From pixels to roads: a deep learning approach for road segmentation

Gustave Paul Besacier, Luna Justine Godier Gallay, Agustina María Zein
School of Computer and Communication Sciences, College of Management of Technology
EPF Lausanne, Switzerland

***Abstract*—Accurately identifying roads from satellite images is crucial for urban planning and navigation. Traditional methods often struggle with the complexity of road networks, leaving room for improvement. This study explores the potential of CNNs, U-Nets and Transformer-based models, using a dataset of 100 augmented satellite images. The U-Net model with a ResNet101 encoder stood out, achieving an f1-score of 0.908 and accuracy of 0.957. While the results are encouraging, challenges like limited data and overfitting remain.**

I. Introduction

Precise and automated road segmentation from satellite images is essential for tasks such as modern urban planning, navigation, and disaster response. Relying on manual tracing or simple image-processing rules is time-consuming and often fails to adapt to different landscapes, possibly conveying misleading results that can have important consequences. In recent years, deep learning approaches, like Convolutional Neural Networks (CNNs), have greatly improved performance by learning robust features directly from raw images. Models like U-Net and other fully convolutional architectures have set strong baselines, demonstrating notable accuracy gains in road extraction tasks.
However, CNNs sometimes struggle with capturing long-range context, such as uneven or partially obstructed roads, which is crucial to identify continuous road networks. To address this limitation, researchers have begun turning to Transformer-based models. Using self-attention mechanisms, Transformers can capture global relationships, thus improving continuity and consistency on segmented roads. In this work, we compare manually implemented CNN- and U-Net- and Transformer-based approaches for road segmentation in satellite imagery. We present a methodology for training, evaluating and comparing these models, highlighting their individual strengths and limitations. Ultimately, our goal is to contribute insights that help refine automated road segmentation techniques, making them more accurate, efficient, and useful for real-world challenges.

II. Data processing

The dataset consists of 100 satellite images of dimension 400×400 pixels. Since the dataset was small, some transformations for data augmentation were needed. In the case of satellite images, the transformations that were used represented plausible real-life occurrences, thus aimed at increasing the accuracy of the different models. More precisely, using `torchvision.transforms.functional`, we applied:

- Random rotations of angles of $k\pi$ radians, $k = 1, 2, 3$
- Horizontal and vertical flips
- Color jittering
- Blurring
- RGB channel permutations

These transforms were applied randomly to both the images and the masks to maintain the consistency between them while making sure the color jittering, channel permutations and Gaussian blurring did not affect the groundtruths. For each image, many augmented samples were generated. A variable `SAMPLES_PER_IMAGE` ensured that there were `SAMPLES_PER_IMAGE - 1` augmented versions and the original image in the final training dataset. This allowed the models trained on this augmented dataset to become robust to noise in the input and invariant to small perturbations, such as changes in orientation, color, or intensity, while still performing well on the original images. This pre-processing measure is a fundamental step in our machine learning pipeline. It enhances the robustness of the learned representations and helps prevent overfitting, ultimately contributing to stronger, more reliable model performance.

We also experimented with bootstrapping our dataset to artificially augment its size and while reducing the computational cost of computation. However, considering the limited amount of data that was provided, this method proved to be counterproductive with our models, which ended up learning the patterns "by core". To that end, we eventually implemented the so-called *mixup* method [1], which creates new data by combining existing one. It has been shown to reduce data memorization and increase robustness against adversarial examples. Using tensorized data points x_i and x_j and their respective labels y_i and y_j, we draw $\lambda \sim \text{Beta}(\alpha, \alpha)$, $\alpha = 0.8$ and create a new data input $\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j$ and $\tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j$.

III. Models and methods

Convolutional Neural Network (CNN): CNNs are deep learning models that learn features from images, such as edges and textures, through convolutional layers that apply filters to extract patterns, making them ideal for the problem at hand.
The proposed CNN is designed for road segmentation classification and includes several components. The architecture

begins with a series of encoder blocks that progressively increase the number of channels, allowing the CNN to extract more and more abstract features from the input each time. Each encoder block is made of two convolutional layers with batch normalization and ReLU activation, ensuring stable training and effective feature learning. Max pooling layers between encoder blocks reduce the spatial dimensions, focusing on the most important details.

In the decoder part, transposed convolutional layers perform upsampling, followed by concatenation with skip connections from the corresponding encoder blocks. This allows for the preservation of spatial information and facilitates the reconstruction of features. The concatenated features are processed through additional convolutional layers to refine the output. (For more details, see Appendix A)

Finally, a 1x1 convolutional layer serves as the output layer head, mapping the features to the desired number of output channels. The combination of skip connections and hierarchical feature extraction makes the architecture particularly well-suited for this segmentation task, as it balances spatial accuracy and contextual understanding.

Transformers: Transformers architecture, introduced in 2017 by Vaswani et al. [2] and widely used for NLP tasks, relies on a self-attention mechanism that allows different heads of the model to each focus on different parts of the input and hence better understand relationships between elements regardless of their distance. This global context understanding makes transformers highly effective for tasks requiring detailed, long-range dependencies. Unlike convolutional networks, transformers do not rely on fixed-size filters, enabling them to model relationships across the entire input. A subset of the Transformers architecture is the Vision Transformers (ViT), first introduced by Dosovitskiy et al. (2020) [3], that classifies input images to categories. They leverage the sequential power of the Transformers power by patching original images to 16×16 pixel sub-images, then sequentially feed into a Transformers model. In our work, we lever previously built models to favor running speed, performance, and reduce potential code issues. First, we adapted a Linformer model [4]. This architecture is derived from transformers but bridges its complexity issues. It reduces the self-attention complexity by using an approximation relying on low-rank matrices. We configured it to have 16 heads, and dropout of 0.2. We feed the model with our images in the model in its original form, but modify the final classifier. The model outputs a sequence of 625 elements that corresponds to each of the 16×16 pixel patches of the image. We then applied `torch.sigmoid` to turn them into 0 to 1 bounded values. Then, we used the `google/vit-base-patch16-224` ViT model, pretrained on the `ImageNet-21k` and fine-tuned on the one-million images dataset `ImageNet` [5][6]. Here again, we changed the classifier in the same way we did for the Linformer. We then fine-tune once again on our custom dataset. The goal is to lever the previously learned capabilities of edge and shape detection and refine them for our task.

Transfer learning: The `smp.Unet` framework is an implementation of the U-Net segmentation architecture presented in [7] integrated with pre-trained ResNet encoders (down-sampling paths). In this approach, the encoder works as a feature extractor, reducing the spatial dimensions of the input image while progressively capturing specificities. We chose encoders that were pre-trained on `ImageNet`, which is an image database available for free for non-commercial research [8]. As such, they provide strong initialization and facilitate transfer learning, improving both training stability and performance. Once the input passes through the encoder, the U-Net's decoder progressively upsamples the feature maps and uses skip connections from the encoder's earlier layers. The decoder (up-sampling path) chosen in this study is symmetrical to the encoder and is used to reconstruct the spatial dimensions progressively and accurately. The different encoders used were `ResNet18`, `ResNet50`, `ResNet101`, and `ResNet152`, along with hyperparameter fine-tuning.

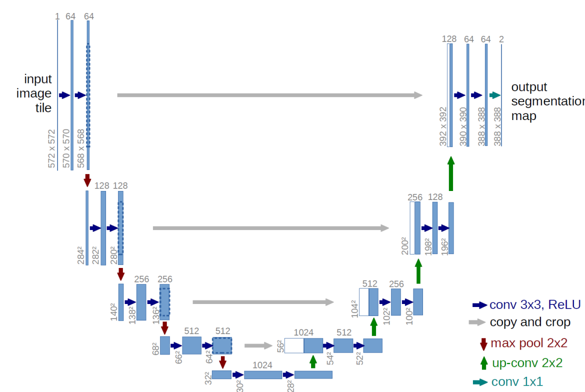

Figure 1: U-Net architecture [9]

IV. Training

The models have been trained in PyTorch (version 2.5.1) using `CUDA` on an NVIDIA GeForce RTX 2070 with Max-Q design and using the `MPS` backend on Apple Silicon M3 Pro and M3 Max. We experienced with different learning rates and learning rate schedulers, namely the cosine and the step scheduler. We also tried different combinations with data augmentation, using the `SAMPLES_PER_IMAGE` variable. Usual values used were `SAMPLES_PER_IMAGE` $= 1, 2, 5$, and this allowed us to fine-tune the different models to find the optimal combination of parameters. Finally, to test the models, we performed inference of the testing images. We used non-random quarter-circle rotations of each image and evaluated the model on each of these transformed inputs. Then, with the output mask, we applied the inverse transform and averaged out the results obtained for the same original image. This allowed us to increase the accuracy and stability of the testing predictions.[1]

V. Results

We first started training the Transformers models, as part of our pipeline. The results are not conclusive at all as they present extremely poor accuracy and f1-score, as per Table I. This can be explained by our difficulties to adapt these two classification models to a larger dimensionality task. This, combined with the important computation time and resources bring us to the other models.

Model	**Loss**	**Accuracy**	**f1-score**
Linformer (*)	0.731	0.351	0.502
Pretrained ViT (*)	0.720	0.514	0.365

Table I: Transformers model training metrics (*) no *mixup* (**) with *mixup*

The respective losses, accuracies, and F1 scores are presented in the following table.

Model	**Loss**	**Accuracy**	**f1-score**
FullCNN	0.538	0.934	0.821
U-Net (ResNet 101)	0.348	0.957	0.908

Table II: Summary of training results

As expected, our built-from-scratch CNN is less accurate than the pre-trained U-Net, although the results are quite promising. This shows that the state of our dataset does not allow us to leverage the full predictive power of U-Net, probably due to a lack of diversity in our images. The different training accuracies and f1-scores obtained using the various `ResNet` encoders are presented in Table IV in the Annex.

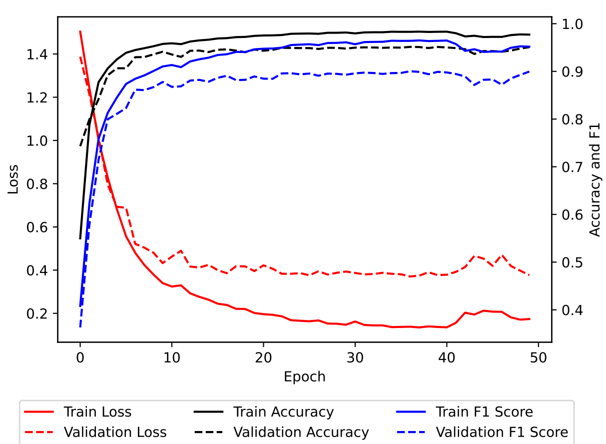

Figure 2: Evaluation metrics for U-Net training as a function of epoch

[1]For computation-time saving purposes, we implemented this inference on the tests done to generate the submissions only.

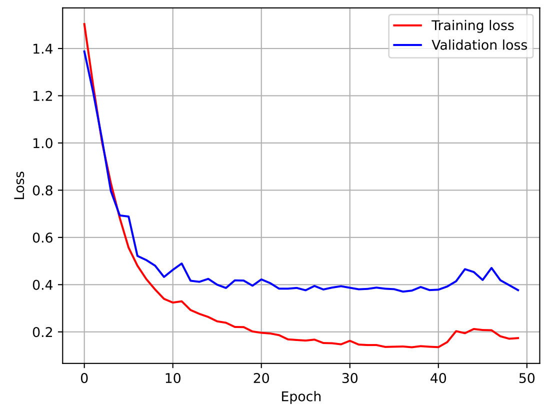

Figure 3: Losses for U-Net training as a function of epoch

The model that overall performs best is the U-Net model using the `ResNet101` encoder, trained during 75 epochs with a batch size of 16. The encoder and decoder learning rates used were respectively 10^{-4} and 10^{-3}. This choice was made purposefully, considering that the encoder is pre-trained and can thus use a smaller learning rate, while the decoder is not. Finally, the segmentation head learning rate was 10^{-3}. Even though we experimented with various learning rate schedulers, this result was obtained without any, and without data mixup.

VI. Discussion

The presented results are promising but allow for improvement. One type of improvement could be leveraging the use of knowledge distillation (KD), using as a teacher model the U-Net-based architecture and as students our manually implemented CNN, that in turn requires to be deeper. Thus, instead of training the student solely on hard, one-hot labels, we guide it with the U-Net's softer probability distributions. Another path that could be explored is the fine-tuning of the decoder block of the U-Net model. Regarding the results, it is important to note that, while the results were satisfying and showed promise, the models used are very large and can thus, due to the small size of our dataset, lead to overfitting, even with the multiple augmentation techniques implemented. This can indeed be observed locally when comparing training and testing f1-scores, which differ by around 10% at the end of the training. One remedy to that issue could be to augment our dataset with external images taken on other datasets, thus tangibly increasing the size of our ”base” available information. However, the computational resources such hypothetical training requires could be a further issue.
Finally, there is an environmental challenge regarding this project. When some computations were too heavy to be run on our computers, we ran on EPFL Izar clusters, and as per

Figure 4, we can see the CO2 consumption of it is enormous. It requires the scientists to use the resources mindfully.

USERNAME : **gbesacie**
Global usage from 2024-12-01 to 2024-12-31

Account	Cluster	# jobs	GPU [h]	CPU [h]	eCO_2 [kg]	Costs [CHF]
ee-559	izar	1	11.8	0.0	0.5	0.0

Figure 4: Computational resources (Transformers)

VII. Ethical Discussion

In order to assess the ethical aspects of our solutions, we used the Digital Canvas developed by EPFL [10] (see Appendix C))

One potential ethical issue with our road segmentation model is a consequence of the characteristics of the dataset and concerns fairness. Most of the images, as well as the labels that we have used, were collected in urban environments, namely, densely built road networks, clearly defined structures, and organized layouts. Thus, the model may work very well in cities or developed areas, able to distinguish between roads and background. However, it may perform poorly in rural or remote areas. In the contexts of the less developed areas, the roads may be mislabeled or even missed while being recognized as other types of landscapes. Such uneven accuracy leads to imbalanced benefits and may negatively impact the communities that do not correspond to the urban environments depicted in the data. In the two figures below, one can compare two satellite images. The first one is of Echandens, and the second one is in Morges. Both are located in the Vaud Canton in Switzerland. As we can see, the colors, structures, and information contained vary significantly and are more than likely to cause inaccuracies when using the model. In order to fix this, the dataset would need to gain in diversity, ensuring a balanced presence of rural and urban samples.

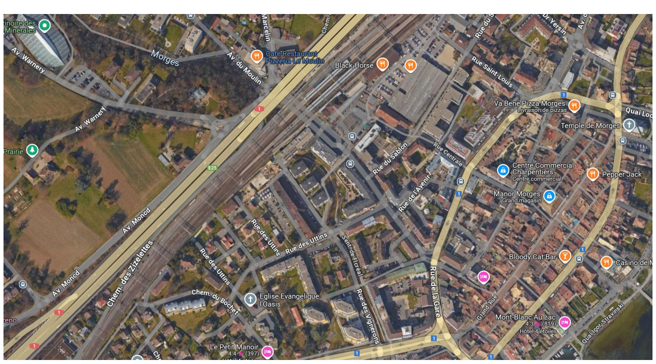

Figure 5: Urban landscape, Morges, VD (Google Image)

Figure 6: Rural landscape, Echandens, VD (Google Image)

Another situation in which our model could prove itself unethical is the case of remote communities, for instance, the Sentinelese people of North Sentinel Island. They are known for isolating themselves and refusing any form of contact with the outside world. A tool like ours, if applied wrongly or forced onto such communities, may be perceived as invasive and rude. High-resolution maps generated in an automatic manner of their territory could raise the interest of researchers, tourists, or even vicious groups. Thus, the community's independence, its cultural identity, and its safety may be at risk due to the availability of proper road or path data.

In addition, although our system is designed to improve the planning of public transport or scientific research, it may also contribute to the development of situations that should not be allowed in certain areas. The indigenous people or isolated communities may experience increased pressure if their land becomes easier to traverse and develop due to better mapping. It is important to find a way of promoting the improvement of infrastructure planning or environmental study without violating the rights of the indigenous people and destroying their culture. A solution to these biases and sensitivities may involve setting low data resolution, engaging the local community in decision-making, and imposing certain conditions for using our tool in culturally or environmentally significant zones.

VIII. Summary

This work explored automated road segmentation from satellite imagery, comparing CNN-based architectures (e.g., U-Net) and transformer-based models (e.g., Linformer, ViT). The dataset, consisting of 100 satellite images of 400×400 pixels, went through a diverse augmentation pipeline, including random rotations to improve model robustness.

The best-performing model was the U-Net with a ResNet101 encoder, achieving locally on the testing sample an f1-score of 0.908 and an accuracy of 0.957, and is linked to AICrowd's submission 277201, yielding an f1-score of 0.889 and an accuracy of 0.937.

Transformer models demonstrated potential in capturing global relationships but underperformed compared to CNNs in this task, likely due to the small dataset size.

Challenges included overfitting, dataset imbalance, and the computational resources required for training. Ethical concerns were also addressed, emphasizing potential biases against rural areas and the ethical implications of using satellite data in sensitive regions.

Future work will focus on expanding the dataset, leveraging techniques like knowledge distillation, and addressing ethical concerns to enhance fairness and applicability across diverse environments.

References

[1] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2018. [Online]. Available: https://arxiv.org/abs/1710.09412

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[3] 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. [Online]. Available: https://arxiv.org/abs/2010.11929

[4] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," 2020. [Online]. Available: https://arxiv.org/abs/2006.04768

[5] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, and P. Vajda, "Visual transformers: Token-based image representation and processing for computer vision," 2020.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[8] ImageNet, "Imagenet: A large scale hierarchical image database," https://www.image-net.org, 2024, accessed: 2024-12-19.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 9351. Springer, 2015, pp. 234–241. [Online]. Available: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

[10] École Polytechnique Fédérale de Lausanne (EPFL), "A visual tool for assessing ethical risks," https://www.epfl.ch/education/educational-initiatives/cede/training-and-support/digital-ethics/a-visual-tool-for-assessing-ethical-risks/, accessed: 2024-12-18.

Appendix

A. Model Architecture Details

Layer	**Input Shape**	**Output Shape**	**Details**
Input	$(B, 3, H, W)$	$(B, 3, H, W)$	RGB Input Image
Encoder			
enc_0	$(B, 3, H, W)$	$(B, 64, H, W)$	Double Conv (3x3)
Pooling	$(B, 64, H, W)$	$(B, 64, H/2, W/2)$	Max Pool (2x2)
enc_1	$(B, 64, H/2, W/2)$	$(B, 128, H/2, W/2)$	Double Conv (3x3)
Pooling	$(B, 128, H/2, W/2)$	$(B, 128, H/4, W/4)$	Max Pool (2x2)
enc_2	$(B, 128, H/4, W/4)$	$(B, 256, H/4, W/4)$	Double Conv (3x3)
Pooling	$(B, 256, H/4, W/4)$	$(B, 256, H/8, W/8)$	Max Pool (2x2)
enc_3	$(B, 256, H/8, W/8)$	$(B, 512, H/8, W/8)$	Double Conv (3x3)
Pooling	$(B, 512, H/8, W/8)$	$(B, 512, H/16, W/16)$	Max Pool (2x2)
enc_4	$(B, 512, H/16, W/16)$	$(B, 1024, H/16, W/16)$	Double Conv (3x3)
Pooling	$(B, 1024, H/16, W/16)$	$(B, 1024, H/32, W/32)$	Max Pool (2x2)
enc_5	$(B, 1024, H/32, W/32)$	$(B, 2048, H/32, W/32)$	Double Conv (3x3)
Decoder			
upconv_5	$(B, 2048, H/32, W/32)$	$(B, 1024, H/16, W/16)$	Transpose Conv (2x2)
dec_5	$(B, 2048, H/16, W/16)$	$(B, 1024, H/16, W/16)$	Double Conv (3x3)
upconv_4	$(B, 1024, H/16, W/16)$	$(B, 512, H/8, W/8)$	Transpose Conv (2x2)
dec_4	$(B, 1024, H/8, W/8)$	$(B, 512, H/8, W/8)$	Double Conv (3x3)
upconv_3	$(B, 512, H/8, W/8)$	$(B, 256, H/4, W/4)$	Transpose Conv (2x2)
dec_3	$(B, 512, H/4, W/4)$	$(B, 256, H/4, W/4)$	Double Conv (3x3)
upconv_2	$(B, 256, H/4, W/4)$	$(B, 128, H/2, W/2)$	Transpose Conv (2x2)
dec_2	$(B, 256, H/2, W/2)$	$(B, 128, H/2, W/2)$	Double Conv (3x3)
upconv_1	$(B, 128, H/2, W/2)$	$(B, 64, H, W)$	Transpose Conv (2x2)
dec_1	$(B, 128, H, W)$	$(B, 64, H, W)$	Double Conv (3x3)
Segmentation Head	$(B, 64, H, W)$	$(B, 1, H, W)$	1x1 Conv

Table III: Layer-wise Details of the FullCNN Model

For generalization purposes, H, W, and B have been left as variables, but in our case, $H = 400$, $W = 400$, and $B = 16$.

B. *Details for learning - Unet pretrained*

Encoder	***mixup***	**LR**	**Loss**	**Accuracy**	**F1**
resnet18	No	No	0.456	0.942	0.877
resnet18	Yes	No	0.645	0.899	0.829
resnet18	No	Linear	1.437	0.588	0.442
resnet18	Yes	Linear	–	–	–
resnet18	No	Cosine	0.413	0.946	**0.886**
resnet18	Yes	Cosine	0.631	0.910	0.836
resnet50	No	No	0.409	0.948	**0.891**
resnet50	Yes	No	0.606	0.911	0.841
resnet50	No	Linear	1.414	0.610	0.482
resnet50	Yes	Linear	–	–	–
resnet50	No	Cosine	0.404	0.948	0.890
resnet50	Yes	Cosine	0.702	0.897	0.821
resnet101	No	No	0.348	0.957	**0.908**
resnet101	Yes	No	0.538	0.921	0.859
resnet101	No	Linear	1.547	0.454	0.489
resnet101	Yes	Linear	–	–	–
resnet101	No	Cosine	0.350	0.954	0.906
resnet101	Yes	Cosine	0.502	0.930	0.866
resnet152	No	No	0.386	0.953	**0.901**
resnet152	Yes	No	0.611	0.922	0.852
resnet152	No	Linear	1.351	0.712	0.451
resnet152	Yes	Linear	–	–	–
resnet152	No	Cosine	0.386	0.952	0.899
resnet152	Yes	Cosine	0.628	0.912	0.843

Table IV: `ResNet` model fine tuning metrics (on test set)*

*75 epochs – linear LR: step $= 1$, $\gamma = 0.5$ – cosine: 15% warmup and 10% restart.

Encoder	**Data**	**LR**	**Loss**	**Accuracy**	**F1**
resnet18	2	No	0.465	0.947	0.890
resnet18	2	Cosine	0.480	0.942	0.878
resnet18	5	No	0.491	0.945	0.882
resnet18	5	Cosine	0.502	0.944	0.882
resnet101	2	No	0.419	0.950	0.895
resnet101	2	Cosine	0.398	0.948	0.892
resnet101	5	No	0.433	0.951	0.897
resnet101	5	Cosine	0.447	0.950	0.893
resnet152	1	No	0.466	0.947	0.887
resnet152	2	No	0.412	0.950	0.898

Table V: `ResNet` model fine-tuning metrics (on test set)**

** 50 epochs – cosine: 15% warmup and 10% restart - Data: data augmentation multiple.

C. Digital Ethics Canvas

Figure 7: Our filled Digital Ethics Canvas