Task Conflict in Meta Learning for Few-Shot Segmentation

Geetank Raipuria, Nitin Singhal
AIRAMATRIX PVT. LTD., MUMBAI, INDIA
geetank.raipuria@airamatrix.com; nitin.singhal@airamatrix.com

Abstract

Few-Shot segmentation has gained significant attention due to wide-ranging applications of dense pixel segmentation and limitation in obtaining large-scale annotated data. A Few-Shot Segmentation model is (meta) trained on base class samples to recognize a target class in a query image based on information from support image(s). Masks are generated for target class in each query-support image set, such that any other class present in the image set is merged with the background. As a result of this training setup as well as overlapping images between base training classes and novel test classes in benchmark dataset like Pascal-5^i, the test classes are labeled as background during training. We illustrate that this causes a task conflict between train and test setup by analyzing the effect of meta training on the distribution of test class features in embedded space. The model learns to recognize test classes as background, leading to poor Few-Shot test performance. We propose a modified meta training setup to prevent conflicting tasks and improve feature representation as well as Few-Shot performance on all classes. We hope that future work address this issue when evaluating few-shot models on simulated dataset like Pascal-5^i

Introduction

State-of-the-art deep learning models require an enormous amount of training data per class to learn representative features. However, in certain scenarios, it may only be possible to gather an extremely limited amount of data, for example, scares traffic signs for autonomous driving or novel pathology in medical image analysis. This has led to the development of Few-Shot learning that aims at learning a model that can generalize well on new classes given a few annotated examples. Additionally, Few-Shot learning also acts as a testbed for a better understanding of deep learning models as well as comparing their learning capabilities with humans.

Significant progress has been made towards the Few-Shot classification (Vinyals et al. 2016; Snell, Swersky, and Zemel 2017; Finn, Abbeel, and Levine 2017; Sung et al. 2018; Qi, Brown, and Lowe 2018). The Few-Shot model is *taught to learn* to recognize a class given a few class samples. Specifically, the model is *meta* trained on the Few-Shot task T_i ($i \in [1, N]$) to identify base classes and is evaluated on similar but novel test task T_{N+k} ($k \in [1, M]$) of identifying unseen classes.

As the field continues to evolve, the development of Few-Shot classification has subsequently led to the research on Few-Shot segmentation models. State-of-the-art Few-Shot segmentation models (Wang et al. 2019; Nguyen and Todorovic 2019; Liu and Qin 2020) use prototype-based metric learning. The setup splits the classes into training and test, such that test class are novel to the model and only seen during the testing phase ($C_{train} \cap C_{test} = \emptyset$). The training and test data are divided into episodes consisting of support set $((I_S, Y_S)_K)$ and query set $((I_Q, Y_Q))$, where I is the image, Y is the image dense label, K represents the number of support images (K-shot). The Few-Shot model obtains masked average embedding from support image(s) for a target class as a prototype feature vector and uses a distance metric for determining the similarity of query image pixels to class prototypes, including target class and background class. The model is (meta) trained on base classes that have an abundance of samples, to learn a mapping from input space to embedded space that cluster class feature vectors (Snell, Swersky, and Zemel 2017). Consequently, the cosine distance metric, which is used as the distance metric by state-of-the-art models is shown to reduce intra-class variations in the embedded feature space (Gidaris and Komodakis 2018; Qi, Brown, and Lowe 2018; Siam, Oreshkin, and Jagersand 2019). That is improving the clustering of the same class samples and increasing inter-class distances. This would lead to a reduced ratio of within-cluster distances to between-cluster distances.

However, in our experiments, we find an increase in this ratio for test class samples using two different feature extractors. Further analysis shows reduced performance for certain test classes after meta training, which collaborates with the increase in above-mentioned ratio. We attribute this to the design of the training setup for Few-Shot segmentation, as described in the above paragraph. The setup was adopted from a Few-Shot classification by (Shaban et al. 2017). However, unlike classification, segmentation dataset like PASCAL VOC have labels of multiple foreground classes in a single image. To tackle this, a binary mask is constructed for support and query images for each episode, such that all classes except the target class are la-

Figure 1: Sample images and corresponding foreground class mask from PASCAL-5^i Dataset for Few-Shot Segmentation Challenge (Shaban et al. 2017). All test classes are masked as background during meta training. This results in a class conflict between train and test setup.

beled as background, as seen in figure 1. As a result, all pixels of test classes are marked as background in all training episodes. The model is trained and evaluated on this episodic setup. This means, although training images do not contain labels for novel test classes, the model is trained to identify test class pixels as background. We find that this conflict between training and test tasks is detrimental in identifying the test classes. It causes undesired clustering of test class feature vectors around background class samples and results in poor performance during evaluation. Furthermore, we observe that the test classes that have the most common images with train classes, and as a result more often labeled as background during training, perform worst on evaluation.

We propose to alleviate this task conflict when training model on a simulated few-shot dataset like Pascal-5^i by not labeling any non-target foreground classes as background in training episodes. We ignore pixels representing non-target class, if present, from the loss during training. Experiments show that this results in learning better feature representations in embedded space. The improved feature representation in turn leads to improved performance on all test classes.

(Shaban et al. 2017) simulated the few-shot task by labeling all 'novel' test classes as background in meta training dataset. However, in a real world application it is unlikely that the test classes are present in a significant number in the training data, labeled as background. Rather, a more realistic scenario is that only a few samples of test classes are available. Then a task conflict between train and test classes would not arise in a real-world application of few-shot learning. It is thus essential to eliminate the task conflict to have a more realistic simulated dataset.

We enlist the contributions of this paper as follows:

- We analyze the effect of meta training on the distribution of test class features in embedded space and find that not all classes benefit from meta training.
- We illustrate that there exists a conflict between training and test tasks using existing meta training setup. This results in poor performance on test classes.
- Consequently, we propose a simple solution that alleviates the task conflict. Furthermore, we show that it leads to improved performance on all test classes.

The paper is organized as follows, section provides a brief background on meta learning and Few-Shot segmentation, section provides model architecture and implementation details, in section we show results of our analysis on the effects of the meta learning process on class feature distribution in embedded space as well as the task conflict in training and testing setups, section provides experimental results using our proposed setup to alleviate task conflict resulting in improved Few-Shot performance.

Background

Meta Learning Meta learning involves generalizing a model on an abundance of training 'task' such that it can quickly learn to solve the novel test tasks based on past knowledge. This approach of learning-to-learn is useful in Few-Shot learning, since recognizing each Few-Shot class can be considered as a task. Recent works on Few-Shot learning(Chen et al. 2019; Tian et al. 2020) frame the training data in episodes, where each episode represents a Few-Shot task. The task consists of support and query images representing N (way) classes, and the model is expected to recognize the classes from query images based on the sup-

port images. Thus during meta training, the model learns to perform a Few-Shot task on the base training classes. Consequently, by generalizing these base classes, the model can perform new tasks of recognizing novel classes.

Few-Shot meta learning algorithms can be organized into two broad categories, optimization-based learners (Finn, Abbeel, and Levine 2017; Nichol and Schulman 2018; Rusu et al. 2018) and metric learners (Snell, Swersky, and Zemel 2017; Vinyals et al. 2016; Qi, Brown, and Lowe 2018). The key idea behind optimization-based learners is to obtain a set of network parameters such that they can be quickly fine-tuned on test tasks given a small number of (Few-Shot) samples. On the other hand, the latter approach of metric learners embeds input data into a latent feature space and uses a distance metric to recognize novel classes based on support samples. The distance metric is typically calculated as Euclidean distance or cosine similarity. The metric learning approach is more efficient and simpler to train, which has made it a preferred choice.

Metric Learning works by finding a mapping from input space to embedded space, such that the distance between samples from the same class is minimized in embedded space, effectively leading to the formation of distinct class clusters. Prior work on Few-Shot classification has shown evidence of such clustering as a result of metric learning, using t-SNE (Gidaris and Komodakis 2018) and LDA (Goldblum et al. 2020) plots.

Few-Shot Segmentation Inspired by Few-Shot recognition, OSLSM (Shaban et al. 2017) addressed Few-Shot segmentation using a conditioning branch which produced a set of parameters based on the support images. These parameters are used by a segmentation branch to generate target object masks. Later works (Rakelly et al. 2018; Dong and Xing 2018; Zhang et al. 2018) build upon this to achieve better performance by applying the two-branch architecture. Importantly, SG-One (Zhang et al. 2018) introduced masked average pooling for extracting target class representative features.

State-of-the-art model, PANet(Wang et al. 2019) employs masked average pooling to build an architecture very similar to prototypical networks for Few-Shot classification(Snell, Swersky, and Zemel 2017) and uses cosine distance to match feature vectors to class prototypes. Additionally, it also proposes a Prototype Alignment Regularize (PAR), which generates prototypes from query images to segment support images. PRNet (Liu and Qin 2020) built upon PANet, by fusing information from support and query images using a prototype refinement module. Parallelly, FWB(Nguyen and Todorovic 2019) employs feature reweighing and boosting during inference for achieving improved Few-Shot performance using class prototypes. AMP (Siam, Oreshkin, and Jagersand 2019) also uses masked average pooling to obtain Few-Shot class prototypes. It does not use meta-training setup, rather it trains the model with a standard segmentation. However, it uses normalized features and final layer weights, which is effectively cosine similarity measure. Thus training setup used is similar to prototypical networks, where the last layer's weights act as prototypes. Importantly, (Siam, Oreshkin, and Jagersand 2019) shows the clustering of class feature vector as a result of training using t-SNE plots.

Model Description

The Few-Shot segmentation model used for experiments consists of a feature extractor $f(\theta)$, which maps input space to an embedded space $F \in R^{d,w,h}$, where d is the feature dimension, and w, h denote the width and height. Target class prototypes P_c are generated from support image by applying a mask on the output of the feature extractor and taking a mean of the masked feature map (masked average pooling), $P_c = \frac{1}{|\tilde{m}|} \sum_{i=1}^{wh} F_i.\tilde{m}_i$, where $\tilde{m}$ is the down-sampled class mask. Prediction on query images is obtained as distance from the class prototypes, followed by a softmax layer. We use cosine similarity as the distance metric, defined as $s_{i,j} = \frac{v_i^T v_j}{||v_i||\,||v_j||}$, $s_{i,j}\ \epsilon[-1, 1]$, where $v_i\ and\ v_j$ are feature vectors.

The loss is composed of two equal components, cross-entropy loss between query image ground truth and model prediction, as well as prototype alignment regularizer(Wang et al. 2019). We multiply the cosine distance by a factor α before the softmax layer. Value of α is kept constant, $alpha = 20$.

We evaluate our experiments on two feature extractors ResNet-50(He et al. 2016) and VGG(Simonyan and Zisserman 2014). Both feature extractors are pre-trained on the ImageNet Object Recognition Challenge dataset. To adapt the model for feature extraction, we remove the fully connected layers, as well as the last Relu activation following (Gidaris and Komodakis 2018; Wang et al. 2019).

Implementation Details

We train the models using SGD with a momentum of 0.9 and weight decay as 5e-4. The learning rate is initialized as 5e-4 with a batch size of 8, we train the model for 15k iterations, lowering the learning rate by a factor of 0.1 after 10k iterations. Input images are resized to (417, 417), augmentation is followed from (Wang et al. 2019). The code was implemented on Pytorch and we use a NVIDIA V100 GPU for running the experiments.

Dataset and Evaluation Metric

As defined by (Shaban et al. 2017), we use PASCAL-5^i dataset for training and evaluation. The dataset consists of 4 splits, each containing 15 training and 5 test classes. In baseline setup (S1), binary masks are generated for support and query images with target class as foreground, any other class if present is labeled as background. In the case of our proposed setup (S2), the generated labels assign a foreground mask to the target class and background mask to only the background class in the image, excluding any other class if present in the image. Also, for setup S2, the evaluation metric excludes any pixel representing the non-target class to maintain task consistency.

For all models we report test performance on two metrics - mean-IOU and Binary-IOU, averaged over 5 runs each hav-

ing 1000 support-query sets. Mean-IOU measures the average IOU overall foreground classes, whereas Binary-IOU takes an average of foreground(class-agnostic) and background IOU. To maintain readability we only evaluate on 1-way 1-shot setup.

Analysis of Few-Shot Meta Learning

We analyze the impact of meta training on the distribution of class feature vectors. This would help us to better understand the learning process. Since the Few-Shot model uses a cosine-based metric learning approach, we expect class prototypes to have reduced intra-class variance and large inter-class distance in embedded space post meta training.

We first perform a qualitative analysis by visualizing class prototypes of the test classes before and after training. Figure 2 shows the t-SNE plots for all the splits(1-4), left and the right column represents features vectors before and post training. Contrary to our expectation that cluster cohesion would improve for all classes, we observed a reduction in cluster cohesion for certain classes. For example: classes 4 & 5 (row 1), class 9 (row 2), classes 11 & 15 (row 3) and classes 16 and 16 (row 4). Rather, feature vectors of these classes are distributed closer to background class 0 samples post meta-training.

To investigate quantitatively, we use Davies-Bouldin score(Davies and Bouldin 1979) (DB score). DB score is a cluster separation measure, that reduces if the distance between clusters increases or (intra) cluster dispersion reduces. Mathematically,

$$DB\ score = \frac{1}{k}\sum_{i=1}^{k} max_{(i\neq j)}\left\{\frac{\Delta(X_i)+\Delta(X_j)}{\Delta(X_i,X_j)}\right\} \quad (1)$$

$\Delta(X_i)$ is intra-cluster distance, $\Delta(X_i, X_j)$ is inter-cluster distance and k is the number of classes.

We calculate DB score for test classes in all four splits from PASCAL-5^i datasets, each split consists of 6 classes, 5 foregrounds, and background. The score is calculated using 1000 random support-query image pairs. Figure 3 plots DB score for each split before ($Init$) and post meta training (MT^{S1}). We would expect that post training, DB score would reduce for all splits if the class samples have better cluster cohesion. However, in the case of ResNet feature extractor, for two splits (1 and 3) DB score increases, signifying dispersion of clusters. For another split (4), the score only marginally reduces. Whereas, for VGG feature extractor DB score increased for split-1. This signifies that although some classes improve clustering post training, other classes may have lost cluster cohesion and merged with background class, causing an increase in overall DB score for the split.

From figure 2 we observed that cluster pattern is different for each foreground class. For example in split 1, classes 1, 2, 3 have clear cluster separations, similarly class 6, 8, 10 (split 2), class 12, 13, 14 (split 3) and 17, 19 (split 4). However, some classes have poor clustering post training. This would correspond to a reduced performance of these classes. We validate this by comparing class IOUs before and after

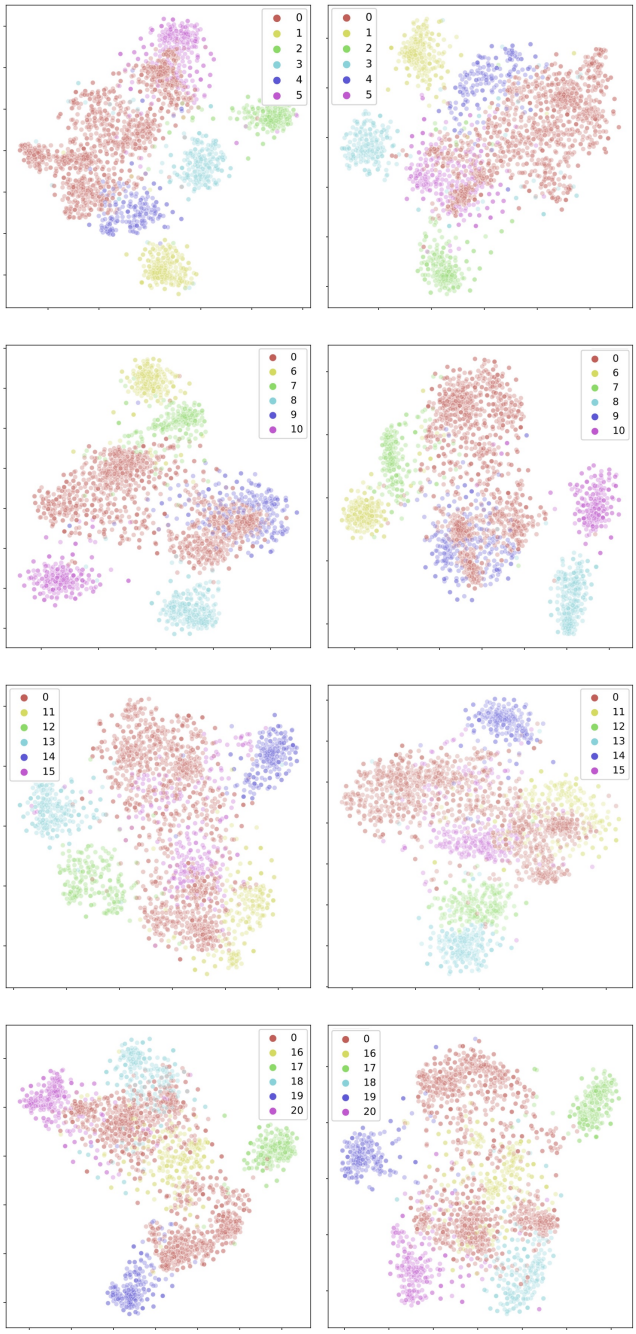

Figure 2: t-SNE plots of test class prototypes for four PASCAL-5^i splits, obtained from 1000 support-query image sets. Row 1 to 4, top: split-1 , bottom: split-3. Left: ImageNet Pre-Trained ($Init_{ResNet-50}$) model, Right: Model $MT^{S1}_{ResNet-50}$, trained on standard setup(S1). The cluster cohesion of certain classes, for example, class 5 or 9 is reduced post meta training which is detrimental to Few-Shot performance.

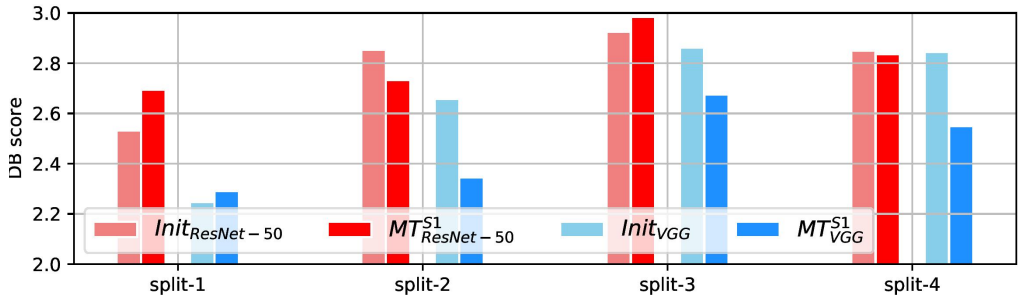

Figure 3: DB scores for PASCAL-5^i training splits, evaluated on standard setup (S1). Reduction in the DB score signifies increase inter-class distances and reduction in intra-class distances. Certain splits show an increase in DB score post meta training which signifies poor class cluster cohesion.

training. Figure 4 plots class IOUs [1]. The plot shows that not all classes benefit from meta learning equally. Importantly, classes 11, 15, 16, and 20 loose accuracy. Also, classes 4, 5, and 9 show only small improvements in IOU. We can correlate these class accuracy with the clustering behavior, for example, class 4 shows poor clustering in t-SNE plots post training. This is also observed for classes 5, 9, 11, 15, 16, and 20.

We hypothesize that the poor performance of certain test classes is due to the design of meta learning setup. This can be attributed to the fact that test classes are masked as background class in images during meta training. Since the learning causes clustering of class feature vectors, features are learned by the model that cluster test classes closer to the background class[2]. This causes a conflict of task between training and testing, during training the model learns to recognize test classes as background, but is evaluated to recognize the same classes as foreground.

To validate this claim we make a matrix representing the count of shared images between all the PASCAL5^i classes, figure 5. The figure shows the number of shared images between two classes, normalized by the total number of images per class. It can be observed that certain classes have a large number of shared images. All the classes which perform poorly post training have a larger number of shared images with corresponding training classes. For example, class 5 (bottle) shares a large number of images with class 11 (dining table); class 9 (chair) also shares a significant number of images with classes 1 (dining table), 16 (potted plant), 18 (sofa) and 20 (TV). This means that these classes are more often identified as background during training. As a result of this, the model learns features that represent these test classes as background, this happens more with classes that have a higher number of shared images with other classes.

We provide results before and after meta learning on the two metrics - MeanIOU and BinaryIOU using the standard setup S1, see table 1. These results signify that the overall performance on both the metrics increases as the meta training is beneficial for a majority of the classes, with only a minority display a loss in accuracy. Also, we benchmark the performance of our model with state-of-the-art PANet (Wang et al. 2019) to verify our model, which is also trained on standard setup corresponding to $S1$ and uses VGG feature extractor.

Improving Meta Learning

The meta trained model is expected to recognize novel test classes during evaluation. However, when we label test classes as background during training, it results in a task conflict between training and testing. Effectively, the model is trained to learn a mapping from input space to embedded space which represents test classes as background. To alleviate this, we propose to ignore labels from test classes during training, rather than labeling them as background. This ensures the model does not learn to recognize novel test classes as background. To further ensure task consistency, we use only the target class as a foreground and ignore any other classes if present except background for both trainings as well as test samples. This ensures the model does not learn to identify classes with false categories - background.

We evaluate our approach by comparing the pre-trained model ($Init$) with a trained model using the proposed setup (MT^{S2}). Additionally, to validate that the new setup is able to perform better class feature representation, we compare performance with MT^{S1} (baseline) which was trained on the original setup (S1) but now evaluated using the new proposed setup (S2). Figure 6 and 7 shows DB score and class IOU for $Init$, MT^{S1} and MT^{S2}, using both ResNet and VGG feature extractors.

DB scores for MT^{S2} are significantly lower than $Init$ for all splits using either feature extractors, unlike observed in figure 3 for MT^{S1}. This signifies that the model was able to learn better feature representation through meta-training which reduced intra-class to inter-class distance ratio. Also, MT^{S2} achieves better clustering as compared to MT^{S1} for all splits, which further validates that the proposed setup results in learning better feature representation. This also validates that removing conflict between training and test tasks improves model performance.

[1] As described above, class 1-5 results are from split 1 training run, class 5-10 results from split 2 training run and so on

[2] Although in each episode all classes except target class including training and test classes are merged with background label, it would not affect the performance of the training classes since each training class is masked as foreground numerous times over the training episodes and that model learns appropriate feature representations to reduce loss. However, this not true for test classes

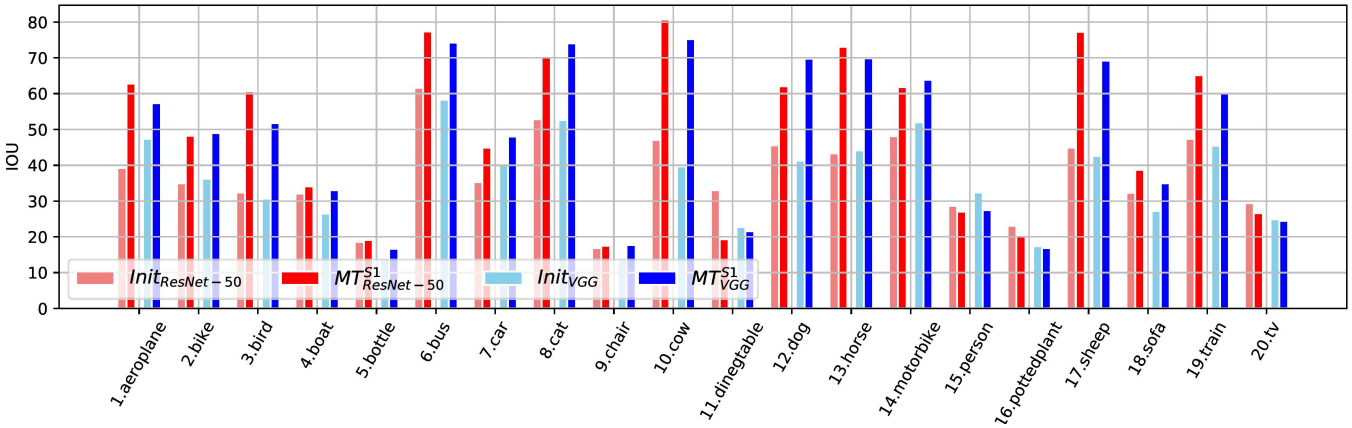

Figure 4: Class IOUs, evaluated on standard setup (S1). Certain classes do not achieve significant performance gain post meta training.

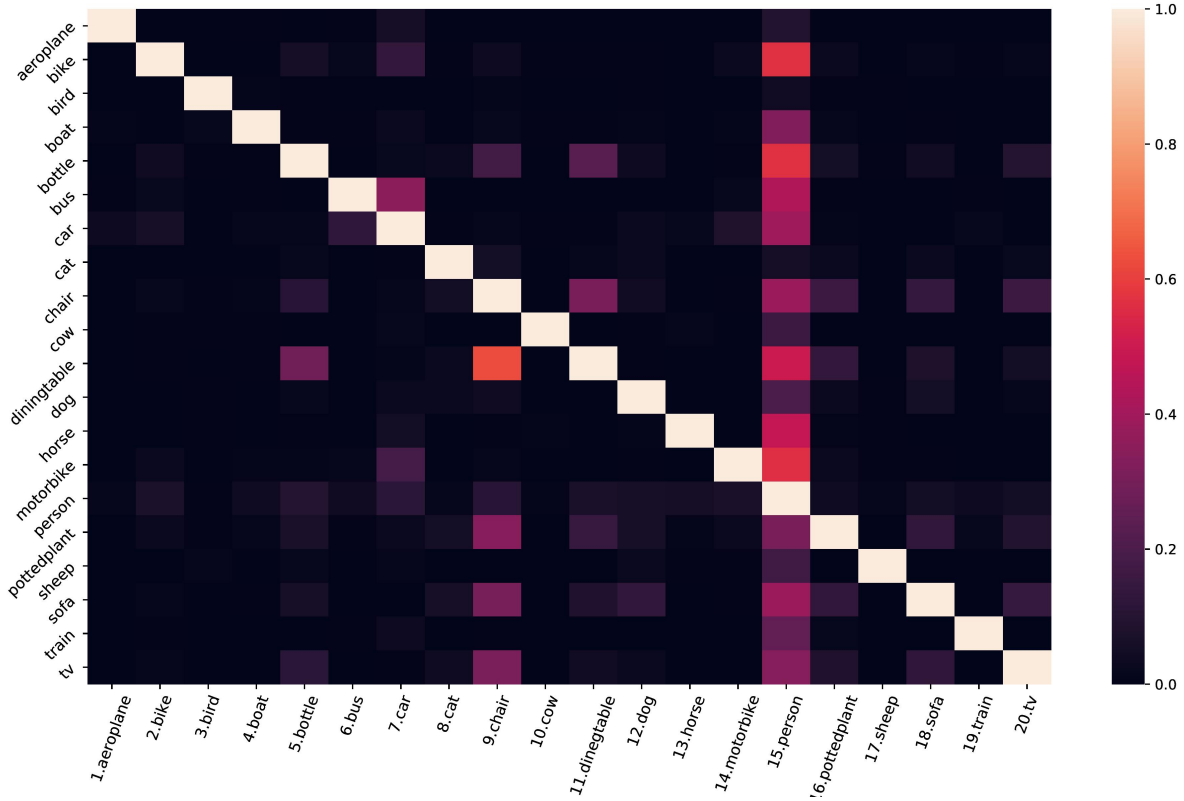

Figure 5: Matrix showing (normalized) number of common images between different PASCAL-5^i dataset classes

	Mean IOU					Binary IOU
	split-1	split-2	split-3	split-4	Mean	
$Init_{R50}$	31.4	42.7	39.7	35.3	37.3	52.8
MT^{S1}_{R50}	44.9	58.1	48.6	45.6	49.3	67.0
$Init_{VGG}$*	30.9	40.8	38.5	31.5	35.4	58.9
MT^{S1}_{VGG}*	41.5	57.8	50.4	41.1	47.7	66.2
$PANet$(Wang et al. 2019)	42.3	58.0	51.1	41.2	48.1	66.5

Table 1: MeanIOU and BinaryIOU results (1-way 1-shot), evaluated on standard setup (S1)

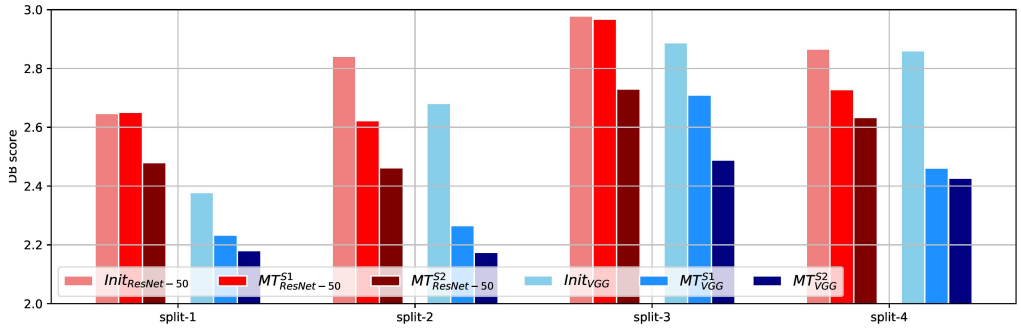

Figure 6: DB scores for PASCAL5^i training split, evaluated on the proposed setup (S2). The model trained used proposed setup MT^{S2} achieves lower DB score than $Init$ which signifies better cluster cohesion.

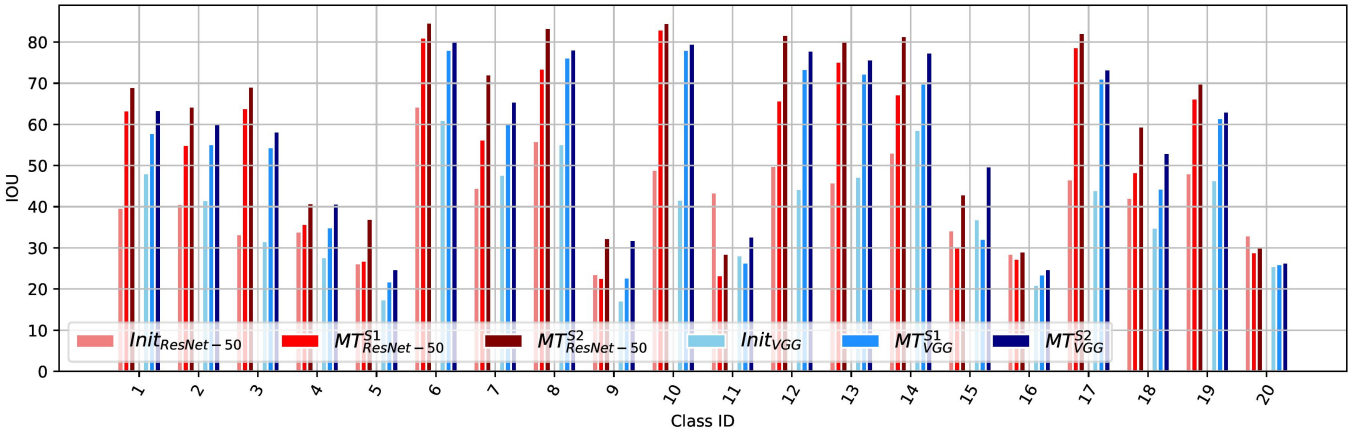

Figure 7: Class IOUs, evaluated on the proposed setup (S2). MT^{S2} achieves higher IOU than MT^{S1} which shows that MT^{S2} is able to learn better features.

	Mean IOU					Binary IOU
	split-1	split-2	split-3	split-4	Mean	
$Init_{R50}$	34.7	47.4	45.3	39.7	41.8	56.1
MT^{S1}_{R50}	48.9	63.3	52.4	50.0	53.6	70.9
MT^{S2}_{R50}	**56.0**	**71.4**	**63.0**	**54.1**	**61.1**	**76.2**
$Init_{VGG}$	33.3	44.5	43.0	34.4	38.8	60.8
MT^{S1}_{VGG}	44.8	63.1	54.8	45.3	52.0	69.7
MT^{S2}_{VGG}	**49.5**	**67.0**	**62.7**	**47.7**	**56.7**	**72.7**

Table 2: MeanIOU and BinaryIOU results (1-way 1-shot), evaluated on the new proposed setup (S2).

Analyzing fig. 7, we see that MT^{S2} improves IOU scores for all classes, this includes classes for which MT^{S1} obtained lower score than $Init$, for example, class 9 (chair), class 15 (person) and class 16 (pottedplant). Also, IOU scores are improved for other classes, since the model is trained without any conflict between training and testing tasks for any class. The higher gain in IOU is observed for classes that had a significant number of overlapping images with training classes.

Table 2 shows the performance of the three models on the MeanIOU and BinaryIOU metrics, evaluating using setup S2. MT^{S2} outperforms both $Init$ and MT^{S1} for all four splits as well as Binary IOU metric. Higher Binary IOU shows the model also able to recognize background class pixels better than MT^{S1}. In the case of MT^{S1}, the model was learned to recognize novel test classes as background, which results in the reduced background as well as foreground class performance. This further validates that preventing conflicting tasks results in better meta training. Lastly, Figure 8 shows qualitative results of MT^{S2} against baseline MT^{S1} and ground truth labels, which re-iterate the above finds as MT^{S2} predicts more accurate masks than MT^{S1}.

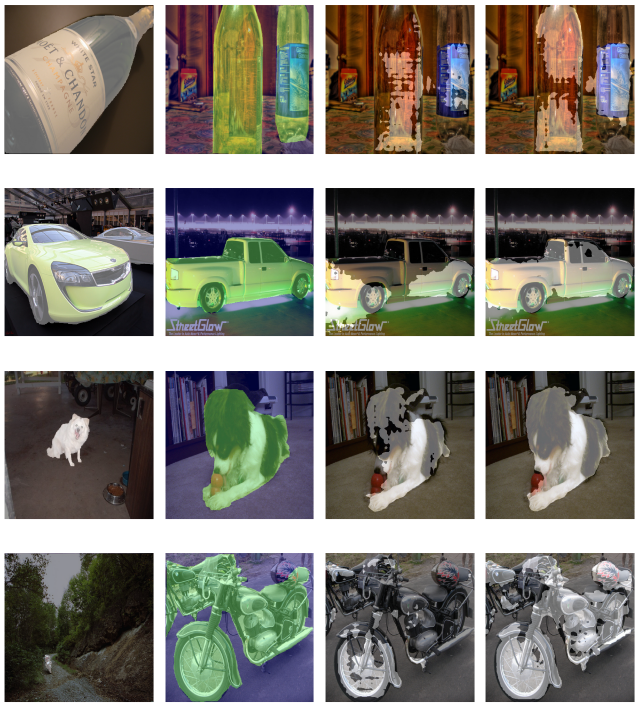

Figure 8: Sample support-query images from the test set evaluated on proposed setup (S2). Column 1: support image with label foreground mask, Column 2: Query Image with label foreground mask, Column 3: prediction from $MT^{S1}_{ResNet-50}$ model trained on standard setup (S1), Column 4: prediction from $MT^{S2}_{ResNet-50}$ model trained on proposed setup (S2).

Conclusion

We analyzed the effect of meta training on the distribution of test class feature vector in embedded space when performing benchmark on PASCAL-5^i dataset, and found undesired clustering of certain test classes with background class. Also, these classes do not achieve as much performance gain from meta training as other classes. We show that this is a result of task conflict between training and testing in the meta learning setup when using simulated dataset like PASCAL-5^i, which may not exist in a real-world application. Specifically, during training on PASCAL-5^i, test class pixel are labeled as background class due to overlapping images between train and test classes. This causes the model to learn to map class samples from these classes closer to background class. Test classes which have more common images with training classes have reduced test performance. We proposed to overcome this by ignoring the labels of any other foreground classes if present except the target class for each episode rather than masking them as background. This resulted in the model learning a better mapping between input space and an embedded space that clusters the same class features. We validate this by illustrating significantly improved performance on a cluster similarity measure, class IOU scores as well as MeanIOU and BinaryIOU scores. We hope that this work would encourage further research to modify the training and evaluation setup when using simulated dataset like PASCAL-5^i.

References

Chen, W.-Y.; Liu, Y.-C.; Kira, Z.; Wang, Y.-C. F.; and Huang, J.-B. 2019. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232* .

Davies, D. L.; and Bouldin, D. W. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence* (2): 224–227.

Dong, N.; and Xing, E. 2018. Few-Shot Semantic Segmentation with Prototype Learning. In *BMVC*, volume 3.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1126–1135. JMLR. org.

Gidaris, S.; and Komodakis, N. 2018. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4367–4375.

Goldblum, M.; Reich, S.; Fowl, L.; Ni, R.; Cherepanova, V.; and Goldstein, T. 2020. Unraveling Meta-Learning: Understanding Feature Representations for Few-Shot Tasks. *arXiv preprint arXiv:2002.06753* .

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Liu, J.; and Qin, Y. 2020. Prototype Refinement Network for Few-Shot Segmentation. *arXiv preprint arXiv:2002.03579* .

Nguyen, K.; and Todorovic, S. 2019. Feature weighting and boosting for few-shot segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 622–631.

Nichol, A.; and Schulman, J. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* 2: 2.

Qi, H.; Brown, M.; and Lowe, D. G. 2018. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5822–5830.

Rakelly, K.; Shelhamer, E.; Darrell, T.; Efros, A.; and Levine, S. 2018. Conditional networks for few-shot semantic segmentation .

Rusu, A. A.; Rao, D.; Sygnowski, J.; Vinyals, O.; Pascanu, R.; Osindero, S.; and Hadsell, R. 2018. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960* .

Shaban, A.; Bansal, S.; Liu, Z.; Essa, I.; and Boots, B. 2017. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410* .

Siam, M.; Oreshkin, B. N.; and Jagersand, M. 2019. AMP: Adaptive masked proxies for few-shot segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 5249–5258.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, 4077–4087.

Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P. H.; and Hospedales, T. M. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1199–1208.

Tian, Y.; Wang, Y.; Krishnan, D.; Tenenbaum, J. B.; and Isola, P. 2020. Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need? *arXiv preprint arXiv:2003.11539* .

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, 3630–3638.

Wang, K.; Liew, J. H.; Zou, Y.; Zhou, D.; and Feng, J. 2019. Panet: Few-shot image semantic segmentation with prototype alignment. In *Proceedings of the IEEE International Conference on Computer Vision*, 9197–9206.

Zhang, X.; Wei, Y.; Yang, Y.; and Huang, T. 2018. Sg-one: Similarity guidance network for one-shot semantic segmentation. *arXiv preprint arXiv:1810.09091* .