Reproducibility of "FDA: Fourier Domain Adaptation for Semantic Segmentation" for ML Reproducibility Challenge 2020

Anonymous Author(s)
Affiliation
Address
email

Reproducibility Summary

*The following paper is a reproducibility report for **FDA: Fourier Domain Adaptation for Semantic Segmentation** [10] published in the CVPR 2020 as part of the ML Reproducibility Challenge 2020. The original code was made available by the author <link>. The well-commented version of the code containing all ablation studies performed derived from the original code along with WANDB[1] integration is available at <link> with proper instructions to execute experiments in README.*

Scope of Reproducibility

The central claim of FDA paper was that their method did not require any training to perform domain alignment and just a simple Fourier Transform could achieve state-of-the-art performance in the current benchmarks, when integrated into a standard semantic segmentation model.

Methodology

We used the publicly available source code provided by the authors. Minor changes were made to the source code in order to load the model weights properly. The reproducibility experiments followed the training protocol as described in the original paper. We first tested the pre-trained models provided by the authors in the original GitHub repository. We also trained all the models mentioned in the paper from scratch and evaluated them on the given target dataset to verify the claims given in the paper.

Results

We verified all claims except those involving Synthia dataset. Overall, we were able to reproduce majority of the results mentioned in the paper within 5% error using our optimized strategy compared to what was mentioned in the paper.

What was easy

The code given in the original repository was very straight forward and well documented. The model architecture is easily implemented as it is built on a pre-existing framework (BDL)[5]

What was difficult

The major challenges faced in reproducing the results in the chosen publication were computation bound in nature, concerning to mainly the batch size, long training hours (40-60 hours) and large CPU RAM for pseudo label generation.

Communication with original authors

We contacted the author of the paper via mail regarding some issues and they were quite friendly, gave a prompt reply to all our queries, and even suggested some changes to the training process.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

1 Introduction

Unsupervised domain adaptation (UDA) refers to adapting a model trained with annotated samples from one distribution (source), to operate on a different (target) distribution for which no annotations are given. Simply training the model on the source data does not yield satisfactory performance on the target data, due to the covariate shift. State-of-the-art UDA methods require difficult adversarial training but the method given in the paper simply computes the (Fast) Fourier Transform (FFT) of each input image, and replaces the low-level frequencies of the target images into the source images before reconstituting the image for training, via the inverse FFT (iFFT), using the original annotations in the source domain. As a paragon (upper bound), we use a state-of-the-art model with adversarial training [19]. A variety of sizes as well as a simple multi-scale method consisting of averaging the results arising from different domain sizes are tested in the paper.

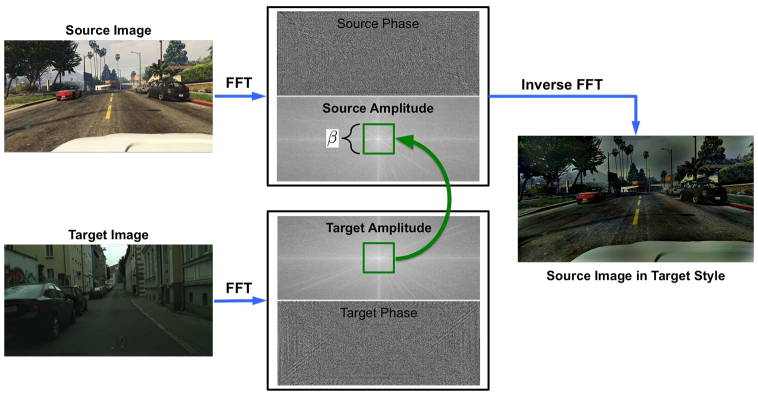

Figure 1: **Spectral Transfer:** Mapping a source image to a target "style" without altering semantic content

2 Scope of reproducibility

The paper revolves around the claim that state of the art performance in unsupervised domain adaptaton can be achieved using the simple method of Fourier transformation which does not require any extra training eliminating the need of difficult adversarial methods.In the paper, models are trained on different values of β and self-supervised training is also implemented. Models are trained on 2 backbones - Resnet101 and VGG16 and results are obtained by doing domain adaptation from GTA5-> CityScapes and Synthia-> CityScapes.

Hence the claims can be summarized as follows:-

GTA5->Cityscapes:

1. The fully trained MBT Resnet101 model achieved 50.45% mIoU when trained using FDA which surpasses the previous state-of-the-art method (BDL) by 1.95%.
2. The fully trained MBT VGG model achieved 42.2% mIoU when trained using FDA which surpasses the previous state-of-the-art method (BDL) by 0.9%.

Synthia->Cityscapes

1. The fully trained MBT Resnet101 model achieved 52.5% mIoU when trained using FDA which surpasses the previous state-of-the-art method (BDL) by 1.1%.
2. The fully trained MBT VGG model achieved 40.5% mIoU when trained using FDA which surpasses the previous state-of-the-art method (BDL) by 0.5%.

3 Methodology

The authors have made the source code associated with the paper publicly available on GitHub as well as links to download their pretrained ResNet-based and VGG-based models. To retrain the models, we followed the training protocol as described in the original paper. The ResNet and VGG, which were pretrained as described in [23] , were employed as the backbone of the model. The codeflow is described in Figures 3 and 4.

3.1 Method descriptions

First a simple Fourier transform is applied which does not require any training and then the loss is described which is used to train the overall semantic segmentation network. As described in **Eq 1**, we can calculate the Fourier Transform for an RGB image and can be efficiently implemented as described in [13].

$$\mathcal{F}(x)(m,n) = \sum_{h,w} x(h,w) e^{-j2\pi\left(\frac{h}{H}m + \frac{w}{W}n\right)}, j^2 = -1 \tag{1}$$

This method requires selecting one free parameter, the size of the spectral neighborhood to be swapped (green square in **Fig. 1**). We define a mask M_β wherein all values are zero except for the center region where $\beta \in (0,1)$:

$$M_\beta(h,w) = 1_{(h,w)\in[-\beta H:\beta H,-\beta W:\beta W]} \tag{2}$$

Given two randomly sampled images x^s D^s, x^t D^t, Fourier Domain Adaptation can be formalized as described in **Eq. 3**, where the low frequency part of the amplitude of the source images $F^A(x^s)$ is replaced by that of the target image x^t.

$$x^{s\to t} = \mathcal{F}^{-1}\left(\left[M_\beta \circ \mathcal{F}^A\left(x^t\right) + (1-M_\beta)\circ \mathcal{F}^A\left(x^s\right), \mathcal{F}^P\left(x^s\right)\right]\right) \tag{3}$$

Source Target Source Image in Target Style

Figure 2: An example of FDA for domain adaptation

Since our training entails different β's in the FDA operations, we call the self-supervised training using the mean prediction of different segmentation networks **Multi-band Transfer (MBT).** This generally lead to an increase of mIoU from its constituent models.

3.2 Datasets

We used GTA5 [7] and Synthia [8] data sets as source datasets and CityScapes[3] as the target dataset. All of the datasets were open-sourced. All the images were resize to 1280x720 and then random cropped to 1024x512.

Dataset	Number of Images
GTA5	24966
Synthia	9400
Cityscapes (train)	2975
Cityscapes (val)	500

Table 1: Datasets

3.3 Hyperparameters

We took the default values of the hyperparameters given in the paper and only changed the values of β and λ_{ent}

3.4 Experimental setup

We ran the training code in Google Colaboratory with GPU (NVIDIA-SMI 460.27.04, Driver Version: 418.67, CUDA Version: 10.1) while we initially generated pseudo labels on a virtual machine in Google Cloud Platform with 40 GB ram and Nvidia Tesla T4 GPU(NVIDIA-SMI 450.51.06 Driver Version:450.51.06 CUDA Version: 11.0). We later optimised the pseudo label generation process so that it could be run on Google Colaboratory as well.

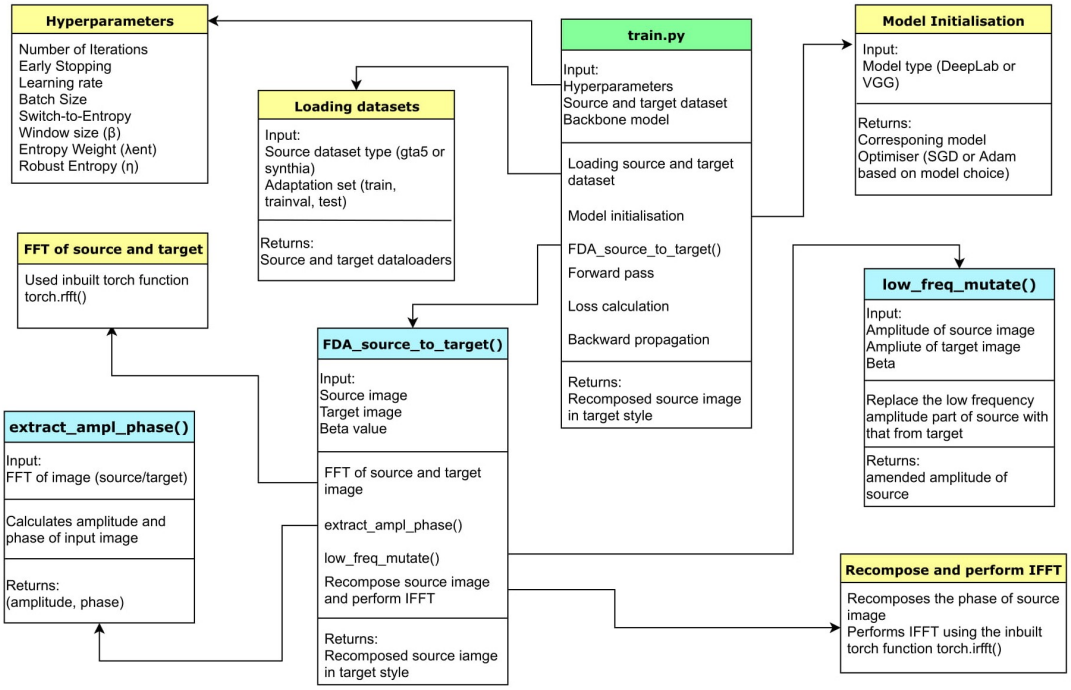

Figure 3: Basic code flow

Table 2: Hyperparameters

Hyperparameter	Value
Number of Iterations	150000
Early Stopping	100000
Learning rate	2.5e-4
Momentum	0.9
Weight decay	0.0005
Power	0.9
Batch Size	1
Switch-to-Entropy	50000
Window size (β)	0.01, 0.05, 0.09
Entropy Weight (λ_{ent})	0.005
Robust Entropy (η)	2

3.5 Computational requirements

The ResNet101-based and VGG16-based models had different requirements on the computational expectations. The Resnet101 models took around 60 hours for training from scratch while the VGG16 models took around 40 hours for training from scratch. The evaluation script took around 30 minutes.

4 Results

The following experiments/ablation studies support the claims made earlier. We varified the claims made in the paper which involved GTA 5 dataset while the claims regarding Synthia dataset were untested. The detailed description of the experiments and their results to support the claim are listed below:-

4.1 Experiments on GTA5 data set

4.1.1 Baseline model on DeepLabV2

We trained DeepLabV2[2] with ResNet101[4] using SGD with 'poly' learning rate scheduler, along with weight decay. Various models were trained for three different β values and self-supervised training was performed at each round. Results of our training are given in Table 3.

Experiment	mIoU (Paper)	mIoU (Final Checkpoint)	Error (%)
0.01(T=0)	44.61	42.71	4.25%
0.05(T=0)	44.6	40.98	8.11%
0.09(T=0)	45.01	41.35	1.33%
0.09 ($\lambda_{ent} = 0$)	44.64	42.62	4.52%
0.09 (SST)	45.42	41.82	7.92%
MBT (T=0)	46.77	44.41	5.04%
0.01(T=1)	47.03	45.02	4.27%
0.05(T=1)	46.8	45.13	3.56%
0.09(T=1)	46.71	45.03	3.59%
MBT(T=1)	48.14	45.38	5.73%
0.01(T=2)	48.77	45.90	5.88%
0.05(T=2)	47.86	45.34	5.26%
0.09(T=2)	47.03	43.61	7.27%
MBT(T=2)	50.45	46.14	8.54%

Table 3: Results of our training of DeepLab model on GTA5 dataset

4.1.2 Baseline model on VGG16

We trained FCN-8s[6] with VGG16[9] backbone using Adam optimizer with learning rate 1e-5 which is decreased by the factor of 0.1 every 50000 steps until 150000 steps. Various models were trained for three different β values and self-supervised training was performed at each round. For VGG16 model, only the final MBT mIoU was mentioned in the paper. We observed an improvement in mIoU for all the 3 models after each round. However in thelast round, we did not observe any improvement in the MBT mIoU. . We present the results of all the rounds of our training below. Results of our training are given in Table 4.

Experiment	mIoU (best)
0.01 (T=0)	35.57
0.05 (T=0)	34.77
0.09 (T=0)	34.13
MBT (T=0)	36.48
0.01 (T=1)	39.23
0.05 (T=1)	39.61
0.09 (T=1)	38.89
MBT (T=1)	40.08
0.01 (T=2)	39.52
0.05 (T=2)	40.96
0.09 (T=2)	39.42
MBT (T=2)	40.06

Table 4: Results of VGG model trained on GTA5 dataset

Comparisons of the final MBT mIoUs for DeepLab and VGG models trained on the GTA5 dataset can be found in Table 5.

Experiment	mIoU (paper)	mIoU (Final checkpoint)	mIoU (best checkpoint)
DeepLab-MBT (T=2)	50.45	46.14	47.42
VGG-MBT (T=2)	42.2	39.42	40.06

Table 5: Results of final MBT mIoUs for DeepLab and VGG.

4.2 Experiments on Synthia data set

We were unable to train on the Synthia dataset due to certain unresolved issues we faced while training. The Synthia dataset had to be trained on 16 classes, unlike the earlier GTA5 source dataset which already had 19 classes. We had to map the labels of the Synthia dataset to match with the classes of Cityscapes. The issue was later resolved as we had communication with the authors. But due to the computational and time constraints we were unable to manage the training of the mordels which required Synthia dataset as the source dataset. The dataloader for the training on Synthia dataset could not be loaded with the pretrained weights. We were unable to overcome this barrier and along with other constraints, we chose to focus on the GTA5 dataset only.

4.3 Improvements (Extra Experiments)

4.3.1 Using best checkpoints

We trained the models using two approaches. In first approach, we took the final checkpoints of the previous round to generate pseudo labels for subsequent rounds. In second approach, we generated pseudo labels using best performing (in terms of mIoU) checkpoints of the previous round. We found that, our latter approach produced results which were closer to the paper's original results and even surpassed them in some cases. However, we could not find the author's approach to generate pseudo labels in the paper. We show the comparisons of the best checkpoints with paper and final checkpoints below in table 6.

Experiment	mIoU (paper)	mIoU (ours)	mIoU (best)	Error
0.01 (T=0)	44.61	42.71	44.36	0.56%
0.05 (T=0)	44.6	40.98	41.79	6.30%
0.09 (T=0)	45.01	41.35	42.84	4.82%
0.09 ($\lambda_{ent} = 0$)	44.64	42.62	42.62	4.52%
0.09 (SST)	45.42	41.82	42.54	6.34%
MBT (T=0)	46.77	44.41	45.89	1.88%
0.01 (T=1)	47.03	45.02	47.37	-0.72%
0.05 (T=1)	46.8	45.13	45.56	2.64%
0.09 (T=1)	46.71	45.03	44.88	3.91%
MBT (T=1)	48.14	45.38	47.81	0.68%
0.01 (T=2)	48.77	45.9	46.65	4.34%
0.05 (T=2)	47.86	45.34	45.85	4.19%
0.09 (T=2)	47.03	43.61	45.26	3.76%
MBT (T=2)	50.45	46.14	47.42	6.00%

Table 6: Results of DeepLab model evaluated using best checkpoints.

4.3.2 Improved the Pseudo label generation code

The pseudo label generation code given in the original repository had extensive hardware requirements and the whole process of generating pseudo labels takes around 35-40 GB of CPU RAM. We optimized the pseudo label generating process such that the code could now be run on Google Colaboratory and less than 15 GB CPU RAM was required. We optimized it by saving the compressed numpy arrays as cache memory rather than in CPU RAM. Around 5.5 GB cache memory was required to save the files and they were deleted after the code was run completely thus not taking any extra disk space. Below is the codeflow for the optimized pseudo label generation process:

We also show the visualized pseudo labels from the original code in the left and our optimized code pseudo labels in the right. The images are exactly same showing that our method was totally correct and also decreased the required CPU RAM.

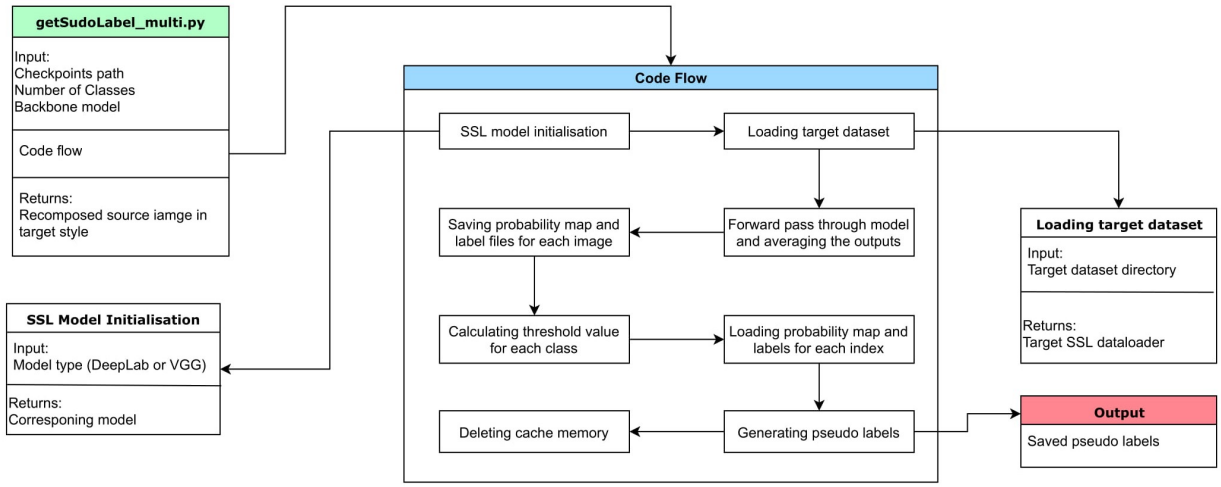

Figure 4: Optimised Pseudo Label Generation Pipeline

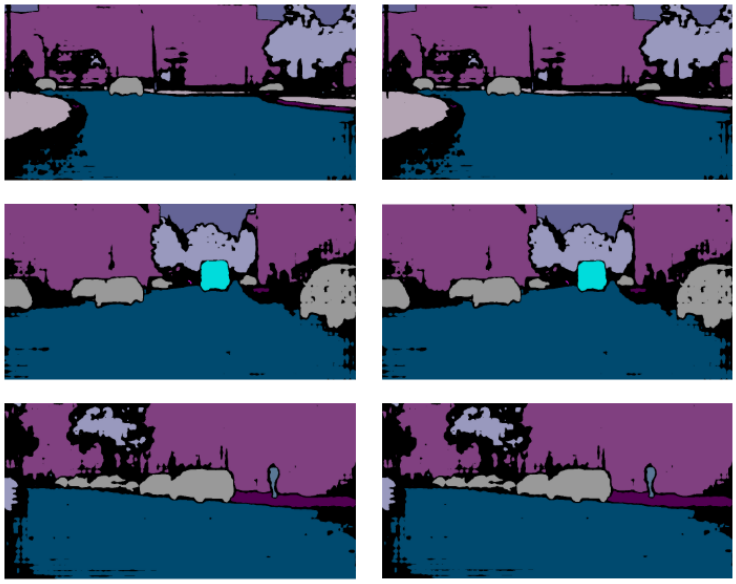

Figure 5: There is no discernible difference between the pseudo labels from the original code (Left) and the ones from our optimised code (Right)

4.3.3 Added automatic saving of Adam optimizer

During the training of VGG models, we found that the mIoU of the checkpoints dipped drastically when the training was resumed after a halt in training . We later found that the intermediate optimizer states were not being saved which is required for resuming especially for Adam optimizer. Hence, we modified the code to save the state dictionary of the optimizer after every 2500 steps. This might have lead to an improvement of our results and made the training process independent of any interruptions.

5 Discussion

Reproducibility of the results posed several obstructions, especially towards students with limited/no access of server grade computations. In order to further study the extent of improvements made by the methods, some other experiments were carried out. We followed the two approaches as mentioned earlier for training. The results produced by using final checkpoints were under 9% of the reported values. While the results produced by using best checkpoints during pseudo label generation improved upon our former approach and were under 6.5% of the reported values in the paper. However in few cases, our later approach surpassed the values reported in the paper. From our results, we can conclude that use of Fourier Transformation does improve the results without any extra training required for unsupervised domain adaptation and add robustness to the domain adaptation problem.

5.1 What was easy

The pretrained models and original source code had been shared by authors. The code given in the original repository was very straight forward and well documented. One can try out the models with little effort as the implementation of the method was quite easy and a standard machine learning framework, PyTorch, was used.

5.2 What was difficult

Initially the computation was started in Google Cloud Platform on a Tesla T4 GPU. A more cost-effective approach was found to by using Google Colaboratory along with a premium Google Drive account for storage of checkpoints labels. The generation of pseudo-labels was computationally very heavy requiring around 35-40GB of memory for execution. However, this was improved by re-writing the code of pseudo label generation, making it possible to run on Google Colaboratory.

5.3 Communication with Authors

We contacted the author of the paper via mail regarding the above mentioned memory issue while generating the pseudo labels and for hardware requirements. We also faced some issues while training on Synthia dataset due to mismatch in the number of classes and contacted the author for the same.

References

[1] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/%5C%7D.

[2] Liang-Chieh Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (June 2016). DOI: 10.1109/TPAMI.2017.2699184.

[3] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: June 2016. DOI: 10.1109/CVPR.2016.350.

[4] K. He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[5] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. "Bidirectional Learning for Domain Adaptation of Semantic Segmentation". In: *arXiv preprint arXiv:1904.10620* (2019).

[6] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[7] Stephan R. Richter et al. "Playing for Data: Ground Truth from Computer Games". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 102–118. ISBN: 978-3-319-46475-6.

[8] G. Ros et al. "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3234–3243. DOI: 10.1109/CVPR.2016.352.

[9] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv 1409.1556* (Sept. 2014).

[10] Yanchao Yang and Stefano Soatto. "FDA: Fourier Domain Adaptation for Semantic Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.