BiSeSAM for Road Segmentation
CIL ETHZ Road Segmentation Kaggle Challenge 2024

Jannek Ulm Douglas Orsini-Rosenberg Raoul van Doren Paul Ellsiepen

Kaggle Team: Long way to go

Abstract

Vision transformers have recently started to appear on top of vision task leaderboards. We extend ideas and models from cutting-edge vision transformers to present **BiSeSAM** - an efficient and novel road segmentation model. BiSeSAM embeds and extracts the generalized power of Meta's Segment Anything (SAM) image encoder in a custom architecture, with several plug-in decoders. Trained with our manually generated dataset, BiSeSAM achieves on-par performance with state-of-the-art models.

1. Introduction

Since they have been released, CNNs have been the dominating architecture for most Computer Vision Tasks. Recently, transformer models have gained popularity, and have started to out-perform CNNs for classical vision problems. Inspired by this trend, we explore the performance of Vision Transformers for tackling the road segmentation problem, i.e. the binary semantic segmentation of satellite images.

We harness the power of Meta's Segment Anything Model (SAM) [6] which we adapt and extend for this specific problem. This work presents **BiSeSAM**: a novel transformer architecture for road segmentation.

Our contributions are as follows:

1. A new model architecture **BiSeSAM**: SAM image encoder with various custom decoders (MLP, Conv, Spatially Aware, Skip Connection).
2. A custom dataset of >12k images and corresponding groundtruth road masks.
3. Evaluation and baseline comparison for all versions of BiSeSAM.

The main idea of BiSeSAM is to use the weights of a "strong" image encoder (SAM) - trained and purposed for generalized segmentation - as a learning starting point. Subsequently, the SAM image encoder is finetuned and combined with several task-specific decoders, including: MLPs, transposed Convolutions and Skip Connections. We choose SAM as a cutting-edge segmentation model, thereby (indirectly) leveraging its vast dataset and training resources[1].

Our approach yields on-par performance with our implementations of baseline models (UNet,UNet++). As an ensemble with our baselines, BiSeSAM achieves the 2nd place in the CIL Kaggle competition (see section D for submission description). Thus, our work follows the trend of Vision Transformers enabling performance improvements and innovation.

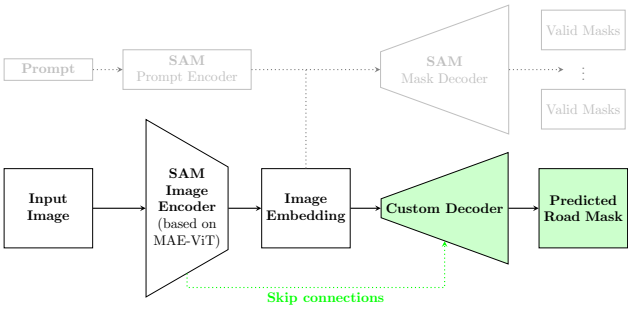

Figure 1. Overview of BiSeSAM *(Original SAM Model architecture in grey, proposed novelties in green)*

2. Related Work

Vision Transformer(s). Recently, several works employ transformers for vision tasks [1–6; 10; 11; 13]. As the classic transformer architecture operates on 1-dimensional sequences [12], a major design challenge is how to present image data to the encoder. The pioneering work, ViT [3], simply appends a positional embedding to a linear projection of image patches[2]. Already being data-hungry by design, vision transformers furthermore observe the difficulty of an extremely irregular information density in most data (including our domain of aerial images), leading to a slower learning process of relevant semantic components [4]. We implicitly address this with our usage of SAM's image encoder which was pre-trained on the large, and model-in-the-loop generated, SA-1B dataset [6].

Segformer. As a specific adjustment for image segmentation, Segformer also uses residual ("skip") connections from several layers (i.e. different focus points) in the encoder as input to the decoder [13]. The ViT paper already identified that early layers in encoder tend to attend locally[3], similarly to early layers in CNN, while higher layers almost exclusively "pay attention" globally. This availability of global and localized information is particularly valuable for im-

[1] SAM has been trained on 256 A100 GPUS for 68 hours [6].

[2] For a visualization, see [3; 4].

[3] The analogous concept to receptive fields in CNN-based architectures. This can be derived from the attention weights at the various layers - with multi-head self-attention, even more data points are available.

age (and road) segmentation - local information is essential for identifying (potential) element boundaries while precise labeling and boundary decisions rely on (more) global information [10].

Our work also makes use of skip connections for some of the custom decoders, while ensuring that sufficient semantic knowledge is represented in the image embedding by using the pre-trained SAM encoder. In contrast to spatial resolution changes and patch merging employed by Segformer [13], the SAM image encoder uses a fixed spatial resolution with global and local information sharing through window attention.

3. BiSeSAM - Model Architecture

3.1. META - Segment Anything Model

SAM is a vision model created by Meta for prompt-based image segmentation and consists of three parts: an image encoder, a prompt encoder and a mask decoder. The image encoder and prompt encoder create embeddings separately. Both are used as an input to the mask decoder, which then produces segmentation masks.

Image Encoder. SAM utilizes a Vision Transformer (ViT) that is based on a masked auto-encoder (MAE) [4] pre-trained model. The encoder first breaks down the input image (1024x1024 pixels with 3 channels) into 64x64 non-overlapping patches (each 16x16 pixels) and then maps each patch to a 768-dimensional embedding space through a convolutional layer. Positional embeddings are added to preserve spatial information. The 12-ViT blocks handle the sequence of patch embeddings. ViT Blocks 2,5,8 and 11 use global attention w.r.t. the other patches, while the other blocks have a window size of 14 [6]. The output of the last transformer block is compressed by a convolutional "neck" layer, producing a final output of shape (64,64,256).

Prompt Encoder. The prompt encoder transforms prompts into embedding vectors. Prompts can be points, boxes and masks.

Mask Decoder. The mask decoder merges the embeddings from the image encoder and prompt encoder to create segmentation masks. It employs a transformer decoder block with self attention and cross-attention mechanisms along with a dynamic mask prediction head that suggests multiple masks for different prompts.

While SAM's mask decoder is powerful for various prompt-based tasks, it requires explicit prompts to generate masks. Since road segmentation only requires road masks, there is no need for a generic mask decoder. Therefore we propose using SAM's ViT image encoder and replacing the prompt encoder + mask decoder with different specialized decoders, as shown in Fig 1. This way, we can still benefit from the feature extraction capabilities of the pre-trained ViT image encoder.

3.2. Decoders

While CNN's have an inherently large inductive bias, transformer architectures need vast amounts of data and compute resources to achieve competitive results [2; 3]. As the SAM mask decoder uses a transformer as a decoder, we refrained from trying to train such an architecture from scratch. Hence we propose and compare a variety of decoders: a standard MLP, a Convolutional Decoder, a MLP with Spatial awareness, and a MLP with skip connections, that uses intermediate results from the image encoder in addition to the final image embedding.

3.2.1. Vanilla MLP Decoder

This decoders uses a simple fully connected multi-layer perceptron (MLP). It consists of 6 layers with LeakyReLU activation functions that maps the ViT's 256-dim image embedding for each 16x16 patch directly to the corresponding patch in the segmentation mask output.

3.2.2. Transposed Convolution Decoder

Additionally, we experimented with a convolutional decoder proposed by [7]. It includes a series of transposed 2d convolution layers (over the 256-channel 64x64 encoded image). Upsampling is performed to increase the spatial resolution of the ViT embeddings while learning to generate segmentation boundaries.

3.2.3. Spatially Aware MLP Decoder

To improve performance, we extend the standard MLP Decoder with enhanced contextual awareness to generate the mask for each path: In addition to the patch currently being decoded, the decoder also receives the 4 neighboring patches as inputs in Spatial-Aware-S, as can be seen in Fig 2. We also implement a full version (Spatial Aware-F), where the decoder incorporates the 8 surrounding patches. We believe this benefits the model's performance since road masks overlap with the neighboring patches, allowing the decoder to generate better transitions between patches.

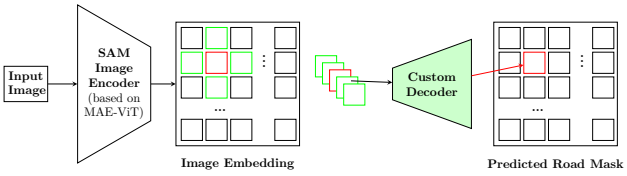

Figure 2. Spatially Aware Decoder

3.2.4. Skip Connection MLP Decoder

For this decoder we employ skip connections for each patch from intermediate states from within the image encoder (756-dim). The intermediate patch representations (between two ViT-blocks) are concatenated together with the final

(256-dim) patch embedding to a patch embedding of form (3x756+256-dim). This richer representation is then used by a patch wise 6-layer leaky ReLu MLP to produce the patch's mask prediction. The skip connections use the outputs of global attention blocks 2,5 and 8 as input. The intermediate embedding states after global attention blocks are represented by green boxes in Fig 3.

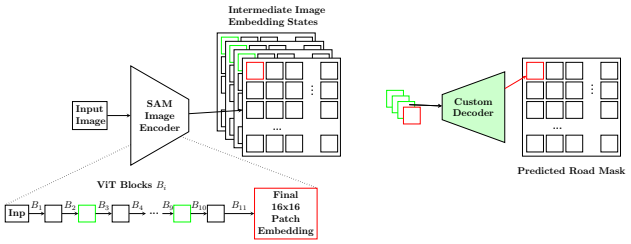

Figure 3. Skip Connection MLP Decoder

3.3. Ensemble

We also used a simple majority vote ensemble that combines the outputs of each BiSeSAM model (differing in the decoder) and the baseline models outputs to predict the final output.

4. Methods

4.1. Data

The Kaggle dataset contains 144 RGB images and their masks. We refer to this dataset as **kaggle**.
To improve the quality of our training we build our own more extensive dataset, using the Google Maps API. We determine the required scale for the images and download an additional 12,990 satellite images and their corresponding road masks from the following U.S. cities: Boston, New York City, Philadelphia, and Austin. Images with less than 3% road pixels were removed. We refer to this secondary dataset with a total of 11500 samples as **gmaps**.
Moreover, we perform data augmentation by randomly rotating the images along their axes. For preprocessing, we rely on SAM's standard preprocessing pipeline, which normalizes pixel values and ensures the inputs are square. All pictures in the dataset have size 400x400. To make them compatible with the SAM image encoder, we upsample to 1024x1024 to train BiSeSAM. Our dataset has a class imbalance of 85% non-road to 15% road pixels. For more details, see Appendix G.

4.2. Training

All models are trained using the Adam Optimizer with a decaying learning rate and a batch size of 5. We train 14 epochs on **gmaps** and 15 finetune epochs on **kaggle**. The layers of BiSeSAM are gradually unfrozen, for more details see Appendix B. Note that due to limited resources we could only afford to tune hyperparameters w.r.t. the loss function and learning rate, and could complete the full training pipeline for each model only once.

4.3. Loss Function

We test various combinations of loss functions including weighted BCE and Dice Loss. All tested functions perform similarly. Further research [14; 8] suggests that the combination $loss = BCE + Dice$ is robust in general, hence, we choose this loss. The convex nature of BCE helps the complex non-convex loss landscape of Dice loss. A quantification of these results can be found in Appendix H.
Furthermore, we explored Focal loss as a promising segmentation loss function [14]. However, it did not yield promising results in preliminary testing. We hypothesize that this is due to the nature of our dataset[4].

4.4. Baselines

As our baseline models we implement the traditional state-of-the-art architectures UNet and UNet++[5]. Both are initialized with two different encoder backbones, `efficientnet-b5` and `resnet34`, resulting in four separate baseline models. As initial model checkpoint, we use pre-trained ImageNet weights. The optimizer and training data is equivalent to BiSeSAM. For more details on training, see C.

5. Results

In the following, we present our findings to compare and evaluate various decoders.

Training all decoders according to the schedule defined in section 4.2, we obtain the results displayed in table 5.
To measure performance we report the mean F1-Score of our algorithms on a validation split of **kaggle**, as well as the public Kaggle F1-submission-score achieved by the algorithm. We compare the baseline implementations (UNet, UNet++) with BiSeSAM in its various decoder configurations.

The obtained results indicate no significant performance difference between the various decoder choices. Overall, BiSeSAM also performs similar to UNet and UNet++, independent of the backbone choice. The performance improvement of BiSeSAM deployed as a majority ensemble is marginal for **kaggle** locally, while yielding better results on

[4]Focal loss performs well for exceptionally unbalanced datasets - in our dataset we observe "only" an imbalance of 85% negatively to 15% positively labeled pixels.

[5]More information about the architecture of UNet and UNet++ can be found in Ronneberger's paper "U-Net: Convolutional Networks for Biomedical Image Segmentation" [9] and Zhou's paper "UNet++: A Nested U-Net Architecture for Medical Image Segmentation" [15].

Model	F1 - Kaggle	F1 - Kaggle Submission
Unet - resnet34	93.20	92.59
Unet++ - resnet34	93.40	92.73
Unet - efficientnet-b5	**94.00**	**93.43**
Unet++ - efficientnet-b5	93.61	93.27
BiSeSAM - Conv	**94.11**	**93.33**
BiSeSAM - MLP	94.06	93.27
BiSeSAM - Spatial Aware-S	94.01	93.21
BiSeSAM - Spatial Aware-F	94.09	93.15
BiSeSAM - Skip Connect	94.07	93.26
BiSeSAMs + Unets - Ens.^{A}	-	**94.22**

Table 1. Model Results, *A: Ensemble of BiSeSAM, Unet, UPP for Kaggle submission described in Section D*

the remote submission. In both evaluations, the ensemble performs better than the baseline models and represents our best-performing submission to the Kaggle competition.

Visualized in Fig 4 and 7, BiSeSAM identifies some road-like structures (paths, parking lots, or driveways) as roads. Most of the segmentation errors (FP and FN) can be attributed to these cases, as the ground-truth for **kaggle** as well as **gmaps** appears inconsistent. See Table F for a quantification of this error analysis.

6. Discussion

In terms of the building blocks of our model, the image encoder appears to be the largest contributor to the performance of BiSeSAM. As shown in table 5, different choices for the decoder do not lead to significant changes in performance. In the scope of this project, we could not complete enough training pipeline iterations to perform a meaningful statistical analysis of the different decoders as their single trial F1 scores range in the same tenth of a percentage point.

BiSeSAM performs at least on-par with state-of-the-art baseline models. As presumed by our architecture choice, this can be attributed to SAM's general image embedding power. However, extensive fine-tuning of the SAM encoder and adjustments to our decoders were necessary for this result: exclusively training the decoders yields poor results (see Table 5).

The performance of UNet and UNet++ seems on-par with our transformer-based approach (i.e. not significantly worse), at least for our training and test data. On the one hand, we attribute this to the established observation that CNN-based approaches perform better with low(er) amounts of data as transformers. The reason for this is that transformers need more training to learn the semantic structure of the input domain, because they have to compensate for their lack of translation equivariance [3] or lack of 2D locality awareness. On the other hand, it is simply expected that state-of-the-art segmentation models perform well on a relatively standard image segmentation task.

Figure 4. Qualitative Analysis. MLP-BiSeSAM on unseen sample from Worcester, MA: TP (Green), FP (Red), FN (Blue)

7. Summary

In this work, we showed that transformers are likely to remain the most promising approach for image segmentation problems:

BiSeSam, our proposed ViT architecture, achieves SOTA performance.

1. BiSeSAM, as a specialized segmentation transformer, with the well-trained, but generalized, SAM image encoder as a starting point, achieves state-of-the-art performance; 2. more architecturally specialized CNN-based approaches in UNet and UNet++ do not out-perform BiSeSAM on the same dataset.

We are convinced that our results do not constitute the limits of the potential of transformers - combining more data and compute, crafting and fine-tuning more specialized encoders, augmenting transformers with sophisticated pre- and post-processing steps, or exploring innovative ideas in the application of 1D performers to 2D domains (such as images) are all promising future advances for binary image segmentation that make an interesting near future for the vision research community.

References

[1] Bao, Y., Sivanandan, S., and Karaletsos, T. Channel vision transformers: An image is worth 1 x 16 x 16 words, 2024. URL https://arxiv.org/abs/2309.16108.

[2] Deininger, L., Stimpel, B., Yuce, A., Abbasi-Sureshjani, S., Schönenberger, S., Ocampo, P., Korski, K., and Gaire, F. A comparative study between vision transformers and cnns in digital pathology, 2022. URL https://arxiv.org/abs/2206.00389.

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[4] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988, 2022. doi: 10.1109/CVPR52688.2022.01553.

[5] Huang, Z., Jin, X., Lu, C., Hou, Q., Cheng, M.-M., Fu, D., Shen, X., and Feng, J. Contrastive masked autoencoders are stronger vision learners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(4):2506–2517, 2024. doi: 10.1109/TPAMI.2023.3336525.

[6] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W., Dollár, P., and Girshick, R. B. Segment anything. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 3992–4003. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00371. URL https://doi.org/10.1109/ICCV51070.2023.00371.

[7] Nalkar, Y. R. *Promptless-TaskSpecific-Finetuning of MetaAI SAM*, 2024. URL https://www.kaggle.com/code/yogendrayatnalkar/promptless-taskspecific-finetuning-of-metaai-sam.

[8] Rajput, V. Robustness of different loss functions and their impact on networks learning capability, 2021. URL https://arxiv.org/abs/2110.08322.

[9] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., III, W. M. W., and Frangi, A. F. (eds.), *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://doi.org/10.1007/978-3-319-24574-4_28.

[10] Strudel, R., Garcia, R., Laptev, I., and Schmid, C. Segmenter: Transformer for semantic segmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7242–7252, 2021. doi: 10.1109/ICCV48922.2021.00717.

[11] Sun, Z., Zhou, W., Ding, C., and Xia, M. Multi-resolution transformer network for building and road segmentation of remote sensing image. *ISPRS Int. J. Geo Inf.*, 11(3):165, 2022. doi: 10.3390/IJGI11030165. URL https://doi.org/10.3390/ijgi11030165.

[12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[13] Xie, E., Wang, W., Yu, Z., Anandkumar, A., Álvarez, J. M., and Luo, P. Segformer: Simple and efficient design for semantic segmentation with transformers. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 12077–12090, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/64f1f27bf1b4ec22924fd0acb550c235-Abstract.html.

[14] Xu, H., He, H., Zhang, Y., Ma, L., and Li, J. A comparative study of loss functions for road segmentation in remotely sensed road datasets. *Int. J. Appl. Earth Obs. Geoinformation*, 116:103159, 2023. doi: 10.1016/J.JAG.2022.103159. URL https://doi.org/10.1016/j.jag.2022.103159.

[15] Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., and Liang, J. Unet++: A nested u-net architecture for medical image segmentation. In Stoyanov, D., Taylor, Z., Carneiro, G., Syeda-Mahmood, T. F., Martel, A. L., Maier-Hein, L., Tavares, J. M. R. S.,

Bradley, A. P., Papa, J. P., Belagiannis, V., Nascimento, J. C., Lu, Z., Conjeti, S., Moradi, M., Greenspan, H., and Madabhushi, A. (eds.), *Deep Learning in Medical Image Analysis - and - Multimodal Learning for Clinical Decision Support - 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings*, volume 11045 of *Lecture Notes in Computer Science*, pp. 3–11. Springer, 2018. doi: 10.1007/978-3-030-00889-5_1. URL https://doi.org/10.1007/978-3-030-00889-5_1.

Appendix

A. Codebase

The code for all the experiments, including data generation, training of baselines and BiSeSAM can be found here:

https://github.com/janulm/eth-cil-road-segmentation

B. Training Procedure - BiSeSAM

All models were trained using the Adam optimizer with parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, no weight decay, a batch size of 5, and the below learning rate and epoch schedule. The training was conducted on a Nvidia RTX 3090 GPU, with each model taking approximately 9 hours to train.

For the image encoder, (A, B) indicates that the first A and last B weights are unfrozen and fine-tuned, the encoder has in total 176 unique parameter "layers", which correspond to 85M parameters. Note that all decoder layers are always trained.

1. 3 epochs with (0, 25) weights unfrozen and a learning rate of 0.001
2. 3 epochs with (0, 65) weights unfrozen and a learning rate of 0.0001
3. 4 epochs with (15, 85) weights unfrozen and learning rates of 0.0001, $\frac{0.0001}{2}$, $\frac{0.0001}{3}$, $\frac{0.0001}{4}$ respectively for each epoch
4. 4 epochs with (25, 105) weights unfrozen and learning rates of 0.0001, $\frac{0.0001}{2}$, $\frac{0.0001}{3}$, $\frac{0.0001}{4}$ respectively for each epoch

B.1. Finetuning

The finetuning process involves training with 80% of the kaggle dataset split for training and 20% for validation. It includes 15 epochs with layers (25-105) unfrozen, using a learning rate of 0.00001.

C. Training Procedure - Baselines

The training was conducted on a Apple M2 Max 38-Core GPU. All of the four models where initialized using the ImageNet weights. The Baseline models were then trained using the Adam optimizer with parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, no weight decay, a batch size of 5, for 10 epochs on the **gmaps** data. After that the models were finetuned for 10 epochs on **kaggle** data.

D. Final Submission

To maximize our kaggle score for the competition we use a majority voting ensemble over all submissions files, generated using all BiSeSAM, Unet and UnetPP models. Our code takes the majority vote on each 16x16 pixel batch, over all models.

E. Analysis of SAM Finetuning

As mentioned above, tuning the full SAM encoder from the beginning of training leads to deteriorating weights due to bad training signals emitted from the uninitialized decoder. To mitigate this, we start by exclusively training the decoder, and then gradually unlock parameter layers of the encoder. We also note that exclusively training the decoder until convergence does not yield good results - tuning SAM layers is both necessary and effective for the peak performance of BiSeSAM.

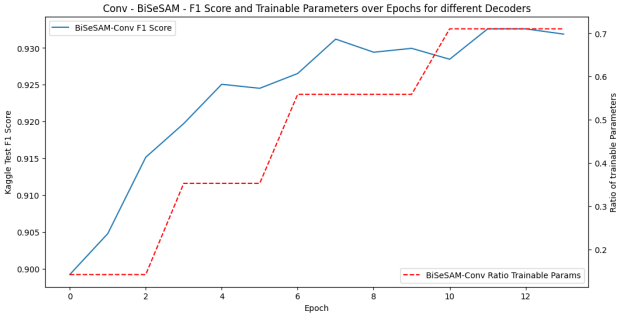

Figure 5. Performance and trainable paraemter development of BiSeSAM by training epoch

F. Confusion Table

Here we report some error analysis for each model. More specifially we provide the TP,TN,FP,FN ratio on the predictions of all models on a test split of **kaggle** data.

Model	TP	TN	FP	FN
Unet - resnet34	14.61	78.63	02.38	04.37
Unet++ - resnet34	14.91	78.46	02.54	04.07
Unet - efficientnet-b5	15.35	78.69	02.32	03.62
Unet++ - efficientnet-b5	14.94	78.69	02.32	04.03
BiSeSAM - Conv	14.71	79.13	02.03	04.11
BiSeSAM - MLP	15.08	78.90	02.27	03.73
BiSeSAM - Sp. Aware-S	14.99	78.99	02.18	03.83
BiSeSAM - Sp. Aware-F	15.02	78.94	02.23	03.79
BiSeSAM - Skip Connect	15.01	78.96	02.21	03.80

Table 2. Confusion Table

G. Data

Our secondary **gmaps** dataset downloaded from the Google Maps API contains 12990 images and corresponding masks. It includes 1370 images from Austin 6500 images from Boston, 1870 images from NYC and a total of 3250 images from the city of Philadelphia. Images with less than 3% road pixels are removed before training.

Fig. 6 shows the distribution of Classes (i.e. road, no road) in each dataset. Our **gmaps** dataset shows similar distribution as the original **kaggle** dataset, making it particularly useful. In both we have a mean of around 16% road pixels.

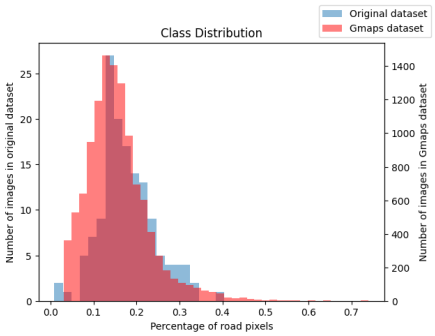

Figure 6. Class Distribution in **kaggle** and **gmaps** datasets

H. Hyperparameter Tuning - Loss Function

Table H shows the similarity in performance between the tested loss functions.

Loss Function	Indicative F1 - Kaggle
BCE	93.00
BCE + Dice	93.08

Table 3. Loss Functions

I. Visualization of Qualitative Results

In Figure 7, we visualize BiSeSAM's performance. From left to right, the original image (input), the ground truth, the predicted mask, and the overlay with mask, FP, and FN are displayed.

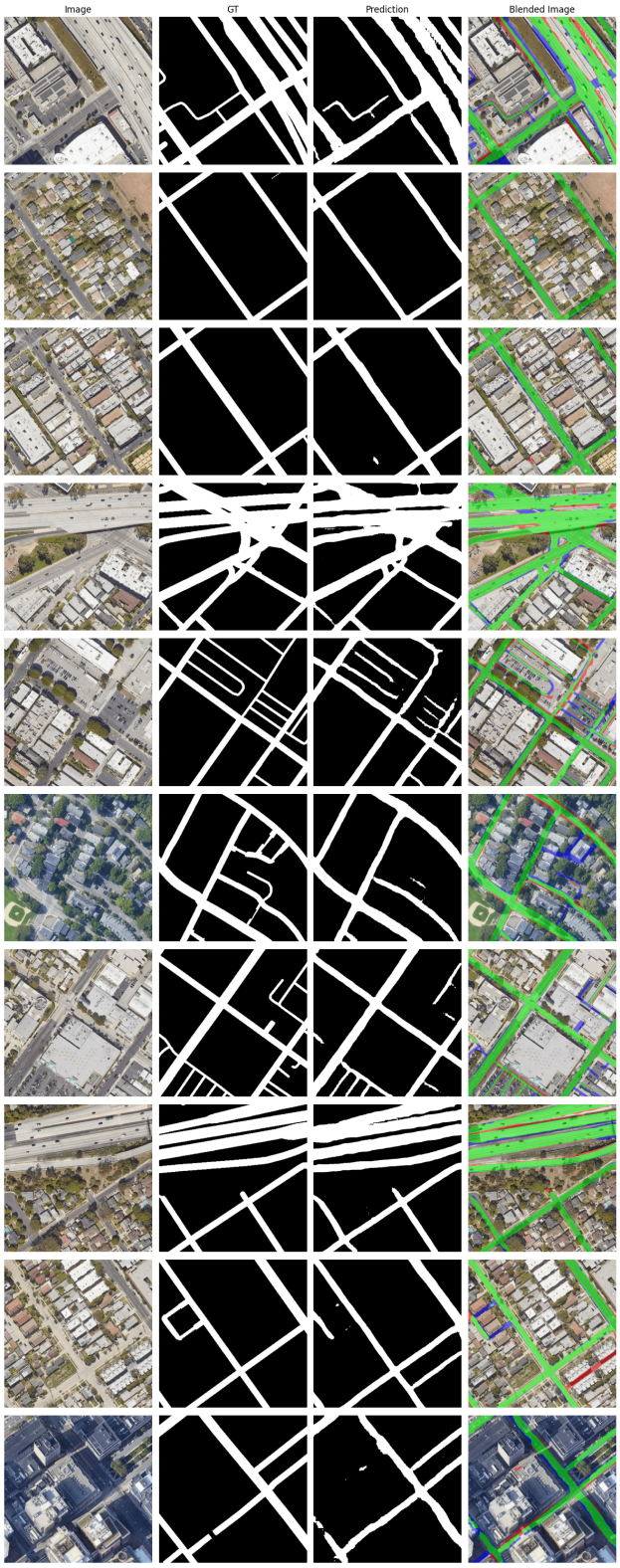

Figure 7. Original Image, Ground Truth, Mask, Overlay
Color codes for Overlay: green - mask, red - FP, blue - FN