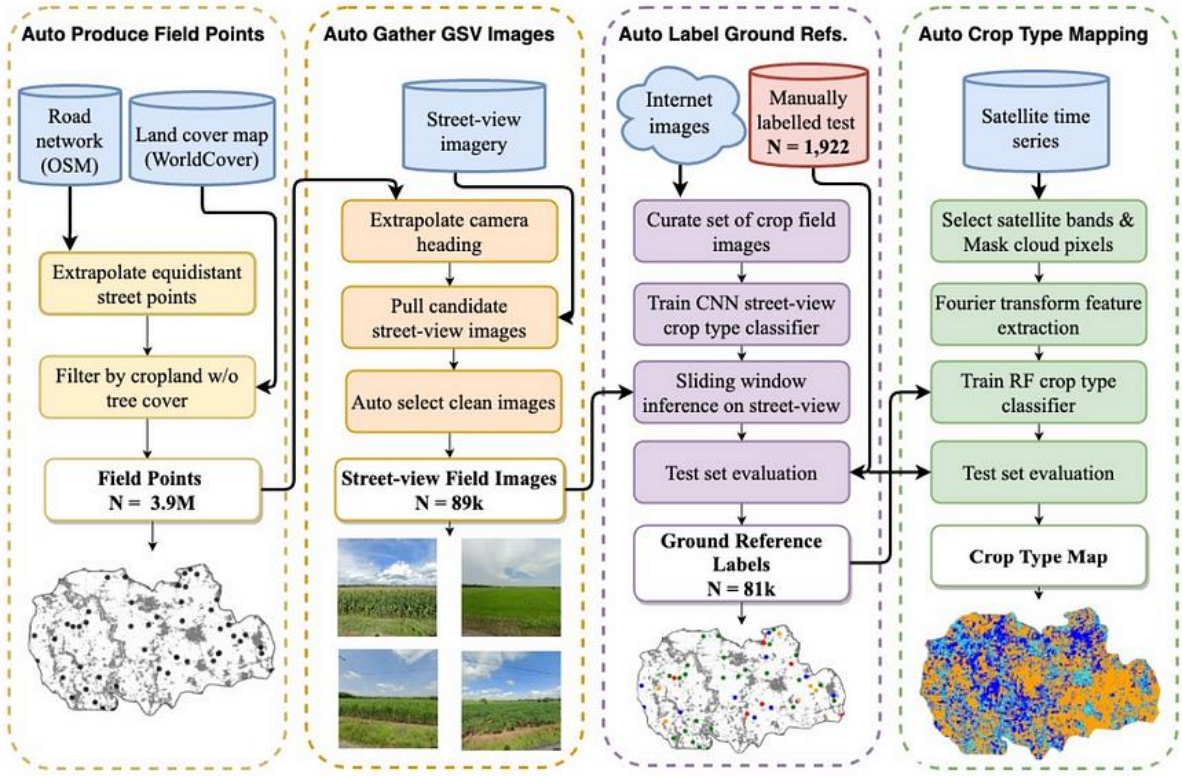


Figure 1The pipeline as presented in the original paper

A short explanation of the discrete steps followed is presented below to shed some more light:

Extrapolation of Equidistant Street Points: The latitude and longitude coordinates of candidate fields and their corresponding roadside image along all roads in the country were gathered. For this task Open Street Map (OSM) was used, along with Overpass API to query all OSM ways within Thailand.

Filter Points Using Land Cover Map: Using existing land cover maps, street points near farmland were filtered to remove field points that are not visible from the street due to obstructions. Sequentially points containing any tree cover within the same 10m radius were also removed and they also only kept points that the Google street view images were taken from during the growing season, for the fields to be visible.

Extrapolation of the Camera Heading and Field Point: The street bearing θ was computed and then the heading for the camera was calculated so as to face the two fields on either side of the street as θ ± 90 degrees. The side not containing the field was filtered on the next step.

Classification of the In-Field Images: A dataset with 2986 images hand-labelled into {field, not field} was created and a binary classifier was trained using a ResNet-18 pre-trained on ImageNet, to get the final transformation of the Google Street View data.

Training set Compilation from the Web: To classify the type of crop training sets of crop type images where created from two sources: Google Images Creative Commons (hereafter "WebCC") and iNaturalist.

Labeling Crop Type Ground References Manually: A test set of GSV images randomly sampled by geography was manually labelled by a plant taxonomy expert.

Train the Street-Level Crop Type Classifier: Models were trained on 6 different datasets, containing combinations of the expert labelled images, the images from WebCC and iNaturalist and GPT-4V labelled images.

Performing Prediction from the Steet-View Images: Sliding windows method across each image was used from the classification on the filtered image.

The last part of the pipeline performed Remote Sensing-Based Crop Type Mapping which is something that our implementation did not try to recreate so it is not further elaborated on.

It should be noted that the expert-labelled dataset gave the highest score of 93% accuracy, but it only uses n = 1153 samples, making it a very viable approach. Also ChatGPT-4V came as a close second making it also a very interesting approach for future exploration.

Our implementation tests the second and the third part of the pipeline for the health of trees in Delft. The dataset used included a label for each tree in Delft, from six possible ones, ranging from "Good" to "Dead" along with their geocoordinates assigned by an expert, just like it was done in their paper.

Our first task was to extract a Google Street-View image for each of the dataset entries that we would then filter and use to train the classifier.

A Google Cloud Console (GCC) account was created to gain access to the Google Street-View (GSV) API. For each different expert label a folder was created that contained the corresponding data and additionally another set of folders that would contain the images. That way the input images will be classified according to their health by being placed on the different folders. Using the geocoordinates of each tree, the images were extracted and placed to the right folder. Given those coordinates, with the help of the metadata the date the GSV picture has been taken was found. Using these dates, the images taken in autumn and spring where taken out of the dataset to increase the chance that there are green leaves on the trees.

We also detected an opportunity to filter the data further by checking the date of the GSV image with the date of expert labelling and keep only close dates. However, almost all pictures were taken more than a year off from the expert labelling, which might cause issues as the health of the trees can change during that time, or the trees may be not there anymore.

Figure 2Tree that does not exist anymore

An issue we encountered that was worth noting was with the heading angles. In the appendix of the original paper there is a mathematical explanation which describes how they extract their heading angle. For example, given a heading angle of 0 degrees to the URL of the API, the camera will be facing north. When a geolocation is entered in the GSV, the API returns the nearest image to that geolocation from the set of available GSV images. With the help of the geopandas library which gives you the option to create a small graph of interpolating points in a road network, one can give the geolocation of the car and set a radius of for example 30 meters (like it was suggested on the original paper) and then it will add all the points it knows about to the graph. You can then extract the two points inside of the geopanda's dataset from your input geolocation and then use some geometry to determine the best approximate heading angle. In our case this did not really work because in the original paper, after they find the heading angle, they don't know on which side of the road the field is, so they call the API twice: once for the heading angle +90 degrees and one for the heading angle -90 degrees and then filter the one that does not contain any field on the next step. For us this did not always work was because of the inaccuracy of the geolocation of the GSV image and the geolocation of the tree. The tree is not always exactly perpendicular to the street at the point of the GSV image. In the paper, where they work with fields, this inaccuracy is less of a problem as a field probably takes up more

space in the image than the tree would. So for this application it is more important to accurately find the tree placed exactly in the middle of the image. We also assume that there might be some inconsistencies with the geolocation of he tree that was reported by the expert in the dataset and the actual geolocation that the tree is in the GSV map.

This issue was solved as we found out that you are actually able to disregard the heading angle entirely in the URL to the API and then the API will adjust its heading automatically to look at the input geocoordinate from the geocoordinate found for the GSV image. A very simple solution but luckily solving this entire issue. It was noted that this angle, while it manages to show the trees in the image it was often not perpendicular to the street at all and the trees where not always aligned to the middle of the image.

Figure 3Tree further on the back

Figure 4Tree not properly aligned

The next step of our implementation included making a filter that only keeps the GSV images that we deem fit for the training of the classifier. Over 2000 images from the dataset that was previously created were annotated by hand using the PyQt image annotation tool [4], in a similar manner to the data filtering classifier the paper used. Then the classes were balanced in such a way that the number of images of trees are similar to other classes, as it was also done in the paper. We end up with the following distribution:{0: 294, 1: 300} where 0-no_tree and 1-tree. Finally the data was divided to train, validation, and test sets with proportion 80% - train + val, 20% - test. Further 85% train, and 15% val. It should be noted that the original paper also used a small amount of expert data for this step as it is one of the main points to prove cheap and fast data generation.

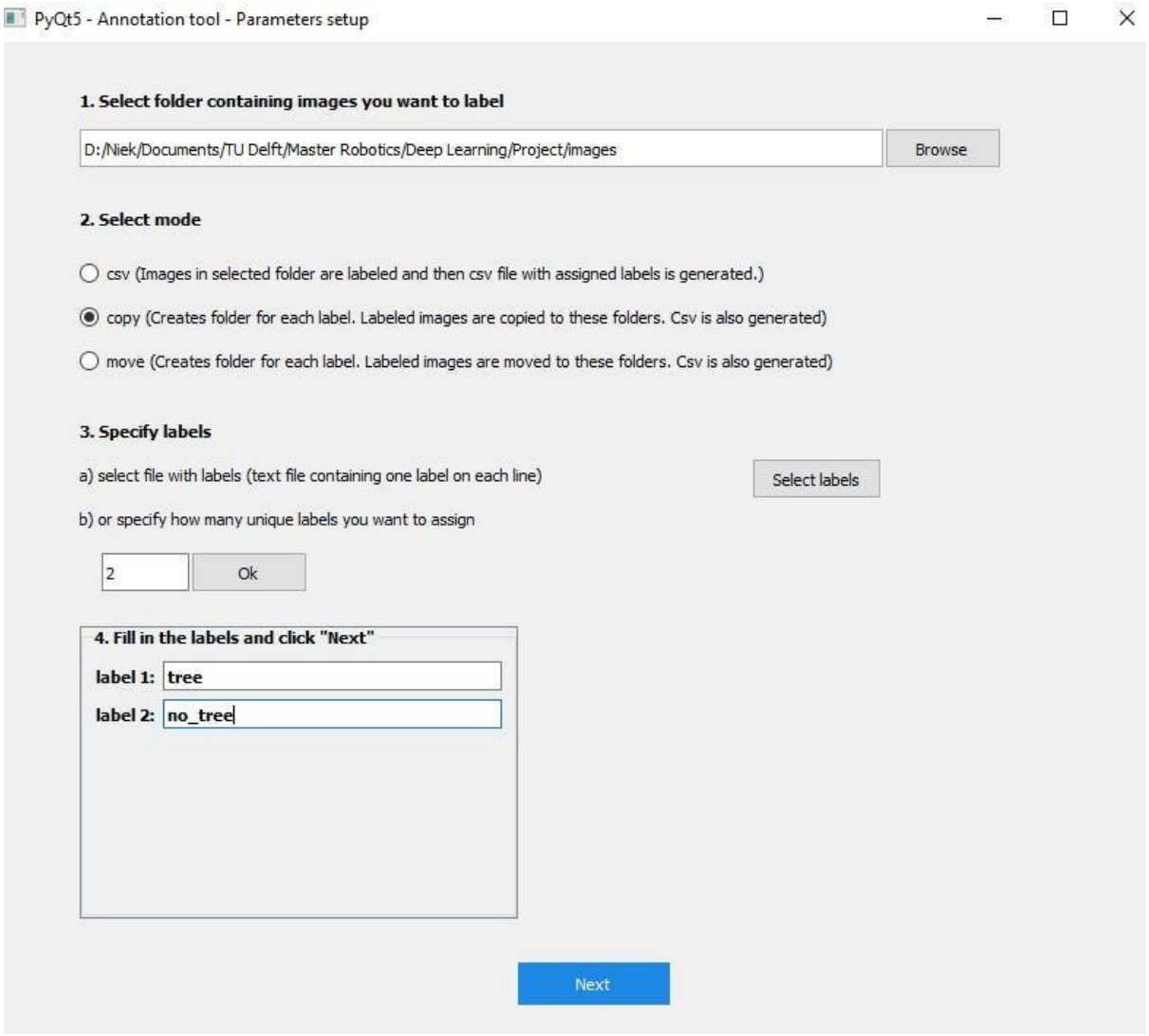


Figure 5The PyQt annotation tool

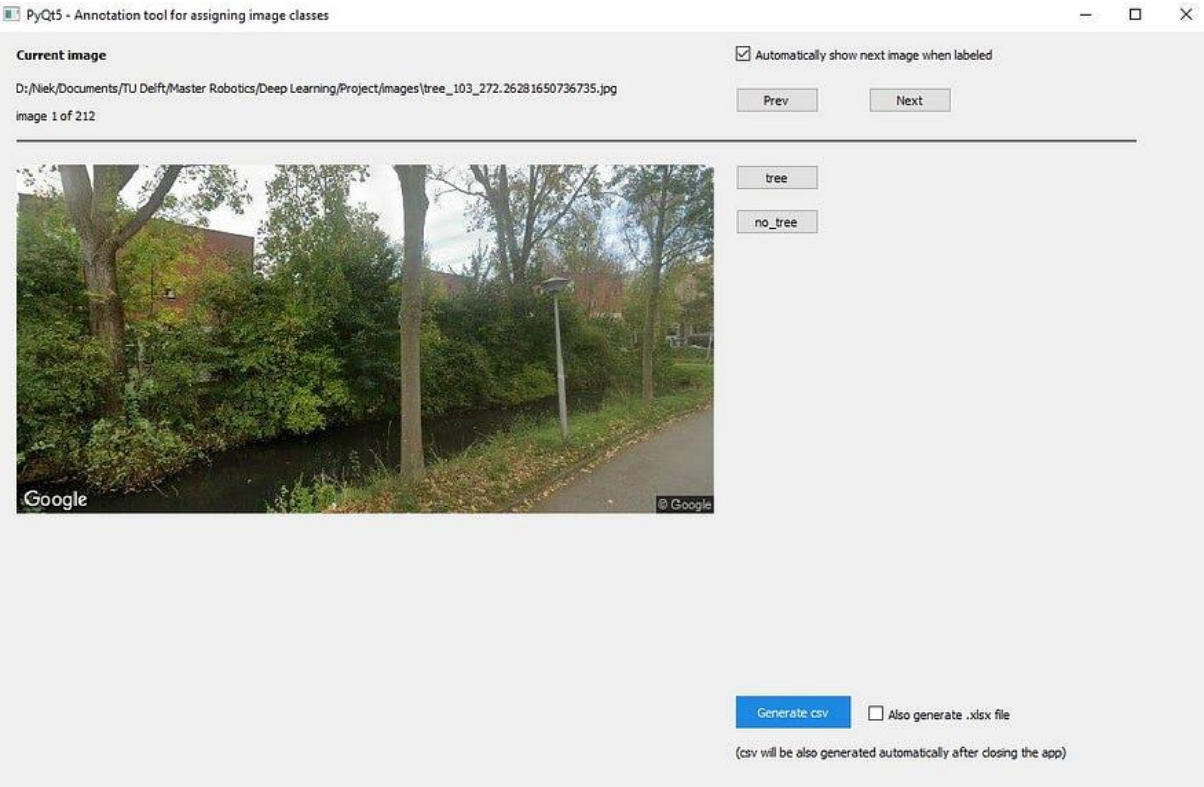


Figure 6Classifying the images with PyQt

Based on that data the ResNet18 was trained. After some testing and adjusting the best performance that was achieved is 70% on the test data. The evaluation was conducted on the test set. We suspect that this may be due to the images from GSV. It is hard to contain the entire tree in the image. In most cases the only visible part of the tree is branch. In addition in some cases there are a lot of trees in the image and not in the centre, as also mentioned above, therefore it is hard to distinguish which tree is being examined, and therefore to decide on label.

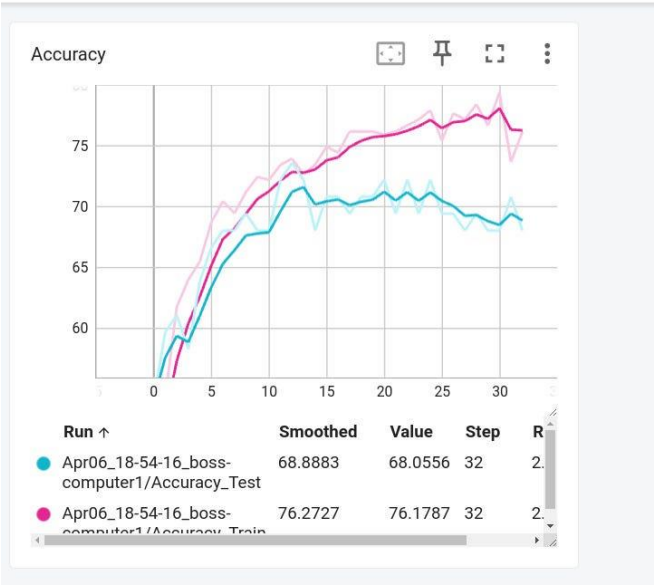


Figure 7Accuracy of the classifier

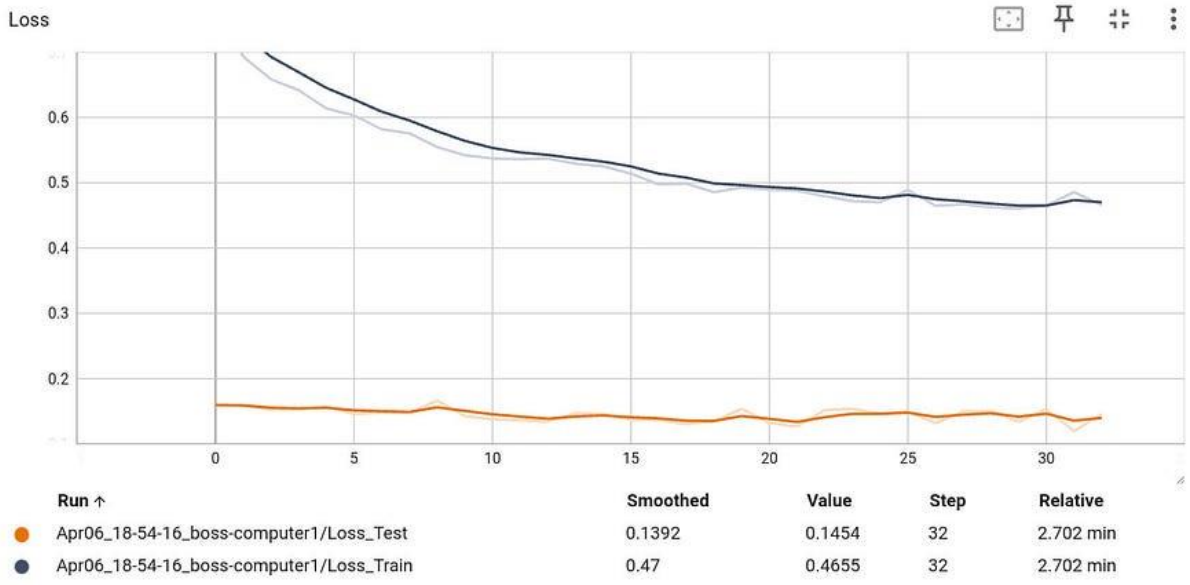


Figure 8Loss of the classifier

From the graphs above it can be observed that the model is overfitting. This problem can potentially be solved by manually annotating more images but we deem that this is counterintuitive to the point of using few expertly labelled data and also the classes would be then unbalanced. Another thing that we would really like to stress is that creating data for the classification of good or bad images for the next classifier is very different across the two problems. For the fields problem it is much easier to determine whether the image is usable based on how visible the field is. Also the fields tend to span across the whole image and be homogenous, while trees only take a small part and can be in very different positions and locations within the image. Finally and most importantly the next step needs to classify the health of the trees according to the experts evaluation, with this goal in mind it is very hard to determine which photos can potentially make good data since we do not know the reasoning behind the experts evaluation. We might see a full tree perfectly aligned in the middle of the screen and deem it as a great candidate but in reality it is not as it has a specific bug living on its leaves, that makes its health condition poor and we need a very close up image to its leaves to determine that. On other cases the same issue might be for its bark or a combination of the two that the image resolution cannot determine. For this decision making we believe that the help of the expert that made the dataset is really important.

Figure 9Example of tree with good location but low resolution

The final part of our recreation included the training of the Restnet50 model using the data obtained above for the tree health evaluation. In the paper sliding window voting was used. This means that multiple model evaluations vote one image and results are aggregated to improve classification results.

The paper used 300 by 300 pixels with a stride of 50 for the windows. The amount of windows the authors did not specify. Every model evaluation votes for one class label or only the evaluations above a certain probability. The results of the original paper showed that sliding window classification improved performance from 82% to 90%. This increased to 93% by only using probability threshold of 0.9. Our reproduction looked only at the sliding windows without probability as we are not certain if probabilities of 0.9 will occur.

Resnet50 needs images of 224 by 224 pixels. In our case, the windows cut outs of the image are this size. A stride of 50 is used on horizontal overlap like the paper, with 3 windows around the centre. This makes sure that the slim trees that are off centre stay within all windows. Vertically the stride is decreased to 38 to make 3 windows fit the Google Street View resolution of 600 by 300 pixels. So together 9 windows are looked at.

The dataset of prefiltered images by Resnet18 are balanced by taking 1000 samples from Matig and Redelijk together with all 947 samples from Slecht. These images are pre-processed by cropping to the window boarder region and normalizing the image, according to the specification mentioned in the Resnet50 input requirements [5].

During training random windows are selected, one for each image. The image of 600 by 300 pixels is first cropped to the region where windows will be present for testing. Then randomly

within this boarder a window will be cut out of the 224 by 224 pixels. Stochastic gradient descent is used with a learning rate of 0.001. The test set contained 72% of data.

Figure 10The region where windows will be present for testing

For the testing, the sliding window voting is used. In the image below you can see the windows that are cut out of one image, all 9 rectangular windows will be evaluated. Together they overlap a large portion of the big trees to get more information than only one image.

Figure 11All of the sliding windows

After the evaluations the most certain class labels are voted upon. After this the mode is taken to find the most voted class. If there are two classes in the mode, then we decided to call the decision invalid, as there is no consensus of the result. The paper also disregarded results that where inconclusive, like results with a low probability.

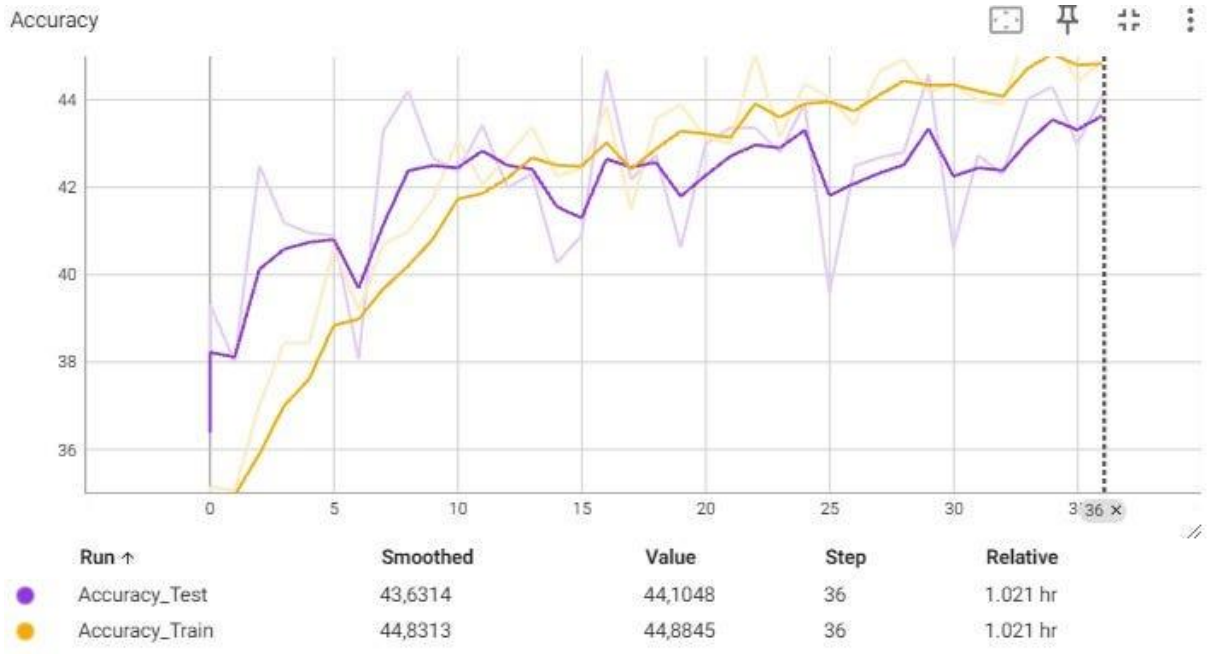
A difference that this approach encounters when classifying tree health instead of field types, is that in the image below two trees are present within the bold window region. The windows on the middle vertical row are drawn in dashed lines. As a result all windows see the second tree as well. Because convolutional neural networks are mostly shift invariant and both the tree we want to classify and the other tree appear sometimes in the middle and sometimes to the side it makes it impossible for the model to determine which tree to classify at the end. This might deter the final performance of the classifier on this particular task.

Figure 12Problematic data

Another point about the sliding windows method used on this particular dataset is that due to the shift invariant nature of convolutional neural networks, the tree below fits in some of the middle windows entirely, hence making the sliding window voting unneeded. There are a lot of these small trees in the dataset. Some trees benefit from the vertical window variants like the image above. Only some trees benefit from all nine windows. This means that the sliding window makes for a less efficient solution that is in most cases not beneficial.

Figure 13Small tree that only fits in a few windows

After training the final model on the three classes as we mentioned above, with balanced data we get the following results.



Run	Smoothed	Value	Step	Relative
Accuracy_Test	43,6314	44,1048	36	1.021 hr
Accuracy_Train	44,8313	44,8845	36	1.021 hr

Figure 14Accuracy of the final model

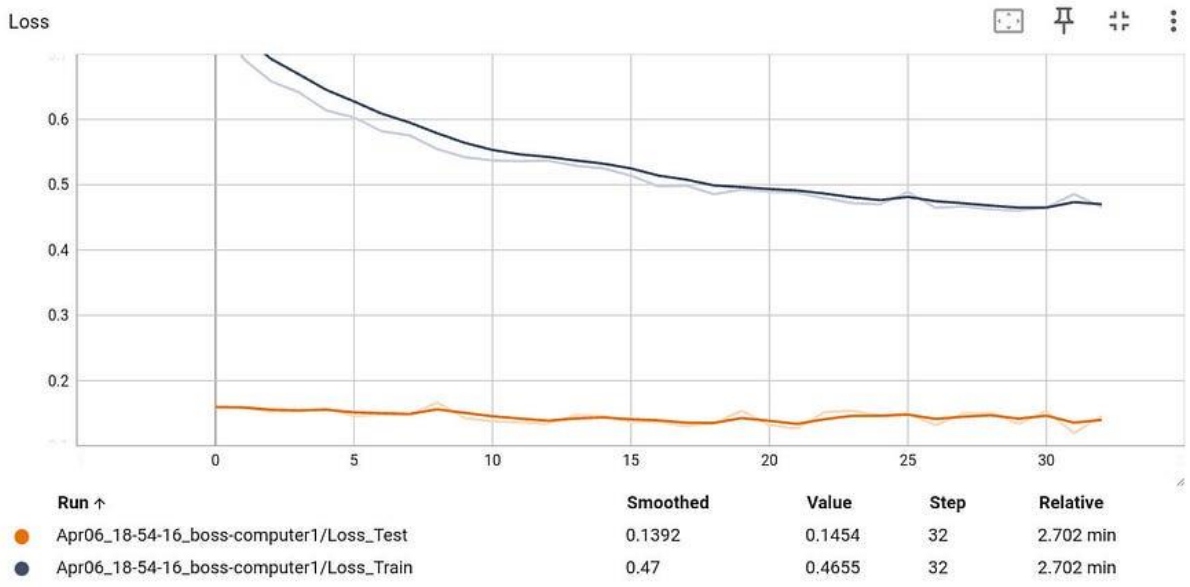


Figure 15Loss of the final model

```
Report per class
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        64
           1       0.41      0.86      0.55        83
           2       0.60      0.37      0.45        82

    accuracy                           0.44       229
   macro avg       0.34      0.41      0.33       229
weighted avg       0.36      0.44      0.36       229

Confusion matrix
[[ 0 54 10]
 [ 2 71 10]
 [ 2 50 30]]

Micro (global F1
0.4410480349344978

Accuracy: 44.10480349344978, Average loss: 1.0646237134933472
```

Figure 16Statistics of the model

From the end results we can infer that the models performance is really poor. With three classes a random vote would have an accuracy of 33% and here 44% is achieved on the test set. For the two major classes it seems that more votes go to the actual class than the other two but the error is still great. The smaller class is also completely disregarded. This type of performance was to be expected for all of the aforementioned reasons we listed.

To sum them up:

The data from the original expert dataset does not include information on how the trees health was assessed, which means that we cannot properly determine what type of data is important to keep and it would also be complicated.

The time of the experts assessment does not align with the time most GSV images were taken.

The sliding windows method is prone to be confused by the nature of the data, as the urban environment can contain many different trees in one GSV image.

While the GSV image may have the tree in question aligned in the centre, because those images are taken from the road the tree might be in a unfavourable position.

The six classes that are originally given by the expert are vague, since a trees health can be reasonable or mediocre based on a combination of very specific attributes and the experts own intuition which is very difficult for a Deep Learning model to capture as it is not deterministic.

All in all, based on the results of our implementation we can believe that using the papers pipeline for the problem of tree health classification is not a good approach. While for problems where our classification objects are smooth, take up the whole image and are invariant to time (crop land does not change within a year, unlike the health or trees), GSV images can be a reliable source of data, that revolutionizes how those problems are approached. For the case of tree health, this unfortunately is not the case as for the current data we cannot determine the features that are crucial for each trees health. A possible solution would be that with the help of the expert that made the assessment we can adjust the data collection method by taking specific pictures of the different parts of the tree that is tested, which might not be possible due to its distance and position towards the vehicle taking the GSV images, along with the resolution that the GSV images offer. We would need a very strict filter on what data is actually usable for us, which then it would consequently restrict the models capability on predicting the health of new trees by using GSV images, since they would also suffer from the same problems.

References:

[1] Tiwari, Tanya & Tiwari, Tanuj & Tiwari, Sanjay. (2018). How Artificial Intelligence, Machine Learning and Deep Learning are Radically Different?. International Journal of Advanced Research in Computer Science and Software Engineering. 8. 1. 10.23956/ijarcsse.v8i2.569.

[2] Whang, S.E., Roh, Y., Song, H. *et al.* Data collection and quality challenges in deep learning: a data-centric AI perspective. *The VLDB Journal* **32**, 791–813 (2023). https://doi.org/10.1007/s00778-022-00775-9

[3] Jordi Laguarta Soler and Thomas Friedel and Sherrie Wang (2024). Combining Deep Learning and Street View Imagery to Map Smallholder Crop Types.

[4] PyQt image annotation tool. https://github.com/robertbrada/PyQt-image-annotation-tool

[5] Resnet50 input requirements [5]. https://pytorch.org/hub/pytorch_vision_resnet/