Semantic Image Segmentation

Hao Lan

May 2023

Abstract

This work is inspired by the Spatial Structure Preserving Feature Pyramid Network (SSPFPN) paper, which proposes an approach to semantic image segmentation that leverages spatial structure information. The objectives of this project are not just to explore and implement the SSPFPN model on the PASCAL VOC 2012 dataset [EVGW$^+$] for semantic segmentation, but other popular methods widely used in the field.

The SSPFPN model has shown promising results in preserving spatial information and improving segmentation accuracy in comparison to other feature pyramid networks. Our implementation of the SSPFPN model involves adapting the architecture and training process for the specific dataset and task at hand. Due to the computational limitations, I might not be able to fully implement the proposed loss term.

Beyond the SSPFPN model, we will also explore other popular models for semantic segmentation such as FCNs, DeepLab, and ResNet plus UNet. My goal is to provide a comprehensive analysis and comparison of the different models and their performance on the PASCAL VOC 2012 dataset, for DeepLabV3+ another dataset CIHP [GLL$^+$18] will be introduced as well. This research mainly focus on studying how semantic segmentation works and what are the components that make semantic segmentation effective. I will introduce more on the SSPFPN method, but more importantly, learn the terminology and concepts of semantic segmentation from that specific method proposed.

1 Introduction

1.1 Background

Fully Convolutional Networks (FCNs) are a popular approach for semantic image segmentation, where the goal is to classify every pixel in an image into a specific class. FCNs achieve this by using convolutional layers to extract features from the input image and then upsampling the feature map to the original image size to get the pixel-level prediction. However, early versions of FCNs were limited by their inability to capture spatial information, leading to coarse segmentations with blurry boundaries. To address this, researchers have proposed various modifications to the original FCN architecture.

One such modification is the Spatial Structure Preserving Feature Pyramid Network (SSPFPN), proposed by Yuan et al.[YFLF19] in 2019. SSPFPN improves on FCNs by preserving spatial structures in feature maps at different resolutions, enabling more accurate segmentation. Specifically, SSPFPN incorporates skip connections between encoder and decoder layers to maintain spatial information and utilizes a pyramid pooling module to aggregate multi-scale features. The resulting architecture achieves state-of-the-art performance on various benchmark datasets, including the PASCAL VOC12 dataset.

1.2 SSPFPN Method

Inorder to preserve pixel-paired-based similarity, the paper proposed a novel Loss function:

$$L = \gamma \cdot L_c + (1 - \gamma) \cdot L \tag{1}$$

where Lc is the classification loss, which mainly ensure the correct predicted category label for each pixel, and L is the proposed term that emphasizes the pixel-pair-based similarity in the image and it is set to 0.85 as initial state.

Speak to loss L, which is the cosine distance between 2 affinity matrix calculated from the input image and the prediction label map generated by the model.

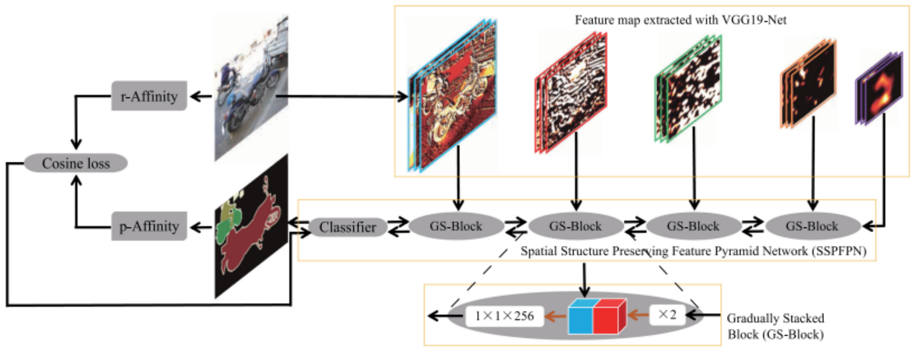
Feature map extracted with VGG19-Net
r-Affinity
Cosine loss
p-Affinity
Classifier
GS-Block
GS-Block
GS-Block
GS-Block
Spatial Structure Preserving Feature Pyramid Network (SSPFPN)
1×1×256
×2
Gradually Stacked Block (GS-Block)

Figure 1: The SSPFPN structure

1.2.1 bottom-up approach

In this implementation, I will use VGG19 as the backbone feature extractor for my FPN structures. We donate these last residual blocks as C1, C2, C3, C4, C5 for conv1, conv2, conv3, conv4, and conv5 outputs (as shown in Figure 1), and note that they have strides of 2, 4, 8, 16, 32 pixels with respect to the input image. Vgg19 layers are freeze, which gives 64,5653 trainable parameters and 20 million+ un-trainable parameters.

1.2.2 top-down approach

As GS-block shown in Figure 1, these features are fused with features from the bottom-up pathway using the proposed GS-Block, which is to keep the discriminative information and the structure details of the image simultaneously. In this process, 2 techniques are involved. One is skip connection, also as known

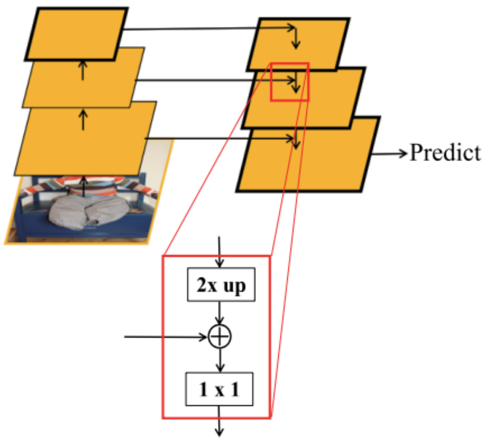

Figure 2: The bottom-up and top-down approach

as residual connection, this is achieved by concatenating the up-sampled the feature map with the previous bigger feature map on the layer below. Another is called lateral connection, which happens after concatenation, this technique is ahcieved by using a conv2D layer of size 1x1 filter, this conv2D layer will then refine the information contained in the original feature map across all channels.

1.2.3 spatial affinity matrix for input

Assume that the input images have the size of m × n × 3, the affinity matrix of the image will be the size of mn × mn. First, to enhance the robustness of the affinity matrix, C and L for each pixel are normalized to c and l. The normalized functions are shown in Equation (2) and Equation (3),

$$C->c:[r,g,b]=[R,G,B]/255 \tag{2}$$

$$L->l:[x,y]=\left[\frac{x}{m-1},\frac{y}{n-1}\right] \tag{3}$$

Then, we concatenate the c and l vector of each pixel as a new vector f = [c;l] and calculate the distance matrix $D^{mn,mn,5}$ using Equation (4):

$$\begin{aligned} \forall 0 \leq i,k \leq m-1,\ 0 \leq j,t \leq n-1, \\ D_{ni+j,nk+t,:} = |f_{i,j,:} - f_{k,t,:}| \end{aligned} \tag{4}$$

Finally, the affinity matrix Sr A is calculated using Equation (5),

$$\begin{aligned} \forall 0 \leq p,q \leq mn-1, \\ (SrA)_{p,q} = \exp\left(-\|D_{p,q,0:2}\|_1 - \lambda \|D_{p,q,3:4}\|_1\right) \end{aligned} \tag{5}$$

where λ is a hyperparameter to balance the the relationship of RGB intensity and the location information.

1.2.4 spatial affinity matrix for prediction

we also need the affinity information of the predicted masks. As we all know, the outputs of network are images class label masks, we reshape the label mask to a vector $M^{mn,1}$ and define an affinity matrix $S_{pA}^{mn,mn}$ as Equation (6):

$$S_{pA}^{(i,j)} = \begin{cases} 1 & \text{if } M_i = M_j \\ 0 & \text{if } M_i \neq M_j \end{cases} \tag{6}$$

Last, the cosine distance is give by

$$L = 1 - \cos(A_{rv}, A_{pv}) \tag{7}$$

where L is the component of loss function introduction earlier in this section, are A_{rV} and A_{pV} are flatten SrA and SpA affinity matrix. On the other hand, it's

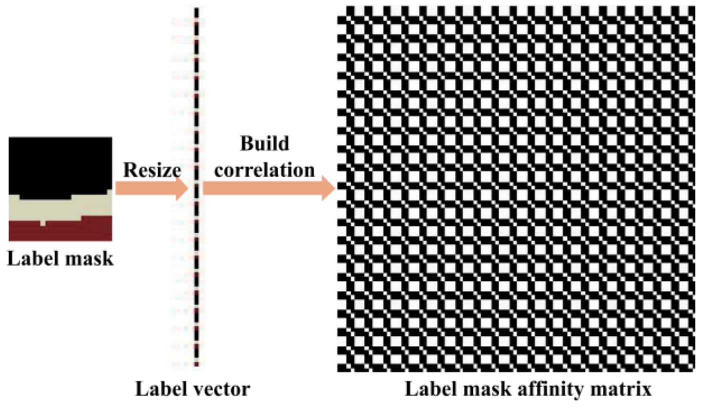

Figure 3: Affinity matrix of predicted label map

worth mention that creating mn x mn matrix will be heavily memory consuming, in the approach, each input image and predicted label maps will be separated into N x N pieces, each piece will be taken to calculate affinity matrix and we will take the mean amount all pieces. Unfortuanatly, due to this nature, my labtop was not able to apply the complete customized loss function in the model.

2 Implementation

To implement the SSPFPN (Spatial Structure Preserving Feature Pyramid Network) method, I have no sources code from the paper and build everything build the model from ground up. I started by using a pre-trained VGG19 as the encoder backbone. Then, I modified the decoder part of the network to create a feature pyrimid network structure by adding skip connections and lateral connections between the encoder and decoder blocks. In particular, I used decoder blocks with upsampling layers to restore the feature map size to the original input size using nearest neighbor up sampling. Finally, I trained the network using the VOC2012 dataset and evaluated the performance using various evaluation metrics, including pixel accuracy, mean accuracy, mean IoU, and frequency-weighted IoU. Overall, the implementation of SSPFPN allowed me to achieve state-of-the-art performance on the semantic segmentation task.

2.1 data processing

The PASCAL VOC12 data set is a well know challenge dataset first started in 2005. In it's 2012 generation, it contains 2913 images for semantic segmentation task, within 21 classes of objects including background, typicall classes are human, horse, bicycle...etc

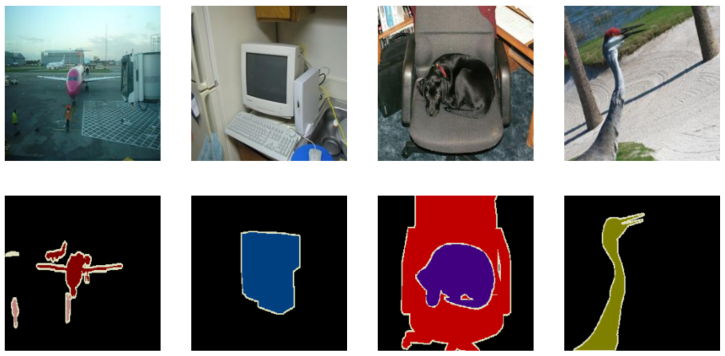

Figure 4: Samples from PASCAL VOC 2012 dataset.

By the limitation of may GPU RAM, I will only process 600 images from the data set, and 20% of them will be used as validation set with constant random state, which ensures that it will have the same split every time I restart the runtime. On the other hand, every image taken will be reshape to the same size of (224, 224, 3) with dtype of float64 for the sake of model integrity and generalization. Before training data and labels pairs are packed into ImageGenerator from Tensorflow as tensor type, I need to format the label maps in a correct way in order to let the model "understand".

2.1.1 Onehot encoder

One-hot encoding is a popular technique used to represent categorical data in a format that can be easily processed by machine learning algorithms. In the context of image segmentation, one-hot encoding is used to represent each pixel's class label as a binary vector. The length of the vector is equal to the number of classes, and each element of the vector corresponds to a specific class. For example, if we have three classes - person, car, and background - then the one-hot encoding for a pixel that belongs to the car class would be [0, 1, 0]. One-hot encoding is preferred over other encoding techniques because it is easy to apply, and it enables us to leverage various metrics such as accuracy, precision, and recall in evaluating the performance of segmentation models. Since we are doing pixel-wised segmentation, each pixel will only belongs to a single class, softmax activation is used. Now, data is ready to train.

2.2 Model structure

Even though the structure of the model is introduced in previous introduction, it's necessary to give a solid view of it. The model structure is shown in table 1 below.

The hype-parameters used in the model are SGD optimizer (Stochastic Gradient Descent) with learning rate=2.5e-4, momentum=0.9, decay=5e-4. Since the customized loss function couldn't run on my device, I use catergorical crossentropy loss instead. The metrix used is accuracy, and I will evaluate the model performance by other criteria that are specifically for segmentation task.

Again, due to the RAM limitation, the batch size is set to 5, and since the model converges very soon, 5 epochs is good enough for model to learn.

Layer	**Operation**	**Output Size**	**Activation**
Input	-	224 x 224	-
C1 (block1_conv2)	VGG19 Feature Map	224 x 224	-
C2 (block2_conv2)	VGG19 Feature Map	112 x 112	-
C3 (block3_conv4)	VGG19 Feature Map	56 x 56	-
C4 (block4_conv4)	VGG19 Feature Map	28 x 28	-
C5 (block5_conv4)	VGG19 Feature Map	14 x 14	-
P5	Conv2D, 1x1, 256	14 x 14	ReLU
P5_upsampled	UpSampling2D	28 x 28	-
P4	Concatenate (C4, P5_upsampled) Conv2D, 1x1, 256	28 x 28	- ReLU
P4_upsampled	UpSampling2D	56 x 56	-
P3	Concatenate (C3, P4_upsampled) Conv2D, 1x1, 256	56 x 56	- ReLU
P3_upsampled	UpSampling2D	112 x 112	-
P2	Concatenate (C2, P3_upsampled) Conv2D, 1x1, 256	112 x 112	- ReLU
P2_upsampled	UpSampling2D	224 x 224	-
P1	Concatenate (C1, P2_upsampled) Conv2D, 1x1, 256	224 x 224	- ReLU
Output	Conv2D, 1x1, 21	224 x 224	Softmax

Table 1: SSPFPN structure

Metric	**Score**
Pixel Accuracy	0.7834
Mean Accuracy	0.0440
Mean IoU	0.0381
Frequency Weighted IoU	0.7992
Precision	0.8234
Recall	0.6408

Table 2: Evaluation Metrics and Scores

One possible reason for the low mean accuracy and mean IoU could be the limited size of the training dataset, which is unavoidable in my device.

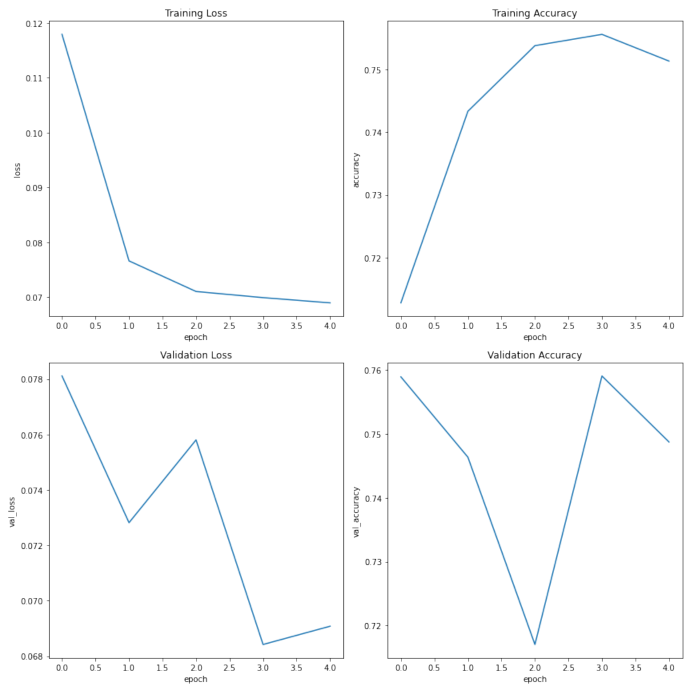

Figure 5: Learning curves for SSPFPN structure

Semantic segmentation models typically require large amounts of labeled data for effective training. If the dataset used for training is small, the model may not learn to generalize well to unseen examples, leading to poor performance on the validation set. Furthermore, if the training dataset does not cover all possible classes or instances of objects, the model may struggle to correctly classify those classes or instances during inference. In such cases, it may be necessary to either acquire more data or use transfer learning to leverage the knowledge learned by a model pre-trained on a larger and more diverse dataset. In this case, pixel accuracy of 78% and frequncey weighted IoU of 79.9% are suitable proof that model performance is just fine.

3 DeepLabV3+ & Implementation

DeepLab is a series of deep learning models for semantic image segmentation developed by Google Research. The original DeepLab model was introduced in 2014 and used atrous convolution (also known as dilated convolution) to increase the receptive field of convolutional layers without increasing their computational cost. This allowed the model to capture more global context while maintaining spatial resolution, making it well-suited for semantic segmentation.

With multiple atrous convolution appiled to the same feature map, then apply residual connection to concatenate them into one feature map is what they called atrous spatial pyramid pooling, Which helps the model to retain bigger scope information as well as texture in fine details.

Since then, several variations of DeepLab have been developed, each building on the previous model with new features and techniques. DeepLab V2 introduced the use of multi-scale inputs and improved the training procedure. DeepLab V3 added a feature pyramid network and a more powerful encoder, while DeepLab V3+ further improved the decoder and introduced an additional ASPP module for better feature extraction.

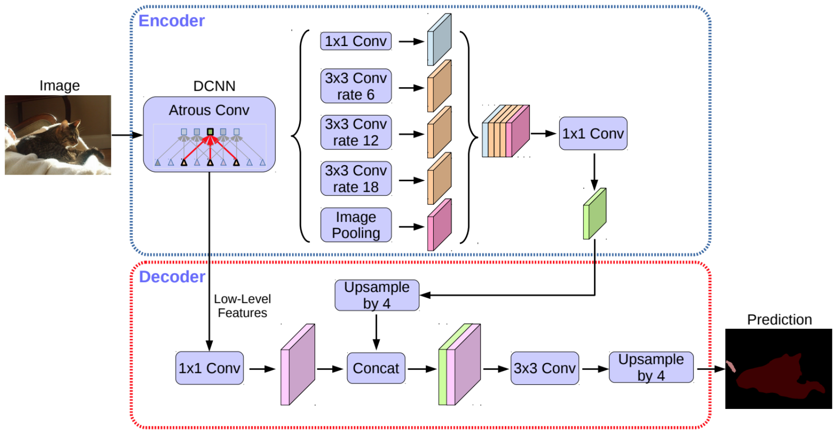

Figure 6: DeepLabV3+ structure with ASPP encoder

Inspird by the tutorial from Keras[con21], I appiled the model with the CIHP dataset used in the tutorial and got similar results.

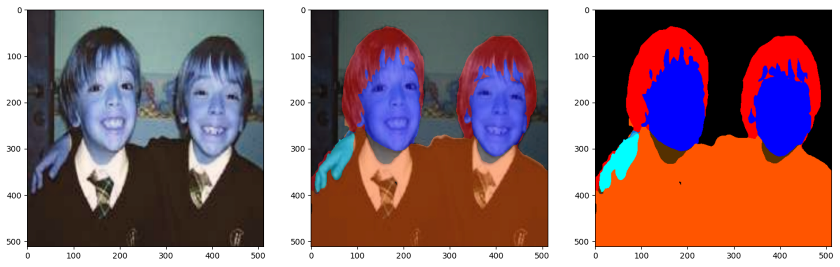

Figure 7: DeepLabV3+ prediction

Since it is worth nothing to show the evaluation using others code and others dataset. According to what I have learned so far, I will now appiled the

DeepLabV3+ on the dateset VOC12 tha't I'm working on.

When working on VOC12, same hyper-parameter setting are applied, the y-labels are converted to onehot format like before, and the model will run 5 epochs with batch size of 4. The optimizer used is adam with learning rate of 0.0001.

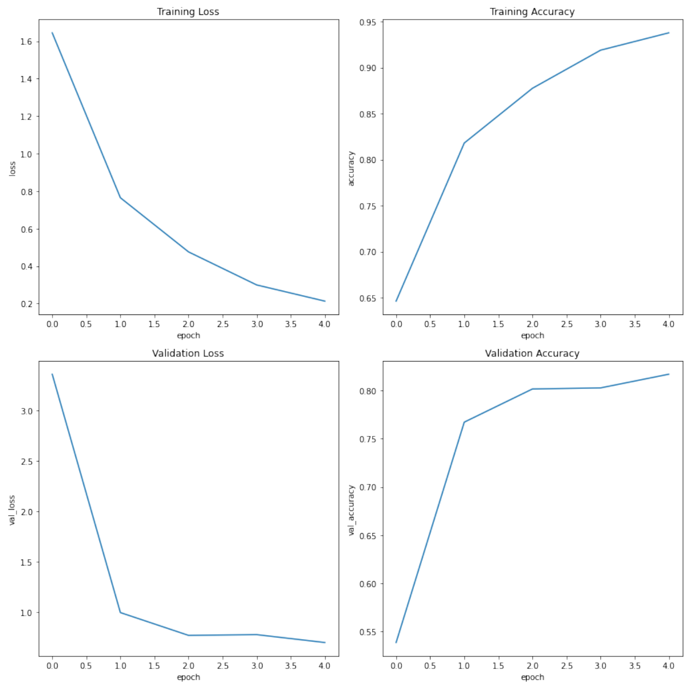

Figure 8: DeepLabV3+ learning curve on VOC12

On the other hand, since the small data of 600 images used, the mean accuracy and mean IoU is still low, but table 3 below also shows some other important evaluation scores that is better telling the performance.

Metric	Value
Validation Precision	0.8355
Validation Recall	0.7905
Pixel Accuracy	0.8883
Mean Accuracy	0.0456
Mean IoU	0.0423
Frequency Weighted IoU	0.8883

Table 3: DeepLabv3+ Model Evaluation Metrics

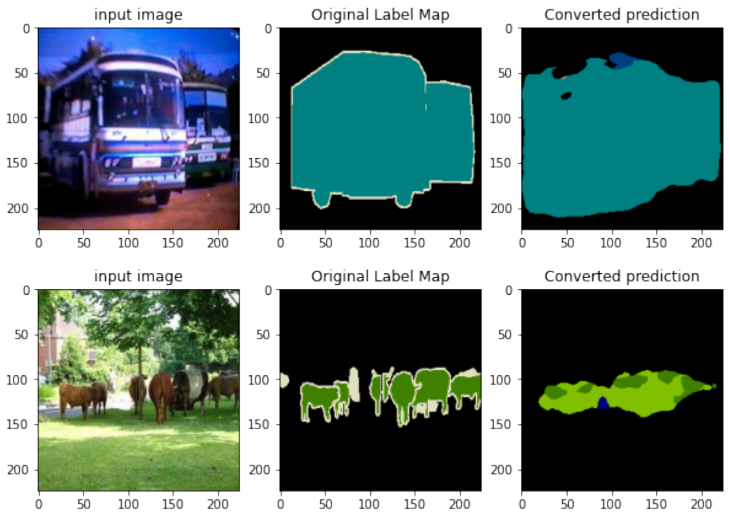

Figure 9: Two prediction of DeepLabV3+ on VOC12

4 FCN8s

Fully Convolutional Network (FCN) is an early semantic segmentation architecture that was proposed in 2015. It aims to use a single end-to-end trainable network to map input pixels to output pixels directly. The FCN8s architecture, a variant of FCN, is characterized by an encoder-decoder structure with skip connections, where the encoder is a pre-trained VGG16 model. Despite being an early architecture, FCN8s remains a popular choice for semantic segmentation tasks, but it did not perform well on VOC12 dataset. In the SSPFPN paper, the authro also achieved really high accuracy using FCN8s, so I think this is the problem on my side about tuning the parameters.

Recently, more sophisticated architectures like DeepLabV3+ and SSPFPN have shown superior performance in semantic segmentation tasks. DeepLabV3+ uses atrous convolutions and an encoder-decoder structure with multi-scale feature fusion to capture the context information and improve the spatial resolution of the segmentation output. On the other hand, the SSPFPN method employs a bottom-up path to capture low-level features and a top-down path to incorporate high-level context information, which is achieved through a multi-stage fusion of feature maps with different scales.

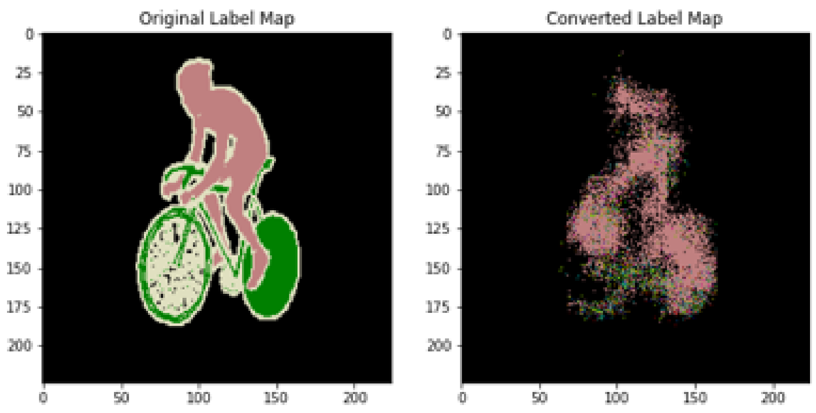

Figure 10: A glance of FCN8s of mine

5 Customized Implementation

The SSPFPN and DeeplabV3+ models have their own limitations and challenges. One of the challenges with the SSPFPN model is that it requires a lot of computational resources and time to train due to its complex architecture. Additionally, the model requires careful tuning of its many hyperparameters to achieve optimal performance. Furthermore, since the SSPFPN model relies on the VGG19 backbone, it might not extract useful position infomation from the input.

On the other hand, the DeeplabV3+ model also has some limitations. One of the main challenges with DeeplabV3+ is that it requires a lot of training data to achieve optimal performance, and may struggle with data that is imbalanced or contains a lot of noise. Another limitation is that it is computationally expensive, requiring high-end GPUs to train and run in a reasonable amount of time. Additionally, the model may struggle to capture object details and may produce inaccurate segmentations around object boundaries.

5.1 Design my own structure

After the experience of previous models, I now come up with my own method to solve VOC12 dataset.First, the encoder and decoder is necessary. ResNet is a powerful backbone network that can be used for various computer vision tasks, including segmentation (where VGG19 typically used in classifications). However, ResNet alone may not be the best option for segmentation, as it was not specifically designed for that task and may not have the necessary architectural features to effectively capture spatial information and object boundaries. After realizing the limitation of hardware, I will only take 1,3,5 stages feature map from ResNet encoder, which means there are 4 time different in size. Thus, the decoder part will also use less conventional filter. It's important to note that less complex model could limit the model's ability to learn from data, but it's a compromise. Like any other segmentation method, my decoder part of the model is composed with skip and lateral connections and convolutions filters. layers.Conv2DTranspose, which is typically used in DeconvNet is also used to upsmaple the feature maps.

Since I've mentioned the hardware limitations. I will not take 5 feature maps from backbone ResNet, instead, last feature map from 2,4,6 blocks are taken. The table 4 below show the complete structure in details.

Layer (type)	**Output Shape**	**Activation**
Input	(224, 224, 3)	-
Encoder		
conv1_relu (ResNet50)	(112, 112, 64)	ReLU
conv3_block4_out (ResNet50)	(28, 28, 512)	-
conv5_block3_out (ResNet50)	(7, 7, 2048)	-
Decoder		
Conv2DTranspose BatchNormalization ReLU	(14, 14, 256) (14, 14, 256) (14, 14, 256)	- - ReLU
UpSampling2D Concatenate Conv2D	(28, 28, 256) (28, 28, 768) (28, 28, 256)	- - ReLU
Conv2DTranspose BatchNormalization ReLU	(56, 56, 64) (56, 56, 64) (56, 56, 64)	- - ReLU
UpSampling2D Concatenate UpSampling2D Conv2D	(112, 112, 64) (112, 112, 128) (224, 224, 128) (224, 224, 64)	- - - ReLU
Final Convolutional Layer		
Conv2D	(224, 224, 21)	Softmax

Table 4: Updated ResNet-Unet Simple Model Structure

In this model, adam optimizier is ued with learning rate of 0.0001, and loss is still catergorical crossentropy since onehot format still applies. The performance

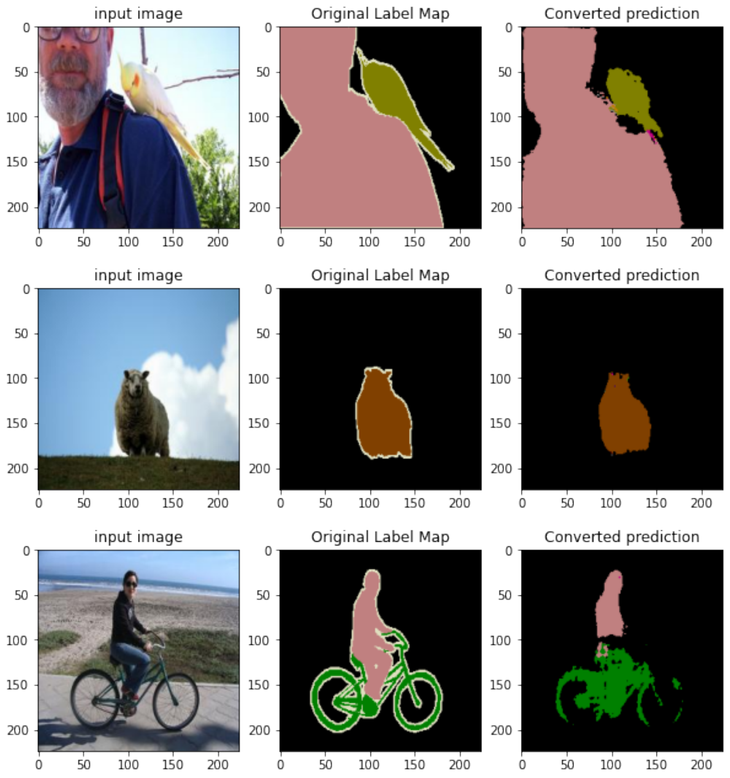

Figure 11: Prediction of Res-Unet structure on VOC12

is much better than using DeepLabV3+ on VOC12 but worse than DeepLabV3+ performing on the CIHP dataset because the author on that script did a lot of tuning to fit the specific CIHP dataset. In my model, it doesn't perform well on small objects and thin textures and this would make sense since the first block of feature map is not taken by ResNet50. On the other hand, bigger connected objects with straigh angles are more easily detected in the model.

Metric	Score
Pixel Accuracy	0.7136
Frequency Weighted IoU	0.7213
Precision	0.9394
Recall	0.8762

Table 5: Res-Unet Model Scores

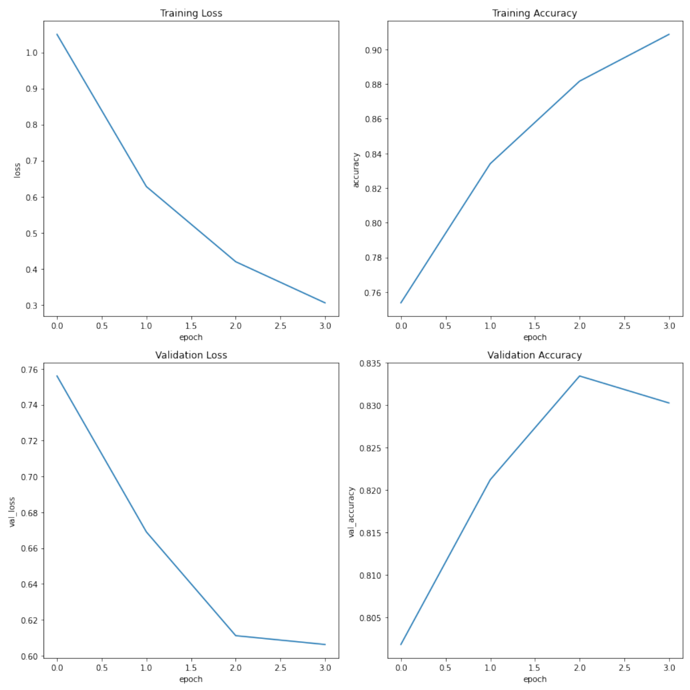

Figure 12: Learning curve for Res-Unet

6 Conclusion

In conclusion, we have explored several popular semantic segmentation models, including FCN8s, DeepLabV3+, and the SSPFPN method. We have also learned about the importance of using one-hot encoding for labels, the challenges of working with small datasets, and the importance of fine-tuning pre-trained models.

The best performance I got would be the Res-Unet structure that created by myself, the pixel accuray and Fw IoU are about 70%, which would be satifactory to me as a beginner of semantic segmentation, there are definitely more tunings I could have made to the structure and hyper-parameters to make it even better, but due to my tight schedule, I will leave it for the future. The last regret is that I could not take the whole dataset to training and evaluating, which leads to extremely low mean pixel accuracy and mean IoU. My RAM was too small for this type of tasks.

In addition, it is such a regret that I wasn't be able to implement the SSPFPN model with the loss term proposed in the article. I reach to the authors for some help and none of them replies. In conclusion, there are several important factors to consider when performing semantic segmentation. Firstly, having high-quality and diverse training data is crucial for achieving accurate results. Especially the distribution of classes should be considered carefully, otherwise mean pixel accuracy and mean intersection over union would drop tremendously. Additionally, choosing an appropriate model architecture and optimizing hyper-parameters can greatly impact the performance of the model. A wrong struction could impact the model indeed and one model might works properly with small amount of data, but once large data is taken more regularization should be used. Preprocessing techniques such as data augmentation and normalization can also improve the model's accuracy. Finally, selecting an appropriate loss function and evaluation metrics can help to accurately assess the model's performance. Overall, a combination of these factors must be considered and optimized in order to achieve the best possible results in semantic segmentation. A combination of structure would help too, especially those well knwon deep learning model such as ResNet, MobileNet, Xception..etc

References

[con21] Keras contributors. Deeplabv3+ image segmentation model. https://keras.io/examples/vision/deeplabv3$_plus/, 2021. Accessed : May4, 2023$.

[EVGW$^+$] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[GLL$^+$18] Ke Gong, Xiaodan Liang, Yicheng Li, Yimin Chen, Ming Yang, and Liang Lin. Instance-level human parsing via part grouping network, 2018.

[YFLF19] Yuan Yuan, Jie Fang, Xiaoqiang Lu, and Yachuang Feng. Spatial structure preserving feature pyramid network for semantic image segmentation. *ACM Trans. Multimedia Comput. Commun. Appl.*, 15(3), aug 2019.