ICLR Reproducibility Challenge 2018
Incremental Learning through Deep Adaptation
CS 395T Final Project

Santhosh K. Ramakrishnan
The University of Texas at Austin
srama@cs.utexas.edu

Venkata Sailesh Sanampudi
The University of Texas at Austin
sailesh101195@gmail.com

1. Introduction

Most of the existing approaches for Image Classification train a network from scratch or use pre-trained weights from a network trained on another related task / dataset such as ImageNet. In order to perform well on a diverse range of datasets (Figure 1), a network has to be trained individually for each of the datasets. This is often expensive computationally or increases the number of parameters significantly as the number of datasets increase. In [8], the authors suggest a novel architecture termed as the Deep Adaptation Network (DAN) to overcome this issue. DAN adapts the weights of a network trained on one dataset (base network) for a target dataset using a Controller Network. More specifically, new convolutional filters are defined as the linear combination of filters from the base network. This serves as the Controller Network for one target dataset. Similar weights can be learned for each target dataset and an overall Dataset Decider selects the Controller to use for a given input image (refer Figure 2). The experimental results suggest that these Deep Adaptation Networks perform on par with models trained individually per dataset at a much lower parameter cost ($2.5 - 4\times$ reduction).

2. Reproducibility Plan

In this work, we focus on reproducing the results provided in Table 1 of [8]. More specifically, we want to verify if the accuracy numbers reported are reproducible based on the experimental settings and method articulated in the paper. We are also interested in answering the following questions:

1. How does the proposed method perform as the number of fully connected layers change?
2. How effective are randomly initialized base networks?
3. If randomly initialized filters serve as a competitive base network, how does a lower capacity CNN learned from scratch perform?

3. Implementation Details

We have made our implementation available on GitHub [1]. The method is implemented in PyTorch [2]. The VGGB implementation and pre-trained weights were obtained from torchvision [3]. The `vgg13_bn` model was used in the experiments performed. We report performance by adhering to the hyperparameter settings reported by the authors. We also further tune the hyperparameters and report our performance on the same tasks. The hyperparameters are shown in Table 1. It must also be noted that this method is equivalent to adding a 1x1 convolution layer immediately after each convolution and learning it's weights. We have provided both implementations in our codebase. However, we restrict out experiments to the method suggested in the paper.

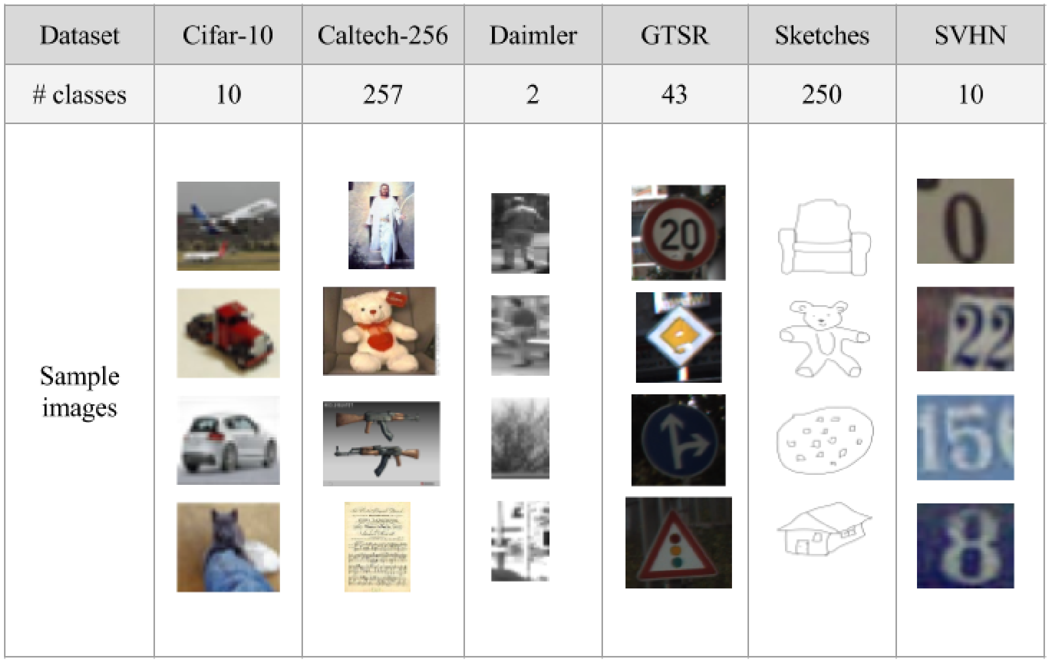
4. Experiments

For a preliminary verification of our implementation of the Deep Adaptation Network, we count the number of parameters in our model. The results are shown in Table 2 for different number of fully connected (FC) layers in the base network. The target datasets that considered were Sketches [1], Caltech-256 [2], SVHN [7], Omniglot [5], Daimler pedestrians(Dped) [6], Plankton [4], GTSR [3] and Cifar-10 [4]. The number of parameters in our 1 FC setting is lesser than the numbers from [8]. This is possibly because we counted the exact number of parameters in the FC layers for each dataset (as opposed to taking one specific ratio and multiplying it by the number of datasets).

4.1. Verifying Table 1 from DAN [8]

We limit our experiments to the following datasets: Cifar-10 (C-10), GTSR, SVHN, Caltech, Daimler Pedes-

[1] https://github.com/srama2512/deep-adaptation-networks-dl
[2] http://pytorch.org/
[3] http://pytorch.org/docs/master/torchvision/models.html
[4] https://github.com/Planktos/PlanktonSet-1.0



Dataset	Cifar-10	Caltech-256	Daimler	GTSR	Sketches	SVHN
# classes	10	257	2	43	250	10
Sample images						

Figure 1. Classification Datasets with large diversity of classes and images

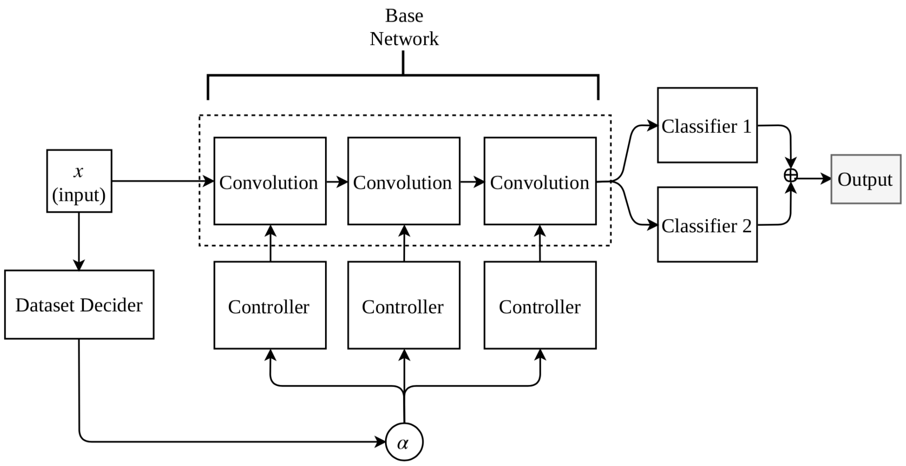


Figure 2. Proposed architecture from [8] consists of a Base Network and multiple Controllers. A Dataset Decider chooses the Controller based on the input image.

Hyperparameter	From [8] (v1)	Ours (v2)
Image Size	64x64	64x64
Pre-processing	Mean subtraction and standard deviation normalization	Mean subtraction and standard deviation normalization
Learning Rate (LR)	1e-3 / 1e-4	1e-3
Epochs	Upto 30	Upto 90
LR Schedule	Halve after 10 epochs	Halve after 30 epochs
Batch size	-	32 / 100
Optimizer	Adam	Adam
LR scaling	-	Reduce by 10 for feature extraction layers
Random init.	-	Xavier / Normal
Weight decay	-	1e-5

Table 1. Hyperparameter settings from [8] and Ours (Unreported hyperparameters as ”-”).

	# param			
Base Network	1 FC	2 FC	3 FC	From [8]
VGG-B Sketches	2.29	3.99	4.91	2.54
VGG-B Caltech	2.29	3.98	4.90	2.54
VGG-B Imagenet	2.27	4.07	5.11	2.76
VGG-B Noise	1.27	3.07	4.11	1.76

Table 2. Relative number of parameters in Deep Adaptation Networks with different base networks.

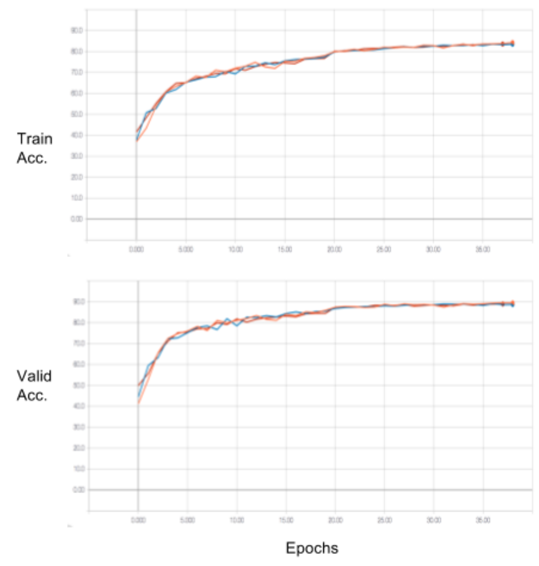
trians (Dped) and Sketches. When the train, valid and test splits were not available, we selected 80% for train and 20% for test. 20% of the train set was reserved for validation. We train our models based on the hyperparameter settings v1 and v2 as explained in Table 1. In Table 4, we list the results for the various models in 3 rows. The first row lists the results from [8], the second and third rows list the results we obtained from hyperparameter settings v1 and v2 respectively. As we can see, our models are able to match / beat the reported results using the v1 parameter settings (which the authors reported) in some cases. We were unable to match their performance with the v1 settings on GTSR, Caltech, Dped and Sketches. By significantly increasing the number of training epochs (v2 settings), we did eventually match their performance (within 1%) on cases such as VGG-B(S) for C-10, SVHN, DAN-noise for Dped and DAN-sketches for SVHN. We also came closer to matching their performance on VGG-B(P) for Dped, VGG-B(S) for Sketches and DAN-imagenet for GTSR. However, there were a few cases where we could not match their performance. Particularly, our performance in Caltech was significantly worse. This is possibly because we may not have used the correct train, valid and test splits (not specified in [8]).

4.2. Varying the number of Fully Connected (FC) layers

Adding more FC layers increase the number of parameters and provide greater model complexity. In our previous experiment, we restricted our models to 1 FC layer as suggested in [8]. Here, we also evaluate models with 3 FC layers to see if the increased number of parameters gives us any extra benefits (at the expense of much more parameters). Our results are enumerated in Table 5. Models with 1 FC layer perform better on C-10, Caltech and Sketches. We get mixed results on GTSR, SVHN and Dped. The large increase in parameters causes the 3 FC layer models to overfit. As a result, it is often better to simply use 1 FC layer for the datasets used.

4.3. Effectiveness of noise base networks

From our results shown in Table 4, we can clearly see that the noise base networks are performing very well, oftentimes beating the imagenet base networks. Our results on DAN-noise are also significantly higher than the results reported in [8], especially on Cifar-10 and Sketches. To explore if these were random occurrences, we perform our re-run our DAN-noise models on Cifar-10 and Sketches datasets with different random seeds (123, 234, 345). The models are trained for only 40 epochs (learning rate scaled down by 2 after 20) with the framework set to ensure reproducibility (at the cost of slower training). Results are shown in Figures 3 and 4. The train and validation accuracy plots are shown as a function of epochs. The random seeds, corresponding colors and the test accuracy are reported as well. The results remain relatively stable for the different random seeds, indicating that DAN-noise is indeed effective.

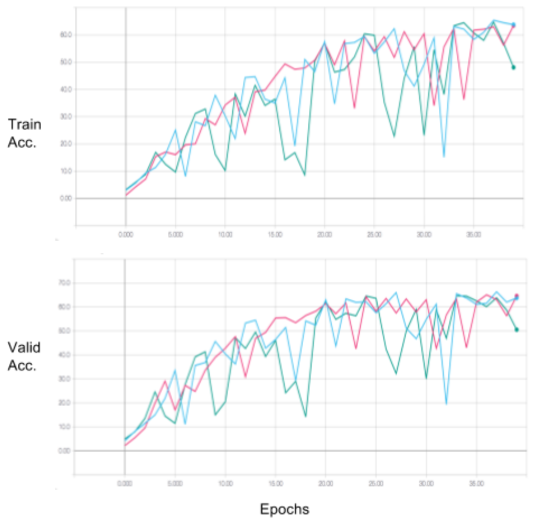


Seed	Color	Test Acc.
123	🟧	89.10
234	🟦	88.32
345	🟥	88.27

Figure 3. DAN-noise training on Cifar with 3 random seeds

4.4. Reducing the capacity of the CNN

Given that the noise base networks are working, we now see whether the performance is something meaningful. If a smaller network trained from noisy initialization (having same capacity as DAN Controller) can perform comparably to the DAN-noise models, then it means that the capacity of the network required for these datasets is much lesser than the original VGG-B model. However, if DAN-noise performs better for the same number of parameters, this style of architecture may have some interesting properties. We train a small CNN with 5 convolutional layers and 1 FC layer which has 1.7M feature parameters comparable to the 1.2M feature parameters in DAN-noise. We call this VGG-small. It is trained for 60 epochs with an initial learning rate



Seed	Color	Test Acc.
123	🟦	65.10
234	🟪	64.90
345	🟩	63.72

Figure 4. DAN-noise training on Sketches with 3 random seeds

of 3e-4 which is scaled down by 2 every 15 epochs. The results are shown in Table 3. DAN-noise significantly outperforms VGG-small on all the datasets. While we do note that our VGG-small architecture may not be optimized for this task and using different architectures may obtain better performance for the same number of parameters, the results are interesting nevertheless.

Dataset	VGG-small	DAN-noise
C-10	78.79	**90.57**
GTSR	86.43	**94.71**
SVHN	90.18	**95.32**
Caltech	37.25	**48.35**
Dped	89.14	**92.40**
Sketches	54.5	**70.10**

Table 3. Low capacity CNN vs DAN-noise

4.5. Filter Visualization

We also visualize the filters learned in Figure 5.One interesting thing to be noted is that some of the Imagenet pretrained filters are nearly zero (very low values of weights). Since these are 3x3 filters, we are unable to observe any significant patterns in the weights. The weights from all the models look equally random.

5. Conclusion

We were able to reproduce / beat the results for several settings. However, our performance is lesser for some settings on GTSR, Dped and Caltech. In particular, our numbers are very different from the numbers reported in the paper on Caltech. The exact splits of the Caltech dataset used is needed to reproduce the results. It would also help to know the hyperparameters which were not specified (batch size, initialization, weight decay and learning rate scaling across layers).

We also obtained significantly better performance on the DAN-noise setting. We verified this performance by visualizing the filters and training with multiple random seeds. This performance is also much better than a smaller model with capacity similar to the trainable weights of DAN-noise models. This could be explored further to understand the exact reasons for this gain.

References

[1] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):44:1–44:10, 2012.

[2] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[3] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

[4] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[5] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[6] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1863–1868, 2006.

[7] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning.

[8] A. Rosenfeld and J. K. Tsotsos. Incremental learning through deep adaptation. *CoRR*, abs/1705.04228, 2017.

Model	Settings	C-10	GTSR	SVHN	Caltech	Dped	Sketches
VGG-B(S)	[8]	92.5	98.2	96.2	88.2	92.9	69.2
	Ours v1	89.81	95.22	95.06	42.12	92.09	64.85
	Ours v2	92.63	95.66	95.32	49.02	91.80	67.55
VGG-B(P)	[8]	93.2	99.0	95.8	92.6	98.7	65.4
	Ours v1	94.57	96.16	96.02	64.02	94.84	72.72
	Ours v2	95.05	95.08	95.71	66.14	97.52	71.90
DAN$_{sketches}$	[8]	77.9	93.3	93.2	86.9	94.0	69.2
	Ours v1	80.99	93.99	92.12	28.20	92.23	64.85
	Ours v2	86.74	94.19	94.34	37.20	89.21	67.55
DAN$_{noise}$	[8]	68.1	90.9	90.4	84.6	91.3	42.7
	Ours v1	87.64	94.05	94.7	34.73	88.29	61.38
	Ours v2	90.57	94.71	95.32	48.35	92.40	70.10
DAN$_{imagenet}$	[8]	91.6	97.6	94.6	92.2	98.7	63.2
	Ours v1	91.14	94.58	94.48	51.09	92.77	66.35
	Ours v2	89.27	95.14	95.30	45.76	90.88	66.07
DAN$_{imagenet+sketches}$	[8]	91.6	97.6	94.6	92.2	98.7	69.2
	Ours v1	91.14	94.58	94.48	51.09	92.77	64.85
	Ours v2	89.27	95.14	95.30	45.76	90.88	67.55

Table 4. DAN Results on Image Classification. The numbers are highlighted in red, blue and green to indicate whether our performance is worse, comparable or better than the reported results respectively.

Net	C-10		GTSR		SVHN		Caltech		Dped		Sketches	
	1 FC	3 FC	1 FC	3 FC	1 FC	3 FC	1 FC	3 FC	1 FC	3 FC	1 FC	3 FC
VGG-B(S)	**92.63**	90.21	95.66	**96.55**	**95.32**	94.92	49.02	**52.52**	91.80	**93.44**	67.55	**68.85**
VGG-B(P)	**95.05**	93.02	95.08	**95.59**	95.71	**95.72**	**66.14**	65.16	**97.52**	94.80	**71.90**	63.17
DAN$_{sketches}$	**86.74**	84.00	**94.19**	93.77	94.34	94.34	**37.20**	29.14	89.21	**90.63**	67.55	**68.85**
DAN$_{imagenet}$	**89.27**	89.17	**95.14**	94.25	**95.30**	94.98	**45.76**	39.14	90.88	**92.38**	**66.07**	61.13
DAN$_{noise}$	**90.57**	90.36	94.71	**96.02**	95.32	**95.35**	**48.35**	46.29	**92.40**	91.50	**70.10**	64.02

Table 5. Results for models with 1 and 3 fully connected (FC) layers

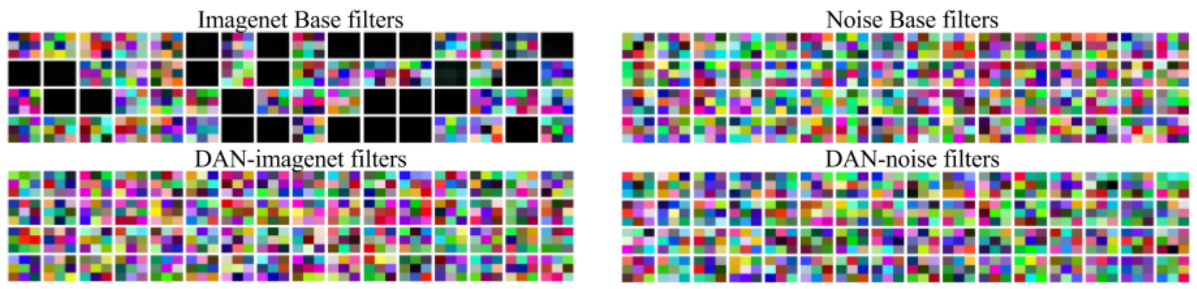


Figure 5. Conv0 filters from Imagenet base, DAN-imagenet, Noise base and DAN-noise