MIX & MATCH: TRAINING CONVNETS WITH MIXED IMAGE SIZES FOR IMPROVED ACCURACY, SPEED AND SCALE RESILIENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional neural networks (CNNs) are commonly trained using a fixed spatial image size predetermined for a given model. Although trained on images of a specific size, it is well established that CNNs can be used to evaluate a wide range of image sizes at test time, by adjusting the size of intermediate feature maps.
In this work, we describe and evaluate a novel mixed-size training regime that mixes several image sizes at training time. We demonstrate that models trained using our method are more resilient to image size changes and generalize well even on small images. This allows faster inference by using smaller images at test time. For instance, we receive a 76.43% top-1 accuracy using ResNet50 with an image size of 160, which matches the accuracy of the baseline model with $2\times$ fewer computations. Furthermore, for a given image size used at test time, we show this method can be exploited either to accelerate training or the final test accuracy. For example, we are able to reach a 79.27% accuracy with a model evaluated at a 288 spatial size for a relative improvement of 14% over the baseline.
Our PyTorch implementation and pre-trained models are publicly available[1]

1 INTRODUCTION

Convolutional neural networks are successfully used to solve various tasks across multiple domains such as visual (Krizhevsky et al., 2012; Ren et al., 2015), audio (van den Oord et al., 2016), language (Gehring et al., 2017) and speech (Abdel-Hamid et al., 2014). While scale-invariance is considered important for visual representations (Lowe, 1999), convolutional networks are not scale invariant with respect to the spatial resolution of the image input, as a change in image dimension may lead to a non-linear change of their output. Even though CNNs are able to achieve state-of-the-art results in many tasks and domains, their sensitivity to the image size is an inherent deficiency that limits practical use cases and requires evaluation inputs to match training image size. For example, Touvron et al. (2019) demonstrated that networks trained on specific image size, perform poorly on other image sizes at evaluation, as shown in Figure 1.

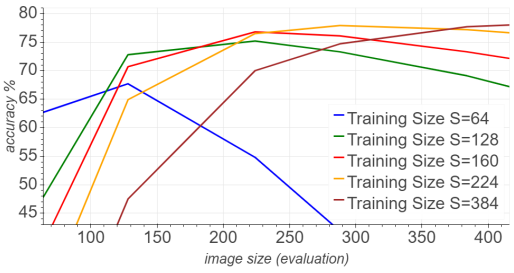


Figure 1: Test accuracy per image size, models trained on specific sizes (ResNet50, ImageNet).

Several works attempted to achieve scale invariance by modifying the network structure (Xu et al., 2014; Takahashi et al., 2017). However, the most common method is to artificially enlarge the dataset using a set of label-preserving transformations also known as "data augmentation" (Krizhevsky et al., 2012; Howard, 2013). Several of these transformations scale and crop objects appearing within the data, thus increasing the network's robustness to inputs of different scale.

Although not explicitly trained to handle varying image sizes, CNNs are commonly evaluated on multiple scales post training, such as in the case of detection (Lin et al., 2017; Redmon & Farhadi,

[1] https://github.com/paper-submissions/mix-match

2018) and segmentation (He et al., 2017) tasks. In these tasks, a network that was pretrained with fixed image size for classification is used as the backbone of a larger model that is expected to adapt to a wide variety of image sizes.

In this work, we will introduce a novel training regime, "MixSize" for convolutional networks that uses stochastic image and batch sizes. The main contributions of the MixSize regime are:

- **Reducing image size sensitivity**. We show that the MixSize training regime can improve model performance on a wide range of sizes used at evaluation.
- **Faster inference**. As our mixed-size models can be evaluated at smaller image sizes, we show up to $2\times$ reduction in computations required at inference to reach the same accuracy as the baseline model.
- **Faster training vs. high accuracy**. We show that reducing the average image size at training leads to a trade-off between the time required to train the model and its final accuracy.

2 Related work

2.1 Using multiple image sizes

Deep convolutional networks are traditionally trained using fixed-size inputs, with spatial dimensions $H \times W$ and a batch size B. The network architecture is configured such that the spatial dimensions are reduced through strided pooling or convolutions, with the last classification layer applied on a 1×1 spatial dimension. Modern convolutional networks usually conclude with a final "global" average pooling (Lin et al., 2013; Szegedy et al., 2015), that reduces any remaining spatial dimensions with a simple averaging operation. Modifying the spatial size of an input to a convolutional layer by a factor γ, will yield an output with size scaled by the same factor γ. This modification does not require any change to the number of parameters of the given convolutional layer, nor its underlying operation. Small changes in the expected size can occur, however, due to padding or strides performed by the layer. It was observed by practitioners and previous works that a network trained on a specific input dimension can still be used at inference using a modified image size to some extent (Simonyan & Zisserman, 2014). Moreover, evaluating with an image size that is larger than used for training can improve accuracy up to a threshold, after which it quickly deteriorates (Touvron et al., 2019).

Recently, Tan & Le (2019) showed a computational-vs-accuracy trade-off in scaling image size used to train and evaluate with a convolutional network. This finding is consistent with past findings, which demonstrated that training with a larger image size can result in a larger classification error (Szegedy et al., 2016; Huang et al., 2018). In addition, previous works explored the notion of "progressive resizing" (Karras et al., 2017; Howard, 2018) — increasing image size as training progresses to improve model performance and time to convergence. More recently, Touvron et al. (2019) demonstrated that CNNs can be trained using a fixed small image size and fine-tuned post-training to a larger size, with which evaluation will be performed. This procedure reduced the train-test discrepancy caused by the change in image size and allowed faster training time and improved accuracy — at the cost of additional fine-tuning procedure and additional computations at inference time.

In this work we will further explore the notion of using multiple image sizes at training, so the CNN performance will be resilient to test time changes in the image size.

2.2 Large batch training of deep networks

Deep neural network training can be distributed across many computational units and devices. The most common distribution method is by "data-parallelism"—computing an average estimate of the gradients using multiple, separably computed data samples. As training NN models is done using batch-SGD method and its variants, scaling this process across more computational devices while maintaining similar utilization for each device inflates the global batch size.

Large batch training is known to affect the generalization capabilities of the networks and to require modification of the regime used for its optimization. While several works claimed that large-batch

training leads to an inherent "generalization gap" (Keskar et al., 2016), more recent works demonstrated that this gap is largely caused from an insufficient number of optimization steps performed and can be partly mitigated by hyper-parameter tuning (Hoffer et al., 2017; Shallue et al., 2018). In order to cope with the changes in the training dynamics of the network, several modifications to the optimization procedure have been proposed such as a linear (Goyal et al., 2017) or a square-root (Hoffer et al., 2017) scaling of the learning rate with respect to the batch size growth. Other modifications include per-layer gradient scaling schemes (You et al., 2017) and optimizer modifications (Ginsburg et al., 2019). Several works also explored using incremented batch-sizes (Smith et al., 2018) in order to decrease the number of training iterations required to reach the desired accuracy.

Recent work by Hoffer et al. (2019) introduced the notion of "Batch Augmentation" (BA)—increasing the batch size by augmenting several instances of each sample within the same batch. BA aids generalization across a wide variety of models and tasks, with the expense of an increased computational effort per step. A similar method called "Repeated Augmentation" (RA) was proposed by Berman et al. (2019). It was also demonstrated that BA may allow to decrease the number of training steps needed to achieve a similar accuracy and also mitigate I/O throughput bottlenecks (Choi et al., 2019). As previous works investigated mostly homogeneous training settings (e.g., using a fixed batch size), an open question still exists on the utility of rapidly varying batch-sizes. We will explore this notion and suggest a new optimizer modification that enables training with multiple varying batch-sizes with limited hyper-parameter tuning.

3 MIXSIZE: TRAINING WITH MULTIPLE IMAGE SCALES

The traditional practice of training convolutional networks using fixed-size images holds several shortcomings. First, CNNs are commonly evaluated using a different size than that used for training (Lin et al., 2017; Redmon & Farhadi, 2018; He et al., 2017) and it was observed that classification accuracy may degrade above or below a certain size threshold (Touvron et al. (2019) and Figure 1). To remedy these issues, we suggest a stochastic training regime, where image sizes can change in each optimization step.

Motivation. In order to motivate our method, we first evaluate the impact of the image size on the training progress of a CNN — by examining gradient statistics during training[2]. Specifically, in Table 1 we measured the correlation of the gradients across image sizes. We see that gradients computed across different scales of the same image have a strong correlation compared to those obtained across different images. This correlation is especially apparent during the first stages of training and decreases as the model converges. This suggests that the small image gradients can be used as an approximation of the full image gradients, with a smaller computational footprint. Therefore, using large images along the entire training process may be sub-optimal in terms of computational resource utilization. More specifically, as the gradients of images of different size are highly correlated at the initial steps of training, it may prove beneficial to sacrifice spatial size in favor of batch size that can be increased. To do so, we suggest the following.

The MixSize training regime. We suggest "MixSize", a stochastic training regime, where input sizes can vary in each optimization step. In this regime, we modify the spatial dimensions H, W (height and width) of the input image size[3], as well as the batch size. The batch size is changed either by the number of samples used, denoted B, or the number of batch-augmentations for each sample (Hoffer et al., 2019), denoted D ("duplicates"). To simplify our notation and use-cases, we will follow the common practice of training on square images and use $S = H = W$. Formally, in the MixSize regime, these sizes can be described as random variables sharing a single discrete distribution

$$(\hat{S}, \hat{B}, \hat{D}) = \{(S, B, D)_i \;\; \textit{w.p.} \;\; p_i\}\,, \tag{1}$$

where $\forall i : p_i \geq 0$ and $\sum_i p_i = 1$.

[2]We used a ResNet-44 model (He et al., 2016), trained on the CIFAR10 dataset (Krizhevsky, 2009), whose standard image size is 32×32. Measurements are performed on the whole network's gradient vector. Images were sampled in uniform. The smaller 24×24 images were down-sampled with bilinear interpolation.

[3]The input image size is changed using bilinear interpolation. The spatial dimensions of all intermediate maps in the CNN are changed accordingly, at the same scale as the input.

Table 1: ResNet-44 gradient correlation on CIFAR10. We measure the Spearman correlation coefficient ρ between different spatial size of random images $\rho\left(x^{(s_1)}, x^{(s_2)}\right)$, as well as non-identical random images of the same size $\rho\left(x^{(s_1)}, y^{(s_1)}\right)$. We also compute the variance $V(x)$ for the gradients of each spatial size.

Measure	Network State		
	Initial	Partially Trained	Fully Trained
Epoch	1	50	100
Test Accuracy	55.12%	87.56%	92.62%
$\rho\left(x^{(32)}, x^{(24)}\right)$	0.2	0.08	0.03
$\rho\left(x^{(32)}, y^{(32)}\right)$	0.086	0.02	-0.004
$V\left(x^{(32)}\right)$	$1.03e^{-6}$	$1.44e^{-6}$	$6.24e^{-7}$
$V\left(x^{(24)}\right)$	$1.95e^{-6}$	$6.34e^{-6}$	$2.26e^{-5}$

As the computational cost of each training step is approximately proportional to $S^2 \cdot B \cdot D$, we choose these sizes to reflect an approximately fixed budget for any choice i such that $S_i^2 B_i D_i \approx Const$. Thus the computational and memory requirements for each step are constant.

Benefits and Trade-offs. We will demonstrate that using such a MixSize regime can have a positive impact on the resiliency of trained networks to the image size used at evaluation. That is, mixed-size networks will be shown to have better accuracy across a wide range of sizes. This entails a considerable saving in computations needed for inference, especially when using smaller models. Furthermore, given a fixed budget of computational and time resources (per step), we can now modify our regime along spatial and batch axes. We will explore two trade-offs:

- **Decrease number of iterations per epoch** – by enlarging B at the expense of S.
- **Improve generalization per epoch** – by enlarging D at the expense of S.

4 Improved training practices for MixSize

MixSize regimes continuously change the statistics of the model's inputs, by modifying the image size as well as batch-size. This behavior may require hyper-parameter tuning and may also affect size-dependent layers such as batch normalization (Ioffe & Szegedy, 2015). To easily adapt training regimes to the use of MixSize as well as improve their final performance, we continue to describe two methods we found useful: **Gradient Smoothing** and **Batch-norm calibration**.

4.1 Gradient smoothing

Training with varying batch and spatial sizes inadvertently leads to a change in the variance of the accumulated gradients. For example, in Table 1, the gradient variance is larger when computed over a small image size (unsurprisingly). This further suggests that the optimization regime should be adapted to smaller spatial sizes, in a manner similar to learning-rate adaptations that are used for large-batch training. This property was explored in previous works concerning large-batch regimes, in which a learning rate modification was suggested to compensate for the variance reduction for larger batch-sizes. Unfortunately, the nature of this modification can vary from task to task or across models (Shallue et al., 2018), with solutions such as a square-root scaling (Hoffer et al., 2017), linear scaling (Goyal et al., 2017) or a fixed norm ratio (You et al., 2017). Here we suggest changing both the spatial size as well as the batch size, which is also expected to modify the variance of gradients within each step and further complicates the choice of optimal scaling.

Previous works suggested methods to control the gradient norm by gradient normalization (Hazan et al., 2015) and gradient clipping (Pascanu et al., 2013). These methods explicitly disable or limit the gradient's norm used for each optimization step, but also limit naturally occurring variations in gradient statistics. We suggest an alternative solution to previous approaches, which we refer to as "Gradient smoothing". Gradient smoothing mitigates the variability of gradient statistics when image sizes are constantly changing across training.

We introduce an exponentially moving weighted average of the gradients' norm $\bar{g}_t$ (scalar) which is updated according to

$$\bar{g}_t = \alpha \bar{g}_{t-1} + (1-\alpha) g_t$$

where

$$g_t = \left\| \frac{\partial E}{\partial w_t} \right\|_2 \quad \text{and} \quad \bar{g}_0 = g_0 \,.$$

We normalize the gradients used for each step by the smoothing coefficient, such that each consecutive step is performed with gradients of similar norm. For example, for the vanilla SGD step, we use a weight update rule of the form

$$w_{t+1} = w_t - \eta \frac{\bar{g}_t}{g_t} \frac{\partial E}{\partial w_t} \,.$$

This running estimate of gradient norm is similar to the optimizer suggested by Ginsburg et al. (2019), which keeps a per-layer estimate of gradient moments. Gradient smoothing, however, is designed to adapt globally (across all layers) to the batch and spatial size modification and can be used regardless of the optimization method used.

We found gradient smoothing to be mostly beneficial in regimes where multiple varying batch sizes are used. Figure 5a in the Appendix demonstrates how gradient smoothing reduces the gap between gradient norms of different sizes. Measuring test error on the same model shows a slight advantage for gradient-smoothing (Appendix Figure 5b).

4.2 Batch-norm calibration for varying image sizes

As demonstrated by Touvron et al. (2019), using a different image size at evaluation may incur a discrepancy between training and evaluation protocols, caused by using different data pre-processing. Touvron et al. (2019) suggested a post-training procedure, where a network trained on a specific fixed-size is fine-tuned on another size, later used for evaluation. Their solution required 10s of training epochs, amounting to 1000s of full forward and back-propagation computations, along with parameter updates for batch-norm and classifier layers. In contrast, we surmise that for networks trained with mixed-regimes, discrepancy issues mainly arise from the use of the batch-norm layers (Ioffe & Szegedy, 2015) and can be solved by targeting them specifically.

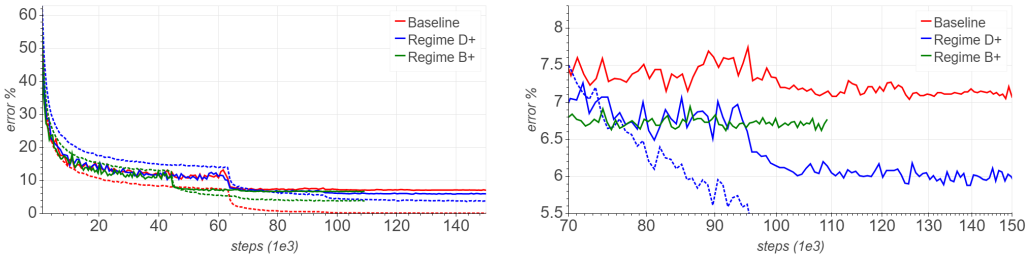
Batch-norm layers introduce a discrepancy between training and test evaluations (Ioffe, 2017), as at inference a running estimate of the mean and variance (of training data) are used instead of the actual mean and variance values. This difference is emphasized further in the use of varying image size, as changing the spatial size of an input map can significantly modify the measured variance of that map. While a fine-tuning process per image size can eliminate this discrepancy (Touvron et al., 2019), we offer a simpler alternative. For each evaluated size, we calibrate the mean and variance estimates used for that size by computing an average value over a small number of training examples. This calibration requires only a few (100s) feed-forward operations with no back-propagation or parameter update and takes only a few seconds on a single GPU.

Interestingly, we highlight the fact that although this process has little or no effect on models trained using a fixed-size input, it does improve our mixed-size models considerably on a wide range of image sizes.

5 Experiments

5.1 MixSize with a fixed image size at test-time: the speed-accuracy trade-off

CIFAR10/100. First, we examine our method using the common visual datasets CIFAR10/100 (Krizhevsky, 2009) that consist of 32×32 color images. We use the ResNet-44 model suggested by (He et al., 2016), Wide Resnet WRN-28-10 (Zagoruyko, 2016) and AmoebaNet (Real et al., 2019) with their original regime and batch size of 64. While for ResNet-44 we use the original augmentation protocol, we apply cutout (DeVries & Taylor, 2017) and auto-augment policies (Cubuk et al., 2018) on WRN-28-10 and AmoebaNet for both datasets (see Appendix A.1 for details).



(a) Test error

(b) Final test error

Figure 2: Training (dotted) and test accuracy vs optimization step (ResNet44, CIFAR10). We compare vanilla training with two computationally equivalent stochastic regimes: increased duplicates (D^+) and increased batch (B^+). B^+ **regime achieves better test accuracy at a reduced number of iterations, while** D^+ **improves accuracy further at a similar computational cost**.

As CIFAR datasets are limited in size, we consider the following balanced stochastic regime chosen:

$$S = \begin{cases} 40, & \text{w.p.} \quad p = 0.2 \\ 32, & \text{w.p.} \quad p = 0.3 \\ 24, & \text{w.p.} \quad p = 0.3 \\ 16, & \text{w.p.} \quad p = 0.2 \end{cases}$$

The regime was designed to be centered around the mean value of 28. As the original image size used for training is 32×32, we are now able to increase either the batch size or number of duplicates for each training step by a factor of $\frac{32^2}{S^2}$ such that $S^2 \cdot B \cdot D$ is approximately constant. We denote our modified mixed-size regimes as B^+ for an increased effective batch-size and D^+ for an increased number of BA duplicates of the same ratio. We used our sampling strategy to train and compare our regime to the baseline results. We use the original hyper-parameters without modification. For the B^+ regime, use our gradient smoothing method, as described in Section 4.1. For each result, we measure our final test accuracy on the original 32×32 image size. We also perform batch-norm calibration as described in Section 4.2. From Table 2, we see that our MixSize regimes on CIFAR datasets yield two possible improvements:

- Reduced number of training steps to achieve a similar test accuracy using B^+ regime.
- Better test accuracy when using D^+ regime.

Training progress on the CIFAR10 using ResNet44 is depicted in Figure 2. Interestingly, although designed only to reduce training time, we can see that our B^+ regime also improves accuracy in some cases. This improvement can be attributed to a regularization effect induced by changing image sizes during training, also manifested by an increase in training error throughout its progress.

ImageNet. We also perform large scale experiments using the ImageNet dataset (Deng et al., 2009) to confirm our findings. We used the ResNet-50 (He et al., 2016) model, with the training

Table 2: Test accuracy (Top-1) results for CIFAR and ImageNet. Each row represents models trained using the same computational and memory budget per step. Steps and accuracy are reported at the completion of a fixed epoch budget (e.g., 90 epochs for ResNet on ImageNet, 200 for ResNet on CIFAR). Accuracy is reported for model's original size (32 for CIFAR, 224 for ImageNet).

Network	Dataset	Steps			Accuracy		
		Baseline	B^+	D^+	Baseline	B^+	D^+
ResNet-44	CIFAR10	156K	**109K**	156K	92.84%	94.30%	**94.46%**
WRN-28-10	CIFAR10	156K	**109K**	156K	96.60%	97.28%	**97.68%**
AmoebaNet	CIFAR10	469K	**328K**	469K	98.16%	98.14%	**98.32%**
ResNet-44	CIFAR100	156K	**109K**	156K	70.36%	72.19%	**73.10%**
WRN-28-10	CIFAR100	156K	**109K**	156K	79.85%	83.08%	**83.52%**
ResNet-50	ImageNet	450K	**169K**	450K	76.40%	76.61%	**78.04%**
EfficientNet-B0	ImageNet	1000K	**376K**	1000K	76.32%	76.29%	**76.53%**

regime suggested by Goyal et al. (2017) that consists of base learning rate of 0.1, decreased by a factor of 10 on epochs $30, 60, 80$, stopping at epoch 90. We used the base batch size of 256 over 4 devices and L_2 regularization over weights of convolutional layers. We used the standard data augmentation and did not incorporate any additional regularization or augmentation techniques.

Additionally, we also used the EfficientNet-B0 model suggested by Tan & Le (2019). We used the same data augmentation and regularization as the original paper, but opted for a shorter training regime with a momentum-SGD optimizer that consisted of a cosine-annealed learning rate (Loshchilov & Hutter, 2016) over 200 epochs starting from an initial base 0.1 value.

For the ImageNet dataset, we use the following stochastic regime found by cross-validation on several alternatives (see Appendix D):

$$S^{(144)}: \quad S = \begin{cases} 256, & \text{w.p} \quad p = 0.1 \\ 224, & \text{w.p} \quad p = 0.1 \\ 128, & \text{w.p} \quad p = 0.6 \\ 96, & \text{w.p} \quad p = 0.2 \end{cases}$$

While the original training regime consisted of images of size 224×224, our proposed regime makes for an average image size of $\bar{S} \times \bar{S} = 144 \times 144$. This regime was designed so that the reduced spatial size can be used to increase the corresponding batch size or the number of BA duplicates, as described in Section 3. We are first interested in accelerating the time needed for convergence of the tested models using our B^+ scheme. We enlarge the batch size used for each spatial size by a factor of $\frac{224^2}{S^2}$ such that $S^2 \cdot B$ is kept approximately fixed. As the average batch size is larger than B_o, which was used with the original optimization hyper-parameters, we scale the learning rate linearly as suggested by Goyal et al. (2017) by a factor of $\frac{\bar{B}}{B_o}$. We note that for the proposed regimes we did not require any learning rate warm-up, due to the use of gradient smoothing.

As can be seen in Figure 3, regime B^+ enables training with approximately $2.7\times$ less training steps, while reaching a better-than-baseline accuracy of 76.61%. As sizes were chosen to reflect in approximately equal computational cost per iteration, B^+ regime offers a similar improvement in total wall-clock time.

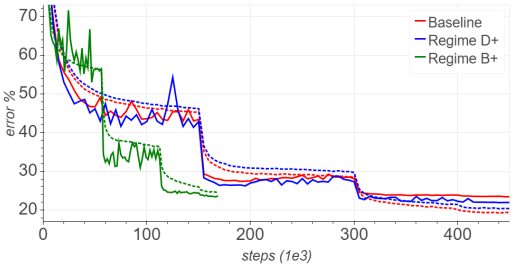


Figure 3: Training (dotted) and test accuracy on ImageNet using the Baseline, B^+ and D^+ regimes (224×224 evaluation size). All regimes required similar computational resources per step. B^+ regime required $\approx 2.7\times$ less steps per epoch.

Next, we perform a similar experiment with a D^+ regime, where the number of BA duplicates is similarly increased with respect to D_o instead of the batch size. This scaling results with an average duplicates of $\bar{D} = 3$.

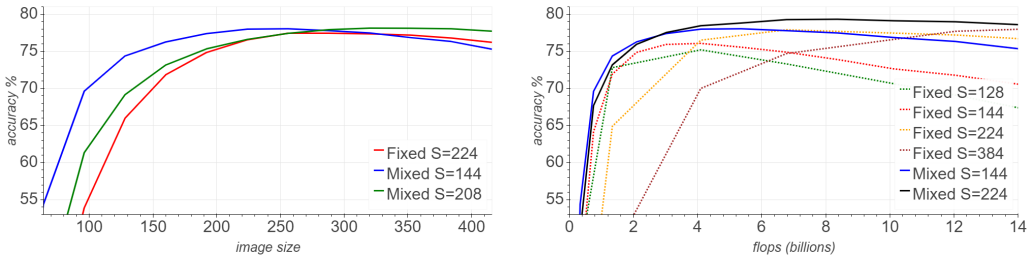
As the computational cost for each step remains approximately constant, as well as the number of required steps per epochs, training a model under this regime requires an equal wall-clock time. However, the increased batch-augmentation improves the final test accuracy to 78.04%, approximately 7% relative improvement over the 76.4% baseline.

5.2 Increasing model resiliency to test-time changes in image size

Next, we examine how MixSize affects the resulting model resiliency to changes in the image size during test-time. We evaluated the models by varying the test-time image sizes around the original 224 spatial size: $S = 224 + 32 \cdot m, m \in \{-6, ..., 6\}$. The common evaluation procedure for ImageNet models first scales the image to a 256 smallest dimension and crops a center 224×224 image. We adapt this regime for other image sizes by scaling the smallest dimension to $\lfloor \frac{8}{7} S \rfloor$ (since $\frac{8}{7} \cdot 224 = 256$) and then cropping the center $S \times S$ patch. Models trained with a mixed regime were calibrated to a specific evaluation size by measuring batch-norm statistics for 200 batches of training samples. We note that for original fixed-size regimes this calibration procedure resulted with degraded results and so we report accuracy without calibration for these models. We did not use any fine-tuning procedure post training for any of the models.

As can be seen in Figure 4a, the baseline model trained using a fixed size, reaches 76.4% top-1 accuracy at the same 224 spatial size it was trained on. As observed previously, the model continues to slightly improve beyond that size, to a maximum of 76.8% accuracy. However, it is apparent that the model's performance quickly degrades when evaluating with sizes smaller than 224.

We compare these results with a D^+ regime, trained with an average size of $\bar{S} = 144$. As described earlier, this model requires the same time and computational resources as the baseline model. However, due to the decreased average size, we were able to leverage more than 1 duplicates per batch on average, which improved the model's top-1 accuracy to 77.14% at size 224. Furthermore, we find that the model performs much more favorably at image sizes smaller than 224, scoring an improved (over baseline) accuracy of 76.43% at only 160×160 spatial size. We analyzed an alternative regime $S^{(208)}$, where the average spatial size is larger at 208×208 (for more details see Appendix D). The model trained with the $S^{(208)}$ regime offers a similar improvement in accuracy, only across a larger spatial size, as it observed an average size of 208×208 during training. Figure 4a demonstrates that while all three models (Fixed with $S = 224$, $S^{(144)}$ and $S^{(208)}$) were trained with the same compute and memory budget, mixed-size regimes offer superior accuracy over a wide range of evaluation sizes. Specifically, mixed-regime at $S = 208$ dominates the baseline fixed-size regime at all sizes, while our mixed regime at $S = 144$ achieves best results at sizes smaller than 224.



(a) (b)

Figure 4: Left: Test accuracy on validation set per image size, all models trained using the same computational and memory resources (regime D^+). Right: Test accuracy per billion flop (at evaluation).

We also compared the classification performance across evaluated image sizes, using networks trained on a variety of fixed sizes and our mixed regimes. As a baseline, we use results obtained by Touvron et al. (2019) (trained with repeated augmentations, without fine-tuning) and compare them with mixed-regime models trained with an equal computational budget, by setting the base number of BA duplicates to $D = 2$. As can be seen in Figure 4b, mixed-regime trained models offer a wider range of resolutions with close-to-baseline accuracy (within a 2% change) and perform better than their fixed-size counterparts at all sizes. As the number of floating-point operations (flops) grows linearly with the number of pixels, using a mixed regime significantly improves accuracy per compute at evaluation. We further note that our $S^{(224)}$ model reaches a top accuracy of 79.27% at a 288×288 evaluation size.

6 Summary

In this work, we introduced and examined a performance trade-off between computational load and classification accuracy governed by the input's spatial size. We suggested stochastic image size regimes, which randomly change the spatial dimension as well as the batch size and the number of augmentation (duplicates) in the batch. Stochastic regime benefits are threefold: (1) reduced number of training iterations; or (2) improved model accuracy (generalization) and (3) improved model robustness to changing the image size. We believe this approach may have a profound impact on the practice of training convolutional networks. Given a computational and time budget, stochastic size regimes may enable to train networks faster, with better results, as well as to target specific image sizes that will be used at test time. As the average size chosen to train is reflected in the optimal operating point for evaluation resolution, mixed regimes can be used to create networks with better performance across multiple designated use cases.

REFERENCES

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

Maxim Berman, Hervé Jégou, Andrea Vedaldi, Iasonas Kokkinos, and Matthijs Douze. Multigrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019.

Dami Choi, Alexandre Passos, Christopher J Shallue, and George E Dahl. Faster neural network training with data echoing. *arXiv preprint arXiv:1907.05550*, 2019.

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.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1243–1252. JMLR. org, 2017.

Boris Ginsburg, Patrice Castonguay, Oleksii Hrinchuk, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, Huyen Nguyen, and Jonathan M Cohen. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. *arXiv preprint arXiv:1905.11286*, 2019.

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.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In *Advances in Neural Information Processing Systems*, pp. 1594–1602, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: better training with larger batches. *arXiv preprint arXiv:1901.09335*, 2019.

Andrew G Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

Jeremy Howard. Fastai - progressive resizing. `https://www.fast.ai/2018/04/30/dawnbench-fastai/`, 2018.

Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*, 2018.

Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in neural information processing systems*, pp. 1945–1953, 2017.

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

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pp. 1150–1157 vol.2, 1999.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318, 2013.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Scale-invariant recognition by weight-shared cnns in parallel. In *Asian Conference on Machine Learning*, pp. 295–310, 2017.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114, 2019.

Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. *CoRR*, abs/1906.06423, 2019. URL http://arxiv.org/abs/1906.06423.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 2016.

Yichong Xu, Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, and Zheng Zhang. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*, 2014.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Komodakis Zagoruyko. Wide residual networks. In *BMVC*, 2016.

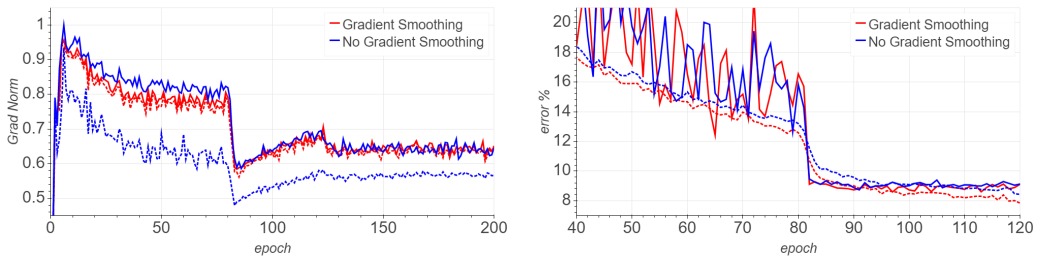
Appendix

A EXPERIMENTAL SETTINGS

A.1 CIFAR

We used the common data augmentation technique as described by He et al. (2016). In this method, the input image is padded with 4 zero-valued pixels at each side, top and bottom. A random 32×32 part of the padded image is then cropped and with a 0.5 probability flipped horizontally. In order to adapt to varying input scales, we add an additional augmentation step, that resizes the images using bilinear interpolation to $S \times S$, depending on a sampled size for each step. We note that this keeps the exact original augmentation procedure for $S = 32$.

B IMPACT OF GRADIENT SMOOTHING



(a) Gradient norm values with and without Grad-smoothing. Solid lines are gradients for $S = 32$ while dotted lines are for $S = 16$.

(b) Training and test error with and with out gradient smoothing. Solid lines are test errors while dotted lines are for training.

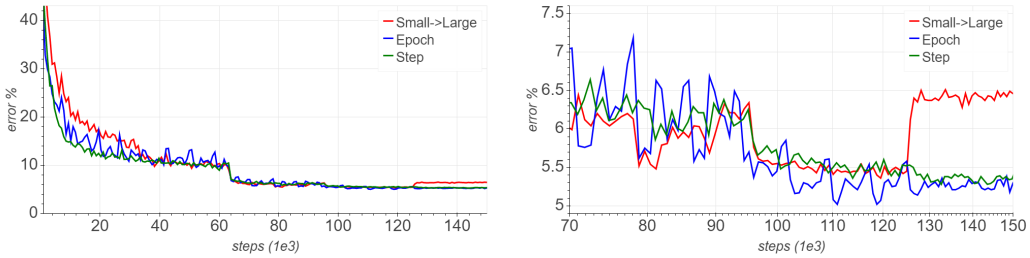
Figure 5: Impact of gradient smoothing on CIFAR10, ResNet-44. The training regime includes two image sizes: 32×32 and 16×16 (average size is $S = 24$). Using a B^+ regime creates two batch sizes: 256 and $2,048$ respectively. Gradient smoothing helps to reduce gap between gradient norms at difference batch sizes and improves final accuracy.

C VARYING IMAGE-SIZE TRAINING REGIMES

We wish to consider training regimes with varying image sizes, such that the average image size is smaller than the desired evaluation size. For example, for the height dimension H, we wish to obtain an average size of $\bar{H} = \sum_i p_i H_i$ such that $\bar{H} < H_o$. We consider three alternatives for image size variations:

- Increase image size from small to large, where each image size is used for number of epochs $E_i = p_i E_{\text{total}}$, where E_{total} is the total number training epochs required.
- Using a random image size for each epoch, keeping the epoch number for each size at E_i
- Sampling image size per training step at probability p_i

As can be seen in Figure 6, we found that random sampling regimes performed better than scaling image size from small to large (Howard, 2018; Touvron et al., 2019). While sampling both at epoch and step time frames performed similarly, replacing sizes on each step seemed to converge faster and to have less noise in measured test accuracy. We note that these behaviours may partly stem from the use of batch-normalization (Ioffe & Szegedy, 2015) which is sensitive to the image size used at evaluation or insufficient hyper-parameter tuning for each specific size (e.g., spiking error at the end of the small-to-large regime). Considering these findings, we continue to perform our experiments using the third regime – sampling image size per training step.



(a) Test error

(b) Final test error

Figure 6: Test accuracy vs step for 3 size sampling regimes: (1) From small to large (2) Sample each Epoch (3) Sample each step. All methods reached a similar accuracy, but sampling each epoch was less noisy and did not require hyper-parameter tuning.

D ALTERNATIVE SIZE DISTRIBUTION REGIMES

We used alternative size regimes balanced around 224, named $S^{(208)}$ and $S^{(224)}$. They can be described by the following distributions:

$$S^{(208)}: \quad S = \begin{cases} 320, & \text{w.p} \quad p = 0.1 \\ 288, & \text{w.p} \quad p = 0.1 \\ 256, & \text{w.p} \quad p = 0.1 \\ 224, & \text{w.p} \quad p = 0.2 \\ 192, & \text{w.p} \quad p = 0.2 \\ 160, & \text{w.p} \quad p = 0.1 \\ 128, & \text{w.p} \quad p = 0.1 \\ 96, & \text{w.p} \quad p = 0.1 \end{cases}$$

$$S^{(224)}: \quad S = \begin{cases} 320, & \text{w.p} \quad p = 0.133 \\ 288, & \text{w.p} \quad p = 0.133 \\ 256, & \text{w.p} \quad p = 0.133 \\ 224, & \text{w.p} \quad p = 0.2 \\ 192, & \text{w.p} \quad p = 0.133 \\ 160, & \text{w.p} \quad p = 0.133 \\ 128, & \text{w.p} \quad p = 0.133 \end{cases}$$