CS 763/CS 764
COMPUTER VISION

PROJECT REPORT
ON

Few-Shot Image Segmentation with Prototypical Network

Submitted by

Group 05

193079001, 193079005, 193079015

Abstract

Image semantic segmentation refers to the process of labelling each pixel of the image into a class. Most of the recent methods use deep convolutional neural networks for the task but these methods generally require large set of strongly labelled data for training. Besides their hunger for training data, these methods also result in poor generalizability for classes not seen during training. Very recently few-shot segmentation approach has been explored for solving semantic segmentation problem which uses comparatively very few annotated examples for training. The paper [1] presents a PANet network designed to tackle few shot learning problem from non parametric metric learning perspective.

Contents

List of Figures

List of Tables

1 PANet

PANet is a few shot learning based method which aims at learning image segmentation using only a few annotated examples for training. It follows a prototypical network where each class is represented by a prototype in an embedded space. These prototypes are learned using a set of images called as Support Image. Using these prototypes, segmentation is performed on another set called as Query Images where each of its pixel is first taken into embedding space and compared to these prototypes, and assigned the class of the prototype it is closest to. Prototypical network generally performs only support-to-query few shot segmentation i.e. while training segmentation is performed on Query Images using prototypes computed from Support Images and segmentation loss is computed using predicted segmentation and ground truth of Query Images. But in addition to this PANet also performs query-to-support few shot segmentation, i.e prototypes are computed using the embedding of query image and used to segment the support image and compute in addition to previous loss, a loss between predicted segmentation of support image and its respective ground truth. Support-to-query forces the model to learn to generate good prototypes and query-to-support forces the model to be consistent with respect to Query and Support embedding prototypes, or in other words encourages alignment between support and query protypes and hence named prototype alignment regularization (PAR) loss.

1.1 Method Overview

1. The dataset consists of two parts, D_{train} and D_{test}. D_{train} consists of one set of classes C_{seen} and D_{test} consists of other set of classes C_{unseen}. The datasets consists of episodes, which forms the basis of episodal training. Each episode (similar concept to batches) consists of two sets of images, support set images S and query set images Q. Example, $D_{train} = (S_i, Q_i)_{i=1}^{N_{train}}$, where N_{train} represents no of episodes used for training. The support images of each episode corresponds to a C way K shot learning task, meaning there are C different classes in one episode, and K different examples of each class in one episode. The support images are pairs of (image,mask), denoted by $S_i = (I_{ck}, M_{ck})$, where k goes from 1 to K, and c goes 1 to C. The query images, Q_i contains N_{query} (image,mask) pairs from same set of classes c used in support images.

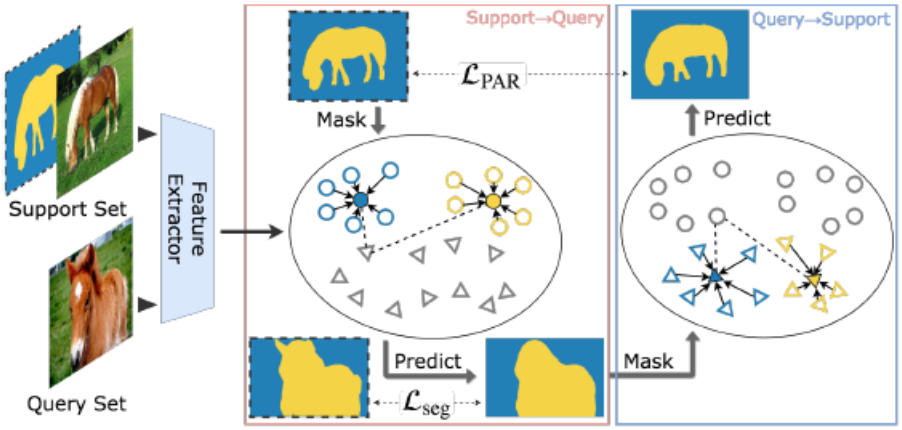

Figure 1: Overview of PANet. PANet first maps the support and query images into embedding features (circles and triangles respectively) and learns prototypes for each class (blue and yellow solid circles). Segmentation over the query is then performed by matching its features to a nearest prototype within the embedding space (dashed lines). PANet further introduces a prototype alignment regularization during training to align the prototypes from support and query images within the embedding space by performing few-shot segmentation reversely from query to support (right panel). Segmentation masks with dashed border denote ground truth annotations.[1]

2. The support images of each episodes is passed through trainable Feature Extractor, which outputs a feature vector for each pixel in an embedding space. All the feature vector corresponding to a class are grouped together in a cluster as shown in Figure 1.

3. Next step is to find a prototype representative of all the classes and background present in the support images. A detailed explanation on how to find this prototype can be found in Section 1.2.

4. Then the query image from the same episode is passed through the same feature extractor into the embedding space. Distance between feature vector of each pixel and the prototypes of background and classes are computed, and each pixel is thus classified to the class corresponding to minimum distance prototype. A non parametric approach is used in these methods, wherein a pairwise distance function is calculated between a reference and test object, the reference here being a prototype and test

object being feature vector.

5. The predicted mask for the query image now is used to compute segmentation loss L_{seg} using query ground truth mask.

6. Then the same process is followed in a reverse way, from-query-to-support segmentation is performed to compute prototypes from query image and predict segmentation on support image and L_{PAR} loss is computed using predicted segmentation and ground truth of support image. L_{seg} and L_{PAR} are then added and backpropagated to train the Feature extractor. The complete loss function is discussed in detail in section 1.4.

1.2 Prototype Learning

The model learns representative and well-separated prototype representation for each semantic class, including the background, based on the prototypical network [2]. Instead of averaging over the whole input image [2], PANet leverages the mask annotations over the support images to learn prototypes for foreground and background separately. In this work, the late fusion strategy is used since it keeps the input consistency for the shared feature extractor. Late fusion directly masks over the feature maps to produce foreground/background features separately [5].
Given a support set $S_i = \{(I_{c,k}, M_{c,k})\}$, let $F_{c,k}$ be the feature map output by the network for the image $I_{c,k}$. Here c indexes the class and $k = 1, ..., K$ indexes the support image. The prototype of class c is computed via masked average pooling [5]

$$p_c = \frac{1}{K} \sum_k \frac{\sum_{x,y} F_{c,k}^{(x,y)} \mathbb{1}[M_{c,k}^{(x,y)} = c]}{\sum_{x,y} \mathbb{1}[M_{c,k}^{(x,y)} = c]}, \tag{1}$$

The the prototype of background is computed by

$$p_{\text{bg}} = \frac{1}{CK} \sum_{c,k} \frac{\sum_{x,y} F_{c,k}^{(x,y)} \mathbb{1}[M_{c,k}^{(x,y)} \notin \mathcal{C}_i]}{\sum_{x,y} \mathbb{1}[M_{c,k}^{(x,y)} \notin \mathcal{C}_i]} \tag{2}$$

The above prototypes are optimized end-to-end through non-parametric metric learning as explained in the section 1.4.1

1.3 Early fusion and Late Fusion

For getting the masked features, two methods can be used, early fusion or late fusion. In early fusion method, the input image is first multiplied by its mask, and then passed through a feature extractor. In another method, late fusion [5], we first pass the input image through feature extractor, then we resize the feature map size to image size, and multiply feature map by the mask to get the masked features. In this method, the later, late fusion, is adopted. The reason for the same being to keep consistent feature extraction for all inputs, and the feature learning space also remains consistent for all inputs.

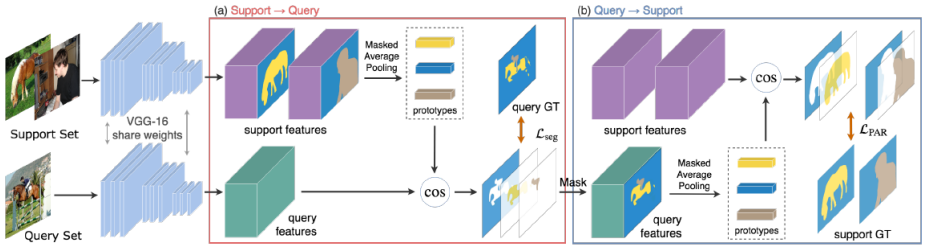

Figure 2: Illustration of the pipeline of PANet method in a 2-way 1-shot example. In block (a), PANet performs a support-to-query few-shot segmentation. The support and query images are embedded into deep features. Then the prototypes are obtained by masked average pooling. The query image is segmented via computing the cosine distance (cos in the figure) between each prototype and query features at each spatial location. Loss L_seg is computed between the segmentation result and the ground truth mask. In block (b), the proposed PAR aligns the prototypes of support and query by performing a query-to-support few-shot segmentation and calculating loss L_PAR. GT denotes the ground truth segmentation masks.[1]

1.4 Loss Function

Two loss functions are used.

1.4.1 Query Loss - Segmentation Loss

A non-parametric metric learning method is used to learn the optimal prototypes and perform segmentation accordingly. Since segmentation can be seen as classification at each spatial location, the distance between the query feature vector at each spatial location is calculated with each computed prototype. Then a softmax layer is applied over the distances to produce a probability map

$\tilde{M}_q$ over semantic classes (including background). Given a distance function d (cosine distance), let $P = \{p_c | c \in C_i\} \cup \{p_{bg}\}$, where p_c is the prototype of class c, and p_bg is prototype of background class, and F_q denote the query feature map. For each $p_j \in P$ we have

$$\tilde{M}_{q;j}^{(x,y)} = \frac{\exp(-\alpha d(F_q^{(x,y)}, p_j))}{\sum_{p_j \in \mathcal{P}} \exp(-\alpha d(F_q^{(x,y)}, p_j))}. \tag{3}$$

The predicted segmentation mask is then given by

$$\hat{M}_q^{(x,y)} = \arg\max_j \tilde{M}_{q;j}^{(x,y)}. \tag{4}$$

After computing the probability map $\tilde{M}_q$ for the query image via metric learning, the segmentation loss L_{seg} is calculated as follows:

$$\mathcal{L}_{\text{seg}} = -\frac{1}{N} \sum_{x,y} \sum_{p_j \in \mathcal{P}} \mathbb{1}[M_q^{(x,y)} = j] \log \tilde{M}_{q;j}^{(x,y)}, \tag{5}$$

This loss function helps in discriminating between different prototypes of different class and learning the embedding space representations in more distinct way to differentiate between different classes.

1.4.2 Align Loss - Prototype Alignment Regularization Loss

Prototype alignment regularization (PAR) exploits support information better to guide the few-shot learning procedure and helps enhance generalizability of the resulted model from a few examples. PAR encourages the resulted segmentation model to perform few-shot learning in a reverse direction, i.e., taking the query and the predicted mask as the new support to learn to segment the support images. This imposes a mutual alignment between the prototypes of support and query images of the same class, and learns richer knowledge from the support.

After obtaining a segmentation prediction for the query image, a masked average pooling is performed accordingly on the query features and obtain another set of prototypes $\tilde{P} = \{\tilde{p}_c | c \in C_i\} \cup \{\tilde{p}_{bg}\}$ following Eqns. 1 and 2. Next, the non-parametric method introduced in the Section 1.4.1 is used to predict the

segmentation masks for the support images.
Within PAR, the segmentation probability of the support image $I_{c,k}$ is given by

$$\tilde{M}_{c,k;j}^{(x,y)} = \frac{\exp(-\alpha d(F_{c,k}^{(x,y)}, \bar{p}_j))}{\sum_{\bar{p}_j \in \{\bar{p}_c, \bar{p}_{\text{bg}}\}} \exp(-\alpha d(F_{c,k}^{(x,y)}, \bar{p}_j))} \tag{6}$$

and the loss L_{PAR} is computed by

$$\mathcal{L}_{\text{PAR}} = -\frac{1}{CKN} \sum_{c,k,x,y} \sum_{p_j \in \mathcal{P}} \mathbb{1}[M_q^{(x,y)} = j] \log \tilde{M}_{q;j}^{(x,y)} \tag{7}$$

1.5 Evaluation Metrics

Evaluation metric used is meanIOU where IOU stands for Intersection-Over-Union. The IOU is ratio of intersection between predicted segmentation and ground truth to union of predicted segmentation and ground truth. Intersection is the no the pixels having value 1 in both predicted and ground truth of segmented mask. Union is {total no of pixels with value 1 in both predicted and ground truth} - {Intersection}

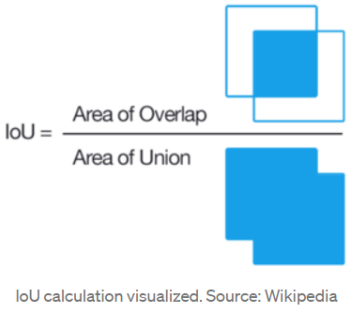

Figure 3: IOU

IOU is computed for each class present in the image and also for the background and Mean IOU is the average of all them.

2 Extensions to Original Paper

2.1 Feature Extractor with Resnet-18 backbone:

ResNet, short for Residual Network [6] is a specific type of neural network that won 1st place in the ILSVRC 2015 classification competition with a top-5 error rate of 3.57% (An ensemble model) and also won the 1st place in ILSVRC and COCO 2015 competition in ImageNet Detection, ImageNet localization, Coco

detection and Coco segmentation.
The problem of training very deep networks has been alleviated with the introduction of ResNet or residual networks and these Resnets are made up from Residual Blocks with skip connections.
The skip connections in ResNet resolve the problem of vanishing gradient in deep neural networks by allowing an alternate shortcut path for the gradient to flow through. These also allow the model to learn the identity functions which ensures that the higher layer will perform at least as good as the lower layer, and not worse.
VGG-16 has a 74.4% Top-1 accuracy on Imagenet while Resnet-18 has 71.71% and Resnet-50 has 83.2% Top-1 accuracy [7]. We wanted to use Resnet-50 as the backbone, but faced resource constraints and hence decided to use similar performance model - Resnet18, hoping for similar or better results. The network is trained end-to-end to optimize the weights of Resnet-18 for learning a consistent embedding space.

2.2 Prototypical Loss L_{PL}:

The PANet paper tries to align the prototypes for images from two different domain, i.e. source (support) domain and target (query) domain. This tries to make the method more generalized, as it intuitively means only the essential features of images are captured, as non-essential features like lightning condition, camera parameters, such will never let the prototypes of the two domains align to each other.

The original implementation for this purpose was done using L_{par} loss in PANet paper, where they tried to indirectly make the prototype learning consistent such that prototype computed from the predicted query image masks should be able to properly predict the support image mask as well, as prototypes from support image will be able to do so.

We tried to approach this problem directly, by making the model learn to reduce the RKHS (Reproducing Kernel Hilbert space) distance between the prototypes, i.e. the distance between the class representatives in the feature embedding space. Our main intuition behind this is simply to align the two domain's together, by making the prototypes representing the domain classes, come closer. We used L2 norm as the distance function to compute the distance between the prototypes, as it has better stability in learning as it is magnitude dependant. This loss function is termed as prototype_loss by us, L_{PL} and is

given as follows:

$$L_{PL} = \sum_{n=0}^{C} (p_c - p'_c)^2 \tag{8}$$

where p are the support image prototypes ,and p' are the query image prototypes.

We experimented with the contribution of this loss to the total loss function, i.e. $L_{total} = L_{seg} + L_{PAR} + \lambda * L_{PL}$, to avoid it affect the learning too much. We varied λ from 0.2, to 1, to 2. It gave pretty good evaluation results on value of 0.2, but the L_{PL} loss didn't reduce much as expected. On value of 1 it started giving less value of L_{PL} loss, but results started deteriorating. On value of 2 it gave very bad results meaning the learning could not happen with this contribution of alignment loss.

3 Implementation details of code

3.1 Github Link:

The github link for our implementation is: course-project-group$_0$5$_1$93079001$_1$93079005$_1$93079015

3.2 Dataloading:

Firstly the images and their mask are loaded using the class VOC. The transform for images is computed using classes RandomMirror (randomly flips image with 50 % probability), Resize (resizes any image to size of 417*417), and ToTensorNormalize (converts images into tensor and standardizes the input with precomputed statistics). Then there are subsets created of each class in the 20 classes, and these are passed through the PairedDataset class to create indexes of pairs of dataset. These indices are given to the fewshot function to finally create the required episode of (support, query) images and masks. The masks are generated by the getMask function.

3.3 Model:

The create episode of support and query images is passed through the class Fewshot. There it is passed through the feature extractor class Encoder, which contains vgg-16 network here. Then the foreground and background features are extracted by multiplying obtained features with corresponding masks in

function getFeatures. Then the processed features are averaged to obtain the prototypes corresponding to the classes of that episode in function getPrototype. The distance of query feature's is calculated to each prototype then using function calDist. Then the output segmented query image is given to function alignLoss, where query predicted mask is formed based on which support image segmentation and loss computation of L_{PAR} is done. The output query segmentation along with this L_{PAR} loss is then returned to the training loop, where loss L_{seg} is calculated using CrossEntropyLoss.

3.4 Evaluation metric:

The Metric class is used for evaluting the model. The record function computes the True Positive, False negative, and False positive values for each image and appends them to a list for each run. True positive (TP) values denote the overlapping white pixels in ground truth mask and segmentation prediction. False positive (FP) values denote the wrongly predicted positive (white) pixels segmentation class. False negative (FN) values denote the positive (white) pixels which are present in ground truth mask but not in the predicted masks. In the function get_mIoU, these values are taken and the intersection is calculated classwise as mIoU = TP / (TP + FP + FN). In the function get_mIoU_binary, the same is done expect for taking only background class, and foreground class (which combines all other classes together).

4 Experimentation

Parameter	**Default Value**
ways	1
K (shot)	1/5
batch size	1
Epochs	200
Feature Extractor	VGG-16
loss function	Segmentation loss + PAR loss
Optimizer	Adam
weight_decay	0.0005
Learning Rate	0.001
Schedular	Multi Step LR with steps at 75 and 150

Table 1: Default parameter values used in the training

Epochs	**Shot**	**Batch Size**	**Feature Extractor**	**Optimizer**	**Schedular**	L_{seg}	L_{PAR}	$L_{prototype}$	**Mean IOU**
30000	1	1	vgg-16	SGD	MultiStepLR	0.285	0.229	-	0.38
30000	5	1	vgg-16	SGD	MultiStepLR	0.24	0.25	-	0.455
1000	1	1	resnet-18	SGD	MultiStepLR	0.568	0.366	-	0.136
1000	5	1	resnet-18	SGD	MultiStepLR	0.57	0.53	-	0.205
1000	1	1	resnet-18	Adam	MultiStepLR	0.65	0.43	-	0.117
1000	5	1	resnet-18	Adam	MultiStepLR	0.58	0.569	-	0.118
1000	1	1	resnet-18	SGD	ExponentialLR	0.608	0.39	-	0.129
1000	5	1	resnet-18	SGD	ExponentialLR	0.605	0.58	-	0.178
1000	1	1	resnet-18	Adam	ExponentialLR	1.014	1.165	-	0.123
1000	5	1	resnet-18	Adam	ExponentialLR	0.784	1.23	-	0.169
10000	1	1	vgg-16	SGD	MultiStepLR	0.35	0.271	0.319	0.35
10000	5	1	vgg-16	SGD	MultiStepLR	0.27	0.275	0.298	0.48
200	1	2	vgg-16	SGD	-	0.378	0.3	-	0.37
200	5	2	vgg-16	Adam	MultiStepLR	0.38	0.37	-	0.45
200	1	2	vgg-16	Adam	MultiStepLR	0.38	0.297	-	0.46
200	5	2	vgg-16	Adam	MultiStepLR	0.37	0.357	-	0.47
200	1	1	vgg-16	SGD	MultiStepLR	0.5	0.39	1.03(with prototype*0.2)	0.21
200	5	1	vgg-16	SGD	MultiStepLR	0.51	0.48	0.75(with prototype*0.2)	0.23
200	1	1	vgg-16	SGD	MultiStepLR	0.67	0.599	0.48(with prototype*1)	0.19
200	5	1	vgg-16	SGD	MultiStepLR	0.62	0.65	0.35(with prototype*1)	0.246
200	1	1	vgg-16	SGD	MultiStepLR	0.68	0.91	0.32(with prototype*2)	0.07

Table 2: Results of various experiments conducted

5 Results

1. Vgg-16 with default loss function - 1-shot learning. Mean IOU = 0.38

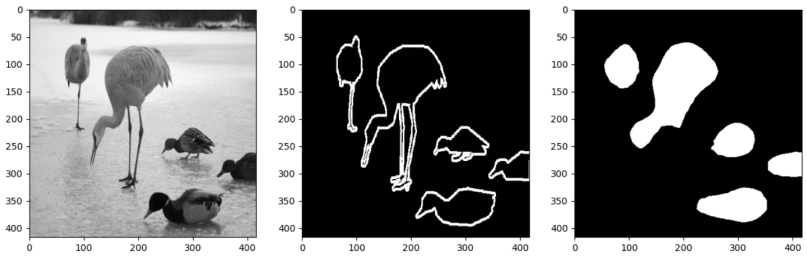

Figure 4: The input test image, the segmentation ground truth, the predicted mask - 1-shot, with default loss, SGD Optimizer, VGG-16, trained for 30,000 iterations.

2. Vgg-16 with default loss function - 5-shot learning. Mean IOU = 0.455.

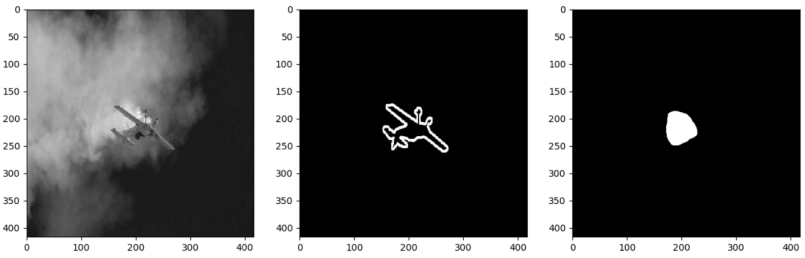

Figure 5: The input test image, the segmentation ground truth, the predicted mask - 5-shot, with default loss, SGD Optimizer, VGG-16, trained for 30,000 iterations.

3. Vgg-16 with prototypical loss function - 1-shot learning. Mean IOU = 0.35

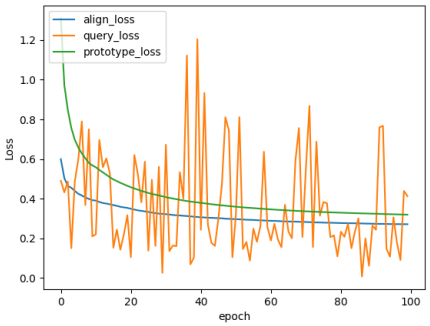

Figure 6: Loss Curve - 1-shot, prototypical loss, VGG-16, SGD Optimizer.

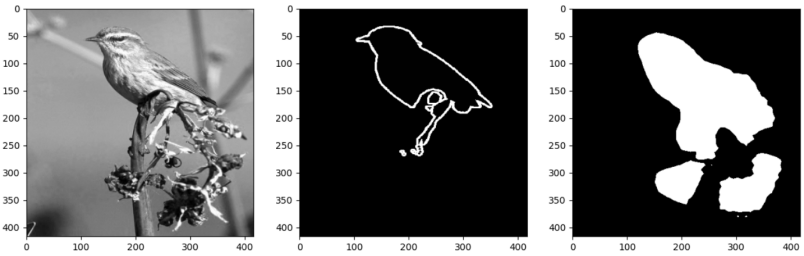

Figure 7: The input test image, the segmentation ground truth, the predicted mask - 1-shot, with prototypical loss, SGD Optimizer, VGG-16, trained for 10,000 iterations.

4. Vgg-16 with prototypical loss function - 5-shot learning. Mean IOU = 0.27

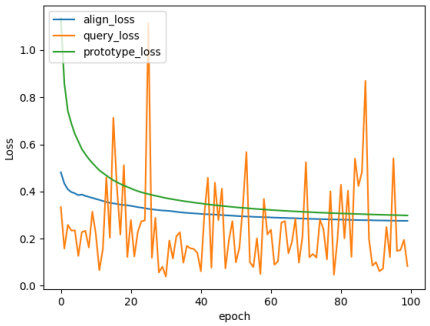

Figure 8: Loss Curve - 5-shot, prototypical loss, VGG-16, SGD Optimizer.

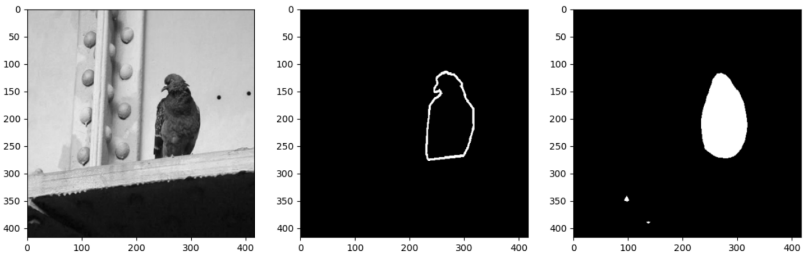

Figure 9: The input test image, the segmentation ground truth, the predicted mask - 5-shot, with prototypical loss, SGD Optimizer, VGG-16, trained for 10,000 iterations.

5. Resnet-18 with default loss function - 1-shot learning. Mean IOU = 0.11

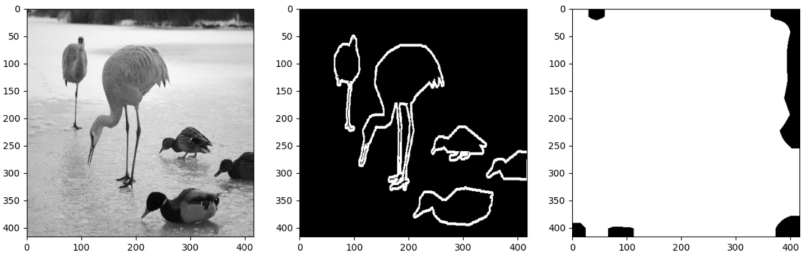

Figure 10: The input test image, the segmentation ground truth, the predicted mask - 1-shot, with prototypical loss, SGD Optimizer, Resnet-18, trained for 1,000 iterations

6. Resnet-18 with default loss function - 1-shot learning. Mean IOU = 0.17

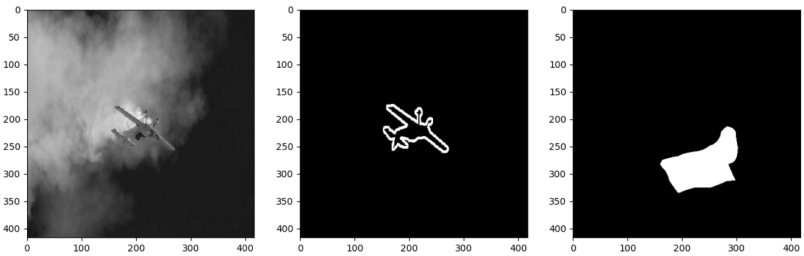

Figure 11: The input test image, the segmentation ground truth, the predicted mask - 5-shot, with prototypical loss, SGD Optimizer, Resnet-18, trained for 1,000 iterations

6 Conclusion

We successfully trained the model for PANet architecture and evaluated its performance on the PASCAL-VOC2012 data. We also tried improving the baseline paper, by adding two novelties,

1. Resnet-18 backbone as model architecture. We can see in Figure.10 and Figure.11 the degraded segmentation mask prediction performance with Resnet as a feature extractor.

2. Modified loss function to directly align the prototypes. We can see in Figure.7 and Figure.9 the segmentation mask prediction performance with Prototypical Loss.

The results obtained by our novelty were not better than the baseline paper results. Baseline code had accuracy of 38% and 45% on 1-shot and 5-shot respectively on our system trained model with 10000 iterations, where as the modified loss function gave results of 35% and 48% respectively, which shows we had 3% better performance on the original architecture for 5-shot, although the original implementation by the paper authors gave performance of 55% for 5-shot. The resnet architecture gave very bad results on even 1000 iterations so we didn't run it for further iterations due to resource limitations. Overall, we can conclude that the modified loss function does help in consistent learning, but the accuracy obtained in 10,000 iterations might not be good enough to obtain better segmentation results.

7 Resource Constraint / Difficulty Faced

1. Plan of Using of ResNet50 for feature extractor backbone did not work as we ran into cuda out of memory error. This lead us into using Resnet-18 architecture as feature extractor's backbone.

2. We had to train for lesser iterations on Google Colab as out of memory issue use to pop up for large iteration values. The authors trained the network for 30,000 iterations while on Google Colab we could go only upto 200 iterations.

3. The initial plan of comparing performance of PANet on VOC dataset with COCO dataset got scraped as COCO dataset was too large to be loaded to Google Colab (Drive's size limit is 15Gb, while COCO dataset is of 25Gb size).

8 References

[1] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou and Jiashi Feng. *PANet: Few-Shot Image Semantic Segmentation with Prototype Alignment. In ICCV, 2019.* `https://arxiv.org/abs/1908.06391`

[2] Jake Snell and Kevin Swersky and Richard S. Zemel. *Prototypical Networks for Few-shot Learning. In NIPS, 2017*
`https://arxiv.org/abs/1703.05175`

[3] Github Code Reference - `https://github.com/kaixin96/PANet`

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015.*

[5] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas Huang. *Sg-one: Similarity guidance network for one-shot semantic segmentation.In TCYB, 2020*
`https://arxiv.org/pdf/1810.09091.pdf`

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *eep Residual Learning for Image Recognition. In CVPR, 2016*
`https://arxiv.org/abs/1512.03385`

[7] `https://paperswithcode.com/sota/image-classification-on-imagenet`