FINAL PROJECT FOR COMPUTER VISION 2022 SPRING

A PREPRINT

Keyu Mao
School of Data Science
Fudan University
220 Handan Road, Shanghai 200433
20307130241@fudan.edu.cn

Junhao Yuan
School of Data Science
Fudan University
220 Handan Road, Shanghai 200433
20307130129@fudan.edu.cn

Zehao Zhang
School of Data Science
Fudan University
220 Handan Road, Shanghai 200433
20307130201@fudan.edu.cn

June 5, 2022

ABSTRACT

Our project contains three parts. (i) Testing and comparing several open source semantic segmentation models on Cityscapes, with which semantic segmentation on a specific video for the output is performed. (ii) Performing and comparing three different ways of training the Faster RCNN: using the Resnet50 pretrained on ImageNet as the backbone, using the parameters of the backbone of Mask RCNN trained on COCO as our initialized backbone, and training the network with all the parameters randomly initialized. (iii) Designing and training a vision transformer on CIFAR-100 dataset with augmentation and comparing it with ResNet-18.

Codes available at GitHub: https://github.com/mskmei/FINAL-PROJECT-CV-2022Spring.git.

Models and other materials can be accessed in appendix A.

Keywords Semantic Segmentation · Object Detection · Vision Transformer

1 Semantic Segmentation

1.1 Introduction

In this section we will use several well-known models in the field of semantic segmentation to test a video downloaded from the Internet to obtain the semantic segmentation results. All models used in this section are open source models pre-trained on the Cityscapes dataset. We will analyze and compare the segmentation results of each model.(1)

Note that the output videos and some explanation on them can be accessed in appendix A.

1.2 Network Architecture

In this section, 3 models will be used in total. They are respectively PSPNet(PSP101,i.e.use ResNet101 to extract the feature map, and so is PSA101), PSANet(PSA101), Deeplabv3. We will give you a brief introduction of the network architecture of these models in this subsection.

1.2.1 PSPNet

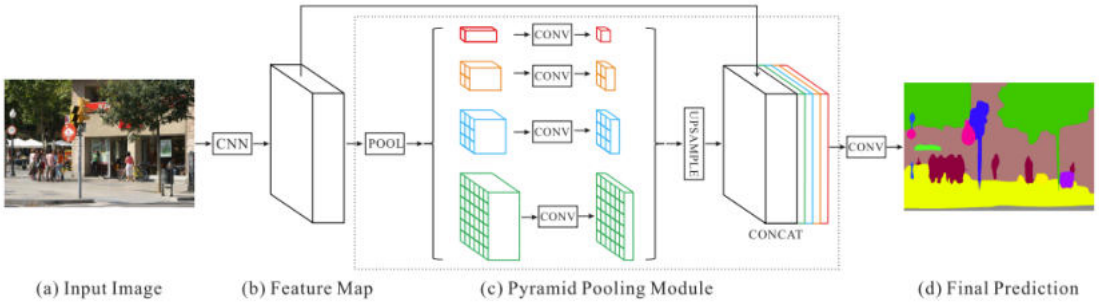


Figure 1: Network architecture of PSPNet

PSPNet, or Pyramid Scene Parsing Network, is a famous semantic segmentation model. The author innovatively proposed pyramid pooling module, a hierarchical global prior, containing information with different scales and varying among different sub-regions, for global scene prior construction upon the final-layer-feature-map of the deep neural network(2). This module can be seen in the part (c) of Figure 1.

To be exactly, PSPNet uses a pretrained ResNet model(3) with the dilated network strategy to extract the feature map. The final feature map size is 1/8 of the input image. On the top of the map, the pyramid pooling module is used to gather context information. 4-level(bin sizes are respectively 1×1,2×2,3×3,6×6) pyramid is used to ensure the pooling kernels can cover the whole, half of, and small portions of the image. They are fused as the global prior. Then concatenate the prior with the original feature map in the final part of (c) in Figure 1. Finally, a convolutional layer is used to generate the final prediction map in the last part of Figure 1.

1.2.2 PSANet

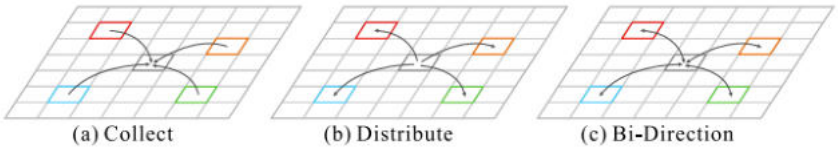


Figure 2: Illustration of bi-direction information propagation model

Point-wise spatial attention network,or PSANet is a network that can aggregate long-range contextual information in a flexible and adaptive manner(4). Each position in the feature map is connected with all other ones through self-adaptively predicted attention maps, thus harvesting various information nearby and far away. The author creatively introduced bi-directional information propagation path for a comprehensive understanding of complex scenes(which can be seen in Figure 2).

The PSA module takes a spatial feature map X with size H × W as input. Through the two branches as illustrated in Figure 3,Through several convolutional layers, pixel-wise global attention maps are generated for each position in feature map X. Then concatenate newly generated representations Z^c and Z^d and apply a convolutional layer with batch normalization and activation layers for dimension reduction and feature fusion. After concatenating the new global

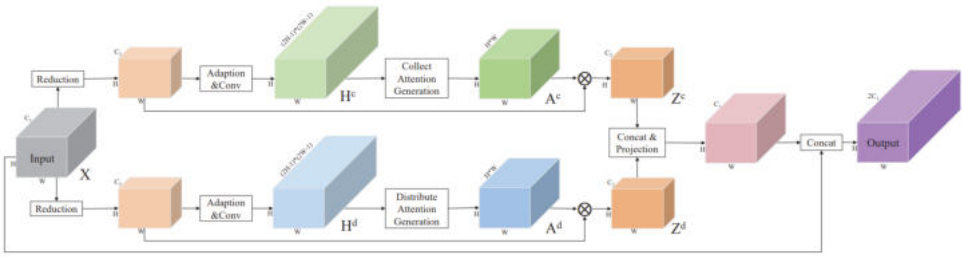


Figure 3: Network Architecture of PSANet

contextual feature with the local representation feature X, one or several convolutional layers with batch normalization and activation layers are followed to generate the final feature map for following subnetworks.(4)

1.2.3 Deeplabv3

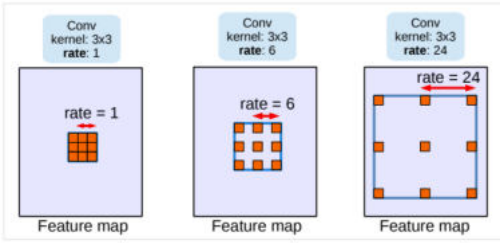


Figure 4: Atrous convolution with kernel size 3 × 3 and different rates.

DeepLabv3 is a semantic segmentation architecture that improves upon DeepLabv2 with several modifications. To handle the problem of segmenting objects at multiple scales, modules are designed which employ atrous convolution(in Figure 4) in cascade or in parallel to capture multi-scale context by adopting multiple atrous rates.(5)

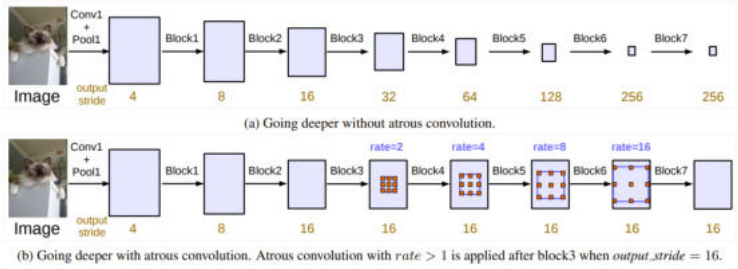


Figure 5: Cascaded modules without and with atrous convolution

The model use a method named "Going Deeper with Atrous Convolution". Authors revisited the Atrous Spatial Pyramid Pooling proposed before(6), where four parallel atrous convolutions with different atrous rates are applied on top of the feature map.

Figure 5 shows the copy of Resnet's Block4 (with three 3*3 convolutions), and then cascaded behind the network to form Block5, Block6, and Block7.In Figure5 (a), No hole convolution is used, so the image resolution has been reduced (by experience, this reduction is very serious for the loss of information). Figure5 (b) shows the situation without changing the resolution and receptive field. Next, the same effect is achieved by using atrous convolution.(5)

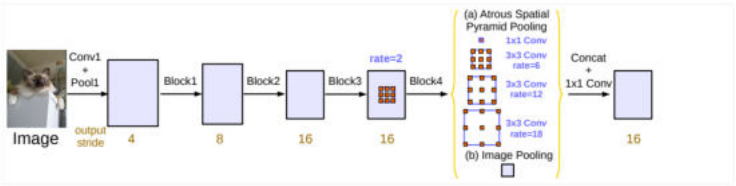


Figure 6: Parallel modules with atrous convolution (ASPP), augmented with image-level features

ASPP(Atrous Spatial Pyramid Pooling) with different atrous rates effectively captures multi-scale information. However, as the block goes deeper, the rate of the hole convolution becomes larger, which will cause the convolution kernel to degenerate to 1×1. To solve it, Image-Level image level features are added, that is, global average pooling is performed for each channel of the input feature map, and then 256 1×1 convolution kernels are passed to construct a new (1, 1, 256) feature map. Obtain the required resolution map (as shown in Figure 6(b)) through bilinear interpolation. Finally concat (a) and (b), and pass through 256 1 * 1 convolution kernels, then a new feature map is obtained.

After the introduction to the method based on atrous convolution, let's have quick look at the structure of deeplabv3 in Figure 7

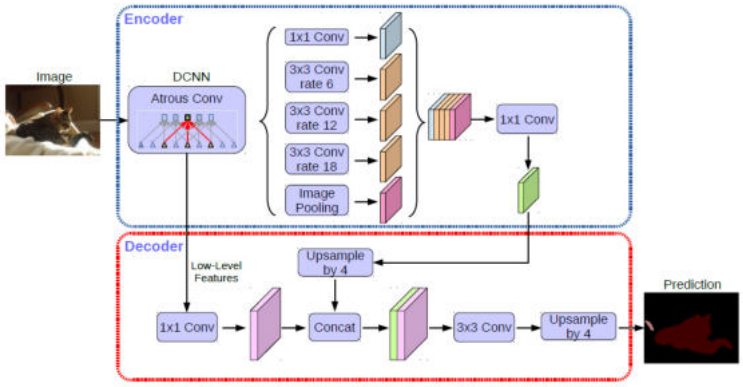


Figure 7: The whole Structure for deeplabv3

The modified aligned Xception improved ResNet-101 is used as Backbone in the original paper, but in our experiments we use mobilenetv2 instead(7).(Structure of Moblilenetv2 can be seen in Figure 8)

In the Decoder, the number of channels is adjusted using 1x1 convolution for the preliminary feature layer compressed twice. It is then stacked with the result of the upsampling of the effective feature layer (the output of the Encoder part) after the atrous convolution. After completing the stacking, two depthwise separable convolutions are performed. At this time, we have obtained a final feature layer, which is the feature condensation of the entire image.

After getting the final feature layer, use a 1x1 convolution to adjust the channel to Num_Classes.Then use resize to upsample so that the final output layer has the same width and height as the input images.

1.3 Result Visualization

After performing the task of semantic segmentation, we got two kinds of results. One is composed of segmentation results only (that is, only color blocks), and the other is a mixture of segmentation results and the original image.

The following Figure 9 will show you an example of these 2 types of results where the figures are produced by PSP101. Obviously, the latter can better reflect the relationship with the original image, as it is able to view the semantic segmentation results and meanwhile compare the differences between the segmentation results and the original

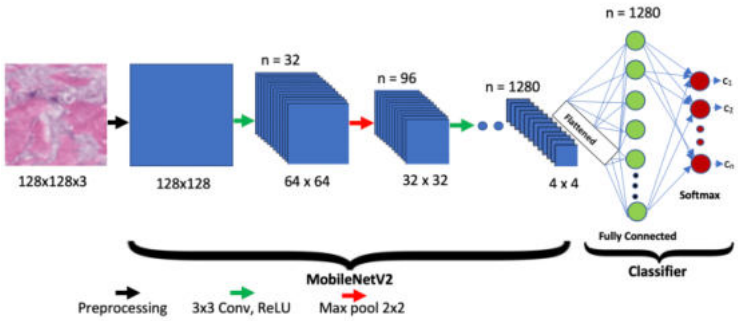
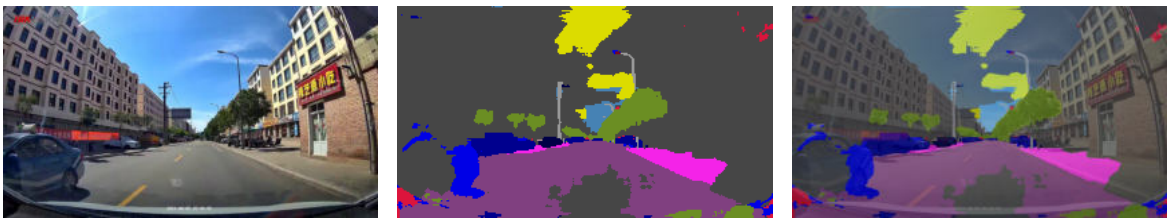


Figure 8: The structure of Mobilenet v2

image.For each model used, we will make two videos based on the two results. But in this part, due to space limitations, we will only show the latter and compare and analyze them.



(a) Origin figure (b) Inferred figure (c) Blended figure

Figure 9: Example of 2 types of results produced by PSP101

In this part, we will sample three images (they are from the 1st, 100th, and 200th frames of the video). The results of semantic segmentation under three different models will be shown respectively.

1.3.1 Original Figures

(a) Figure 1 (b) Figure 2 (c) Figure 3

Figure 10: Original Figures

The original pictures are 3 frames from the video(The video was divided in to 254 frames) in Figure 10. The pictures are took from China's streets and it's simple to find that the main elements in pictures are cars, roads, sky, people and trees. As we use models pre-trained on Cityscapes dataset, whose images also mainly consist of these elements, we believe these models will make sense.

1.3.2 Results by PSP101

Obviously, PSPNet is relatively poor in the division of backgrounds such as roads and sky. By observing its unmixed segmentation results(Figure 9(b)), we can find that PSPNet tends to classify parts of the road and sky as houses, which is also the biggest flaw of PSPNet in this segmentation task. In addition, in Figure 11(b), PSPNet segmented the

(a) Figure 1 (b) Figure 2 (c) Figure 3

Figure 11: Results by PSP101

motor vehicles parked on the roadside into pedestrians, indicating that PSPNet is relatively sensitive to identifying pedestrians.Besides, PSPNet performs well in segmenting cars and trees.

1.3.3 Results by PSA101

(a) Figure 1 (b) Figure 2 (c) Figure 3

Figure 12: Results by PSA101

PSANet obviously outperforms PSPNet in segmentation of backgrounds such as road and sky. Especially the road, the whole road is almost completely divided. For the segmentation of the sky, although the part is still divided into houses, the effect is generally better than that of PSPNet. PSANet does not misjudge motor vehicles as pedestrians in the above three examples in Figure 12. In other aspects, such as the division of sidewalks, trees, poles and cars, the effect is similar to PSPNet.

1.3.4 Results by Deeplabv3

(a) Figure 1 (b) Figure 2 (c) Figure 3

Figure 13: Results by Deeplabv3

Deeplabv3 also outperforms PSPNet in segmentation of local backgrounds such as roads and sky. Its effect in segmenting the sky is particularly nice, where the misjudged part can be ignored. Although there are occasional segmentation results of other categories on the road, we believe that it is caused by the interference of the shadows of trees and cars. Although in Figure 13(b), Deeplabv3 misjudged motor vehicles as pedestrians more seriously than PSPNet, this phenomenon does not appear frequently in other examples (even in other parts of the video), so it can be considered an accidental event.

1.3.5 Extra Comparisons

To figure out which network performs relatively well in this experiment, we have to add an extra comparison. Through the results above, it seems that PSANet performs well in both background and objects(only PSANet doesn't mistake vehicles as pedestrians). Nevertheless, We wonder if PSA is too insensitive to pedestrian segmentation. Unfortunately, the answer is yes. Let's see the figures in Figure 14, it is easy to find that in the segmentation results of PSANet(Figure 14(b)), the success of riding battery cars on the left and right sides is not segmented into pedestrians. But this is achieved in both Deeplabv3 and PSP (at least part of it is segmented into pedestrians). This proves that PSANet does have the disadvantage of being insensitive to pedestrians.

(a) Results by PSP101 (b) Results by PSA101 (c) Results bu Deeplabv3

Figure 14: Results by Deeplabv3

1.4 Conclusion

In this experiment, we use three models PSPNet, PSANet, and Deeplabv3 pre-trained on the Cityscapes dataset to perform semantic segmentation on the same video. We intercepted some representative frames to discuss and analyze the semantic segmentation effects of the three models.

First of all, all three achieved relatively good semantic segmentation results on the videos we downloaded, indicating that the models trained on the Cityscapes dataset have certain transfer capabilities.

Second, PSPNet is less effective in semantic segmentation of backgrounds, especially roads and sky. PSANet and Deeplabv are significantly better than PSPNet in background segmentation, and they have achieved excellent segmentation results in the sky and roads, respectively. Although PSPNet and Deeplabv3 both incorrectly segment motor vehicles into pedestrians, this situation is accidental and has little impact. In contrast, although PSANet does not have the above-mentioned misjudgment, it is too insensitive to pedestrian segmentation, resulting in frequent omission of pedestrians in semantic segmentation tasks.

To sum up, we believe that Deeplabv3 has the best comprehensive performance in this experiment and achieved the best segmentation effect, followed by PSANet and PSPNet respectively. If you want to get information about the results of this semantic segmentation experiment, please go to the video download link at the beginning of the article to download the videos corresponding to the three models to watch.

2 Faster RCNN

2.1 Introduction

In the field of object detection, the two-stage method Faster RCNN has shown great vitality, which is still the basis of many object detection algorithms to this day. The Region Proposal Network (RPN) is the key of Faster RCNN, which proposes some regions of interest for the RoIHead to perform classification and bounding box regression. However, Faster RCNN was not designed for pixel-to -pixel alignment between the network inputs and outputs. This is most evident in how RoIPooling, the *de facto* core operation for attending to instances, performs coarse spatial quantization for feature extraction.

This problem was fixed by the proposal of *RoIAlign* in *Mask RCNN*, a simple, quantization-free layer that faithfully preserves exact spatial locations. Mask RCNN extends Faster RCNN by adding a branch for predicting segmentation masks on each Region of Interest, in *parallel* with the existing branch for classification and bounding box regression and improves the accuracy a lot. In our experiment, apart from training the Faster RCNN with pretrained backbone on ImageNet, we have also trained the network with trained backbone of Mask RCNN on COCO, to see if the backbones of the two network have some impact on the detection results. Besides, we have also trained the network with all the parameters randomly initialized. Comparing the three ways of training, we discovered that pretrained weights do accelerate the training process.

2.1.1 Faster RCNN

A Faster RCNN network takes an entire image as input and generate a feature map with several convolutional and max pooling layers. Then the Region Proposal Network outputs a set of proposals each with an objectness score and the offsets towards the ground truth bounding box. After that, for each object proposal a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map. Each feature vector is fed into a sequence of fully connected layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a "background" class and another layer that outputs four numbers for each of the K object classes. Since the RPN shares the full-image convolutional features with the detection network, it is almost cost-free to generate region proposals.

However, there is still lots of shortcomings of the Faster RCNN. For example, the two rounding of *RoIPooling* causes the loss of accuracy, which is solved by the proposed *RoIAlign* in the subsequent Mask RCNN.

2.1.2 Mask RCNN

Mask RCNN is a conceptually simple, flexible, and general framework for object instance segmentation. This method efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. To fix the misalignment in Faster RCNN, the *RoIAlign* in Mask RCNN preserves exact spatial locations. Despite being a seemingly minor change, *RoIAlign* has a large impact, showing bigger gains under stricter localization metrics. Moreover, instead of generating just one feature map like in Faster RCNN, Mask RCNN uses a Feature Pyramid Network (RPN) to extract multi scale features, which can improve the detection accuracy of small objects.

RoI Pooling is a standard operation for extracting a small feature map from each RoI. *RoIPooling* first quantizes a floating-number RoI to the discrete granularity of the feature map, this quantized RoI is then subdivided into spatial bins which are themselves quantized, and finally feature values covered by each bin are aggregated. These quantizations introduce misalignments between the RoI and the extracted features. Although this may not influence classification a lot, which is robust to small translations, it has a great negative effect on bounding box regression.

Therefore, we suppose the change from *RoIPooling* to *RoIAlign* may improve the accuracy of bounding box regression, which has been verified in our experiments.

2.2 Experiments

2.2.1 Experiment Settings

In our experiment, we use the training data and validation data of VOC2007 as our training set, and the testing data of VOC2007 as our validation set. When testing the model, we use the testing data of VOC2012 as our testing set. When training the network, we first freeze the backbone and fine-tuning for 30 epochs, and then unfreeze the backbone and keep training for 40 epochs except the randomly initialized model. Considering that the learning rate would become too small as the number of epochs goes up, we turnde up the learning rate every 30 or 40 epochs. The whole learning rate during the training process is shown in Figure 15.

Other detailed settings are as follows:

Freeze stage:

· Batch size: 4

· Iteration: 2505 iterations when training and 246 iterations when validation.

· Optimizer: Adam.

· Weight decay: 0.0005

· Initial learning rate: 0.0001

· Learning rate decay: $lr = \lambda \times lr$ ($\lambda = 0.96$, updates every one epoch)

Unfreeze stage:

· Batch size: 2

· Iteration: 2505 iterations when training and 2476 iterations when validation.

· Optimizer: Adam.

· Weight decay: 0.0005

· Initial learning rate: 0.00001

· Learning rate decay: $lr = \lambda \times lr$ ($\lambda = 0.96$, updates every one epoch)

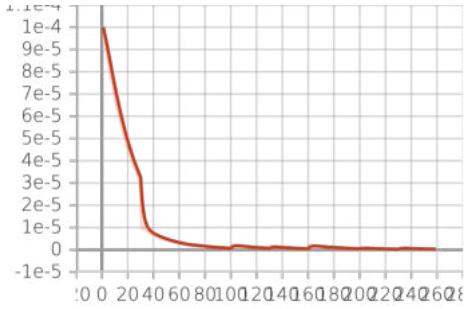
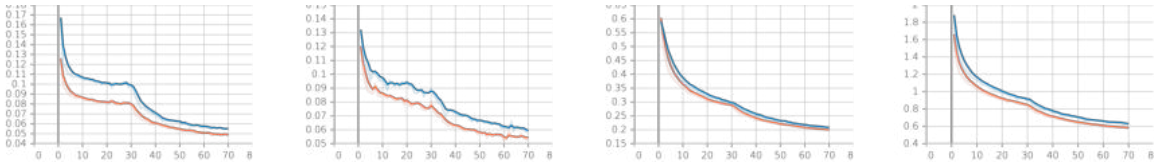


Figure 15: learning rate



(a) rpn_cls_loss (b) rpn_cls_loss (c) roi_cls_loss (d) roi_loc_loss

Figure 16: rpn_loss and roi_loss

2.2.2 Comparison Between the Two Pretrained Training Process

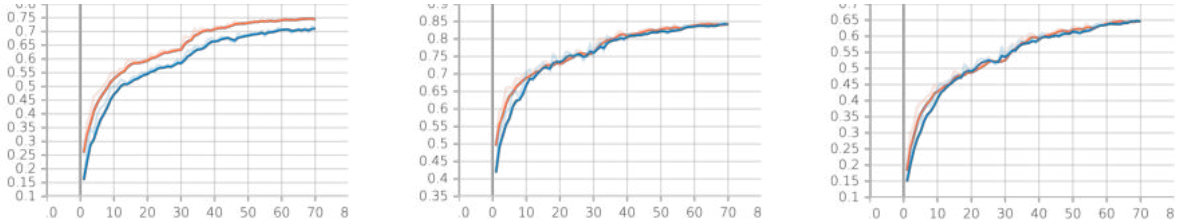
For the training with pretrained backbone, we have used two different pretrained weights, one of which was trained on ImageNet, and the other was the trained backbone of Mask RCNN on COCO. Both of the weights were downloaded from Pytorch's official website. Since our code can not load the weights trained by Mask RCNN directly, we have matched the corresponding layers manually, and the new weights has been uploaded to Google Drive as **mask.pth**.

As we can see in Figure 17, both of the loss when training and validation of COCO-based-model are lower than ImageNet-based-model, and reach a lower loss eventually. As for the three evaluation metrics along the training process (shown in Figure 18), acc and mIOU of the two model have little differences, but the mAP of the COCO-based-model

is much higher than the other. If we have a detailed look at the four components of the total loss (shown in Figure 16), we will find that the COCO-based-model mainly increases the accuracy of classification and bounding box regression in Region Proposal Network, which is the upstream module in the network.

(a) train_loss (b) val_loss

Figure 17: train_loss and val_loss



(a) mAP@.5 (b) acc (c) mIOU

Figure 18: the comparison of mAP@.5, acc and mIOU between the two ways

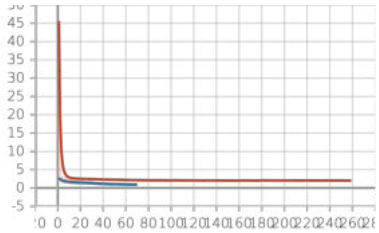
As we have mentioned above, Mask RCNN replaces the RoIPooling in Faster RCNN with RoIAlign, which is pixel-to-pixel alignment between the inputs and outputs, thus have a higher accuracy of bounding box regression. Among the three evaluation metrics, mAP is the most suitable measurement of the bounding box regression, and that's why the COCO-based-model improves the mAP significantly while the other metrics almost stay the same.

As the experimental results have shown to us, although the pretrained weights on ImageNet is an excellent module for extracting features in an image, a more task-related pretrained weights may have the ability of extracting better features for the specific task.

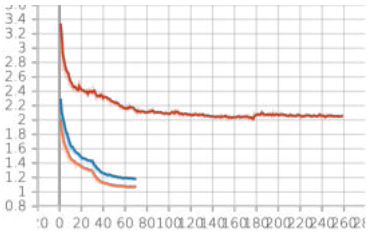
2.2.3 Comparison Among the Three Training Process

As we have shown above, those two ways of training the model using pretrained backbones did not have much difference in general. But if we compare them with the randomly initialized training process, we can find that the randomly initialized model learned very slow. As shown in Figure 20, the randomly initialized model needed about 260 epochs to reaches the effect of the pretrained models which wss just trained one epoch. Besides, since the initial parameters are very bad, the total loss is very high at the beginning but become better after just a few epochs. Obviously, we can derive that the pretrained backbone speeds convergence of the network a lot. But, whether the network can reach a good level if all of the parameters are randomly initialized?

If we view the convergence of the randomly initialized mode separately, we can find that the slopes the curves are still large relatively (shown in Figure 21), which indicates that the network is still in a learning state, and is sure to learn better given enough time, though the speed is not that impressive. Therefore, the pretrained weights are indispensable when training a network.

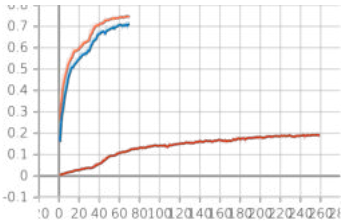


(a) train_loss

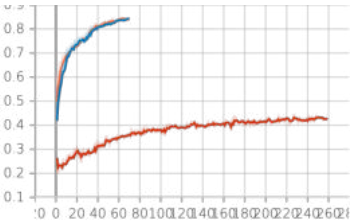


(b) val_loss

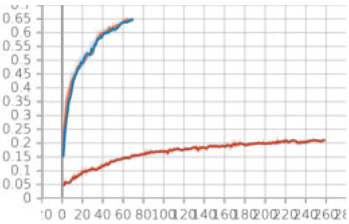
Figure 19: train_loss and val_loss



(a) mAP@.5



(b) acc



(c) mIOU

Figure 20: the comparison of mAP@.5, acc and mIOU between the three ways

2.2.4 Results

We have selected the best model for each of the three training ways, and tested them on the testing set. On the whole, the COCO-based-model performed best, exceeding 0.027 than ImageNet-based-model in mAP and just falling behind 0.012 and 0.006 in mIOU and acc respectively (shown in Table 1).

2.3 Conclusion

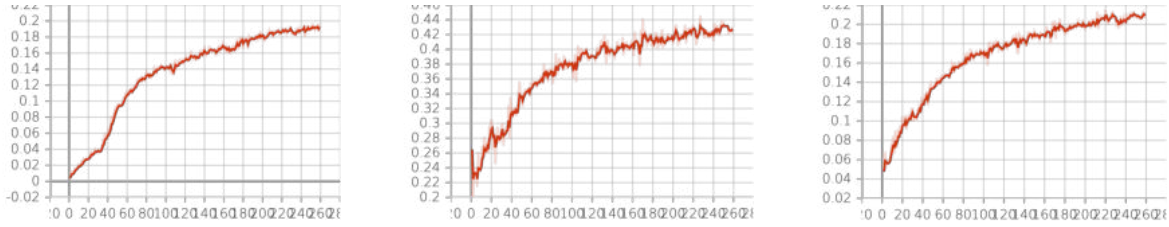
Compared to the random initialization, a pretrained model can indeed learn faster, even reach the level that the former can not reach in some cases. However, although a pretrained model like a ResNet on ImageNet do have a generalization ability, and is useful for extracting the features in many tasks, a more task-related pretrained model might learn faster or even reach a better performance.

3 Vision Transformer

3.1 Introduction

Having long been dominated by convolutional layers, the field of computer vision finally witnessed a revolution since the birth of vision transformer in the autumn of 2020 (8). Motivated by the work by Vaswani et al. (9) in the field of natural language processing, it translates the network to fit the graphical input and has achieved satisfying results. It innovatively mimics the attention of a human by learning the weight assigned to the input. Previous works include Squeeze-and-Excitation net (10) which can be traced back to the time when the attention module was proposed (9), where an additional attention adjusting path is attached to the network. Yet the vision transformer, as will explained in this paper, jumps out of the conventional CNN architectures.

Regardless of its recent rapid development, vision transformer seems to rely on very large datasets and even ImageNet (11) where one million images are available is oftentimes insufficient. Overfittings and poor generalization ability if trained from scratch with small datasets are commonly reported (12), not to mention the CIFAR-100 dataset (13) where there are only $50,000$ samples to learn from.



(a) mAP@.5 (b) acc (c) mIOU

Figure 21: the mAP@.5, acc and mIOU of the randomly initialized model

(a) train_loss (b) val_loss

Figure 22: train_loss and val_loss

In this paper we will elaborate the architecture of a vision transformer and will conduct experiments on CIFAR-100 without any pretraining or transfer learning. Details on overfitting and the benefit of data augmentation will get analyzed and comparisons between vision transformer and ResNet (3) are made. In the last we will introduce the visualization of the attention map generated by vision transformers.

3.2 Network Architecture

An outline of a vision transformer is showcased in Figure 23. Each input image is first partitioned into grids, known as patches. Then, to each patch a value is assigned according to the pixels inside. These two steps, in both our implementation and in (8), are jointly achieved by a stride convolution with stride equal to the kernel size. Now we flatten out those patches in each channel to obtain a matrix with size $P \times C$ where P stands for the number of patches while C for channels. Then a class token is concatenated to each channel, represented by a single learnable scalar each channel. As a consequence, we obtain a matrix of size $(P + 1) \times C$.

3.2.1 Positional Encoding

Before feeding the $(P + 1) \times C$ matrix into the encoders, one of the most essential parts of our vision transformer is that we shift the matrix by a bias. In our implementation, it is a learnable $P \times C$ vector added to the matrix. This is known as the positional encoding. Positional encoding might have various possible forms, and can even be some fixed parameters (8)(14).

3.2.2 Multihead Attention

We place the introduction of multihead attention module ahead of the encoder, as it serves as an important component. It is a popular architecture proposed in (9). Inspired by the human's attention, the layer redistribute the weight on each pixel, each representing the attention on the pixel. Higher ones result in larger intervention and would receive stronger feedback in the backward pass.

In a attention module, the input is first tripled by a linear layer and is then divided evenly into three parts, queries, keys and values. Intuitively, values are the embedding of the input and has similar intrinsic pattern. Queries are the

model	mAP@.5 (M)	mIOU (M)	accuracy
ImageNet-based	0.649	**0.608**	**0.806**
COCO-based	**0.676**	0.596	0.800
Random-Initialized	0.125	0.161	0.356

Table 1: The comparison of the evaluation metrics among the three different training ways

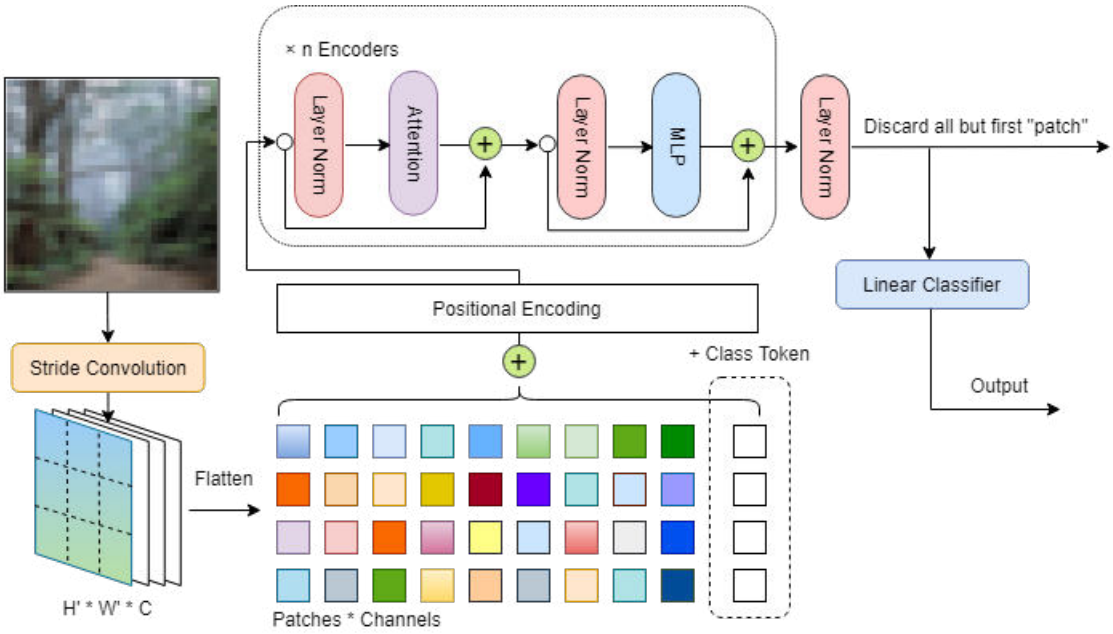


Figure 23: An outline of the architecture of a standard vision transformer.

target of our classification, which together with keys will determine our attention on the input. Formally, the query and key is multiplied and scaled, followed by a softmax. This we obtain the so-called attention of the values. The output is thus a multiplication of the attention and the values. As suggested in (9), the scaling factor is $1/\sqrt{d}$ where d equals to $C + 1$ in a vision transformer, the dimension of the channels plus the class token, i.e.

$$\text{output} = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V. \tag{1}$$

Multihead attention is a slight modification of the standard attention module. Rather than dividing the transformed input into three parts, it is divided into $3N$ parts, N pairs of queries, keys and values. Each pair is interpreted as a head. All the heads perform the process stated above in a parallel manner and the output is the concatenation, see Figure 24.

3.2.3 Encoder

Having included the positional encoding, the data go through n encoders. Every encoder is built up by four layers, a layer normalization, a multihead attention, another layer normalization and finally an MLP. The MLP, standing for multi-layer perceptron, consists of two linear embeddings and an activation function in between. Also, following the idea of ResNet, the inputs are added to after the attention layer, and the addition is further added to the result of the MLP, see Figure 23.

In the last we send the result to layer normalization (15) and drop the last but first patch so that the input becomes a batch of feature vectors. The vector goes through the final ‘ linear classifier to attain the predicted probability for the classes.

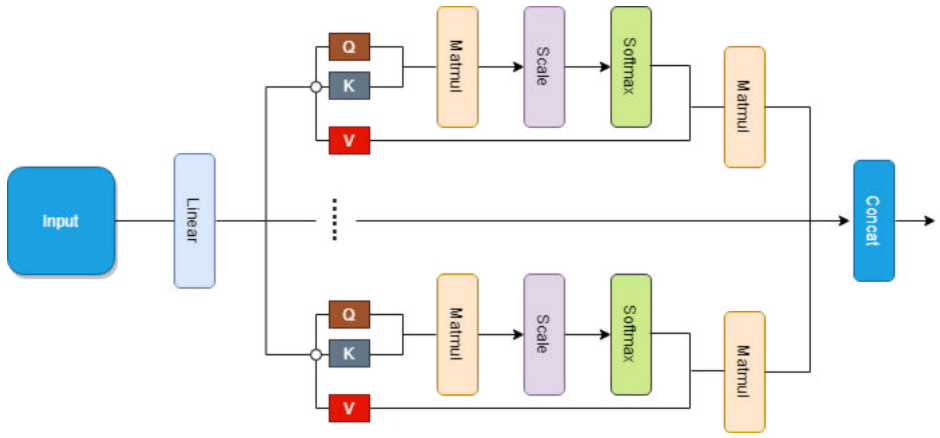


Figure 24: Illustration of a multihead attention module. Letters 'Q', 'K', 'V' stand for queries, keys and values respectively.

3.2.4 Dropout

We have omitted the dropout in the explanation above and in Figure 23 for brevity. If one is willing to add in dropout layers, they can be placed after the positional encoding, the multihead encoders or each linear layer in the MLPs. Stochastic depth strategy (16) can be also adapted to those encoders.

3.2.5 ResNet VIT

An evolution of the standard vision transformer is generally known as ResNet VIT where "VIT" is short for vision transformer. It is inspired by the ResNet (3), an architecture that has presented robust feature-capture capability. With the knowledge that positional encoding as bias is powerful on transformer, as will illustrated in the experiment section, one would question that whether other transforms on patches would help. The answer is yes by ResNet. It is implemented by placing ResNet blocks in the front of the transformer, downscaling a 224×224 image to rather small size, say, 14×14 after 4 stages. Then the vision transformer fetches the patches from this feature map. The CIFAR-100 images should be first resized to 224×224 or persumably greater to support the aggressive downscaling.

3.3 Experiments

3.3.1 Dataset

We have carried out all our experiments on the CIFAR-100 dataset (13). It contains 50000 images for training and 10000 for testing. Each image is categorized into one of a hundred classes, and has resolution 32×32. Common metrics include the top 1 accuracy and the top 5 accuracy.

3.3.2 Experiment Settings

We use Adam as optimizer and apply the linear learning rate warm-up scheduler with 780 steps. No further learning rate decay strategy is applied after the learning rate gets fixed at 3×10^{-4} at the end of warm-up. The transformers are trained with batch size 256 and up to 200 epochs. All models are trained from scratch.

For data augmentation we have applied the method of CutMix (17) and color jitter (18). The CutMix is a simple cut-and-paste strategy that merges two images in one. The color jitter is randomly adjusting the brightness, contrast, saturation and hue of the image.

For table 2, every vision transformer has input size 32×32, 192 channels and a size of 768 for the hidden layers in MLPs. There are 3 heads in the multihead attention modules and 12 encoders in total. No ResNet modules are introduced and no dropout is used.

3.3.3 Accuracy and Loss

As far as we are concerned, traditional vision transformer face severe overfitting on the small dataset CIFAR-100. Despite its unfruitful performance on the testing data, it, in most cases, exhibit no difficulty reaching a $\mathbf{99}\%$ top 1 accuracy on the training data when trained sufficiently. The overfitting on CIFAR-100 of so small transformers indicates no necessity in working with larger ones.

Table 2 elaborates the training results. With over 99% on the training data, the testing accuracy varies with the augmentation. Composed, complicated augmentation, especially CutMix plus color jitter, outperforms any other.

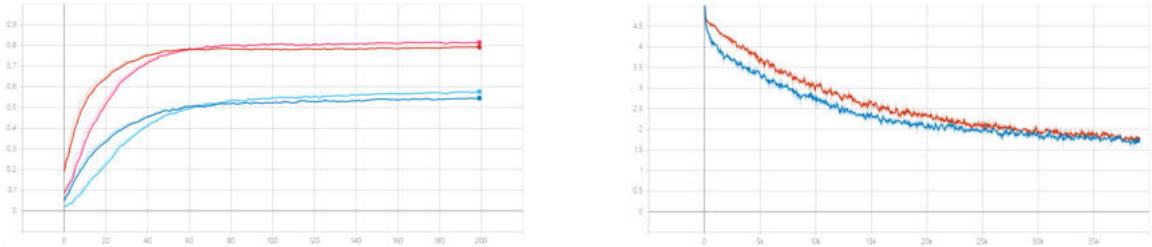
Positional encoding is also essential for the vision transformer. It induces similar outcomes as CutMix, pushing the accuracy by 10% or more. But CutMix and positional encoding are parallel, meaning that their combination benefits from both. Relatively larger patch number also gives slight improvement, lifting the top-1 accuracy by around 2% to 3%.

Patch Num	Data Aug	Pos Encoding	Top-1 Acc (%)	Top-5 Acc (%)
64	baseline	✓	43.78	71.77
64	CutMix	×	39.72	69.40
64	CutMix	✓	53.13	78.33
64	jitter	✓	40.13	69.08
64	CutMix+jitter	✓	54.86	79.69
256	baseline	✓	42.22	71.84
256	CutMix	×	44.54	71.16
256	CutMix	✓	55.53	79.91
256	CutMix+jitter	✓	**57.85**	**81.89**

Table 2: Training result with identical transformer architecture but different settings.

The accuracy curve as well as the training loss also signals overfitting. Displayed in Figure 25 are two examples of training with CutMix and CutMix plus jitter respectively. Less robust augmentation, purely CutMix, leads to faster convergence on the training loss and leads the testing accuracy in the beginning of training. Yet it suffers more serious bottleneck and is likely to be surpassed by CutMix plus jitter further on.

Moreover, in spite of the slow growth in testing accuracy from around 70 epochs, the training losses still witness stable decays. It indicates that the model is now heading towards the training data rather the real world.



Testing Accuracy Training Loss

Figure 25: Stuck testing accuracy in contrast to steadily decaying training loss experimented with CutMix and CutMix + jitter. Left: Blue lines are for top-1 while red for top-5. Light-colored ones that win at the end are for CutMix + jitter. Right: Red line represents the CutMix + jitter while blue for pure CutMix. Loss recorded every 50 iterations.

3.3.4 Different Input Size

We have experimented with 32×32 and 224×224 input size and see no noticeable improvement for the standard vision transformers. We believe that it is due to the patch separation in the front. Even though a CIFAR-100 image is resized to 224×224, the interpolation included in the resizing operation is canceled by the projection. Hence working with images with images at resolution 32×32 and 224×224 ones behaves similar, if no extra details are added. We

shall clarify that this is the case of standard vision transformer, while the rule does not apply on the mutation with ResNet structures.

3.3.5 Comparsion with ResNet

We have also made comparsions with ResNet (3) on CIFAR-100. We have merely used the ResNet-18, which is fairly enough to beat the classical vision transformers, ones that do not include ResNet blocks in the front. All networks in this section are trained from scratch and only CutMix is introduced in data augmentation. We have trained ResNets with resolution 224×224 and 32×32 as for comparisons.

Two standard vision transformers are taken into account, of which the smaller one contains 192 channels, 768 hidden size, 3 heads and 12 encoders. The larger one has 240 channels, 1464 hidden size, 6 heads and 12 encoders to match the parameter size of the ResNet-18.

For ResNet-VIT, we use $(3, 4, 9)$ residual blocks for three ResNet stages. The CIFAR-100 images are first resized to 224×224 and downsampled by the ResNet backbones by $2^4\times$ to 256 channels of 14×14 feature maps. The patch number for the vision transformer is set to $14 \times 14 = 196$.

Table 3 has shown that architectures with residual blocks perform impressively better than those without. ResNet-18 trained on 224×224 inputs requires the most forward flops, meanwhile the least forward and backward pass size. Vision transformers seem to have small size of learnable parameters and cute forward flops, but they demand more forward and backward pass size. There is little difference in small and large VITs. Also, when trained on 32×32 images, ResNet-18 has the fewest parameters but rivals the result of VITs.

Network	Params (M)	Flops (M)	Pass Size (MB)	Top-1 Acc(%)	Top-5 Acc (%)
ResNet-18 (224)	11.23	1800	39.75	**72.63**	**91.57**
ResNet-18 (32)	2.74	10.93	1.53	52.07	78.10
VIT-small	4.97	5.55	48.16	53.13	78.33
VIT-large	10.64	11.37	72.63	53.47	78.48
ResNetVIT	6.47	550	82.22	63.17	86.23

Table 3: Comparison of different networks on CIFAR-100. Column 'flops' only include multi-adds in the forward pass. Column 'pass size' stands for the auxiliary variables generated in the forward and the backward pass.

3.4 Visualization

The visualization of the attention map of the attention modules follows the process in Figure 26 and is inspired from (19). Recall that we in each encoder generate attention weights by a softmax layer, they are extracted and then averaged along the head axis. We have canceled down the head dimension in this step and obtain a matrix W of size $(P+1) \times (P+1)$ where P is the number of patches for each input. We shift the matrix in the diagonal by identity, $W \leftarrow W + I$, which guarantees the positiveness of W by the property of nonnegative matrices. Then we normalize W so that every row of W sums up to 1 by an entrywise normalizer S. Multiple Ws from all encoders are multiplied in order to yield an accumulated weight $\mathbf{W}$.

$$\mathbf{W} = \prod \left(\left(\frac{1}{N} \sum \operatorname{softmax} \left(\frac{QK^T}{\sqrt{d}} \right) + I \right) \odot S \right) \tag{2}$$

Since we only preserve the first patch in the forward pass, we discard the 2 to $P+1$ rows in $\mathbf{W}$ similarly. Also, the class token should be removed and now the $\hat{\mathbf{W}}$ is a vector of length P. To restore the image size, entries of $\hat{\mathbf{W}}$ are mapped to $[0, 1]$ and it is reshaped to $\sqrt{P} \times \sqrt{P}$. Finally, we inverse the step of patch separation and resize it back to the original size. This can be implemented by a transposed convolution.

One of the simplest ideas of visualizing the attention map is the direct heatmap plots. To combine it with the original image, one can try out applying an entrywise multiplication between the image and the attention map. In this

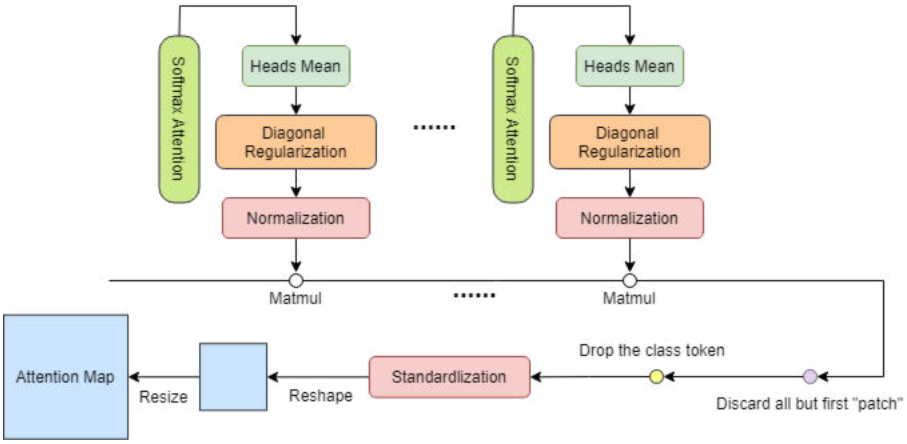


Figure 26: The outline of the visualization of attention map in vision transformer.

case, minor attention scores correspond to low RGB values, which turns out to be dark and masked in vision. It is also suggested that we place the attention map on the alpha channel of the input image as an adjustment of the transparency where focused parts remain solid and opaque.

Figure 27 illustrates 6 randomly chosen samples from CIFAR-100 training dataset generated by a 256-patch vision transformer. The attention seems to have served its purposes. For example, in the second column where the attention of a lawn mower is displayed, bright colors concentrate on the entity. The plain showcased in column 4 is also an interesting example, where we learn from the attention map that the focus of the network lies on the land rather sky.

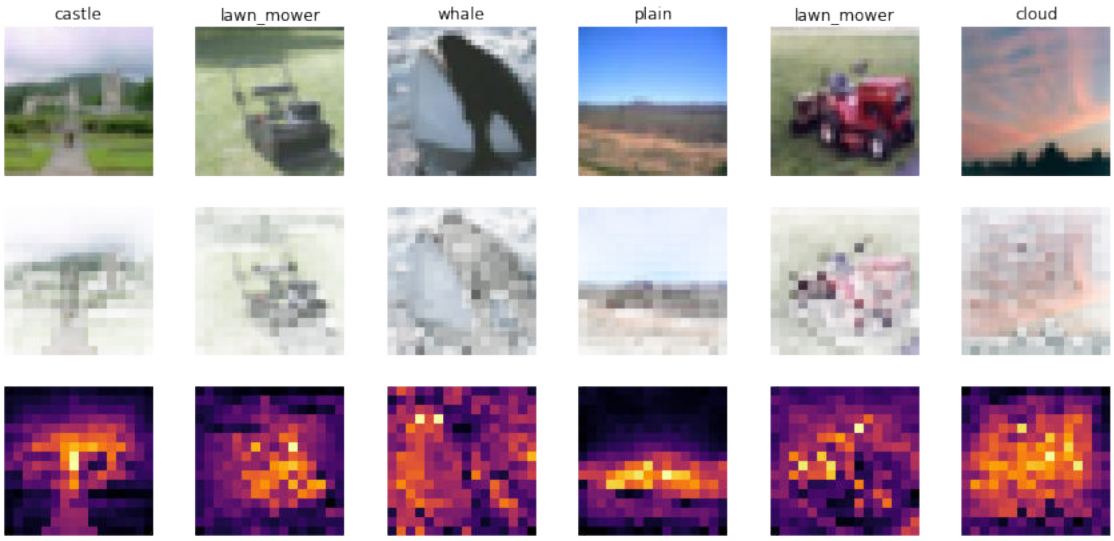


Figure 27: Attention maps for 6 CIFAR images, each partitioned into 16×16 patches. The first row displays the original images. The second row displays the attention-guided images by adjusting the alpha channel accordingly. The third row illustrates the attention maps in inferno theme, where brighter colors stand for larger attentions.

3.5 Conclusion

On the one hand, the vision transformer shows poor generalizing ability on the CIFAR-100 dataset if trained from scratch. On the other, it reveals astonishing fitting ability on the training data. The importance of the data augmentation

highlighted in the experiment implies that, the transformer could persumably unleash its power on larger datasets and stronger data augmentation.

3.6 Future Work

Apart from simply resorting to larger datasets or pretraining at a high expense, there are other potential treatments. Data augmentation, as we have shown, displays good effect on boosting the performance. Semi-supervising or knowledge distillation might also help, see (20). The fact that ResNet VIT generalizes better suggests the exploration of new network architectures.

References

[1] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *IEEE Computer Society*, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia, "Psanet: Point-wise spatial attention network for scene parsing," *European Conference on Computer Vision*, 2018.

[5] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017.

[6] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," no. 4, 2016.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[10] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *CoRR*, vol. abs/1709.01507, 2017.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[12] X. Chen, C. Hsieh, and B. Gong, "When vision transformers outperform resnets without pretraining or strong data augmentations," *CoRR*, vol. abs/2106.01548, 2021.

[13] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[14] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *CoRR*, vol. abs/1705.03122, 2017.

[15] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[16] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 646–661, Springer International Publishing, 2016.

[17] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *CoRR*, vol. abs/1905.04899, 2019.

[18] A. Howard, "Some improvements on deep convolutional neural network based image classification," 12 2013.

[19] H. Chefer, S. Gur, and L. Wolf, "Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 387–396, 2021.

[20] R. Liu, K. Yang, H. Liu, J. Zhang, K. Peng, and R. Stiefelhagen, "Transformer-based knowledge distillation for efficient semantic segmentation of road-driving scenes," 02 2022.

[21] Z. Dong, K. Xu, Y. Yang, H. Bao, W. Xu, and R. W. H. Lau, "Location-aware single image reflection removal," *CoRR*, vol. abs/2012.07131, 2020.

A Additional Materials

Besides having posted our codes at https://github.com/mskmei/FINAL-PROJECT-CV-2022Spring.git, already-trained models in this project are available at:

1. Semantic Segmentation

   (1) PSPNet (PSP101 pretrained on Cityscapes)

   https://drive.google.com/file/d/1KaM0XLX60awJ6VzEPel84jr5AouNoLxi/view?usp=sharing

   (2) PSANet (PSA101 pretrained on Cityscapes)

   https://drive.google.com/file/d/1qqktI7aIEM4Vucqk7XVi-L2FFDLJEQtO/view?usp=sharing

   (3)Deeplabv3 (pretrained on Cityscapes)

   https://drive.google.com/file/d/1XlP8CzbkVkv8UZ2f6_0wtCT8P0Pu4nmD/view?usp=sharing

2. Faster RCNN

   (1) Model with ImageNet pretrained

   https://drive.google.com/file/d/1Ujds2mvsNLc8cXHH6827S-K_SfmV0Ssg/view?usp=sharing

   (2) Model with COCO pretrained:

   https://drive.google.com/file/d/1icZWvFdUXmKHTi719KJD_4gbl-Y73DUI/view?usp=sharing

   (3) Model with random initialization:

   https://drive.google.com/file/d/1BNturJuVtnZBV4J_pvCtrUSJeZrJM7aS/view?usp=sharing

3. Vision Transformer

   https://pan.baidu.com/s/1Fkca1pHfBrN6YPAKHE8oZw?pwd=l56w

Furthermore, videos obtained by semantic segmentation for the first task are available at:

1. Original video(25 seconds, downloaded from bilibili)

   https://drive.google.com/file/d/1kx3J8dYsMFggJBKlFWBUie5j3kZ397_k/view?usp=sharing

2. Result videos are available in the directory:

   https://drive.google.com/drive/folders/1pVmLX6EhfBikLpH_pUjXWYzlMmM7s8Ny?usp=sharing

   Explanation: There are 2 videos for each model so there are 10 videos in total. The suffixes "_psp", "_psa" and "_deep" represent the three models of PSPNet, PSANet, and Deeplabv3, respectively. The suffix name "_mix" indicates that the video is a mixture of the semantic segmentation result and the original video (at a ratio of 7:3), and no "_mix" suffix name indicates the result of semantic segmentation (composed of only color blocks).

B Datasets

B.1 Cityscapes

Cityscapes is a large dataset focused on semantic segmentation concerning urban scene. It provides semantic, instance, and dense pixel annotations for 30 categories divided into 8 categories (flat surfaces, humans, vehicles, constructions, objects, nature, sky, and void). The dataset consists of about 5000 finely annotated images and 20000 coarsely annotated images. Additionally, data was collected in 50 cities in good weather conditions. It was originally recorded as a video, then the frames were manually selected to have the following characteristics: a large number of dynamic objects, different scene layouts, and different backgrounds.It has become one of most famous dataset in the field of semantic segmentation.

The Cityscapes dataset is mainly intended for

1. assessing the performance of vision algorithms for major tasks of semantic urban scene understanding: pixel-level, instance-level, and panoptic semantic labeling;
2. supporting research that aims to exploit large volumes of (weakly) annotated data, e.g. for training deep neural networks.

In this project, we won't use Cityscapes to train our model, but use models pre-trained on Cityscapes to test and analyze.

B.2 VOC2007&2012

The PASCAL VOC dataset(21) is one of the most commonly used standard datasets in the field of target detection. It can be divided into 4 categories and 20 subcategories in terms of categories. The category information is shown in the Figure 28. The official of PASCAL VOC published a series of datasets from 2005 to 2012, among which VOC2007 and

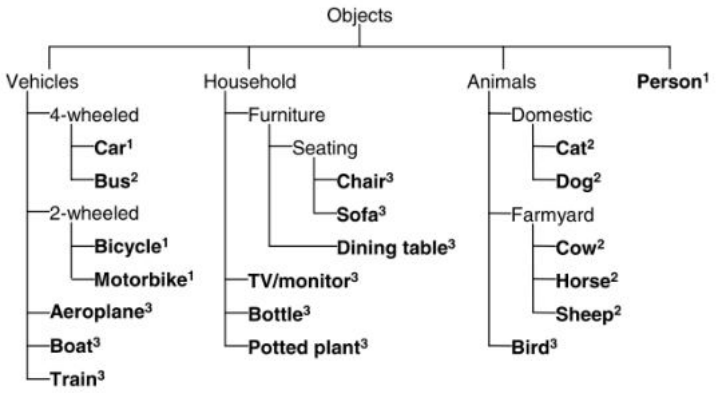


Fig. 2 VOC2007 Classes. Leaf nodes correspond to the 20 classes. The year of inclusion of each class in the challenge is indicated by superscripts: 2005[1], 2006[2], 2007[3]. The classes can be considered in a notional taxonomy, with successive challenges adding new branches (increasing the domain) and leaves (increasing detail)

Figure 28: PASCAL VOC Dataset

VOC2009 turned to be more well-known. That's because VOC2007 and VOC 2012 are mutually exclusive. VOC2012 includes all the images from VOC2008 to VOC2011 but has nothing to do with VOC2007. Therefore, Training and testing with these two datasets has been universally acknowledged as a superb approach.

In this project, we will use the training and validation datasets of VOC2007 as a new training set, use test dataset of VOC2007 as our new validation set, and finally use the validation dataset of VOC2012 as our new test set. Anyway, we

must ensure that there will be no interation among our training, validaiton and test sets.The following table will visually show you the split of our train-validation-test set.

Our set(No interaction)	Training set	Validation set	Test set
Corresponding set	VOC2007 training set and validation set	VOC2007 test set	VOC2012 validation set

Table 4: The split of train-validation-test set based on VOC2007 and VOC2012

B.3 COCO

The Common Object in Context (COCO) is one of the most popular large-scale labeled image datasets available for public use. It represents a handful of objects we encounter on a daily basis and contains image annotations in 80 categories (shown in Table 5), with over 1.5 million object instances.

person	fire hydrant	elephant	skis	wine glass	broccoli	dining table	toaster
bicycle	stop sign	bear	snowboard	cup	carrot	toilet	sink
car	parking meter	zebra	sports ball	fork	hot dog	tv	refrigerator
motorcycle	bench	giraffe	kite	knife	pizza	laptop	book
airplane	bird	backpack	baseball bat	spoon	donut	mouse	clock
bus	cat	umbrella	baseball glove	bowl	cake	remote	vase
train	dog	handbag	skateboard	banana	chair	keyboard	sciccors
truck	horse	tie	surfboard	apple	couch	cell phone	teddy bear
boat	sheep	suitcase	tennis racket	sandwich	potted plant	microwave	hair drier
traffic light	cow	frisbee	bottle	orange	bed	oven	toothbrush

Table 5: COCO Classes

The COCO dataset is used for multiple CV tasks:

- **Object detection and instance segmentation**: COCO's bounding boxes and per-instance segmentation extend through 80 categories providing enough flexibility to play with scene variations and annotation types.
- **Image captioning**: the dataset contains around a half-million captions that describe over 330,000 images.
- **Keypoints detection**: COCO provides accessibility to over 200,000 images and 250,000 person instances labeled with keypoints.
- **Panoptic segmentation**: COCO's panoptic segmentation covers 91 stuff, and 80 thing classes to create coherent and complete scene segmentations that benefit the autonomous driving industry, augmented reality, and so on.
- **Dense pose**: it offers more than 39,000 images and 56,000 person instances labeled with manually annotated correspondences.
- **Stuff image segmentation**: per-pixel segmentation masks with 91 stuff categories are also provided by the dataset.

B.4 CIFAR-100

CIFAR-100, introduced in (13), is a widely used dataset in the field of computer vision. It contains $60,000$ natural images categorized in 100 different classes, $50,000$ of which are treated as training data, while the rest of which are for testing. All the images have resolution 32×32 in RGB format. Commonly employed metrics include the top-1 accuracy and the top-5 accuracy.