Semantic Attention Flow Fields v2

Yingwei Song & Jiayang Zhu
Brown University

Abstract

This short report summarizes the process of adapting Semantic Attention Flow Fields (SAFF) [5] with a new base model DINOv2, performance comparisons, as well as implications for future work. SAFF is a model that combines semantic features in the form of pyramids from a large foundational model with Neural Fields for semantic segmentation of scenes.

1. Introduction and Related Works

DINOv2 was released in 14 Apr 2023 [6], which was around a year after the original model. DINO served as the backbone for the first version of SAFF. The use of self-supervised learning Vision Transformers (ViT) was a new approach for computer vision, which also provided deep features that strongly encoded semantic information [1], allowing for a variety of applications that utilized that property.

When contrasting DINOv2 and v1, the greatest difference in architecture is the introduction of a new patch level objective: iBOT loss. The point of iBOT is to capture visual semantics by "enforcing the similarity of cross-view images on class tokens" [9], which is done by randomly masking patches. This means the patch size, which depends on stride, influences the granularity of features as well as tokenization for the DINOv2 model as a whole. In addition to this fundamental change, there are a number of additions (such as resolution, regularization) in order to optimize the training of a larger dataset, which further improve model performance on general tasks.

\[ \mathcal{L}_{iBOT} = - \sum_i p_{ti} \log p_{si} \]

Where \( p_{ti} \) and \( p_{si} \) are the probability distributions for the teacher and student iBOT heads at patch index \( i \) for masked tokens.

A recent paper by FAIR, the makers of DINO, Guided Distillation for Semi-Supervised Instance Segmentation [2] is a direct application for a similar downstream task.

Semantic segmentation is the task of dividing an image into parts and classifying each part into one of several predefined categories, meaning it is able to identify and distinguish different objects. This has a variety of applications in robotics, manufacture, medicine, AR, et cetera. Empirically, this ability tends to be measured through the Jaccard index, which is the size of the intersection divided by the union, which for the task translates as how much a bounding box of the real object and the box drawn by the system coincides.

2. Method

In order to support DINOv2, we started by adapting the work in Deep ViT Features [1] that extracted layers from the model. While the number of layers in the model was the same, pulling the cls tokens required custom support from model layer, as well as new performance comparisons over whether the tokens of the last layer remained the most performant features. Then, we focused on constructing the semantic pyramid as well as building the NeRF layer. There were a number of constraints here regarding what could realistically be run on one machine and how to parallelize work that required a rewriting of the data loading, handling, and saving process for the semantic pyramid, since we also had to experiment with different strides, which resulted in drastic differences in total volume. We wrote a custom script that supported this and also in order to run, test, and evaluate the core part of SAFF on the original Nvidia Dynamic Scene dataset [8]. A significant amount of this work was inspired by the work done for NSFF. Specifically regarding the NeRF and actually doing the various cosegmentation tasks [4].

3. Results

The results of SAFFv2 are very close to the SAFF in all kinds of evaluative metrics. A potential problem is the performance highly depends on the fine-tuned hyperparameters, which are hard to optimize due to the long training process, roughly two to three days on a TITAN RTX. Regardless, the performance improvement solely from changing the model does not seem to be significant even in ideal circumstances.
4. 3D Gaussian Splatting

In the course of studying SAFF, we also explored some knowledge about 3D Gaussian splatting, with LangSplat [7] being an outstanding article that attracted our attention. It fully utilizes semantic information and embeds it into 3DGS [10], exhibiting similarities to SAFF in its handling of semantic information. Therefore, we conducted some research and exploration on LangSplat as well.

Gaussian Splatting may be highly relevant to the SAFF model as a replacement the NeRF portion of the architecture. By significant improving simplifying the rendering process, we can better explore the feature engineering as well as hyper parameter tuning process.

See below.

4.1. Autoencoder of the CLIP model latent dimension analysis

To facilitate the accelerated training and learning of 3D Gaussian models, the authors of LangSplat compressed the semantic information to a dimension of 3 using an autoencoder. We first explored the dimensionality of this latent code, aiming to achieve a balance between training time and accuracy across different dimensions.

Training and rendering times per feature dimension graph

The chart illustrates that the IOU score didn’t have a significant increase while the autoencoder latent dimension increased on a small scale, while the training time increased significantly.

4.2. Replacing the autoencoder with the speed-up convolution layer from feature 3DGS in LangSplat

In the feature 3DGS article, the authors followed a similar approach to LangSplat but without a defined autoencoder. Instead, they directly embedded the decoder into the
neural network. The reason for using a very simple CNN decoder is that they aimed to achieve knowledge distillation through the rendering process of 3D Gaussian Splatting and learn an explicit representation of the feature field, without using a separate neural network for learning the feature. Therefore, the CNN decoder is only used for upsampling the rendered feature map channels, and a 1x1 convolution layer is sufficient for this purpose.

So we transferred this concept and technique to LangSplat. Due to limitations in computer performance, I assumed the ground truth language feature to be 128 (instead of the actual 512). The results here are only a demonstration to prove the feasibility of this method.

The quality of the original picture is largely preserved in the embedded speed-up.

5. Future Work

The patch level objective documented in the DINOv2 paper, means that over or underestimating stride can lead to unsatisfying results. Either way, there are realistic implications for performance as well as training times.

We believe there are a few potential approaches to the dynamic stride problem, which may provide insight and direction for future study.

1. Training a much smaller auxiliary network that identifies image features and heuristically estimates an appropriate stride
2. Processing the image at multiple strides with a fixed resolution and reintegrating the output when building semantic attention pyramids
3. Define some rule based protocol that quickly determines visual density in an image to serve as a proxy
4. Rethink extraction of features in comparison to Deep ViT

The second option seems like a natural addition to the resolution based semantic pyramid structure during the feature extraction process. However, there are a few problems: the way DINOv2 works, the stride is less adjustable then something like resolution which can be easily scaled: only a small fixed number of values (which are factors of each other) result in a renderable solution. Another issue is the fact that not all strides may contain valuable semantic information. Our experimentation shows the low quality of semantic maps both from overshooting and undershooting the patch size, showing the effect that the iBOT process has on model performance as a whole. This would also require some sort of feature fusion or architectural change.

A learning based approach would require designing and training another network after gathering data regarding performance metrics, as well as image characteristics. Since this would be a classification task for a small amount of discrete strides, a simple model might suffice. However, the obvious issue may be gathering enough performance data in order to feed such a model.

The emerging field of Dynamic Neural Networks may provide some insight as well [3], since parameter adjustment is a dynamic question. The three approaches largely coincide with the mentioned list.

6. Appendix

6.1. Code

Repository is linked [here]

6.2. Acknowledgements

Thanks to Prof. Tompkin and Yiqing for their support.

References


