ReLUifying Smooth Functions: Low-Cost Knowledge Distillation to Obtain High-Performance ReLU Networks

Soosung Kim, Yeonhong Park, Hyunseung Lee, Sungchan Yi, and Jae W. Lee

Seoul National University, Seoul, South Korea
{soosungkim, ilil96, hs_lee, calofmijuck, jaewlee}@snu.ac.kr

Abstract. Smooth activation functions like Swish, GeLU, and Mish have gained popularity due to their potentially better generalization performance than ReLU. However, there is still high demand for ReLU networks due to their simplicity, yielding higher execution efficiency and broader device coverage. To meet such practical demand, there has been research on producing ReLU networks from pretrained smooth function networks within a limited training time—a process termed ReLUification. Specifically, knowledge distillation (KD) has been key tool for this endeavor. While KD-based ReLUification shows effectiveness to certain extent, the previous approach fails to fully leverage the potential of KD, resulting in suboptimal outcomes. Through in-depth empirical analysis, we uncover that employing a high learning rate synergizes effectively with KD, leading to a substantial improvement in KD-based ReLUification. Additionally, we introduce a novel approach of selectively excluding a portion of the network from ReLUification, significantly enhancing accuracy with negligible additional latency compared to the use of all ReLU networks. Thus, our proposed method produces ReLU networks substantially surpassing the quality of independently trained ReLU networks with an order of magnitude smaller training time.

Keywords: ReLU · Knowledge Distillation · Efficient Computer Vision

1 Introduction

In recent years, there have been increasing trends toward using smooth activation functions, such as Swish [1], GeLU [2] and Mish [3]. This is due to their superior generalization performance compared to the previous frontrunner, ReLU. Table 1 highlights how the incorporation of smooth activation functions improves test accuracy. By allowing a more efficient flow of information through the network, smooth activation functions effectively address some of the potential issues associated with ReLU, such as dead neurons [4]. Consequently, smooth activation functions often lead to improved generalization performance.

Despite its limitations, there are deployment scenarios in which ReLU remains the best, or even the only option, due to hardware constraints or specific

Table 1. Training results with different activation functions (Top-1 Accuracy in %) and the best results are highlighted.

Dataset	Model	ReLU	Swish	Mish	GeLU
CIFAR100	ResNet18	75.25	**75.79**	75.53	75.68
	ResNet34	75.76	75.70	**75.94**	75.53
	InceptionV3	74.51	73.89	**76.20**	74.43
	ShuffleNetV1	69.04	**70.97**	70.33	70.24
	ShuffleNetV2	67.16	68.60	**68.71**	68.24
	MobileNetV1	67.35	**69.39**	68.75	68.74
ImageNet	ResNet18	69.96	70.79	70.83	**70.83**
	MobileNetV3	63.71	67.31	67.04	**67.45**

service requirements. The hardware cost associated with supporting diverse activation functions makes many DNN inference processors support only a limited set of activation functions [5,6], or in extreme cases, solely ReLU [7]. This is especially true for mobile/edge processors which have limited area and power budget. Furthermore, even when the deployment of a smooth activation function becomes feasible through algorithmic approximation or additional hardware expenditure [8], ReLU may still remain the preferred option. This is because ReLU is extremely straightforward to implement and generates a large number of zeros. This characteristic may result in faster inference [9], rendering ReLU the ideal choice when low latency inference is required.

Looking forward, there exist discernible tradeoffs between ReLU and smooth activation functions. While ReLU is straightforward to implement and improves inference speed [10], smooth activation functions demonstrate superior generalization performance. Considering the high cost of examining the tradeoff between the two options, MobileOne [11] falls back to using ReLU rather than exploring the potential advantage of smooth functions. Although MobileOne identifies activation function as a key bottleneck for inference, this approach would leave on the table the potential advantages of better generalization offered by smooth activation functions.

Instead of simply sacrificing one for the other, there has been an attempt to *ReLUify* smooth functions [12] which utilizes knowledge distillation (KD) as a key tool for making ReLU networks out of smooth function networks in a short training time. In the process of extending this approach, initially designed for NLP tasks, to vision tasks, we identify a suboptimal aspect—the failure to fully harness the potential of KD. Through a comprehensive experimental study on this suboptimal aspect, we observe that increasing the learning rate is a key to enhancing the KD-based ReLUification process. Additionally, we introduce a novel approach to selectively exempt a portion of the network from ReLUification, resulting in a substantial increase in accuracy with only negligible sacrifice of inference efficiency. These two proposals improve the KD-based ReLUifica-

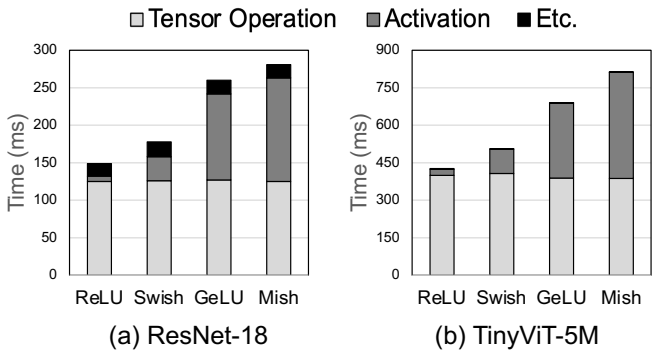

Fig. 1. End-to-end Inference Time of (a) ResNet-18 and (b) TinyVit using different activation functions on Google Pixel 4.

tion process, yielding a high-quality student ReLU model with reduced training epochs, thereby lowering the overall training cost.

Our contributions are summarized as follows:

- We are the first to empirically explore the applicability of KD-based ReLUification to vision tasks and identify the suboptimality of the existing KD-based ReLUification.
- We reveal the criticality of incorporating a higher learning rate, which synergizes effectively with KD-based ReLUification for vision tasks.
- We propose to exclude the rear part of the network from ReLUification, which significantly increases accuracy with only a minimal increase in inference latency.
- We successfully generate ReLU networks that significantly outperform baseline ReLU networks with an order of magnitude less training time compared to training ReLU networks from scratch. Notably, in some cases, our approach yields ReLU networks that match or even surpass the accuracy of networks employing smooth activation functions.

2 Background

2.1 Benefits of ReLU in Practical DNN Deployment

Despite the impressive generalization performance of smooth activation functions like Swish, GeLU, and Mish, ReLU remains prevalent in real-world applications, especially for edge devices. This is because ReLU allows much more efficient inference compared to smooth functions. In this section, we present two main reasons behind the relative advantage of ReLU in terms of inference efficiency.

Table 2. Feature map memory footprint and computation reduction of using ReLU on CIFAR-100.

Model	Memory Footprint Reduction	Computation Reduction
ResNet18	23.94%	50.61%
ShuffleNetV2	17.48%	43.77%
MobileNetV1	15.33%	42.74%

Limited Support for Non-ReLU Activation Functions. While ReLU is simple and easy to implement in hardware, non-ReLU activation functions often involve exponential and logarithmic functions which require complex dedicated hardware units. Exact calculations of these complex operations necessitate deeply pipelined arithmetic units [13]. This cost can be prohibitive, particularly for edge platforms with limited hardware budgets.

There have been some proposals to reduce hardware implementation costs of complex non-linear functions via approximation techniques [8,14]. However, they are not widely adopted yet. Most of the commodity edge processors for DNN inference still support very limited set of activation functions. For example, ARM Ethos-U65 NPU [5] natively supports only four activation functions: ReLU, leaky ReLU, tanh, and sigmoid. In contrast, Google's Edge TPU products [6] extend support to PReLU [15] in addition to these four functions. As an extreme case, some IPs within Xilinx's Vitis AI [7] framework only support ReLU and ReLU6.

When employing activation functions not native to DNN inference processors, the only option is to fall back on the CPU, resulting in a notable performance degradation. To demonstrate the latency increase due to the CPU fallback, we have evaluated the end-to-end inference time of two DNN models, ResNet-18 and TinyViT, incorporating four different activation functions (ReLU, Swish, GeLU, Mish) on Google Pixel 4 which is equipped with an edge TPU as its DNN inference engine. As edge TPU only supports ReLU among those four activation functions, when executing the other three activation functions, it has to call for CPU's assistance. Figure 1 shows the breakdown of the end-to-end inference time. For both models, ReLU incurs minimal latency overhead, making up less than 5% of the total inference time. On the other hand, the utilization of the complex activation functions which necessitates CPU fallback results in significant latency overhead. Notably, in the case of Mish, the end-to-end inference time nearly doubles. The limited parallelism of CPUs compared to dedicated DNN inference engines leads to considerable latency penalty during CPU fallback.

ReLU-induced Sparsity. Unlike other smooth activation functions, ReLU offers unique opportunities for reducing memory footprint and computation cost by zeroing out a significant portion of intermediate feature maps. In fact, the neural processing unit (NPU) embedded in the recent generation of a flagship mobile system-on-chip (SoC) [9,16] aggressively exploits dynamically generated

zero values by compressing intermediate feature maps and skipping ineffectual computations. To estimate the potential performance benefits of ReLU in this case, we profile three networks (ResNet18, ShuffleNetV2, MobileNetV1) on CIFAR100 to gauge potential savings for both memory footprint and computation. We assume the quadtree-based compression scheme used in Samsung's NPU to measure the memory footprint reduction and counted the number of ineffectual computations that can be reduced. The results in Table 2 indicate that using ReLU reduces feature map memory footprint by 18.92% on average (23.94% at maximum). In addition, ReLU enables us to skip 45.71% of multiply-accumulate (MAC) computation on average (50.61% at maximum), potentially resulting in corresponding reductions in latency and power consumption.

2.2 ReLUification: Deriving ReLU Network out of Non-ReLU Network

Needs for ReLUification. As elaborated in Section 2.1, incorporating ReLU provides significant advantages in terms of inference efficiency, especially to the platforms that lack support for smooth activation functions. To harness the benefit of ReLU while accommodating diverse deployment scenarios with varying hardware configurations and service requirements, practical approach involves training and managing multiple sets of parameters, each associated with a candidate activation function. However, naively training N sets of parameters may proportionally increase the training cost. Thus, an effective ReLUification method, which converts smooth activation functions into ReLU models, offers a valuable solution for practitioners.

MA-BERT: KD-based ReLUification. Despite its practical significance, to the best of our knowledge, there has been limited discussion on ReLUification of vision models so far. Meanwhile, in the NLP domain, MA-BERT [12] has successfully replaced the GeLU activation function in the pre-trained BERT models with ReLU within a limited number of training steps of the fine-tuning phase. MA-BERT leverages knowledge distillation (KD), where the pre-trained BERT model with GeLU acts as a teacher, and the BERT model with GeLU replaced by ReLU becomes a student. The student inherits all the trainable parameters from the teacher model at initialization, and a small learning rate is used, following the typical fine-tuning process.

3 KD-based ReLUification for Vision Models

In this section, we apply the MA-BERT-style KD-based ReLUification method [12] to vision models. We aim to answer the following two questions: (i) Does the KD-based ReLUification method also work well on vision models? (ii) To what extent does the incorporation of KD enhance the ReLUification process?

Table 3. MA-BERT-style ReLUification results with and without knowledge distillation (Top-1 Accuracy in %). S, M and G indicate that teacher model is using Swish, Mish and GeLU. Better results are highlighted between ReLUification methods.

Dataset	Model	Baseline		ReLUification	
		ReLU	Teacher	w/ KD	w/o KD
CIFAR100	ResNet18	75.25	75.79 [S]	**75.25**	75.00
	ResNet34	75.76	75.94 [M]	75.61	**75.62**
	InceptionV3	74.51	76.20 [M]	73.43	**74.82**
	ShuffleNetV1	69.04	70.97 [S]	68.78	**69.40**
	ShuffleNetV2	67.16	68.71 [M]	66.4	**67.02**
	MobileNetV1	67.35	69.39 [S]	64.87	**65.35**
ImageNet	ResNet18	69.96	70.83 [G]	**69.94**	68.99
	MobileNetV3	63.71	67.45 [G]	63.89	**64.31**

3.1 Methodology

We evaluate the effectiveness of the KD-based ReLUification methods on vision tasks with six models (ResNet18, ResNet34, InceptionV3, ShuffleNetV1, ShuffleNetV2 and MobileNetV1) on CIFAR100 and two models (ResNet18 and MobileNetV3) on ImageNet. We train these networks from scratch using three smooth activation functions (Swish, GeLU and Mish) to make the baseline teacher models. In addition, we also train ReLU networks from scratch for the comparison baseline. The detailed training hyperparameters can be found in Table 15 in the Appendix D.

For ReLUification, we follow the approach of MA-BERT [12]. We initialize student ReLU models with teacher models' parameters and set the initial learning rate to be 0.01, which is an order of magnitude smaller than the value used for the training model from the scratch. The weight for distillation loss is set to be 0.9. The student models are distilled for 15 epochs on CIFAR100 and 6 epochs on ImageNet. The learning rate schedule is same as in Table 15, scaled proportionally to the reduced training epochs. As a vision-specific refinement, we have modified which features from the teacher model to extract for distillation. While student models in MA-BERT learn from the features of every single layer of the teachers, we utilize features only from layers located before downsampling operations–a strategy commonly employed in knowledge distillation for vision models. Extracting too many features from the teacher in vision models is known to potentially have a negative impact on the quality of the student model [17]. To assess the effectiveness of KD in the process of ReLUification, we also perform ReLUification without KD.

3.2 Experimental Results

Table 3 shows the MA-BERT style ReLUification results. Among all three teacher-student pairs for each combination of dataset and model, in this table, we only

report results for a single pair whose teacher has the highest baseline accuracy for conciseness. The results for the other pairs are available in Appendix A. In general, the KD-based ReLUification works fairly well. Except for a few cases like ShuffleNetV2 and MobileNetV1 on CIFAR100, the ReLU models produced from the KD-based ReLUification match the accuracy of the baseline ReLU models, which are trained independently from the scratch. This is impressive results, given the very small number of distillation epochs. However, interestingly, the ReLUification without KD also shows competitive results. For 5 out of 8 cases, the ReLUification without KD is even better than the KD-based ReLUification.

From this observation, we can conclude that KD does not improve the ReLUification process under the current methodology. Instead, the utilization of the teacher's parameters for the student model initialization contributes more to the effectiveness of ReLUification. Popular smooth activation functions, including those employed in our evaluation, are usually similar to ReLU in shape. Thus, reusing the teacher's parameters may position the student ReLU model near a relatively good minimum-loss point. Given that teacher models with smooth activation functions demonstrate significantly higher accuracy than their ReLU counterparts, these results leave much to be desired. There may be a better way of KD-based ReLUification if we can fully utilize the merits of learning from the good teachers.

4 Improving KD-based ReLUification for Vision Models

Our goal is to improve the KD-based ReLUification, thus achieving two seemingly contradictory goals: short training time and high accuracy. Specifically, we aim to produce ReLU models that not only match but surpass the accuracy of independently trained ReLU models, within a minimal number of distillation epochs. For this purpose, we propose two improvements. Section 4.1 presents the first one, which is to increase the learning rate, a key technique to highlight and make full use of the merits of KD. Section 4.2 introduces an option to further improve the quality of the ReLU model by excluding sensitive layers from ReLUification, with only marginal loss of inference efficiency. Section 4.3 gives a summary of the accuracy and training cost benefits of our ReLUification methods.

4.1 Optimizing Learning Rate

Limitations of Existing KD-based ReLUification. We have analyzed the issues of the KD-based ReLUification. Figure 2 shows how accuracy changes as we increase the number of training epochs by 2× and 4× for KD-based ReLUification. On CIFAR100, the accuracy tends to increase with the number of epochs, eventually surpassing the baseline ReLU models. However, the pace is too slow. Many models underperform their baseline even with a double increase in epochs, and the meaningful accuracy improvement (>1.0%) requires fourfold increase in epochs.

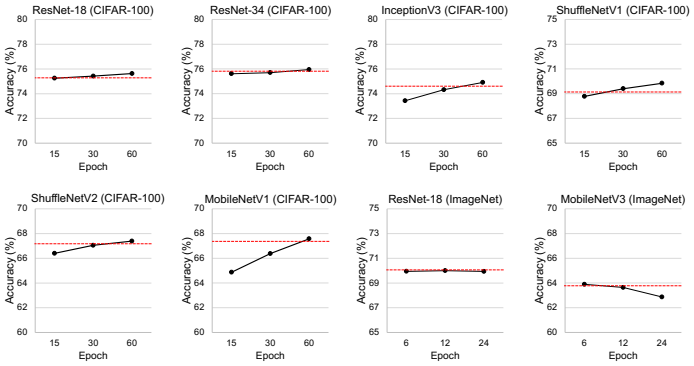

Fig. 2. Results for MA-BERT style KD-based ReLUification for longer epochs using low learning rate. The accuracy of ReLU baseline is drawn with red dotted lines.

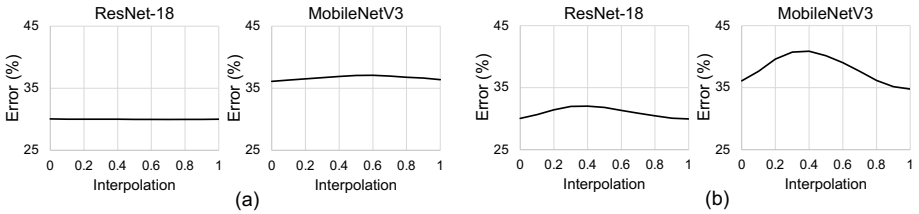

Fig. 3. Test error curve of the linear interpolation between distilled ReLU students trained with (a) low LR for 6 and 12 epochs (b) low and high LR for 6 epochs. ReLU student trained for 6 epochs with low learning rate is located in $x = 0$ and the other is located in $x = 1$.

On large-scale dataset, ImageNet, the problem becomes worse. While the pace of optimization is the problem on CIFAR100, the *trajectory* of optimization itself seems to be the problem on ImageNet. On ImageNet, even with an increased number of training epochs, the accuracy remains stagnant and does not improve beyond the level of the baseline ReLU models. Figure 3(a) illustrates the error of linear interpolation between two ReLUified networks—one with 6 epochs of ReLUification and the other with 12 epochs of ReLUification—for ResNet-18 and MobileNetV3 on ImageNet. For both models, the minima found by the two ReLUified networks are connected by a path of non-increasing error, indicating that they are located in the same minima. This observation leads to the conclusion that the reliance on teacher models' parameters guides student models to converge to the same specific local minima [18]. Given that the accuracy does not improve with an increase in training epochs, it suggests that while the local minima are satisfactory to some extent, they are not optimal enough to surpass the quality of the baseline ReLU models. This underscores the necessity of navigating away from these local minima.

Table 4. ReLUification results using high learning rate (Top-1 Accuracy in %). For each ReLUification method, the improvements of using high learning rate compared to low learning rate are also reported.

Model	Baseline		ReLUification w/ KD		ReLUification w/o KD	
	ReLU	Teacher	LR=0.1	LR=0.01	LR=0.1	LR=0.01
CIFAR100						
ResNet18	75.25	75.79	75.59 (▲0.34)	75.25	73.91 (▼1.09)	75.00
ResNet34	75.76	75.94	75.79 (▲0.18)	75.61	74.68 (▼0.94)	75.62
InceptionV3	74.51	76.20	75.43 (▲2.10)	73.43	76.29 (▲1.47)	74.82
ShuffleNetV1	69.04	70.97	70.49 (▲1.71)	68.78	69.43 (▲0.03)	69.40
ShuffleNetV2	67.16	68.71	67.58 (▲1.18)	66.4	67.78 (▲0.76)	67.02
MobileNetV1	67.35	69.39	67.04 (▲2.17)	64.87	44.15 (▼21.2)	65.35
ImageNet						
ResNet18	69.96	70.83	70.05 (▲0.11)	69.94	65.54 (▼3.45)	68.99
MobileNetV3	63.71	67.45	65.23 (▲1.34)	63.89	63.52 (▼0.79)	64.31

Solution: Increasing Learning Rate. To address issues above, we advocate for the adoption of a higher learning rate. In general, the higher the learning rate, the faster the training progresses. Moreover, a higher learning rate helps escaping local minima by enabling active exploration of the loss surface. However, this property is a double-edged sword. In the context of ReLUification, active exploration may elevate the risk of diverging from the initial point and forgetting the information inherited from the teachers by reusing the parameters. This can pose a significant problem as this information plays a crucial role in enhancing both the efficiency and effectiveness of ReLUification, as discussed in Section 3.2. However, the incorporation of Knowledge Distillation (KD) is expected to alleviate such potential risks associated with the higher learning rate. By consistently guiding the students to learn from the teachers, it may help minimize the risk of forgetting the information inherited from the teachers. In other words, the KD-based ReLUification and a higher learning rate are mutually complementary.

Table 4 shows how the use of higher learning rate affects ReLUification with and without KD. We have modified the learning rate from 0.01 to 0.1. In tandem with the learning rate increase, we have reduced the weight decay parameter from 10^{-4} to 10^{-6}. The reduction in the weight decay parameter is a commonly employed technique in conjunction with the adoption of a higher learning rate, as it stabilizes the convergence process [19]. The remaining settings are the same with the experiments conducted in Section 3.2.

As anticipated, in ReLUification with KD, an increase in the learning rate consistently results in a remarkable improvement in accuracy. The error of linear interpolation illustrated in Figure 3(b) shows that ReLUification with high learning rate resulted in different minima with the model using low learning rate.

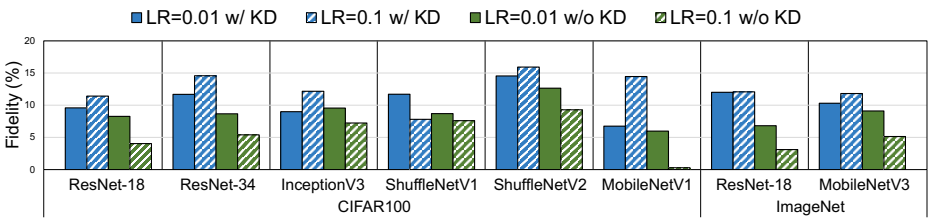

Fig. 4. Fidelity measurement results of ReLUification with and without KD with two different learning rates (0.01 and 0.1). High fidelity implies higher resemblance between the teacher and student models.

This implies that adopting high learning rate offers more chance of exploration, which in consequence, can result in better local minima. Except for the ResNet family, the accuracy sees an increase of more than 1%. Note that for the ResNet family, both the ReLU baseline and the smooth function baseline do not exhibit substantial accuracy gaps, suggesting very limited room for improvement. However, the use of a high learning rate for ReLUification without KD occasionally leads to a significant drop in accuracy, as observed in MobileNetV1 on CIFAR100 and ResNet18 on ImageNet.

The negative impact of a high learning rate in ReLUification without KD is attributed to the forgetting of the information inherited from the teachers. Figure 4 illustrates the changes in fidelity for ReLUification with and without KD when the learning rate is increased. Fidelity, in the context of KD, is a metric used to measure the match rate between the output labels of two models [20]. It provides a quantitative measure of how closely the student model resembles the teacher model. Without KD, a higher learning rate leads to lower fidelity, while with KD, it rather increases fidelity. This indicates that the use of KD mitigates the side effects of a high learning rate while retaining its benefits.

4.2 Selective Exclusion

Rather than ReLUifying the entire network, we investigate a trade-off between accuracy and inference efficiency by selectively excluding a part of the model from ReLUification. The more we exclude, the higher the accuracy but at the expense of decreased inference efficiency. Ideally, the optimal choice would be a part whose exclusion from ReLUification significantly improves accuracy while incurring minimal increases in inference latency. Fortunately, excluding only the last few layers of the network satisfies both traits.

To examine how each part of the network affects the accuracy and the inference efficiency, we grouped layers of the network into three parts: front, middle, and rear. In modern CNN networks, there are multiple downsample layers that reduce the feature map size. We defined a group of layers separated by downsample layers as a block, and heuristically designated the first few blocks as the front, the last block as the rear, and the intervening blocks as the middle, based on

Table 5. ReLUification results using selective exclusion (Top-1 Accuracy in %). All models are separated into three parts and during ReLUification, one of them remained as smooth function layer. Best ReLUification results are highlighted for each model.

Dataset	Model	Excluded Part			All ReLU
		Front	Middle	Rear	
CIFAR100	ResNet18	75.7	75.53	**75.96**	75.79
	ResNet34	**75.92**	75.62	75.73	75.79
	InceptionV3	75.24	**76.18**	75.34	75.43
	ShuffleNetV1	70.94	**70.95**	70.80	70.49
	ShuffleNetV2	67.78	68.06	**68.31**	67.58
	MobileNetV1	69.26	69.26	**69.39**	67.04
ImageNet	ResNet18	70.12	**70.54**	70.51	70.05
	MobileNetV3	65.70	66.05	**66.22**	65.23

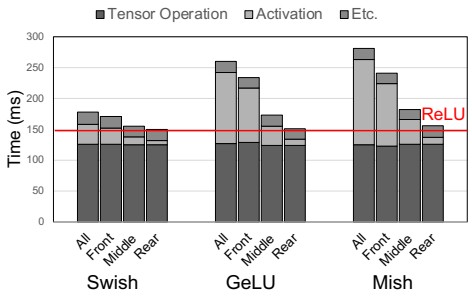

Fig. 5. Latency measurement of ReLUification with selective exclusion using ResNet18 on Pixel 4. Each label stands for the Non-ReLU part of the models and the red line displays the latency of the all-ReLU model.

the network architecture. The visualization of how we defined the front, middle, and rear for each network is provided in the Appendix B.

Table 5 is the accuracy results after ReLUification with selective exclusion. While different models exhibit different trends, as a general observation, preserving smooth activation functions in the rear part tends to yield the best results. In four out of the eight cases, this option produces the highest accuracy. Since the rear part determines the final feature which is the input layer to the linear classifier, not replacing their activation function with ReLU results in better performance. Figure 5 shows how the exclusion of each part from ReLUification increases the inference latency on ResNet-18 with ImageNet. Again, the rear part is the optimal choice for the exclusion. Keeping smooth functions in the rear part barely increases the inference latency compared to the ReLU-only model. This is expected, considering that the feature map size reduces each time it passes through downsample layers. The feature map size in the rear part is typically an order of magnitude smaller than that of the front and the middle, implying

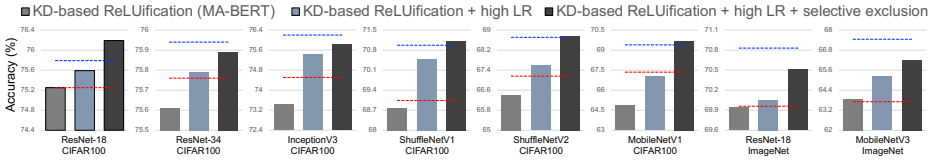

Fig. 6. Summarized improvements of ReLUification incorporating proposed methods (Top-1 Accuracy in %). Red line stands for their ReLU baseline accuracies and blue line for their teacher models' accuracies.

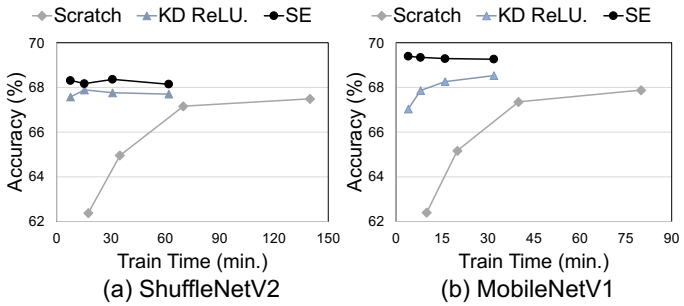

Fig. 7. Accuracy-training time comparison of 1) training from scratch, 2) KD-based ReLUification with high LR (KD ReLU) and 3) KD ReLU with Selective Exclusion (SE). Accuracies are displayed with the total training time (min.).

that the overhead of performing activation functions in this part is minimal. To conclude, we recommend exempting the rear part from ReLUification.

4.3 Putting It All Together

Figure 6 shows how the incorporation of our two key proposals improve the KD-based ReLUification process. By increasing learning rate and selectively keeping smooth activation functions in the latter layers, the accuracy of the resulting student models consistently surpasses that of the baseline ReLU models across all scenarios. In some cases, our approach achieves parity with or even outperforms the accuracy of the teacher models.

What is even more noteworthy is the speed at which we can produce such high-quality ReLU models. In Figure 7, we compare the training time against accuracy for our enhanced KD-based ReLUification methods with the approach of training ReLU models from scratch by executing multiple training runs with different epochs for each method. We present two variants of our ReLUification methods: one includes the exclusion of the rear part, while the other does not. All results are measured on a single RTX 3090 GPU. Here we report results for MobileNetV1 and ShuffleNetV2 on CIFAR100. Corresponding graphs for other models and datasets can be found in the Appendix C.

Table 6. ReLUification results of TinyViT on ImageNet

Model	Baseline	ReLUification	
	GeLU	w/ KD	w/o KD
TinyViT-5M	78.98	78.30	76.29
TinyViT-11M	81.33	80.85	78.92
TinyViT-21M	82.97	82.63	81.06

Table 7. ReLUification results with different feature selection for distillation (Top-1 Accuracy in %).

Features Extracted	ResNet18	MobileNetV3
All	70.09	65.40
Final	70.16	65.25
Downsample	70.05	65.23
Downsample w/o Final	69.73	64.94

The use of KD results in a slightly longer per-epoch training time compared with standard training. However, our methods enable the achievement of high accuracy with a remarkably low number of training epochs, leading to a significantly improved trend in training time to accuracy. In fact, our ReLUification without the exclusion of the rear part, resulting in all-ReLU networks, requires an order of magnitude less training time to reach the same level of accuracy compared to standard training for both MobileNetV2 and ShuffleNetV2. By accepting a marginal compromise in inference efficiency, specifically through the exclusion of the rear part from ReLUification, we observe a further enhancement in the training time-to-accuracy trend.

5 Discussion

Application to Transformer-Based Model. We applied our ReLUification method to TinyViT models [21], which originally use GeLU activation, starting with a standard learning rate of 0.01 for transformer-based models using SGD [22]. Table 6 shows that after 4 epochs of distillation, TinyViT-ReLU reported only 0.34-0.68% accuracy loss compared with the GeLU network. We believe that further optimizations specific for transformer-based models can improve the results. However, still, given the minimal training expense, the slight decrement in accuracy could be considered an acceptable trade-off.

Analysis on Feature Extraction In the process of knowledge distillation, intermediate features can be extracted from various positions [12,17,23]. To assess the efficacy of each feature, we examined various feature selection configurations: all layers, final pre-classification layer, before all downsampling layers with or

Table 8. ReLUification results on ImageNet with advanced distillation methods and optimizers (Top-1 Accuracy in %). RMS, AG and AD stands for RMSProp, AdaGrad and AdaDelta, respectively.

Model	Ours	Distillation Methods				Optimizers				
		RKD	AT	SKD	DKD	Adam	AdamW	RMS	AG	AD
ResNet18	70.05	61.89	69.04	69.90	69.22	69.88	69.91	69.68	69.87	69.84
MobileNetV3	65.23	61.65	64.52	64.74	65.36	65.00	65.07	64.85	64.74	64.36

without the final layer. Table 7 shows that including the final layer's feature, and thereby producing a similar feature map after final layer, is most important.

Effect of Advanced Distillation Methods and Optimizers We examine the effect of using advanced distillation methods: parametric (RKD) [24], non-parametric (AT) [25], distillation-only (SKD) [26] and non-feature distillation (DKD) [27] approaches. Table 8 suggests that the advanced distillation methods do not necessarily lead to better ReLUification results than standard distillation that we advocate. Previous research on distillation has mainly focused on training models from scratch, unlike the limited training time in ReLUification.

Table 8 also presents results of using alternative optimizers in place of SGD. We test multiple learning rates and report the best results for each optimizer. Using a high learning rate—equivalent to the initial learning rate used when training from scratch—yields the best results for all optimizers. Detailed information on the hyperparameter selection process is provided in Appendix D. None of the advanced optimizers demonstrate improvements over SGD.

6 Conclusion

Despite the generalization strengths of smooth activation functions, ReLU still remains essential for deployment efficiency. Knowledge distillation (KD) has emerged as a method to swiftly convert smooth models into ReLU models, but its use in vision tasks remains underexplored. In this study, we analyze KD-based ReLUification for vision tasks and propose a recipe that includes using a high learning rate. Our approach produces high-quality ReLU models with significantly reduced training time.

Acknowledgments. This work was supported by a research grant from Samsung Advanced Institute of Technology (SAIT), Institute of Information & Communications Technology Planning & Evaluation (IITP) and the National Research Foundation of Korea (NRF) grants funded by the Korea Government (MSIT) (2021-0-00105 and RS-2024-00405857). Jae W. Lee is the corresponding author.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.
2. Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*, 2016.
3. Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 2019.
4. Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. *in ICML Workshop on Deep Learning for Audo, Speech, and Language Processing (WDLASL)*, 2013.
5. ARM Ethos-U65 NPU. `https://developer.arm.com/documentation/102023/0000`, 2020.
6. Coral dev board. `https://coral.ai/docs/dev-board/datasheet/`, 2020.
7. Vitis ai. `https://docs.xilinx.com/r/en-US/ug1414-vitis-ai`, 2020.
8. Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. I-BERT: integer-only BERT quantization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5506–5518. PMLR, 2021.
9. Jun-Woo Jang, Sehwan Lee, Dongyoung Kim, Hyunsun Park, Ali Shafiee Ardestani, Yeongjae Choi, Channoh Kim, Yoojin Kim, Hyeongseok Yu, Hamzah Abdel-Aziz, Jun-Seok Park, Heonsoo Lee, Dongwoo Lee, Myeong Woo Kim, Hanwoong Jung, Heewoo Nam, Dongguen Lim, Seungwon Lee, Joon-Ho Song, Suknam Kwon, Joseph Hassoun, SukHwan Lim, and Changkyu Choi. Sparsity-aware and re-configurable NPU architecture for samsung flagship mobile soc. In *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*, pages 15–28. IEEE, 2021.
10. Xiaohu Tang, Shihao Han, Li Lyna Zhang, Ting Cao, and Yunxin Liu. To bridge neural network design and real-world performance: A behaviour study for neural networks. In Alex Smola, Alex Dimakis, and Ion Stoica, editors, *Proceedings of Machine Learning and Systems 2021, MLSys 2021, virtual, April 5-9, 2021*. mlsys.org, 2021.
11. Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Mobileone: An improved one millisecond mobile backbone. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 7907–7917. IEEE, 2023.
12. Neo Wei Ming, Zhehui Wang, Cheng Liu, Rick Siow Mong Goh, and Tao Luo. MA-BERT: towards matrix arithmetic-only BERT inference by eliminating complex non-linear functions. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
13. Jing Chen and Xue Liu. A high-performance deeply pipelined architecture for elementary transcendental function evaluation. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 209–216, 2017.
14. Joonsang Yu, Junki Park, Seongmin Park, Minsoo Kim, Sihwa Lee, Dong Hyun Lee, and Jungwook Choi. Nn-lut: Neural approximation of non-linear operations for efficient transformer inference. In Rob Oshana, editor, *Proceedings of the 59th*

ACM/IEEE Design Automation Conference, DAC '22, page 577–582, New York, NY, USA, 2022. Association for Computing Machinery.
15. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015.
16. Jun-Seok Park, Jun-Woo Jang, Heonsoo Lee, Dongwoo Lee, Sehwan Lee, Hanwoong Jung, Seungwon Lee, Suknam Kwon, Kyung-Ah Jeong, Joon-Ho Song, SukHwan Lim, and Inyup Kang. 9.5 A 6k-mac feature-map-sparsity-aware neural processing unit in 5nm flagship mobile soc. In *IEEE International Solid-State Circuits Conference, ISSCC 2021, San Francisco, CA, USA, February 13-22, 2021*, pages 152–154. IEEE, 2021.
17. Linfeng Zhang, Yukang Shi, Zuoqiang Shi, Kaisheng Ma, and Chenglong Bao. Task-oriented feature distillation. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
18. Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR, 2020.
19. Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2017.
20. Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A. Alemi, and Andrew Gordon Wilson. Does knowledge distillation really work? In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6906–6919, 2021.
21. Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXI*, volume 13681 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 2022.
22. Cream. `https://github.com/microsoft/Cream`.
23. Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1921–1930. IEEE, 2019.
24. Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5008–5017. Computer Vision Foundation / IEEE, 2021.
25. Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

26. Defang Chen, Jian-Ping Mei, Hailin Zhang, Can Wang, Yan Feng, and Chun Chen. Knowledge distillation with the reused teacher classifier. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11923–11932. IEEE, 2022.
27. Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11943–11952. IEEE, 2022.