NEIGHBOURHOOD DISTILLATION: ON THE BENEFITS OF DISTILLING SMALL MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge Distillation is a popular method to reduce model size by transferring the knowledge of a large teacher model to a smaller student network. We show that it is possible to independently replace sub-parts of a network without accuracy loss. Based on this, we propose a distillation method that breaks the end-to-end paradigm by splitting the teacher architecture into smaller sub-networks - also called neighbourhoods. For each neighbourhood we distill a student independently and then merge them into a single student model. We show that this process is significantly faster than Knowledge Distillation, and produces students of the same quality. From Neighbourhood Distillation, we design Student Search, an architecture search that leverages the independently distilled candidates to explore an exponentially large search space of architectures and locally selects the best candidate to use for the student model. We show applications of Neighbourhood Distillation and Student Search on CIFAR-10 and ImageNet models on model reduction and sparsification problems. Our method offers up to $4.6\times$ speed-up compared to end-to-end distillation methods while retaining the same performance.

1 INTRODUCTION

As Deep Neural Networks improve on challenging tasks, they also become deeper and bigger. BERT, a state-of-the art language models has 340M parameters (Devlin et al., 2018), SE-Net an image classification model 145M (Hu et al., 2018). However, the sizes of such models make them impractical for applications that require real-time inference with limited computational resources. Model compression seeks to take advantage of these trained cumbersome models by reducing their size while minimally impacting their performance. Knowledge Distillation (Bucilua et al., 2006; Hinton et al., 2015), in particular, transfers the knowledge of a large teacher model to a lightweight student model by training the student to reproduce the logits output of its teacher. However, designing an optimal Student architecture, one that most approaches the accuracy of the Teacher while satisfying the resource constraints (size, latency, etc) is a challenging task. The potential search space is endless. Given the large search space of possible architectures, designing a Student network from scratch can be prohibitively expensive in terms of computational resources and time.

Our work seeks to overcome this limitation by drastically reducing the cost of finding and distilling a good student model. We present Neighbourhood Distillation, a novel distillation method that breaks a deep architecture into smaller components that are distilled independently. There are multiple benefits for working on small neighbourhoods as compared to full models: training a Neighbourhood takes significantly less compute than a larger model; during training gradients in a Neighbourhood only back-propagate through a small number of layers making it unlikely that neighbourhoods suffer from vanishing or exploding gradients. By breaking a model into smaller units, training can be done in parallel, significantly reducing wall-time for training as well as enabling training on CPUs which are cheaper than custom accelerators but are seldom used in Deep Learning as they are too slow for larger models.

In addition to speeding-up the distillation process, Neighbourhood Distillation can be used to efficiently find a student model with favorable trade-off between size and accuracy. *Student Search* locally selects candidates that are trained independently by Neighbourhood Distillation. This allows a much faster exploration of an exponential space, as it doesn't require retraining each combination

of neighbourhoods from scratch. The number of models to train is not only drastically reduced, but the neighbourhoods are much smaller than the final model and their training can be parallelized.

We show that Neighbourhood Distillation and Student Search can be successfully applied to different experimental settings. We conduct experiments on CIFAR-10 and ImageNet with the goal of optimizing the architecture of the teacher by reducing the number of parameters or sparsifying the weights.

We sum up our contributions below:

- We provide empirical evidence of the thresholding effect, a phenomenon that highlights deep neural networks' resilience to perturbations of their weights. This motivates the idea of distilling networks in a non-end-to-end manner.
- We present Neighbourhood Distillation, a distillation method that independently trains subcomponents of the student and retains the performance of Knowledge Distillation while being faster.
- Using Neighbourhood Distillation, we design an efficient search algorithm that efficiently explores an exponential number of possibilities to find an optimal student architecture for distillation.

2 Related Work

We highlight two research directions that aim at producing high-accuracy, resource-efficient networks.

Neural Architecture Search Training smaller networks that reach state-of-the art performance requires careful design of network architectures. While efficient building blocks have been introduced in the past (Howard et al., 2017; Zhang et al., 2017a; Iandola et al., 2016), designing optimal architectures is a tedious task. Neural Architecture Search methods automate the design process by exploring a given search space (Liu et al., 2018; Pham et al., 2018; Liu et al., 2017) and have been successfully used to find more efficient and better performing models (Tan & Le, 2019; Zoph et al., 2017).

Model compression Another approach relies on the observation that models are overparameterized and can be compressed while preserving accuracy through processes such as quantization (Wei et al., 2018), weight pruning (LeCun et al., 1990) or distillation (Hinton et al., 2015). Our work focuses on distillation, a general technique that has been successfully applied to vision (Crowley et al., 2018) and language problems (Hahn & Choi, 2019). While various methods have been developped to improve the supervision between teacher and student (Aguilar et al., 2019; Romero et al., 2014; Zagoruyko & Komodakis, 2016; Yim et al., 2017; Hyun Lee et al., 2018; Zhang et al., 2017b; Mirzadeh et al., 2019; Jin et al., 2019; Cho & Hariharan, 2019) all these methods have focused on easing the distillation between a fixed student and teacher.

At the intersection of both worlds, recent papers study how to apply Knowledge Distillation to improve Neural Architecture Search methods (Trofimov et al., 2020; Macko et al., 2019) and find better suited students for a fixed teacher (Kang et al., 2019; Liu et al., 2020). These methods rely on distilling the student end-to-end. Closer to our work, Li et al. (2020) divide a supernet into blocks and use Knowledge Distillation to train it. Unlike in our work, they focus on extracting the architectural knowledge from the teacher, ignoring the parameters learned during the search process. Our work also aims at providing an improvement to Knowledge Distillation that can be used of NAS.

Non end-to-end training Our work is also inspired by recent advances in non end-to-end training of deep neural networks. One class of non end-to-end learning methods relies on splitting a deep network into gradient-isolated modules trained with local objectives (Löwe et al., 2019; Nøkland & Eidnes, 2019). Layerwise training (Belilovsky et al., 2018; Huang et al., 2017) also splits the target network into modules that are sequentially trained in a bottom-up approach. These approaches differ from ours as modules still depend on each other during the forward pass, while our neighbourhoods are distilled independently.

3 MOTIVATION

Due to their size, modern neural networks are usually overparameterized. Their learned representations are redundant, which allows for compression techniques such as distillation to reduce the size of trained networks. Recent empirical studies conducted on ResNets (Zhang et al., 2019; Veit et al., 2016) also showed that it is possible to drop or reset layers in a trained network without hurting their performance. Therefore we hypothesize that less drastic modifications in a network, such as replacing part or all of their sub-components by imperfect approximations will not result in dramatic error accumulation. In the following section, we provide empirical evidence of an interesting property that supports this: sub-components of a trained model may be perturbed without damaging the model's accuracy, as long as individual local errors remain under a certain threshold. We call this phenomenon the thresholding effect.

We first define formally the notion of neighbourhood that will be used throughout the rest of the paper. we consider a teacher network $\mathcal{T}$ and a student network $\mathcal{S}$. Modern deep neural networks are built by stacking a succession of blocks of operations such as convolutions and non-linear layers. We express this by defining $\mathcal{T}$ as a composition of n sub-networks.

$$\begin{aligned} \forall i \in \{1 \dots n\}, \mathcal{T}_i : \mathbb{R}^{F_i} \rightarrow \mathbb{R}^{F_{i+1}} \\ \mathcal{T} = \mathcal{T}_n \circ \mathcal{T}_{n-1} \circ \cdots \circ \mathcal{T}_1 \end{aligned} \tag{1}$$

We call neighbourhood the portion of the network that is delimited by one sub-component $\mathcal{T}_i$. This neighbourhood represents a logical construction block in the network and may be replaced by variants of its architecture as long as the input and output shapes match.

The question we set out to answer is the following. Imagine we want to replace part or all of the neighbourhoods by imperfect approximations, how good do these approximations need to be to prevent a drastic loss of performance in the modified model?

We consider different pre-trained models and estimate how their accuracy is impacted by perturbations of the network's intermediate features. To do so, we perturb each neighbourhood by artificially introducing some gaussian noise of amplitude ϵ to their outputs.

$$\begin{aligned} \mathcal{S}_i = \mathcal{T}_i + \delta \\ \delta \sim \mathcal{N}(0, \epsilon^2) \end{aligned} \tag{2}$$

The student neighbourhoods are then used to reconstruct a full student network $\mathcal{S} = \mathcal{S}_n \circ \mathcal{S}_{n-1} \cdots \circ \mathcal{S}_1$ which we evaluate.

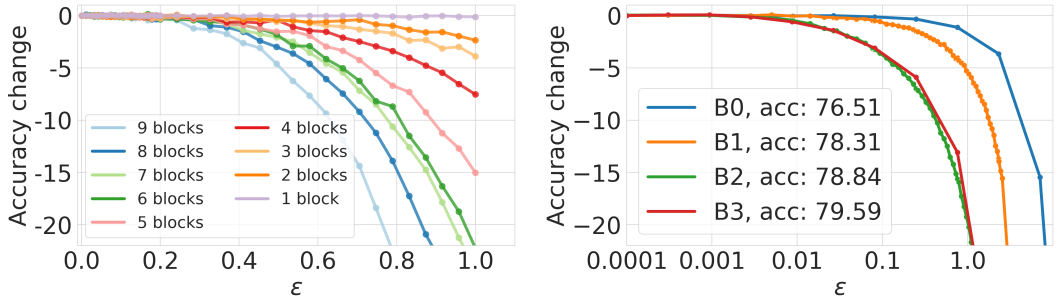
On CIFAR-10 (Krizhevsky et al., 2009), we train a ResNetV1-20 (He et al., 2016) model and define a neighbourhood as one bottleneck block which consists of a two-layer convolutional network and a skip connection. We reiterate a similar experiment on a large-scale dataset. On ImageNet (Deng et al., 2009), we train several EfficientNet (Tan & Le, 2019) models and define a neighbourhood as one mobile inverted bottleneck block.

Figure 1 shows how errors of different amplitudes accumulate across networks when replacing some or all neighbourhoods by approximations. In particular, we consistently witness a thresholding effect in all networks. When the amplitude ϵ of the noise is small enough, the final accuracy of the network is not impacted by accumulated perturbations. This threshold appears to depend on the number of neighbourhoods.

Our experiments on the thresholding effect show that it is possible to locally replace sub-components of a model without hurting the performance of the reconstructed model. This observation is what motivates Neighbourhood Distillation: neighbourhoods trained to approximate their teacher outputs can be used to reconstruct student networks with no or limited accuracy drop.

4 NEIGHBOURHOOD DISTILLATION

Neighbourhood Distillation leverages the representation of a network as a stack of neighbourhoods. Instead of distilling the entire network end-to-end, we use each neigbourhood as an independent teacher. For each teacher neighbourhood, a student neighbourhood $\mathcal{S}_i$ is trained to match the outputs



(a) ResNetV1-20 with 91.61% accuracy on CIFAR-10. Each line represents a different number of perturbed blocks.

(b) EfficientNet models trained on ImageNet. Each line represents a different EfficientNet model for which all neighbourhoods have been perturbed.

Figure 1: We perturb the intermediate outputs at regular locations in the network using a gaussian noise of amplitude ϵ and measure the effect of these perturbations on the accuracy of the model. We show that accuracy remains stable (accuracy change close to 0) as long as ϵ remains under a certain threshold. The threshold differs between network architectures and number of perturbed neighbourhoods. Note that for all models, even when perturbing all neighborhoods, there is still a range for which there is virtually no loss in accuracy.

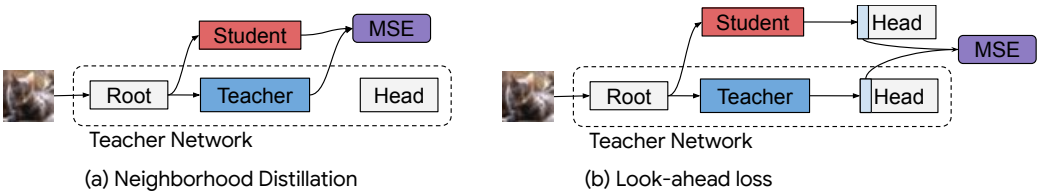


Figure 2: Computation graphs for Neighbourhood Distillation. (a) Neighbourhood Distillation: the Teacher and the Student neighbourhoods receive activations from the Root of the Teacher Network. The Student Neighbourhood is trained to reproduce the output of the Teacher. (b) Look-ahead loss: the Teacher and Student outputs are propagated to the head of the Teacher Network and additional activations between Teacher and Student networks are compared. The look-ahead loss gives an additional training signal for the Student to reproduce the Teacher.

of its teacher by minimizing the Mean Square Error between their produced features. The training inputs are obtained by forward-propagation of the original training images through the first part of the teacher network $\mathcal{T}_{i-1} \circ \cdots \circ \mathcal{T}_1$. The computational graph for Neighbourhood Distillation is shown in Figure 2. These activation maps can be pre-computed and stored before training. This makes training each student Neighbourhood extremely fast as one does not need to compute activations through a big model in each training step.

Different student neighbourhoods can be distilled *independently from each other* and composed into a final student model $\mathcal{S} = \mathcal{S}_n \circ \cdots \circ \mathcal{S}_1$. The final student model is then fine-tuned by optimizing the Knowledge Distillation loss from Equation 3.

$$\mathcal{L}_{KD}(\mathbf{x}, \mathbf{y}) = \mathcal{L}_{CE}(\hat{\mathbf{y}}^{\mathcal{T}}_{\tau}, \hat{\mathbf{y}}^{\mathcal{S}}_{\tau}) + \lambda \mathcal{L}_{CE}(\mathbf{y}, \hat{\mathbf{y}}^{\mathcal{S}}_{1}) \tag{3}$$

$$\hat{\mathbf{y}}^{\mathcal{T}}_{\tau} = softmax(\frac{\mathcal{T}(\mathbf{x})}{\tau}) \tag{4}$$

where $\mathcal{L}_{CE}$ is the standard cross-entropy loss.

While the idea of minimizing the mean-square error between output features is similar to Hint-Training (Romero et al., 2014), our method is more general as we do not limit ourselves to distill one prefix sub-network. Neighbourhood Distillation also provides more flexibility by enabling distillation on many sub-networks in parallel. Note also that although student neighbourhoods are distilled independently, we ultimately use them to build a bigger student model. As we want to be able to

compose them properly, we constrain the student Neighbourhood $\mathcal{S}_i$ to have the same input and output dimensions as its teacher.

Look-ahead losses In order to provide additional training signals to the student, we also consider using look-ahead losses. In addition to forcing the output of $\mathcal{S}_i$ to be close to the output of $\mathcal{T}_i$, we feed each to the next teacher Neighbourhood $\mathcal{T}_{i+1}$ and minimize the Mean Square Error of its output:

$$\mathcal{L}_{i,1} = \frac{1}{F_{i+2}} \mathbb{E}_{a \sim p(a)} \left[\| \mathcal{T}_{i+1}(\mathcal{S}_i(a)) - \mathcal{T}_{i+1}(\mathcal{T}_i(a)) \|_2^2 \right] \quad (5)$$
$$a = \mathcal{R}_i(x), x \sim \mathcal{D}_{\text{train}}$$

The look-ahead loss is shown in Figure 2. These losses can be combined to form a total loss of:

$$\mathcal{L}_i = \mathcal{L}_{i,0} + \sum_{j=1}^{n-i} \alpha_j \mathcal{L}_{i,j} \quad (6)$$

where α_j are hyper-parameters. While lookahead losses may improve the accuracy of the distilled model by providing additional training supervision, it also leads to increased computational cost due to needing to propagate inputs and gradients through a higher number of layers.

5 Student Search

Usually, when performing Knowledge Distillation, the Student network is a variant of the Teacher network with a smaller number of parameters (Hinton et al., 2015; Romero et al., 2014; Zhang et al., 2017b). Other candidate architectures with the same number of parameters could better capture the teacher's knowledge but searching for the ideal student architecture would require retraining different models from scratch and be computationally expensive. Leveraging Neighbourhood Distillation's ability to distill sub-networks independently, we propose Student Search, an architecture search method that looks for the best local modifications to apply on smaller components of an architecture.

Formally, let's note $\mathcal{C}_i$ the set of possible candidates for a given neighbourhood $\mathcal{T}_i$. This set could contain variants of the same architecture with different parameters like the size of the intermediate layers, the level of sparsity, the non-linearity used. The number of possible student model architectures to distill and evaluate would be $\prod_{i=1}^{n} |\mathcal{C}_i|$. Student Search leverages Neighbourhood Distillation by first distilling all $\sum_{i=1}^{n} |\mathcal{C}_i|$ neighbourhoods completely independently.

We would also like to be able to evaluate the quality of each candidates independently, so as to avoid having to evaluate all $\prod_{i=1}^{n} |\mathcal{C}_i|$ possible models. To this end, we define a quality function $q_i : \mathcal{C}_i \rightarrow \mathbb{R}$ that quantifies how good a student candidate is, independently from the other students. For example, we will consider the MSE between the student and teacher on a validation set. We will further discuss this choice in the experiments section.

Student Search selects a candidate for each Neighbourhood by solving a constrained optimization problem that seeks to minimize the size of the student while maximizing the quality of the selected candidates. The selected candidate neighbourhoods are then combined together to form the Student Network, which may then be fine-tuned. In practice, we find that solving the above optimization problem approximately is enough to yield a good student architecture.

6 Experiments

We conduct experiments on ResNet models trained on two benchmark classification datasets: CIFAR-10 (Krizhevsky et al., 2009), and ImageNet (Deng et al., 2009). Two different settings are considered for Neighbourhood Distillation and Student Search. We first aim at reducing the number of model parameters by modifying the filter size of the residual units with a multiplier that we only apply to intermediate layers. We then demonstrate how Neighbourhood Distillation can be used with one-layer neighbourhoods by sparsifying the teacher network's weights.

Table 1: CIFAR-10 test accuracies for different values of k on ResNet-20. Our Neighbourhood Distillation method reaches the same accuracy as Knowledge Distillation (KD) and performs better than retraining from scratch (CE). As k decreases, fine-tuning becomes critical in order to recover good accuracy.

k	# PARAMS	CE	KD	NDISTILL	NDISTILL + FT
1	269K	91.61	**91.92**	91.57	91.57
0.9	239K	91.31	**91.74**	91.28	91.72
0.8	213K	90.76	91.48	90.7	**91.72**
0.75	202K	90.89	91.57	87.57	**91.68**
0.6	160K	90.49	90.59	90.76	**91.07**
0.5	136K	90.17	90.98	88.25	**90.98**
0.4	105K	89.34	89.75	83.55	**90.00**

Table 2: Distillation results on ResNet-50 for neighbourhoods of 2 and 3 layers with different multipliers. Neighbourhood Distillation (ND) performs similarly to Knowledge Distillation (KD) and better than retraining from scratch (CE). As the gap between the student and the teacher increases, fine-tuning becomes necessary to recover good model accuracy.

MODEL	# PARAMS	CE	KD	NDISTILL	NDISTILL + FT
TEACHER RN50	23 M	76.18	-	-	-
RN50-0.75-2	21 M	75.42	76	72.79	**76.35**
RN50-0.75-3	18 M	75.08	75.5	64.75	**76.07**
RN50-0.50-2	17 M	73.59	**74.91**	68.8	74.87

We first explore the architecture space of ResNetV1 models by expanding or shrinking the number of filters within a bottleneck block with a multiplier k applied to inner layers. On CIFAR-10, we consider a ResNetV1-20 model as the teacher model where a neighbourhood is a two-layer residual block. For ImageNet, we consider ResNetV1-50 model as the teacher where the neighbourhood is a three-layer residual block. For both models, weights that are not affected by the multiplier (e.g: the first and last layer of the model) are kept as-is.

After the initial distillation step, we recompose the student neighbourhoods that have the same multiplier into a final student model. The student model is then fine-tuned on the training set using the usual Knowledge Distillation loss until convergence. Table 1 and Table 2 compare the final accuracies obtained for different student models trained with Cross-Entropy loss, Knowledge Distillation or Neighbourhood Distillation. We observe that our Neighbourhood Distillation method yields similar result as Knowledge Distillation, and both distillation methods improve on training with Cross-Entropy loss.

Timing We timed Neighbourhood Distillation and compare its runtime to that of Knowledge Distillation and report the total GPU time for each, i.e. the total time that would have been needed for one GPU P100 to run. Results are reported in Table 3a and 3b. Neighbourhood Distillation is **3.6**$\times$ faster on ImageNet and **2.3**$\times$ on CIFAR-10. The end-to-end fine-tuning step is the main computational bottleneck but is faster than using Knowledge Distillation from scratch. Student neighbourhoods learn a meaningful approximation of their respective teachers, which allows the fine-tuning step to converge faster than Knowledge Distillation from scratch.

Ablation Study We perform an ablation study on the CIFAR-10 model to show how look-ahead and fine-tuning impact our Neighbourhood Distillation results. We experiment with combining two look-ahead losses, i.e $\forall j \geq 3, \alpha_j = 0$ and train each neighbourhood either without look-ahead or with two-neighbourhood lookahead. Models are evaluted before and after fine-tuning and test accuracies are reported in Table 4. We observe that training neighbourhoods with look-ahead leads to better accuracy before fine-tuning. Fine-tuned models always perform better than their non-fine-tuned counterparts. While there is no significant difference between the no look-ahead and the 2-look-ahead fine-tuned models, we also noticed that neighbourhoods distilled with look-ahead losses

Table 3: Comparison of GPU time between Knowledge Distillation and Neighbourhood Distillation. In both cases, fine-tuning is the main computational bottleneck but is still faster than Knowledge Distillation.

(a) On CIFAR-10, Neighbourhood Distillation is **2.3**× faster than Knowledge Distillation.

METHOD		TRAINING TIME (H)
NDISTILL	1-UNIT	0.035
	ALL UNITS SEQUENTIALLY	0.32
	FINETUNING	1.07
	ALL UNITS + FINETUNING	**1.39**
KD		3.2

(b) On ImageNet, Neighbourhood Distillation is **3.6**× faster than Knowledge Distillation.

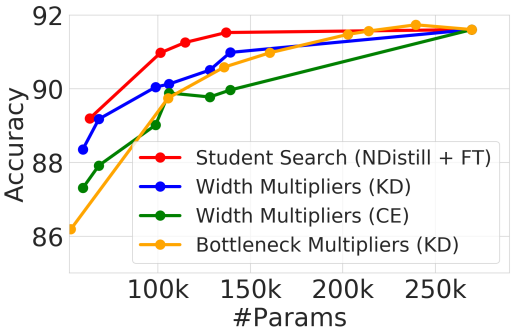
METHOD		TRAINING TIME (H)
NDISTILL	1-UNIT	1.42
	ALL UNIT SEQUENTIALLY	22.75
	FINETUNING	33
	ALL UNIT + FINETUNING	**56**
KD		200

Table 4: Ablation Study. Impact of fine-tuning (FT) and look-ahead (2-LA) on final test accuracy for ResNet-20. While fine-tuned model always perform better than pre-finetuned models, the use of look-ahead tightens the gap between pre and post finetuned accuracy.

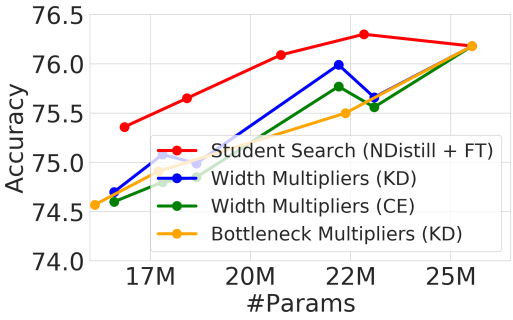
k	NDISTILL	NDISTILL + FT	2-LA NDISTILL	2-LA NDISTILL + FT
0.90	91.1	**91.96**	91.37	**92.12**
0.75	89.74	**91.86**	90.61	**91.83**
0.50	85.7	**91.31**	87.08	**91.47**

converge faster during the fine-tuning step - 30k iterations instead of 50k. Look-ahead losses can therefore be used to balance the cost of Neighbourhood distillation and the cost of fine-tuning.

Student Search Apart from speeding-up distillation, Neighbourhood Distillation can also be used to efficiently find student architectures. We use Student Search to select a student architecture from the space of ResNet variants built with different multipliers applied to each block. For each neighbourhood, we independently distill 10 candidates, obtained by varying the bottleneck multiplier k in $\{0.1, 0.2, ..0.9, 1.0\}$. The quality of a candidate $c_i(k)$ is measured by the accuracy of the partial model $T_n \circ ...T_{i+1} \circ c_i(k) \circ T_{i-1} \circ T_1$. We solve the optimization problem with a greedy method by selecting for each neighbourhood the smallest candidate that leads to an accuracy drop of less than $x\%$.



(a) ResNetV1-20 Student Search on CIFAR-10



(b) ResNetV1-50 Student Search on ImageNet

Figure 3: Trade-off curve between number of parameters and final accuracy for ResNetV1-20 and ResNetV1-50 models. Models found with Student Search are obtained by recombining units of different sizes that were distilled independently with Neighbourhood Distillation. Width multipliers refers to applying a uniform multiplier on all blocks. Bottleneck multipliers refers to applying a uniform multiplier on the inner layers of all blocks. Student Search efficiently finds better student architectures than naively applying uniform transformations to the teacher.

Figure 3a and 3b shows how our Student Search compares to using a uniform multiplier on all layers when searching for a student architecture for Knowledge Distillation. Width multiplier refers

to applying the same multiplier to *all* the bottleneck layers of the model. This setting cannot be explored with Neighbourhood Distillation as it modifies the input/output behavior of the neighbourhoods. Bottleneck multiplier refers to applying the same multiplier to the inner layers of each residual block: this setting is equivalent to applying Neighbourhood Distillation with the same multiplier k for all neighbours. Student Search efficiently finds a better student model for distillation than uniformly applying the same transformations on the teacher model. In particular, the found student architecture can be directly fine-tuned using the student candidates trained by Neighbourhood Distillation.

Sparsification experiments We apply Neighbourhood Distillation with the goal of distilling a sparse student from a fully-trained teacher network. During distillation, the student weights are sparsified by magnitude pruning (Zhu & Gupta, 2017) without changing the input or output shape. This allows us to consider one layer neighbourhoods and to initialize the student weights using the teacher's learned weights. For each neighbourhood, the sparsification rate is ramped-up to the target sparsity rate with a polynomial decay schedule for 20k steps and then maintained for another 20k steps. We show in Table 5 that Neighbourhood Distillation is able to reach the same performace as Knowledge Distillation for low sparsity rates, *without fine-tuning*. For high sparsity rates, the sudden drop in accuracy for models distilled with Neighbourhood Distillation is a symptom of the thresholding effect. When the target sparsity is too high, the student doesn't have enough capacity to approximate its teacher well enough. On instances where it succeeds, Neighbourhood Distillation is **1.7**$\times$ faster than traditional distillation in GPU time.

For Student Search, we consider using the Relative MSE between the student and the teacher output as the quality metric. We observe empirically that this metric correlates well with the accuracy of the reconstructed model. For different ResNetV1-50 models we train neighbourhoods with different sparsity levels and randomly sample 20 models that we reconstruct and evaluate. The Spearman Coefficient (Dodge, 2008) measured on these samples show a correlation of **-0.97** between the Relative MSE and Accuracy. Results of Student Search (Figure 4) are compared with naively applying uniform sparsity rates for the student. While student search is able to improve on Neighbourhood Distillation, it only improves on Knowledge Distillation on lower sparsity rates.

Table 5: Classification accuracies and GPU Time comparison between Neighbourhood Distillation and Knowledge Distillation. Different values of target sparsities are used for the student architecture.

TARGET SPARSITY	ND	KD
0.1	**76.00**	75.63
0.2	**76.03**	75.94
0.3	**75.82**	75.69
0.4	**75.63**	75.45
0.5	**75.26**	75.20
0.6	73.78	**74.60**
0.7	69.52	**69.61**
0.8	49.83	**70.33**
0.9	**0.60**	0.10
GPU TIME (H)	**144**	250

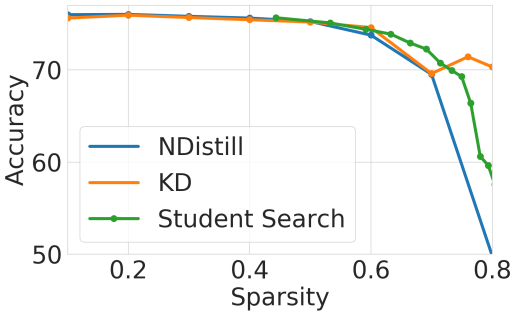


Figure 4: Student Search for Sparsification in ResNetV1-50. Models found with Student Search are obtained by recombining layers with different sparsity rates that were distilled independently with Neighbourhood Distillation. Student Search efficiently finds a better student architectures for distillation than naively applying uniform sparsity rates.

7 DISCUSSION

In this paper, we presented a novel approach to distillation that breaks away from end-to-end training. Distilling small neighbourhoods yields two main advantages compared to traditional Knowledge Distillation: we free ourselves from the computational limitations that come with training very deep models and speed-up distillation by exploiting parallelism. Second, we can reuse the trained

candidates to efficiently explore an exponential number of local architecture changes with Student Search.

We also highlight an interesting phenomenon in deep neural networks that empirically justifies the idea of Neighbourhood Distillation. The thresholding effect explains why replacing all neighbourhoods by distilled students can yield a reconstructed student with reasonable performance. Additionally, the thresholding effect enables efficient diagnosis for cases where the student is too small to properly approximate the teacher and is not suited for Neighbourhood Distillation.

In the appendix, we present preliminary results for further research directions for Neighbourhood Distillation. Understanding the thresholding effect is key to broadening the scope of Neighbourhood Distillation. Studying what impacts the model tolerance threshold could better inform how to limit failure cases and how to trade-off between the speed of training on small neighbourhoods and the accuracy of the distilled student.

Finally, distillation in a context where the original dataset is not accessible is of critical interest for privacy-sensitive applications. Methods for distillation in this setting essentially rely on generating synthetic image datasets and have shown limited success on very deep architectures trained on complex and large-scale datasets. We show in the appendix that Neighbourhood Distillation shows promising results for distillation using only gaussian noise inputs.

REFERENCES

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations, 2019.

Pouya Bashivan, Mark Tensen, and James J DiCarlo. Teacher guided architecture search. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5320–5329, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. *CoRR*, abs/1812.11446, 2018. URL `http://arxiv.org/abs/1812.11446`.

Kartikeya Bhardwaj, Naveen Suda, and Radu Marculescu. Dream distillation: A data-independent model compression framework, 2019.

Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4794–4802, 2019.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. Bam! born-again multi-task networks for natural language understanding. *CoRR*, abs/1907.04829, 2019. URL `http://arxiv.org/abs/1907.04829`.

Elliot J Crowley, Gavin Gray, and Amos J Storkey. Moonshine: Distilling with cheap convolutions. In *Advances in Neural Information Processing Systems*, pp. 2888–2898, 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Yadolah Dodge. Spearman rank correlation coefficient. *The Concise Encyclopedia of Statistics. Springer, New York*, pp. 502–505, 2008.

Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

Sangchul Hahn and Heeyoul Choi. Self-knowledge distillation in natural language processing, 2019.

Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression, 2019.

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.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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. *CoRR*, abs/1704.04861, 2017. URL `http://arxiv.org/abs/1704.04861`.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Furong Huang, Jordan T. Ash, John Langford, and Robert E. Schapire. Learning deep resnet blocks sequentially using boosting theory. *CoRR*, abs/1706.04964, 2017. URL `http://arxiv.org/abs/1706.04964`.

Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. Self-supervised knowledge distillation using singular value decomposition. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$1mb model size. *CoRR*, abs/1602.07360, 2016. URL `http://arxiv.org/abs/1602.07360`.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. *CoRR*, abs/1904.09149, 2019. URL `http://arxiv.org/abs/1904.09149`.

Minsoo Kang, Jonghwan Mun, and Bohyung Han. Towards oracle knowledge distillation with neural architecture search, 2019.

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

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1989–1998, 2020.

Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *CoRR*, abs/1712.00559, 2017. URL `http://arxiv.org/abs/1712.00559`.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Yu Liu, Xuhui Jia, Mingxing Tan, Raviteja Vemulapalli, Yukun Zhu, Bradley Green, and Xiaogang Wang. Search to distill: Pearls are everywhere but not the eyes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7539–7548, 2020.

Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

Sindy Löwe, Peter O'Connor, and Bastiaan S. Veeling. Greedy infomax for biologically plausible self-supervised representation learning. *CoRR*, abs/1905.11786, 2019. URL http://arxiv.org/abs/1905.11786.

Vladimir Macko, Charles Weill, Hanna Mazzawi, and Javier Gonzalvo. Improving neural architecture search image classifiers via ensemble learning. *arXiv preprint arXiv:1903.06236*, 2019.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *arXiv preprint arXiv:1902.03393*, 2019.

Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R Venkatesh Babu, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. *arXiv preprint arXiv:1905.08114*, 2019.

Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. *arXiv preprint arXiv:1901.06656*, 2019.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *CoRR*, abs/1802.03268, 2018. URL http://arxiv.org/abs/1802.03268.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *CoRR*, abs/1807.11626, 2018. URL http://arxiv.org/abs/1807.11626.

Ilya Trofimov, Nikita Klyuchnikov, Mikhail Salnikov, Alexander Filippov, and Evgeny Burnaev. Multi-fidelity neural architecture search with knowledge distillation, 2020.

Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *CoRR*, abs/1605.06431, 2016. URL http://arxiv.org/abs/1605.06431.

Yi Wei, Xinyu Pan, Hongwei Qin, Wanli Ouyang, and Junjie Yan. Quantization mimic: Towards very tiny cnn for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 267–283, 2018.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4133–4141, 2017.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Li Yuan, Francis E. H. Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisit knowledge distillation: a teacher-free framework, 2019.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal?, 2019.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017a. URL `http://arxiv.org/abs/1707.01083`.

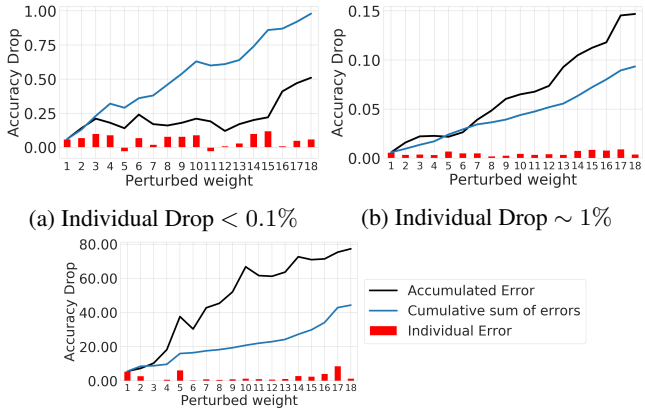
Zhi Zhang, Guanghan Ning, and Zhihai He. Knowledge projection for deep neural networks. *arXiv preprint arXiv:1710.09505*, 2017b.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. URL `http://arxiv.org/abs/1707.07012`.

A Additional Results on the Thresholding Effect

We conducted an additional experiment to show how accuracy can be impacted by direct perturbation of a network's weights. For each convolution layer kernel, we choose to add noisy perturbation with varying standard deviations. We then verify whether the perturbations have an additive effect on the final accuracy by replacing weights from the original network by their perturbed version. Figure 5 shows how errors on the CIFAR-10 test set accumulate when progressively replacing the layers of a pre-trained ResNetV1-20 model, starting from the first layer. The figure compares that to the hypothetical additive regime. We see that when we constrain individual layers to cause small drops in accuracy ($< 0.1\%$), the errors accumulate sub-linearly. On the other hand, when individual layers are perturbed so much that they cause a drop of 2% in accuracy each, the errors accumulate super-linearly.



(a) Individual Drop $< 0.1\%$ (b) Individual Drop $\sim 1\%$

(c) Individual Drop $\sim 2\%$

Figure 5: Error accumulation caused by perturbing network layers. The bar plot shows the individual accuracy drop for a network with perturbed weights for one layer. The blue line plot shows the cumulative sum of errors caused by perturbing a given number of layers from left to right. The black line plot shows the empirical error accumulation evaluated on the test set after perturbing a given number of layers. (a) Individual accuracy drop of each layer is $< 0.1\%$ and the errors accumulate sub-linearly. (b) Individual accuracy drop of each layer is $\sim 1\%$ and errors accumulate linearly. (c) Individual accuracy drop of each layer is high ($\sim 2\%$) and the errors accumulate super-linearly.

We show that a simple regularization technique applied during training can increase a network's resilience to local perturbations. Considering a ResNetV1-50, we add gaussian noise drawn from $\mathcal{N}(0, \sigma)$ at train time after every residual unit. Once these models are trained, we replace each units with perturbed versions again and report the drop in accuracy. Figure 6 shows that for high noise levels σ introduced between residual units at train time, the perturbation threshold is higher, although regularizing too much also degrades the overall performance of the original network.

Table 6: Impact of regularizing the teacher model on Neighbourhood Distillation. Accuracies are obtained before fine-tuning. Using a regularized teacher improves the accuracy of the student before fine-tuning.

STUDENT	TEACHER	NDISTILL ACCURACY	CHANGE
RN50-0.75-3	RN-50	57.63	+0.%
-	RN-50+$\mathcal{N}(\sigma = 0.1)$	63.21	**+5.6%**
RN50-0.5-3	RN-50	25.7	+0.%
-	RN-50+$\mathcal{N}(\sigma = 0.1)$	32.9	**+7.2%**

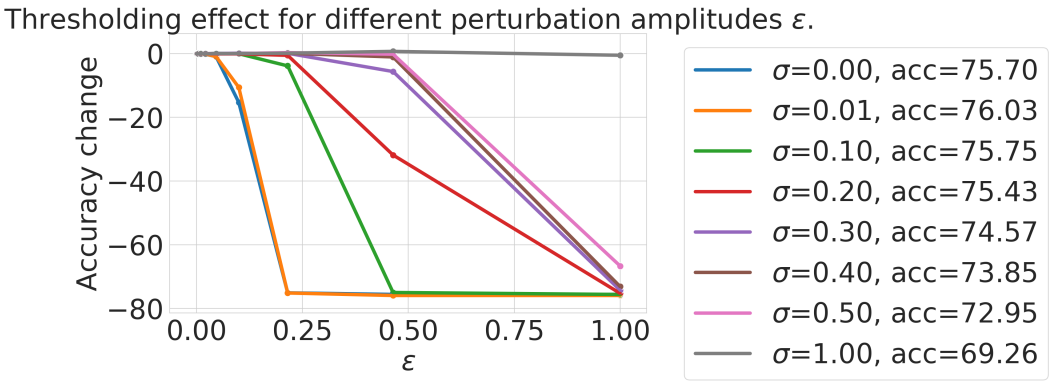


Figure 6: Drop in accuracy after perturbing different ResNetV1-50 models trained with Gaussian Noise $\mathcal{N}(0, \sigma)$ introduced after each unit. ϵ is the amplitude of perturbations introduced at test time. Original accuracies of the models before perturbation are given in the caption. Models trained with high gaussian noise amplitudes σ have a higher resistance threshold to test time perturbations.

More importantly, we observe that increasing the teacher's tolerance threshold directly impacts the accuracy of models trained with Neighbourhood Distillation. We take a ResNetV1-50 trained with regularization $\sigma = 0.1$ as the teacher and consider bottleneck multipliers $k = 0.5$ and $k = 0.75$ for the student neighbourhoods.

B DATA-FREE KNOWLEDGE DISTILLATION

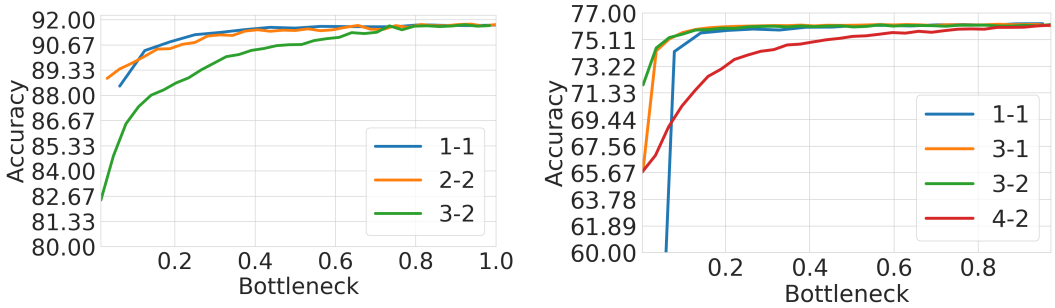
Recent data-free model distillation methods have focused on creating a synthetic image dataset to replace the original one. The synthetic dataset is usually obtained by directly optimizing some input noise with regards to a pre-determined loss (Lopes et al., 2017; Nayak et al., 2019; Bhardwaj et al., 2019; Yin et al., 2020). Generating a substitute dataset using these methods is costly and some fail to generalize to large-scale architectures. Previous work (Haroush et al., 2019) show that it is possible to successfully distill from Gaussian Noise in limited settings (fine-tuning or calibrating quantized models) where the gap between the student and the teacher weights is small. By switching the focus on small and shallow sub-networks, we show that Neighbourhood Distillation can be successfully adapted to use gaussian noise inputs to distill models from scratch.

We rerun our ResNet20 bottleneck multiplier experiments with bottleneck multiplier $k = 1.0, 0.9, 0.75, 0.5$ and compare our method with Zero-Shot Knowledge Distillation Nayak et al. (2019), a data-free method that generates a synthetic training set by optimizing noise inputs with gradient descent. To highlight the benefits of training on shallow sub-networks, we also attempt to use Gaussian Noise inputs to distill the entire student network end-to-end. Table 7 shows that Neighbourhood Distillation outperforms both end-to-end baselines but shows degrading performance as the compression rate decreases. In particular, we are able to perform self-distillation ($k = 1.0$) without losing accuracy, which could not be achieved by the other baselines we consider.

These results show that distillation in a data-free regime can benefit from looking at shallow sub-networks in isolation. As deep neural networks model complex and high-dimensional functions, it

Table 7: Classification accuracies for different data-free distillation methods. Different values of k are used for the student architecture including $k = 1$ (self-distillation). We compare accuracies obtained after different training methods: fully-supervised cross-entropy loss (CE), Gaussian Noise Neighbourhood Distillation (Gaussian-NDistill), Gaussian Noise Neighbourhood Distillation with Gaussian Noise Finetuning (Gaussian-NDistill+FT), Gaussian Noise Knowledge Distillation (GN-KD) and Zero-Shot Knowledge Distillation (Nayak et al., 2019) (ZSKD). Gaussian Noise Neighbourhood Distillation surpasses all other data-free distillation methods we compared to.

k	CE	GAUSSIAN-NDISTILL	GAUSSIAN-NDISTILL+FT	GN-KD	ZSKD
1	91.60	91.51	**91.51**	16.00	21.50
0.9	91.31	88.78	**88.78**	12.97	22.77
0.75	90.89	80.64	**83.74**	13.00	21.54
0.5	90.17	53.40	**54.83**	12.74	22.17



(a) ResNet-20 on CIFAR 10

(b) ResNet-50 on ImageNet

Figure 7: Impact of applying a multiplier k on the distilled accuracy of different Residual Units. x-y refers to unit y of block x of the ResNet. For each unit, accuracy remains unchanged as long as the multiplier remains above a certain threshold. This thresholding effect can be observed for different residual units, but each unit has a different threshold.

is not possible to distill them successfully from unstructured gaussian noise and generally requires generating complex inputs that simulate images. On the other hand, we have demonstrated with Neighbourhood Distillation that it is possible to approximate functions represented by shallow sub-networks using gaussian noise. Notably, when neighbourhoods are only two-layer deep like in the ResNet-20 model, the function is simple enough that it can be distilled almost perfectly using gaussian noise when $k = 1$.

C MOTIVATION FOR STUDENT SEARCH

We observe empirically on the ResNet-20 and ResNet-50 models that there is no reason to think that applying the same multiplier to all layers would yield the best compressed student model. For a number of chosen residual units, we reduce the number of parameters by applying a multiplier $k < 1$ on the number of filters of the first layer and report the best accuracy we can get after training this variant with Neighbourhood Distillation. Figure 7 shows the final accuracy against k for different units of ResNet-20 and ResNet-50. We observe another form of thresholding effect: the accuracy drops sharply below a given value of the multiplier, showing that residual units can be compressed up to a certain point before hurting the network's accuracy. Most importantly, we observe that the threshold is different for each residual unit, which suggests that a better strategy for a student architecture is to choose a different multiplier per unit.

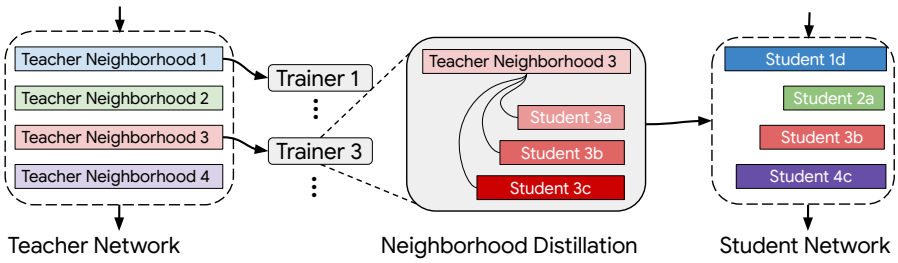


Figure 8: Student Search with Neighbourhood Distillation. A Teacher Network is broken down into several neighbourhoods. Each Neighbourhood can be used to train several student neighbourhoods independently from the rest. Selected student neighbourhoods are then merged back into a single Student Network. This allows us to explore a large search space without having to retrain all models.