Layer Autotuning: Automated Fine-tuning Algorithms For Transfer Learning

Kelechi Uhegbu
Department of Symbolic Systems
Stanford University
kuhegbu@stanford.edu

Shaunak Bhandarkar
Department of Mathematics
Stanford University
shaunakb@stanford.edu

Kai Fronsdal
Department of Computer Science
Stanford University
kaif@stanford.edu

Abstract[1]

In the domain of transfer learning, model fine-tuning plays a central part in the construction of many models, especially in a low data regime. Pretraining on a larger source dataset followed by fine-tuning on a target dataset has been shown to improve generalization and performance on the target distribution. One recently proposed fine-tuning approach is that of surgical fine-tuning, whereby fine-tuning some small subset of the layers of the neural network can lead to better performance than fine-tuning all of the layers in terms of learning a distribution shift [1].

However, it is still an open question whether the process of surgical fine-tuning can be effectively automated, i.e. whether the process of determining which layers one should fine-tune can be incorporated as part of the transfer learning process. Keeping in line with the methodology of [1], we specifically focus on modeling three kinds of distribution shifts: input-level shifts, population-level shifts, and output-level shifts. We investigate multiple optimization algorithms that automate the fine-tuning process and evaluate their performance relative to the full and single-layer fine-tuning benchmarks.

One class of models we explore are multi-armed bandit (MAB) models, which treat each layer of the pretraind network as a stochastic, non-stationary source of reward; the goal of fine-tuning is then to determine the optimal policy for tuning the layers of the network. We test two kinds of MABs: a modifed epsilon-greedy model and a best-empirical-sampled-average (BESA) model. Our other auto-fine-tuning model is based on the the MAML algorithm [2], whereby each layer's learning rate is thought of as a learnable meta-parameter.

We find that the MAML-based model is best suited for fine-tuning on input- and output-level distribution shifts, as it performs on par with the full and single-layer fine-tuning benchmarks. While the MAB approaches perform fairly for these same kinds of distribution shifts, they do not always reach the benchmark. Thus, future work would involve finding MABs that are better-suited to model non-stationary distributions as well as finding a more reliable metric for reward.

In this research, we have tested layer autotuning algorithms on relatively simple distribution shifts, but future work would involve a theoretical analysis of distribution shifts and how fine-tuning different layers helps with learning such shifts; a

[1]Our code can be accessed at the following Github repository https://github.com/kelechiu10/CS330Project

better theoretical understanding could better inform future MAB-based approaches for automated fine-tuning.

1 Introduction

Deep neural networks have performed impressively at many tasks over the last decade. However, these models tend to be brittle to small distribution shifts between the source and target data [3, 4]. One common approach to improve generalization is to first pretrain a model on a large source dataset and then fine-tune on a small target dataset. Collecting small target datasets can be a cost-effective and efficient way to improve downstream performance that can outperform other generalization and unsupervised adaptation methods [5].

Fine-tuning methods generally attempt to adapt to the new data while still preserving relevant information from the source domain. This information retention is important to ensure the generalization to the target distribution and prevent overfitting to the small target dataset. Then a model will be able to take advantage of the general representational structure of the source domain to perform better in then target domain.

Recently, Lee et al (2022) proposed a *surgical fine-tuning* approach, which refers to carefully fine-tuning a small subset of the layers of a network rather than fine-tuning the entire network. In particular, they found that, for certain distribution shifts, surgically fine-tuning a pretrained model can result in better performance than full fine-tuning. Moreover, it is entirely possible that fine-tuning earlier layers could result in better performance than fine-tuning later layers [1].

In their paper, Lee et al (2022) consider three classes of simple distribution shifts: input-level shifts (e.g. data corruption), feature-level shifts (e.g. changes in data subpopulation), and output-level shifts (e.g. data label changes). However, one drawback of this approach is that *a priori* it may not be clear what kind of distribution shift occurred between the source and target distributions. Thus we would be required to fine-tune the model many times to find the layers that work best. Ideally, we would be able to fine-tune a model while automatically finding the optimal set of layers to fine-tune on.

It should be noted that Lee et al (2022) do consider an automated fine-tuning algorithm, called Auto-SGN (which we refer to throughout this paper as "GradNorm") [1]. During fine-tuning, this algorithm simply scales the learning rate of each layer by the relative gradients obtained for that layer; in other words, layers with larger gradients get fine-tuned more. While Auto-RGN works fairly well, this approach does not always perform as expected and results in worse accuracy than claimed – a result that we corroborate in our present work. Therefore, the central goal of our work to explore other automated fine-tuning approaches and investigate whether they can perform on par with, if not outperform, the full and surgical fine-tuning approaches.

In this paper, we explore two alternative approaches for automating the fine-tuning process and compare them to previous approaches:

1. Treating surgical fine-tuning as a multi-armed bandit problem from reinforcement learning. In particular, if we view each layer as an arm that yields reward based on relative decrease in loss due to fine-tuning that layer, then automating the surgical fine-tuning process is akin to finding the optimal policy for choosing the different arms.
2. Treating the learning rate α_ℓ associated to each layer ℓ as a trainable meta-parameter. Consequently, we can apply a variant of the model-agnostic meta learning (MAML) algorithm to determine suitable learning rates for each layer.

2 Related Work

Fine-tuning There is a current interest in improving the way we do fine-tuning. Work on transfer learning spans over two decades, and encompasses classification, regression, and clustering problems [6]. Prior work has explored fine-tuning pretrained neural networks to a target distribution in the context of computer vision [7, 8], as well as more generally factors that affect transferability of features [9]. Additionally, the surgical fine-tuning approach relates to earlier work on facilitating transfer learning through specific uses of parameter freezing [10, 11, 12]. In particular, it has been shown that full-fine tuning can sometimes lead to worse performance on novel tasks [13].

Distribution Shifts There has been prior work in analyzing different distribution shifts [14] and how best to adapt to them, within the domains of computer vision and natural language processing [15] [16]. There has also been recent work on working towards domain generalization [17].

Multi-Armed Bandits To the best of our knowledge, there is no prior work investigating multi-armed bandits as a method for automating the fine-tuning process. However, there has been work on using multi armed bandits to the collaborative filtering problem [18] and in studying the sequential transfer problem in an online learning setting [19]. Numerous variations of multi-armed bandits – such as BESA [20], DiscountedThompson [21], SlidingWindowUCB [22], and KL-UCB [23] – have been proposed in the context of non-stationary reinforcement learning problems.

3 Datasets

In our present work, we focused on three types of distribution shifts discussed in [1]. Future work would certainly involve analyzing datasets with more complex distribution shifts.

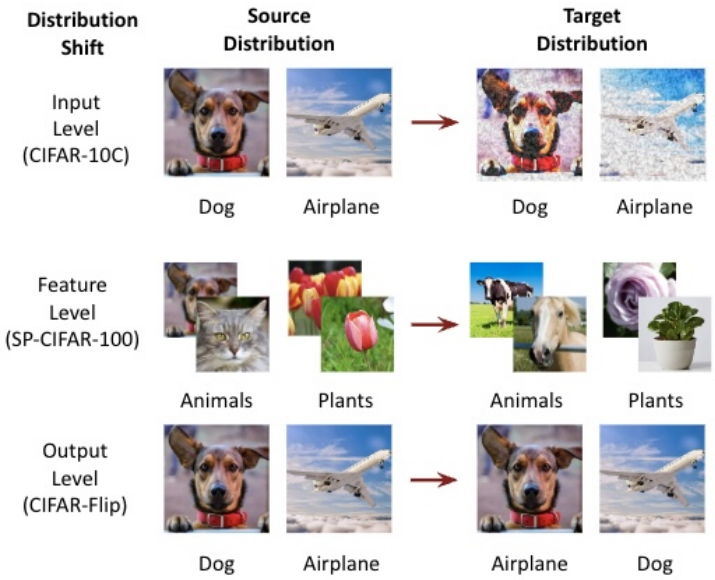

Figure 1: Description of each dataset used and the distribution shift that it corresponds to.

3.1 Input-Level Distribution Shifts

To model an input-level distribution shift, we sampled images from the CIFAR-10C dataset [24], a relatively small dataset (compared to the CIFAR-10 dataset [25]) that consists of images from the CIFAR-10 dataset that have been corrupted through one of 15 possible image corruptions (e.g. gaussian noise, motion blur, added brightness, etc.). We sampled 1995 images in total, with 133 images from each of the 15 classes of corruptions.

3.2 Feature-Level Distribution Shifts

To model a feature-level distribution shift, we sampled a collection of training images from the CIFAR-100 dataset [25], a dataset with 100 classes of images that are grouped into 20 superclasses. Then, to create a test set, we sampled from different representatives within each of the 20 superclasses than those that we had sampled to create the test set. For example, within the "flowers" superclass, one might sample from the subclasses "poppies" and "tulips" when creating the test set rather than from "roses" and "sunflower," if the latter were used to create the train set.

3.3 Output-Level Distribution Shifts

To model an output-level distribution shift, we sampled images from the CIFAR-10 dataset, except where the label corresponding to any given image was shuffled, i.e. we swap each class label in the CIFAR-10 dataset with 9 minus that label to create the CIFAR-Flip dataset. For example, if a "cat" image in CIFAR-10 has the label '3', it will have the flipped label '6' in CIFAR-Flip. We sampled 2000 images in total.

4 Methodology

For each of our experiments, we started with a pretrained ResNet 50 network that was either trained on the CIFAR-10 or ImageNet dataset. Given that all of our tests involved different fine-tuning approaches, we embedded each fine-tuning algorithm within a PyTorch optimizer. In particular, we had an optimizer for full fine-tuning, for single-layer fine-tuning, our multi-armed bandit approaches, and for our MAML approach.

4.1 Optimizers

For each of the below optimizers, we performed hyper-parameter optimization over the the different learning rates $\{1\text{e}{-3}, 1\text{e}{-4}, 1\text{e}{-5}\}$. For methods that group layers together into blocks, we treat each of the four ResNet layers as one block and the final fully connected layer as the fifth block. We used cross entropy loss for all datasets.

4.1.1 Full

Full fine-tuning consisted of an Adam optimizer (with default parameters). During fine-tuning we train the entire network in the same manner it was trained on the source dataset.

Mathematically, this process works as follows. Suppose the pretrained neural network has parameters θ and we wish to further train (i.e. fine-tune) this network on a smaller training dataset $\mathcal{D}^{tr}$ using the loss function $\mathcal{L}$. Then, letting $\phi^{(t)}$ denote the parameters of the fine-tuned network at time step t, we have that $\phi^{(0)} = \theta$ and

$$\phi^{(t+1)} = \phi^{(t)} - \alpha \nabla_{\phi} \mathcal{L}(\phi^{(t)}; \mathcal{D}^{tr})$$

where α denotes the fine-tuning learning rate.

4.1.2 Layerwise

As in [1], we split the model into five blocks and fine-tune the model five times, only updating the parameters of a single layer each time. We then select the best layer overall for that run (i.e. the layer that results in the highest validation accuracy).

4.1.3 GradNorm

This optimizer is based on a method proposed by [1] as a method of automatic fine-tuning. The main idea behind this method is that layers with gradients of larger magnitude likely will contain more information about the target task. In particular, at each time step, we compute the relative gradient magnitude for layer i as $\frac{\|g_i\|}{\|\phi_i\|}$, where g_i denotes the matrix of gradients at layer i and ϕ_i denotes the current parameters at layer i. We then normalize the relative gradient magnitudes and scale each layers' learning rate by its normalized relative gradient magnitude.

4.1.4 Multi-Armed Bandit (MAB) Optimizer

We can formulate choosing which layers to fine-tune can as a non-stationary multi-armed bandit problem where we need to explore different layers to fine-tune, but we also want to spend most of our effort fine-tuning the most effective layers. The fine-tuning scenario is non-stationary because the distribution of rewards (decrease in loss) will change over time as the model parameters are updated. This exploration/exploitation trade off is well suited for black-box multi-armed bandit solutions. Additionally, we need to be able to handle arbitrary reward magnitudes due to the stochastic nature of training. One benefit of this approach is that it only fine-tunes the model once.

Each layer of the network is treated as one arm of the bandit. At each step of training i, the multi-armed bandit chooses a single layer to fine-tune, and receives a reward at that time step of

$$R^{(i)} = \frac{L^{(i-1)} - L^{(i)}}{L^{(i-1)}}$$

where $L^{(i)}$ is the loss at time step i. In other words, the reward obtained for fine-tuning a given layer is reflective of the relative drop (or lack thereof) in the training loss.

Unfortunately, most state-of-the-art multi-armed bandit solutions–such as klUBC variants, BayesUBC, and Discounted Thompson–are not well suited to this environment as they are either designed for Bernoulli rewards and/or stationary reward distributions. Thus we settled on the following two approaches:

1. **Epsilon Greedy**. We employ a variant of the standard epsilon greedy approach. Similar to the standard epsilon greedy algorithm, at each time step, with probability $\varepsilon = 0.1$ a random layer is chosen to be fine-tuned, and with probability $1 - \varepsilon$ the layer with the current maximum cumulative reward is fine-tuned. However, we alter the algorithm to only use the last 100 rewards in order to adapt to changing reward distributions.
2. **Best-Empirical Sampled Average (BESA)**. BESA is a state-of-the-art non-stationary multi-armed bandit agent based on the idea that comparing empirical means of different sample sizes is not "fair." Instead the algorithm subsamples the received rewards to get estimates of each arm's mean reward with the same sample size. It then selects the arm with the largest estimate. For more information as to why this algorithm works so well, see [20].

4.1.5 MAML

Instead of trying to find the best layers to fine-tune heuristically (where best here refers to achieving the largest accuracy on the target distribution), we can directly learn them. Pulling from the Model-Agnostic Meta-Learning (MAML) method [2] , for each layer ℓ with parameters θ_ℓ we treat its learning rate λ_ℓ as a trainable meta-parameter. Formally, for each time step i we first compute

$$\theta_\ell^{(i)} = \theta_\ell^{(i-1)} - \alpha_\ell^{(i-1)} \nabla_{\theta^{(i-1)}} \mathcal{L}(\theta_\ell^{(i-1)})$$

for each layer ℓ. We then perform optimization with the new parameters $\theta^{(i)}$ using Adam to get $\alpha_\ell^{(i)}$.

For our implementation we split each batch in half, and compute $\theta_\ell^{(i)}$ with the first half of the batch and $\alpha_\ell^{(i)}$ with the second half. While this method achieves very good results, it can be slow to run as the optimizer must take second order gradients.

4.2 Experiments

For each optimizer and dataset we tested against three different learning rates: $\{1\text{e}{-3}, 1\text{e}{-4}, 1\text{e}{-5}\}$.

- **Input-Level Shift.** We fine-tuned a ResNet 50 pretrained on CIFAR-10 for 15 epochs on the sampled 1995 images from CIFAR-10C, using each of the optimizers above.
- **Feature-Level Shift.** We took a pretrained ResNet 50 and trained it on the source distribution for SP-CIFAR-100 (36000 images). Then, we fine-tuned on the target SP-CIFAR-100 dataset (2400 sampled images) for 15 epochs for each optimizer.
- **Output-Level Shift.** We fine-tuned a ResNet 50 pretrained on CIFAR-10 for 15 epochs on the sampled 2000 CIFAR-Flip images, using each of the above optimizers.

5 Results

The main results of our work are summarized in Table 1 below. Validation accuracies leading up to the final accuracies presented in Table 1 are presented in Figure 3. Additionally, to better understand which layers each of our methods were choosing throughout the fine-tuning process, we generated plots of each method's layer-selection frequency breakdown, shown in Figure 2.

Method	CIFAR-10C (±3)	SP-CIFAR-100 (±4)	CIFAR-Flip (±2)
Full	**96.4**	**65.8**	82.2
Layerwise	**95.6**	**66.9**	**96.3**
GradNorm	93.1	38.9	93.2
MAB (epsilon greedy)	**94.3**	38.4	88.6
MAB (BESA)	93.4	41.2	92.3
MAML	**94.9**	47.3	**97.3**

Table 1: Comparison of the average accuracy for each optimizer across three distribution shifts. Standard error is indicated in parenthesis. Of the automatic methods, MAML performs the best; however layerwise fine-tuning is always within the top two methods and dramatically out performs all automatic methods for feature level distribution shifts.

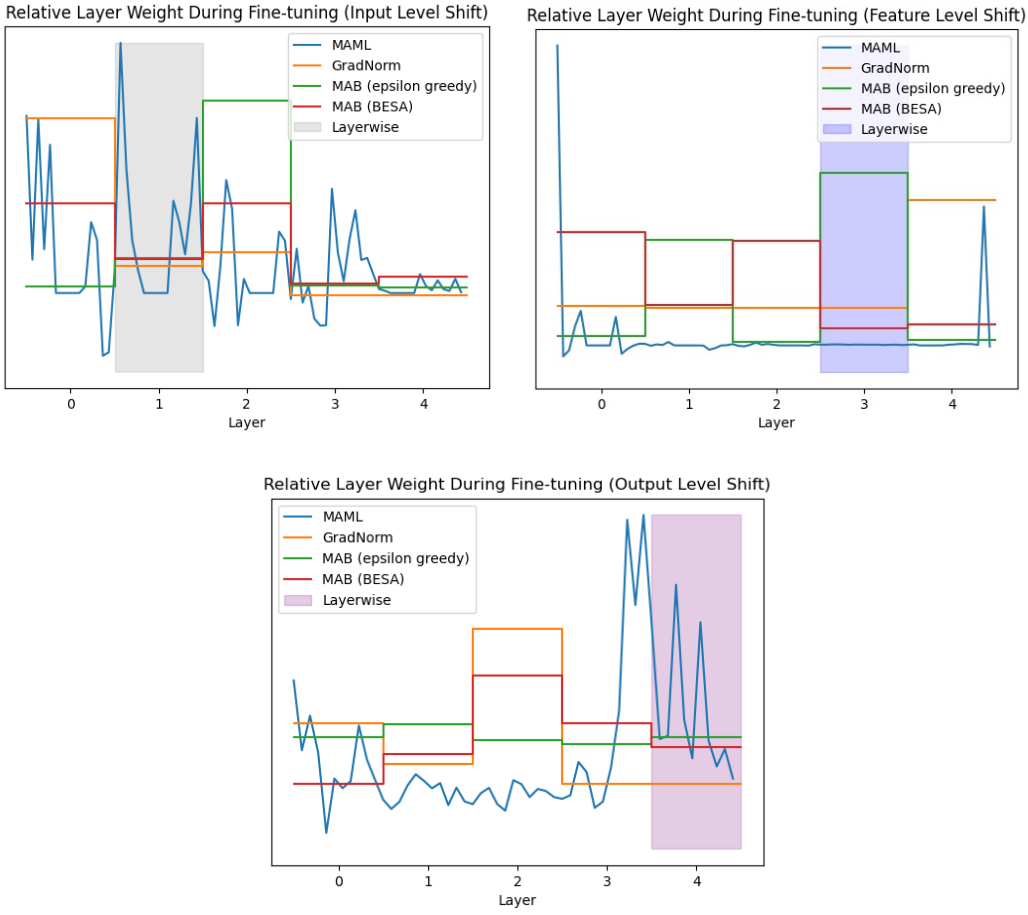

Figure 2: A comparison of the layer-selection frequencies employed by our different approaches for input-level, feature-level, and output-level distribution shifts. We can see that MAML most consistently chooses the optimal layer, but fails dramatically under feature level shifts (but still achieves reasonable accuracy).

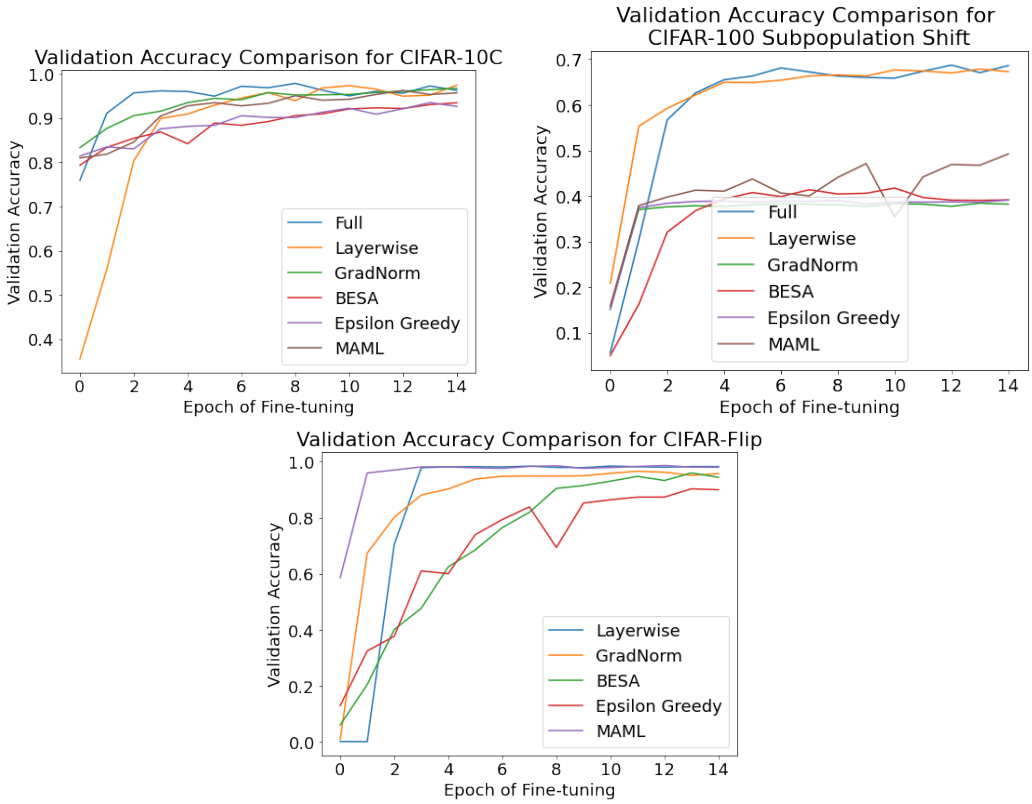

Figure 3: A comparison of the validation accuracies throughout 15 epochs of fine-tuning for the different approaches on CIFAR-10C (input-level shift), SP-CIFAR-100 (feature-level shift), and CIFAR-Flip (output-level shift). MAML tends to converge much quicker in general than the other methods – even for SP-CIFAR-100, where it does not perform as well as the baseline methods.

5.1 Input-Level Shift Results

For our experiments with the CIFAR-10C dataset, we found that full fine-tuning resulted in the best final validation accuracy (96.4%), though MAML and the Epsilon Greedy bandit performed on par with this baseline (given that the 95% confidence interval was 3%). Our best non-baseline automated method was MAML, which produced a final validation accuracy of 94.9%; indeed, MAML tended to choose layer 1, which was the layer that maximized validation accuracy for layerwise fine-tuning. BESA and GradNorm also performed reasonably well, though they did not perform up to the full fine-tuning benchmark. Interestingly, the MABs did not show a strong preference for layer 1; Epsilon Greedy in fact preferred fine-tuning layer 2 and BESA tended to choose layers 1 and 3. This highlights the fact that our MAB approaches enable multiple layers to be selected throughout a given policy, but also the limitation that the reward metric for our MABs isn't necessarily always aligned with layerwise optimization.

5.2 Feature-Level Shift Results

For the feature-level shift data, our experiments resulted in lower validation accuracies across the board. This might be because the distribution shift of different subpopulations is a harder problem to generalize than adding a corruption (input-level shift) or moving around labels (output-level shift). Additionally, at an intuitive level, a feature-level shift may be less interpretable. For instance, fine-tuning layers 2 and 3 had similar performance, which could mean that this kind of distribution shift may not be learned best by tuning a single layer. In line with this idea, it's possible that a limitation of our MAB approaches is that they only fine-tune a single layer at a time, whereas fine-tuning multiple layers (e.g. 2 layers) at a time could lead to better results for feature-level distribution shifts. It's also possible that fine-tuning for more epochs may have allowed our non-baseline approaches to perform

on par with full and layerwise fine-tuning, although it should be noted that a key aspect of fine-tuning is that it should be done for a relatively small number of epochs (so as to *quickly* learn the target distribution).

The performance of the automatic methods show that a feature-level shift can not be easily explained through a synthethic distribution shift of the parameters themselves as claimed by [1]. They previously describe how fine-tuning with GradNorm is able to match synthetic distribution shifts on a single layer. However, it seems that the feature-level shift of SP-CIFAR-100 can not be easily explained through changing the weights of just one layer. This may mean that real world data distribution shifts might be more complex than what were initially expected.

5.3 Output-Level Shift Results

Finally, for the output-level shift data, MAML outperformed the full and layerwise baselines, as indicated in Table 1. In this regime, while GradNorm and the bandits performed fairly, they fell noticeably short of the layerwise benchmark. As seen in Figure 2, MAML focused on the layers that would be most beneficial to the distribution shifts. For the output level shift it focused on the final layers. However, the MAB algorithms and GradNorm weighed the middle layer the most, which may relate to how the performance was worse in comparison. Just like with input-level shifts, this finding underscores the need to further refine the reward metric for future MAB approaches, e.g. rather than being based on train loss, perhaps reward could be based on validation loss or accuracy.

6 Conclusion

This paper builds on the surgical fine-tuning work done in [1] to incorporate optimal layer selection into the transfer learning process. We generate baseline results from full and layerwise fine-tuning – and also from the automated GradNorm approach originally introduced in [1] – for input-level, feature-level, and output-level shifts. We explore the concept of a MAB optimizer where we treat the optimization problem as a multi-armed bandit and that of a MAML-based fine-tuning algorithm against. We find that MAML generally performs at the same level or above full and individual layer fine-tuning on these datasets. However, MAML is a much slower algorithm than the other automated methods. Although the MAB methods did not outperform the baseline results, we suggest that there is scope to refine this approach in future work.

For the three kinds of distribution shifts we explored, layerwise fine-tuning was generally the best-performing baseline method. However, given the fact that layerwise fine-tuning takes *N* times longer on average than the MAB approaches (for N possible layers to fine-tune), it appears that there is a slight tradeoff between compute power and accuracy. We suggest that, in situations where a machine learning researcher does not have the time or compute to run their model for each possible layer to fine-tune, our automated MAB fine-tuning approaches or GradNorm constitute a reasonably runtime-efficient substitute.

7 Future Work

7.1 Incorporating Runtime Analysis

In our paper, we explored multiple different approaches for "layer autotuning;" however, as alluded to in the previous section, we have not yet analyzed the runtime versus performance for each of our fine-tuning approaches. For instance, while MAML converges the fastest, it also takes longer to train over all. In particular, layerwise fine-tuning and MAML were typically the best performing methods across the three datasets, but they also had noticeably longer train times than the other models. Therefore, future work would further investigate the tradeoff between runtime and performance for these fine-tuning approaches across different numbers of epochs of training or different dataset sizes, and also quantify the threshold for choosing one approach over another.

7.2 Refining MAB Approaches

Additional automated fine-tuning methods, such as other variations of multi-armed bandits, remain to be explored. It would also be interesting to compare our current MAB optimizers - Epsilon Greedy

and BESA - against the full and layerwise benchmarks for more complex distribution shifts; in this setting, it is possible that these MAB approaches hold promise. In addition, there may be other reward functions besides relative drop in training loss that can provide more fruitful results for performance. One possibility is to directly incorporate validation accuracy into the reward metric, though it remains to be explored how this may be achieved in a runtime-efficient manner.

7.3 Theoretical Analysis of Distribution Shifts

In future work, we hope to make our results more interpretable. Specifically, for the tests that we ran, the corresponding distribution shifts were known in advance; for datasets with less clear (and more complex) distribution shifts, it is unclear what combination of layers should be fine-tuned, much less why. Furthermore, we do not have a good understanding on why different MAB approaches work better for some distribution shifts over others. More theoretical work on how fine-tuning different layers affects learning a distribution shift could better inform future MAB approaches. Finally, it remains to be seen whether automated fine-tuning approaches, such as MAB and MAML, could feasibly be combined to better learn certain distribution shifts.

8 Contributions

- Shaunak: Created dataloader for CIFAR-Flip (and worked on parts of CIFAR-10C dataloader), implemented full and layerwise baselines, and tinkered with different MAB approaches and reward metrics.
- Kai: Worked on CIFAR-10C and CIFAR-Flip dataloaders, implemented the GradNorm, Multi-armed Bandit, and MAML optimizers. Created general code outline. Performed hyperparamter optimization. Researched best MAB solutions.
- Kelechi: Sourced and created dataloaders for the datasets, created source and target CIFAR-100 and SP-CIFAR-100 datasets, created model to fine-tune on for SP-CIFAR-100, and fine-tuned models for SP-CIFAR-100.

All group members were involved in running models and logging the results.

References

[1] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017.

[3] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts, 2020.

[4] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet?, 2019.

[5] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations, 2022.

[6] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[7] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1717–1724, 2014.

[8] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition, 2014.

[9] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? 2014.

[10] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[11] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021.

[12] Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C. Mozer. Head2toe: Utilizing intermediate representations for better transfer learning. 2022.

[13] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

[14] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre Alvise-Rebuffi, Ira Ktena, Krishnamurthy, Dvijotham, and Ali Cemgil. A fine-grained analysis on distribution shift, 10 2021.

[15] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance, 2014.

[16] Yoonho Lee, Huaxiu Yao, and Chelsea Finn. Diversify and disambiguate: Learning from underspecified data, 2022.

[17] Marvin Mengxin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: Learning to adapt to domain shift. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[18] Anonymous. Neural collaborative filtering bandits via meta learning. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. under review.

[19] Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Sequential transfer in multi-armed bandit with finite set of models, 2013.

[20] Akram Baransi, Odalric-Ambrym Maillard, and Shie Mannor. Sub-sampling for multi-armed bandits. In *Machine Learning and Knowledge Discovery in Databases*, page 115–131. Springer, 2014.

[21] Vishnu Raj and Sheetal Kalyani. Taming non-stationary bandits: A bayesian approach, 2017.

[22] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems, 2008.

[23] Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. 2011.

[24] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[25] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).