Semi-Supervised Segmentation: Mean-Teacher vs. Uncertainty-Aware Mean-Teacher approaches

Rianur Rahman

1 Introduction

Semantic segmentation is a challenging problem in computer vision, where the ultimate goal is to segment images into predefined classes by assigning each pixel to one of the classes. In this project, we focus on segmentation of images from The Oxford-IIIT Pet Dataset. Specifically, this is done in a semi-supervised setting where we have a mix of labelled and unlabelled data. The semi-supervised setting is one often seen in the biomedical imaging field where quality labelled data can be scarce due to patient confidentiality or challenges in acquiring sufficient manual annotations and such. This also why many of the approaches taken in the project are adapted from the biomedical field. The dataset consists of 37 categories of cat and dog breeds, with about 200 images per category. Each image has a corresponding ground truth segmentation mask that assigns each pixel to one of three possible classes: pet pixel, non-pet pixel, or boundary between pet and non-pet pixel, though in the project we mainly aim to distinguish between pet and non-pet pixels. In the first part we take a Mean teacher-based[8][9] approach which involves two models being trained, a student and a teacher, and wherein the student model learns to mimic the teacher model. The student model learns from the teacher model by minimizing a segmentation loss on the labelled set and a consistency loss (on both sets) with the pseudo labels for the unlabelled set generated by the teacher model. The segmentation loss ensures the model learns to segment the labelled images whereas the consistency loss allows the model to make use of the unlabelled data by ensuring the unlabelled images(and labelled ones) are given consistent predictions by the student model when compared to predictions provided by teacher model on same inputs but with perturbations applied. The student's weights are updated via gradient descent using a combination of the two losses whereas the teacher's weights are updated via an exponential moving average of the student's weights. In the second part, we deal with one of the problems of Mean-Teacher algorithms and its variants as suggested by the UAMT paper[9]. That is, the target labels provided by the teacher model can be unreliable and noisy. In the paper[9], this is alleviated by having the teacher output along the pseudo-labels also a measure of uncertainty for each of them, which essentially ensures only the more reliable ones are used in the consistency loss. Thus, we extend the algorithm to include a measure of uncertainty via different procedures such as Monte-Carlo Dropout(as done by the UAMT paper[9]), Mixup[11] inspired by this paper[1] and Laplace approximation[4] inspired by this paper[10] and we investigate and compare how the performance and segmentation capability of the model changes as the model is trained with uncertainty measures provided by each of these.

2 Methods

We adapted the mean-teacher framework from the UAMT paper[9]. In the paper they work with 3D MR images using a V-Net architecture. Thus we had to adapt the framework to 2D images and change the architecture. The adapted architecture is based on the U-Net[6] and MobileNet-V2 architectures[7]. It uses an encoder-decoder approach for semantic segmentation, where the encoder is a modified pretrained/frozen MobileNet V2 model that extracts high-level features from the input image, and the decoder is just adapted from a standard U-Net model that upsamples the extracted features to generate the final segmentation map. Initially, we were training the encoder along with the decoder, but we made the decision to leave the encoder frozen as this led to the convergence being 4 times faster. The model also includes skip connections that connect corresponding encoder and decoder layers to better preserve spatial information during the upsampling process. Additionally, the decoder includes 2D convolution transpose, batch normalization, dropout with probability 0.5(though in this first part we don't have dropout enabled), and ReLU activation layers. As mentioned in the introduction, we have two parallel models, a student and a teacher both with the same architecture. The student model learns from the teacher model by minimizing a segmentation loss on the labelled data and a consistency loss (on labelled and unlabelled data) with the pseudo labels for the unlabelled set generated by the teacher model. Though we adapt from the UAMT paper[9] which makes extensive use of uncertainty measures for pseudo labels, in the first part we do not actually make use of any uncertainty measures yet. One important thing to note is that we augment the training set images by applying some transformations (horizontal flip of images, colour jittering [with parameters: brightness=0.1, contrast=0.1, saturation=0.1, hue=0.05] and cropping [cropping ratio 0.95]). Also, the images in the dataset have different sizes thus they are pre-processed and resized to 224 by 224 pixels. Furthermore, even though we're trying to build a model that segments and distinguishes the pet vs non-pet pixels, our ground truth masks actually consists of 3 classes not 2. These are: pet pixels, non-pet pixels and boundary between pet and non-pet pixels. The boundary pixels unfortunately are quite fuzzy and noisy thus ambiguous therefore throughout the project we mask them so that we can ignore/filter them out. The Oxford−IIIT Pet Dataset is easily accessible and can be downloaded straight from the torchvision.datasets module. From the module we loaded the trainval and test portions of the data and decided to randomly split the trainval into 0.8-0.2 ratios of training and validation data. The validation and testing data are loaded in batches of size 20, without any shuffling. When dealing with training data we have two batch size parameters, the batch_size and the labelled_batch_size. The batch_size determines the total training batch size whereas the labelled_batch_size determines how many of batch_size images should be from the labelled set. Throughout the project, the labelled_batch_size is set to 25 whereas the batch_size is usually set to 50 (except for when we want to run experiments in a fully supervised setting, wherein we set it to 25 as well). When dealing with the fully supervised case we load the training batches in a

fashion similar to the validation and test batches, with the only difference being that we shuffle the data at each epoch to avoid overfitting. In the fully supervised case, we do not have a teacher model as we do not have unlabelled data that needs pseudo-labelling. However, when dealing with the semi-supervised case the loading of training batches is a different process. We make use of a custom PyTorch sampler "TwoStreamBatchSampler" that allows us to iterate over two sets of indices during training for a given dataset. It allows for the primary set of indices(labelled) to be iterated through once per epoch, while the secondary set of indices(unlabelled) can be iterated through multiple times per epoch. Important to note that the labelled and unlabelled data are concatenated preserving the separation of the two so that they're easier to index and work on. The segmentation/supervised loss is simply calculated as a sum of cross entropy loss and mean (of class dice losses on batch) dice loss (both very standard functions in the field and literature) between the student predicted logits and the ground truth labels on the labelled set. Whereas, the consistency loss is calculated by way of a mean squared difference between the teacher and student predicted logits on the same input images but where the teacher model is passed the input image with added pixel noise/perturbations generated by a 0 centred Gaussian distribution with std 0.1 but with values forced to remain within -0.2 and 0.2 by remapping anything out of the range to the closest range end point(following convention of code[3] accompanying[9]). Important to note that these two logits are not normalised thus we softmax them first before taking the mean squared difference. We denote by N and M the number of labelled and unlabelled points in our training batch. Now that we have defined the supervised and consistency losses, we can define our training loss function on a batch as:
$0.5 \times$ (cross entropy+dice) $+ \lambda(t) \times$(consistency loss) where consistency loss $= \sum_{i=1}^{N+M} L_c(teacher(x_i + noise), student(x_i))$ with x_i the ith image in training and $L_c(a,b) = \frac{\sum_{pixelp} \mathbb{1}(uncertainty_p < H)\|a-b\|^2}{\sum_{pixelp} \mathbb{1}(uncertainty_p < H)}$ where we make use of the uncertainty at pixel p and the threshold H to only use the more certain pixels when computing consitency loss. However note that in the first part we do not compute uncertainties and so by default set them to 0 thus the indicator function lets all the pixels through in the calculation of the consistency loss. $\lambda(t) = scale \times (e^{-5(1-\frac{t}{tmax \times upper})^2})$ is a ramp-up parameter that determines how much we weigh the consistency loss during training at timestep t of total tmax=8000 timesteps. It increases with time, in[9] it is a sigmoid that goes from 0 to 0.1 (scale=0.1, upper=1) over the full length of training however during our experiments we had to use scale=1.0 and upper either 1 or 0.5 to get the consistency loss to come into play at the right time and with enough weight to affect training. Similarly, we had $H = e^{-5(1-\frac{t}{tmax})^2} \times \frac{Umax}{4} + \frac{3}{4}Umax$ with $Umax = log(2)$ as suggested by paper[9] to have a process that ramps up the max uncertainty threshold H from 3/4Umax to Umax as time goes on. The student model is trained by gradient descent on training loss with an SGD optimiser with hyperparameters: learning rate=0.01, momentum=0.9, weight decay=0.0001 as suggested by[9]. The teacher's weights at time t are set as

$\theta_t^{teacher} = \alpha\theta_{t-1}^{teacher} + (1-\alpha)\theta_t^{student}$ an exponential moving average of the student weights with $\alpha = 0.999$ due to the paper[9] suggesting 0.99 as a good starting point. In the second part, we start incorporating uncertainty measures(that we can now use in the consistency loss instead of setting uncertainty of every pixel to 0 like in the first part) via 3 ways. The first one is like[9], monte carlo dropout. We now enable the dropout (with probability 0.5) layers of our teacher model. We then perform T=8 stochastic forward passes on each training batch and return the uncertainty of pseudo-label for a given pixel at time t via $\mu_c = \frac{1}{T}\sum_t \mathbf{p}_t^c$ and $uncertainty = -\sum_c \mu_c log(\mu_c)$ where $\mathbf{p}_t^c$ gives the probability that the given pixel is from class c. The second way is via the mixup algorithm[11][1]. For every given image in our training batch, we sample a value κ from a beta distribution Beta(0.5,0.5). We then set the mixing parameter λ_{mix} as max(κ, 1-κ) to try and retain maximal information about the given image(as the given image will have weighting λ_{mix} during the mixup process, we want to keep the given image dominant). We then randomly sample 16 more images and mix them with the given image using the mixup algorithm with our λ_{mix}. This turns the given image into 16 augmented versions of itself. We then pass the augmented images through the prediction model and softmax the prediction logits. We average the latters across corresponding pixels in the 16 augmented images and finally, we compute the negative entropy at each pixel which gives us a pixel level uncertainty measure. The final way is a Laplace approximation[10][5][2] where we take a Bayesian approach to come up with uncertainty estimates. Bayesian neural networks estimate probability distributions over model parameters, unlike traditional neural networks that only provide a point estimate. To do this, we use an empirical diagonal Fisher approximation with a binary Bernoulli log-likelihood. We estimate gradients by backpropagating on the average log-likelihood of the labelled set per batch. The square of the gradients and prior variance(in our project set to 10^{-2}) are used to estimate the posterior variance using the empirical diagonal Fisher approximation. This estimate is then used along with the MAP estimate of the best parameters(in our case just set this to current model parameters though strictly speaking we cannot claim them to be MAP estimates as we do not use the log-likelihoods as the training loss) to parameterize a single-variate gaussian distribution on each model parameter. We now have a distribution that we can sample models from and we sample 10 such. We then apply the models to each image in the batch and softmax the prediction logits. Again we average these across corresponding pixels and finally, we compute the negative entropy at each pixel which gives us a pixel level uncertainty measure.

3 Experiments

We repeat each experiments 5 times with different seed values. In the first part of the project we conduct 2 experiments, the setup for these 2 is explained in the methods section where we discuss in detail part 1 of the project but we will reiterate any important details. In the first experiment we investigate a fully supervised setting. We do this by setting the number of unla-

belled data points M=0 and training the model for number of labelled points N= $[50, 100, 250, 500, 1250, 2500, 2925]$. As mentioned in the methods section this is when both labelled_batch_size = batch_size = 25 and the teacher model is unused. For this experiment, in terms of ramp-up $\lambda(t)$ the scale=0.1 and upper=1. For the second experiment, we investigate the effects of introducing unlabelled data in the system, now in a mean-teacher setting without measures of uncertainty(thus by default the pixel uncertainties are 0). To do this we train the model with every possible of the 15 combinations of M and N from $M = [0, 1000, 2000]$ and $N = [50, 100, 250, 500, 750]$. Important to note that, labelled_batch_size=25, batch_size = 50 except for any combination where M=0 in which case labelled_batch_size = batch_size = 25. For this experiment, in terms of ramp-up $\lambda(t)$ the scale=1 and upper=0.5. Next we move on to the second part of the project. In this third and last experiment, we introduce measures of pixel uncertainty that can be used in the consistency loss. We train the model 3 times with M=2000 and N=100: once with uncertainty provided by monte carlo dropout, once provided by Laplace and once provided by mixup. We also train with M=0(fully supervised) and N=100 with no uncertainty as a baseline.

4 Results

We report the percentage dice losses on test set averaged over the 5 seed runs for each experiment.

N	**M**	**Mean**	**Std**
50	0	17.3485	0.8277
100	0	14.0556	0.7684
250	0	12.0152	0.1747
500	0	10.9404	0.5737
1250	0	9.6487	0.2657
2500	0	9.0467	0.4285
2925	0	9.0688	0.3442

Table 1. Experiment 1, fully supervised setting.

	M=0	**M=1000**	**M=2000**	**M=0vsM=1000**	**M=0vsM=2000**
N=50	17.35 ± 0.83	16.12 ± 0.61	16.23 ± 0.59	-1.23 ± 0.35	-1.12 ± 0.31
N=100	14.06 ± 0.77	13.08 ± 0.21	13.13 ± 0.36	-0.98 ± 0.73	-0.92 ± 1.00
N=250	12.02 ± 0.18	11.04 ± 0.16	10.98 ± 0.13	-0.97 ± 0.21	-1.04 ± 0.28
N=500	10.94 ± 0.57	10.21 ± 0.30	10.35 ± 0.30	-0.73 ± 0.38	-0.59 ± 0.40
N=750	10.29 ± 0.32	9.69 ± 0.38	9.67 ± 0.32	-0.60 ± 0.19	-0.61 ± 0.24

Table 2. Experiment 2, semi-supervised setting. Mean±std.

	Laplace	Dropout	Mixup	No Uncertainty	Laplace vs No Uncertainty	Dropout vs No Uncertainty	Mixup vs No Uncertainty
M=2000, N=100	14.13 ± 1.38	13.05 ± 0.32	13.60 ± 0.23	13.11 ± 0.15	1.02 ± 1.34	-0.06 ± 0.21	0.49 ± 0.19

Table 3. Experiment 3, uncertainty aware setting. Mean±std.

In the fully supervised setting in Table 1, we see that as the number of labelled data points increases the loss decreases however this trend is broken after N=2500. In the uncertainty unaware setting in Table 2, we see that increasing N for a fixed M always lowers the loss. For a fixed N, going from M=0 to either M=1000 or M=2000 always lowers the loss. However, most of the time going from M=1000 to M=2000 increases the loss. Lastly, in the uncertainty aware setting in Table 3, we notice that adding an uncertainty measure almost always increases the loss than not having an uncertainty measure.

5 Discussion

The results in Table 1 are as expected; in the supervised setting we benefit from a greater amount of labelled data up to a certain point where we start to overfit the data. The results in Table 2 are more interesting. They show that the model benefits from the semi-supervised setting and adding unlabelled data has a significant improvement on performance. However, there seem to be diminishing returns with the amount of unlabelled data (i.e.it is not worth doubling M from 1000 to 2000). It is likely that this is due to the unlabelled data points outnumbering the labelled set too much thus the labelled set not being able to fully capture the statistical distribution of the unlabelled set. It would be interesting to investigate how performance behaves with even higher values of M and whether very high values of M can actually perform better if we have values of N with similar magnitudes(for example does M=2000 outperform M=1000 if we make N much higher than the investigated range). Again, as expected we see that increasing N always benefits the model. Important to note is that small increases N have far greater improvement than very big increases in M. The results in Table 3 seem to indicate that there is no benefit whatsoever in adding uncertainty measures and actually it is detrimental. Even though the dropout scheme does decrease the mean loss slightly, looking at the std of the loss we have to note that this is most likely just noise.

6 Conclusion

In conclusion, it is clear that the semantic segmentation does benefit from the semi-supervised framework and it performs better with more labelled and unlabelled data but more labelled data is more beneficial. Furthermore, the task (at least on this specific dataset) does not benefit from uncertainty aware methods.

References

1. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.A.: Mixmatch: A holistic approach to semi-supervised learning. Advances in neural information processing systems **32** (2019)
2. Hein, M., Hennig, P., Kristiadi, A.: Being bayesian, even just a bit, fixes overconfidence in relu networks. arXiv **2002** (2020)
3. Lequan Yu, Shujun Wang, X.L.C.W.F.P.A.H.: Uncertainty-aware self-ensembling model for semi-supervised 3d left atrium segmentation. https://github.com/yulequan/UA-MT
4. Perone, C.S., Silveira, R.P., Paula, T.: L2m: Practical posterior laplace approximation with optimization-driven second moment estimation. arXiv preprint arXiv:2107.04695 (2021)
5. Ritter, H., Botev, A., Barber, D.: A scalable laplace approximation for neural networks. In: 6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings. vol. 6. International Conference on Representation Learning (2018)
6. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
7. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
8. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. Advances in neural information processing systems **30** (2017)
9. Yu, L., Wang, S., Li, X., Fu, C.W., Heng, P.A.: Uncertainty-aware self-ensembling model for semi-supervised 3d left atrium segmentation. In: Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part II 22. pp. 605–613. Springer (2019)
10. Yun, P., Liu, M.: Laplace approximation based epistemic uncertainty estimation in 3d object detection. In: Conference on Robot Learning. pp. 1125–1135. PMLR (2023)
11. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)