

Computer Vision 2025

## Project 10

# Transformer-based Satellite Image and Segmentation Generation for Ground-to-Aerial Image Matching

Michele Capponi

# Summary

- Introduction
- Dataset
- Proposed architecture
- Image Generation (1-4)
- Segmentation (1-2)
- Feature Extraction
- Metrics
- Future developments
- Conclusions



# Ground-to-aerial image matching

**Matching** the right satellite image for each streetview image is a very complex task due to **semantic** differences and **domain** adaptation

*Streetview*



*Corresponding satellite*



This project aims to **tackle this gap** without the need of more informations (like GPS) but only using the streetview image

# Dataset: CVUSA

The [version](#) I used of **Cross-view USA** (CVUSA) contains:

- **Streetview** images
- **Satellite** images
- Satellite **segmentation**
- Polarmap (segmented and not)

The original dataset can be found [here](#), a large dataset containing millions of pairs of ground-level and aerial/satellite images from across the United States.



# Proposed architecture

Input Streetview



Satellite image generation

Transformed

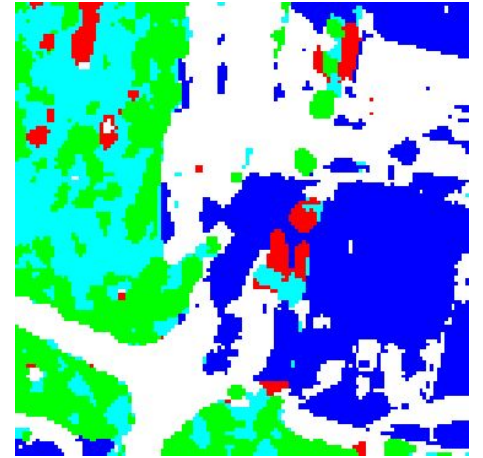


Joint Feature Learning Net

Feature Fusion Net

Matching

Segmentation



Candidate Satellite

# Image Generation (1)

The main thing I tried were **Diffusion Transformers** I based my work on this [paper](#) by Meta

Scalable Diffusion Models with Transformers

William Peebles\*  
UC Berkeley

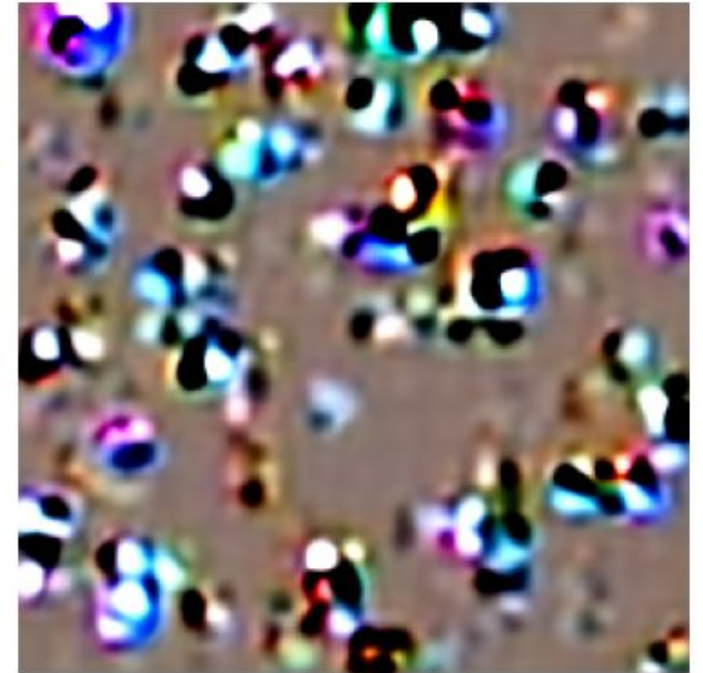
Saining Xie  
New York University



**DiT-XL/2** model, one of the biggest, with **VisualCLIP** and **stability-ai-sd-mse** VAE

**Results?**

Very **difficult conditioning**, as you can see...



Probably my main problem was the VAE since it **wasn't able to decode well its own encodings**

# Image Generation (2)

I tried using a **VQGAN** encoder ([Taming-Transformers](#)), but the **checkpoints are not available** and I wasn't able to train my own.

Then I tried a **Unet** approach still using Diffusion. I opted for the **Stable Diffusion v1-v4** implementation (pre-trained on LAION-5B).

Reference: [huggingface](#)

- VisualCLIP encoding **Openai/clip-vit-base-patch32** (pre-trained on public images)→ 10 epochs for last 2 layers
- **VAE default** of Stable Diffusion → Frozen
- **2 linear layer MLP projection** from CLIP to VAE→ Trained



# Image generation (3)

Pre-trained



Final Result



Original



- AdamW
- ReduceLrOnPlateau
- Loss?



# Image generation (4)

Losses:

1. Noise loss: **MSE** (lr=1e-5)
2. Clip: 1-**CosineSimilarity** (lr=1e-6)
3. VAE: **lpips**

In my last epoch this were my results:

Noise: 0.1791

Clip: 0.3957

VAE: 0.6177

Total: 1.5882

This means that the hardest task still is conditioning...

- in fact every time I regenerated an image I got a (substantially) **different output**
- good denoising but bad conditioning
- only learning truly **major features** (e.g. color, big roads or houses)

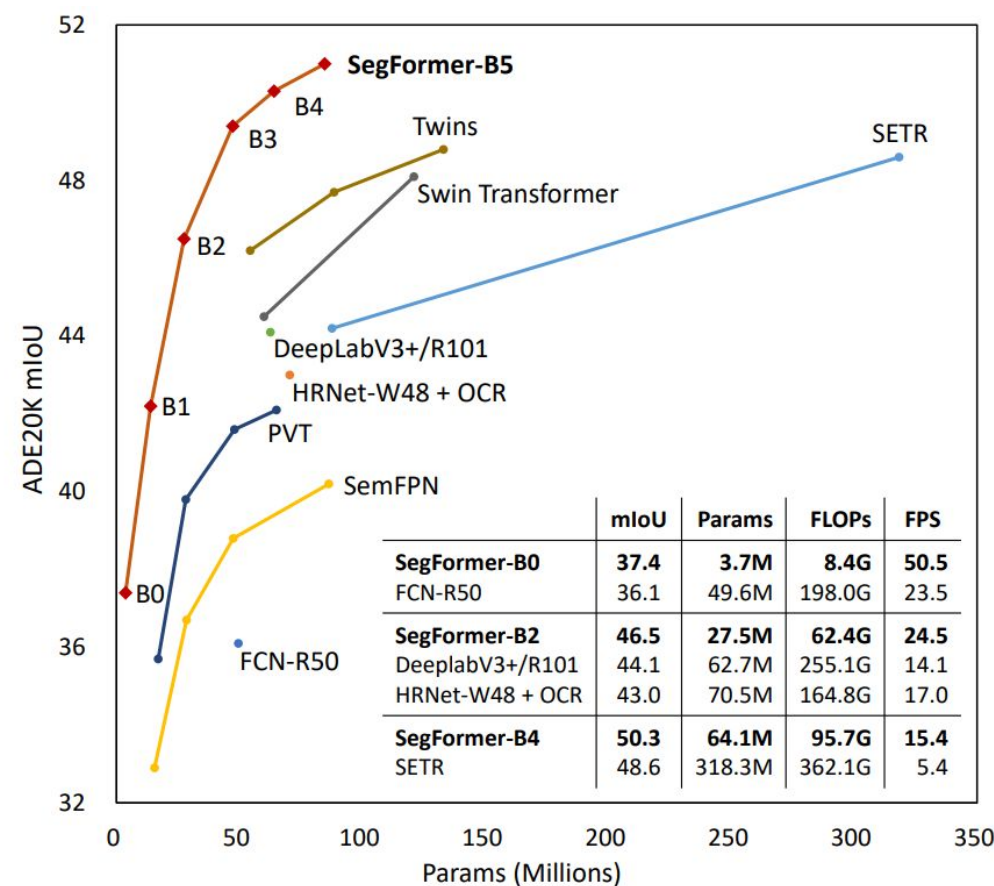
# Segmentation (1)

I used **SegFormer** because there are available checkpoints pre-trained on [Ade20K](#) which has many spatial informations and domain closer to urban and aerial image.

The chosen model was [SegFormerB3](#).

It makes a significant **improvement** from B2 model **without having too many parameters** and it's quite close to B4 and B5 with much less parameters.

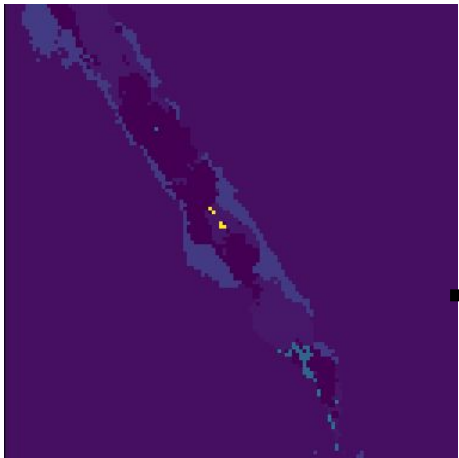
[Reference](#)



# Segmentation (2)

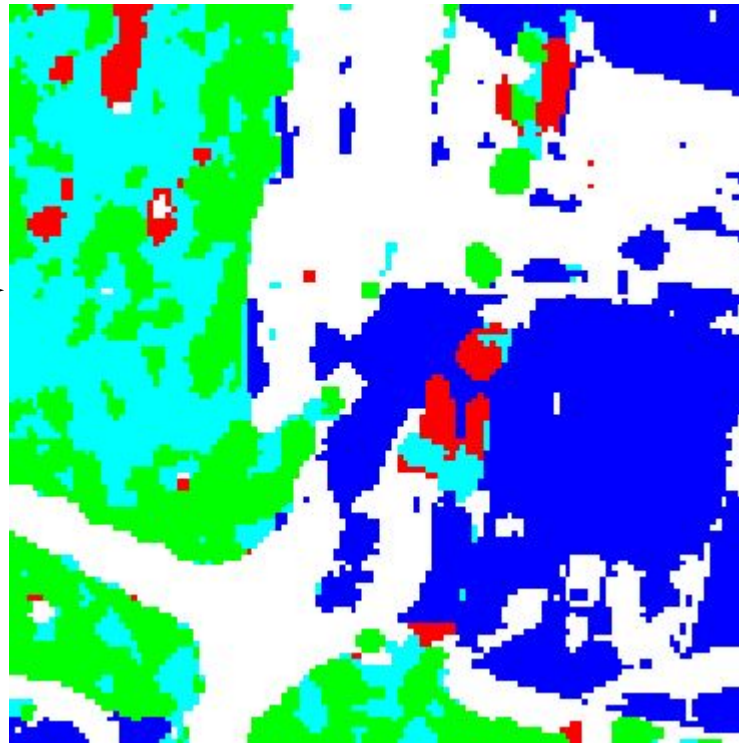
- `CrossEntropyLoss`
- `AdamW`
- `ReduceLrOnPlateau` ( $\text{lr}=3\text{e-}5$ )

Pre-trained

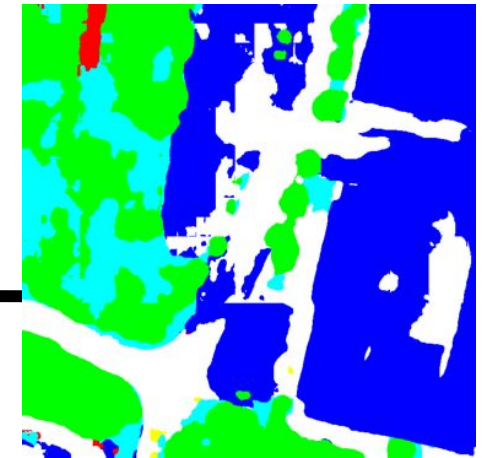


SegFormerB3 on  
Ade20K:  
**mIoU = 0.49**

Fine-tuned



Original



Fine-tuned on  
CVUSA:  
**mIoU = 0.39**

# Feature extraction

## **JointFeatureLearningNet:**

- 4 pre-trained VGG16 (Ground, Synthetic, Seg Synthetic, Candidate)  
→ Joint representation

## **FeatureFusionNet:**

- 1 FFN trained from scratch  
→ Concatenate
- **Triplet Loss**
- **AdamW** (different lr for VGG and FFN)

Last epoch's loss: **0.15** (but small-to-none improvement)



# Metrics

**Base accuracy: 51%**

With 10 competitors:

**Top1: 9%**

**Top5: 49%**

For comparison (with more competitors)

[ALCOR LAB's SAN](#) (with 360 FOV)

Top1: 77%

Top5: 92%

[Regmi and Shah FFN](#)

Top1: 49%

It clearly is unfair to compare my task with those results but that's the **baseline** at the moment. My task of course was broader but nonetheless these are the results.

# Future Developments

Having more hardware capabilities (and time):

- Try again with **Diffusion Transformers** either fine-tuning the **VAE** or training it from scratch
- More fine-tuning on **VisualClip**
- More training on **SegFormer3**
- Adding **segmap** to the JointFeatureLearningNet
- Try with **polarmaps**
- **More complex training loop** for FFN and VGG
- Would be nice to see it reversed (**from satellite to streetview**, maybe embedded in a satellite?)

# Conclusions

What's the take away from this experiment?

- Synthetic Satellite generation is a complex task, especially with bigger architectures like transformers
- The technique works but needs **more experimenting**
- Not an easy project, but
- It was very **interesting and fun** to experiment with
- Will be much **easier and more effective** in the future

**Thanks for your attention**

