Status Report: Object Detection by DETR with Swin Transformer

Jiahui Zhu, Jingmei Yang, Zexin Sun, Jinzhou Zhao
{buzjhcs, jmyang, zxsun, jinzhou}@bu.edu

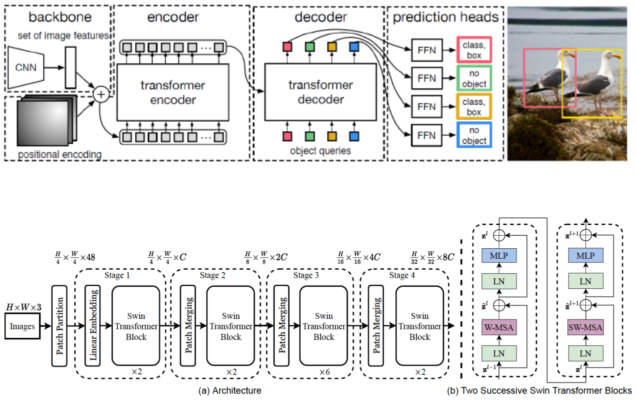


Figure 1. We propose a new framework based on DETR [3] shown in the top figure and the Swin transformer structure [1] shown in the bottom.

1. Task

In this project, we are interested in object detection in 2D images. The transformer has gained enormous achievements in different fields, varying from computer vision to natural language processing. A new backbone with the shifted windows strategy--Swin Transformer--was presented recently and attracted enormous attention from the computer vision research community because of its remarkable performance and efficiency. We will further explore its applicability in object detection. We reimplemented the Detection Transformer (DETR) framework [3] and the Swin Transformer structure in this project. Moreover, the novelty of our task lies in fusing the Swin transformer to the existing backbone of the DETR detection framework to improve the training efficiency and performance power.

2. Related Work

Object detection plays an essential role in autonomous driving systems. Many researchers have demonstrated different frameworks. In [3], the authors presented the DETR (DEtection TRansformer) framework. The DETR framework is distinguishable from the previously shown object detection frameworks since it employs end-to-end training without incorporating hand-crafted ingredients from prior knowledge. Besides, it utilizes the transformer structure gracefully without non-maximum suppression (NMS). Using the attention mechanism and set-to-set loss [3] to train the networks, DETR outperforms previously presented frameworks such as Faster RCNN and UPSNet [2]. However, there are still some limitations of DETR, especially in targeting small objects. Another drawback is the extra-long training time. Meanwhile, the Swin transformer [1] has attracted much attention recently because of its highly efficient and effective performance on diverse vision problems. It produces a hierarchical feature representation and has linear computational complexity.

3. Approach

DETR and Swin Transformer codes are available in the Github repositories (https://github.com/facebookresearch/detr, https://github.com/microsoft/Swin-Transformer/blob/main/models/swin_transformer.py). We reimplemented the Swin Transformer based on the architecture given in [1]. Attention is calculated: Attention(Q,K,V) = SoftMax($QK^T/\sqrt{d}$+B)V, where Q, K, and V are query, key, and value matrices, respectively. [1] suggests that adding a relative position bias in computing self-attention can significantly increase model performance. We then followed the same path in computing similarity. Another trick that the inventors of the Swin transformer used is the cyclic shift. Shifted window partitions generate more windows than the unshifted configuration, and they are usually smaller than the regular unshifted window size, which causes headaches in batch computation. Hence, the authors adopted a more efficient batch computation method by cyclic-shifting toward the top-left direction, as shown in Figure 2. We also follow the same route in our implementation.

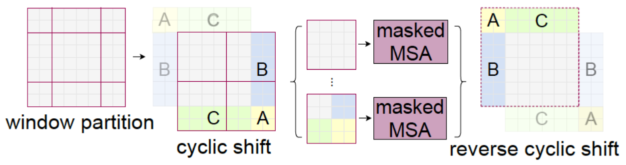


Figure 2. Shifted-window partitioning approach. [1]

We reimplemented the DETR based on the architecture given in [3]. One of the main difficulties of training is to score predicted objects concerning the ground truth, and hence how to design Loss is important. In [3], Loss produces an optimal bipartite matching between predicted and ground-truth objects, and then optimizes object-specific (bounding box) losses. As stated in [3], the ground truth set of objects is denoted by y, and ŷ = {ŷ$_i$} is the set of *N* predictions. Assuming *N* is larger than the number of objects in the image, y also as a set of size *N* padded with ∅ (no object). To find a bipartite matching between these two sets, a permutation of *N* elements $\sigma \in \check{\mathcal{G}}_N$ with the lowest cost can be searched by the following equation:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$$

The second step is to compute the Hungarian loss for all pairs matched previously. We define the Hungarian loss as:

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

The second part of the Loss function is the bounding box loss. Using such an approach simplifies the implementation and poses an issue with relative scaling of the loss. To reduce this issue, a linear combination of the L$_1$ loss and the generalized IoU loss—(L$_{iou}$) is used. Therefore, the box loss is:

$$\mathrm{L}_{box}(b_i, \hat{b}_{\sigma(i)}) = \lambda_{iou} L_{iou}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{L1} ||b_i, \hat{b}_{\sigma(i)}||_1$$

These two losses are normalized by the number of objects inside the batch.

Then, we will merge the Swin Transformer to the DETR. Overall, DETR consists of 3 main parts, a ResNet50 as a backbone to extract representation, an encoder and decoder transformer, and a feed-forward network. We propose to replace a ResNet50 backbone with the Swin transformer and keep other structures unchanged. We will modify the related classes and functions in the Swin transformer reimplementation such that the output dimension matches the input of the following structure in the DETR.

Finally, we will train this proposed framework and test its performance in the WIDER FACE dataset and modify the loss function part in the original code to investigate different IOU loss functions in the Transformer script. If time permits, this model will also be applied to the COCO (sub)dataset [4] for training and testing. Then, we will make comparisons of its results with existing methods in [3] according to various metrics.

4. Dataset and Metric

Initially, we planned to use the 2017 COCO dataset to train and test our proposed model, which contains 118K annotated samples in the training set, 5k in the validation, and 41K in the testing set from 53 stuff categories. Due to the giant sample size of the 2017 COCO training set, it took about 1 hour to train for one epoch. We decided to validate our code using a smaller dataset called WIDER FACE, which includes 32203 images and labels 393703 faces. We randomly selected 40%, 10%, and 50% images for training, validating, and testing. Luckily, the reimplemented DETR achieved a decent performance on the WIDER FACE dataset. After validating our reimplemented DETR, we switched our focus to training DETR using the COCO dataset. Next, we plan to train our proposed framework using a subset of the COCO 2017 training set. This curated mini-training set includes about 25K images, roughly 20% of the COCO 2017 training set. Due to time limitations, we will use fewer epochs than the original configuration, so the testing performance is expected to be worse than the results shown in [3].

Metrics: We will use the same evaluation protocols by COCO, including average precision (AP), AP Across Scales, and Average Recall (AR). It is hoped to show that the average error can be reduced compared to the approach in [1] and [3] with our proposed method if using the same number of epochs.

5. Preliminary Results

We have already successfully implemented Swin Transformer and DETR and tested our DETR code on a smaller dataset called WIDE FACE and the 2017 COCO dataset with 10 epochs. Swin transformer part was reimplemented by Jingmei Yang and Zexin Sun. Jiahui Zhu and Jinzhou Zhao are responsible for reimplementing the DETR part.

Jingmei Yang was responsible for five classes and functions, including patch embedding, patch merging, window partition, reverse window partition, and shifted window transformer blocks. Patch embedding is mainly used for splitting an image into 56* 56 patches, and a 96 dimensions embedding is used to represent each patch. This operation is done by taking advantage of the convolutional layer. After passing to a convolutional layer, the feature maps were passed to a linear layer so that the embedding is projected to the desired dimension. The window partition function is mainly used for partitioning an image into 56 * 56 windows, and each window contains 7 * 7 patches. The window partition is an essential preprocessing step for the successive window attention operation. Window reverse partition function is the inverse operation. Patch merging is mainly used for downsampling, which is similar to the pooling layer in the convolutional neural network, reducing the resolution of input and increasing the depth. The Swin transformer block is the essential building block. The Swin transformer block consists of a layer normalization layer, unshifted windows or shifted window attention block, and a feed-forward fully connected layer.

Zexin Sun was responsible for 4 classes and their functions, including MLP, WindowAttention, BasicLayer, and SwinTransformer. The MLP class is the classic multilayer perceptron structure with two fully connected layers and a GELU hidden layer. WindowAttention class is used for multi-head self-attention, which supports both shifted (SW-MSA), and non-shifted windows (W-MSA) configurations. In W-MSA, the input dimension of the Swin transformer block is B x 56^2 x 96, where B is the batch size, 56^2 is the height and width of a window, and 96 is the embedding dimension. After passing through a normalization layer and a window_partition function with window size set to be 7, we obtain an output with a dimension 8^2B x 7^2 x 96, which means each image has 64 windows, and each window owns 49 patches. Then, the output from previous functions is passed to the WindowAttention function to yield QKV and relative position bias information. An SW-MSA immediately follows a W-MSA to make information intercommunicate across different local windows. The cycle-shifting method is used to facilitate the computation of window self-attention since SW-MSA returns windows with irregular sizes. By shifting the windows and applying a mask to a window, we can efficiently compute the shifted window attention. Here, we emphasize that attention computation is applied to each local window, thus reducing the computational complexity. The BasicLayer class is a basic Swin transformer layer on one stage, which contains Swin Transformer Block and Patch Merging, as shown in Figure 1. The SwinTransformer class is constructed and will be used to replace the backbone DETR, as shown in Figure 1. More specifically, the number of layers in SwinTransformer is 2, 2, 6, and 2, and the number of channels in the hidden layers is 96.

Jiahui Zhu was responsible for 2 classes and their functions, including DETR and SetCriterion. DETR is the model for object detection, whose input includes Backbone, Transformer, Num_classes, and Num_queries. They represent the backbone to be used, the transformer architecture, and the number of object classes and queries respectively. For the COCO dataset, 100 queries are used, while for WIDER FACE, we set the number of queries to be 10, which means the maximal number of objects that DETR can detect in a single image. The forward function expects a NestedTensor, which consists of batched images with the shape BxCxHxW and a binary mask. The output contains the classification logits for all queries and the normalized boxes coordinates. In this implementation, we use ResNet50 as our backbone, and the input is images with the shape 3xH_0xW_0. Output is a downsample feature map with the shape CxHxW, where C=2048 is the number of channels, H=H_0/32, and W=W_0/32. SetCriterion class is the main part of computing DETR loss, which computes the Hungarian assignment between the ground truth boxes and the model's outputs and supervises each pair of matched ground-truth with prediction.

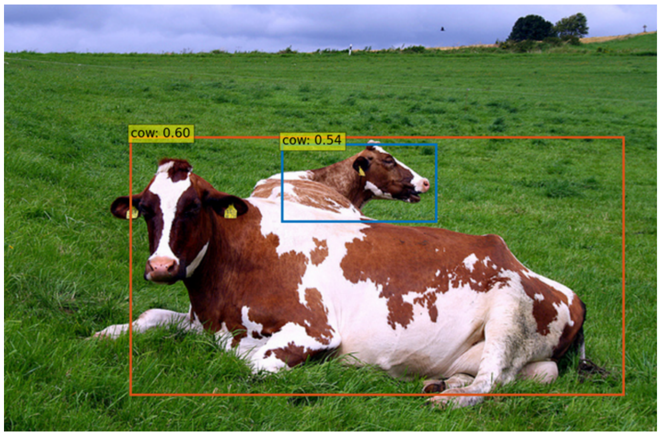
Here, we will further discuss the training result obtained in 5 epochs and 10 epochs in terms of class_error and loss.

5 epochs:
Specifically, it takes about 10 hours to train for 5 epochs. As given in the result table, class_error is 74.18 (losses['class_error'] = 100 - accuracy(src_logits[idx], target_classes_o)[0]) and loss is 14.3653. Here, we take the result of detecting cows in an image as an example. If the classification threshold is set to 50% for the category of the cow in the testing phase, the network fails to yield any predicted bounding boxes and associated classification probability.

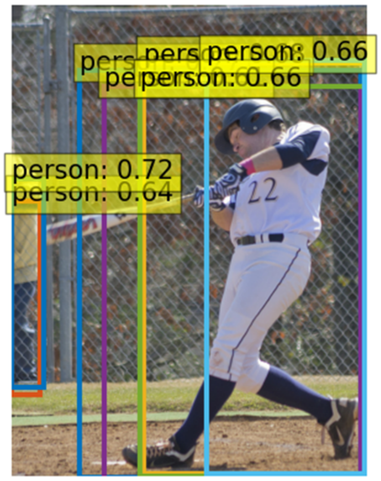
	5 epochs	**10 epochs**
class_error	74.18	34.20
loss	14.3653	13.4672
Training time	10 hours	18 hours

10 epochs:

It takes about 18 hours to train for 10 epochs. After training for 10 epochs, the network achieves a class error of 34.20 and a loss of 13.4672. Compared with the result obtained from 5 epochs, the result obtained from 10 epochs achieves a better loss. Even though the network can successfully recognize cows in an image, the classification probability is close to 50%, which is still very low.



As we all know, networks achieve better results when it is sufficiently trained. In addition to the number of training iterations, the complexity of an image also dramatically affects the prediction power and the difficulty of image recognition. Here, let us take detecting persons as an example to illustrate the effect of image complexity further. Even though the network was only trained for 5 epochs, it achieves a decent detection power. As we increase the recognition threshold, a large number of bounding boxes are dropped, therefore significantly improving the recognition effect.



6. Detailed Timeline and Roles

Task	Deadline	Who
Reimplement Swin Transformer	03/20/22	Jingmei Yang Zexin Sun
Reimplement DETR	03/25/22	Jiahui Zhu Jinzhou Zhao
Test DETR on the WIDER FACE dataset	04/01/22	All teammates
Upgrade DETR framework with Swin transformer	04/10/22	All teammates
Test the upgrade model on the COCO (mini-)dataset and analyze the results	04/20/22	All teammates
Prepare the report and presentation	04/26/22	All teammates

7. Preliminary Code

Our project's repository on GitHub contains our up-to-date reimplementation of DETR, Swin Transformer, and a brief description, which can be accessed by the following link.

GITHUB: https://github.com/JingmeiY/DL523DETR

References

[1] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, arXiv e-prints arXiv:2103.14030, 2021

[2] Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E.,Urtasun,R.:Upsnet: A unified panoptic segmentation network. In: CVPR, 2019.

[3] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In ECCV, 2020.

[4] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dolla ́r, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV, 2014.

[5] Jiahui Yu , Yuning Jiang , Zhangyang Wang , Zhimin Cao, Thomas Huang. UnitBox: An Advanced Object Detection Network[J]. ACM, 2016.