Road Segmentation

Class Project 2

Jan Kokla, Siim Markus Marvet
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—This paper explores and compares various machine learning models for binary segmentation task of satellite images. We compare classical machine learning approaches with convolutional neural networks and transformer architecture along with data augmentations, ensembling and a description of the process for patch predictions. The final solution achieved F1 score of 0.863 with 0.925 accuracy on the test set of AIcrowd.**

I. Introduction

The aim of this paper is to document the process of finding the best performing model for a road segmentation challenge. The goal is to distinguish road from background and do it for 16x16 pixel patches. The paper is structured as follows. First, we describe the data processing and augmentation approaches, followed by model selection, ensembling and hyperparameter tuning. We then cover the results obtained from cross-validating different setups and finally discuss the results and analyse ethical risks associated with the project.

II. Feature Processing

A. Inference Approach

The first decisions must be done regarding the nature of the input data. First, the training (400x400px) and test images (608x608px) having different sizes was solved by linearly scaling the images and their corresponding masks in the training set to match the dimensions of the test set. Another challenge stems from the fact, that we are required to produce predictions for 16x16 pixel patches. We could adjust the architecture of neural networks to take that into account. Alternatively, we can let the model predict for every pixel and based on a threshold come up with a final outcome for every patch. We chose the latter as it proved to be simpler, especially when using high level API-s for training neural networks.

B. Image Augmentation

The limited size of our training dataset - consisting only of 100 images - meant that our model could have a hard time generalizing. In order to make the inference more robust, we built a pipeline to randomly apply different transforms to each image before it was used to train the model. For that we used the Python library Albumentations [1], which simplified the task of transforming a training image and its corresponding ground truth mask in sync when appropriate (e.g for spacial augmentations: rotations, flips) and *vice versa*, transforming only the training image where the mask should not be altered (e.g when applying color shifts). For visualizations of the used transforms please see Figure 1.

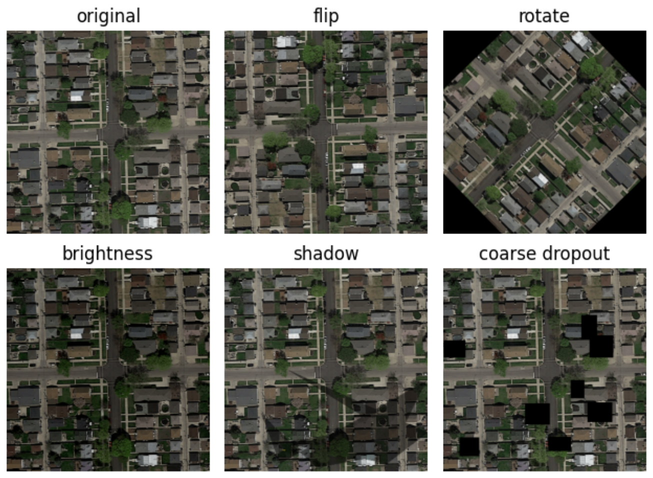


Fig. 1. Transformation visualizations.

It should be noted that our training and test dataset were largely taken from relatively similar urban, suburban and industrial regions with similar housing density, weather conditions and road proportions. This meant that not all transform would be suitable for the dataset (e.g zooming in on one area of the image would alter the proportions of the roads) and this was our main consideration when choosing which transformations to test.

III. Modelling

Considering the size of the dataset and the nature of the task, we decided to use approaches from 3 distinct domains: (1) classical machine learning, (2) convolutional neural networks (CNN), (3) transformers.

A. Baseline Models

To obtain a baseline estimation for expected performance, we trained three classical machine learning models with 3-fold cross-validation: LogisticRegression and GradientBoostingClassifier from the scikit-learn [2] and finally XGBoost [3] using the Flaml [4]. Whereas the first is a simple linear regression model, the last two use a learning method based on tree boosting. Feature extraction was done as color averaging with variance over a 16x16 pixel area, adapted from the EPFL CS-433 course materials [5].

B. Convolutional Neural Networks

CNNs have shown notable performance across various tasks, including image segmentation, and have proven effective in the analysis of satellite imagery among other applications

[6]. The key components of a CNN include the encoder and decoder. Encoder is responsible of generating the features through convolutional layers, pooling operations, and non-linear activation functions. Decoder, on the other hand, must transform the features to the desired output, which in our case meant segmenting the pixels into $\{0, 1\}$.

TABLE I
THE LIST OF CHOSEN ARCHITECTURES ALONGSIDE SHORT DESCRIPTION

Decoder	**Description**
Unet [7]	Designed for semantic segmentation tasks, having a U-shaped structure with contracting and expanding paths for effective feature extraction.
Unet++ [8]	Enhances the U-Net by incorporating nested skip connections, improving information flow and spatial context in segmentation applications.
MAnet [9]	Incorporates attention mechanisms to focus on relevant image regions, improving the precision of semantic segmentation.
PSPNet [10]	Uses different pooling sizes to capture multi-scale contextual information, enhancing the network's ability to understand diverse spatial contexts.
DeepLabV3 [11]	Uses dilated convolutions, which allows to capture information at different scales without increasing the complexity of the model.
DeepLabV3+ [12]	Extends DeepLabv3 by adding an effective decoder to refine the segmentation results.

Considering the limited amount of data, computational resources and time constraints, we decided to use transfer learning, which entails using pre-trained encoder with a decoder fine-tuned for the specific task. It leverages the knowledge learned from a pre-trained model to solve a new, but related problem. Therefore, we use different decoders in combination with various pre-trained encoders. For that purpose, high level API for Pytorch, *Segmentation Models* [13] is used.

As the number of encoders we aim to benchmark is higher than the one of decoders, we first fix the encoder (ResNet) and only focus on the performance of the selected decoders. This will help us to understand how sensitive the overall model performance is to changes in the decoder. The list and short description of the decoders can be found from Table I.

TABLE II
THE LIST OF CHOSEN ENCODERS ALONGSIDE SHORT DESCRIPTION

Encoder	**Description**
ResNet [14]	Introduces skip connections to ease training and mitigating the vanishing gradient problem.
ResNeXt [15]	Enhances ResNet by introducing a cardinality parameter, enabling more diverse feature representations and reducing the number of hyperparameters.
VGG [16]	Classical CNN with deep layer stacking in order to increase the model performance.
EfficientNet [17]	Optimizes model efficiency by balancing network depth, width, and resolution through compound scaling, having good performance with fewer parameters.
Inception [18]	Employs parallel operations with different filter sizes to increase the depth and width of the network while keeping the computational budget constant.

After finding the best performing decoders, we then focus on the performance of different encoder-decoder architectures. The encoders included in the comparison with the reasoning can be found from Table II.

C. Transformers

In addition to natural language processing, Transformer architecture has shown good performance on segmentation tasks [19]. We decided to use SegFormer framework [20] that unifies Transformer architecture with lightweight multilayer perception (MLP) decoder. The API mentioned above conveniently offered the pre-trained version of the encoder that we use together with Unet decoder.

D. Ensembling

To further improve the results, we use ensembling that has shown good results on segmentation tasks in other areas [21]. Although simple neural-networks or regression models can be used for merging the predictions of the several models, we decided to use a simpler approach, namely majority voting. We take the binary outcomes of the models and use the majority rule for deciding the outcome of every pixel, which will then be used for coming up with the prediction for every patch.

E. Hyperparameter Tuning

After finding the best image transformations and encoder-decoder combinations, we will tune the hyperparameters of all the models included in the ensembling. For that purpose, a Python library developed by Microsoft Research, Flaml [4], is used. Flaml was chosen as it offers convenient API along the efficient optimization algorithm [22]. In addition to batch size, learning rate and number of epochs, we experimented with two different loss functions. Dice loss can handle well the class imbalance for foreground and background and is often used for segmentation. Alternatively, focal loss gives more priority to hard examples during the training of the network and is an extension of cross entropy [23]. Therefore, these two loss functions are also included in the hyperparameter tuning.

IV. RESULTS

A. Image Augmentation

Due to the vast computational cost of training the CNN-s, we were not able to perform a full grid search over all possible combinations of transformations. Instead we opted to run a search over a smaller subset of candidates and use our domain knowledge in choosing the specific transformations to test.

The results outlined in Table III show the mean pixel-based F1 score of the best performing k-fold epoch along with the corresponding standard deviation calculated over the k-folds[1]. The transforms were compared using 3-fold cross validation using pre-trained ResNet (50M parameters, trained on ImageNet) with DeepLabV3+ as the decoder (optimizer: Adam with lr=0.0005, loss function: Dice Loss, scheduler: CosineAnnealingLR, batch size: 4, epochs: 100).[2]

[1]This is the metric reported in all other experiments if not stated otherwise.
[2]Notebook with the benchmark of transformations can be found here.

TABLE III
K-FOLD (MEAN) RESULT OF BEST EPOCH

transform	F1
none	0.831 ± 0.014
flip	0.844 ± 0.014
rotate	0.851 ± 0.014
brightness	0.837 ± 0.006
shadow	0.836 ± 0.011
coarse dropout	0.840 ± 0.005
flip, rotate	0.845 ± 0.016
flip, brightness	0.850 ± 0.015
flip, shadow	0.848 ± 0.017
flip, coarse dropout	0.843 ± 0.010
rotate, brightness	**0.855 ± 0.008**
rotate, shadow	0.850 ± 0.013
rotate, coarse dropout	0.852 ± 0.010
brightness, shadow	0.832 ± 0.009
brightness, coarse dropout	0.839 ± 0.012
shadow, coarse dropout	0.832 ± 0.014

The results illustrate that when used individually the largest improvements can be achieved by applying the spacial transforms (rotations and flips) as well as coarse dropout. Adjusting the brightness or adding shadows showed less of an improvement over the baseline, but yielded a marginally better F1 score nevertheless. When comparing the results of transform-pairs we surprisingly observed no significant improvement over the best results obtained by using only one of the transforms with rotate+brightness barely outperforming rotate alone.

B. Modelling

Our baseline models achieved F1 scores between 0.460-0.507. However in this case it is also important to note the accuracies and true positive rates (TPR) of the models in Table IV. The LogisticRegression (C=1e5[3], class_weight=balanced) was overly eager in labelling pixels as road resulting a higher TPR and lower accuracy, whereas the XGBoost and GradientBoostingClassifier (n_estimators=50, learning_rate=1.0, max_depth=10) developed a bias towards predicting the dominant non-road class, leading to the opposite trend in TPR and accuracy.[4]

TABLE IV
BASELINE MODEL PERFORMANCES

Model	F1	Accuracy	True Positive Rate
LogisticRegression	0.460	0.588	0.176
GradientBoostingClassifier	0.479	0.745	0.117
XGBoost	0.507	0.792	0.107

As classical machine learning models did not perform too well, we turned to CNN-s compared different decoders. We fixed encoder to ResNet (21M parameters trained on ImageNet) and used 5-fold cross validation (optimizer: Adam with lr=0.0005, loss function: Dice Loss, scheduler: CosineAnnealingLR, batch size: 4, epochs: 150, augmentations: flip, rotate, brightness, snow). As the results in Figure 3 show, the F1 scores of the best epochs are within 2 percentage points and the standard deviations of the models overlap, indicating that there is no statistically significant difference in variability among the models. Thus, given the dataset and fixed features, the choice of the decoder has not too important role in the overall performance. However, Unet++ and DeepLabV3 have the highest mean and, therefore, will be used in comparing different encoder-decoder combinations.

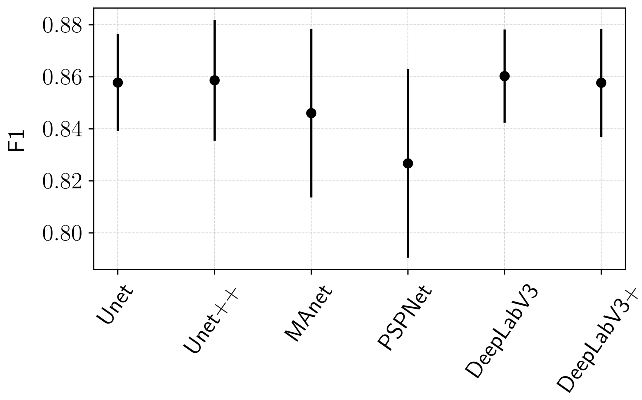


Fig. 2. Comparison of the best mean F1 scores of the epoch with standard deviation.

The encoder-decoder combinations were compared similarly as before: using cross-validation with same parameters as when comparing decoders. To limit the training costs, we confined with 3-fold cross validation. The choice of the encoder-decoder combinations was determined by two limiting factors. Firstly, as high-level API was used and some of the architectures applied dilation to increase the receptive field [24], it made some combinations of encoders and decoders unfeasible. What is more, as the choice of the decoder does not play significant role in the performance as previously shown and to limit the training costs, all encoders were used in combination with only one decoder.[5]

The results in Figure 3 show that even the performance of the encoder-decoder architectures are not too different from each other and are within the standard deviations. Inception with Unet++ as the decoder has the highest mean F1 score. Interestingly, EfficientNet with its limited number of parameters manages to be competitive with more complex models, having the second highest result, closely followed by ResNeXt. The only transformer architecture used (SegFormer) did not outperform CNN-s and had moderate performance while only having higher F1 than ResNet and VGG.

C. Ensembling

Although we compared 6 different architectures in the previous paragraph, we were forced to limit the ensembling into 3 models as this was the reasonable amount that the GPU-s available for us were capable of handling. While taking

[3]Refers to the inverse of regularization strength.
[4]Notebook with the comparison of base models can be found here.

[5]Notebook with the comparison of decoders can be found here and benchmark of architectures here.

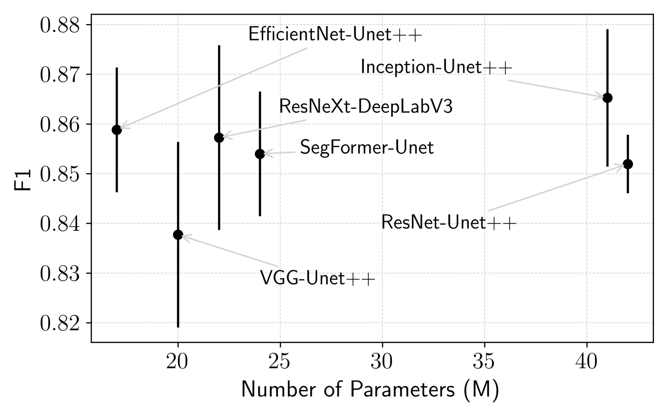


Fig. 3. Comparison of the best mean F1 scores of the epoch with standard deviation.

into account the performance of the models and keeping in mind that ensembling performs best with the least correlated models, we confined with EfficientNet-Unet++ (smallest model), SegFormer-Unet (Transformer architecture) and Inception-Unet++ (best performing).

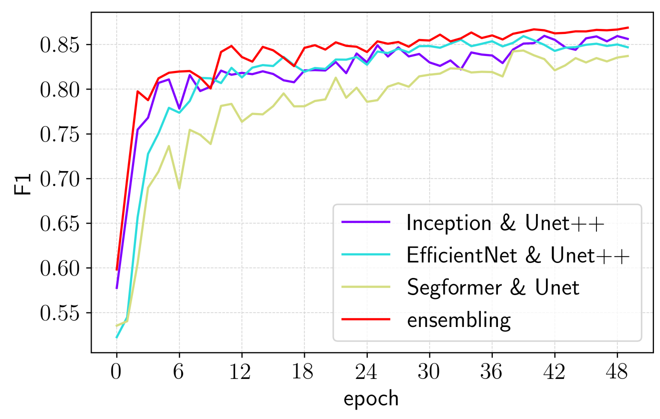


Fig. 4. Comparison of the validation mean F1 scores per epoch.

We ran 3-fold cross-validation with those 3 models and compared the mean epoch performance with ensembled result (optimizer: Adam, loss function: Dice Loss, scheduler: CosineAnnealingLR, batch size: 4, epochs: 50, augmentations: rotate, brightness). We limited the epochs to 50 as the aim is to simply compare the performance of the ensembling with separate models for every epoch. The results in Figure 4 show that ensembling manages to outperform all the models separately.[6]

D. Hyperparameter Tuning

Hyperparameter tuning was run for all 3 best performing architectures used in ensembling (EfficientNet-Unet++, SegFormer-Unet, Inception-Unet++) with best performing augmentations (rotation and brightness). Interestingly, EfficientNet-Unet++ (epochs: 80, learning rate: 0.00085, batch size: 3, criterion: dice loss) and SegFormer-Unet (epochs: 147, learning rate: 0.0008, batch size: 3, criterion: dice loss) resulted with notably higher learning rate than Inception-Unet++ (epochs: 150, learning rate: 0.0001, batch size: 5, criterion: focal loss). What is more, Inception-Unet++ showed better results with focal loss, the other two with dice loss. As two of the three models ended with epoch count close to the set maximum (150), then it might be that further training would marginally improve the results. With the best set of hyperparameters, we could now train the models on full training set for the final inference.[7]

E. ”Patch Problem”

There are two parameters that we have left so far unnoticed, but that could improve the final result. These are (1) classification threshold for sigmoid values when classifying pixels and (2) ”foreground threshold” that is used to decide whether the 16x16 pixel patch belongs to the fore- or background. While they were 0.5 and 0.25 (25% of pixels in the patch must be road to qualify the patch as ”road”) during the experiments, we evaluate the results with different thresholds. There is no hard science here because there is no information about the relation between the pixel and patch values. Or in other words, we are provided with ground truth for every pixel, but not for patches. The visual analysis resulted with sigmoid threshold of 0.1 and foreground threshold of 0.2, which is really aggressive towards classifying patches as road. That gave the final AI Crowd test F1 score of 0.863 with 0.925 accuracy. What is more, we experimented with different post-processing techniques, which did not improve the results and will not be discussed further.[8]

V. Discussion & Summary

The aim of this paper was to describe the process for finding the best performing model for road segmentation. We experimented with different image augmentation techniques, tried several combinations of encoder-decoder pairs and compared separate models with their ensembled version. The experiments led us to the ensembled solution with three CNN-s (EfficientNet-Unet++, SegFormer-Unet, Inception-Unet++) for which best hyperparameters were found and final training on the full dataset performed.

This combination showed 0.863 F1 score in AI Crowd, which is not among the top teams. There could be several reasons for this, starting from the approach we took with predicting every pixel and feeding quite big (608x608 px) images into neural network. Additionally, we could have experimented with more aggressive data augmentations as we had very small dataset. Although, we benchmarked most of the popular CNN-s used for segmentation, we included only 3 models into ensembling analysis. Furthermore, different architectures prefer different input sizes, which could be another avenue for future research.

[6]Notebook with the ensembling results can be found here.

[7]Hyperparameter tuning can be found here and final training here.

[8]The experiments with post-processing can be found here.

VI. Ethical Risks

Our goal with this project was to train a prediction model on a Google Maps dataset provided through the AICrowd platform [25]. The dataset we used is already in public domain and satellite images could be freely obtained. We assess such images to be distant enough that they do not impose a risk on the privacy of individuals nor require a special framework for empowerment on an individual level. As the road segmentation could be applied in disaster management for understanding the impact of the natural disasters (how many roads were impacted) and acting upon it, it could influence peoples lives. Additionally, we have determined multiple aspects of how the training and use of the model could lead to ethical risks.

The provided satellite images depict an urban or suburban environment from seemingly similar region(s). This could constrain the ability of the model to generalize its predictions in more diverse regions, which in turn raises an ethical concern around the fairness of the model. It will likely perform better in similar urban regions as was the source for the training dataset. The issue could be amended by introducing a larger and more diverse training dataset, however, as the goal of this project is to predict on a test set of images from a similar region, such actions were not needed nor practical.

Another issue with our project is the carbon and water-usage footprint of training such complex models. We have aimed to lessen that impact by applying transfer learning techniques and building on top of pre-trained models, that require less resources to be optimized for our use case.

Finally we see possible issues with misuse of the model. The model differentiates only between pixels that are road or background and does that with a margin of error. It does not however do more complex tasks such as deciding whether an image depicts a level intersection or one road passing over another on a bridge. Mistakenly using that model in more complex task like self-driving could lead to accidents with serious consequences. Another thing to note is that since satellite images could originate from different countries, it is important to keep local laws and regulation (e.g relating to objects of national security) in mind when using this model.

For the full ethical risk canvas, created based on the EPFL DigitalEthics Canvas [26], please refer to appendix A.

References

[1] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. ACM, Aug. 2016. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785

[4] C. Wang, Q. Wu, M. Weimer, and E. Zhu, "Flaml: A fast and lightweight automl library," in *MLSys*, 2021.

[5] "EPFL Machine Learning Course CS-433, segment_aerial_images.ipynb," 2023. [Online]. Available: https://github.com/epfml/ML_course/blob/master/projects/project2/project_road_segmentation/segment_aerial_images.ipynb

[6] A. Albert, J. Kaur, and M. Gonzalez, "Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale," 2017.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[8] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," *CoRR*, vol. abs/1807.10165, 2018. [Online]. Available: http://arxiv.org/abs/1807.10165

[9] T. Fan, G. Wang, Y. Li, and H. Wang, "Ma-net: A multi-scale attention network for liver and tumor segmentation," *IEEE Access*, vol. 8, pp. 179 656–179 665, 2020.

[10] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *CoRR*, vol. abs/1612.01105, 2016. [Online]. Available: http://arxiv.org/abs/1612.01105

[11] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[12] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611

[13] P. Iakubovskii, "Segmentation models pytorch," https://github.com/qubvel/segmentation_models.pytorch, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[15] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *CoRR*, vol. abs/1611.05431, 2016. [Online]. Available: http://arxiv.org/abs/1611.05431

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[17] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[20] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Álvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *CoRR*, vol. abs/2105.15203, 2021. [Online]. Available: https://arxiv.org/abs/2105.15203

[21] T. Dang, T. T. Nguyen, J. McCall, E. Elyan, and C. F. Moreno-García, "Two layer ensemble of deep learning models for medical image segmentation," *CoRR*, vol. abs/2104.04809, 2021. [Online]. Available: https://arxiv.org/abs/2104.04809

[22] Q. Wu, C. Wang, and S. Huang, "Frugal optimization for cost-related hyperparameters," in *AAAI*, 2021.

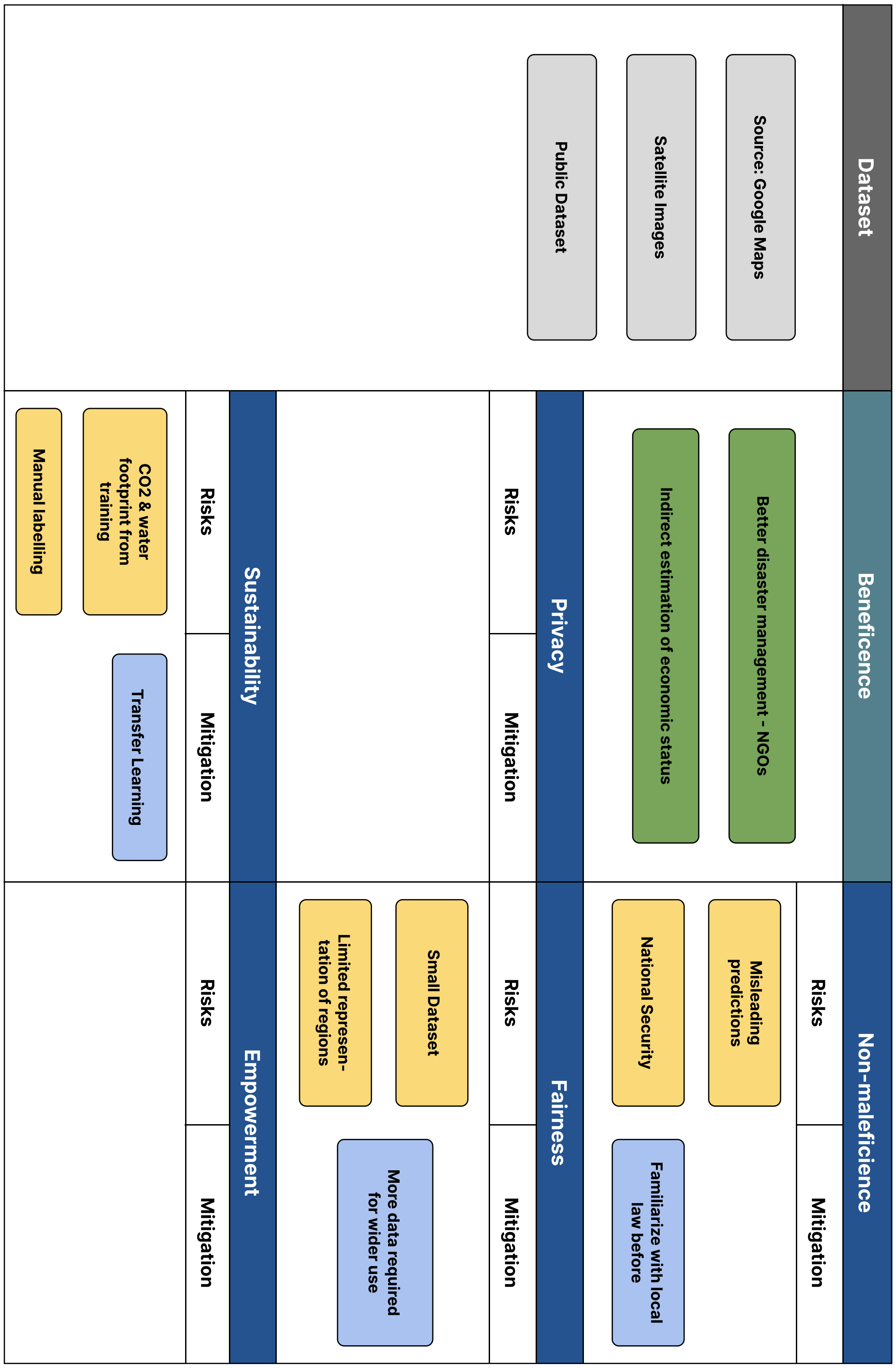
[23] V. Rajput, "Robustness of different loss functions and their impact on networks learning capability," 2021.

[24] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2016.

[25] "AICrowd — EPFL Road Segmentation Challenge," 2023. [Online]. Available: https://www.aicrowd.com/challenges/epfl-ml-road-segmentation

[26] "EPFL Digital Ethics Canvas," 2023. [Online]. Available: https://www.epfl.ch/education/educational-initiatives/cede/training-and-support/digital-ethics/a-visual-tool-for-assessing-ethical-risks/the-digital-ethics-canvas-how-to/

Appendix A
Digital Ethics Canvas



Dataset
Source: Google Maps
Satellite Images
Public Dataset
Beneficence
Better disaster management - NGOs
Indirect estimation of economic status
Non-maleficience
Risks
Mitigation
Misleading predictions
National Security
Familiarize with local law before
Privacy
Risks
Mitigation
Fairness
Risks
Mitigation
Small Dataset
Limited represen-tation of regions
More data required for wider use
Sustainability
Risks
Mitigation
CO2 & water footprint from training
Manual labelling
Transfer Learning
Empowerment
Risks
Mitigation