RETHINKING THE HYPERPARAMETERS FOR FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning from pre-trained ImageNet models has become the de-facto standard for various computer vision tasks. Current practices for fine-tuning typically involve selecting an ad-hoc choice of hyper-parameters and keeping them fixed to values normally used for training from scratch. This paper re-examines several common practices of setting hyper-parameters for fine-tuning. Our findings are based on extensive empirical evaluation for fine-tuning on various transfer learning benchmarks. (1) While prior works have thoroughly investigated learning rate and batch size, momentum for fine-tuning is a relatively unexplored parameter. We find that picking the right value for momentum is critical for fine-tuning performance and connect it with previous theoretical findings. (2) Optimal hyper-parameters for fine-tuning in particular the effective learning rate are not only dataset dependent but also sensitive to the similarity between the source domain and target domain. This is in contrast to hyper-parameters for training from scratch. (3) Reference-based regularization that keeps models close to the initial model does not necessarily apply for "dissimilar" datasets. Our findings challenge common practices of fine-tuning and encourages deep learning practitioners to rethink the hyper-parameters for fine-tuning.

1 INTRODUCTION

Many real-world applications often have limited number of training instances, which makes directly training deep neural networks hard and prone to overfitting. Transfer learning with the knowledge of models learned on a similar task can help to avoid overfitting. Fine-tuning is a simple and effective approach of transfer learning and has become popular for solving new tasks in which pre-trained models are fine-tuned with the target dataset. Specifically, fine-tuning on pre-trained ImageNet classification models (Simonyan & Zisserman, 2015; He et al., 2016b) has achieved impressive results for tasks such as object detection (Ren et al., 2015) and segmentation (He et al., 2017; Chen et al., 2017) and is becoming the de-facto standard of solving computer vision problems. It is believed that the weights learned on the source dataset with a large number of instances provide better initialization for the target task than random initialization. Even when there is enough training data, fine-tuning is still preferred as it often reduces training time significantly (He et al., 2019).

The common practice of fine-tuning is to adopt the default hyperparameters for training large models while using smaller initial learning rate and shorter learning rate schedule. It is believed that adhering to the original hyperparameters for fine-tuning with small learning rate prevents destroying the originally learned knowledge or features. For instance, many studies conduct fine-tuning of ResNets (He et al., 2016b) with these default hyperparameters: learning rate 0.01, momentum 0.9 and weight decay 0.0001. However, the default setting is not necessarily optimal for fine-tuning on other tasks. While few studies have performed extensive hyperparameter search for learning rate and weight decay (Mahajan et al., 2018; Kornblith et al., 2018), the momentum coefficient is rarely changed. Though the effectiveness of the hyperparameters has been studied extensively for training a model from scratch, how to set the hyperparameters for fine-tuning is not yet fully understood.

In addition to using ad-hoc hyperparameters, commonly held beliefs for fine-tuning also include:

- Fine-tuning pre-trained networks outperforms training from scratch; recent work (He et al., 2019) has already revisited this.

- Fine-tuning from similar domains and tasks works better (Ge & Yu, 2017; Cui et al., 2018; Achille et al., 2019; Ngiam et al., 2018).
- Explicit regularization with initial models matters for transfer learning performance (Li et al., 2018; 2019).

Are these practices or beliefs always valid? From an optimization perspective, the difference between fine-tuning and training from scratch is all about the initialization. However, the loss landscape of the pre-trained model and the fine-tuned solution could be much different, so as their optimization strategies and hyperparameters. Would the hyperparameters for training from scratch still be useful for fine-tuning? In addition, most of the hyperparameters (e.g., batch size, momentum, weight decay) are frozen; will the conclusion differ when some of them are changed?

With these questions in mind, we re-examined the common practices for fine-tuning. We conducted extensive hyperparameter search for fine-tuning on various transfer learning benchmarks with different source models. The goal of our work is not to obtain state-of-the-art performance on each fine-tuning task, but to understand the effectiveness of each hyperparameter for fine-tuning, avoiding unnecessary computations. We explain why certain hyperparameters work so well on certain datasets while fail on others, which can guide future hyperparameter search for fine-tuning.

Our main findings are as follows:

- Optimal hyperparameters for fine-tuning are not only dataset dependent, but also depend on the similarity between the source and target domains, which is different from training from scratch. Therefore, the common practice of using optimization schedules derived from ImageNet training cannot guarantee good performance. It explains why some tasks are not achieving satisfactory results after fine-tuning because of inappropriate hyperparameter selection. Specifically, as opposed to the common practice of rarely tuning the momentum value beyond 0.9, we verified that zero momentum could work better for fine-tuning on tasks that are similar with the source domain, while nonzero momentum works better for target domains that are different from the source domain.
- Hyperparameters are coupled together and it is the effective learning rate—which encapsulates the learning rate, momentum and batch size—that matters for fine-tuning performance. While effective learning rate has been studied for training from scratch, to the best of our knowledge, no previous work investigates effective learning rate for fine-tuning and is less used in practice. Our observation of momentum can be explained as small momentum actually decreases the effective learning rate, which is more suitable for fine-tuning on similar tasks. We show that the optimal effective learning rate actually depends on the similarity between the source and target domains.
- We find regularization methods that were designed to keep models close to the initial model does not apply for "dissimilar" datasets, especially for nets with Batch Normalization. Simple weight decay can result in as good performance as the reference based regularization methods for fine-tuning with better search space.

2 Related Work

In transfer learning for image classification, the last layer of a pre-trained network is usually replaced with a randomly initialized fully connected layer with the same size as the number of classes in the target task (Simonyan & Zisserman, 2015). It has been shown that fine-tuning the whole network usually results in better performance than using the network as a static feature extractor (Yosinski et al., 2014; Donahue et al., 2014; Huh et al., 2016; Mormont et al., 2018; Kornblith et al., 2018). Ge & Yu (2017) select images that have similar local features from source domain to jointly fine-tune pre-trained networks. Cui et al. (2018) estimate domain similarity with ImageNet and demonstrate that transfer learning benefits from pre-training on a similar source domain. Besides image classification, many object detection frameworks also rely on fine-tuning to improve over training from scratch (Girshick et al., 2014; Ren et al., 2015).

Many researchers re-examined whether fine-tuning is a necessity for obtaining good performance. Ngiam et al. (2018) find that when domains are mismatched, the effectiveness of transfer learning is negative, even when domains are intuitively similar. Kornblith et al. (2018) examine the fine-tuning performance of various ImageNet models and find a strong correlation between ImageNet top-1 accuracy and the transfer accuracy. They also find that pre-training on ImageNet provides minimal benefits for some fine-grained object classification dataset. He et al. (2019) questioned whether ImageNet pre-training is necessary for training object detectors. They find the solution of training from scratch is no worse than the fine-tuning counterpart as long as the target dataset is large enough. Raghu et al. (2019) find that transfer learning has negligible performance boost on medical imaging applications, but speed up the convergence significantly.

There is much literature on the hyperparameter selection for training neural networks from scratch, mostly on batch size, learning rate and weight decay (Goyal et al., 2017; Smith et al., 2018; Smith & Topin, 2019). There are few works on the selection of momentum (Sutskever et al., 2013). Zhang & Mitliagkas (2017) proposed an automatic tuner for momentum and learning rate in SGD and empirically show that it converges faster than Adam (Kingma & Ba, 2014). There are also studies on the correlations of the hyperparameters, such as linear scaling rule between batch size and learning (Goyal et al., 2017; Smith et al., 2018; Smith, 2017). However, most of these advances on hyperparameter tuning are designed from training from scratch, but not examined on fine-tuning tasks for computer vision problems. Most work on fine-tuning just choose fixed hyperparameters for all fine-tuning experiments (Cui et al., 2018) or use dataset dependent learning rates in their experiments (Li et al., 2018). Due to the huge computational cost for hyperparameter search, only a few works (Kornblith et al., 2018; Mahajan et al., 2018) performed large-scale grid search of learning rate and weight decay for obtaining the best performance.

3 Tuning Hyperparameters for Fine-tuning

In this section, we first introduce the notations and experimental settings, and then present our observations on momentum, effective learning rate and regularization. The fine-tuning process is not different from learning from scratch except for the weights initialization. The goal of the process is still to minimize the loss function $L = \sum_{i=1}^{N} \ell(f(x_i, \theta), y_i) + \frac{\lambda}{2}\|\theta\|_2^2$, where ℓ is the loss function, N is the number of samples, x_i is the input data, y_i is its label, f is the neural network and θ is the model parameters. Momentum is widely used for accelerating and smoothing the convergence of SGD by accumulating a velocity vector in the direction of persistent loss reduction (Sutskever et al., 2013; Goh, 2017). The commonly used Nesterov momentum SGD (Nesterov, 1983) iteratively updates the model in the following form:

$$v_{t+1} = mv_t - \eta_t \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(f(x_i, \theta_t + mv_t), y_i) \tag{1}$$

$$\theta_{t+1} = \theta_t - v_{t+1} - \eta\lambda\theta_t \tag{2}$$

where θ_t indicates the model parameter at iteration t. The hyperparameters include the learning rate η_t, batch size n, momentum coefficient $m \in [0, 1)$, and the weight decay λ.

3.1 Experimental Settings

We evaluate fine-tuning on seven widely used image classification datasets, which covers tasks for fine-grained object recognition, scene recognition and general object recognition. Detailed statistics of each dataset can be seen in Table 1. We use ImageNet (Russakovsky et al., 2015), Place365 (Zhou et al., 2018) and iNaturalist (Van Horn et al., 2018) as source domains for pre-trained models. We resize the input images such that the aspect ratio is preserved and the shorter side is 256 pixels. The images are normalized with mean and std values calculated over ImageNet. For data augmentation, we adopt the common practices used for training ImageNet models (Szegedy et al., 2015): random mirror, random scaled cropping with scale and aspect variations, and color jittering. The augmented images are resized to 224×224. Note that state-of-the-art results could achieve even better performance by using higher resolution images (Cui et al., 2018) or better data augmentation (Cubuk et al., 2018).

We mainly use ResNet-101-V2 (He et al., 2016a) as our base network, which is pre-trained on ImageNet (Russakovsky et al., 2015). Similar observations are also observed on DenseNets (Huang et al., 2017) and MobileNet (Howard et al., 2017) (see Appendix B). The hyperparameters to be tuned (and ranges) are: learning rate (0.1, 0.05, 0.01, 0.005, 0.001, 0.0001), momentum (0.9, 0.99, 0.95, 0.9, 0.8, 0.0) and weight decay (0.0, 0.0001, 0.0005, 0.001). We set the *default* hyperparameter to be

Table 1: Datasets statistics. For the Caltech-256 dataset, we randomly sampled 60 images for each class following the procedure used in (Li et al., 2018). For the Aircraft and Flower dataset, we combined the original training set and validation set and evaluated on the test set. For iNat 2017, we combined the original training set and 90% of the validation set following (Cui et al., 2018).

Datasets	Task Category	Classes	Training	Test
Oxford Flowers (Nilsback & Zisserman, 2008)	fine-grained object recog.	102	2,040	6,149
CUB-Birds 200-2011 (Wah et al., 2011)	fine-grained object recog.	200	5,994	5,794
FGVC Aircrafts (Maji et al., 2013)	fine-grained object recog.	100	6,667	3,333
Stanford Cars (Krause et al., 2013)	fine-grained object recog.	196	8,144	8,041
Stanford Dogs (Khosla et al., 2011)	fine-grained object recog.	120	12,000	8,580
MIT Indoor-67 (Sharif Razavian et al., 2014)	scene classification	67	5,360	1,340
Caltech-256-60 (Griffin et al., 2007)	general object recog.	256	15,360	15,189
iNaturalist 2017 (Van Horn et al., 2018)	fine-grained object recog.	5,089	665,571	9,599
Place365 (Zhou et al., 2018)	scene classification	365	1,803,460	36,500

batch size 256[1], learning rate 0.01, momentum 0.9 and weight decay 0.0001. To avoid insufficient training and observe the complete convergence behavior, we use 300 epochs for fine-tuning and 600 epochs for scratch-training , which is long enough for the training curves to converge. The learning rate is decayed by a factor of 0.1 at epoch 150 and 250. We report the Top-1 validation error at the end of fine-tuning. The total computation time for the experiments is more than 10K GPU hours.

3.2 Effect of Momentum and Domain Similarity

Momentum 0.9 is the most widely adopted value for training from scratch (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; He et al., 2016b), and is also widely adopted in fine-tuning (Kornblith et al., 2018). To the best of our knowledge, it is rarely changed, regardless of the network architectures or target tasks. To check the influence of momentum on fine-tuning, we first search the best momentum values for fine-tuning on the Birds dataset with different batch size and weight decay. Figure 1(a) shows the performance of fine-tuning with or without weight decays. Surprisingly, momentum zero actually outperforms the nonzero momentum. We also noticed that the optimal learning rate increases when the momentum is disabled (Figure 1(b) and Appendix A).

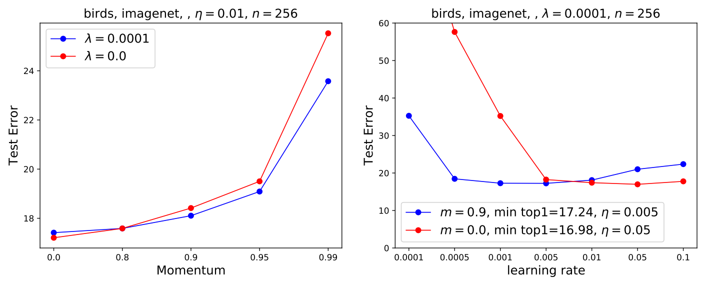


Figure 1: (a) Searching for the optimal momentum on Birds dataset with fixed learning rate 0.01 and different weight decays. Detailed learning curves and results of other hyperparameters can be found in Appendix A. (b) Comparison of momentum 0.9 and 0.0 with different learning rates on the Birds dataset. λ is fixed at 0.0001.

To verify this observation, we further compare momentum 0.9 and 0.0 on other datasets. Table 2 shows the performance of 8 hyperparameter settings on seven datasets. We find a clear pattern that disabling momentum works better for Dogs, Caltech, Indoor datasets, while momentum 0.9 works better for Cars, Aircrafts and Flowers.

Interestingly, datasets such as Dogs, Caltech, Indoor and Birds are known to have high overlap with ImageNet dataset[2], while Cars/Aircrafts are identified to be difficult to benefit from fine-tuning from

[1] For ResNet-101 and batch size 256, we use 8 NVIDIA Tesla V100 GPUs for synchronous training, where each GPU uses a batch of 32 and no SyncBN is used.

[2]Stanford Dogs Khosla et al. (2011) was built using images and annotation from ImageNet for the task of fine-grained image categorization. Caltech-256 has at least 200 categories exist in ImageNet (Deng et al., 2010). Images in the CUB-Birds dataset overlap with images in ImageNet.

Table 2: Top-1 errors on seven datasets by fine-tuning pre-trained ResNet-101 with different hyperparmeters. Each row represents a network fine-tuned by a set of hyperparameters (left four columns). The datasets are ranked by the relative improvement by disabling momentum. The lowest error rates with the same momentum are marked as bold. Note that the performance difference for Birds is not very significant.

m	n	η	λ	Dogs	Caltech	Indoor	Birds	Cars	Aircrafts	Flowers
0.9	256	0.01	0.0001	17.20	14.85	23.76	18.10	9.10	17.55	**3.12**
0.9	256	0.01	0	17.41	14.51	24.59	18.42	9.60	**17.40**	3.33
0.9	256	0.005	0.0001	14.14	13.42	24.59	17.24	**9.08**	18.21	3.50
0.9	256	0.005	0	14.80	13.67	22.79	17.54	9.31	17.82	3.53
0	256	0.01	0.0001	11.00	12.11	21.14	17.41	11.07	20.58	5.48
0	256	0.01	0	10.87	12.16	21.29	**17.21**	10.65	20.46	5.25
0	256	0.005	0.0001	10.21	11.86	21.96	18.24	13.22	24.39	7.03
0	256	0.005	0	**10.12**	**11.61**	**20.76**	18.40	13.11	23.91	6.78

Table 3: Verification of the momentum effect on similar source-target domains. The other hyperparameters are $n = 256$, $\eta = 0.01$, and $\lambda = 0.0001$. Momentum 0 works better for transferring from iNat-2017 to Birds and transferring from Places365 to Indoor-67 comparing to momentum 0.9 counterparts.

Source domain	m	Indoor	Birds	Dogs	Caltech	Cars	Aircrafts
iNat-2017	0.9	**30.73**	14.69	24.74	**20.12**	**11.16**	**19.86**
	0	34.11	**12.29**	**23.87**	21.47	16.89	27.21
Place-365	0.9	22.19	**27.72**	**30.84**	**22.53**	**11.06**	**21.27**
	0	**20.16**	32.17	32.47	22.60	14.67	25.29

pre-trained ImageNet models (Kornblith et al., 2018). According to Cui et al. (2018), in which the Earth Mover's Distance (EMD) is used to calculate the distance between a dataset with ImageNet, the similarity to Birds and Dogs are 0.562 and 0.620, while the similarity to Cars, Aircrafts and Flowers are 0.560 and 0.555, 0.525[3]. The relative order of similarity to ImageNet is

Dogs, Birds, Cars, Aircrafts and Flowers

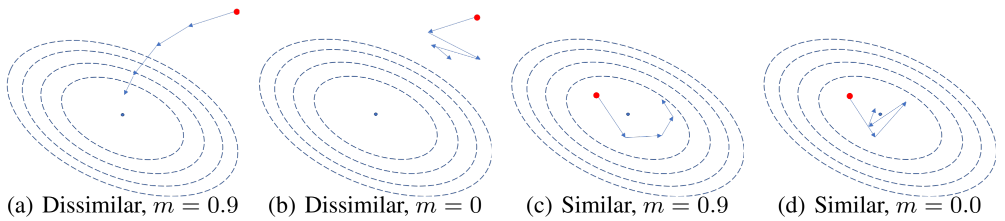
which aligns well with the transition of optimal momentum value from 0.0 to 0.9.

To verify this dependency on domain similarity, we fine-tune from pre-trained models of different source domains. It is reported that Place365 and iNaturalist are better source domains than ImageNet for fine-tuning on Indoor and Birds dataset (Cui et al., 2018). We can expect that fine-tuning from iNaturalist works well for Birds with $m = 0$ and similarly, Places365 for Indoor. Indeed, as shown in Table 3, disabling momentum improves the performance when the source and target domains are similar, such as Places for Indoor and iNaturalist for Birds.

Large momentum works better for fine-tuning on different domains but not for tasks that are close to source domains Our explanation for the above observations is that because the Dogs dataset is very close to ImageNet, the pre-trained ImageNet model is expected to be close to the fine-tuned solution on the Dogs dataset. In this case, momentum may not help much as the gradient direction around the minimum could be much random and accumulating the momentum direction could be meaningless. Whereas, for faraway target domains (e.g., Cars and Aircrafts) where the pre-trained ImageNet model could be much different with the fine-tuned solution, the fine-tuning process is more similar with training from scratch, where large momentum stabilizes the decent directions towards the minimum. An illustration of the difference can be found in Figure 2.

Connections to early observations on decreasing momentum Early work (Sutskever et al., 2013) actually pointed out that reducing momentum during the final stage of training allows finer convergence while aggressive momentum would prevent this. They recommended reducing momentum from 0.99 to 0.9 in the last 1000 parameter updates but not disabling it completely. Recent work (Liu

[3]The distance values are from Figure 5 in (Cui et al., 2018). The calculation process is illustrated in their Section 4.1



(a) Dissimilar, $m = 0.9$ (b) Dissimilar, $m = 0$ (c) Similar, $m = 0.9$ (d) Similar, $m = 0.0$

Figure 2: An illustration of the effect of momentum on different fine-tuning scenarios from the loss-landscape perspective. The red point is the the pre-trained model and the blue point is the final fine-tuned solution. The dashed lines are loss contours. In (a, b) where the initial solution is far from the optimal point, large momentum accelerates the training process. In (c, d) where the initialization is close to the solution, large momentum may impede the convergence. These figures assumes a fixed learning rate.

et al., 2018; Smith, 2018) showed that a large momentum helps escape saddle points but can hurt the final convergence within the neighborhood of the optima, implying that momentum should be reduced at the end of training. Liu et al. (2018) find that a larger momentum introduces higher variance of noise and encourages more exploration at the beginning of optimization, and encourages more aggressive exploitation at the end of training. They suggest that at the final stage of the step size annealing, momentum SGD should use a much smaller step size than that of vanilla SGD. When applied to fine-tuning, we can interpret that if the pre-trained model lies in the neighborhood of the optimal solution on the target dataset, the momentum should be small. Our work identifies the empirical evidence of disabling momentum helps final convergence, and fine-tuning on close domains seems to be a perfect case.

3.3 Coupled Hyperparameters and the View of Effective Learning Rate

Now that we had studied the effectiveness of momentum by fixing other hyperparaemters and only allow momentum to change. But note that the two difficult scenarios faced in Figure 2 (b) and (c) can be mitigated by increasing learning or decreasing learning rate. That is, hyperparameters are coupled and varying one hyperparameter can change the optimal values of the other hyperparameters that lead to the best performance. Optimal values of neural network hyperparameters depend on the values of other hyperparameters in systematic ways. For example, learning rate is entangled with batch size, momentum and weight decay. Smith et al. (2018) interpret SGD as integrating a stochastic differential equation and show that the scale of random fluctuations in the SGD dynamics, $g = \eta(\frac{N}{B} - 1)$, where B is the batch size. The notion of *effective learning rate* (ELR) (Hertz et al., 1991; Smith & Le, 2018) for SGD with momentum is follows:

$$\eta' = \eta/(1 - m) \tag{3}$$

which was shown to be more closely related with training dynamics and final performance rather than η (Smith et al., 2018; Smith & Le, 2018). The effective learning rate with $m = 0.9$ is $10\times$ higher than the one with $m = 0.0$ if other hyperparameters are fixed, which is probably why we see an increase in optimal learning rate when momentum is disabled in Figure 1(b) and Appendix A.

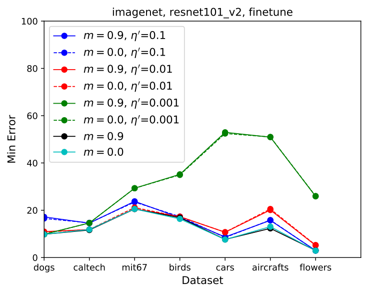
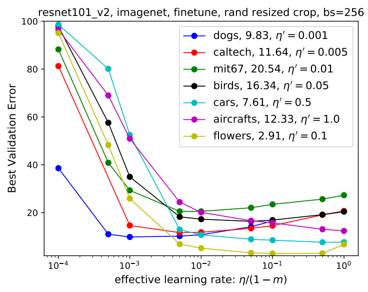
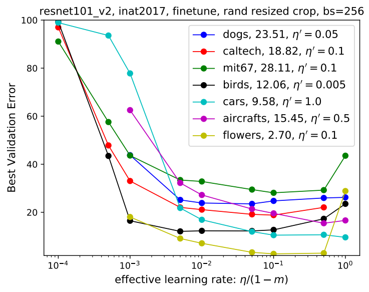


Figure 3: Test errors obtained by different momentum with fixed effective learning rate η'. It shows that when η' is the same, momentum = 0 and 0.9 are almost equivalent. When η' is allowed to change, there is almost no difference between momentum 0.9 and 0.

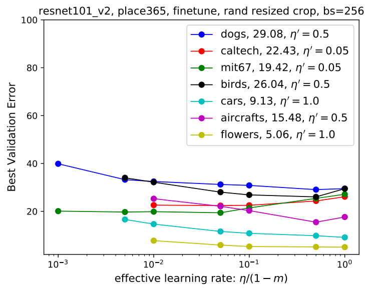
Because learning rate and momentum are coupled, looking at the performance with only one hyperparameter varied can give a misleading understanding of the effect of hyperparameters. Therefore, we report the best result with and without momentum. which does not affect the maximum accuracy obtainable with and without momentum, as long as the hyperparameters explored are sufficiently close to the optimal parameters. We review previous experiments that demonstrated the importance of momentum tuning when the effective learning rate $\eta' = \eta/(1-m)$ is held fixed instead of the learning rate η. Figure 3 shows that when η' is constant, momentum 0.0 and 0.9 are actually equivalent. In addition, the best performance obtained by momentum 0.9 and momentum 0 is equivalent when other



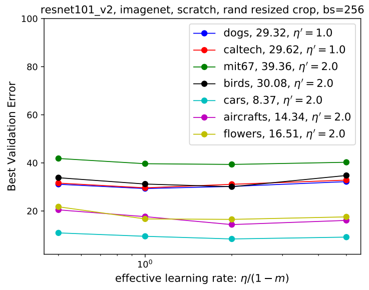
(a) ImageNet



(b) iNat2017



(c) Place365



(d) Training from scratch

Figure 4: The relationship between optimal ELR and source datasets. (a, b, c) show the best test errors obtained by different ELRs while allowing other hyper-parameters to change. The title of each sub-figure describes the source domain datasets. The optimal ELR for each target dataset are in the interior of the search space. Note that the best ELR for each target dataset changes when the source domain is different, e.g., fine-tuning from ImageNet require a small ELR for Dogs but requires a larger one when fine-tuned from iNat2017 and Places365. (d) shows that the optimal ELRs for training from scratch of each dataset are very similar (1.0 or 10.0).

hyperparameters are allowed to change. However, different effective learning rates results in different performance, which indicates that it is effective learning rate that matters for the best performance. It explains why the common practice of changing only learning rate generally works, though changing momentum may results in the same effect. They both change the effective learning rate.

Optimal effective learning rate and weight decay depend on the similarity between source domain and target domain. Now that we have shown ELR is critical for the performance of fine-tuning, we are interested in the factors that determine the optimal ELR affected. Smith & Le (2018) found that there is an optimum fluctuation scale which maximizes the test set accuracy (at constant learning rate). However, the relationship between ELR and domain distance is unknown, which is important for fine-tuning. Effective learning rate encapsulates the effect of learning rate and momentum for fine-tuning. We varied other hyperparameters and report the best performance for each η'. As shown in Figure 4, a smaller η' works better if source and target domains are similar, such as Dogs for ImageNet and Birds for iNaturalist. On the other hand, the ELR for training from scratch is large and relative stable.

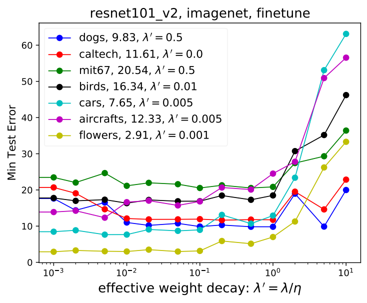


Figure 5: The relationship between optimal effective weight decay and source datasets. The optimal effective weight decay is smaller when the source domain is similar with the target domain..

The relationship between weight decay and effective learning rate are also well-studied (Loshchilov & Hutter, 2018; van Laarhoven; Zhang et al., 2018). It was shown that the optimal weight decay value λ is inversely related with learning rate η. The ‘effective‘ weight decay is $\lambda' = \lambda/\eta$. We show in Figure 5 that the optimal effective weight decay is larger when the source domain is similar with the target domain.

3.4 The choice of Regularization

L_2 regularization or weight decay is widely used for constraining the model capacity (Hanson & Pratt, 1989; Krogh & Hertz, 1992). Recent work (Li et al., 2018; 2019) pointed that standard L_2 regularization, which drives the parameters towards the origin, is not adequate in transfer learning. To retain the knowledge learned by the pre-trained model, reference based regularization was used to regularize the distance between fine-tuned weights and the pre-trained weights, so that the fine-tuned model is not too different from the initial model. Li et al. (2018) propose L_2-SP norm, i.e., $\frac{\lambda_1}{2}\|\theta' - \theta_0\|_2^2 + \frac{\lambda_2}{2}\|\theta''\|_2^2$, where θ' refers to the part of network that shared with the source network, and θ'' refers to the novel part, e.g., the last layer with different number of neurons.

While the motivation is intuitive, there are several issues for adopting reference based regularization for fine-tuning: (1) Many applications actually adopt fine-tuning on target domains that are quite different from source domain, such as fine-tuning ImageNet models for medical imaging (Mormont et al., 2018; Raghu et al., 2019). The fine-tuned model does not necessarily has to be close with the initial model. (2) The scale invariance introduced by Batch Normalization (BN) (Ioffe & Szegedy, 2015) layers enables models with different parameter scales to function the same, i.e., $f(\theta) = f(\alpha\theta)$. Therefore, when L_2 regularization drives $\|\theta\|_2^2$ towards zeros, it could still have the same functionality as the initial model. On the contrary, a model could still be different even when the L_2-SP norm is small. (3) It has been shown that the effect of weight decay on models with BN layers is equivalent to increasing the effective learning rate by shrinking the weights scales (van Laarhoven; Zhang et al., 2018). Regularizing weights with L_2-SP norm would constrain the scale of weights to be close to the original one, therefore not increasing the effective learning rate, during fine-tuning. As a small effective learning rate is beneficial for fine-tuning from similar domains, which may explain why L_2-SP provides better performance. If this is true, then by decreasing the effective learning rate, L_2 regularization would functions the same.

To examine these conjectures, we revisited the work of (Li et al., 2018) with additional experiments. To show the effectiveness of L_2-SP norm, Li et al. (2018) conducted experiments on datasets such as Dogs, Caltech and Indoor, which are all datasets close to the source domain (ImageNet or Place-365) according to previous sections. We extend their experiments on other datasets that are relatively "far" away from ImageNet, such as Birds, Cars, Aircrafts and Flowers. We use the source code of Li et al. (2018) to fine-tune on these datasets with both L_2 and L_2-SP regularization. For fair comparison, we performed the same hyperparameter search for both methods (detailed experimental setting in Appendix C). As expected, Table 4 shows that L_2 regularization is very close to if not better than L_2-SP on Birds, Cars, Aircrafts and Flowers, which indicates that reference based regularization methods may not be able to generalize for fine-tuning on dissimilar domains.

Table 4: The average class error of (Li et al., 2018) and the extension of their experiments of on "dissimilar" datasets. The *italic* datasets and numbers are our experiments results.

Method	Dogs	Caltech	Indoor	*Birds*	*Cars*	*Flowers*	*Aircrafts*
L_2 (Li et al., 2018)	18.6	14.7	20.4	*22.51*	*10.10*	*5.70*	***13.03***
L_2-SP (Li et al., 2018)	**14.9**	**13.6**	**15.8**	***22.32***	***9.69***	***5.28***	*13.31*

4 Discussion

The two extreme ways for selecting hyperparameters—performing exhaustive hyperparameter search or taking ad-hoc hyperparameters from scratch training—could be either too computationally expensive or yield inferior performance. Different with training from scratch, the default hyperparameter setting may work well for random initialization, the choice of hyperparameters for fine-tuning is not only dataset dependent but is also influenced by the similarity between the target domain and the source domains. The rarely tuned momentum value could impede the performance when the target domain and source domain are close. These observations connect with previous theoretical works on decreasing momentum at the end of training and effective learning rate. We further identify the optimal effective learning rate depends on the similarity of source domain and target domain. With this understanding, one can significant reduce the hyperparameter search space. We hope these findings could be one step towards better hyperparameter selection strategies for fine-tuning.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. *arXiv preprint arXiv:1902.03545*, 2019.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE T-PAMI*, 40(4):834–848, 2017.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 2018.

Jia Deng, Alexander C Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, pp. 71–84. Springer, 2010.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

Weifeng Ge and Yizhou Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *CPVR*, 2017.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Gabriel Goh. Why momentum really works. *Distill*, 2(4):e6, 2017.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

Stephen Jose Hanson and Lorien Y. Pratt. Comparing biases for minimal network construction with back-propagation. In D. S. Touretzky (ed.), *NIPS*, pp. 177–185. Morgan-Kaufmann, 1989.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016b.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *ICCV*, 2019.

John Hertz, A Krogh, and Richard G Palmer. Introduction to the theory of neural computation. 1991.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pp. 4700–4708, 2017.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pp. 448–456, 2015.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *NIPS*, pp. 950–957, 1992.

Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, and Jun Huan. Delta: Deep learning transfer using feature map with attention for convolutional networks. In *ICLR*, 2019.

Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. *ICML*, 2018.

Tianyi Liu, Zhehui Chen, Enlu Zhou, and Tuo Zhao. Toward deeper understanding of nonconvex stochastic optimization with momentum using diffusion approximations. *arXiv preprint arXiv:1802.05155*, 2018.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

Romain Mormont, Pierre Geurts, and Raphaël Marée. Comparison of deep transfer learning strategies for digital pathology. In *CVPR Workshops*, pp. 2262–2271, 2018.

Yurixi E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V Le, and Ruoming Pang. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*, 2018.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, 2008.

Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning with applications to medical imaging. *arXiv preprint arXiv:1902.07208*, 2019.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

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. *IJCV*, 115(3):211–252, 2015.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR workshops*, pp. 806–813, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Leslie N Smith. Cyclical learning rates for training neural networks. In *WACV*, pp. 464–472. IEEE, 2017.

Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 1100612. International Society for Optics and Photonics, 2019.

Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. In *ICLR*, 2018.

Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. In *ICLR*, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. In *NIPS*.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Junyuan Xie, Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, and Mu Li. Bag of tricks for image classification with convolutional neural networks. *arXiv preprint arXiv:1812.01187*, 2018.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

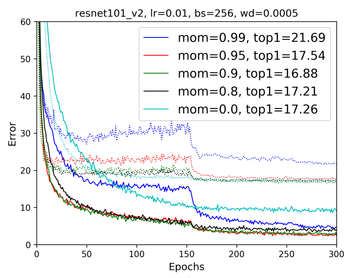
Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*, 2018.

Jian Zhang and Ioannis Mitliagkas. Yellowfin and the art of momentum tuning. *arXiv preprint arXiv:1706.03471*, 2017.

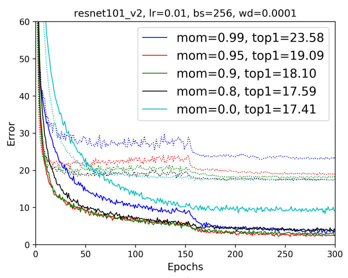
Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE T-PAMI*, 40(6):1452–1464, 2018.

A Search Optimal Momentum on Birds

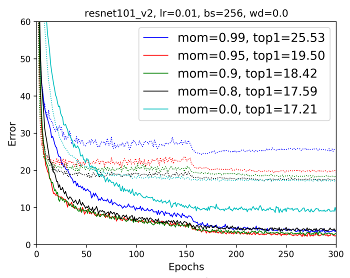
To check the influence of momentum on fine-tuning, we first search the best momentum values for fine-tuning on the Birds dataset with different batch size and weight decay. Figure 6 provides the convergence curves for the results shown in Figure 1(a), which shows the learning curves of fine-tuning with 6 different batch sizes and weight decay combinations. Zero momentum outperforms the nonzero momentum in 5 of the 6 configurations.



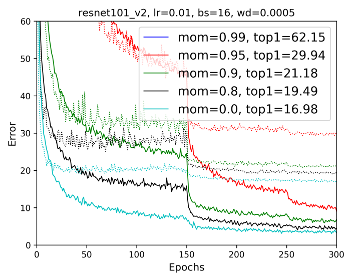
(a) $n = 256, \lambda = 0.0005$



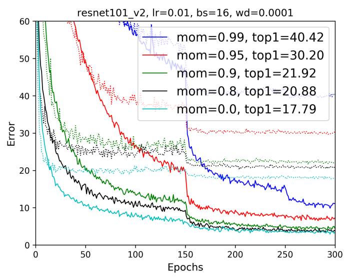
(b) $n = 256, \lambda = 0.0001$



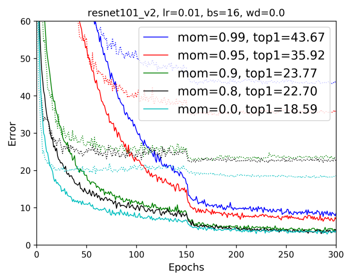
(c) $n = 256, \lambda = 0.0$



(d) $n = 16, \lambda = 0.0005$



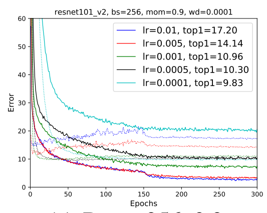
(e) $n = 16, \lambda = 0.0001$



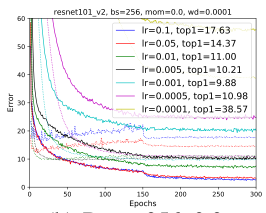
(f) $n = 16, \lambda = 0.0$

Figure 6: Searching for the optimal momentum on Birds dataset with fixed learning rate 0.01 and different weight decays. The solid lines are training errors and the dashed lines are validation errors.

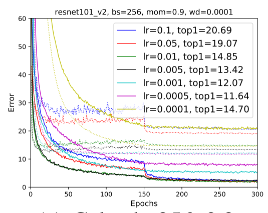
Optimal learning rate increases after disabling momentum. Figure 7 compares the performance of turning on/off momentum for each datasets with different learning. For datasets that are “similar" to ImageNet (Figure 7 (a-h)) and fixed learning rate (e.g., 0.01), the Top-1 validation error decreases significantly after disabling momentum. On the other hand, for datasets that are “dissimilar" to ImageNet (Figure 7(g-n)) and fixed learning rate, disabling momentum hurts the top-1 accuracy. We can also observe that the optimal learning rate generally increase 10x after changing from 0.9 to 0.0, which is coherrent with the rule of effective learning rate.



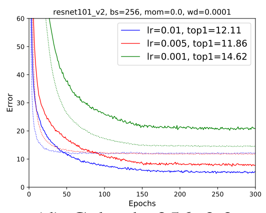
(a) Dogs, 256, 0.9



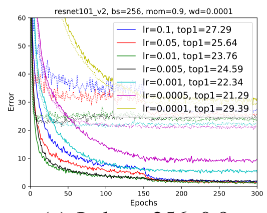
(b) Dogs, 256, 0.0



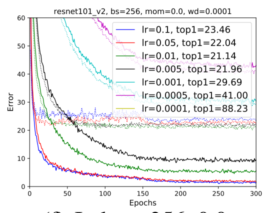
(c) Caltech, 256, 0.9



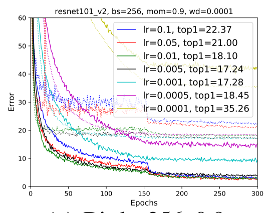
(d) Caltech, 256, 0.0



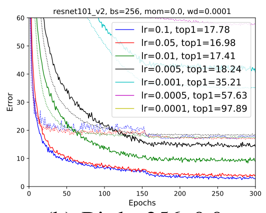
(e) Indoor, 256, 0.9



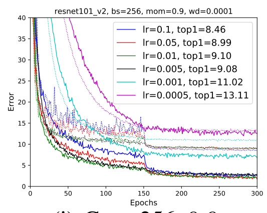
(f) Indoor, 256, 0.0



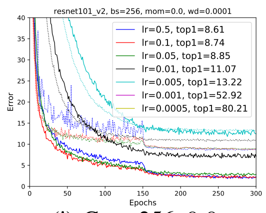
(g) Birds, 256, 0.9



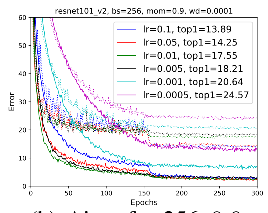
(h) Birds, 256, 0.0



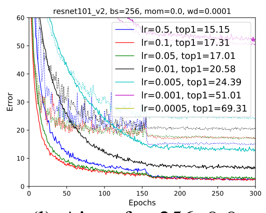
(i) Cars, 256, 0.9



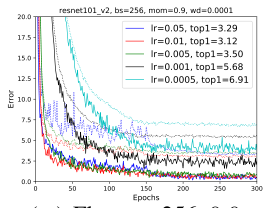
(j) Cars, 256, 0.0



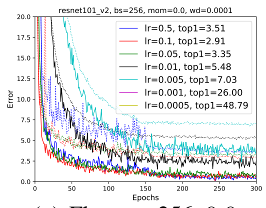
(k) Aircrafts, 256, 0.9



(l) Aircrafts, 256, 0.0



(m) Flowers, 256, 0.9

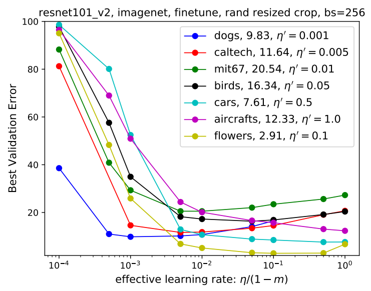


(n) Flowers, 256, 0.0

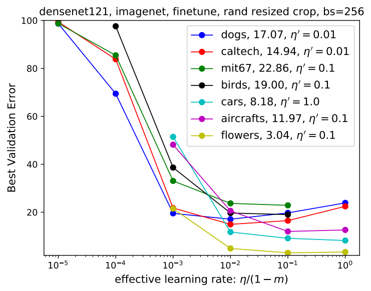
Figure 7: (a-h) Fine-tuning without momentum works better for datasets that are "similar" to ImageNet. The subtitle of each figure contains dataset name, batch size and momentum. (g-n) Fine-tuning with momentum works better for tasks that are dissimilar with ImageNet.

B Verfiication on Different Architectures

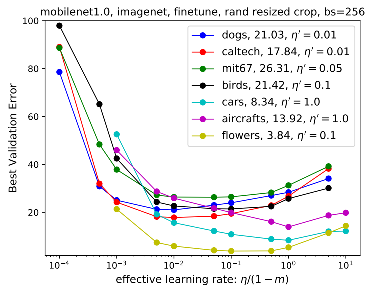
We also verified our observations on DenseNet-121 (Huang et al., 2017) and MobileNet-1.0 (Howard et al., 2017) with smilar settings. As seen in Figure 8 (b) and (c), the optimal effective learning rates for Dogs/Caltech/Indoor datasets are much smaller than these for Aircrafts/Flowers/Cars when fine-tuned from ImageNet, similar with ResNet-101 in Figure 9(a). On the other hand, Figure 8 (d) shows the optimal effective learning rate for fine-tuning birds from iNaturalist is much smaller than pre-training from ImageNet. This shows that our claims are valid for a variety of architectures and datasets.



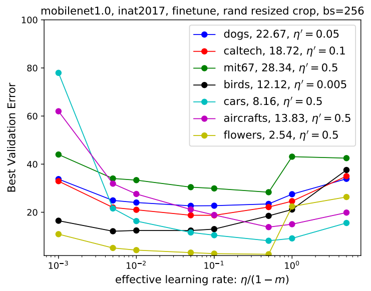
(a) ResNet, ImageNet



(b) DenseNet, ImageNet



(c) MobileNet, ImageNet



(d) MobileNet, iNaturalist

Figure 8: The performance of different architectures with different effective learning rates. The title of each subfigure describes the architecture and source domain dataset.

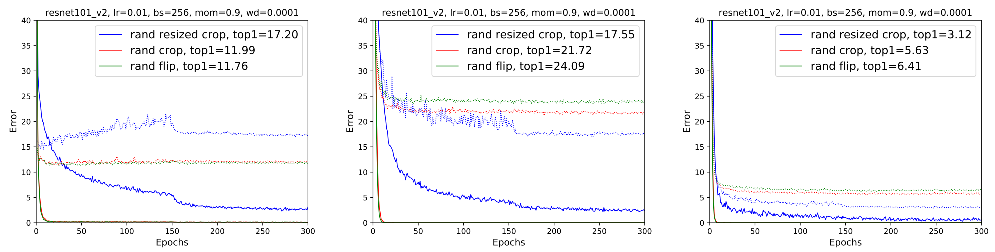
C Experimental Settings for the comparison between L_2 and L_2-SP

We use the code[4] provided by the authors. The base network is pretrained ResNet-101-V1. The model is fine-tuned with batch size 64 in 9000 iterations, and the learning rate is decayed at iteration 6000. Following the original setting, we use momentum 0.9. We performed grid search on learning rate and weight decay, with the range of $\eta : \{0.02, 0.01, 0.005, 0.001, 0.0001\}$ and $\lambda_1 : \{0.1, 0.01, 0.001, 0.0001\}$, and report the best average error for both methods. For $L2$-SP norm, we follow the authors setting to use constant $\lambda_2 = 0.01$. Different with the original setting for L_2 regularization, we set $\lambda_2 = \lambda_1$ to simulate the normal L_2-norm.

[4] `https://github.com/holyseven/TransferLearningClassification`

D Data Augmentation

Data augmentation is an important way of increasing data quantity and diversity to make models more robust. It is even critical for transfer learning with few instances. The effect of data augmentation can be viewed as a regularization method and the choice of data augmentation method is also a hyperparameter. Most current widely used data augmentation methods have verified their effectiveness on training ImageNet models, such as random mirror flipping, random rescaled cropping[5], color jittering and etc Szegedy et al. (2015); Xie et al. (2018) and they are also widely used for fine-tuning. Do these methods transfer for fine-tuning on other datasets? Here we compare three settings for data augmentation: 1) random resized cropping: our *default* data augmentation; 2) random crop: the same as *standard* data augmentation except that we use random cropping with fixed size; 3) random flip: simply random horizontal flipping.



(a) Dogs (b) Aircrafts (c) Flowers

Figure 9: Fine-tuning with default hyperparameters but different data augmentation methods. Dashed curves are the validation errors. Strong data augmentation is harder to train as it converge slowly and needs more number of epochs to observe the advanced performance on datasets such as Aircrafts. Simple data augmentation usually converge in 10s of epochs.

The effect of data augmentation is dataset dependent and has big impact on the convergence time The training and validation errors of fine-tuning with different data augmentation strategies are illustrated in Figure 9. We find that advanced cropping works significantly better on datasets like Cars, Aircrafts and Flowers but performs worse on Dogs. The choice of data augmentation methods has dramatic influence to the convergence behaviour. Simpler data augmentation usually converge very quickly (e.g., in 20 epochs), while the training error for random resized cropping converges much slower. We see that default hyperparemter and data augmentation method lead to overfitting on Dogs dataset. This can be solved by disabling momentum as we can see in Table 2, and result in better performance than random cropping. We can expect that random resized cropping adds extra variance to the gradient direction and the effect of disabling momentum is more obvious for this case.

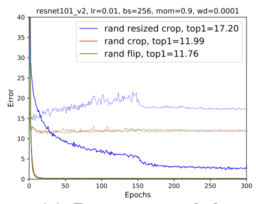
Disabling momentum improves performance on datasets that are close to source domains Here we compare data augmentation methods with different momentum settings. As can be seen in Table 5, random resized cropping consistently outperforms random cropping on datasets like Cars, Aircrafts and Flowers. Using momentum improves the performance significantly for both methods.

We see that advanced data augmentation method with default hyperparameters ($m = 0.9$ and $\eta = 0.01$) leads to overfitting on Dogs and Caltech dataset (Figure 10 (a) and (c)). Random resized cropping with zero momentum solves this problem and results in better performance than random cropping. When momentum is disabled for random cropping, the performance is still better for Dogs, but decreases for other datasets. This can be expected as random cropping produces images with less variation and noise than random resized cropping, the gradients variation is less random and momentum can still point to the right direction. This can be further verified as we increase the learning rate for random cropping, which adds variation to the gradients, and disabling momentum shows better performance that nonzero momentum on datasets that are close.

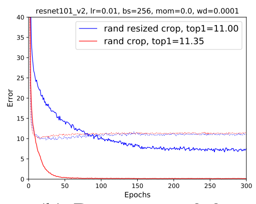
[5]Randomly crop a rectangular region with aspect ratio randomly sampled in [3/4, 4/3] and area randomly sampled in [8%, 100%] (Szegedy et al., 2015)

Table 5: Comparison of data augmentation methods with different momentum values. The rest of the hyperparameters are: $n = 256$ and $\lambda = 0.0001$.

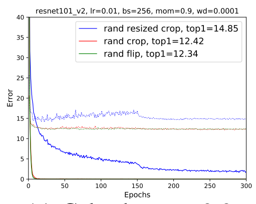
Data Augmentation	m	η	Dogs	Caltech	Indoor	Birds	Cars	Flowers	Aircrafts
Rand resized crop	0.9	0.01	17.20	14.85	23.76	18.10	**9.10**	**3.12**	**17.55**
	0	0.01	**11.00**	**12.11**	**21.14**	**17.41**	11.06	5.48	20.58
Rand crop	0.9	0.01	11.99	**12.42**	**23.39**	**20.31**	**17.77**	**5.63**	**21.72**
	0	0.01	**11.35**	12.89	25.19	22.11	23.87	7.76	29.04
	0.9	0.05	16.85	14.80	23.46	**18.81**	**13.70**	**4.85**	**17.64**
	0	0.05	**11.79**	**12.52**	**23.24**	20.69	20.00	7.06	23.43



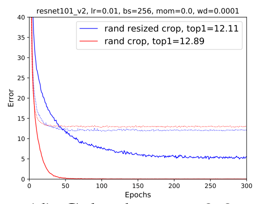
(a) Dogs, $m = 0.9$



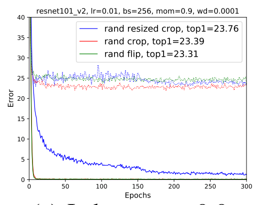
(b) Dogs, $m = 0.0$



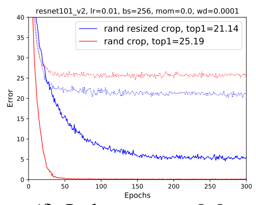
(c) Caltech, $m = 0.9$



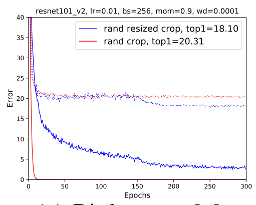
(d) Caltech, $m = 0.0$



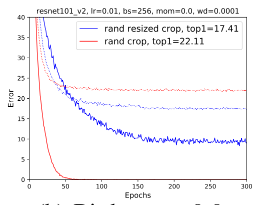
(e) Indoor, $m = 0.9$



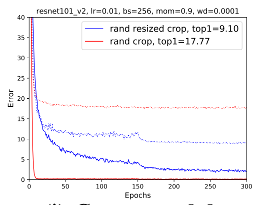
(f) Indoor, $m = 0.0$



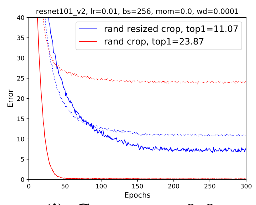
(g) Birds, $m = 0.9$



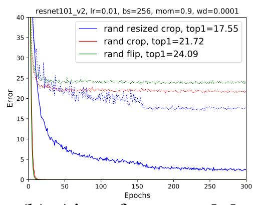
(h) Birds, $m = 0.0$



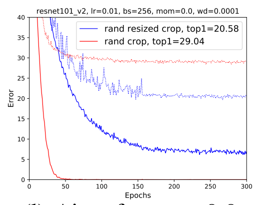
(i) Cars, $m = 0.9$



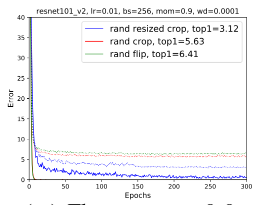
(j) Cars, $m = 0.0$



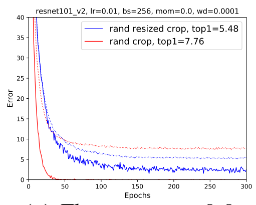
(k) Aircrafts, $m = 0.9$



(l) Aircrafts, $m = 0.0$



(m) Flowers,$m = 0.9$



(n) Flowers, $m = 0.0$

Figure 10: Comparison of data augmentation methods with different momentum values. The rest of the hyperparameters are: $n = 256$, $\eta = 0.01$ and $\lambda = 0.0001$.

E Source Domains

Transfer learning from similar source domains helps but does not guarantee good performance We consider two ImageNet subsets: 449 Natural objects and 551 Man-made objects, following the splits of (Yosinski et al., 2014) (supplementary materials). From the bottom of Table 6, we can see that fine-tuning from ImageNet-Natural pre-trained models performs better on Birds and Dogs dataset, whereas Caltech-256 and Indoor benefit more from ImageNet-Manmade pretrained models. The performance gap between ImageNet-Manmad and ImageNet-Natural on Cars and Flowers are not as significant as for Birds and Dogs. It is surprising to see that fine-tuning from ImageNet-Manmade subset yields worse performance than ImageNet-Natural on the Cars and Indoor dataset. The fine-tuning results on both subsets do not exceed the pre-trained models with full ImageNet.

Scratch training can outperform fine-tuning with better hyperparameters We further re-examine the default hyperparameters for scratch training. For most tasks, training from scratch with default hyperparameters is much worse than fine-tuning from ImageNet. However, after slight hyperparameter tuning on learning rates, momentum and weight decay, the performance of training from scratch gets close to the default fine-tuning result (e.g., Cars and Aircrafts). Scratch training HPO on Cars and Aircrafts even surpasses the default fine-tuning result. Previous studies Kornblith et al. (2018); Cui et al. (2018) also identified that datasets like Cars, Aircrafts do not benefit too much from fine-tuning.

Table 6: Fine-tuning results from different source domains. Comparison to existing fine-tuning methods. FT ImageNet Default is the fine-tuning result with the *default* hyperparameters. Scratch Train use similar hyperparameters as *default* fine-tuning, which are $\eta = 0.1$, $n = 256$, $\lambda = 0.0001$ and $m = 0.9$ with doubled length of training schedules. HPO refers to the best results with hyperparameter grid search. Note our Indoor dataset result is fine-tuned from ImageNet. Results of ResNet-101 DELTA refers to (Li et al., 2019), and Inception-v3 refers to (Cui et al., 2018).

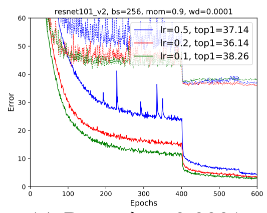
Method	Birds	Dogs	Cars	Flowers	Aircrafts	Caltech	Indoor
ResNet-101 DELTA	19.5	11.3	-	-	-	11.3	-
Inception-v3 299	17.16	15.81	8.69	3.74	14.51	-	-
Inception-v3 iNat 299	10.74	21.54	11.69	2.36	17.38	-	-
FT ImageNet Default	18.10	17.20	9.10	3.12	17.55	13.42	23.76
FT ImageNet HPO	**16.62**	**9.83**	**7.66**	**2.63**	**14.25**	**11.61**	**20.76**
FT ImageNet-Nat Default	21.25	17.74	11.03	4.59	22.77	22.88	33.88
FT ImageNet-Man Default	29.43	32.52	13.48	4.70	20.46	18.90	25.26
Scratch Train Default	43.72	38.26	16.73	22.88	26.49	36.21	45.28
Scratch Train HPO	30.08	29.32	8.37	16.51	14.34	29.62	39.36

Pre-trained models on ImageNet-Natural and ImageNet-Manmade We train ResNet-101 from scratch on each subset using standard hyperparameters, i.e., initial learning rate 0.1, batch size 256, momentum 0.9. We train 180 epochs, learning rate is decayed at epoch 60 and 120 by a factor of 10. Table 7 illustrates the Top-1 errors of training ResNet-101 on each source datasets.

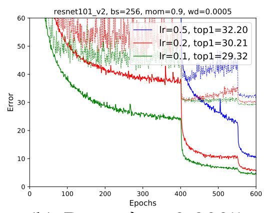
Table 7: The performance of ResNet-101 trained on subsets of ImageNet.

Dataset	class	Top-1 error
ImageNet	1000	21.4
ImageNet-Natural	551	17.6
ImageNet-Manmade	449	27.5

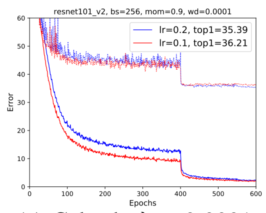
Scratch Training HPO Figure 11 shows the training/validation errors of training from scratch on each dataset with different learning rate and weight decay. We use initial learning rate 0.1, batch size 256. For most dataset, we train 600 epochs, and decay the learning rate at epoch 400 and 550 by a factor of 10. The parameters to search is $\eta \in [0.1, 0.2, 0.5]$ and $\lambda \in [0.0001, 0.0005]$ with fixed momentum 0.9 and batch size 256. We observe weight decay 0.0005 consistently performs better than 0.0001.



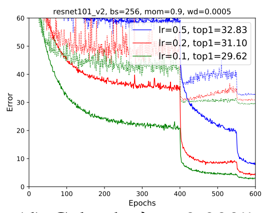
(a) Dogs, $\lambda = 0.0001$



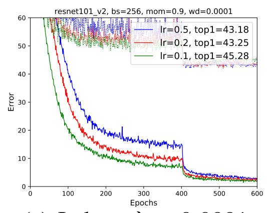
(b) Dogs, $\lambda = 0.0005$



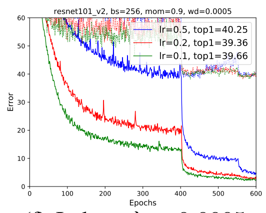
(c) Caltech, $\lambda = 0.0001$



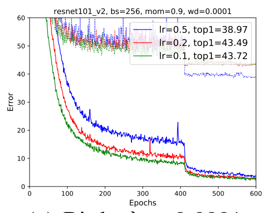
(d) Caltech, $\lambda = 0.0005$



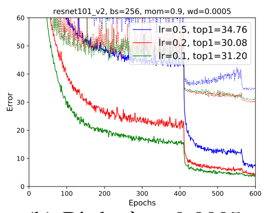
(e) Indoor, $\lambda = 0.0001$



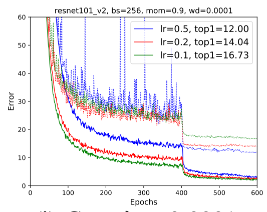
(f) Indoor, $\lambda = 0.0005$



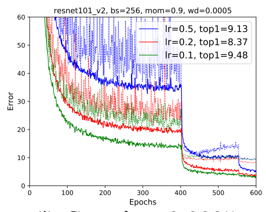
(g) Birds, $\lambda = 0.0001$



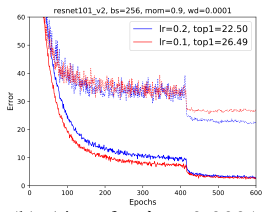
(h) Birds, $\lambda = 0.0005$



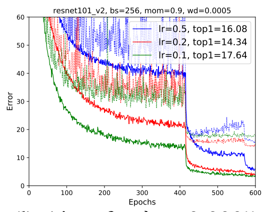
(i) Cars, $\lambda = 0.0001$



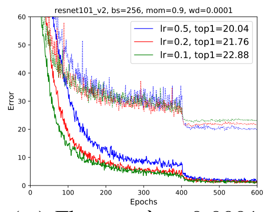
(j) Cars, $\lambda = 0.0005$



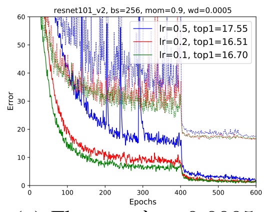
(k) Aircrafts, $\lambda = 0.0001$



(l) Aircrafts, $\lambda = 0.0005$



(m) Flowers, $\lambda = 0.0001$



(n) Flowers, $\lambda = 0.0005$

Figure 11: Training from scratch with different learning rates and weight decays. The batch size is 256 and the momentum is 0.9.