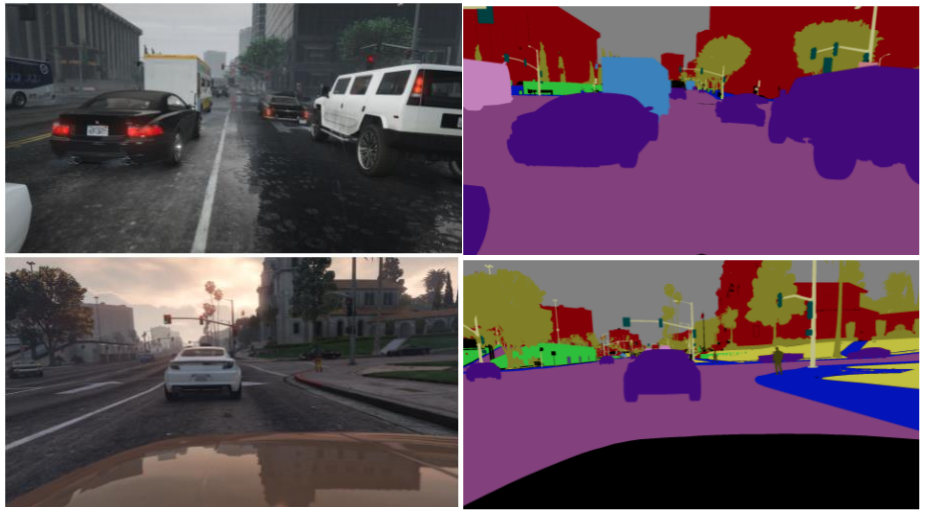
SWADES:
Unsupervised Source Free Domain Adaptation

By Aditya, Akash, Ashlyn and Shreyas

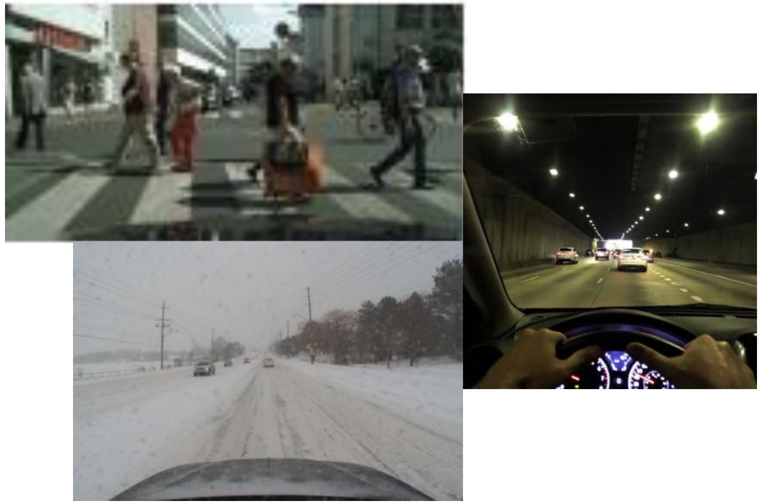
Problem Statement

- **Problem Statement**
- Related Work
- Proposed Approach
- Experiments and Results
- Ablation and Analysis
- Conclusion

Problem statement (A)



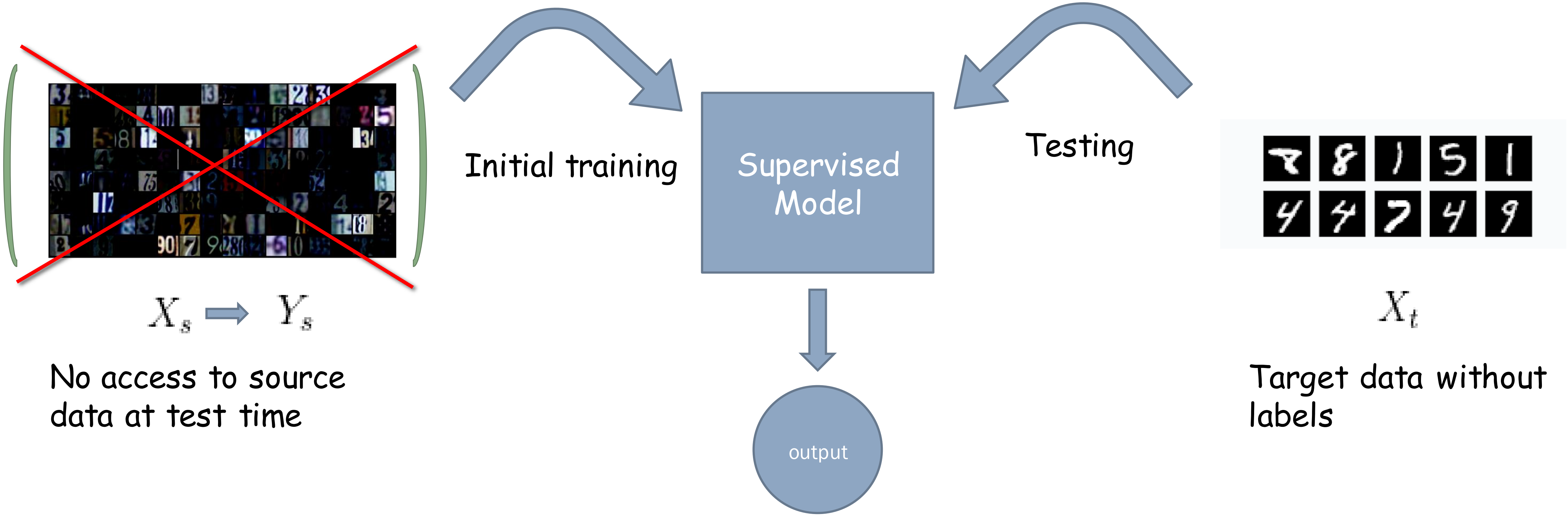
Source dataset (GTA)
Images and GT



Target dataset
Images in the wild

Richter et al., Playing for Data: Ground Truth from Computer Games

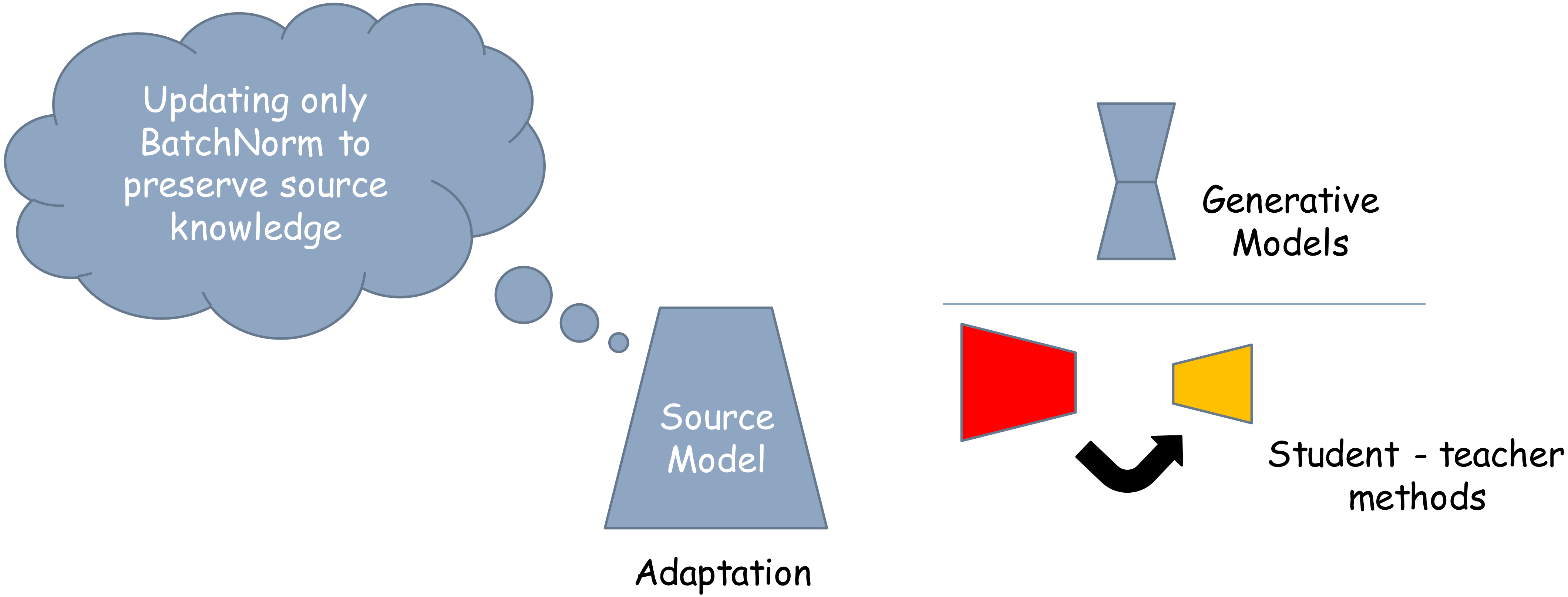
Problem statement (B)

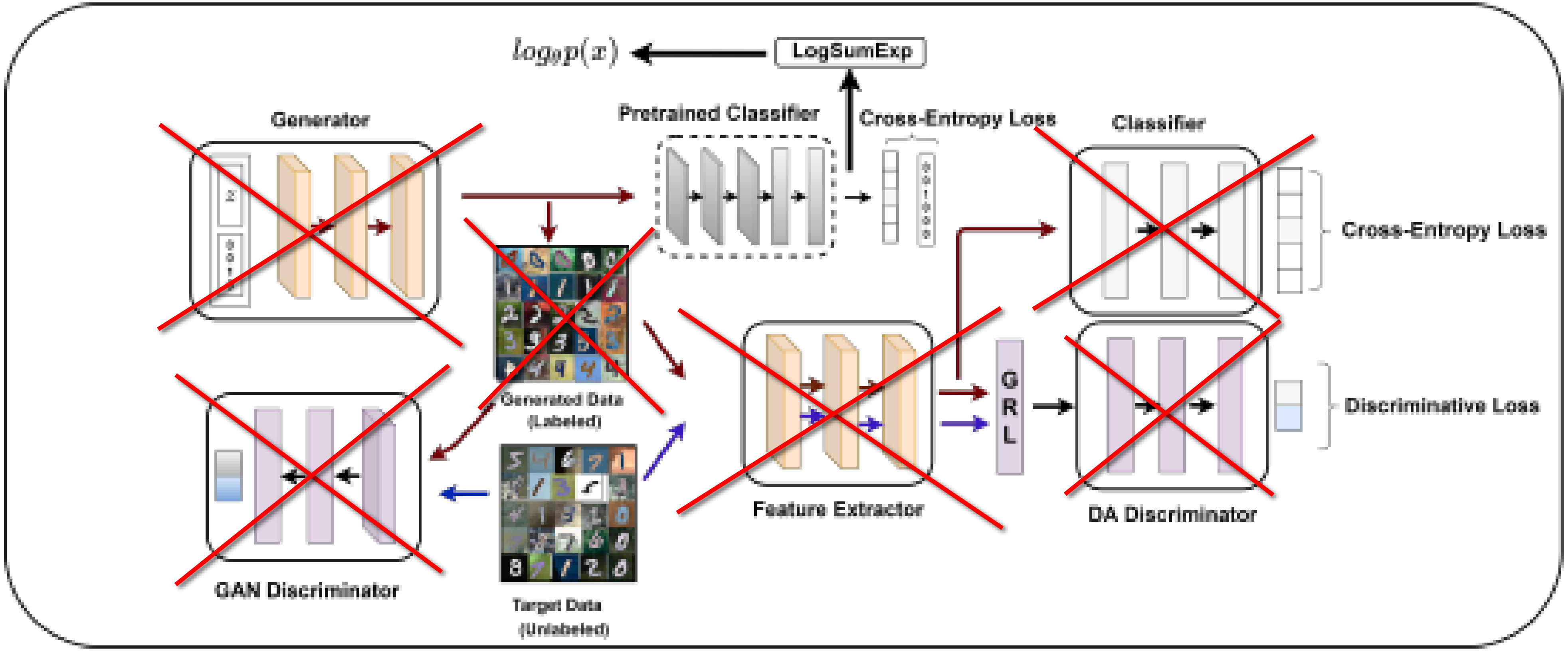


Related Work

- Problem Statement
- **Related Work**
- Proposed Approach
- Experiments and Results
- Ablation and Analysis
- Conclusion

Trade-off in existing methods

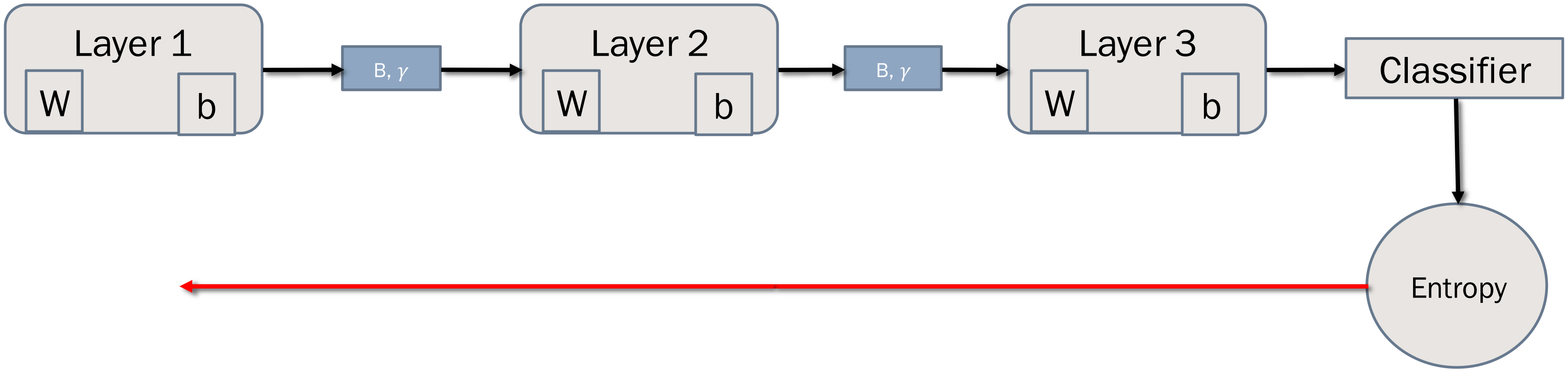




Improve test time adaptation performance without adding additional architectures.

Kurmi et al., Domain Impression: A Source Data Free Domain Adaptation Method

Related Work (A) TENT

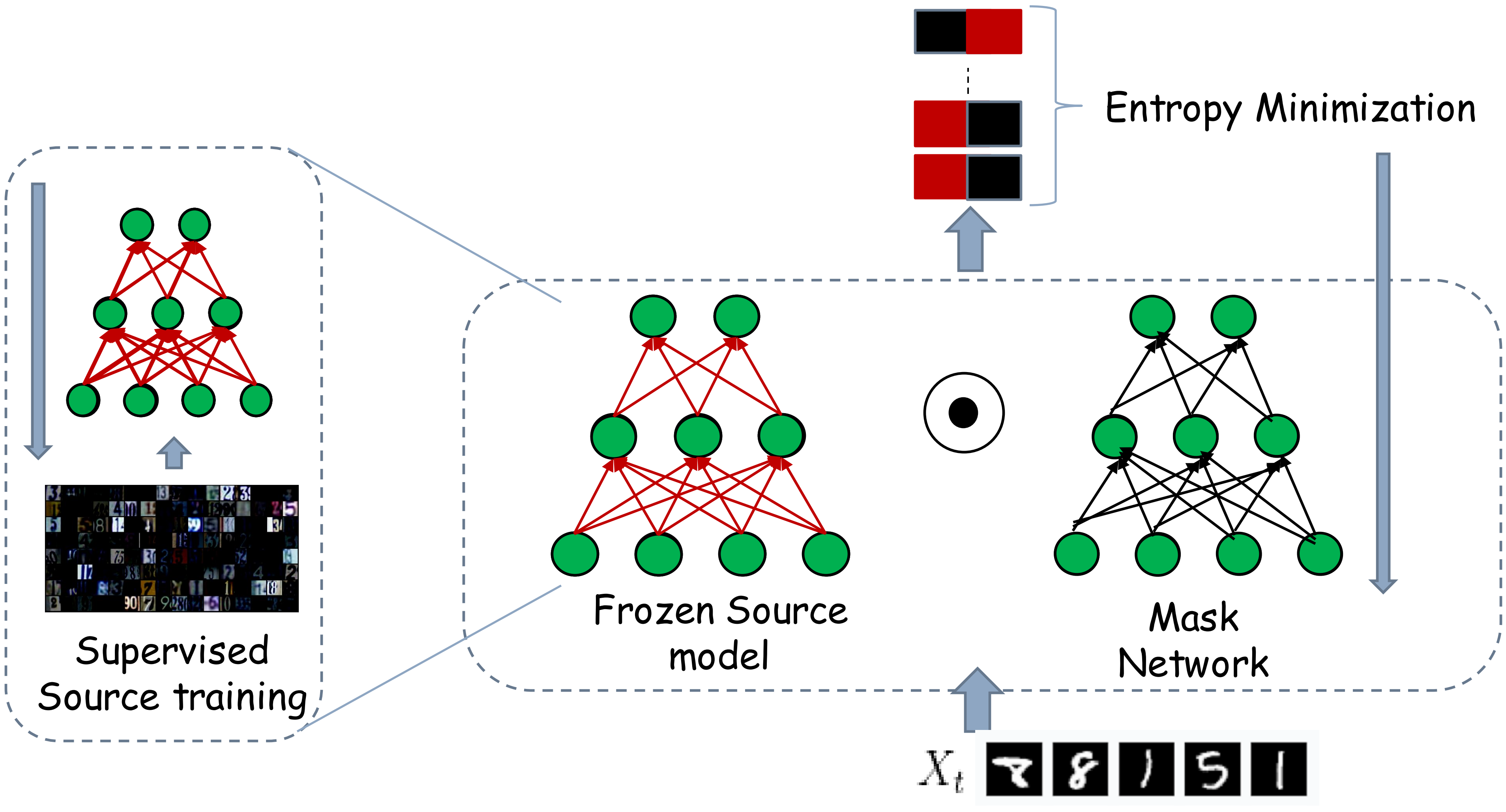


Wang et al., Tent: Fully Test-time Adaptation by Entropy Minimization

Proposed Approach

- Problem Statement
- Related Work
- **Proposed Approach**
- Experiments and Results
- Ablation and Analysis
- Conclusion

Proposed Approach (A and B)



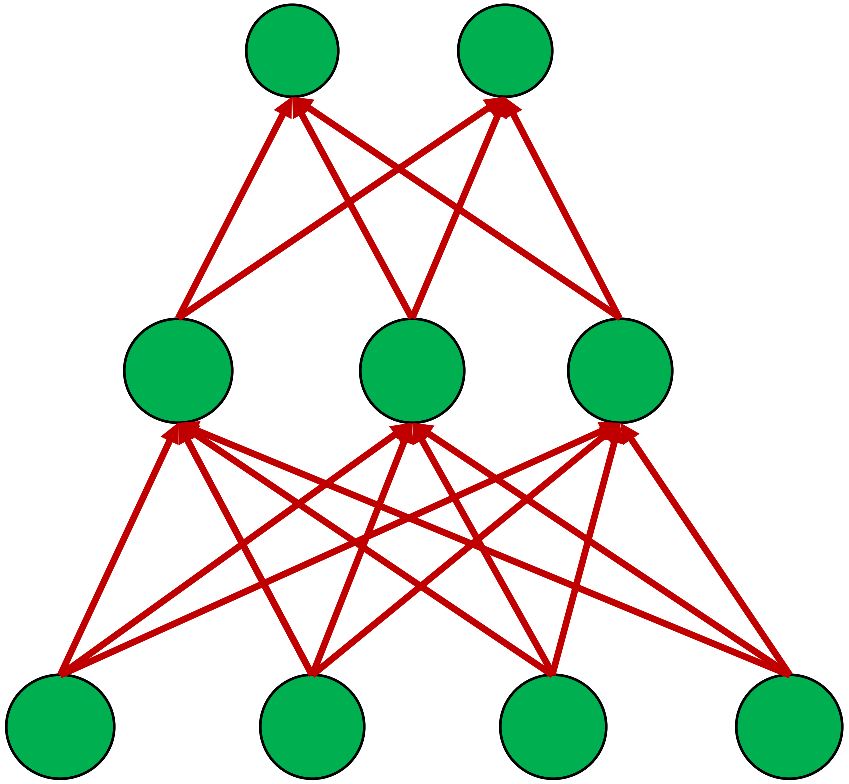
Contributions and Re-use

Contributions*

- Novel method to add mask on Source network for Domain Adaptation.
- Explore the intersection of unsupervised domain adaptation and sparse neural networks

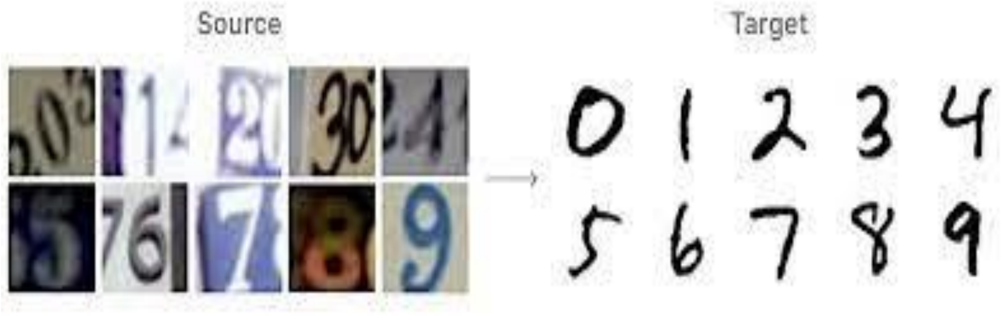
Re-use from prior art

- Test time adaptation via unsupervised loss



Experiments And Results

Experiment Setup



SVHN to MNIST

Backbones
- VGG 11, VGG 16, VGG 19

Setup
- Developed our own codebase (PyTorch)
- Chose best hyperparameters for TENT and Swades
- Used W&B for experiment tracking

Experimental Setup (Baselines)

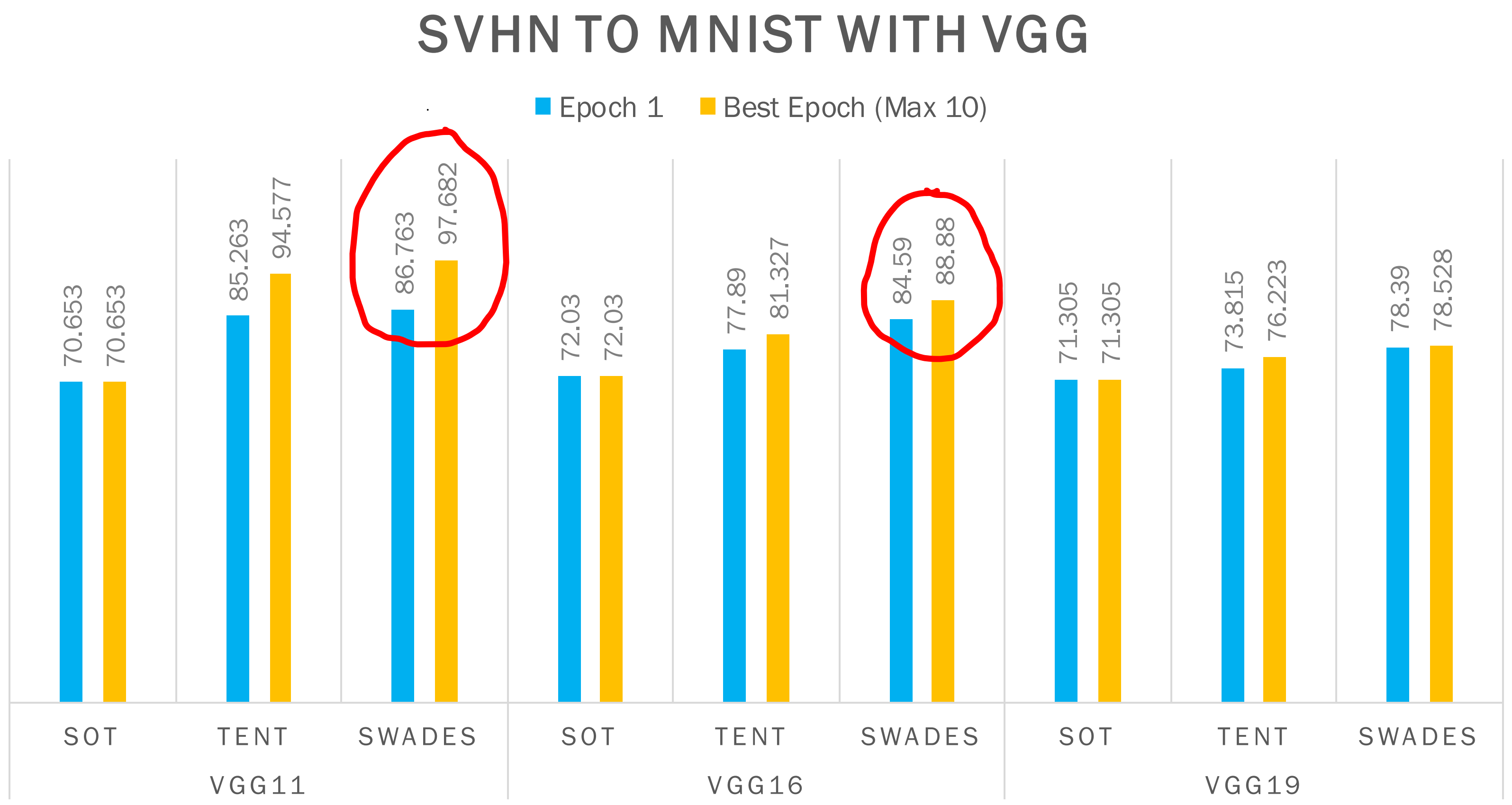
Source on Target

1. Performance of the pre-trained source network on target domain
2. This is our performance lower bound

Test-time entropy minimization

1. Method explained in Wang et al.,
2. We implemented TENT with VGG backbones and verified our implementation with the official codebase

Accuracy on target domain

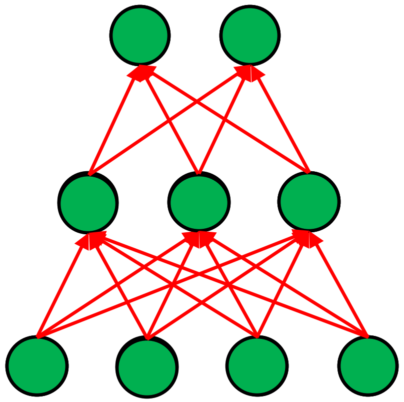


Ablation and Analysis

What if we binarize the masks ?

Can we find a sparse sub-network within source trained network, that already performs well on target???

And that too without using labels.



Binary mask initialization and training

Given source trained network and sparsity, p = 90%

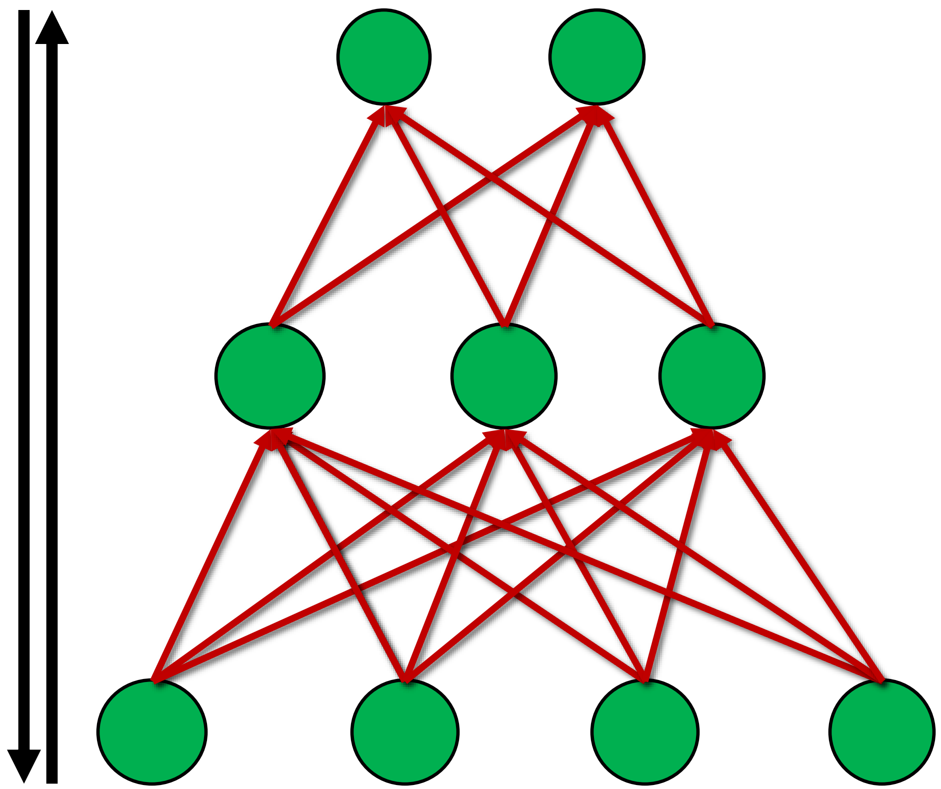
Initialize Scores : scores[i] = weight_value[i]

Binarize Masks : mask[i] = 1 if scores[i].abs() in p percentile

Update scores and mask :

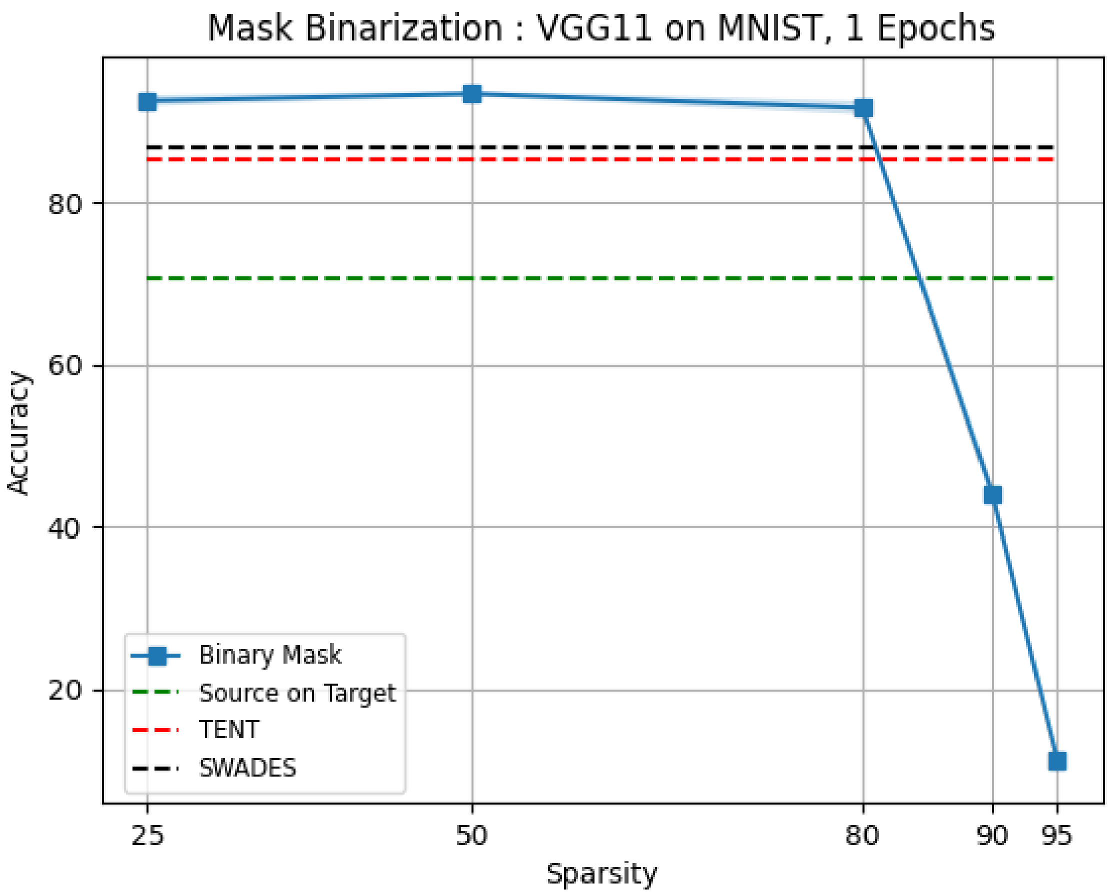
For each batch :

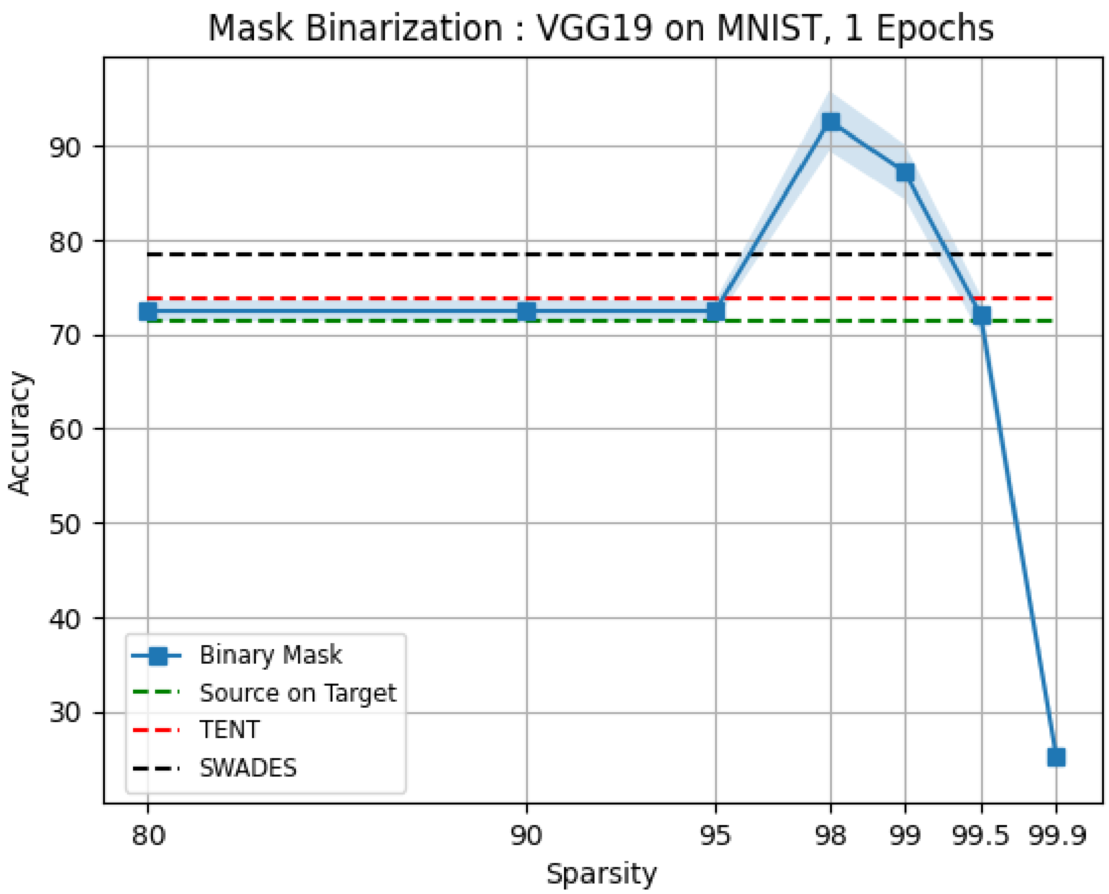
- Sparse Forward -> Compute Entropy
- Dense Backward -> Compute gradient
- scores[i] = scores[i] – learning_rate x gradient[i]
- scores -> Binarize Masks



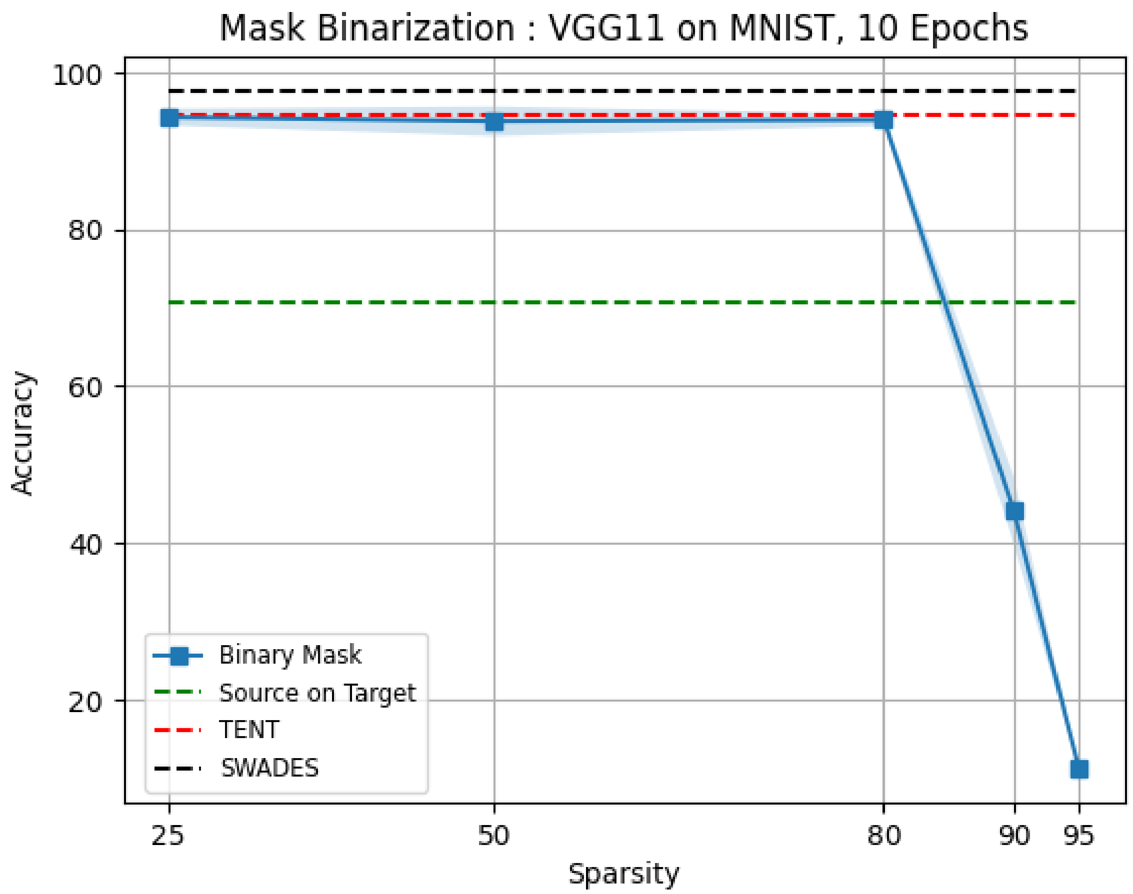
Ramanujan et. al. CVPR 2020

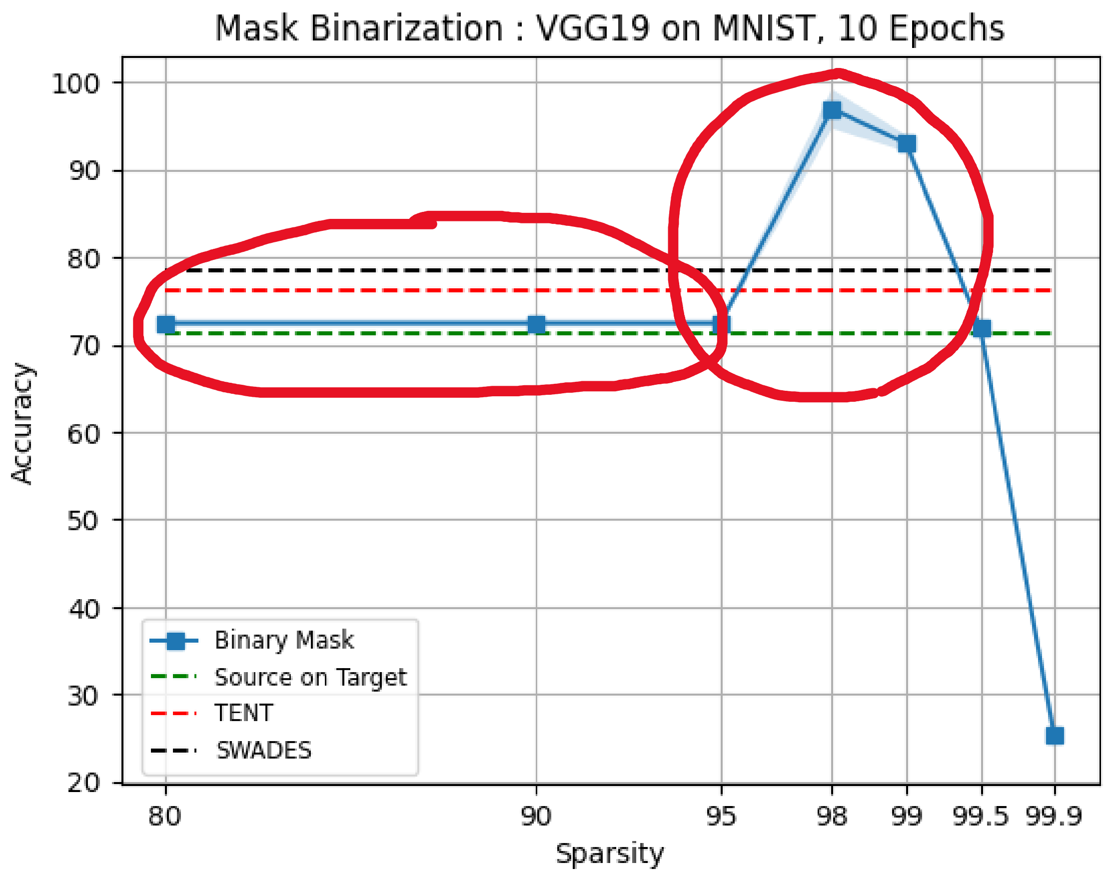
Binary masks found, achieve better test-time adaptation (1 epoch) accuracy than TENT and SWADES



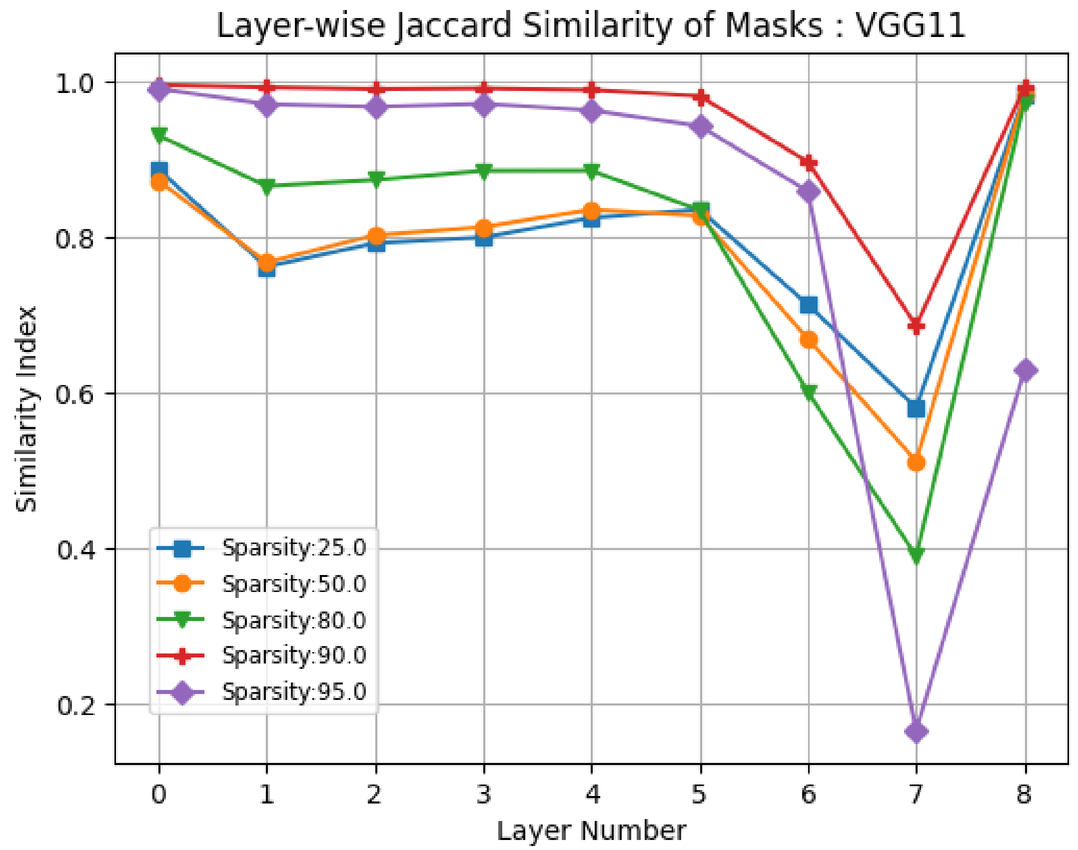


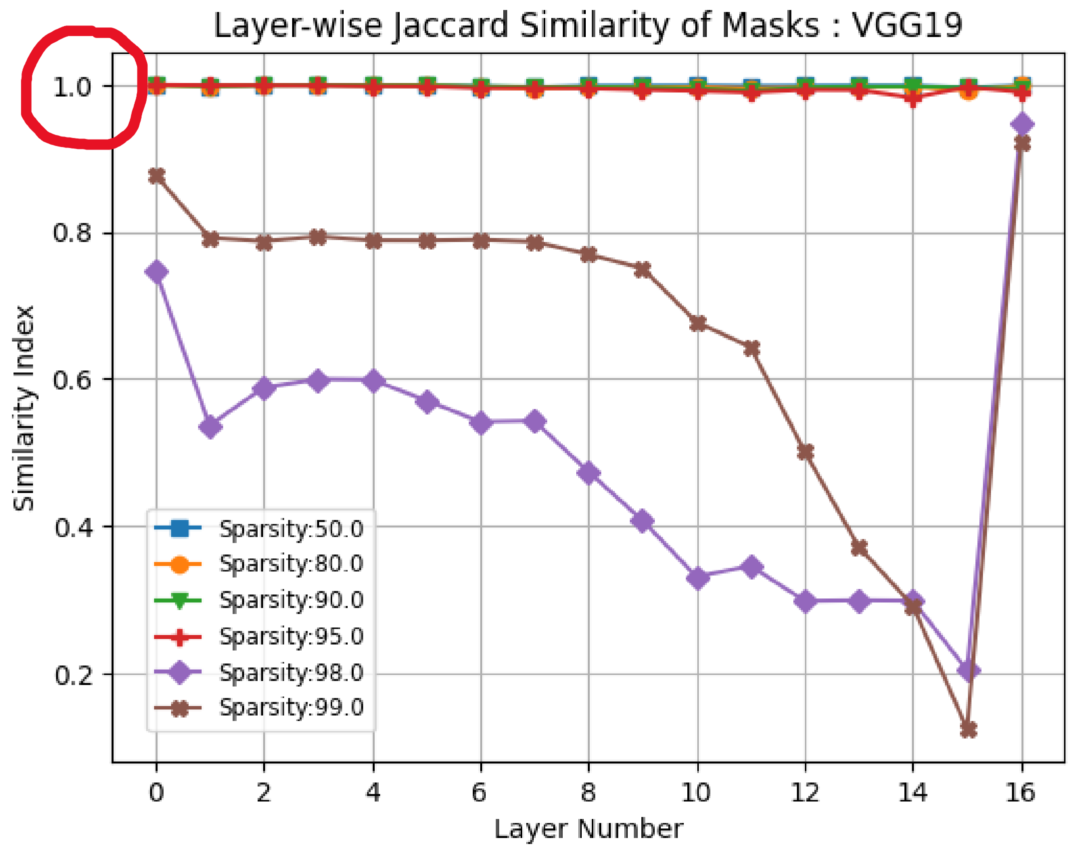
Binary masks trained for 10 epochs comparison with TENT and SWADES





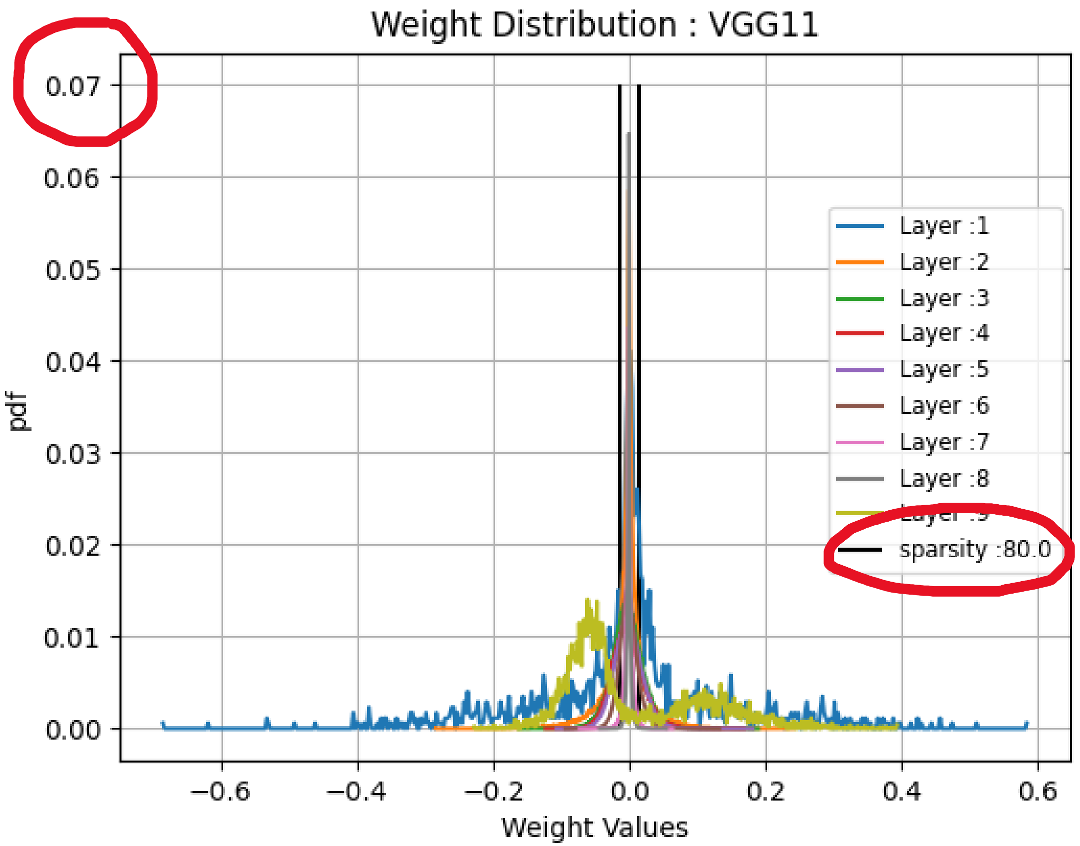
At lower sparsity levels initialized masks do not get updated for VGG19

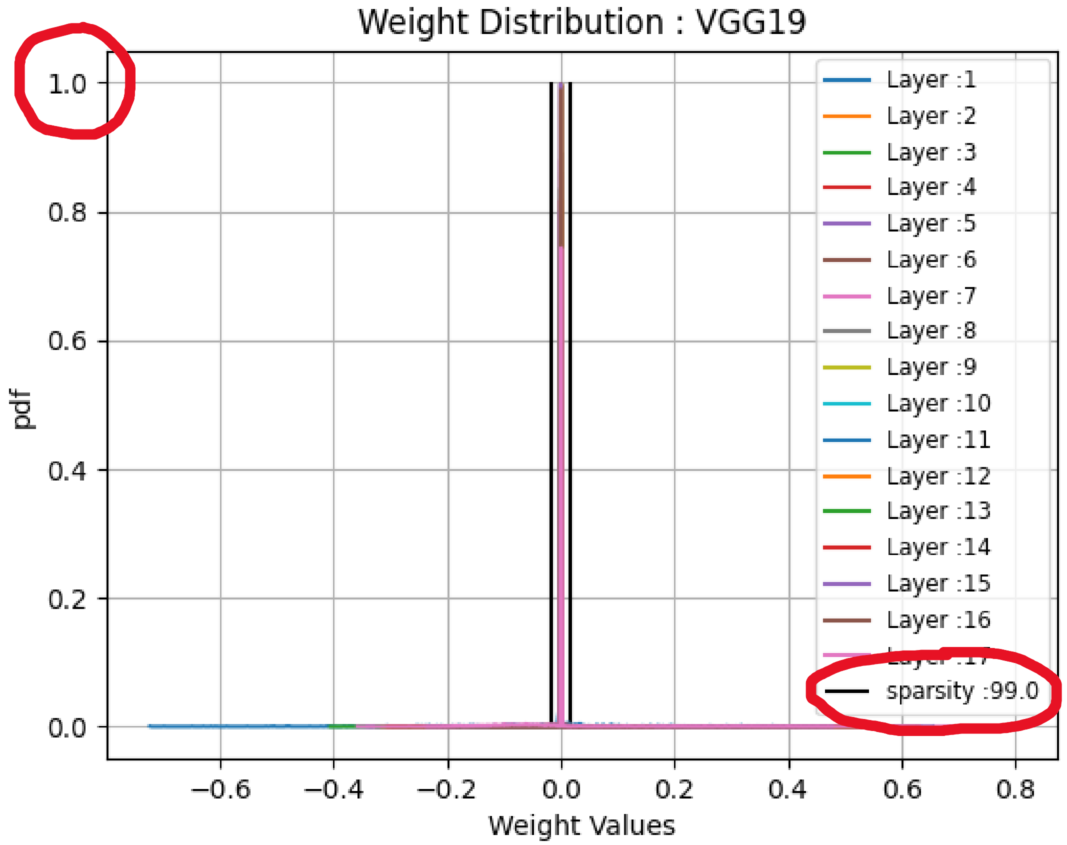




Why don't the masks get updated for VGG19 at lower sparsity levels as in VGG11 ??

The difference lies in the weight distribution of pre-trained models





Only 1-2% of the weights define the network function in VGG19

Let us assume that the sparsity level is low, ~50%

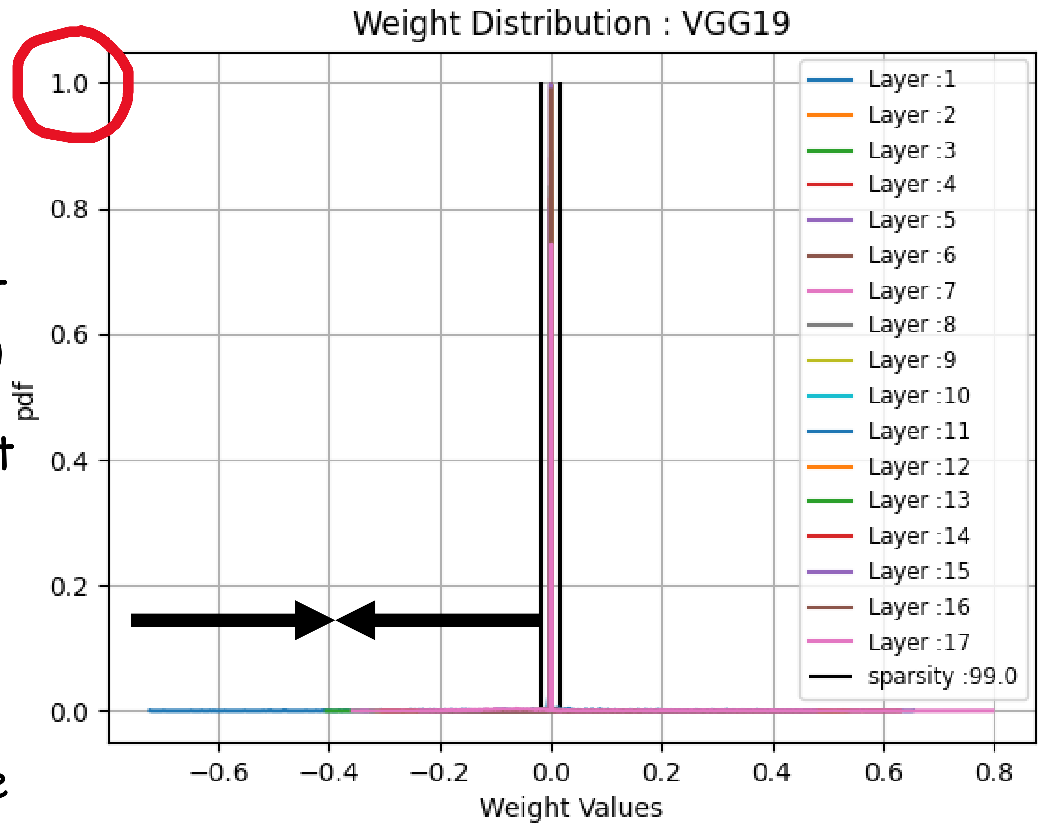
- (> 98%) of weights are ~ 0,
- mask initialized by thresholding weight-magnitude

Sparse networks at initialization are functionally equivalent to the dense network. (all non-zero weights are in subnet)

*** To update the network function, non-zero weights must be removed.

*** Initial non-zero scores/weights are so few and apart that training scores may not bridge the gap enough.

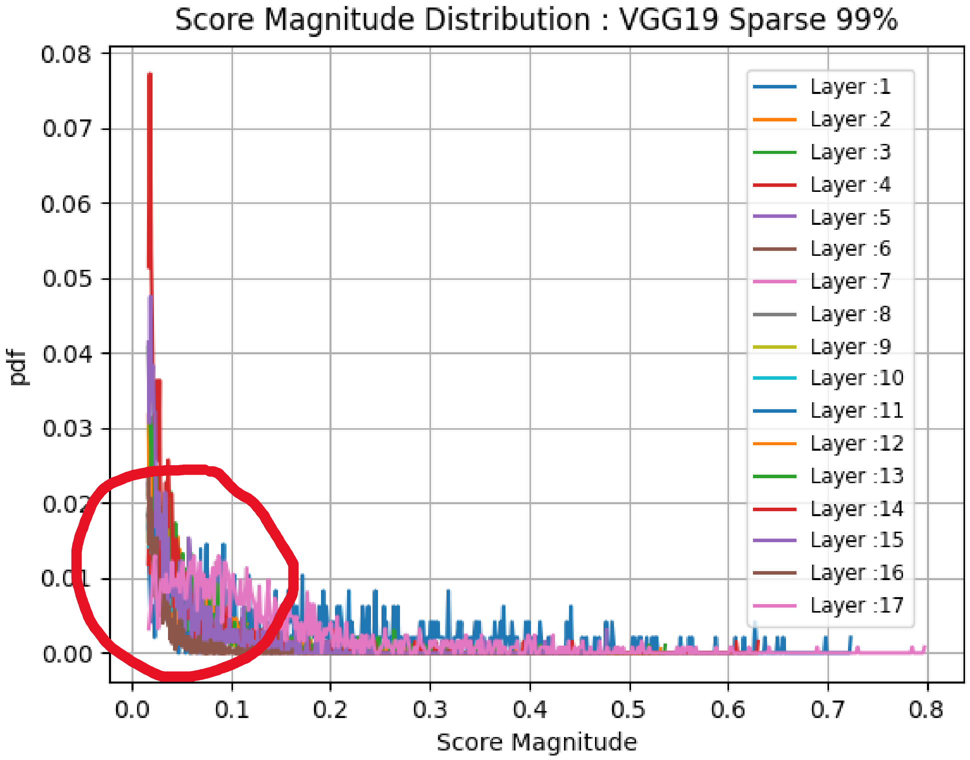
Therefore, no change in masks and network function, hence accuracy same as dense source network on target.



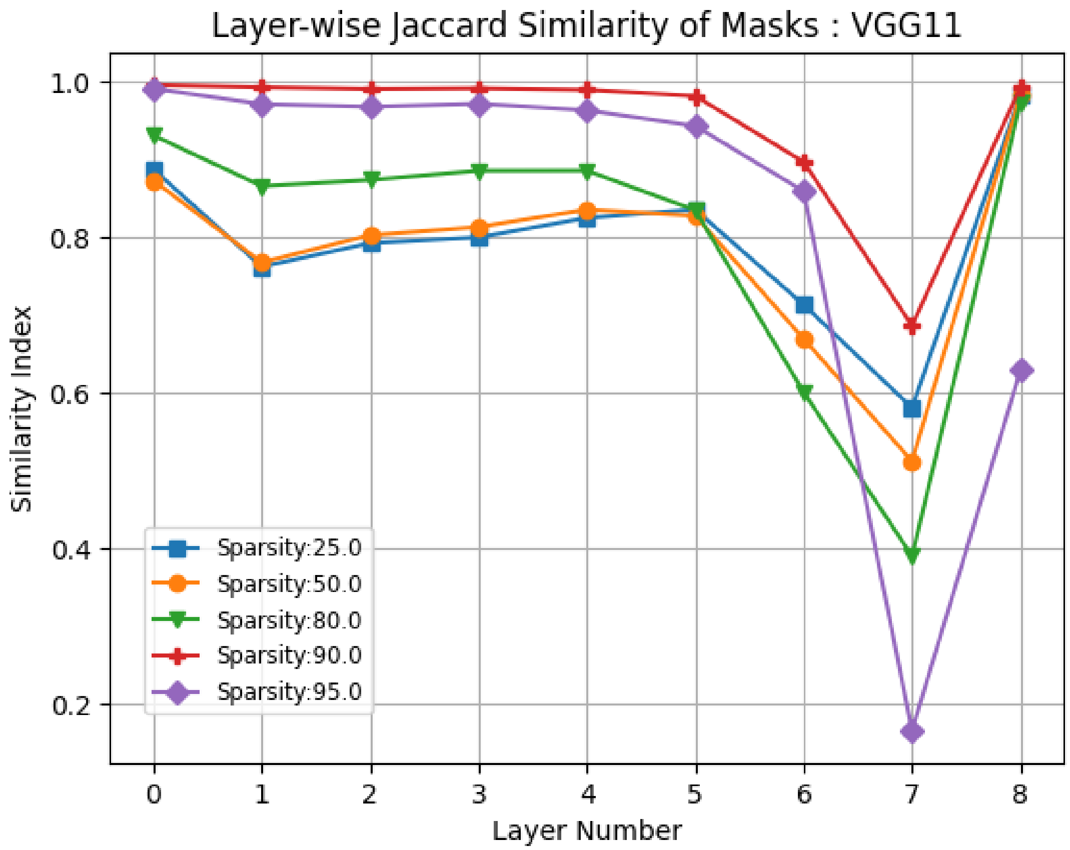
Sparsity threshold hypothesis for over-parametrized networks

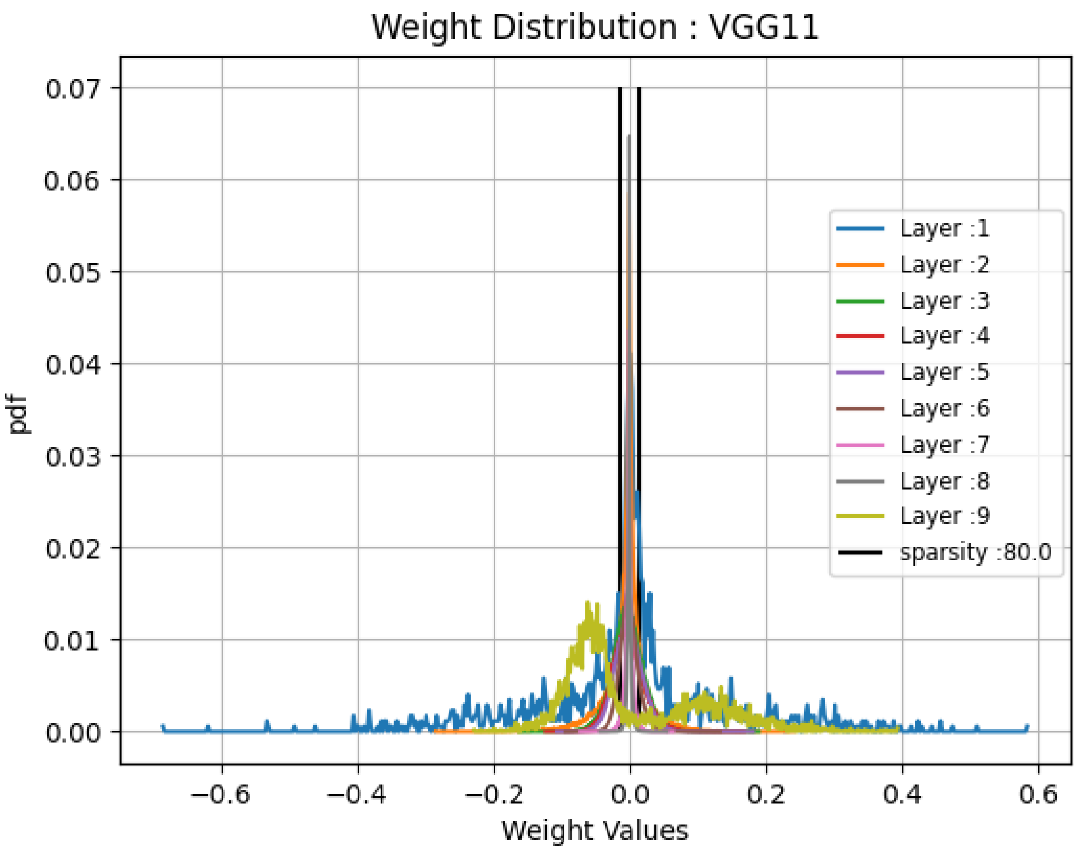
Once the sparsity and corresponding threshold are high enough :

- ❑ High magnitude non-zero weights can move freely in and out of the sub-network changing the network function.



In VGG11 a lot more initial scores lie close to the threshold, so they can easily cross the barrier





Conclusion

Synthesis of results,
what can we claim ??

1. Adding more updatable parameters by adding masks improves adaptation performance (TENT vs Swades) [Ref slide: 15]
    1. without terribly increasing the architecture complexity
    2. the compute requirements
2. There do exist a series of subnets within the source network [Ref: slide 19, 20]
    1. that can be found using an unsupervised loss
    2. achieve high target performance

Future work

1. Validate the sparsity threshold hypothesis
2. Experimentation on more complex datasets (CIFAR10-C, ImageNet-C, VisDA)
3. Experiments on FLOP efficient networks (MobileNet, EfficientNet)
4. Layer-wise ablation and masking

Challenges

1. Access to certain network architectures to validate TENT implementation
2. Limited compute resources
3. Consolidating and prioritizing experiments (ideas flowing in all directions, but not enough resources or time)

Thank you

Code will be available (mid December) : https://github.com/ShreyasMalakarjunPatil/Swades-SFDA
Results will be available here : https://wandb.ai/swades

Why Comic Sans?

https://www.bdadyslexia.org.uk/advice/employers/creating-a-dyslexia-friendly-workplace/dyslexia-friendly-style-guide

Appendix

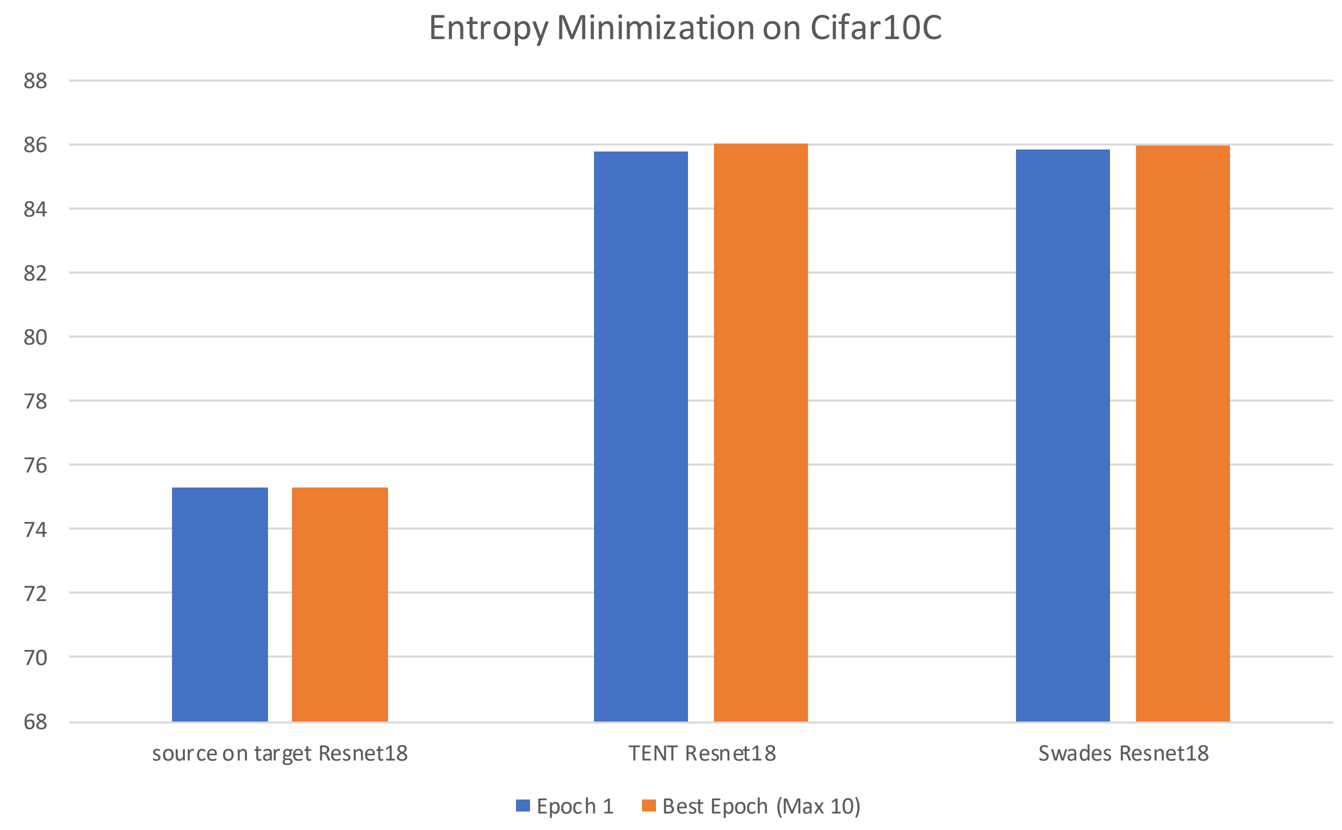
Optimal Sparsity Hypothesis for highly over-parametrized networks and SWADES-Binarization

Given a highly over-parametrized network (ex. Vgg19) trained on source dataset (ex. SVHN), target dataset (ex. MNIST) and consider the binary masks based adapting algorithm,

There exists an optimal sparsity level, S_optimal such that,

1. If Sparsity < S_optimal, adaptation accuracy = source on target accuracy, due to stark difference in weight values and inability of training process to bridge the gap.
2. If Sparsity = S_optimal, adaptation accuracy -> Best, providing enough (> 0) weight values to adapt network function.
3. If Sparsity > S_optimal, adaptation accuracy < Best, due to unavailability of weights to learn the required function.

Appendix 1: Results Cifar10 → Cifar10c



Note : Due to computational resource constraints we were unable to optimize the hyper-parameters, therefore these results may not be indicative of improvements.