Unsupervied Domain Adaptation for Semantic Segmentation

Arshad Kazi Bhuyashi Deka Sadman Sakib

1. Abstract

In read-world scenario, it might be required to apply a deep learning model on a different distribution of data. Retraining a model to adapt to a different distribution, technically referred as domain adaptation, can be expensive. We propose to adapt a model on unseen distributions in unsupervised manner during test time. We follow the classical Adaptive Batch Normalization method and propose an improvement on that method to utilize the training data in-hand during test time. We have chosen to test our method on semantic image segmentation problem. We have tested our method on Cityscape images and adapted the model for images taken in adversarial conditions such as fog. With Adaptive Batch Normalization and optimization, we get an improvement of 11.7% in average DICE score on foggy Cityscapes dataset. The code is available at this link - https://github.com/sadmankiba/Domain-Adaptation

2. Introduction

In recent years, deep learning models have achieved significant advancements, particularly in tasks involving complex data distributions. These models are adept at learning patterns from the training distribution, but they often struggle when faced with out-of-distribution (OOD) data. For example, a model trained on clear weather cityscape images may perform well under similar conditions but fails to generalize to images captured in foggy or other adverse weather conditions. A straightforward solution to this problem would involve collecting new data from the OOD distribution and retraining the model in a supervised manner. However, this approach is both time-consuming and requires extensive annotation efforts, making it impractical for real-world applications where new OOD scenarios can frequently emerge.

Covariate shift is the issue where the input distribution to a model changes. As the model parameters are less effective in this distribution, normalization is performed to keep the input distribution static. Similarly during mini-batch training, the input to a layer might change across batches that can slow down its training. This problem is referred as Internal covariate shift and batch normalization was proposed to handle this problem [1]. The mean and variance of a batch normalization layer are approximated during training time and they are frozen during test time to produce deterministic output. However, these statistics may not be suitable for OOD data. So, we follow AdaBN [2] to calculate these statistics at test time. Even keeping all other layers frozen and updating only the batch normalization layers has proven effective to adapt to OOD data.

3. Related Works

Test-time adaptation techniques to tackle distribution shift includes two main approaches- 1) re-estimating normalization statistics from current test input, and 2) optimizing model parameters in unsupervised manner. Prediction-time batch normalization [4] recomputes the batch normalization layer statistics for each mini-batch in test-time. However, this requires the batch size to be sufficiently large to provide a good estimation of test data distribution. Test-time normalization [3] interpolates between training and test time statistics by learning the optimal mixture parameter during training with augmented data.

4. Method

Mean and variance at a batch normalization layer is calculated for each batch and the input is normalized. Since it can restricts all layers to same distribution, a scale and a shift parameter are learned.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

To keep test time model performance deterministic, frozen mean and variance from training data are used in test

time. This is approximated over training by applying exponential moving average (EMA) on the statistics calculated from each minibatch.

In AdaBN, the mean and variance are calculated over the whole test domain. To make this efficient, we perform this step using EMA on the statistics from mini-batches in test data. During adaptation, we keep the all layers except batch normalization layers frozen.

However, calculating the statistics only on test data completely forgets the statistics from training data. The statistics from training data can help achieve better generalization. So, we propose an optimization to AdaBN to mix training data with a ratio α, where α is a hyperparameter. The combination of training and test data is shuffled. So, mini-batches can contain samples from both training and test data.

5. Experimental Setup

We implemented two baseline models trained on Cityscape data and Cityscape data with augmentation. The augementations include flip, rotate, color and gamma distortions. The baseline models are used to compare against the AdaBN model. We have reduced the number of classes from 34 to 20 by combining labels of similar classes to get a good performance on the small dataset.

Baseline Models Training: The segmentation models have a U-net architecture loaded from `segmentation_models_pytorch` library with ResNet-50 model as encoder. We used batch size 6, learning rate 0.0001 and Adam optimizer. We used cross-entropy loss for classification and DICE loss for segmentation and added the two losses with equal weights. The training has been done on finely annotated Cityscape dataset with 3500 train, 500 validation and 1000 test images.

Test Time Domain Adaptation: During test-time domain adaptation, the model's layers are frozen, excluding the batch normalization layers. The running statistics (mean and variance) of these layers are dynamically updated using the test-time data. To mitigate the risk of the model fully overfitting to the test domain, a mixing hyperparameter, α, is introduced to compute a weighted average between the test and training data distributions during the adaptation process.

6. Results

6.1. Baseline Results

Table 1 shows the DICE scores on the validation dataset and foggy dataset. The scores are obtained with the models trained on Cityscapes dataset with and without augmentation. The foggy dataset has 3 levels of fog which increases from 0.005 to 0.02. As expected, the DICE score is higher for model with augmentation and decreases as the level of fog increases.

	Normal	**fog(0.005)**	**fog(0.01)**	**fog(0.02)**
w/o aug	0.7	0.68	0.64	0.55
w/ aug	0.72	0.7	0.67	0.62

Table 1. Average DICE score of 20 classes (including background) for the best-trained models

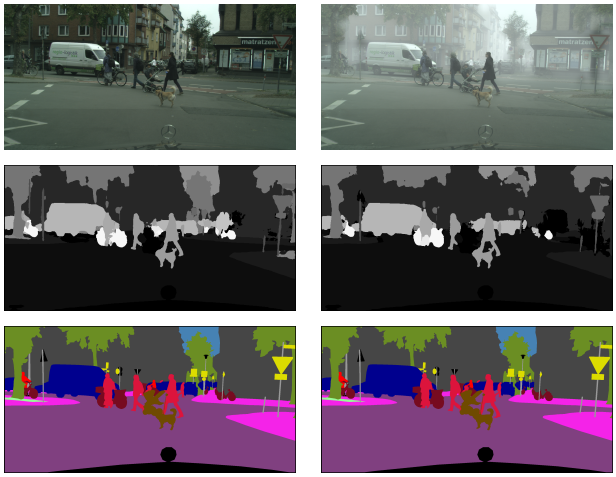
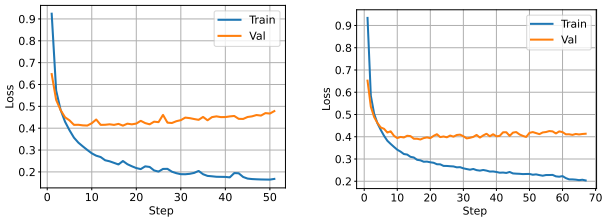


Figure 1. (top) An image from Cityscapes dataset and its foggy version (middle) Semantic segmentation identified for the two images by our model (trained with augmentation) (bottom) ground truth for semantic segmantation of both images

Figure 1 shows the semantic segmentation produced by the model for an image and its foggy version. In the figure, we can see that some objects in the foggy image are misclassified or partially detected. For example, the trees in the background for the foggy image are not segmented by the model. It is likely being classified as part of the buildings due to its unclear boundary.



(a) Without augmentation (b) With augmentation

Figure 2. Loss curve of the model trained with and without augmentation

Figure 2 shows the loss curves of training and validation dataset when the model is trained with and without augmentation. As expected, the validation loss is slightly lower when the model is trained with data augmentation.

6.2. Domain Adaptation Results

We experimented test domain adaptation of with various values of alpha and found that alpha with 0.2 gives the best results on test data for both the models, trained with augmentations and without augmentations. Table 2 shows the average DICE score with different amount of training data mix.

Train:Test	**No mix**	α **= 0.2**	α **= 0.5**	α **= 1**
w/o aug	0.645	0.645	0.639	0.631
w/ aug	0.689	0.693	0.691	0.682

Table 2. Average DICE score of 20 classes (including background) on foggy Cityscapes dataset with different training data mix.

Figure 3 shows the comparison between baseline model outputs vs adapted models with alpha value of 0.2. It is clearly visible that adapted model is able to segment the image with better accuracy.

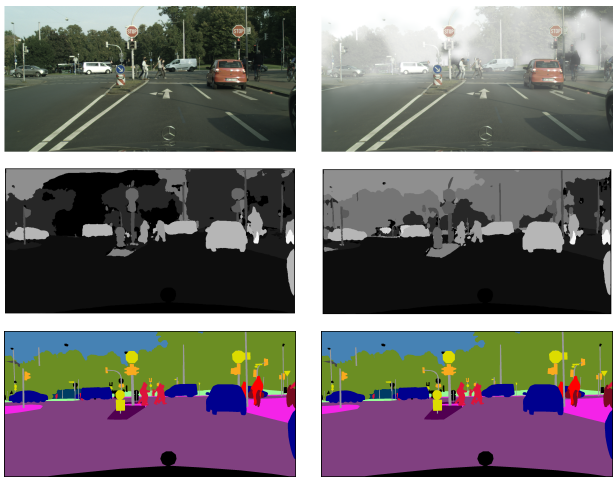


Figure 3. (top) An image from Cityscapes dataset and its foggy version (middle) Semantic segmentation identified for the foggy image by baseline model with augmentation before and after AdaBN (bottom) ground truth for semantic segmantation of both images

7. Conclusion and Future Work

The proposed test-time domain adaptation technique offers a substantial performance boost in scenarios characterized by domain shift. The method's advantage lies in its ability to operate without requiring additional training steps, making it a highly efficient and adaptable solution for real-world applications.

While this study has demonstrated the potential to improve performance in image segmentation, several avenues for future research remain unexplored. A more comprehensive investigation into the relationship between batch size and the mixing parameter α and their impact on performance could be undertaken. Additionally, the performance of this technique on other types of datasets, such as medical images or satellite imagery, can be evaluated to check the generalisability of this technique. Finally, exploring its application to tasks like object detection and instance segmentation presents an intriguing direction for future work.

8. Contribution

All authors contributed to the project development, report and presentation preparation. Arshad Kazi setup the training pipeline. Bhuyashi Deka ran experiments on different settings. Sadman Sakib applied domain adaptation on different training data mix.

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 1

[2] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018. 1

[3] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. Ttn: A domain-shift aware batch normalization in test-time adaptation, 2023. 1

[4] Zachary Nado, Shreyas Padhy, D. Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift, 2021. 1