AdaptSGL: Domain Adaptation using Neural Architecture Search and Small Group Learning

Dongxia Wu *
Electrical and Computer Engineering Dept.
University of California, San Diego
`dowu@ucsd.edu`

Aditya Kulkarni *
Electrical and Computer Engineering Dept.
University of California, San Diego
`adkulkar@ucsd.edu`

Balaji Shankar Balachandran *
Electrical and Computer Engineering Dept.
University of California, San Diego
`bbalacha@ucsd.edu`

Abstract

The problem of unsupervised domain adaptation is very crucial in deep learning mainly because of the cost involved in obtaining new labels and the resources needed to re-design models. Models often need hand-crafted architectures for a task and do not generalize well on unseen data distribution (target domain). Inspired by series of work by (Xie et al., 2020a) we aim to achieve unsupervised domain adaptation by using multiple learners in a collaborative environment and Neural Architecture Search techniques. Our method, AdaptSGL, enforces the multiple learners to capture domain invariant features using Maximum Mean Discrepancy (MMD) at multiple learning stages. We experiment with different learning schemes to perform the task of image classification from CIFAR-10 (source domain) to STL-10 (target domain). We observed an improvement of 6% on target domain top-1 accuracy, and at the same time observed 11% boost in source domain test performance.

1 Introduction

The efficacy of machine learning models is currently dependent on the type and amount of labeled data available. Having huge labeled data for every type of data can be labour intensive and economically taxing. Methodologies enabling transferring knowledge from data rich domain to data scarce domains can be very valuable. Techniques that can help generalization to target domain, known as domain adaptation is a widely studied topic by the community that aims in translating superior performance of models from one domain to another (Ganin and Lempitsky, 2015; Glorot et al., 2011). However, deep learning models often fail to maintain their performance on new domains (Nam et al., 2019), which is a key limitation. We aim to enhance the performance of deep learning systems in real-world applications by solving the problem of domain adaptation.

There are several real world scenarios where having multiple learners across domains will be common in the future. Imagine a world where there is a fleet of commercial robots deployed in different parts of the globe solving for a particular task, say autonomous food delivery. There is a notion of domain shift across different cities. In order to deploy a new robot in a new area where much labeled data is not readily available, it would be tempting to reuse existing learners knowledge. Under such scenarios, having a learner that generalizes well to unseen data should improve performance of the newly deployed robot. Also the ability of the best learners in a domain to distill knowledge to other learners across the globe should also help in generalization of each of the learners. Once such knowledge is transferred, each of the domain learners can continually learn domain specific characteristics enhancing their performance on their own domains. There is also a growing interest in autonomous vehicle industry to transfer knowledge from learners trained on simulated environments like CARLA (Dosovitskiy et al., 2017) to real world environments as simulation platforms enable better coverage of situations in the wild. Clearlt there is a pressing need to address this problem of domain adaptation to enable key real-world applications.

The prominent work in domain adaptation field has relied on handcrafted network architectures as in (Kang et al., 2019), (Wang and Breckon, 2019), but these can often lead to sub-optimal solutions (Li and Peng, 2020). However, we know that humans are very good at transferring knowledge obtained from one domain to another domain as well as relearning quickly if such transferring does not happen. Hence, in this project, we aim to explore the effectiveness of training machine learning models inspired by humans learning abilities and Neural Architecture Search techniques for the task of image classification. We use (Xie et al., 2020a) framework that provides mathematical equivalents to human learning techniques: small-group learning (Du and Xie, 2020a) with applications to Neural Architecture Search. We modify the multi learner framework to solve the problem of domain adaptation for image classification.

Our key contribution in this work are:

- Designing ways to incorporate unlabeled target domain data in the Small Group Learning setup for the task of domain adaptation
- Introducing Domain invariance by incorporating Maximum Mean Discrepancy in Small Group Learning setup for the task of domain adaptation

2 Previous Works

The idea of transferring insights from labeled source data to unlabeled target data by characterizing shifts in the source and target domains is an active area of research. Some common shifts between the source and target domains are Co-variate shift, Prior shift and Concept shift (Wouter M. Kouw, 2019). There are several approaches to solve this question of when and how can a classifier generalize from a source to a target domain. The aim of pseudo-labeling Lee et al. (2013); Shi et al. (2018) and self-training approaches McClosky et al. (2006) is to generate pseudo-labels for unlabeled target domain data with a model trained on labeled data and introduce regularization effects. Also, Xie et al. (2020b) show that self-training can be used to improve the performance of benchmark supervised classification tasks.

By incorporating methods to capture domain discrepancy at various layers of the learners, we can explicitly promote domain invariant feature learning during training. (Mingsheng Long, 2015) proposed a new Deep Adaptation Network architecture, in which hidden representations of all task-specific layers are embedded in a reproducing kernel Hilbert space where the mean embeddings of different domain distributions can be explicitly matched. Mingsheng Long (2017) proposed a new approach to domain adaptation in deep networks that can jointly learn adaptive classifiers and transferable features from labeled data in the source domain and unlabeled data in the target domain by relaxing a shared-classifier assumption made by previous methods.

Unsupervised Domain Adaptation (UDA) involves incorporating unlabeled data of target domain during training process to learn knowledge of it. Based on number of source domains, domain adaptation methods can be split into two categories: single-source and multi-source. He et al. (2021) proposed a novel multi-source domain adaptation framework based on collaborative learning for semantic segmentation task where they leverage multiple learners who are experts in their domains. The approach of having multiple independent learners that are trained to solve the same task in a collaborative environment can perform better than stand alone models as each model has an opportunity to capture insights that other learners might miss. The training procedure is costly but such approaches can greatly help in generalization to unseen target domains. French et al. (2017) uses self-ensembling approaches by introducing two networks: a student network and a teacher network. Such approaches seems to perform well.

Most of the prominent works in domain adaptation has relied on handcrafted network architectures as in (Kang et al., 2019), (Wang and Breckon, 2019), but these can often lead to sub-optimal solutions (Li and Peng, 2020). One of the key issues in domain adaptation problem is in the way to come up with a domain-agnostic neural architecture that performs well on target domain. Incorporating George Kyriakides (2020) Neural Architecture Search techniques seems intuitively a good approach for this problem of domain adaptation that is expected to minimize empirical risk on unlabeled target domain.

Inspired by (Xie et al., 2020a) framework that provides mathematical equivalents to human learning techniques: Small-Group Learning (Du and Xie, 2020a) in particular, we come up with ways to incorporate unlabeled target domain data, Maximum Mean Discrepancy, and Neural Architecture Search techniques in a multi learner collaborative environment. In this work we explore the effectiveness of different training schemes for the above setup.

3 Methodology

We use the mathematical framework as delineated in Du and Xie (2020b) as our base framework, which includes K-learners and three stages of optimization. Each learner has a total of two models with weight V, W that have the same architecture A as described in Liu et al. (2018). The training process includes three stages as shown in Fig 1, 2. In the first stage of training, the models are trained in a fully supervised manner using source data and labels. For our method (AdaptSGL) we use regularization based on the technique of Maximum Mean Discrepancy(MMD) Gretton and ¨ Smola (2012) as described below to ensure that the model learns domain invariant features. In the second step of training, the trained optimum weights V^* are used to infer labels ("pseudo-labels") for target domain data. These pseudo-labels, along with source data (human generated labels) are used to perform supervised training of model W. In the final stage each learner optimizes their architectures by minimizing the validation loss on the validation split of the source data. In the sections below we describe the mathematical representation of the vanilla SGL method and our proposed improvements.

3.1 Small Group Learning for Domain Adaptation

SGL setup has K learners, and let each learner have a learnable architecture A_k and two sets of learnable networks weights V_k, W_k. Let the D_s^{tr} be the source training data, D_s^{val} be corresponding validation dataset, D_t^u be the unlabeled target domain dataset. Then the first stage is defined as the optimization:

$$V_k^* = min_{v_k} L(V_k, A_k, D_s^{tr}) \tag{1}$$

Where L is the cross entropy loss. The second stage of optimization involves use of pseudo labels for the unlabeled target domain $D_t^u = \{x\}_{i=1}^N$ dataset. The labels obtained are $f(x_i; V_k^*(A_k))$. Where $f(.; V_k^*(A_k))$ is the optimized model from stage one. The pseudo labels generated by each learner and the human labeled source data is used to supervise the training of W_k for all the other learners. This results in an optimization:

$$W_k^*(A_k, \{V_j^*(A_j)\}_{j\neq k}^K) = min_{W_k} L(W_k, A_k, D_s^{(tr)}) + \lambda \sum_{j\neq k}^{K} L(W_k, A_k, D_t^{(pl,j)}) \tag{2}$$

Where the first term is a supervision based on human labeled data and the second term, $D_t^{(pl,j)}$, is the pseudo-label generated by learner j, and λ is the tradeoff parameter between human labeled data and the pseudo-labeled data. Next for stage three, the learners optimize the architectures by minimizing the validation losses:

$$min_{\{A_k\}_{k=1}^{K}} \sum_{k=1}^{K} L(W_k^*(A_k, V_j^*(A_j)_{j\neq k}^{K}), A_k, D_s^{(val)}) \tag{3}$$

Similar to Liu et al. (2018), we represent the architecture A_k of the learner in a differential way. Since our source dataset is same as the dataset used in Du and Xie (2020b) we use the same search space as well. The search space is composed of number of building blocks. The output of each block is associated with weights indicating the importance of each block. After training, the blocks of model W with largest importance are retained to form the final architecture. The training steps are summarized in

Algorithm 1: SGL + target domain pseudo labels for Domain Adaptation - k^{th} Learner

```
Create a mixture operation $o^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each cell and build $A_k$
for $j = 1 : 5$ do
    Update weights V$_k \rightarrow \frac{\partial}{\partial V_k} L(V_k, A_k, D_s^{tr})$
end for
while not converged do
    Update weights V$_k \rightarrow \frac{\partial}{\partial V_k} L(V_k, A_k, D_s^{tr})$
    Update weights W$_k \rightarrow \frac{\partial}{\partial W_k} L(W_k, A_k, D_s^{tr}) + \lambda \sum_{j\neq k}^{K} L(W_k, A_k, D_t^{pl,j})$
    Update weights $\alpha_k \rightarrow \sum_{k=1}^{K} L(W_k(A_k, V_j(A_j)_{j\neq k}^{K}), A_k, D_s^{(val)})$
end while
```

3.2 Maximum Mean Discrepancy (MMD)

Suppose we have samples $X = (\,x_1, x_2, ..., x_n)$ and $Y = (\,y_1, y_2, ..., y_m)$ drawn from distributions $\mathbb{P}_X$ and $\mathbb{P}_Y$, MMD Gretton et al. (2012) is a criterion that compares the two distributions by embedding each distribution into Reproducing Kernel Hilbert Space (RKHS). The kernel embeddings of distributions are to embed a distribution P into an RKHS $\mathbb{H}_k$ with a kernel k. P is represented as an element μ_P in the RKHS where $\mu_P := E_{x\sim\mathbb{P}}\left[k\left(.,x\right)\right] \epsilon H_k$ where k is kernel embedding and $E_{x\sim\mathbb{P}}\left[f\left(x\right)\right]$ denotes expectation of function f with respect to x. Given two distributions $\mathbb{P}_X$ and $\mathbb{P}_Y$, the MMD between these distributions are computed as

$$\begin{aligned} MMD\left(\mathbb{P}_X, \mathbb{P}_Y\right) &:= \left\|\mu_{\mathbb{P}_X} - \mu_{\mathbb{P}_Y}\right\|_{H_k} \\ &= \left\|E_{x\sim\mathbb{P}_X}\left[k\left(.,x\right)\right] - E_{y\sim\mathbb{P}_Y}\left[k\left(.,y\right)\right]\right\|_{H_k} \end{aligned} \tag{4}$$

where $\|.\|_{H_k}$ is the RKHS norm. When the kernel k is characteristic, $MMD\left(\mathbb{P}_X, \mathbb{P}_Y\right) = 0$ if and only if $\mathbb{P}_X = \mathbb{P}_Y$. When the kernel k is the polynomial kernel of degree d, $MMD\left(\mathbb{P}_X, \mathbb{P}_Y\right) = 0$ means that up to the d-th moments of the two distributions, $\mathbb{P}_X$ and $\mathbb{P}_Y$, are the same Nishiyama et al. (2014). The empirical estimate of the squared MMD using two datasets, X and Y, is given by

$$\begin{aligned} MMD^2\left(X, Y\right) &= \left\|\sum_{n=1}^{N} \frac{k\left(.,x_n\right)}{N} - \sum_{m=1}^{M} \frac{k\left(.,y_m\right)}{M}\right\|_{H_k}^2 \\ &= \sum_{n,m}^{N} \frac{k\left(x_n, x_m\right)}{N^2} - 2\sum_{n,m}^{N,M} \frac{k\left(x_n, y_m\right)}{NM} + \sum_{n,m}^{M} \frac{k\left(y_n, y_m\right)}{M^2} \end{aligned} \tag{5}$$

3.3 AdaptSGL

Our method tries to incorporate the notion of domain invariant features from MMD and SGL. This essentially involves modifying the optimizations involved in different stages of SGL by providing

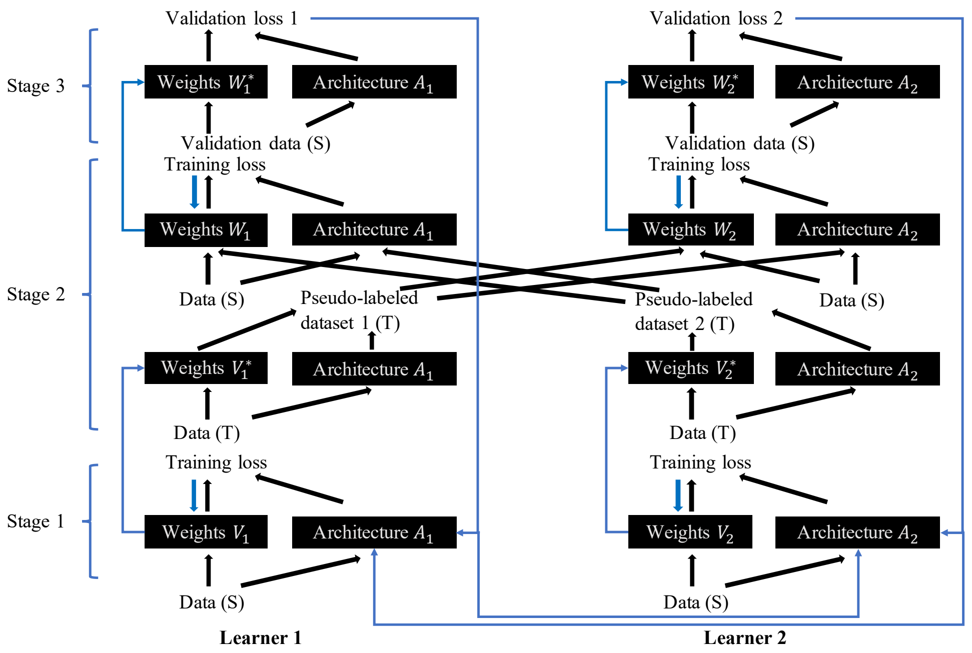


Figure 1: default setup of small-group learning. The black arrows denote the processes for loss calculation and predictions. The blue arrows denote the processes for backpropagation. Data (S) denotes the data used in source domain. Data (T) denotes the data used in target domain.

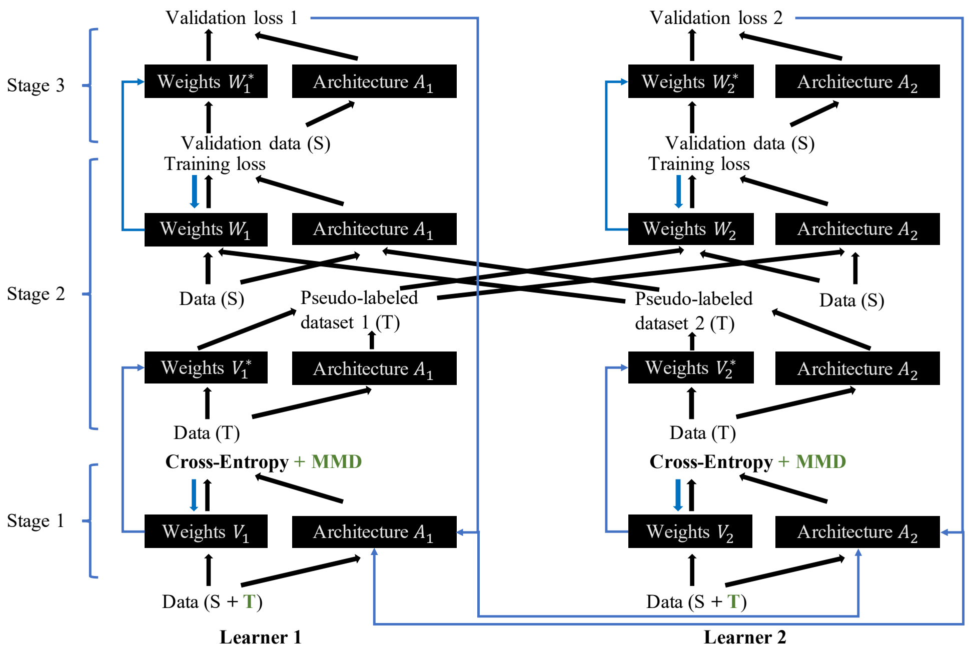


Figure 2: Setup of AdaptSGL. The input at Stage 1 combines both dataset in source domain and target domain to compute the loss of cross-entropy and MMD. Same training loss is used at Stage 2 (cross-entropy + MMD).

Algorithm 2: MMD + SGL for Domain Adaptation - k^{th} Learner

```
Create a mixture operation $o^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each cell and build $A_k$
for $j = 1:5$ epochs do
    Update weights V$_k \rightarrow \frac{\partial}{\partial V_k} L(V_k, A_k, D_s^{tr})$
end for
while not converged do
    Update weights V$_k \rightarrow \frac{\partial}{\partial V_k} L(V_k, A_k, D_s^{tr})$
    Update weights W$_k \rightarrow \frac{\partial}{\partial W_k} L(W_k, A_k, D_s^{tr}) + \lambda \sum_{j \neq k}^{K} L(W_k, A_k, D_t^{pl,j}) + \gamma MMD^2$
    Update weights $\alpha_k \rightarrow \sum_{k=1}^{K} L(W_k(A_k, V_j(A_j)_{j \neq k}^{K}), A_k, D_s^{(val)})$
end while
```

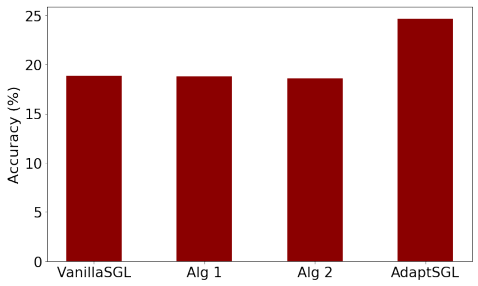
regularization. Hence the optimization of our method involves
Stage one:

$$V_k^* = min_{v_k} L(V_k, A_k, D_s^{tr}) + \gamma MMD^2 \tag{6}$$

Where γ is a hyper-parameter that controls the importance of MMD and supervision. More experiments related to this can be found in the discussion and limitation section.

Algorithm 3: Adapt SGL for Domain Adaptation - k^{th} Learner

```
Create a mixture operation $o^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each cell and build $A_k$
for $j = 1:5$ epochs do
    Update weights V$_k \rightarrow \frac{\partial}{\partial V_k} L(V_k, A_k, D_s^{tr}) + \gamma MMD^2$
end for
while not converged do
    Update weights V$_k \rightarrow \frac{\partial}{\partial V_k} L(V_k, A_k, D_s^{tr}) + \gamma MMD^2$
    Update weights W$_k \rightarrow \frac{\partial}{\partial W_k} L(W_k, A_k, D_s^{tr}) + \lambda \sum_{j \neq k}^{K} L(W_k, A_k, D_t^{pl,j}) + \gamma MMD^2$
    Update weights $\alpha_k \rightarrow \sum_{k=1}^{K} L(W_k(A_k, V_j(A_j)_{j \neq k}^{K}), A_k, D_s^{(val)})$
end while
```



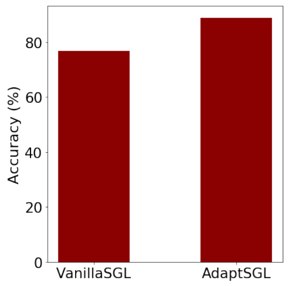


Figure 3: Left: Accuracy among four different approaches on the target domain. Right: Accuracy between VanillaSGL and AdaptSGL on the source domain.

4 Experiments and Results

We use CIFAR-10 as the source domain and use STL-10 as the target domain in all experiments for the task of domain adaptation for image classification task. We tried numerous experiments that modified the training setup in how the data was fed to stage 1 and 2 of the SGL. Also we tried approaches involving learning domain invariant features. In all these experiments the test

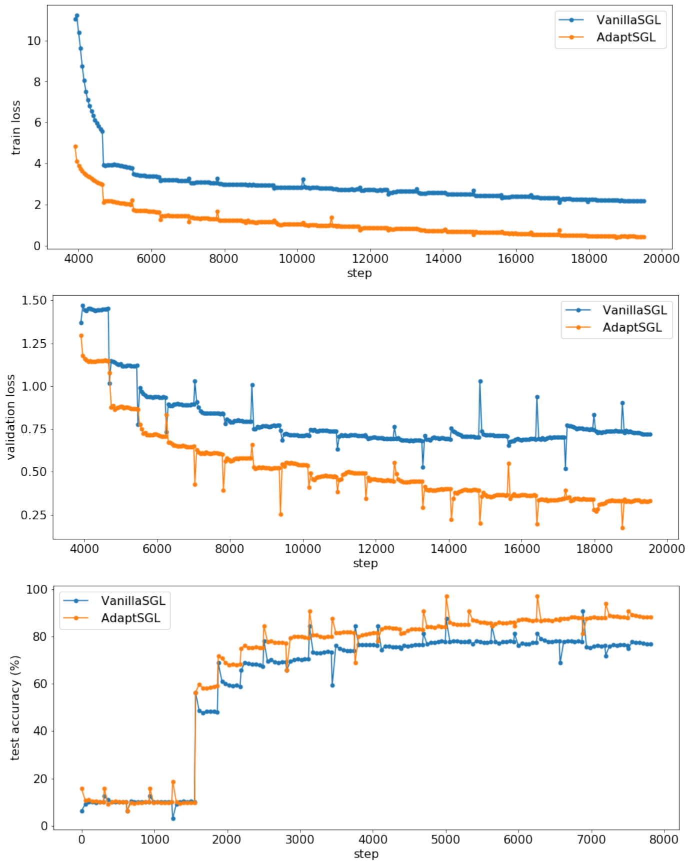


Figure 4: Loss plot (train+validation) and test accuracy plot (source domain).

environment was the same. The model was evaluated for its performance on the STL-10 dataset (target domain). We use train batch size of 32 with 25 epochs for all experiments and use the first 5 epochs of pretraining in stage 1. We experimented with a reduced search space by removing kernels of size 5x5, 7x7 to enable faster training. Other hyperparameter setup is the same as (Du and Xie, 2020b). We did not observe any significant drop in performance after reducing the search space for the baseline model hence we chose to run all the experiments based on the reduced search space. In this section we briefly discuss about the main experimental setup that we used and more detailed discussion can be found in the next section. The test performance for all the methods can be seen in Fig 3. Our code is implemented using PyTorch and is publicly available in repository GitHub.

First approach is the baseline method, which uses vanilla SGL. We monitored the training loss and found that both the learners converged well. By monitoring the validation loss, we observed that the Architecture search was also doing well. However we noticed that the naive setup does not do generalize well on the target domain data which we found during the test part.

Second approach involves incorporating unlabeled target domain data in stage 1 and generating pseudo-labels. The pseudo-labels from a particular learner is then fed to other learners in stage 2. Here the model weight W_k is trained in a supervised approach using both the source domain labeled data and the target domain data pseudo-labels. We observed that the effect of introducing Pseudo labels did not significantly improve test and validation performance compared to baseline Vanilla SGL.

In the third approach we enforced explicit domain invariance by introducing MMD to the training loss only at the second stage. The weight of the MMD loss is fixed at 10. This method builds on top of second approach and we found that this approach also did not help significantly in the performance.

Fourth approach, our method, AdaptSGL involves incorporating MMD at both stage 1 and stage 2 of SGL setup as well as incorporating target domain data during training. The weight of the MMD loss increases from 4 to 100 linearly during the training process. As in Fig 3, this approach worked the best for us as we observed significant increase in performance for target domain top-1 accuracy.

5 Discussion and Limitation

To enable faster training, we reduced the Neural Architecture Search space by removing kernels of size 5x5, 7x7. On comparison with Full Search Space techniques, we did not observe a significant dip in performance. So, we decided to continue with Reduced Search space for all the experiment.

Initially we experimented with Vanilla SGL and expected that it would perform well on the target domain due to its inherent design which takes advantage of multiple learners. We observed a top-1 accuracy on STL-10 data as 18.88 %. In order to improve the performance, we looked at the semi-supervised approach where we incorporated target domain pseudo-labels in the training. We observe a performance of 18.76 % on the target domain were not reliable. This can be attributed to the pseudo labels learnt during the first stage were not being reliable hence making the training harder.

To overcome this, we introduced MMD in stage 2, the stage that is used to perform the end task of image classification. The weight of MMD loss is fixed at 10. We expected improved performance but we observed an accuracy of 18.57 %. The reason is that we didn't add MMD loss in stage 1 to generate valid pseudo labels.

We then came up with AdaptSGL, which performed the best. Here we used MMD at both the stages. The hyperparameter involving weightage given to MMD loss was set to linearly increase from 4 to 100 with epochs. Setting a lower weight to MMD loss initially would mean that the training loss is majorly driven by the cross entropy loss from the labeled source domain. This would help in ignoring target domain's influence on the initial stages of training. We observed an increase of 6 % on STL test top-1 accuracy. This is mainly because the multiple learners captured domain invariant features. We can confirm that the learners now learn domain invariant features as we observed an significant boost of 11 % on the performance on source domain.

We also monitored the loss throughout the multiple stages of training. This can be seen in the Fig 4. It clearly shows that AdaptSGL not only performs well on target domain but also on source domain as we see an improvement from baseline VanillaSGL. Also we see that the validation loss decreases much faster and is not trapped in the local optima. AdaptSGL also outperforms VanillaSGL consistently throughout the training process as shown in the Fig 4. This indicates that the model has truely learnt domain invariant features.

We also observed that the two learners in the setup performed similarly on both source and target domain. However our model still has lot of room for improvement because the current state-of-the-art approaches for domain adaptation achieve results close to 90 %. We think our results can be improved significantly by using larger batch size and introducing more techniques from current domain adaptation literature.

6 Conclusion

We tried to solve the problem of unsupervised domain adaptation with little to no hand-crafted architecture tuning. We conducted various experiments to perform the task of domain adaptation for image classification using Small Group Learning and report the performance of the learners on the target domain. We find that VanillaSGL does not necessarily generalize well but using well established techniques in domain adaptation like MMD and target pseudolabels helped to improve the performance on target domain without hurting performance on source domain, in fact improving it.

7 Acknowledgements

We would like to thank Prof. Pengtao Xie for his guidance, insights and continued support throughout the completion of the project.

References

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16.

Du, X. and Xie, P. (2020a). Small-group learning, with application to neural architecture search. *arXiv preprint arXiv:2012.12502*.

Du, X. and Xie, P. (2020b). Small-group learning, with application to neural architecture search.

French, G., Mackiewicz, M., and Fisher, M. (2017). Self-ensembling for visual domain adaptation. *arXiv preprint arXiv:1706.05208*.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR.

George Kyriakides, K. M. (2020). An introduction to neural architecture search for convolutional networks. *arXiv:2005.11074*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

Gretton, A.; Borgwardt, K. M. R. M. J. S. B. and ¨ Smola, A. (2012). A kernel two-sample test. *JMLR 13(Mar):723– 773*.

He, J., Jia, X., Chen, S., and Liu, J. (2021). Multi-source domain adaptation with collaborative learning for semantic segmentation. *arXiv preprint arXiv:2103.04717*.

Kang, G., Jiang, L., Yang, Y., and Hauptmann, A. G. (2019). Contrastive adaptation network for unsupervised domain adaptation.

Lee, D.-H. et al. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3.

Li, Y. and Peng, X. (2020). Network architecture search for domain adaptation.

Liu, H., Simonyan, K., and Yang, Y. (2018). DARTS: differentiable architecture search. *CoRR*, abs/1806.09055.

McClosky, D., Charniak, E., and Johnson, M. (2006). Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159.

Mingsheng Long, Han Zhu, J. W. M. I. J. (2017). Unsupervised domain adaptation with residual transfer networks. *arXiv:1602.04433*.

Mingsheng Long, Yue Cao, J. W. M. I. J. (2015). Learning transferable features with deep adaptation networks. *arXiv:1502.02791v2*.

Nam, H., Lee, H., Park, J., Yoon, W., and Yoo, D. (2019). Reducing domain gap via style-agnostic networks. *arXiv preprint arXiv:1910.11645*.

Nishiyama, Y., Kanagawa, M., Gretton, A., and Fukumizu, K. (2014). Model-based kernel sum rule: Kernel bayesian inference with probabilistic models. *arXiv preprint arXiv:1409.5178*.

Shi, W., Gong, Y., Ding, C., Tao, Z. M., and Zheng, N. (2018). Transductive semi-supervised deep learning using min-max features. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Wang, Q. and Breckon, T. P. (2019). Unsupervised domain adaptation via structured prediction based selective pseudo-labeling.

Wouter M. Kouw, M. L. (2019). An introduction to domain adaptation and transfer learning. *arXiv:1812.11806v2*.

Xie, P., Du, X., and Ban, H. (2020a). Skillearn: Machine learning inspired by humans' learning skills.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. (2020b). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698.