ECE570 Course Project Final

Anonymous submission

Abstract

This project reimplements OneFormer; it is a transformer-based model that addresses the challenges of multi-task image segmentation, specifically targeting panoptic, instance, and semantic segmentation in a single architecture. My approach introduces a task-conditioned transformer that adjusts and changes the conditions of each layer dynamically to each segmentation task, utilizing task-specific embeddings generated through contrastive learning. The backbone network extracts multi-scale features processed by a transformer decoder conditioned on task-specific queries. I also implemented a contrastive loss and used a Hungarian matcher for optimal assignment during loss calculation. I evaluated the model on the COCO dataset with mean Intersection over Union metrics. While the contrastive loss decreases with additional epochs, the model's segmentation performance is low, with an mIoU score of 0.0012. Additionally, the cross-entropy loss currently outputs zero, indicating challenges in class prediction accuracy, and this could be a reason for a low mIoU score. These results emphasize there are areas for improvement, but after weeks of debugging and reviewing the original code, I faced many challenges and decided to conclude this project with a low model performance.

Code — https://anonymous.4open.science/r/REIMPLEMENTATION-ONEFORMER-4272/README.md

1. Introduction

Image segmentation plays a fundamental role in numerous computer vision applications, from autonomous driving and medical imaging to industrial automation. The ability to accurately distinguish and label different parts of an image helps machines interpret visual data in a way that can drive informed decision-making. If image segmentation is performed in a wrong way, it could lead to critical errors or consequences in any of those fields.

Image segmentation is typically divided into three primary tasks: semantic segmentation, instance segmentation, and panoptic segmentation. Semantic segmentation assigns a class label to each pixel, and instance segmentation differentiates individual object instances within the same class. Semantic segmentation and instance segmentation basically work together to get the results. Panoptic segmentation combines these by simultaneously assigning category labels and distinguishing instances within those categories.

Historically, all the models years before these transformers began gaining popularity in the computer vision field, had one model for each segmentation task. If they were to do image segmentation on semantic, instance, and panoptic segmentation, they would need three separate models to train. The performance of these traditional methods of image segmentation did not have low performance. If anything, these models were performing well considering that they didn't have these ideas of new machine learning algorithms we currently have. However, the downside of these models was their efficiency. Each segmentation task having its own model to evaluate led to increased computational costs and complex deployment pipelines. Therefore, in recent years, researchers have shifted toward unified architectures where these segmentation tasks can be done by one model handling multiple tasks within a single framework. This was possible due to the development and improvement of transformers. Transformer-based models were initially used in natural language processing, which had a huge success in. Now, transformers are also used in the computer vision field, including segmentation. Their ability to model long range dependencies and capture context effectively helped the segmentation tasks in many different ways. When these transformers are coupled with computer vision and contrastive learning, these models could differentiate similar and dissimilar representations, and these architectures offered a strong approach with a lot of potential for multi-task segmentation.

In this project, I explore a transformer-based model called OneFormer [1], designed to handle panoptic, instance, and semantic segmentation using a unified approach. Key components of this architecture include multi-scale feature extraction, task-conditioned query formulation, and contrastive learning, which together aim to enhance the model's segmentation quality across tasks. Specifically, the contrastive loss helps align task embeddings effectively, while the primary loss is calculated using a Hungarian-matching-based criterion. To evaluate model performance, I use the Cross-entropy Loss and Mean Intersection over Union (mIoU) metric to provide insight into segmentation accuracy across different tasks.

2. Related Work

Some of the existing research or methods related to my project are:

2.1 Fully Convolutional Networks

- **Fully Convolutional Networks, introduced by Long, Shelhamer, and Darrell in 2015 [2]**

  This paper introduces the usage of convolutional neural networks for pixel-level image segmentation. Before this was introduced, CNNs were mostly used for classification tasks, which output a single class label for an entire image (ex. Dog and cat images, and model predicts "dog" or "cat"). Fully Convolutional Networks innovated this idea by removing the fully connected layers which are traditionally layers added at the end of CNNs to connect input neurons to output neurons, and instead adding their fully convolutional layers so that models can make dense pixel-wise predictions. The critical feature of this was their use of deconvolution layers, or learnable upsampling, to recover spatial resolution lost while passing through convolutional and pooling layers. These deconvolution layers upsampled the feature maps, which allowed their models to generate outputs at the same resolution. It also introduced skip connections which fused lower level layers with higher level semantic features from deeper layers. There is no need to compare this model to OneFormer [1], as this paper laid the groundwork for further advancements in segmentation, which led ideas to OneFormer [1].

2.2 Segmentation Masks

- Mask R-CNN, introduced by He et al. in 2017 [3]

  This paper extended Faster R-CNN [9] to support pixel-level instance segmentation. Faster R-CNN [9] was actually a model for object detection, which output bounding boxes and class labels for detected objects, similar to YOLO models. Mask R-CNN adds a parallel branch that generates segmentation masks, allowing the model to perform both object detection and segmentation within a single framework. For this model, they have a backbone network for feature extraction, which is similar to OneFormer [1], called ResNet (or ResNeXt) combined with a Feature Pyramid Network. Feature Pyramid Network helps the network handle objects of different sizes, and this backbone generates feature maps that feed into the region proposal network. This was actually one of the first models to have a unified framework, which OneFormer [1] also aims to develop. However, this model had some limitations: Even though this model performed two different tasks within a single framework, it was not entirely unified as they had to use separate branches for each task. The difference between this model and OneFormer [1] is that OneFormer [1] doesn't need to separate any models or branches; one model controls and handles all segmentation tasks.

2.3 Unified Transformer-Based Segmentation

- Per-Pixel Classification is Not All You Need for Panoptic Segmentation by Cheng et al. in 2022 [4]

  They introduce a novel approach to panoptic segmentation by building on Mask2Former's [10] framework. This paper actually has a very similar problem definition and motivation to OneFormer [1], which is to create a transformer-based model that is more general across multiple segmentation tasks using a single framework. They use task-conditioned queries that allow the model to adjust to each segmentation task, and they serve as input to the transformer decoder, which has the same structure as OneFormer [1]. Then their architecture includes a feature pyramid network, which is used in Mask R-CNN [3], and a cross attention mechanism to refine the task-conditioned queries by focusing on relevant regions of the multi-scale feature maps. I would say that this is the most similar model architecture compared to OneFormer [1]. However, OneFormer [1] uses a more advanced task-conditioned transformer that conditions the entire transformer architecture instead of only conditioning the queries. Also, this model misses one key factor of OneFormer [1]: contrastive learning.

Other related papers:

- Swin Transformer: Hierarchical Vision Transformer using Shifted Windows [5]

  Swin Transformer is a hierarchical vision transformer architecture that uses shifted windows. This allows the model to capture multi-scale representations efficiently. The authors of OneFormer [1] used this transformer, and evaluated the results, but I did not have a full understanding of what this transformer can do, I did not use this architecture.

- Segmenter: Transformer for Semantic Segmentation [6]

  Segmenter adapts transfomers for semantic segmentation tasks. It also uses the attention mechanisms of transformers, it captures the global contextual relationships across an image. For OneFormer's [1] semantic segmentation task, this paper is highly related, but is not used in my project.

- Panoptic Segmentation [7]

  This paper introduces the concept of panoptic segmentation. It unifies instance and semantic segmentation into a single framework, which is crucial for OneFormer's

[1] panoptic segmentation process. This paper lays the groundwork for Mask R-CNN [3] and OneFormer [1].

- End-to-End Object Detection with Transformers [8]

  This paper introduces the DETR model, used by OneFormer [1]. It is a transformer-based approach to object detection that does not need anchor boxes. Its framework simplifies the pipelines, and the authors of OneFormer [1] uses this model. This is not included in my project.

3. Problem Definition

The core challenge addressed by this project is to develop a single, unified model capable of performing panoptic, instance, and semantic segmentation tasks within a single framework, and with better performance overall. Like mentioned above, its focus is on the computational complexity and costs, and training separate models for each segmentation task is not efficient. By creating a unified model for segmentation would help its model complexity to be simpler, as well as maintaining or increasing the model performance, as unified model architecture only requires one model to be trained for all three image segmentation tasks, making it efficient.

Technically, this project aims to unify the model through a transformer-based architecture that can handle and adjust dynamically to each segmentation task, without having to separate them. Key challenges of this project include developing an effective multi-scale feature extraction to accommodate for different spatial resolutions for an accurate segmentation, and creating a task-conditioned queries that appropriately maps the mask inputs to its task-specific outputs, and utilizing contrastive learning to align task embeddings.

It specifically addresses these technical aspects:

- Multi-scale feature extraction: extracting features across different scales to handle objects of varying sizes while maintaining context.

- Task-conditioned query formulation: implementing queries that adjust to the requirements of each segmentation task, aiming to increase flexibility within the transformer architecture.

- Contrastive learning integration: Using contrastive loss to improve the separation and alignment of task embeddings, allowing the model to handle diverse segmentation tasks within one framework.

4. Methodology

4.1 Experimental setup

The experimental setup for this project is designed to evaluate the effectiveness of my transformer-based model for unified segmentation, which is a reimplementation of OneFormer [1]. My primary objective is to calculate the contrastive loss, cross-entropy loss, and mIoU metrics to measure the model performance, using the COCO dataset as our benchmark.

I started by setting up a Python environment and I utilized these libraries: PyTorch, PIL, numpy, and scipy. I was originally planning on using TensorFlow, but to enhance my experience in different libraries, I used PyTorch. Using CUDA was also a part of my plan originally, but due to some issues with Google Colab and my model, I decided to use a small batch of my dataset with fewer epochs. For the dataset, I use the COCO 2017 dataset for training, mostly for its panoptic segmentation subset, which includes both instance and semantic segmentation labels for each image. These datasets were downloaded from the COCO dataset website, and all images were imported and resized to 512x512 to standardize input dimensions. Originally, I was going to use the ADE20k and Cityscapes datasets, but they were not available for download, and it required an account or a license to use them. I was searching the web for similar datasets, like from Kaggle, but was not able to find a suitable one. COCO datasets were available to the public, and I was able to download them using wget. For training, I only use the first 5000 images of the dataset, due to my setup's computational limitations.

4.2 Algorithms and Techniques

My algorithms and techniques are based on OneFormer's [1] model architecture.

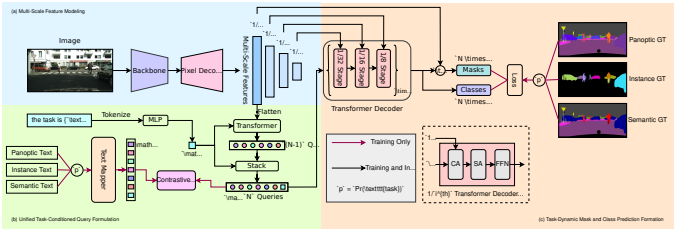

Figure 1: Overview of the OneFormer [1] architecture, illustrating the multi-scale feature modeling, task-conditioned query formulation, and task-dynamic mask and class prediction formation.

- Part A: Multi-Scale Feature Modeling

  For this part of the architecture, I created 2 functions:

  Backbone function: I implemented this function using a ResNet-50 model, adapted with a feature pyramid network. This was customized and it generates feature maps at 1/32, 1/16, 1/8, and 1/4 scales.

  Pixel Decoder: This function is necessary after feature extraction, as it processes the multi-scale features to

enhance their spatial resolution. This decoder upsamples the feature maps to a common resolution, maintaining its original resolution and details. This also lowers the computational cost.

- Part B: Unified Task-Conditioned Query Formulation

  For this part of the architecture, I created 4 functions:

  Text Mapper: I implemented this function to encode the task type (instance, semantic, and panoptic) into embeddings and they allow the model to adjust its output for each segmentation type by acting as a task-specific signal.

  Task Tokenizer and MLP: These functions tokenize and process task descriptions. The tokenizer takes in textual task descriptions and tokenizes them, so that these tasks are mapped to a set of numerical embeddings. Then they are processed through a multi-layer perceptron, which is the MLP function, and they pass through fully connected layers with ReLU activations, ensuring they are aligned with the dimensions from the transformers' input requirements.

  Task-Conditioned Query Formulator: This function generates task-conditioned queries (labeled as Q_task in the diagram) based on the task embeddings, and they serve as input to the transformer decoder. This function uses the output of the text mapper function and this function conditions the queries on the specific task, so that the model is able to adjust its attention and feature processing for different segmentation. My implementation for this part could be wrong, as I did not quite understand what the diagram was doing with the stack and producing a Q with N-1 queries.

  Contrastive Loss: This function computes a contrastive loss between the task-conditioned queries (Q_task) and text-based embeddings (Q_text). This loss tells us how distinct these task-specific embeddings are, and helps the model to have more consistent outputs.

- Part C: Task-Dynamic Mask and Class Prediction Formation

  Transformer Decoder: This function was actually available in the PyTorch library, but I was not able to use it as our model structure needed customized multi-scale features as input, and it was very difficult for me to reshape the features and use them as input, so I decided to implement this function without it. This function basically processes the task-conditioned queries, received by the task-conditioned query formulator, and uses the multi-scale feature maps and pass them through self-attention and cross-attention layers, refining the features based on the segmentation task at different scales.

  Criterion and Hungarian Matcher: The criterion function calculates the primary loss (cross-entropy loss) for training, using a hungarian matching approach, which performs optimal matching between the predicted and ground-truth objects.

  Mask and Class Predictor: This function predicts the class labels and segmentation masks based on the refined features from the transformer decoder. This has a fully connected layer, and a convolutional layer to generate predictions, and it was calculated with a cross-entropy loss using the criterion function.

Those are all the helper functions and techniques that I used for this project, and they were all used in the main file.

Inside the main file, I first defined the hyperparameters that I will use to train and test this model. Then, I imported the COCO dataset that I downloaded from the website, and transformed them to be 512x512 images. I take the first 5000 images of this dataset with a batch size of 1. After the dataset was preprocessed, I declared all the helper functions with the appropriate inputs, and an Adam optimizer. For training, I generate the multi scale features from the backbone function, and I decode the features using the pixel decoder function. Then I make a separate variable for the ¼ decoded feature as it gets passed into the part for conditioning the queries. I use the tokenizer function to generate task_embeddings, and pass them through the MLP function to refine them. Then, Q_text is generated with the text_mapper function, and we formulate the task-conditioned queries using the task_query formulator, using the output of the MLP as input. Then Q_text and Q_task are passed into the contrastive loss function, and the code calculates the contrastive loss. For the last part, Q_text and Q_task are reshaped and goes through the transformer decoder with the multi-scale features, flatten the image, then the output of these functions are combined and gets passed into the mask and class predictor, and the criterion function to generate a calculated cross-entropy loss. Then the predicted masks and class labels are compared using the mIoU calculation, and generates the mIoU metrics of this model.

5. Experimental Results

The key performance metrics tracked throughout training are contrastive loss, cross-entropy loss, and mean intersection over union (mIoU). However, I encountered challenges in cross-entropy calculation, which impacted the mIoU results significantly.

5.1 Model performance overview

Due to issues with the cross-entropy loss calculations, the model's primary loss component (loss_ce) remained at

zero across all epochs. This loss calculation is essential for optimizing the classification of segmentation classes, and because of this issue, it impacts the training process and the mean intersection over union calculation also. On the other hand, contrastive loss calculations seem correct, and the loss decreases steadily across epochs, suggesting that the model is effectively learning to align task-conditioned queries with task embeddings. The only problem is that the miscalculations of cross-entropy loss is preventing the model from learning class-specific segmentation.

I have been debugging this issue for weeks, and could not come up with a solution. My guess is that it's either calculation and reshaping errors of the criterion function, meaning the masks and classes are not shaped properly to make a prediction. This problem also leads to possible issues with the mask and class prediction function.

5.2 Main Results

Epoch	Contrastive Loss	Cross-Entropy Loss	mIoU
1	5.837608	0.0	0.000008
2	5.774187	0.0	0.000009
3	5.711468	0.0	0.000009
4	5.649393	0.0	0.000009
5	5.587609	0.0	0.000009
6	5.526156	0.0	0.000009
7	5.464902	0.0	0.000009
8	5.403699	0.0	0.000009
9	5.342478	0.0	0.000009
10	5.281090	0.0	0.00001

Table 1: Training Metrics Over Epochs, shows steady decrease in contrastive loss over epochs, and zero cross-entropy loss, as well as the low mIoU score.

With the correct implementation and training, contrastive loss should be close to 0, which indicates a good result. In Figure 1, you can see that the contrastive loss value is decreasing every epoch, and I believe that the loss can be decreased to a much smaller loss value with more epochs. However, due to computational limitations, I cannot show the results after 60000 to 90000 epochs (the authors of OneFormer [1] iterated 60k-90k times). For cross-entropy loss, you can see that it remains zero, and the mIoU value is a very small number. Based on the results shown in the OneFormer [1] paper, it shows that the mIoU value should be around 50 to 60, which I failed to output.

As you can see in Figure 2, it shows that the output of mask prediction is not performed properly, leading to having pixels and weird shapes, and the image is unidentifiable. This could be due to reshaping and scaling the images in a wrong way, as our model's decoder should be preserving the original resolution after getting passed through multiple layers. It seems like the original resolution is not kept properly, and shows bigger pixels.

Figure 2: Generated Prediction Mask, shows unclear object shape, inconsistent pixel mapping.

6. Conclusion and Future Directions

6.1 Conclusion

This project aimed to reimplement the OneFormer [1] architecture to achieve unified image segmentation across panoptic, instance, and semantic tasks within a single framework. By implementing features like multi-scale feature extraction, task-conditioned queries, and contrastive learning, the model was designed to adjust dynamically to each segmentation task. However, the experimental results showed limitations and misunderstandings of a few concepts, leading to decreased model performance, in terms of cross-entropy loss and mean intersection over union (mIoU) metrics.

Besides that, contrastive learning was integrated successfully, and we could see that the contrastive loss was decreasing with more epochs. Despite having the proper contrastive learning algorithm set up, the model was not able to learn the accurate class-specific segmentation due to internal issues with the cross-entropy loss, also impacting the mIoU score. Upon further analysis, it became clear to me that there is a possibility that these issues come from miscalculations in the criterion function or reshaping errors within the mask and class prediction function. The generated prediction mask (Figure 2) works as proof, and it shows a lack of clear object shapes and inconsistent pixel mapping.

6.2 Future Directions

It is obvious that the future direction of this project should be fixing the issue with cross-entropy loss so that the model can actually train and learn the accurate class-specific

segmentation, as well as having the correct image output for the mask prediction. That would be the immediate priority for future work and this would involve examining the criterion function and ensuring that both masks and classes are properly shaped for accurate predictions. If that does not seem to be the solution, I will look into the loss calculation, such as focal loss or dice loss, and this might fix the issue and provide better performance.

Another option is to look into the mask and class prediction function that I implemented, to preserve the spatial resolution of predicted segmentation masks. I could implement an adaptive resizing strategy or using additional layers while processing the input image to preserve image quality and object boundaries.

Other than the model performance issue, I can increase the number of epochs and the size of the dataset that I am training the model with. The original OneFormer [1] achieved high mIoU scores after training for 60,000 to 90,000 epochs using the full dataset. My project was limited to 10 epochs, and used a subset of the COCO dataset due to constraints. If I conduct an extensive training, it could potentially yield better results, and allowing the model to learn and improve the mIoU scores.

It would also be a good idea to explore different transformer architectures, such as the Swin Transformer [5] which the OneFormer [1] utilized, and have a different backbone other than ResNet-50. The Swin Transformer could enhance multi-scale feature extraction and improve the model's overall accuracy. Given that OneFormer [1] already demonstrates strong performance with transformer-based segmentation, experimenting with different architectures could lead to further advancements.

That would conclude the options for fixing the issues with this project, and hoping to have better results and an overall performance. If I were to extend this project even further after fixing the issues, I would like to integrate preprocessing techniques like histogram equalization, as I have experience in them from conducting a computer vision research in a lab. That could possibly increase the robustness of the model. I could also utilize different types of layers, or refine the cross-attention mechanism to improve the model's ability to distinguish between different segmentation tasks.

Overall, even though this project didn't yield the best results, it showed valuable insights into the challenges and complexities of reimplementing advanced transformer-based architectures for multi-task segmentation. It was my first time working on image segmentation, so I had a hard time understanding all the aspects of this topic. However, I do believe that this project was very helpful, and future iterations of this work will leverage those insights and possible solutions to build a fully working unified segmentation model with improved performance.

References

[1] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. *arXiv preprint arXiv:2211.06220*, 2022.

[2] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[4] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-Pixel Classification is Not All You Need for Semantic Segmentation. *CoRR*, abs/2107.06278, 2021.

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *CoRR*, abs/2103.14030, 2021.

[6] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for Semantic Segmentation. *CoRR*, abs/2105.05633, 2021.

[7] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic Segmentation. *CoRR*, abs/1801.00868, 2018.

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. *CoRR*, abs/2005.12872, 2020.

[9] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*, abs/1506.01497, 2015.

[10] Bowen Cheng, Ishan Misra, Alexander G. Schwing, and Alexander Kirillov. Masked-attention Mask Transformer for Universal Image Segmentation. *CoRR*, abs/2112.01527, 2021.