# Differentiable Weight Masks for Domain Transfer

Samar Khanna* Skanda Vaidyanath* Akash Velu *

Stanford University

{samarkhanna,svaidyan,avelu}@cs.stanford.edu

## Abstract

*One of the major drawbacks of deep learning models for computer vision has been their inability to retain multiple sources of information in a modular fashion. For instance, given a network that has been trained on a source task, we would like to re-train this network on a similar, yet different, target task while maintaining its performance on the source task. Simultaneously, researchers have extensively studied modularization of network weights to localize and identify the set of weights culpable for eliciting the observed performance on a given task. One set of works studies the modularization induced in the weights of a neural network by learning and analysing weight masks. In this work, we combine these fields to study three such weight masking methods and analyse their ability to mitigate "forgetting" on the source task while also allowing for efficient finetuning on the target task. We find that different masking techniques trade-off retaining knowledge in the source task with learning to perform well on the target task.*

## 1. Introduction

Deep learning algorithms have proven to be extremely successful in a wide range of supervised learning tasks [2] [19] in vision and language. These methods are capable of utilizing large datasets to learn models which can generalize to new datapoints which are within the training data distribution. Although some particularly large models such as GPT-3 and DALLE-2 have exhibited emergent qualities of generalizing to *new* data distributions, smaller models in vision and language tend to suffer from *distribution shift*.

Often, when practitioners want to improve the performance of their model on a new domain that is out of distribution from the one they trained on, a natural idea is to fine-tune the weights of the model with data from this new domain. This often tends to improve the performance on the new (target) domain but tends to lose performance on the original (source) domain. However, we'd like models to be able to maintain their performance on the source domain while improving on the target domain. For instance, a model used in a self-driving car which is adapted to a new driving environment should retain strong performance in the original environment it was trained on, while achieving competitive performance in the new environment. Another example is when an image classifier trained on paintings is fine-tuned on a new domain of cartoon images, and it loses its performance on the original painting domain. This is related to the *catastrophic forgetting* problem in continual learning where a model trained on a new task tends to forget older tasks it learned.

In this work, we examine methods by which models can be adapted to a new domain *without* experiencing significant forgetting. One way to achieve this could be by modularizing the network and splitting the weights as being specific to the source domain and others that can be edited to improve performance on the target domain without a drop in the source domain. We take inspiration from methods that edit the weights of a trained model as a means of adapting its performance. Specifically, we focus on differentiable weight masks [4] and model editor networks literature such as [17], methods which either directly modify or mask the weight matrices of a trained neural network in order to analyze the network's properties or to edit its performance on specific datapoints. In our setting, we use methods similar to these to identify weights that can be modified to retain source domain performance while achieving improved target domain performance.

We analyse these different masking methods on an image classification task and present that trade-offs that each method offers in this paradigm.

## 2. Related Work

Our setting is similar to transfer learning, where a model trained on one dataset is finetuned on another dataset [5, 6, 1] and domain adaptation, where the learning algorithm

*Equal contribution. Published in Out of Distribution Generalization in Computer Vision (OOD-CV) workshop at ICCV 2023.

has access to training data from *multiple* source domains and must transfer to a target domain [8, 9]. However, we differ from both because our learning algorithm must adapt to a distribution shift (typically in the inputs) and not a new prediction task and we explicitly care about performance in the first task in addition to the model's performance on the transfer task. Further, we assume access to only one source domain, in contrast to several domain adaptation methods.

Multi-task learning [20, 23] is another setting that tackles the problem of simultaneously learning multiple learning tasks at once. However, in this setting, methods typically assume simultaneous access to all training tasks of interest. Our setting perhaps is closest to a two-task continual learning paradigm [16, 24] where tasks are presented in a sequential manner and at the end of training, the model is evaluated on *all tasks*, and must hence retain predictive performance on previous tasks.

We also note that prior work [14, 13, 15] has tackled the problem of pruning the weights and filters of a CNN but this was done with the goal of network optimization rather than domain generalization.

## 3. Problem Setup

Consider a source domain task $\mathcal{S}$ and a model trained on this domain, $f_\theta$, using a source dataset $\mathcal{D}_\mathcal{S}$. *We assume that we no longer have access to $\mathcal{D}_\mathcal{S}$ once the model has been trained on it.* We have a target domain task $\mathcal{T}$ we would like to generalize to. We would like to modify $f_\theta$ to obtain a model $f_{\theta'}$ that performs well on both the source and target domains. This modification must be achieved with a dataset $\mathcal{D}_\mathcal{T}$ from the target domain. We would like to achieve this by modularizing the weights of $f_\theta$ by splitting the parameters $\theta$ into $\theta_{\text{specialize}}$ and $\theta_{\text{reuse}}$ as described in [4].

## 4. Method

The core idea is that modularizing the weights of a pretrained network into $\theta_{\text{reuse}}$ and $\theta_{\text{specialize}}$ can mitigate forgetting on $\mathcal{S}$. Concretely, we can freeze the $\theta_{\text{specialize}}$ weights when fine-tuning on $\mathcal{T}$ so we don't change the weights that are important to $\mathcal{S}$.

There are several methods one can think of to split the weights into $\theta_{\text{specialize}}$ and $\theta_{\text{reuse}}$ . One common theme is to generate a decision function that determines whether each individual weight $w$ can be modified or not. For a given layer of weights $\mathcal{W} \in \mathbb{R}^D$, we would like to learn a binary mask $\mathcal{M} \in \{0,1\}^D$, where each weight can be changed during finetuning if its mask value is 0, and vice-versa.

### 4.1. Naive masking

First, we try a naive method to learn a mask $\mathcal{M}$ from the weights $\mathcal{W}$ . To this end, we first analyze the distribution of the weights in $\mathcal{W}$ and note that the trained weights are normally distributed (fig. 1).

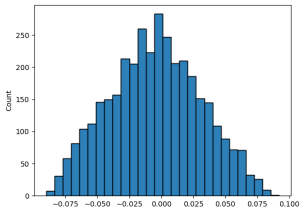


Figure 1. Weight distribution after training on the Photo domain

One way to interpret this distribution is that weights that are further away from the mean are more important for the task than weights that are close to the mean. The idea is that the "extreme" weights are important to the source task and represent $\theta_{\text{specialize}}$ . These must be kept frozen for the fine-tuning process.

Concretely, let the mean of the weights in $\mathcal{W}$ be $\mu$ and the standard deviation be $\sigma$. We set $\mathcal{M}$ such that the values in $\mathcal{M}$ are 1 if $(w > \mu + \sigma) \mid (w < \mu - \sigma)$ where $w$ is each weight in $\mathcal{W}$ .

### 4.2. Editor networks based

Directly learning a mask $\mathcal{M}$ , while ideal, can prove to be challenging since discrete distributions cannot be differentiated through easily [11, 22]. One solution is to borrow concepts from model editor networks literature [17, 3] to learn an auxiliary $\Delta\mathcal{W} \in \mathbb{R}^D$ from which we can recover $\mathcal{M}$ using simple thresholding. We train a model on some source task $\mathcal{S}$ to convergence and let the last layer weights of this learned model be $\mathcal{W}$ . We would like to learn a $\Delta\mathcal{W}$ such that we can edit the weights of the learned model by replacing $\mathcal{W}$ with $\mathcal{W}$ + $\Delta\mathcal{W}$ *which we get by training on the same source domain.* $\Delta\mathcal{W}$ should allow us to make as many edits as possible to the already tuned weights $\mathcal{W}$ without degrading performance on the source domain. If it is possible to edit the weight while maintaining performance on the source domain, then the weight probably does not belong to $\theta_{\text{specialize}}$ . We would also like $\theta_{\text{specialize}}$ to be a small set so we have enough network capacity when we finetune on the target domain. Hence, we add an L1 penalty that encourages the $\Delta\mathcal{W}$ values to be non-zero. So the overall objective is to reduce the cross-entropy loss (in the source domain) while also increasing this L1 penalty. Note that this objective only updates $\Delta\,\mathcal{W}$ and not $\mathcal{W}$ itself.

Now that we have our trained auxiliary mask, $\Delta\mathcal{W}$ , we can recover a mask $\mathcal{M}$ by heuristically thresholding the values in $\Delta\mathcal{W}$ . Following a similar strategy from section 4.1, we first compute the magnitude of $\Delta\mathcal{W}$ by taking the absolute value of its entries and freeze weight values more than one standard deviation from the mean.

### 4.3. Binary masks

We now explore a method that directly learns a binary mask. If we can learn a $\mathcal{M}$ directly, we can simply freeze

the weights indicated by a 1 in $\mathcal{M}$ and fine-tune the rest. The weights that are getting masked out should not be very useful for the source task since the performance on the source task doesn't degrade when we mask these weights.

In general, since we cannot differentiate through discrete sampling operations, it is difficult to learn discrete masks $\mathcal{M}$ with gradient based methods. However, Jang et al. [11] show that the Gumbel distribution can be used to generate discrete samples in a differentiable fashion and has been used successfully in downstream works [4, 18]. We encourage the reader to refer to the original paper [11] for a detailed description of the technique.

The training procedure is as follows: once we have a pretrained model on the source domain, we would like to learn a binary weight mask $\mathcal{M}$ that will replace the final $\mathcal{W}$ of the model with $\mathcal{W}$ * $\mathcal{M}$ where * represents an element-wise multiplication operation. Our objective is to decrease cross-entropy loss while masking out as many weights as possible. We use a penalty term similar to the one we saw in the previous section to achieve this. We find that this penalty term is crucial in recovering a fairly sparse mask at the end of training. During training, for stability, we sample multiple masks per batch as prescribed by Csordás et al. [4].

### 4.4. Using the trained masks for fine-tuning

Once we have trained our masks on the source domain $\mathcal{S}$, we can fine-tune our model on the new target domain $\mathcal{T}$, while not losing performance on the source domain. For all parameters $\theta_i$ such that $b_i = 1$ (i.e. $\theta_i \in \theta_{\text{specialize}}$), we let $\theta_i = \theta_i^{\mathcal{S}*}$, where $\theta^{\mathcal{S}*}$ denotes the fully trained weights on the source domain. For all parameters $\theta_j$ such that $b_j = 0$ (i.e. $\theta_j \in \theta_{\text{reuse}}$), we have three options:

(i) $\theta_j = \theta_j^{\mathcal{S}*}$, where we start updating $\theta_j$ from its final trained value on $\mathcal{S}$

(ii) $\theta_i^{\mathcal{S}}$, where $\theta^{\mathcal{S}}$ denotes the original initialisation of the weights for the model *before* it was trained on the source domain $\mathcal{S}$. This motivation is inspired from the Lottery Ticket Hypothesis [7].

(iii) $\theta_i^{\text{random}}$, where we consider a fully random re-initialisation of $\theta_{\text{reuse}}$.

We study the effect of these initialization strategies in section 5.3. Our default initialization strategy is (ii).

## 5. Experiments

In this section we explore the effect of varying masking strategies with respect to (i) performance on the target domain and (ii) performance on the source domain after fine-tuning on the target domain.

**Dataset** We conduct our experiments on the PACS dataset [12]. PACS is an image dataset for domain generalization. It consists of four domains, namely Photo (1,670 images), Art Painting (2,048 images), Cartoon (2,344 images) and Sketch (3,929 images). Each domain consists of the same seven classification categories: dog, elephant, giraffe, guitar, house, and person. We wish to study how training on a source domain (eg: Photos), then training on a target domain (eg: Sketch), affects the performance of the model on the original source domain (eg: Photos).

**Model** We use a ResNet-18 [10] pretrained with ImageNet weights [21]. Early experiments determined that using a larger model or finetuning all weights yield negligible improvements in accuracy compared to tuning just the last layer. Therefore, we finetune and mask only the final layer (the MLP head) of the model in all our experiments below.

### 5.1. Upper and Lower Bounds

We compare each masking method relative to finetuning all weights of the MLP head on $\mathcal{D}_{\mathcal{S}}$ (similar to the masking methods), and then also to finetune all weights of the MLP head on $\mathcal{D}_{\mathcal{T}}$. We denote this as unmasked finetuning. We expect unmasked finetuning to show the largest drop in performance on $\mathcal{S}$ and the biggest gain on $\mathcal{T}$ compared to all masking methods, demonstrating the problem of "forgetting" by giving us approximate lower and upper bounds, respectively. Thus, rather than plotting the absolute accuracy of each masking method and unmasked finetuning, we compare the difference in accuracy between each masking method and unmasked finetuning on: (i) the source domain $\mathcal{S}$, denoted as **source gain**, to indicate the increase in performance achieved by the masking method on the source domain over unmasked finetuning (ii) the target domain $\mathcal{T}$, denoted as **target gain**, reasoned similarly. These two metrics allow us to compare the benefits of each masking method relative to each other, and across domains.

### 5.2. How do the different masking strategies compare against each other?

We plot the source gain and target gain for each masking method in figure 2. We find that learned binary masks are much better than other masking methods on the source gain metric, thus "remembering" more from the source domain even after being finetuned on the target domain. However, this comes at the cost of decreased performance on the target domain. Naive masking shows no discernible improvement or regression on source gain, but is consistently lower than unmasked tuning on the target domain. Finally, editor-based masking shows improvements on the target domain but at the cost of "forgetting" more on the source domain. Across all masking methods, there seems to be a non-linear tradeoff in improving source domain performance at the cost of decreasing performance on the target domain. Interestingly, directly learning binary masks does more to combat forgetting than editor-based masks, while the latter un-freezes more of the right weights to improve target gain.


Figure 2. Performance of different masking strategies on the source and target domains after fine-tuning on the target domain. The *x-axis* represents the gain in accuracy in source performance when compared to a model that is finetuned directly on the target domain and the *y-axis* the gain in accuracy on target domain performance under the same setting. Ideally we would like large positive values on both axes. The shapes indicate the different source domains as shown in the legend and the colors indicate the different masking strategies. The shading represents the standard deviation on each axis and the center is the mean across 3 seeds. For binary masking, the variance in target gain is larger than for source gain, which still demonstrates its increased capacity to avoid "forgetting".

| Domain | Photo | | Art | | Cartoon | | Sketch | |
|---|---|---|---|---|---|---|---|---|
| Method | $\mathcal{S}$ Gain | $\mathcal{T}$ Gain | $\mathcal{S}$ Gain | $\mathcal{T}$ Gain | $\mathcal{S}$ Gain | $\mathcal{T}$ Gain | $\mathcal{S}$ Gain | $\mathcal{T}$ Gain |
| $\mathcal{S}$ Start | **0.009** | **-0.028** | 0.041 | **-0.020** | **0.029** | **-0.019** | **0.088** | -0.086 |
| $\mathcal{S}$ End | **0.009** | -0.029 | 0.050 | -0.054 | 0.026 | -0.036 | 0.057 | -0.050 |
| Random | 0.005 | **-0.028** | **0.051** | -0.056 | 0.026 | -0.036 | 0.054 | **-0.048** |

Table 1. Ablation study to determine best method of initialisation for $\theta_{\text{reuse}}$. Refer to section 5.1 for a description of $\mathcal{S}$ Gain and $\mathcal{T}$ Gain.

### 5.3. How does weight initialization affect the target-domain performance of binary masking?

We compare the three different weight initialization strategies for the binary masking method as described in section 4.4. The results are shown in table 5. The different initialization methods perform similarly but we notice that using initial weights from $\mathcal{S}$ as described in method (ii) in section 4.4 is almost consistently better and this is in line with the results from Frankle and Carbin [7].

## 6. Conclusion and Future Work

In this work, we compared the efficacy of 3 different masking strategies and their effect on alleviating the forgetting problem on transfer learning settings. We showed that the binary masking strategy is effective at alleviating forgetting while the real-valued masking strategies are better at target domain performance. We also study the effect of different weight initializations in the fine-tuning process.

For future work, we would like to delve into the reason behind the differences in source and target gain performance across the masking methods. Additionally, we would explore more complex masking strategies that are founded in Information Theory and linear algebra ideas such as low-rank approximations of weights. We also hope to investigate continual learning and multi-task settings where there are more than two domains to generalize to.

## References


[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[3] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *EMNLP*, 2021.

[4] Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=7uVcpu-gMD.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[7] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1): 2096–2030, jan 2016. ISSN 1532-4435.

[9] Jiang Guo, Darsh J. Shah, and Regina Barzilay. Multi-source domain adaptation with mixture of experts. In *EMNLP*, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[11] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. URL https://arxiv.org/abs/1611.01144.

[12] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Deeper, broader and artier domain generalization. pages 5543–5551, 10 2017. doi: 10.1109/ICCV.2017.591.

[13] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[14] Jian-Hao Luo and Jianxin Wu. An entropy-based pruning method for cnn compression, 2017.

[15] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. Thinet: Pruning cnn filters for a thinner net. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(10):2525–2538, 2019. doi: 10.1109/TPAMI.2018.2858232.

[16] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018. doi: 10.1073/pnas.1803839115. URL https://www.pnas.org/doi/abs/10.1073/pnas.1803839115.

[17] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0DcZxeWfOPt.

[18] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations, 2017. URL https://arxiv.org/abs/1703.04908.

[19] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

[20] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ry8dvM-R-.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014. URL https://arxiv.org/abs/1409.0575.

[22] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018.

[23] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization, 2020.

[24] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3987–3995. JMLR.org, 2017.