Masked Layer Distillation: Fast and Robust Training Through Knowledge Transfer Normalization

Derek Wan
University of California, Berkeley
derekwan@berkeley.edu

Tianjun Zhang
University of California, Berkeley
tianjunz@berkeley.edu

Paras Jain
University of California, Berkeley
paras_jain@berkeley.edu

Joseph E. Gonzalez
University of California, Berkeley
jegonzal@berkeley.edu

Kurt W. Keutzer
University of California, Berkeley
keutzer@berkeley.edu

Ion Stoica
University of California, Berkeley
istoica@berkeley.edu

Abstract

Knowledge distillation is a widely used method for compressing models, accelerating training, and improving model performance. Distillation boosts accuracy beyond what from-scractch training can achieve. However, prior distillation approaches are benchmarked using ResNet and VGG while little attention has been paid to emerging network architectures. In this work, we surprisingly find that distillation incurs significant accuracy penalties for small networks such as EfficientNet and MobileNet-V3. To this end, we propose Masked Layer Distillation (`MLD`), a new training algorithm that trains a sensitivity mask which is used for matching intermediate features. The mask relaxes the distillation loss for sensitive neurons thereby improving generalization of the model during subsequent fine-tuning. `MLD` state-of-the-art distillation approaches by at least 2% on EfficientNet and MobileNet. `MLD` also translates well to other models such as ResNets and VGGs. Overall, `MLD` accelerates the training convergence of tested architectures by $2\times$ to $5\times$.

1 Introduction

Distillation is an important method to transfer knowledge from one model (a teacher) to another model (a student). Distillation enables condensing an ensemble of teachers to one student (cite dean et al), compression of large teachers to few-parameter students and improvements to training accuracy (label refinery). Distillation often yields a student model with higher accuracy than would be possible from simply training from scratch without a teacher.

However, previous work has suggested that depthwise separable convolutions (DSCs) are not resilient to compression [9], which led us to investigate whether these networks that depend on DSCs are amenable to distillation. We find a remarkable breakdown of distillation when applied to modern architectures. Specifically, when we apply competitive knowledge distillation methods to EfficientNet and MobileNet, we find that both models do significantly worse than training from scratch. When distilling EfficientNet-b0 to an equivalent student, top-1 ImageNet accuracy drops by 6.1% while MobileNetv3 drops by 7.0%.

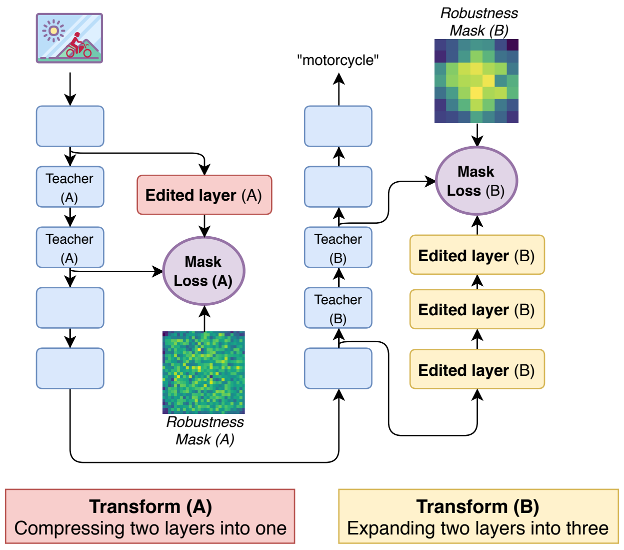


Figure 1: Masked Layer Distillation allows rapid retraining of modified neural network architectures. Given a single change, we can retrain the modified DNN 2-5x faster than from scratch. Our key contribution is the Mask Loss, a loss function that estimates robustness per-layer and then rapidly performs fine-grained knowledge transfer from old layer to new layer. We can apply architecture transformations in parallel enabling a wide class of graph-to-graph transformations. We show two sample operations we support for DNNs: (A) deleting a single layer and (B) inserting a new layer.

We hypothesize that this breakdown in distillation is due to training instability from the depthwise separable convolution (DSC) block which factorizes a dense convolution into a composition of a channel-independent and spatial-independent operations. While this factorization enables compressing parameters, we find this block is hard to distill. In a small microbenchmark, we isolate the training instability to DSC and find per-layer gradient norms are significantly noisier than those from ResNet. DSC instability dramatically increases with deeper networks and out-of-distribution inputs.

In this work, we demonstrate propose a remedy for this curious case of distillation degradation. Many recent groups have demonstrated state-of-the-art accuracy on ResNets, or other similar models that are known to have favorable distillation properties, but some of the most competitive of those methods fail on EfficientNet and MobileNet. Our method, however, is compatible with both EfficientNet / MobileNet and more popular models such as ResNets, offering superior or competitive performance and significantly faster training across a variety of architectures and arbitrary architecture edits.

To improve distillation accuracy, we introduce Masked Layer Distillation, a novel training algorithm that significantly outperforms competitive distillation methods on diverse architectures like EfficientNet. To avoid distillation failure due to depth, we add a layerwise training phase prior to conventional distillation which distills segments of a network independently with a dense loss signal. In order to improve sensitivity of distillation under small perturbations of layer inputs, we introduce a series of learnable masks to prioritize learning a robust model of in-distribution inputs while ignoring out-of-distribution samples during distillation.

Overall, Masked Layer Distillation dramatically improves ImageNet accuracy for MobileNetv3 and EfficientNet architectures over competitive distillation baselines including conventional distillation and Label Refinery [3] [29]. As an additional benefit, we also find Masked Layerwise Distillation significantly accelerates convergence for common model architectures by 2-5x when compared with standard end-to-end fine-tuning.

2 Related Work

Several approaches analytically derive a transformation from the source to the target networks, as in [26, 27, 8]. Network Morphism [26] examines expanding the number of filters and the kernel size of a DNN layer. Net2Net [8] introduces equations to add a single new convolution and a to increase the number of filters in a layer. However, their approach is limited to a small handful of basic architecture transformations. Moreover, their approach does not enable transformations that delete a layer or change a layer. Our approach is more general, works across a wider range of transformations and architectures, and allows both adding and removing parameters.

Model compression provides a general framework for taking the knowledge contained in a complex model (either deep or an ensemble) and transferring the knowledge to a simpler network [6, 2, 17]. Model distillation [17] efficiently distills an ensemble into a single simpler model. While model compression and model distillation are general, they require many training iterations to converge to full performance. Our approach is a more precise and fine-grained form of distillation which significantly accelerates the knowledge transfer process over whole-model distillation. These approaches are complementary to Masked Layer Distillation – we utilize model distillation during Stage 2 of our algorithm.

Warm starting [1] allows models that have been partially trained on a subset of a dataset (in an online setting, for example) to be efficiently updated without sacrificing generalization performance. Similar to our goals, warm starting solutions also aim to leverage existing parameters to boost predictive performance. FitNets [22] train networks using curriculum learning that distills information from a source model to a deeper target model in various stages with an L2 loss. The local distillation in our technique resembles the FitNets approach, but we accelerate training and achieve better accuracy recovery with our Mask Loss.

Contrastive learning is used in self-supervised learning to encourage models to learn the similar internal representations [13] [25]. These methods inspired us to develop our methodology, which similarly leverages intermediate representations for knowledge transfer. We found that these intermediate embeddings contain rich information that can greatly boost performance if incorporated into a carefully designed loss function, which we explain in the next section.

3 Methodology

Gradient descent is sensitive to choice of initialization [15]. We propose Masked Layer Distillation as a regularization objective for deep networks that accelerates training speed while improving robustness. Additionally, Masked Layer Distillation allows us to swap out an entire layer group in a source model with a new target layer group. Unlike prior approaches to transfer knowledge, Masked Layer Distillation is more general: we enable knowledge transfer from any source architecture to new target architecture with few limitations. This includes adding, replacing or deleting any layer groups.

Masked Layer Distillation is composed of three stages: (1) Mask generation, during which we measure the sensitivity of the activations by randomly initializing the student layers and injecting them into the teacher (2) Local distillation (3) End-to-end fine tuning. Further details are shown in algorithm 1 and Figure 2

Masked Layer Distillation is fast and accurate due to early phases of *local distillation* combined with a sensitivity-aware *Mask Loss*. Local distillation breaks down learning a complex model into learning small groups of layers, simplifying and speeding up the process of retraining the new layer groups. The Mask Loss enables the target layer groups to achieve an overall higher accuracy approximation of the source layer groups.

3.1 Fine-to-coarse knowledge transfer

Because this work builds on model distillation [17], we provide a brief overview and introduce key notation. Classic model distillation can be formulated as an empirical risk minimization problem:

$$\min_{W_s} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i; W_t), g(x_i; W_s)), \tag{1}$$

where $f(\cdot; W_t)$ is a pretrained source network parameterized by the trained weights W_t, $g(\cdot; W_s)$ is a target network parameterized by W_s, and the loss function $\mathcal{L}$ is taken between the outputs of the target and source networks. Model distillation exploits two key insights. First, because the student learns from teacher predictions on data instead of human labels, we can train on a significantly larger dataset without labeling it. Training models with sufficient capacity on significantly larger datasets

Algorithm 1: Layer Distillation Pseudocode for a Single Mask (illustrated in Figure 2)

```
Input: Dataset $\mathcal{D}$
Input: Pre-trained $m$ layer model $f_m(f_k(f_l(x)))$
Input: Target $n$ layer model $g_n(g_i(g_j(x)))$
Input: Task-specific loss function $L(\hat{y}, y)$
Alignment: Source $\{f_l(x), f_k(f_l(x))\}$
Alignment: Target $\{g_j(x), g_i(g_j(x))\}$
——————————— Phase 1: Mask Generation ———————————
Initialize mask $M$ with learnable weights
Freeze source and target model weights
for $(x, y) \in \mathcal{D}$ do
    Inject noise $f(x) \leftarrow f_m(f_k(f_l(x)) + (1 - M)g_i(f_l(x)))$
    Calculate loss $L' \leftarrow L(f(x), y)$
    Update mask $M$ via SGD $M \leftarrow M - \eta * \nabla L'$
end for
——————————— Phase 2: Layer Distillation ———————————
Unfreeze $g_\theta$ target weights
for $(x, y) \in \mathcal{D}$ do
    Calculate masked loss $L_m = \|(1 - ReLU(M)) * (f_k(f_l(x)) - g_i(g_j(x)))\|_1$
    Update $g_\theta$ via SGD $g_\theta \leftarrow g_\theta - \eta * \nabla L_m$
end for
——————————— Phase 3: End-to-End Distillation ———————————-
for $(x, y) \in \mathcal{D}$ do
    Calculate KD loss $L = L_{KD}(f(x), g(x))$
    Update $g_\theta$ via SGD $g_\theta \leftarrow g_\theta - \eta * \nabla L$
end for
```

such as JFT-300M has been shown to boost performance by several percentage points [24] [12] [11] Second, by leveraging the appropriate loss function $\mathcal{L}$ we can transfer information about the labels and their uncertainty. This allows student models to learn to generalize more quickly, as the student learns not only to classify an image as X or Y but also to predict the *probability* that image is X or Y. These "soft targets" have higher entropy and confer much more information than the labels alone.

3.2 Local distillation

The first step of Masked Layer Distillation is local distillation. In contrast to classic model distillation, local distillation trains individual layers or groups of layers in the target network to approximate the corresponding layers in the source network. We denote the individual layers of the source and target networks (f and g respectively) as:

$$x_t^{(l)} = f^{(l)}\left(x_t^{(l-1)}; W_t^{(l)}\right),\ x_s^{(l)} = g^{(l)}\left(x_s^{(l-1)}; W_s^{(l)}\right), \tag{2}$$

where $x_t^{(l-1)}, x_s^{(l-1)}$ represent the input to layer-l in the source and target network, $x_t^{(l)}, x_s^{(l)}$ are the output of layer-l, and $W_t^{(l)}, W_s^{(l)}$ are the weights. Local distillation decomposes the problem (Eq. 1) into a separate distillation problem for each layer l:

$$\min_{W_s^{(l)}} \sum_{x \in \mathcal{X}} \mathcal{L}\left(f^{(l)}\left(x_t^{(l-1)}; W_t^{(l)}\right), g^{(l)}\left(x_t^{(l-1)}; W_s^{(l)}\right)\right). \tag{3}$$

This decomposition provides much faster convergence than full model distillation by reducing the parameter complexity and eliminating many of the challenges of vanishing gradients. [4] [30]. In our experiments, we find that only a few epochs of local distillation are sufficient to recover most of the accuracy of the target network.

3.3 Mask Loss: Sensitivity-aware distillation loss

As with model distillation, the choice of loss function can have a significant impact in the overall performance. FitNets [22] proposed a loss function that minimizes the L_2 loss between intermediate

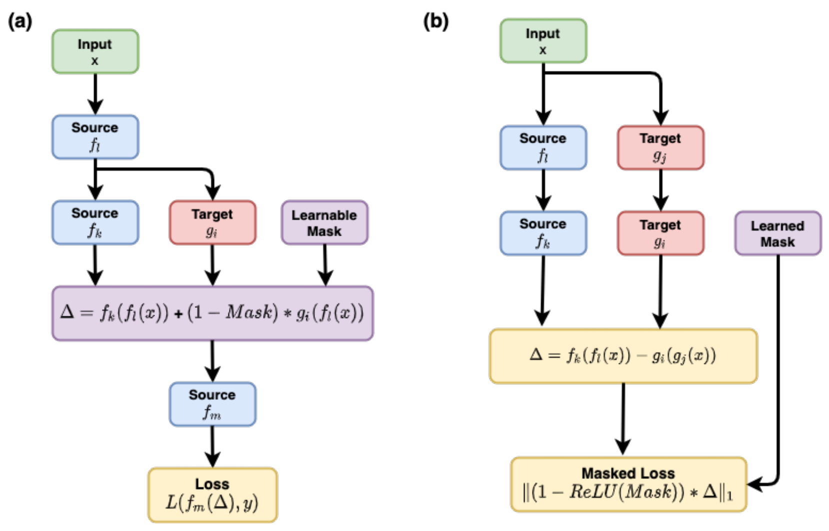


Figure 2: Layer distillation for a single mask. A per-pixel sensitivity mask is learned for each activation. (a) The mask is learned by injecting noise from a randomized target model into the source's activations and minimizing the subsequent downstream loss. (b) The learned mask is then used to weight the L_1 loss between pixels of the source and target.

representations of the teacher and student networks. However, we find that the L_2 loss alone can lead to large reconstruction errors between the source network and the target networks. When we attempt to distill to ResNet-34 by applying layer distillation to only a single block, the student is only able to achieve -12.1% of baseline teacher accuracy on ImageNet.

Directly minimizing the L_2 loss between the source and target at each layer is a poor way to formulate the objective because different pixels have different weight in influencing the final prediction. For example, corner pixel activations may have far less influence on the prediction than pixels in the center of the activation map. Therefore, we need a mechanism to identify the sensitivity of the classification loss to errors in the individual layer outputs.

In addition, it is well known that L_1 loss is favorable to L_2 loss when recovering signals with sharp features or when outlier noise is significant [7]. This makes the L_1 loss preferable in applying Masked Layer Distillation, as these signals become more frequent when comparing the intermediate outputs of two very different models. For this reason, we base our objective on the L_1 loss.

3.3.1 Mask Loss Formulation

To better capture the varying effect of errors in the layer outputs we introduce the Mask Loss, a local distillation technique that can transfer knowledge from source to target layer with a small error penalty. The Mask Loss improves the efficiency of Masked Layer Distillation as it increases accuracy recovered by the first local distillation phase. Intuitively, the Mask Loss reweights different parts of the reconstruction error based on the overall model's sensitivity to noise.

The mask loss first models the layer-wise sensitivity to noise in each dimension of the output activation (channel and spatial). This noise map represents the per-layer sensitivity to noise. We notice that certain channels can tolerate very little error but others are much more sensitive.

In order to measure the sensitivity to noise in the feature map, we use a randomly initialized target to generate noise. We want the model to tolerate as much noise as possible without significantly impacting loss. The details of the masked loss and training technique are illustrated below as well as in Figure 2. The user first manually selects a few "breakpoints" at which intermediate representations of source and target models are compared. In Equation 4 and Equation 5, x_{si} and x_{ti} denote source and target activations respectively at the ith user-defined breakpoint, M_i denotes the learned mask at the ith breakpoint, and $f(x)$ denotes the final output of the source model after injecting masked noise.

Learning the mask values The exact mask generation technique is as follows:

1. Randomly initialize the weights and bias of all target layer groups.
2. Freeze the source and target network
3. Inject noise into the source network at predefined breakpoints as such
$$x'_{si} := x_{si} + (1 - M_i) * x_{ti} \tag{4}$$
4. Apply gradient steps to the masks to minimize Eq. 5. α is a hyperparameter that trades off accuracy and noise resistance.
$$L_m = CrossEntropy(f(x), y) + \alpha \sum_i \|Mask_i\|_2 \tag{5}$$

Equation for the masked loss Once our mask has been trained, we apply masked L_1 loss in our training strategy and optimize this function.

$$M_{\text{Loss}} := \sum_{i=1}^{N} \|(1 - ReLU(M_i)) * (x_{si} - x_{ti})\|_1 \tag{6}$$

In Equation 4, we can see that the mask will tend toward larger values (i.e., 1) for pixels that are sensitive to noise, and in Equation 6, those larger values will prevent the loss from exploding if x_{si} and x_{ti} differ significantly at sensitive indices. This prevents the gradient of the target model from being disproportionately influenced by a very small minority of pixels.

3.4 Model distillation and fine-tuning

After local distillation, the majority of knowledge distillation has been completed. For a ResNet-34 model, we find that we can recover nearly the full accuracy potential in just 12 epochs. However, although the Mask Loss improves the accuracy gap of ResNet34 from -12% to -3% as compared to the FitNets approach, we wanted to close the gap even further. Model distillation [17] is a technique that has been proven to improve the training speed as well as smooth the loss function. Further utilizing the source network's knowledge, we add a fine tuning stage using global model distillation to train the target model. We use two variants of traditional knowledge distillation, Label Refinery (LR) and Softmax Regression Representation Learning (SRRL), proposed by [3] and [29], which offer competitive accuracy boosts on ImageNet. In cases in which the target model is expected to exhibit superior performance to the source (i.e., when distilling from a smaller model to a larger model), we additionally perform global fine tuning on the labels. These post-Layer Distillation steps allow us to achieve full target network accuracy on ResNets and significantly higher accuracy than baseline end-to-end distillation on other models in significantly fewer epochs.

4 Experiments and applications

We evaluate Masked Layer Distillation in three diverse applications. In subsection 4.1, we first demonstrate Masked Layer Distillation's robustness to a range of initialization procedures, showcasing its regularization properties. In subsection 4.2, we apply our method to MobileNet and EfficientNet to highlight a case of distillation degradation. In subsection 4.3, we repeat those experiments using a modern state-of-the-art distillation benchmark. Finally in subsection 4.4, we evaluate our approach in a practical setting where a user makes many changes to an architecture. We perform the conventional ResNet34 to ResNet18 distillation task and demonstrate competitive results, and then we demonstrate Masked Layer Distillation's versatility by distilling from ResNet18 to ResNet34. We show our approach has significant speedups over training from scratch and distillation.

4.1 Robustness to Initialization Choice

It is well known that the choice of parameter initialization is crucial to the speed of deep neural network training [28]. We evaluated Masked Layer Distillation's robustness by training target models on CIFAR-10 with varying intialization schemes and batch sizes. Results are shown in Table 1 and

Table 1: Masked Layer Distillation (MLD) vs training from scratch on CIFAR-10 with different initialization schemes. Training from scratch is very sensitive to the different initializations. Masked Layer Distillation is more robust.

Configuration			ResNet34 to ResNet18		VGG13 to VGG19		ResNet101 to ResNet50	
	Initialization	Batch size	Scratch	MLD	Scratch	MLD	Scratch	MLD
1	$\theta_i \sim \mathcal{N}(0,1)$	256	92.06	**94.22**	90.88	**92.55**	93.00	**93.59**
2	$\theta_i \sim \mathcal{N}(0,1)$	1024	71.61	**92.73**	72.86	**91.91**	-	-
3	$\theta_i \sim \mathcal{N}(1,1)$	256	81.13	**94.19**	38.61	**92.26**	41.96	**92.29**
4	$\theta_i \sim \mathcal{N}(1,1)$	1024	22.24	**92.94**	18.16	**92.17**	-	-
5	$\theta_i = 1$	256	63.50	**94.00**	33.34	**92.31**	31.76	**92.34**
6	$\theta_i = 1$	1024	18.25	**91.10**	13.39	**92.16**	-	-
7	Xavier Uniform	256	**94.34**	94.33	**92.67**	92.31	**94.03**	93.65
8	Xavier Uniform	1024	**92.91**	92.38	90.42	**91.29**	-	-

confirm our hypothesis that Masked Layer Distillation is a regularizer as well as a tool to accelerate training. Across nearly all configurations, poor initialization choices have an outsize effect on training from scratch but close to no effect on models trained using our method. We propose a theory for Masked Layer Distillation's robustness to initialization schemes:

Initializations with large variance likely produce more intermediate pixels that are sensitive to noise, as parameter magnitudes may be very large and therefore compound errors. The mask learns to hide these pixels, which prevents the poorly initialized parameters from having undue influence on the loss. The Lottery Ticket Hypothesis suggests that large networks contain "lucky," significantly smaller subnetworks that are sufficient to maintain the accuracy of the full model [14]. Our masking mechanism likely weights the pixels in a way that heavily encourages optimization of the lucky subnetwork instead of the parameters that compound errors.

For initializations with 0 variance, the mask prevents all units in a layer from having the same gradient by assigning a different weight to each pixel in the activation map. The naive L_1 loss would take $\|f(x) - g(x)\|_1$ where f is a source layer and g is a target layer. $f(x)$ is a constant function because the source weights are frozen, so the gradient would reduce to $-x$ if all of g's weights were the same, causing each hidden unit to have the same gradient vector for each input. If the weights are initialized to be the same value and every gradient step is the same for each hidden unit in the same layer, then the neurons in that layer never differ from each other throughout training; the layer's capacity never exceeds that of one neuron. Since a single neuron does not have sufficient capacity to learn meaningful representations on CIFAR-10, performance suffers dramatically, which is the reason from-scratch training (Configuration 6 in Table 1) barely does better than guessing.

However, with the introduction of the mask loss, $\|(1 - ReLU(Mask)) * (f(x) - g(x))\|_1$, the gradient becomes $(ReLU(Mask) - 1) * x$. Since the mask values corresponding to each hidden unit are different, the gradients will be different for each unit, alleviating the main issue that comes with constant initializations. The different weights of the mask make each gradient step different for different neurons, which allows the hidden units to learn different representations and achieve higher performance, as shown in Table 1.

4.2 Rectifying Distillation Degradation

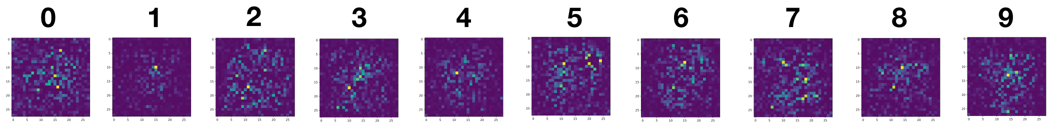
Recent work [13] has found that contrastive learning does not work well on small networks. Since Masked Layer Distillation also relies on internal representations, we were curious to see how our method would perform on smaller models whose distillation / compression properties are not as well studied as ResNets. We were specifically interested in EfficientNet and MobileNet because previous work has indicated that depthwise separable convolutions (DSCs) are less resilient to compression [9] [23], and these two architecture families contain many DSCs.

We applied Masked Layer Distillation to EfficientNet-B0 and MobileNetV3-Small and evaluated their performance on ImageNet. We manually chose breakpoints in both networks where we compared the intermediate outputs of the teacher and student models and learned a mask. We kept the choice of breakpoints the same across different configurations. One could experiment with different breakpoint

Table 2: Masked Layer Distillation (MLD) vs Label Refinery (LR) on ImageNet. MLD significantly improves the accuracy of EfficientNet and MobileNet comparing to label refinery (up to 3.6%).

	Configuration		MLD + LR		LR			
	Model	% Deleted	Accuracy	Epochs	Accuracy	Epochs	Speedup	Teacher Accuracy
1	MobileNet	0	**63.9**	21	60.4	120	5.7x	67.4
2	MobileNet	7.7	**63.8**	34	60.2	107	3.1x	67.4
3	MobileNet	15.4	**62.1**	45	59.2	113	2.5x	67.4
4	MobileNet	23.1	**61.1**	53	58.9	117	2.2x	67.4
5	EfficientNet	0	**74.1**	26	71.6	80	3.1x	77.7
6	EfficientNet	6.3	**74.0**	26	71.5	74	2.9x	77.7
7	EfficientNet	12.5	**73.9**	25	71.3	76	3.0x	77.7
8	EfficientNet	25.0	**72.7**	26	71.0	80	3.1x	77.7
9	EfficientNet	37.5	**71.2**	26	70.5	80	3.1x	77.7

Figure 3: **Visualization of learned masks** For MNIST, we visualize learned masks. We see that the masks cluster around the center of the image and trace a rough outline of objects (see 1 or 2).



positions and the number of breakpoints to shoot for state-of-the-art accuracy, but we found that even our arbitrary choices presented competitive results.

We also challenged Masked Layer Distillation by applying it to edited versions of MobileNet and EfficientNet in which we randomly deleted several blocks. Arbitrary scaling of network depth often results in sub-optimal accuracy, and using the same teacher architecture as the student often degrades distillation performance [19] [18]. In addition, some initialization schemes are sensitive to deletions; for example Xavier initialization initializes weights depending on the number of units in the adjacent layer [15], a process that would obviously be perturbed in the presence of block deletion. We were interested whether Masked Layer Distillation on DSCs is robust even in such adverse circumstances. Note that our goal here was not compression, as compression often requires a carefully chosen teacher model and careful tuning of depth, width, and resolution to avoid destabilizing the network. We instead wanted to examine how our method performed even when faced with very coarse block deletions and the model itself as a teacher.

As shown in Table 2, Masked Layer Distillation outperforms distillation by a large margin in both speed and accuracy across a broad range of architectures and edits. Each student model is guided by a full teacher model of the same type i.e., all EfficientNet-B0 students have an EfficientNet-B0 teacher with no block deletions.

4.3 SOTA Comparisons

Although Label Refinery is a competitive distillation technique, it is unclear how it compares to other recent advances in knowledge transfer. To better understand the performance of Masked Layer Distillation, we additionally evaluated EfficientNet and MobileNet distillation using Softmax Regression Representation Learning (SRRL) [29], which has been shown to achieve state-of-the-art performance on ImageNet distillation and outcompetes [17] [31] [25] [20] [16]. Results are shown in Table 3. Using SRRL as our tuning method, we generally perform +1% higher than with Label Refinery, and we reconfirm that Masked Layer Distillation beats or matches existing end-to-end methods in significantly fewer epochs.

4.4 Fast ResNet Training

To evaluate the efficiency of our training technique, we used a pretrained ResNet-34 to train a randomly initialized ResNet-18. We evaluate performance on the Top-1 and Top-5 accuracy on the

Table 3: Masked Layer Distillation (MLD) vs Softmax Regression Representation Learning (SRRL) on ImageNet. MLD shows significant improvement over SOTA method SRRL as well (by up to 2.2%.

	Configuration		MLD + SRRL		SRRL			
	Model	% Deleted	Accuracy	Epochs	Accuracy	Epochs	Speedup	Teacher Accuracy
1	MobileNet	0	**64.4**	25	62.1	100	4.0x	67.4
2	MobileNet	7.7	**63.8**	33	61.7	94	2.8x	67.4
3	MobileNet	15.4	**62.8**	32	61.8	80	2.5x	67.4
4	MobileNet	23.1	**62.3**	33	61.9	87	2.6x	67.4
5	EfficientNet	0	**75.1**	25	73.1	100	4.0x	77.70
6	EfficientNet	6.3	**74.9**	25	72.8	98	3.9x	77.16
7	EfficientNet	12.5	**74.6**	26	73.1	100	3.8x	76.61
8	EfficientNet	25.0	**73.5**	26	72.7	97	3.7x	75.52
9	EfficientNet	37.5	**72.0**	32	72.0	99	3.1x	74.45

Table 4: ResNet distillation results on ImageNet. Masked Layer Distillation (MLD) is compared against Label Refinery (LR) and from scratch training (Scratch). MLD shows significantly faster training time (by 4.4x) over training from scratch.

	Configuration		MLD + LR		LR			Scratch		
	Student	Teacher	Accuracy	Epochs	Accuracy	Epochs	Speedup	Accuracy	Epochs	Speedup
1	ResNet18	ResNet34	70.1	19	70.3	61	3.2x	69.8	83	4.4x
2	ResNet34	ResNet18	73.3	64	-	-	-	73.2	90	1.4x

validation set of ImageNet 2012 dataset. The baseline accuracy for the pre-trained ResNet-34 used in our experiment is **73.3%** [10] and the baseline for ResNet-18 is **69.76%**[10].

In order to distill ResNet-34 to ResNet-18, we perform the following source-to-target transformations: we distill both blocks 1.1 and 1.2 from ResNet-34 to block 1.1 in ResNet-18; we next distill blocks $\{2.0, 2.1\}$ to 2.0, $\{2.2, 2.3\}$ to 2.1, $\{3.0, 3.1, 3.2\}$ to 3.0, $\{3.3, 3.4, 3.5\}$ to 3.1, $\{4.0\}$ to block 4.0 and finally $\{4.1, 4.2\}$ to block 4.1.

For comparison, we trained ResNet-18 using random initialization as well as using label refinery [10] (source network of ResNet-34) for 90 epochs. Our re-implementation of ResNet-18 (random initialization) has a testing accuracy of **69.15%** and ResNet-18 (label refinery) has a validation accuracy of 70.3%. We show that we can achieve **70.1%** accuracy in 19 epochs of training, **4.4x** times faster than training from scratch and 3.2x times faster than model distillation method.

We also note that our results are competitive with [5], who achieved a 3.6x speedup *with* a distributed cluster, compared to our method that can achieve 4.4x speedup *without* additional system complexity. All of our experiments in this section were run on a single node.

We additionally challenged Masked Layer Distillation to distill knowledge from a ResNet-18 to ResNet-34. In 64 epochs of Masked Layer Distillation with additional global fine tuning, we were able to achieve 73.3% accuracy, compared to 73.2% accuracy which we achieved from 90 epochs from scratch training.

These experiments indicate to us that Masked Layer Distillation is not just a tool for exploring minor edits during architecture search; it is also a general tool for transferring knowledge to and from vastly different source and target layer groups.

5 Conclusion

We introduce a novel distillation method, Masked Layer Distillation, which performs significantly better than current competitive works across a range of aggressive edits on EfficientNet and MobileNet. We show that a paradigm shift away from the excessive focus on ResNet and ResNet-like models in the distillation community is long overdue. The state-of-the-art backbone on ImageNet has been EfficientNet for a few years now, yet even the smallest base models from this family incurs large performance degradation when attempting to perform knowledge transfer on ImageNet. This is especially relevant because at the time of this writing, the top performing methodology on ImageNet is Meta Psuedo Labels [21], which draws heavily on both the EfficientNet backbone and distillation. We expect that our work on recouping distillation losses may help future practitioners on achieving state-of-the-art accuracy on ImageNet, given Masked Layer Distillation's versatility and robustness.

References

[1] Ash, J. T. and Adams, R. P. On the difficulty of warm-starting neural network training. *CoRR*, abs/1910.08475, 2019. URL `http://arxiv.org/abs/1910.08475`.

[2] Ba, J. and Caruana, R. Do deep nets really need to be deep? In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2654–2662. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf`.

[3] Bagherinezhad, H., Horton, M., Rastegari, M., and Farhadi, A. Label refinery: Improving imagenet classification through label progression. *CoRR*, abs/1805.02641, 2018. URL `http://arxiv.org/abs/1805.02641`.

[4] Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

[5] Blakeney, C., Li, X., Yan, Y., and Zong, Z. Parallel blockwise knowledge distillation for deep neural network compression. *CoRR*, abs/2012.03096, 2020. URL `https://arxiv.org/abs/2012.03096`.

[6] Buciluă, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pp. 535–541, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150464. URL `http://doi.acm.org/10.1145/1150402.1150464`.

[7] Chan, T. F. and Shen, J. J. *Image processing and analysis: variational, PDE, wavelet, and stochastic methods*, volume 94. Siam, 2005.

[8] Chen, T., Goodfellow, I. J., and Shlens, J. Net2net: Accelerating learning via knowledge transfer. *CoRR*, abs/1511.05641, 2015.

[9] Chin, T.-W., Chuang, P. I.-J., Chandra, V., and Marculescu, D. One weight bitwidth to rule them all, 2020.

[10] Chintala, S. et al. PyTorch torchvision models, 2016. URL `https://pytorch.org/docs/stable/torchvision/models.html`.

[11] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

[12] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL `https://arxiv.org/abs/2010.11929`.

[13] Fang, Z., Wang, J., Wang, L., Zhang, L., Yang, Y., and Liu, Z. SEED: self-supervised distillation for visual representation. *CoRR*, abs/2101.04731, 2021. URL `https://arxiv.org/abs/2101.04731`.

[14] Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[15] Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[16] Heo, B., Kim, J., Yun, S., Park, H., Kwak, N., and Choi, J. Y. A comprehensive overhaul of feature distillation, 2019.

[17] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL `http://arxiv.org/abs/1503.02531`.

[18] Koratana, A., Kang, D., Bailis, P., and Zaharia, M. LIT: block-wise intermediate representation training for model compression. *CoRR*, abs/1810.01937, 2018. URL `http://arxiv.org/abs/1810.01937`.

[19] Mishra, A. and Marr, D. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy, 2017.

[20] Park, W., Kim, D., Lu, Y., and Cho, M. Relational knowledge distillation, 2019.

[21] Pham, H., Xie, Q., Dai, Z., and Le, Q. V. Meta pseudo labels. *CoRR*, abs/2003.10580, 2020. URL `https://arxiv.org/abs/2003.10580`.

[22] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.

[23] Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L., and Aleksic, M. A quantization-friendly separable convolution for mobilenets. *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, Mar 2018. doi: 10.1109/emc2.2018.00011. URL `http://dx.doi.org/10.1109/EMC2.2018.00011`.

[24] Sun, C., Shrivastava, A., Singh, S., and Gupta, A. Revisiting unreasonable effectiveness of data in deep learning era. *CoRR*, abs/1707.02968, 2017. URL `http://arxiv.org/abs/1707.02968`.

[25] Tian, Y., Krishnan, D., and Isola, P. Contrastive representation distillation. *CoRR*, abs/1910.10699, 2019. URL `http://arxiv.org/abs/1910.10699`.

[26] Wei, T., Wang, C., Rui, Y., and Chen, C. W. Network morphism. In *International Conference on Machine Learning*, pp. 564–572, 2016.

[27] Wei, T., Wang, C., and Chen, C. W. Modularized morphing of neural networks. *CoRR*, abs/1701.03281, 2017.

[28] Yam, J. Y. and Chow, T. W. A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing*, 30(1):219–232, 2000. ISSN 0925-2312. doi: https://doi.org/10.1016/S0925-2312(99)00127-7. URL `https://www.sciencedirect.com/science/article/pii/S0925231299001277`.

[29] Yang, J., Martinez, B., Bulat, A., and Tzimiropoulos, G. Knowledge distillation via softmax regression representation learning. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=ZzwDy_wiWv`.

[30] Zagoruyko, S. and Komodakis, N. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL `http://arxiv.org/abs/1605.07146`.

[31] Zagoruyko, S. and Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, 2017.