Context is All You Need: An Unsupervised Segmentation Approach

Priyam Mazumdar
Department of Statistics

Jongwon Lee
Department of Aerospace Engineering

Minkyung Kim
Department of Mechanical Engineering
{ priyamm2, jongwon5, mk58 }@illinois.edu

Abstract

Pixel-wise image segmentation has grown in importance over the last few years, and this technique has great potential in many fields, especially autonomous driving. One of the downfalls of traditional approaches is the difficulty in generating datasets with labeled semantic maps on a large scale, which motivates the necessity of developing unsupervised approaches to the semantic segmentation. We found that the Vision Transformer (ViT), which is effective in perceiving an image's information in a global context, brings a significant performance gain in semantic segmentation in a supervised manner. However, the existing literature lacks the application of a ViT backbone in an unsupervised setting and instead focuses on conventional approaches leveraging a convolutional neural network (CNN) represented as an "encoder-decoder" architecture. This aspect establishes our objective throughout the project. We will be exploring three different architectures that each have unique approaches to building these segmentation maps: An unsupervised convolutional approach called W-Net, a supervised Vision Transformer known as Segmenter, and lastly a different approach to leveraging attention maps in the self-supervised DINO architecture.

1. Background

There is currently not a lot of literature tackling the problem of unsupervised semantic segmentation so we will consider some of the unique approaches that have been attempted here.

1.1. Convolutional Approach

Convolutions have been the main method of image processing and feature extraction for computer vision applications for years. UNet [4] was originally made for efficient end-to-end biomedical imaging segmentation, but today is used for an abundance of applications in semantic segmentation. The power of UNet comes from its unique skip connections, where features learned in the encoder are directly concatenated to the decoder. The intuition of the encoder/decoder architecture is that we hope to use convolutional feature extraction where we force the model to extract only the most relevant features. The features that are most important are chosen by the ability to remap these images back to either a segmentation map or the original image depending on if we are in a supervised or unsupervised setting.

1.2. Contrastive Detection

Contrast has been a very popular approach used in unsupervised learning and pre-training. The general purpose of contrastive loss during learning is to force the model to group similar classes together. For example, when learning the embeddings of pixels for all the images, we want to have pixels originating from the same class (without directly being given the class label) to group together in the vector space. This method effectively converts our images into embeddings on which we can use standard unsupervised techniques such K-Means clustering to segment our different proposed classes.

1.3. Vision Transformer

Convolutions have been known to have some major limitations: 1) Poor translational invariance, 2) Lack of information regarding position or orientation, and 3) Assumption that pixels close together are of more importance. The Vision Transformer architecture can solve many of the limitations of convolutions though with a penalty of higher computational cost. The Vision Transformer adopts much of the popular Transformer architecture used widely in NLP, but with a new edition of the Patch Encoding. As we know in NLP, the input of a Transformer has to be a sequence, but we don't have sequences of pixels in the same way that we do text. To solve this, the authors leverage patch encodings which split the image into 16x16 patches, flatten them and use a linear layer to generate our representational embeddings. The unique ability of this technique is that the model can learn contextual relationships across the entirety of the image which is crucial in a segmentation problem.

The most important aspect of the transformer architecture is the multi-headed attention mechanism. What this does is given an embedded sequence we will generate three separate vectors: key, query, and value. Our goal is to find the weights that emphasize the parts of the keys and queries that are highly correlated, and then we use this as a weighted average across all of our values. By doing so, we will be emphasizing the portions of the values that are indicative of our output while muting the others. Another feature of the transformer is the input and output tensor shapes are identical and this allows us to easily stack multiple transformer layers to learn many representations of the image. Mathematically it can be shown as

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V.$$

Next, we also have to include information about the position of a patch in the image known as a positional encoding. This encoding is a 1D parameter that will indicate the position of a patch with respect to all the others. By doing so, when finding the weights between keys and queries, maintains the spatial positioning between them. The other reason that we need positional encodings is transformers are by design permutation equivariant. This would imply that the order in which we feed sequences does not determine the weights learned between them during self-attention, but this does not fit our need as we are modeling images with clear spatial dependencies. The original paper recommends that we use the following as our positional encodings

$$PE_{\text{pos},2i} = \sin(\text{pos}/10000^{2i/d_{\text{model}}})$$
$$PE_{\text{pos},2i+1} = \cos(\text{pos}/10000^{2i/d_{\text{model}}}),$$

where pos is our position and i is the dimension. But there is an entire literature of types of positional encodings we can use and it does become problem-dependent. Instead, we opted to use learnable parameters for our encodings that will be updated during the training process.

2. Methods

2.1. Convolutional W-Net

The supervised approach in image segmentation has difficulties in getting a large amount of pixel-level labeled data sets. To tackle this issue, unsupervised approaches using FCN have been investigated recently. The encoder-Decoder model is widely used for unsupervised feature learning. Based on the idea of U-Net architecture, W-Net [10] suggests encoder-decoder models using two FCN architectures where each architecture resembles the shape of U-Net. The biggest difference between this architecture and U-Net is the use of depth separable convolution layers. While U-Net uses 3x3 convolutional layers repeatedly, W-Net uses a depth separable convolution layer which consists of depthwise and pointwise convolutions. This enables W-Net to examine spatial and depth dimensions independently.

Each of the two architectures works as an encoder and a decoder. To jointly train the encoder and the decoder, W-Net uses different loss functions at each step. Firstly, in the encoder, it uses Soft N-Cut Loss which is based on the N cut loss proposed by [5]. Cut loss measures dissimilarity between the two disjoint sets using the total weight of the edges that have been removed:

$$Ncut_K(V) = \sum_{k=1}^{K} \frac{\sum_{u \in A_k, v \in V - A_k} w(u, v)}{\sum_{u \in A_k, t \in V} w(u, t)},$$

where k is a label in K, A is a set of pixels in each label weight, and V is the set of all pixels. Weight w computes the likelihood of two pixels belonging to one object. Brightness and the spatial location information of the pixel are used to compute the similarity between two pixels i and j:

$$w_{ij} = \begin{cases} e^{\frac{-||\mathbf{F}(i)-\mathbf{F}(j)||_2^2}{\sigma_I} + \frac{-||\mathbf{X}(i)-\mathbf{X}(j)||_2^2}{\sigma_X}} & \text{if } ||\mathbf{X}(i) - \mathbf{X}(j)||_2 \leq r \\ 0 & \text{otherwise,} \end{cases}$$

where each of X and F represent the spatial location and feature vectors of each pixels. To apply this loss function to the training optimization process, the author of W-Net uses soft version N-cut loss to make it differentiable. Therefore, the soft N-cut loss used to train the encoder is computed as follows:

$$J_{\text{soft}-\text{Ncut}}(V, K)$$
$$= K - \sum_{k=1}^{K} \frac{\sum_{u \in V} p(u = A_k) \sum_{u \in V} w(u, v) p(v = A_k)}{\sum_{u \in V} p(u = A_k) \sum_{t \in V} w(u, t)}.$$

In the decoder, it minimize reconstruction error which is a mean square loss between the raw inputs and the decoder outputs:

$$J_{\text{recon}} = ||\mathbf{X} - \mathbf{U}_{\text{Dec}}(\mathbf{U}_{\text{Enc}}(\text{X}; W_{\text{Enc}}); W_{\text{Dec}})||_2^2.$$

Since the outputs of the FCN are coarse, post-processing is necessary after getting the initial prediction from the trained model. In W-Net, Conditional Random Fields (CRF) and hierarchical segmentation are used to smooth the coarse output of FCNs and merge over segmented partitions.

2.2. Vision + Mask Transformer (Segmenter)

Segmenter [6] proposes to perform supervised semantic segmentation with an end-to-end transformer architecture.

Genrating Patchwise Embeddings: An image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$—where H, W, C stand for height, width, and channel correspondingly—is split in to sequence of patches $\mathbf{x} = \{x_1, \ldots, x_N\} \in \mathbb{R}^{N\times P^2\times C}$, where each patch has a size (P, P) and N, the number of patches, is HW/P^2. After applying flattening and linear transformation to them, we obtain their embeddings $\mathbf{x}_0 \in \mathbb{R}^{N\times D}$. Learnable position embeddings $\mathbf{pos} = \{\text{pos}_1, \ldots, \text{pos}_N\} \in \mathbb{R}^{N\times D}$ are added to $\mathbf{x}_0$ to yield the sequence of input tokens $\mathbf{z}_0 = \mathbf{x}_0 + \mathbf{pos} \in \mathbb{R}^{N\times D}$, which will be encoded by transformer with attention mechanism.

Encoding by Transformer: A transformer composed of L layers is applied to the input tokens $\mathbf{z}_0$, where every layer consists of a multi-headed self-attention (MSA) block followed by multi-layer perceptron (MLP) block with layer normalization (LN) and skip connection:

$$\begin{aligned}\mathbf{a}_{i-1} &= \text{MSA}(\text{LN}(\mathbf{z}_{i-1})) + \mathbf{z}_{i-1}\\ \mathbf{z}_i &= \text{MLP}(\text{LN}(\mathbf{a}_{i-1})) + \mathbf{a}_{i-1},\end{aligned}$$

where $i \in \{1, \ldots, L\}$. Hence, the L-layered transformer yields encoded tokens $\mathbf{z}_L \in \mathbb{R}^{N\times D}$, the sequence of contextualized tokens containing rich semantic information, from the input tokens $\mathbf{z}_0$ of embedded patches with position encoding.

Decoding by Transformer: The resulting encoded tokens $\mathbf{z}_L$ are processed along with class embeddings $\mathbf{c} = \{c_1, \ldots, c_K\} \in \mathbb{R}^{K\times D}$—which are also learnable like the position embeddings ($\mathbf{pos}$). The class embeddings are initialized at random, and will serve as masks for generating segmentation maps after transformation, which will be explained in the next paragraph. A transformer composed of M layers—which is different from what is used for generating the encoded tokens $\mathbf{z}_L$—is applied to both $\mathbf{z}_L$ and $\mathbf{c}$ and generates their L2-normalized output $\mathbf{z}'_M \in \mathbb{R}^{N\times D}$ and $\mathbf{c}_M \in \mathbb{R}^{K\times D}$, respectively. The scheme leveraging instance masks generated by a transformer to facilitate yielding meaningful information from patch token outputs, so-called mask transformer, is inspired by DETR [1], Max-DeepLab [7], and SOLO-v2 [8], but Segmenter [6] is distinct from them in the sense that it shares the same transformer to process both patch-wise tokens and instance masks.

Semantic Map Generation: The set of class masks indicating the per-patch class score is $\mathbf{s} = \mathbf{z}'_M \mathbf{c}^T \in \mathbb{R}^{N\times K}$. This is reshaped into $\mathbf{s} \in \mathbb{R}^{H/P\times W/P\times K}$ to be a 2-D shape and bilinearly upsampled to the original image size to obtain a feature map $\mathbf{s}' \in \mathbb{R}^{H\times W\times K}$. A softmax is then applied on the class dimension followed by a layer norm to obtain pixel-wise class score to form the final outcome so-called “segmentation map”.

2.3. DINO Attention Maps

DINO is a self-supervised architecture that was developed by Facebook AI and has the ability to learn very detailed representations from unlabeled data. This is not your standard semantic segmentation approach where we are given pixel labels to understand what class each object belongs to. Instead, the model uses contrast to cluster together similar objects from ImageNet and by doing so our attention layers in the vision transformer give a nice separation between objects of interest and the background.

Let us delve deeper into how this architecture works. The main process the model uses is called self-distillation where there are two parallel and identical networks known as the student and teacher. These networks can technically be any standard architecture used in computer vision, but vision transformers give the best results due to their attention mechanism. Next, it is important to note that no training happens in the teacher network, but rather the weight of the teacher will be an exponentially weighted average of the student. We can draw on the Momentum Contrast (MoCo) to better understand this.

The first important note of contrastive loss is that given two crops of an image, where one may have been modified with different augmentations, the model is supposed to learn that they still contain the same mutual information. Therefore the model will assign similar latent representations to similar images. The only issue with contrastive loss is that the comparisons are made across a single batch, and with large models and image sizes, this can be a very small set of samples. Momentum contrast instead performs something similar to gradient accumulation where we will aggregate multiple mini-batches for contrast. An exponential moving average of the weights are then calculated based on the accumulated student weights and then directly copied over to the teacher. The other benefit of this method is that it does not allow for mode collapse to occur because if the teacher directly copied the student weights, they may output identical embeddings regardless of input.

Another step taken towards avoiding mode collapse is centering and sharpening the inputs of the teacher network. Centering is the process of subtracting the mini-batch mean from the current batch and sharpening emphasizes peaks in the target distribution to exaggerate differences between the image embeddings. Both the student and teacher models then use a softmax and we use cross-entropy loss to try to make the output distribution of our student embeddings to be as close as possible to the teacher embeddings. By doing so the model will force similar embeddings closer together in the vector space, and we can use K-nearest neighborhood (KNN) to calculate the accuracy of how well the classes were clumped together.

3. Implementation and Results

3.1. Implementation Details

Initial model training on our smaller MP4 dataset was done on Google Colab. Additional training on the ADE20K MIT Scene Parsing Dataset [11] was done in a distributed fashion across two Nvidia Titan RTX GPUs. Specifics of our implementations varied between the different models.

In our replication of the Segmenter model, we followed similar techniques used by the original paper. We leveraged a pre-trained "tiny" vision transformer [9] with a patch size of 16 on images of size 384x384. We also used the SGD optimizer with a learning rate of 0.001 and momentum of 0.9. To help curb over-fitting we added dropout weights of 0.2 to our linear and attention layers. Lastly, we implemented a learning rate scheduler to reduce during the plateau.

W-Net uses Adam optimizer both for the encoder and the decoder training. When training on the MP4 dataset, we use a learning rate of 0.003 for the encoder and 0.004 for the decoder with 30 epochs and set K = 9 which is meant to be the number of classes. For ADE20K dataset, we use a learning rate of 0.0005 with 30 epochs. Since W-Net trained on MP4 shows a satisfactory result, we use K = 9 in ADE20K dataset due to the computational limitation. Additionally, we use a specialized loss function to reduce intensive computation which will be also explained in the following section.

Lastly, we were unable to successfully train the DINO architecture from scratch given our current resources so we focused on applications of a pre-trained model.

3.2. W-Net Results

The original soft N-cut loss method is dealing with the pixel-wise weights for every pixel in the set which includes computationally intensive matrix computation. Due to the computational limitation, we decided to use one of its variants suggested by Odom [3]. It uses a Gaussian spatial filter to compute the pixel-wise weights which improves the efficiency of loss computation a lot.

To check how it works, we first trained W-Net in the MP4 dataset and compare its result with U-Net. The test results from two models can be found in Fig. 1 and Fig. 2. Attached W-Net outputs are post-processed images using CRF implemented by [3]. Compared to the U-Net, W-Net better captures the details like a wheel of a car or a sharp edge of a tower. In this project, only CRF is applied but if we can combine this together with the hierarchical merging or another processing method, we could improve the encoder result further. The decoder of W-Net is able to reconstruct the image similar to the raw input Fig. 3.

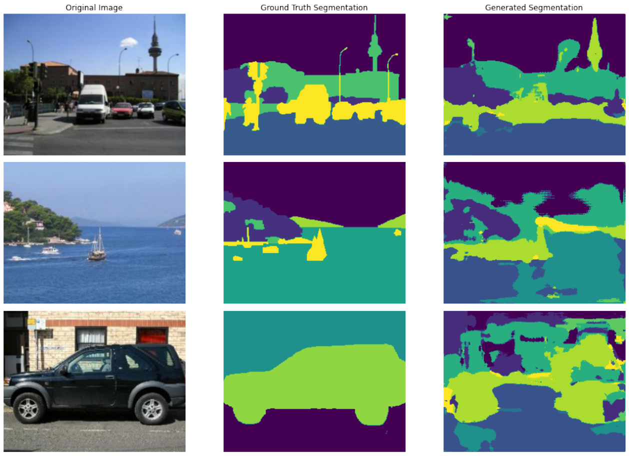

Figure 1. U-Net results on the MP4 test dataset

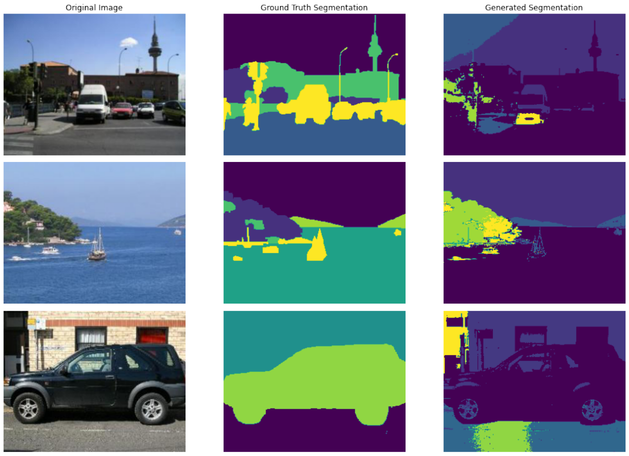

Figure 2. W-Net results on the MP4 test dataset

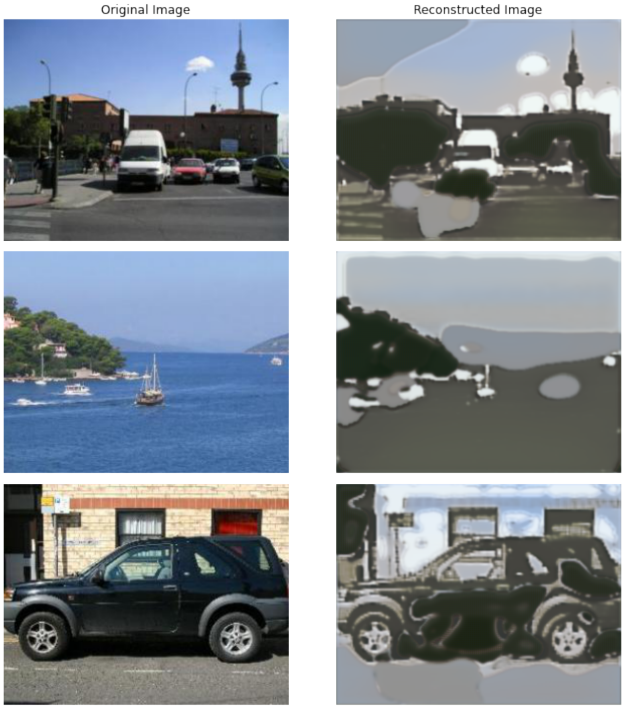

Figure 3. Image reconstruction using W-Net

Next, W-Net is trained on the ADE20K dataset to be used as a baseline. Similar to the previous result, W-Net can capture some edge features. We can see the detailed features of a tower and frames of a window in the encoder output. Fig. 4. However, it fails to segment the entire object. One of our biggest limitations is computational resources and time. We set K= 9 and use 30 epochs to reduce computation but this might prevent the model from converge. Also, the use of a variant method of Soft N-Cut Loss could affect the performance either. As future work, it could be worthwile to train the W-Net with K = 150 and additional epochs to find the optimal parameters for ADE20K dataset.

3.3. Vision + Mask Transformer (Segmenter) Results

As indicated previously, our backbone for the Segmeter model is a "tiny" vision transformer, where the "tiny" indicates the reduced size of each token (192), which is much smaller than the its standard size (768). This does not allow as much information to be embedded but makes the model feasible to train.

Regardless, we obtain satisfactory results as seen in Fig. 5 from our Segmenter implementation. There appear pretty clear separations and fine details of the different objects even with the comparatively large patch size of 16x16 to the original image size of 384x384. although in the image of the chairs it doesn't seem as accurate. The original paper states that improved results may be obtained by increasing the embedding depth with more transformer layers or by decreasing the patch size but this comes at the penalty of training cost; the tiny vision transformer returns about 58 frames per second whereas the larger models can drop to less than 1 per second. Therefore, if this model is implemented in any form of autonomous robotics, we have to consider this important trade-off.

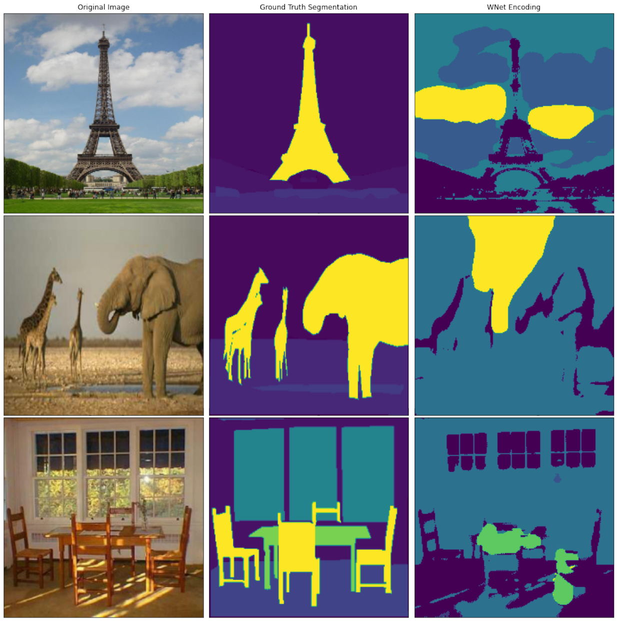

Figure 4. W-Net results on the ADE20K dataset

To compare our model to what the large state-of-the-art model can produce, we used pre-trained models offered by MMSegmentation [2] from Open MMLab. We can clearly see in Fig. 6 that it has resolved much finer details especially when looking at the complex structures of the chair/table.

3.4. DINO Results

DINO is an extremely large model that has two concurrent vision transformer models ranging from 20 to over 80 million parameters. Also like most other state-of-the-art unsupervised models, the model has to also train on extremely large datasets. The original implementation offered by Facebook had pre-trained models that were trained for roughly 2 days on the entirety of ImageNet across a 16 GPU multi-node system. Running their optimized model locally on our available system would have taken between 14 and 16 days to train to reach comparable performance.

Instead of attempting to train this model ourselves,

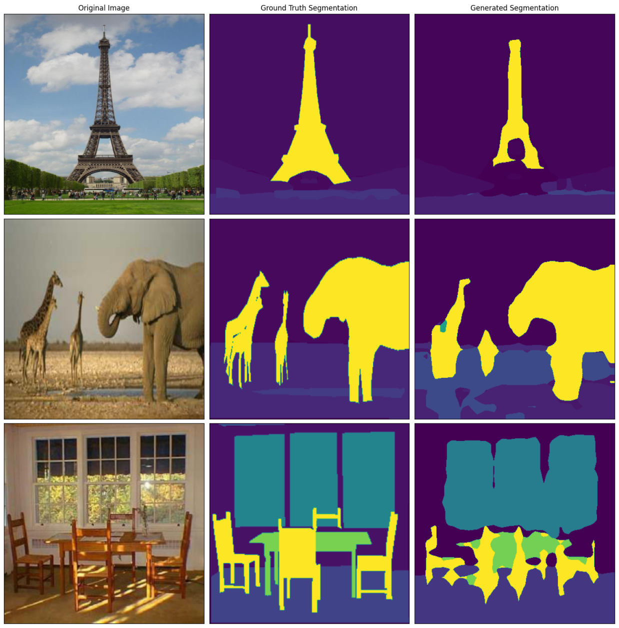

Figure 5. Segmenter results with a patch size of 16x16 on the ADE20K dataset

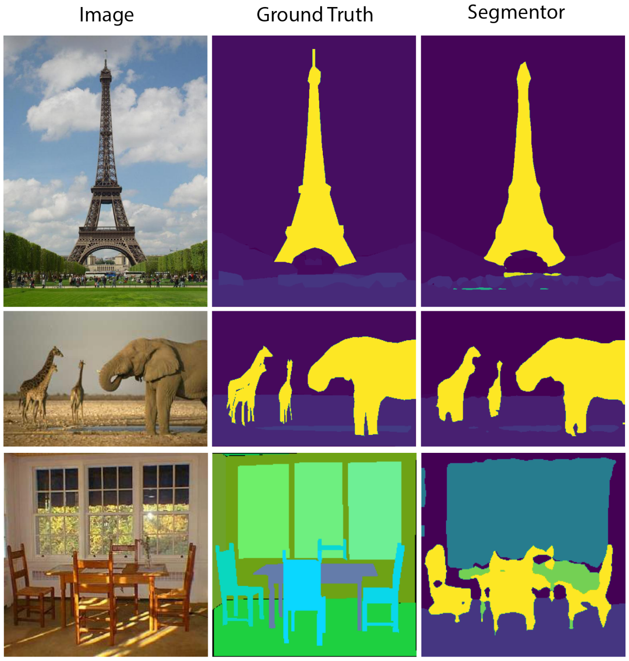

Figure 6. Segmenter results with a patch size of 8x8 on the ADE20K dataset

we decided to explore some of the properties that it can extrapolate from an image or video. The main aspect that we were interested in was plotting the attention maps to see what parts of an image is focused on. To get the most detailed maps, we used the large vision transformer with a patch size of 8x8. The process of extracting the attention map is to run an image through the entire model and store the keys, queries, and values of the last transformer block. We can permute our keys and queries back to the correct shapes and then perform a dot product between the queries and transpose the keys. Beware that these are the weights that we use that are then scaled and applied softmax operation before multiplying to our values. Here are some of the visual outputs of our fully unsupervised DINO segmentation:

We can see in our Fig. 7 that the generated attention maps can perform very detailed segmentation of the most salient parts of an image. We can see evidence of this detail especially from the image of the deer, where it was able to select the horns as well. Another point of interest is the model tends to focus on the face of the animals. We can see the highest weights in the deer and bird being around the eye area, which is also similar to attention behaviors of humans. A limitation of DINO is that we can only reliably expect this segmentation behavior in classes found in ImageNet as that is the underlying dataset it was trained on. Fortunately, due to the unsupervised nature of DINO, we can always extract additional images from sources like Flickr to add more training data.

4. Conclusion

In this exploration, we investigated the benefits and drawbacks of the architectures W-Net, Segmenter, and DINO. The main purpose was to see the capabilities of the attention mechanism within transformers for semantic segmentation and if we are able to capture more global structures within the image.

W-Net was a baseline for us to understand how traditional convolutional approaches can be used in unsupervised image segmentation. We found that it is able to extract edge details quite confidently from images, but has two issues: 1) It focuses on specific features of an object rather than the entire object such as the wheels on a car, and 2) It requires a high amount of computational resources and training time.

Segmenter is a supervised segmentation method that uses a vision transformer backbone as well as a novel mask transformer. We had very clear segmentation on the ADE20K dataset using the tiny vision transformer as our backbone. When using a pre-trained version on the large vision transformer, there was a clear increase in edge clarity and detail retention of the segmentation.

Lastly, DINO is an unsupervised contrastive embedding method whose the goal is to learn embedded representations of images that group objects of high simi-

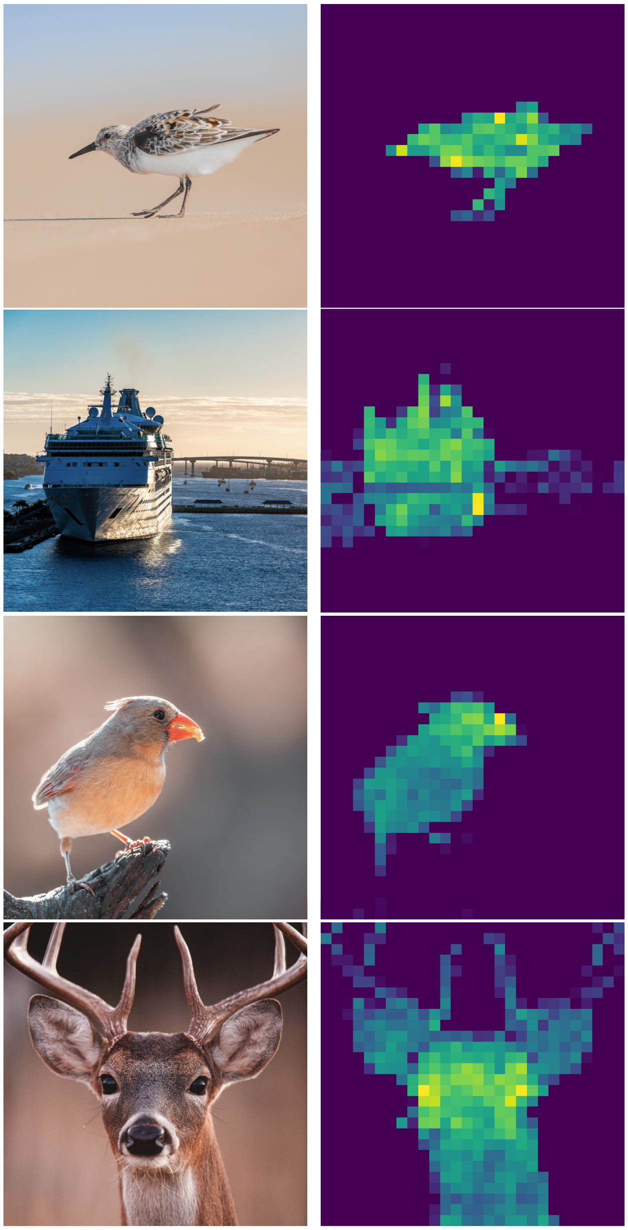

Figure 7. Attention maps generated by DINO

larity together. By doing this procedure and studying the weighted attention maps of the image, we get very clear segmentation. This implies that in the process of contrasting the embeddings, the model inherently learns what parts of an image to focus on.

Although unsupervised approaches can give great segmentation results, it is still important to be able to identify at the pixel level what class the object belongs to. In NLP and speech recognition, large models like BERT and Wav2Vec undergo similar self-supervised pre-training on large corpora of text or audio and then fine-tuned on the limited labeled training data available. We can most likely employ a similar technique in our case, where we fine-tune the Student Vision Transformer from DINO for different tasks. The benefit of this type of contrastive learning is that the model is not learning specific features for a task, but rather a global sense of how images are related to one another, making it possible to use for a multitude of downstream applications.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In European conference on computer vision, pages 213–229. Springer, 2020. 3

[2] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 5

[3] Frank Odom. wnet-unsupervised-image-segmentation. https://github.com/fkodom/wnet-unsupervised-image-segmentation, 2019. 4

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. CoRR, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597. 1

[5] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation, 2000. 2

[6] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation, 2021. 3

[7] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5463–5474, 2021. 3

[8] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. Advances in Neural information processing systems, 33:17721–17732, 2020. 3

[9] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 4

[10] Xide Xia and Brian Kulis. W-net: A deep model for fully unsupervised image segmentation, 2017. 2

[11] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing

through ade20k dataset. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5122–5130, 2017. doi: 10.1109/CVPR.2017.544. 4