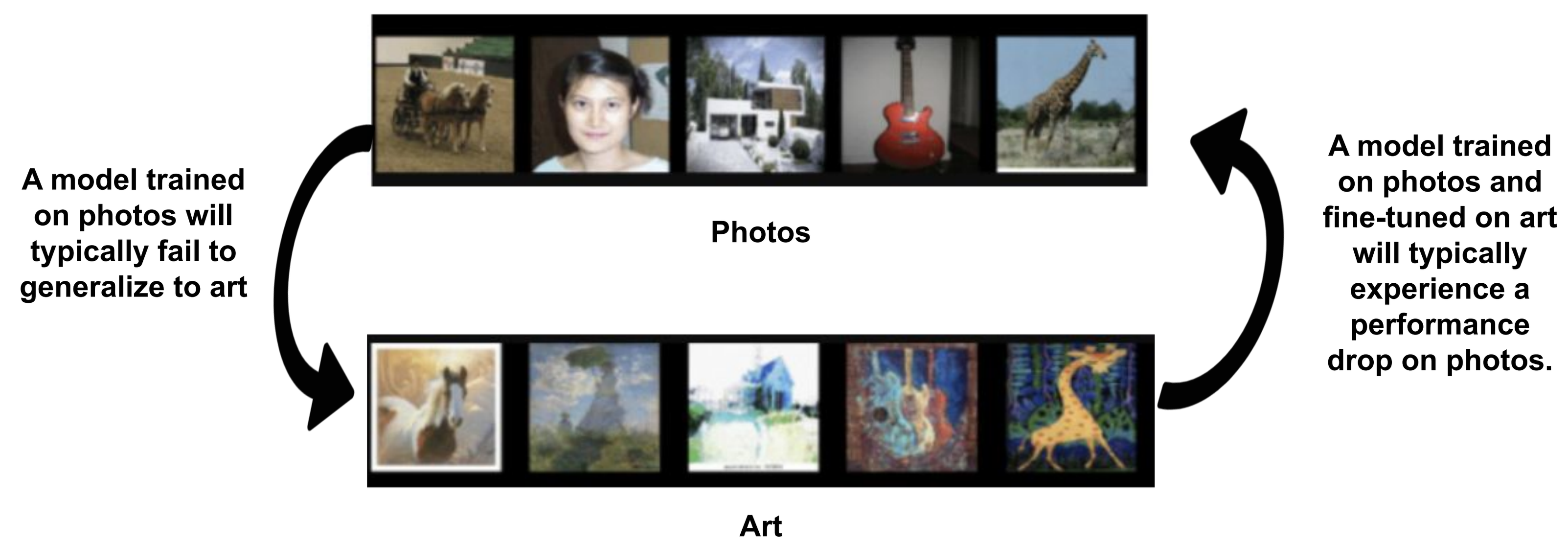
Differentiable Weight Masks for Domain Transfer

Akash Velu, Samar Khanna, Skanda Vaidyanath
{avelu, samar99, svaidyan} [at] stanford.edu

Introduction

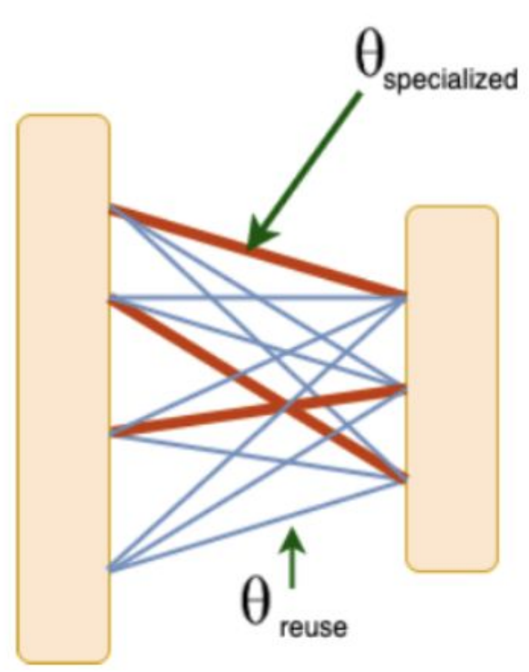
One of the major drawbacks of deep learning models has been their lack of ability to generalize to new, but similar domains while maintaining their performance on the original domain they were trained on.



Fine-tuning is a popular approach to transferring to a new task. However, this usually results in a performance drop in the source task.

Motivation + Hypothesis

- The key idea is to modularize the weights of a network into those **specialized** for the source task and those we can **reuse** and fine-tune for the target task.
- If we only finetune these weights and freeze the specialized weights, the hypothesis is that we will not lose too much performance on the source dataset.
- We provide three methods to learn a binary mask that indicates whether a weight should be frozen or not when fine-tuning on the target domain.

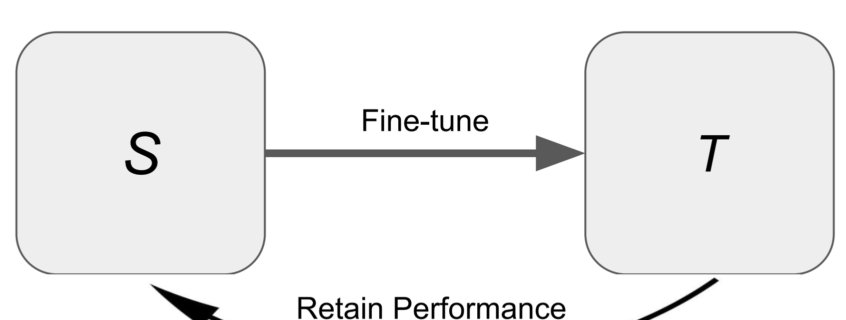


Problem Statement + Dataset

Problem Statement:

A model f_θ is trained on a **source domain** *S*. There is a **target domain** *T* to which we would like to generalize.

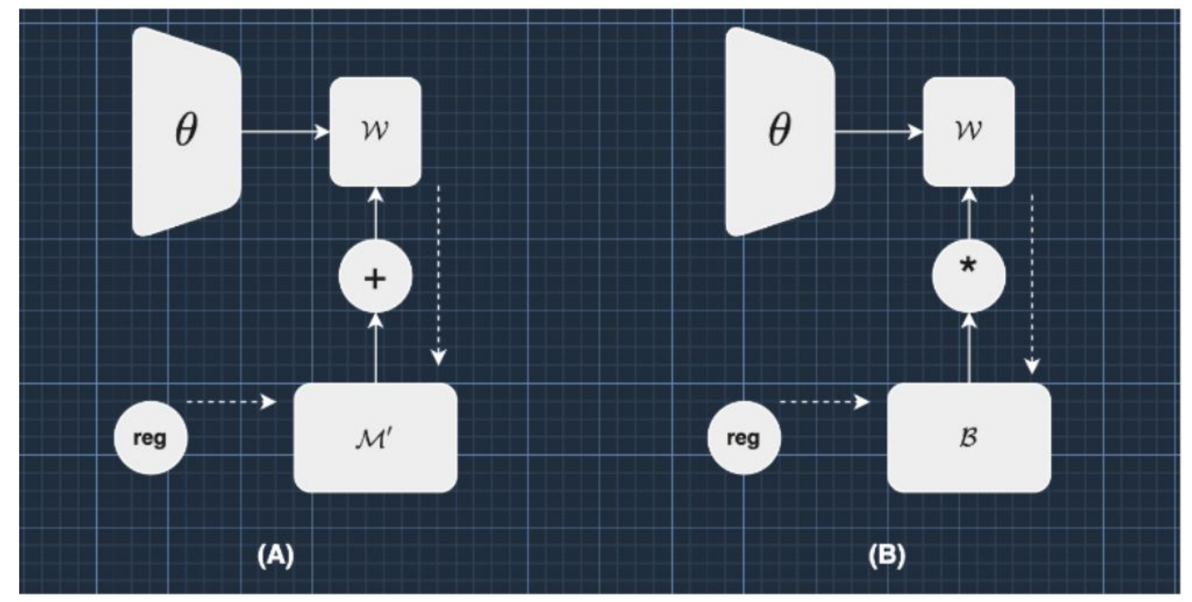
Goal: generalize to target domain *T* **while retaining strong performance on *S*.**



Dataset:

We utilize the **PACS** dataset. PACS consists of 4 domains: **photos, art, cartoon, and sketch**. Each domain consists of the 7 classification classes.

Methodology



Naive Masking

- We identified that the weights after training on *S* are distributed normally and hence re-initialized the weights that were one within one standard deviation from the mean.
- The idea is that extremes contribute more to the output than the non-extremes.

Edit Masking (A)

- Learn a real-valued edit for each weight after training on *S* such that the cross-entropy loss on *S* does not decrease too much.
- Regularizer: "edit as many weights without affecting performance on *S*"
- The actual weights *W* are frozen during mask training.
- If a weight is edited more than a certain threshold after mask training, we hypothesise that it is not specialized to *S* and re-initialize it.

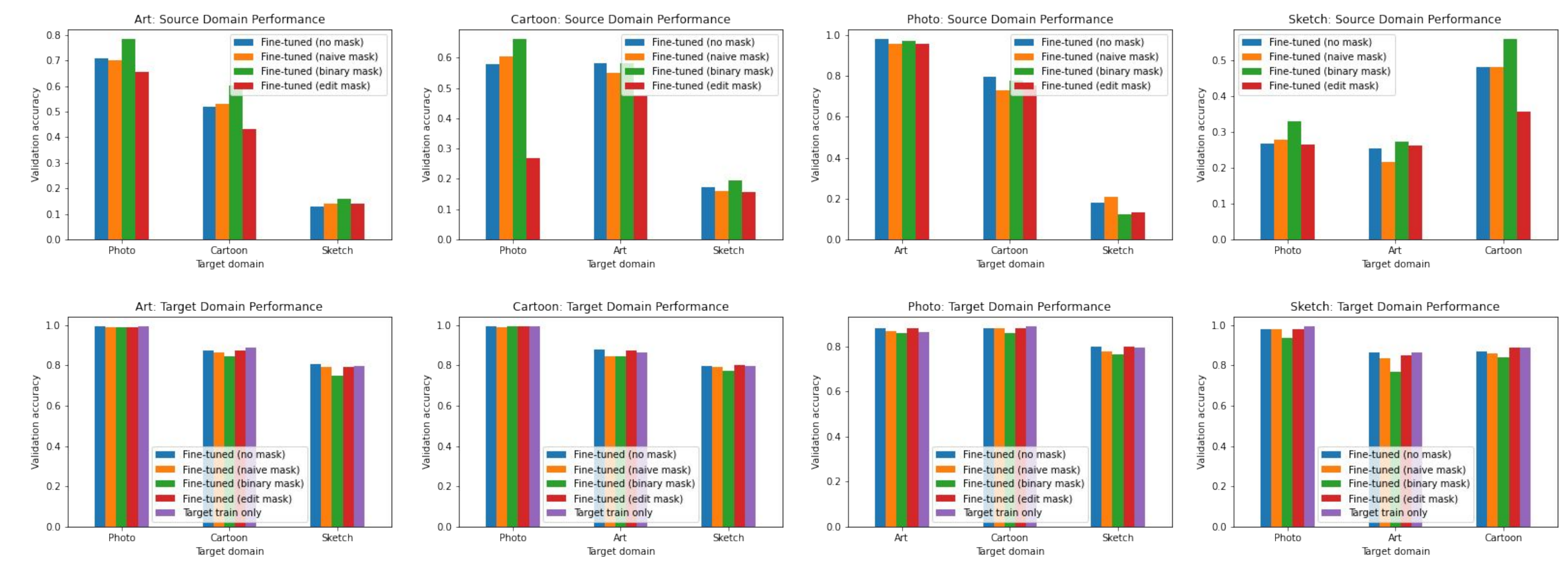
Binary Masking (B)

- Instead of learning a real-valued mask and thresholding it to get a binary mask, learn a binary mask directly.
- During mask training, sample binary masks from the real-valued logits using Gumbel-Sigmoid and a straight-through estimator so it is end-to-end differentiable.
- Once we sample a mask, multiply it with the frozen *W* to mask out some weights and learn a mask that doesn't reduce cross entropy loss too much and the regularizer encourages masking as many weights as possible.
- After mask training, re-initialize weights with negative logits and freeze the other weights.

Future Work

- **Information-theoretic masking strategies:** identify weights that give maximum information about logits; mask weights such that distance between output probability distributions before and after masking are similar
- **Linear algebraic masking strategies:** make edits on the low-rank approximation of the weight matrix preserving the eigenvectors corresponding to larger eigenvalues.
- Extending to multi-task and continual learning settings.

Experimental Results.



Discussion and Analysis

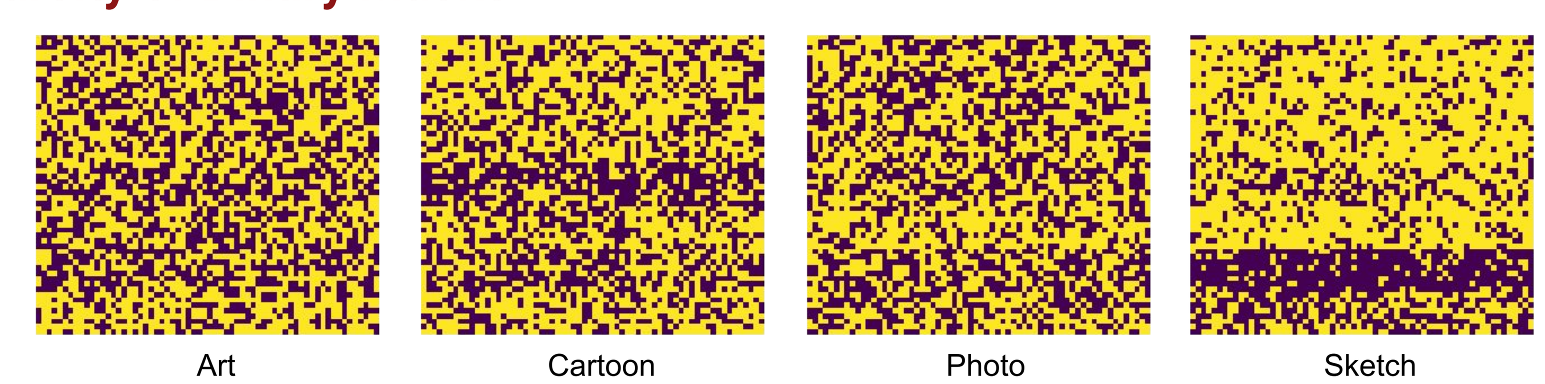
Source Domain Performance

- Binary masking consistently yields the strongest performance when the fine-tuned model is evaluated in the source domain.
- Edit and naive masking surpass the no-masking baseline at times.
- **Hypothesis**: edit and naive masking are sensitive to the threshold at which to binarize; binary masking provides a more natural way to obtain a binary mask.

Target Domain Performance

- Masking methods typically achieve **very similar results** in the target domain.
- This is **despite freezing most weights** before fine-tuning in the target domain.
- Edit masking slightly outperforms binary masking in several scenarios.

Sparsity of Binary Masks



How should θ_{reuse} be initialized?

Domain	Photo		Art		Cartoon		Sketch	
Method	$\mathcal{S}$ Gain	$\mathcal{T}$ Drop	$\mathcal{S}$ Gain	$\mathcal{T}$ Drop	$\mathcal{S}$ Gain	$\mathcal{T}$ Drop	$\mathcal{S}$ Gain	$\mathcal{T}$ Drop
$\mathcal{S}$ Start	-0.029	**-0.025**	0.062	**-0.030**	**0.035**	-0.019	**0.054**	**-0.056**
$\mathcal{S}$ End	**-0.014**	-0.027	0.051	-0.038	0.021	**-0.016**	0.043	-0.059
Random	**-0.014**	-0.027	**0.064**	-0.036	0.001	-0.021	0.023	-0.058

- We explore 3 methods for initializing θ_{reuse}: values *before* starting source domain training (S_{start}), values *after* source domain training (S_{end}), and random.
- Values are similar across strategies, but S_{start} yields the best performance most often, similar to the findings in the Lottery Ticket Hypothesis.

Acknowledgements and References

We thank the CS231N course staff for providing guidance throughout this project.

R. Csordas, S. van Steenkiste, and J. Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In International Conference on Learning Representations, 2021
Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning. Fast model editing at scale. In International Conference on Learning Representations, 2022.
Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635, 2018.