ZEROLIERS: DIMINISHING LARGE OUTLIERS IN RELU-LIKE ACTIVATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

As the number of learnable parameters is getting bigger and bigger, overfitting is still one of the main challenges in training DNNs. Even though DNNs with billions or even a few hundred billions of parameters are proposed and used, it is still hard to determine the appropriate training set size that prevents overfitting. In this work, we propose a new activation function, called *ZeroLiers*, to prevent overfitting. It eliminates the need to use Dropout and leads to better generalization when training DNNs with fully connected layers. ZeroLiers can be easily implemented by replacing large outliers in ReLU-like activations with zeros. We perform an empirical evaluation of ZeroLiers' regularization effect against Dropout. Interestingly, the validation loss decreases much faster using ZeroLiers than Dropout, and the generalization performance improves. Moreover, we train several recent DNNs with fully connected layers and investigate the effect of ZeroLiers. Specifically, we find that ZeroLiers accelerates the convergence speed of both their training and validation losses.

1 INTRODUCTION

One of the most critical aspects of Deep Neural Networks (DNNs) is how well the network generalizes to unseen data. In supervised learning, if a DNN model has been trained too well on training data, it will not generalize. That is, *overfitting* occurs when the model makes inaccurate predictions for new data while doing well with the training data. It is easy to prevent overfitting if there is enough amount of training data. However, recent DNNs with millions or even billions of parameters (Cheng et al., 2018) are prone to overfitting because preparing sufficient training data for them is a difficult task.

To prevent overfitting, many regularizers, such as Dropout (Hinton et al., 2012) and DropConnect (Wan et al., 2013), have been proposed without requiring additional training data. Moreover, while originally used to prevent *units* (a.k.a. *nodes* or *neurons*) from co-adapting too much, a wide range of stochastic techniques inspired by Dropout has been proposed (Labach et al., 2019).

Currently, the most widely-used activation function in neural networks is the Rectified Linear Unit (ReLU) (Nair & Hinton, 2010; Glorot et al., 2011). Even though neural networks conventionally employ a sigmoidal non-linearity function, sigmoidal neural networks may suffer from the *vanishing gradient problem* (Bengio et al., 1994). First introduced by Nair & Hinton (2010), ReLU enables faster and better convergence than a sigmoid function by alleviating the vanishing gradient problem (Hochreiter, 1998; Hochreiter et al., 2001).

GELU is a ReLU-like activation function and is also being widely used in various large-scale DNNs, such as BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019). It combines the properties of Dropout, Zoneout (Krueger et al., 2016), and ReLU to boost regularization performance. Zoneout stochastically forces some hidden units to maintain their previous values using random noise to improve generalization. However, BERT and GPT-2 still use GELU together with Dropout. Although non-linearities and Dropout together determine a unit's output, they remain distinct and work independently. This implies that GELU does not have enough regularization effect in general.

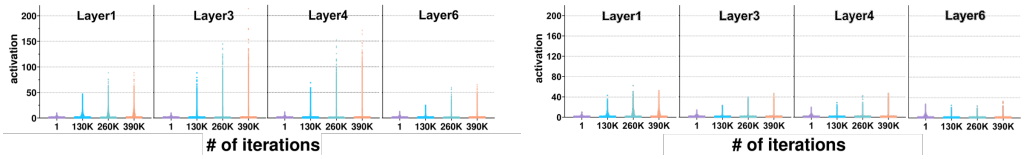
In this paper, we propose a new activation function, called *ZeroLiers* (**Zero**ing out**Liers**). By preventing overfitting when training DNNs with fully connected layers, ZeroLiers eliminates the need to use Dropout and leads to better generalization. It replaces large *outliers* with zeros or diminishes their effect, especially in ReLU-like activations. An outlier is an observation that lies outside the

overall pattern of a distribution (Moore & McCabe, 2002). The ReLU-like activations can be easily affected by the presence of outliers because of their semi-linear property.

Despite the widespread adoption of Dropout, it is still not clear that dropping out what kind of activations improves the generalization capability of DNNs. By proposing ZeroLiers, this paper essentially addresses the following question:

- Can we exploit activation functions to prevent DNNs from overfitting instead of using Dropout?

To the best of our knowledge, ZeroLiers is the first work that relates overfitting to outlier values in activations to avoid it.



(a) Without Dropout (Baseline). (b) With Dropout (the dropout probability q = 0.3).

Figure 1: Training a seven-layer MLP on CIFAR-10 dataset with ReLU: scatter plots of ReLU activations at some sampled layers over iterations.

Figure 1 shows the scatter plots of activations at four sampled layers over iterations for a seven-layer MLP on CIFAR-10. The y-axis represents the value of activations. Figure 1(a) shows the values without Dropout and Figure 1(b) with Dropout. We obtain the activations of units before applying Dropout to the units.

We observe that the values of activations are concentrated around zero. As iteration goes over and over, the number of values that deviate from the group around zero is increasing and their activation values become larger. On the other hand, after applying Dropout, the number of values that deviate from the group around zero is much smaller, and their activation values become much smaller too. Based on the observation, we can treat Dropout as a technique that randomly removes not just non-outliers but also outliers. Unlike Dropout, ZeroLiers can be thought of as a technique that identifies adversely behaving activations as outliers and drops them out. It provides DNN models with better generalization capability than Dropout in general.

The contributions of this paper are summarized as follows:

- We show that when using ZeroLiers in training a multilayer perceptron (MLP) and two types of Autoencoders that originally use a variant of ReLU, the validation loss often decreases much faster, and a lower validation loss than that of Dropout can be achieved.
- One of the limitations of widely used ReLU-like activation functions, such as GELU, SiLU, and Mish, is that they require an additional Dropout step even though they were originally invented to boost regularization performance. We show that ZeroLiers overcomes this limitation.
- The experimental result of BERT using ZeroLiers indicates that they can be trained much faster than using ReLU-like activation functions. Moreover, both the training and validation losses significantly improve.

2 Preliminaries and Related Work

This section briefly describes the standard Dropout technique and popular ReLU-like activation functions used in DNNs. We also summarize their related work.

2.1 Dropout

Dropout was first introduced in the seminal work by Hinton et al. (2012). It is a technique that prevents overfitting by randomly dropping out units in a neural network. Consider a neural network with L hidden layers. Let $l \in \{1, \cdots, L\}$ be the indices of the hidden layers. Let $\boldsymbol{z}^l$ be the input

vector to layer l, and $\boldsymbol{y}^l$ the output vector from layer l. In addition, let $\boldsymbol{w}^l$ and $\boldsymbol{b}^l$ be the weights and biases at layer l, respectively. Then, the feed-forward operation with Dropout is described as:

$$
\begin{aligned}
r_j^l &\sim \text{Bernoulli}(p), \\
z_i^{l+1} &= w_i^{l+1} \left(\boldsymbol{r}^l \odot \boldsymbol{y}^l \right) + b_i^{l+1}, \\
y_i^{l+1} &= f\left(z_i^{l+1} \right),
\end{aligned}
$$

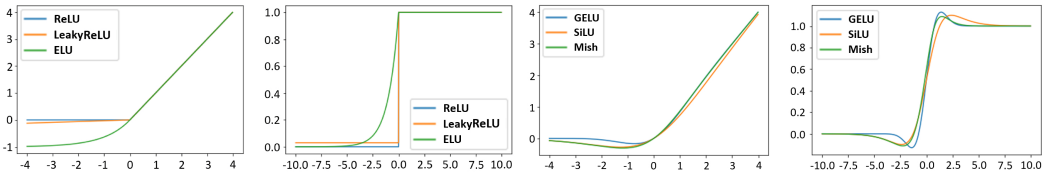
where $\boldsymbol{r}^l$ is a vector of independent Bernoulli random variables each of which has probability p of being 1 (that is, the probability q of being 0 is equal to $1 - p$), $\odot$ denotes an element-wise vector (Hadamard) product, and f is any activation function. Once the weights are learned, the weights are rescaled by multiplying p during test time.

By dropping a unit out in the training phase, it prevents units from co-adapting too much. Co-adaptation implies that some units are highly dependent on other units. An adversely behaving activation from a unit may affect significantly the dependent units resulting in altering the model performance. This might be the reason for overfitting. With Dropout, the model is forced to have more essential units that are necessary to learn target features while not counting on other units. Consequently, the resulting model can be more robust to unseen data.

Work related to Dropout. Dropout can also be interpreted as a way of regularizing a neural network by adding noise to its hidden units (Srivastava et al., 2014). Wager et al. (2013) treat Dropout as an adaptive l_2 regularization and establish a connection between Dropout and AdaGrad (Duchi et al., 2011). Wang & Manning (2013) show how to do fast Dropout training by sampling from or integrating a Gaussian approximation. Inspired by Dropout, a number of variants have been proposed, such as DropConnect (Wan et al., 2013), Standout (Ba & Frey, 2013), Fast Dropout (Wang & Manning, 2013), Variational Dropout (Kingma et al., 2015), DropBlock (Ghiasi et al., 2018), Guided Dropout (Keshari et al., 2019), and Augmented Dropout (Gao et al., 2019). Wen et al. (2018) propose SmoothOut to smooth out sharp minima in DNNs. SmoothOut perturbs several copies of the DNN by injecting noise to SGD and averages these copies. They also propose AdaSmoothOut that is an adaptive variant of SmoothOut.

On the other hand, some approaches attempt to study the generalization performance of Dropout theoretically. Gao & Zhou (2016) show that Dropout can help to reduce the Rademacher complexity of DNNs. Mou et al. (2018) study the regularization effect of Dropout under the distribution-free setting. Labach et al. (2019) summarize the history of Dropout methods, their various applications, and recent areas of Dropout research.

Handling overfitting using other methods. There have also been several approaches to handle overfitting using methods other than Dropout. Salman & Liu (2019) show that overfitting is due to continuous gradient update and scale sensitiveness of the cross-entropy loss. Ge et al. (2020) propose a new measure for overfitting. Rice et al. (2020) find that overfitting is a dominant phenomenon in the adversarially robust training of deep networks.



(a) (b) (c) (d)

Figure 2: Graphs of ReLU-like activation functions and their first derivatives. (a) ReLU, LeakyReLU, and ELU. (b) The first derivatives of ReLU, LeakyReLU, and ELU. (c) GELU, SiLU, and Mish. (d) The first derivatives of GELU, SiLU, and Mish.

2.2 Variants of ReLU

An activation function can be defined as a function $f : \mathbb{R} \to \mathbb{R}$ that is differentiable almost everywhere (Gulcehre et al., 2016). ReLU (Nair & Hinton, 2010) became the most popular activation function after it was proposed to solve the vanishing gradient problem suffered by the saturating activation functions, such as tanh and sigmoid. Glorot et al. (2011) show for the first time that deep

purely supervised networks can be trained using ReLU while only shallow networks can be trained using the tanh non-linearity.

The ReLU function is an identity function for positive arguments and zero otherwise:

$$f(x) = \max(0, x) \tag{1}$$

When ReLU activation is above 0, its derivative is 1. Thus vanishing gradients do not exist along the paths of active hidden units in the network. However, ReLU has a potential disadvantage during optimization because the gradient is 0 whenever the unit is not active (Maas et al., 2013). Unlike sigmoid or tanh, ReLU is not *right saturate*. The limit of $f'(x)$ is one, not zero when $x \to \infty$.

Definition of Variants of ReLU. There are many functions that fall in the category of *variants of ReLU* or *ReLU-like* activation functions. However, they have different properties from each other. Since it is still not clear what the definition of variants of ReLU is, in this paper, we define a variant of ReLU or ReLU-like activation function as an activation function f such that the limit of $f'(x)$ is one when $x \to \infty$.

In addition to ReLU, widely used variants of ReLU include LeakyReLU (Maas et al., 2013), ELU (Clevert et al., 2015), GELU (Hendrycks & Gimpel, 2016), SiLU (Elfwing et al., 2018), and Mish (Misra, 2019). Figure 2 compares these six ReLU-like activation functions. It also shows the first derivatives of these activation functions. Section A in the appendix describes the definitions of the five variants other than ReLU.

Other activation functions. There are many studies on new activation functions. In addition to the variants of ReLU mentioned above, they include Maxout (Goodfellow et al., 2013), Parametric Rectified Linear Unit (PReLU) (He et al., 2015), a noisy activation function by (Gulcehre et al., 2016), Swish (Ramachandran et al., 2017), Scaled Exponential Linear Units (SELUs) (Klambauer et al., 2017), a probabilistic activation function by (Lee et al., 2019), and ConvReLU (Gao et al., 2020).

3 ZEROLIERS

ZeroLiers exploits the non-linearity of the ReLU variants. However, it is significantly different from them in that ZeroLiers relates overfitting to outliers in activations and eliminates their effect by setting them to zeros.

3.1 THE ACTIVATION FUNCTION

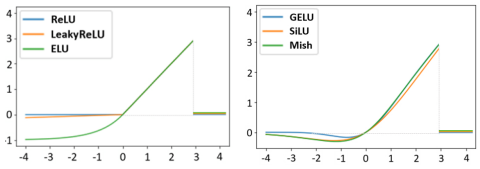
ZeroLiers' activations can be made by just replacing large outliers of original ReLU-like activations with zeros. Consider a neural network with L hidden layers. Let $l \in \{1, \cdots, L\}$ be the indices of the hidden layers. Suppose the original activation function $f : \mathbb{R} \to \mathbb{R}$ is one of the variants of ReLU mentioned previously. The key idea of ZeroLiers is that like a variant of ReLU, ZeroLiers transfers the input of a unit to the output as it is when it is smaller than a specific threshold. Otherwise, ZeroLiers treats it as an outlier and makes the output zero. At a layer l, ZeroLiers' activation function $g : \mathbb{R} \to \mathbb{R}$ and its first derivative g' are defined by,

$$g(x) = \begin{cases} f(x) & \text{if } x \leq \mu^l + k\sigma^l \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad g'(x) = \begin{cases} f'(x) & \text{if } x \leq \mu^l + k\sigma^l \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $f(x)$ is the original activation function, μ^l and σ^l are the mean and standard deviation of the original activations at layer l, respectively, and the constant $k \geq 0$ defines and controls the distance between the threshold $\mu^l + k\sigma^l$ and μ^l.

The presence of outliers in activations was first considered by Mazumdar & Rawat (2018). However, ZeroLiers is significantly different from theirs because we relate the outliers to overfitting. If the value of the original activation of a unit at layer l is larger than $\mu^l + k\sigma^l$, we treat the value as an *outlier*. We introduce only one hyperparmeter k in Equation 2.

Figure 3 shows the action of ZeroLiers for the six variants of ReLU. When overfitting occurs in the training of a DNN model, we replace the original ReLU-like function with ZeroLiers and remove the Dropout step that was combined with the original activation function. As shown in Equation 2, the derivative of ZeroLiers can be easily computed because it is the same as the derivative of f or zero.



(a) (b)

Figure 3: The action of ZeroLiers for the variants of ReLU. (a) For ReLU, LeakyReLU, and ELU. (b) For ReLU, LeakyReLU, and ELU.

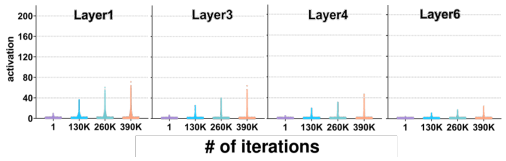
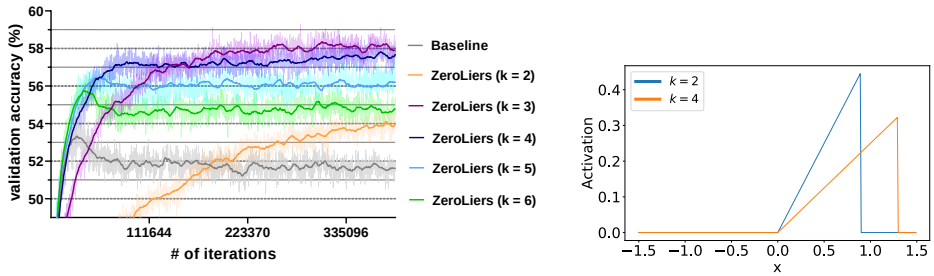


Figure 4: Training a seven-layer MLP on CIFAR-10 with ZeroLiers when $k = 3$.

Figure 4 shows the activation values with ZeroLiers when $k = 3$. We obtain the activations of units before setting the activations to zero in ZeroLiers. The trend in the figure is similar to that of Dropout in Figure 1(b). That is, the number of outliers is significantly reduced compared to the baseline in Figure 1(a).



(a) (b)

Figure 5: k as a learnable parameter when the original activation function is ReLU. (a) Training an MLP on CIFAR-10 using ZeroLiers: validation accuracy over iterations using various values of k. (b) $\tilde{g}(x, k)$ of ZeroLiers-L-k when k varies given the same value of k_0.

3.2 k AS A LEARNABLE PARAMETER

Figure 5(a) shows the result of training a seven-layer MLP on CIFAR-10 by applying ZeroLiers to ReLU. It plots the validation accuracy over iterations using different values of k. Note that k is the parameter that identifies adversely behaving outlier activations. The baseline model of comparison is the case when ReLU is used without Dropout. In this figure, the behavior of the validation accuracy with respect to k does not show any specific tendency. However, certainly the MLP using ZeroLiers does not overfit while the baseline does. When $k = 3$, ZeroLiers achieves the best validation accuracy. The result implies that determining the best value of k should be done empirically. However, the search space of k is huge if it is assumed to be a real value. Instead, we find an integer value for k empirically and use it as an initial value to set k up as a learnable parameter.

If k is a learnable parameter, ZeroLiers can be trained by backpropagation algorithms (LeCun et al., 1989). However, as shown in Equation 2, g is not a function of k and neither g'. The gradient of g with respect to k is always zero if $x \neq \mu^l + k\sigma^l$. Consequently, it does not contribute to the minimization of the training loss during backpropagation. To solve this problem and to make k a learnable parameter, we modify the activation function g as follows:

$$\tilde{g}(x, k) = \frac{k_0}{k} g(x) \quad \text{and} \quad \frac{\partial \tilde{g}(x, k)}{\partial k} = -\frac{k_0}{k^2} g(x) \tag{3}$$

where k_0 is determined empirically and is used to set $k = k_0$ initially during training. We name ZeroLiers with a learnable parameter k as *ZeroLiers-L-k*.

Suppose that k_0 is the actual cutoff threshold value of k that makes outliers set to zeros. When $k_0 < k$, g may not filter out enough outliers. Otherwise, g may filter out too many outliers. In either way, ZeroLiers' regularization effect is smaller than that of setting up $k_0 = k$. To compensate for the smaller regularization effect, we multiply k_0/k to g to construct $\tilde{g}(x, k)$ and to make the magnitude of unfiltered adversely affecting outliers smaller when $k_0 < k$ and bigger when $k_0 > k$.

For example, Figure 5(b) shows how the values of $\tilde{g}(x, k)$ change when the value of k increases from 2 to 4 given the same value of k_0. As shown in Equation 3, the value of the original function ReLU is reflected in the output inversely proportional to k.

4 EXPERIMENTS

In this section, we evaluate ZeroLiers by comparing it with Dropout. We first show the effectiveness of ZeroLiers using a multi-layer perceptron (MLP), a standard Autoencoder (AE) (Rumelhart et al., 1988), a Denoising Autoencoder (DAE) (Vincent et al., 2008). Then we show its effectiveness on more general DNN models with fully connected layers, such as ResNet (He et al., 2016), VGG (Simonyan & Zisserman, 2014), Transformer (Vaswani et al., 2017), BERT (Devlin et al., 2018), and GPT-2 (Radford et al., 2019).

Since Dropout is effective when overfitting occurs, we induce typical overfitting behaviors of the target network by increasing the number of its learnable parameters or by appropriately adjusting its training data size. The criterion of detecting overfitting is the case when the validation loss diverges or does not decrease while training loss is decreasing. When we train the models with Dropout, we try various values of the dropout probability q and choose the one that gives the best result.

The model architectures and datasets used in the evaluation are summarized in Table4 and Table 5 in Section B.1 of the appendix, respectively. In addition, Figure 9 in the appendix shows the way how to modify a given fully connected layers for the experiments.

Table 1: Validation accuracy (%) on CIFAR-10 after training a seven-layer MLP.

Original activation	Dropout or ZeroLiers	Validation accuracy			Epochs
		Top-1	Top-5	Top-10	
ReLU	Baseline	54.42	54.23	54.11	500
	Dropout (q = 0.3)	57.07	56.99	56.94	500
	ZeroLiers (k = 3)	59.31	59.12	59.03	500
	ZeroLiers-L-k (k_0 = 3)	**59.67**	**59.63**	**59.60**	500
LeakyReLU	Baseline	54.95	54.33	54.20	500
	Dropout (q = 0.2)	57.28	57.04	56.98	500
	ZeroLiers (k = 3)	59.43	59.31	59.22	500
	ZeroLiers-L-k (k_0 = 3)	**59.56**	**59.39**	**59.32**	500
ELU	Baseline	58.14	57.65	57.53	1000
	Dropout (q = 0.3)	58.13	58.05	58.00	1000
	ZeroLiers (k = 3)	58.74	58.60	58.54	1000
	ZeroLiers-L-k (k_0 = 3)	**59.74**	**59.60**	**59.56**	1000
GELU	Baseline	56.70	56.41	56.28	500
	Dropout (q = 0.2)	57.35	57.25	57.19	500
	ZeroLiers (k = 3)	59.89	59.80	59.75	500
	ZeroLiers-L-k (k_0 = 3)	**60.01**	**59.94**	**59.91**	500
SiLU	Baseline	57.64	57.58	57.50	1000
	Dropout (q = 0.3)	58.51	58.46	58.41	1000
	ZeroLiers (k = 3)	60.76	60.60	60.52	1000
	ZeroLiers-L-k (k_0 = 3)	**61.11**	**60.95**	**60.91**	1000
Mish	Baseline	57.13	56.96	56.90	1000
	Dropout (q = 0.3)	58.40	58.33	58.27	1000
	ZeroLiers (k = 3)	60.41	60.32	60.27	1000
	ZeroLiers-L-k (k_0 = 3)	**61.01**	**60.87**	**60.77**	1000

Table 2: Validation loss on CIFAR-10 after training an eight-layer AE

Original activation	Dropout or ZeroLiers	Validation loss			Epochs
		Top-1	Top-5	Top-10	
ReLU	Baseline	0.485	0.494	0.500	500
	Dropout (q = 0.05)	0.425	0.428	0.431	500
	ZeroLiers-L-k (k_0 = 3)	**0.311**	**0.313**	**0.315**	500
LeakyReLU	Baseline	0.524	0.537	0.543	500
	Dropout (q = 0.05)	0.427	0.439	0.442	500
	ZeroLiers-L-k (k_0 = 3)	**0.302**	**0.305**	**0.307**	500

Table 3: Validation loss on CIFAR-10 after training an eight-layer DAE

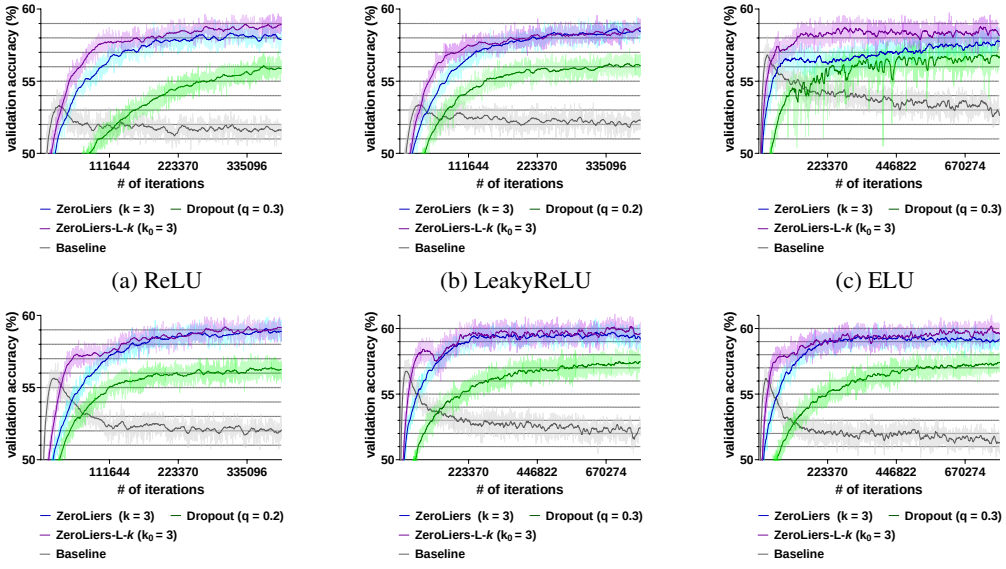
Original activation	Dropout or ZeroLiers	Validation loss			Epochs
		Top-1	Top-5	Top-10	
ReLU	Baseline	0.509	0.515	0.521	500
	Dropout (q = 0.05)	0.423	0.433	0.436	500
	ZeroLiers-L-k (k_0 = 3)	**0.302**	**0.303**	**0.305**	500
LeakyReLU	Baseline	0.559	0.563	0.568	500
	Dropout (q = 0.05)	0.412	0.419	0.424	500
	ZeroLiers-L-k (k_0 = 3)	**0.302**	**0.303**	**0.305**	500

4.1 MLPS

We first verify that ZeroLiers is more effective than Dropout when training an MLP that uses the variants of ReLU. We train MLP that contains 7 fully connected layers with ReLU, LeakyReLU with $\alpha = 0.01$, ELU with $\alpha = 1.0$, GELU, SiLU, and Mish on CIFAR-10 (Krizhevsky et al., 2009) with a batch size of 64 for 500 epochs. Each hidden layer of the MLP contains 1024-hidden units. We use the Adam optimizer (Kingma & Ba, 2014) and weights are initialized using the uniform distribution described in He et al. (2015). To monitor progress and detect the overfitting behavior, we use the test images of CIFAR-10 as a validation dataset.

Table 1 shows the comparison between ZeroLiers and Dropout. We compare the average top-N validation accuracies of the baseline, Dropout, ZeroLiers, and ZeroLiers-L-k for each variant of ReLU. The baseline model of comparison is the case when one of the variants of ReLU is used without Dropout. We perform experiments with various values of k for ZeroLiers and choose the

value that achieve the best validation accuracy. We also use it as the value of k_0 in ZeroLiers-L-k. Each layer in the MLP has a different learnable parameter k. Interestingly, we observe that ZeroLiers-L-k achieves the highest accuracy compared with the baseline, Dropout, and ZeroLiers for all ReLU-like activation functions.



(a) ReLU (b) LeakyReLU (c) ELU

(d) GELU (e) SiLU (f) Mish

Figure 6: Validation accuracy of training the seven-layer MLP on CIFAR-10 over iterations for the six variants of ReLU.

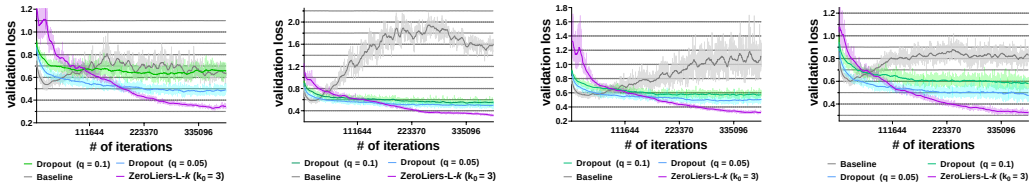
Moreover, as shown in Figure 6, we observe that the validation accuracies of ZeroLiers and ZeroLiers-L-k increase more rapidly and are much better than those of the baseline and Dropout. In addition, the validation accuracy of ZeroLiers-L-k is much better than that of ZeroLiers. When we see the case of the baseline for each activation function, we observe that the validation accuracy abruptly decreases at the beginning of training. This implies that the baseline model overfits CIFAR-10.

Update pattern of k in training. In the MLP with ZeroLiers-L-k, each layer has a different k as its own learnable parameter. We observe two things for the update pattern of k during training. One is that it is different for a different layer and a different original activation function. The other is that activation functions that have a similar shape (e.g., groups of {ReLU, LeakyReLU}, {ELU}, and {GELU, SiLU, Mish}) have a similar update pattern of k for each layer. Figure 10 in Section C.1 of the appendix shows the update patterns of k for all the six variants of ReLU.

Effect of optimizers. To investigate the effectiveness of ZeroLiers across various optimization algorithms, we also train an MLP using SGD, SGD with momentum, and RMSprop (Tieleman et al., 2012). Figure 11(a) in Section C.2 of the appendix compares the validation accuracy of the seven-layer MLP with ZeroLiers-L-k ($k_0 = 3$) for various gradient descent optimizers. The original activation function used in the MLP is ReLU. The validation accuracy of the SGD with momentum optimizer increases more rapidly than those of others at the beginning. However, the validation accuracies of SGD and RMS prop optimizer increases more rapidly than those of others later. The SGD optimizer achieves the highest accuracy. The result indicates that the convergence speed and validation accuracy of the MLP using ZeroLiers-L-k highly depend on the optimizer used.

4.2 AUTOENCODERS

We also train a standard AE (Rumelhart et al., 1988) and a DAE (Vincent et al., 2008) on CIFAR-10. To induce overfitting behaviors, we use networks that contain 8 fully connected layers. The networks consist of 7 hidden layers of widths 1024, 512, 256, 128, 256, 512, 1024 and in that order. We use the Adam optimizer, a batch size of 64, and the mean squared loss. The weights are again



(a) AE + ReLU (b) AE + LeakyReLU (c) DAE + ReLU (d) DAE + LeakyReLU

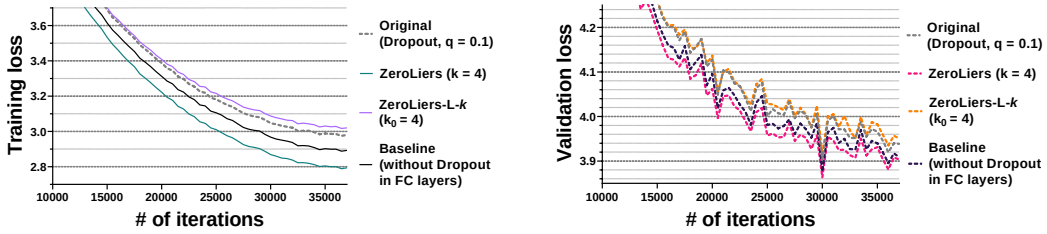
Figure 7: The validation losses of the eight-layer AE and DAE on CIFAR-10. ZeroLiers-L-k ($k_0 = 3$) is compared with the baseline model and Dropout.

initialized using the uniform distribution described in He et al. (2015), and we use the test images of CIFAR-10 as a validation dataset.

Table 2 and Table 3 show the results of the comparison between ZeroLiers and Dropout for the AE and DAE, respectively. We compare the average top-N validation losses of the baseline, Dropout, and ZeroLier-L-k for ReLU and LeakyReLU. Similar to the seven-layer MLP, we observe that ZeroLiers-L-k achieves the best validation loss compared with the baseline and Dropout. We do not include the results for GELU, ELU, SiLU, and Mish in the tables because the AE and DAE using them do not overfit CIFAR-10 with the same number of parameters used for the AE and DAE using ReLU or LeakyReLU.

Figure 7 shows the validation loss of the AE and DAE over iterations. Clearly, the baseline model overfits CIFAR-10. We also observe that ZeroLiers-L-k finally achieves the lowest validation loss even though Dropout converges faster at the beginning of the training.

Effect of optimizers. We also train AEs using SGD, SGD with momentum, and RMSprop. Figure 11(b) in Section C.2 of the appendix shows the validation loss of the eight-layer AE with ZeroLiers-L-k ($k_0 = 3$) for various gradient descent optimizers. The original function used in the AE is ReLU. In this case, the validation loss of the RMSprop optimizer decreases most rapidly and achieves the lowest. Similar to the seven-layer MLP, the result indicates that the convergence speed and validation loss of the AE using ZeroLiers-L-k highly depend on the optimizer used.



(a) Training loss (b) Validation loss

Figure 8: The training and validation losses of the three-layer BERT model on Wikitext-2.

4.3 BERT

BERT is pre-trained contextual representations based on a huge multilayer Transformer encoder architecture (Vaswani et al., 2017). We train the three-layer BERT (Devlin et al., 2018) from scratch on an unlabeled dataset, Wikitext-2 (Merity et al., 2016), aiming at the masked language modeling task. We use publicly available *Transformers* [1] (Wolf et al., 2020) and *Tokenizers* [2] libraries of HuggingFace.

Figure 8(a) and (b) show the training and validation MLM losses (the lower, the better) over iterations of the three-layer BERT model on Wikitext-2, respectively. The baseline model is made by removing all Dropouts only in the fully connected layers of the original BERT model. Dropout in the figure is the original BERT model. Note that the original BERT model uses GELU in its fully

[1] https://github.com/huggingface/transformers
[2] https://github.com/huggingface/tokenizers

connected layers. However, the location of Dropout is not right after GELU in the original BERT model. In both cases of Figure 8(a) and (b), the loss of ZeroLiers decrease much faster than the baseline and Dropout. Moreover, It is consistently smaller than those of Baseline and Dropout.

It is known that l_2-norms of the gradients can be continuously increasing even though the training loss converges (Goodfellow et al., 2016). We observe that the l_2-norms of ZeroLiers are consistently greater than those of Dropout and the baseline. This implies BERT with ZeroLiers learns faster and explains its faster convergence than Dropout and the baseline.

4.4 GPT-2

We train another Transformer-based model, GPT-2 from scratch. The model consists of a single Transformer decoder layer (Table 6 in Section B.1 in the appendix). We also train the single-layer GPT-2 by modifying its fully connected layers for the standard conditional language modeling task. We appropriately adjust the training dataset (Wikitext-103_{v2}) size and induce overfitting.

Similar to BERT, the best validation loss after applying Dropout using various q values is higher than that of the baseline (Figure 12 in Section C.3 in the appendix). The same thing is true for ZeroLiers, and it is the worst. Moreover, ZeroLiers does not boost the convergence speed of the single-layer GPT-2 either. We observe that l_2-norms of the gradients in ZeroLiers are almost the same as those of the baseline. This explains the reason why ZeroLiers does not boost the convergence speed of the single-layer GPT-2.

We also observe that the single-layer GPT-2 requires a much bigger training dataset than the three-layer BERT to avoid overfitting even though the number of parameters is almost the same as that of the three-layer BERT. We observe that even with a training dataset that is almost $10\times$ bigger than that of the three-layer BERT, the single-layer GPT-2 overfits. We would like to note that the model size might not be a good metric of model complexity that affects overfitting (Neyshabur et al., 2015). We conjecture that different self-attention operations between BERT and GPT-2 are the reason. The masked self-attention in GPT-2 makes more information lost than BERT by masking out almost 50% of values in the input to the softmax function.

4.5 Effect on Other Architectures

CNNs. As Hinton et al. (2012) found that Dropout is far less advantageous in convolutional layers, we also find that both Dropout and ZeroLiers are not effective when training ResNet (He et al., 2016) and VGG (Simonyan & Zisserman, 2014).

Vanilla Transformers. We also perform several experiments with vanilla Transformers (Vaswani et al., 2017). However, the standard Dropout technique has better regularization effect than ZeroLiers. We find using ZeroLiers in each self-attention head is more effective than using it in fully connected layers. For each self-attention head, we put ZeroLiers right before concatenating the attention heads.

5 Conclusions

In this paper, we propose a new activation function ZeroLiers to improve the generalization performance of DNNs. ZeroLiers is effective for ReLU-like activation functions. It replaces both the ReLU-like activation function and Dropout and filters out adversely affecting outliers. It has one learnable hyperparameter to set up the cut off value for the outliers. We thoroughly evaluate and investigate its behaviors on CIFAR-10 with an MLP, Autoencoder, and Denoising Autoencoder. We also evaluate its effectiveness for various DNN architectures with fully connected layers, such as BERT, GPT-2, CNNs, and Transformers. The experimental result indicates the following: First, it does not require additional Dropout at fully connected layers. Second, it achieves a much better convergence speed than or comparable to that of Dropout depending on the DNNs with fully connected layers. Finally, it achieves better generalization performance than or comparable to that of Dropout after convergence. Especially, ZeroLiers achieves much better convergence speed and smaller loss than those of Dropout for BERT.

REPRODUCIBILITY STATEMENT

Source code and datasets that are capable of reproducing all results in Table 1, Table 2, Table 3, Figure 6, Figure 7, Figure 8, and Figure 11 are provided as the supplementary materials in the OpenReview submission system. For the result in Figure 12, only the source code and datasets of ZeroLiers-L-*k* ($k_0 = 3$) are provided in the supplementary materials because the maximum file size limit is set to 100MB in the OpenReview system. The description of the commands required to reproduce the results is also included in the supplementary materials.

REFERENCES

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. *Advances in neural information processing systems*, 26:3084–3092, 2013.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35 (1):126–136, 2018.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Expected energy-based restricted boltzmann machine for classification. *Neural networks*, 64:29–38, 2015.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

Hongchang Gao, Jian Pei, and Heng Huang. Demystifying dropout. In *International Conference on Machine Learning*, pp. 2112–2121. PMLR, 2019.

Hongyang Gao, Lei Cai, and Shuiwang Ji. Adaptive convolutional relus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3914–3921, 2020.

Wei Gao and Zhi-Hua Zhou. Dropout rademacher complexity of deep neural networks. *Science China Information Sciences*, 59(7):1–12, 2016.

Liangzhu Ge, Yuexian Hou, Yaju Jiang, Shuai Yao, and Chao Yang. Veca: A method for detecting overfitting in neural networks (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13791–13792, 2020.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890*, 2018.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323. JMLR Workshop and Conference Proceedings, 2011.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pp. 1319–1327. PMLR, 2013.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy activation functions. In *International conference on machine learning*, pp. 3059–3068. PMLR, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Rohit Keshari, Richa Singh, and Mayank Vatsa. Guided dropout. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4065–4072, 2019.

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

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28:2575–2583, 2015.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proceedings of the 31st international conference on neural information processing systems*, pp. 972–981, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Aaron Courville, and Chris Pal. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. Survey of dropout methods for deep neural networks. *arXiv preprint arXiv:1904.13310*, 2019.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Joonho Lee, Kumar Shridhar, Hideaki Hayashi, Brian Kenji Iwana, Seokjun Kang, and Seiichi Uchida. Probact: A probabilistic activation function for deep neural networks. *arXiv preprint arXiv:1905.10761*, 5:13, 2019.

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3. Citeseer, 2013.

Arya Mazumdar and Ankit Singh Rawat. Representation learning and recovery in the relu model. *arXiv preprint arXiv:1803.04304*, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 4:2, 2019.

David S. Moore and George McCabe. *Introduction to the Practice of Statistics*. W. H. Freeman, 4 edition, 2002.

Wenlong Mou, Yuchen Zhou, Jun Gao, and Liwei Wang. Dropout training, data-dependent regularization, and generalization bounds. In *International conference on machine learning*, pp. 3645–3653. PMLR, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401. PMLR, 2015.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pp. 8093–8104. PMLR, 2020.

David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. IEEE Massachusetts, 1988.

Shaeke Salman and Xiuwen Liu. Overfitting mechanism and avoidance in deep neural networks. *arXiv preprint arXiv:1901.06566*, 2019.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. *Advances in neural information processing systems*, 26:351–359, 2013.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066. PMLR, 2013.

Sida Wang and Christopher Manning. Fast dropout training. In *international conference on machine learning*, pp. 118–126. PMLR, 2013.

Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv preprint arXiv:1805.07898*, 2018.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

A DESCRIPTIONS OF THE FIVE RELU-LIKE FUNCTIONS

In this section, we describe the five ReLU-like functions other than ReLU.

LeakyReLU. Since ReLU has a potential disadvantage that the gradient is 0 whenever the unit is not active, LeakyReLU allows for a small, non-zero gradient when the unit is not active. The LeakyReLU function with $0 < \alpha < 1$ is defined by,

$$f(x) = \max(\alpha \cdot x, x), \tag{4}$$

The hyperparameter α prevents the network from learning too slow. LeakyReLU was originally proposed with $\alpha = 0.01$.

ELU. In contrast to ReLU, ELU has negative values that move the mean of activations closer to zero. The ELU function with $\alpha > 0$ is defined by,

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha \cdot (e^x - 1) & \text{otherwise} \end{cases}, \tag{5}$$

where hyperparameter α controls the value to which an ELU saturates for negative net inputs. Pushing the mean of the activations closer to zero speeds up learning by bringing the normal gradient closer to the unit natural gradient (Amari, 1998) because of a reduced bias shift effect (Clevert et al., 2015).

GELU. Even though non-linearities and Dropout together determine a unit's output, they remain distinct. GELU combines properties from Dropout, Zoneout (Krueger et al., 2016), and ReLU together. Zoneout stochastically forces some hidden units to maintain their previous values using random noise to improve generalization. GELU is defined by,

$$f(x) = x \cdot \Phi(x), \tag{6}$$

where $\Phi(x) = P(X \leq x), X \sim N(0, 1)$ is the cumulative distribution function of the standard normal distribution. The standard normal distribution is chosen because input values of units tend to follow a normal distribution, especially with Batch Normalization (Hendrycks & Gimpel, 2016).

SiLU. SiLU is defined by,

$$f(x) = x \cdot \sigma(x), \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$. SiLU has a global minimum value of approximately -0.28 and a self-stabilizing property, which is demonstrated empirically in Elfwing et al. (2015).

Mish. Mish is a self-regularized non-monotonic activation function defined by,

$$f(x) = x \cdot \tanh(\ln(1 + e^x)), \tag{8}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function. Mish often improves the performance of DNNs compared to those of ReLU and LeakyReLU across different computer vision tasks.

B EXPERIMENTAL SETUP

B.1 MODEL ARCHITECTURES AND DATASETS

Table 4 and Table 5 summarize the model architectures and datasets used for the evaluation of ZeroLiers in Section 4. To evaluate ZeroLiers and Dropout, we increase the model size (the number of parameters) or decrease the training data size to introduce overfitting. We train MLPs, Autoencoders, Denoising Autoencoders, and ResNet on CIFAR-10. We use CIFAR-100 to train VGG. The three-layer BERT is trained on Wikitext-2 and the one-layer GPT-2 is trained on Wikitext-103_{v2} that was made with first 102.2MB of Wikitext-103. The reason we made Wikitext-103_{v2} is that the one-layer GPT-2 severely overfits Wikitext-2. Even though we applied Dropout with $q = 0.7$, the validation loss diverges. Therefore, we did not remove the original Dropout at fully connected layers, which is not applied right after GELU, when establishing a baseline for the one-layer GPT-2. Table 6 shows the settings for the baseline model for BERT and GPT-2 in Section 4.3 and Section 4.4.

Table 4: Model architectures

Model	Layers	# of parameters (millions)	Batch size
Multilayer Perceptron	7 FC layers	8.4	64
Autoencoder	8 FC layers	7.7	64
Denoising Autoencoder	8 FC layers	7.7	64
BERT	3 Transformer encoder layers	45.7	128
GPT-2	1 Transformer decoder layers	46.4	24
ResNet	50 weighted layers	45.8	128
VGG	19 weighted layers	143.7	128
Transformer	6 encoder layers & 6 decoder layers	56	256

Table 5: Datasets

Dataset	Source	Task	Training Set Size (MB)
CIFAR-10	Krizhevsky et al. (2009)	Image classification	147.5
CIFAR-100	Krizhevsky et al. (2009)	Image classification	148.1
Wikitext-2	Merity et al. (2016)	Language modeling	10.4
Wikitext-103_{v2}	Merity et al. (2016)	Language modeling	102.2

Table 6: Baseline settings of BERT and GPT-2

Parameter	BERT	GPT-2
Activation function	GELU	GELU
Vocabulary size	30,522	50,257
FC layer dimension	3,072	3,072
# of attention heads	12	12
Epsilon in layer normalization	10^{-12}	10^{-5}
Dimension of encoder and pooling layers	768	768
Dropout in self-attention layers	$q = 0.1$	$q = 0.1$
Original Dropout in FC layers	None	$q = 0.1$
Dropout right after GELU	**None**	**None**
Dropout in other layers	None	$q = 0.1$

B.2 Fully Connected Layer Construction

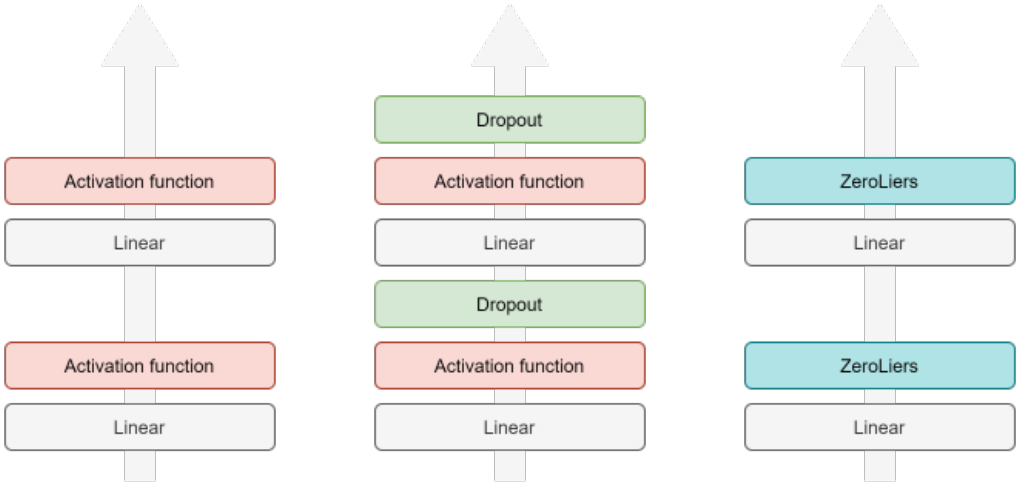
Figure 9 shows the way how to modify a given fully connected layers for the experiments in Section 4. Figure 9(a) shows the baseline. Figure 9(b) is for Dropout. We insert Dropout for the output of the activation function in the baseline. The activation function and the Dropout mechanism together are replaced by ZeroLiers in Figure 9(c) to evaluate ZeroLiers.

C More Experimental Results

In this section, we provide more detailed experimental results with figures for the experiments in Section 4.1 and Section 4.4

C.1 Update Patterns of k in ZeroLiers-L-k

Figure 10 shows the change of the k value over epochs in ZeroLiers-L-k for the seven-layer MLP. As described in Section 4.1, each layer has a different k as its own learnable parameter. The graphs show two things. One is that the update pattern of k is different for a different layer and a different original activation function. The other is that activation functions that have a similar shape (e.g., the group of {ReLU and LeakyReLU}, and the group of {GELU, SiLU, and Mish}) have a similar update pattern of k for each layer.



(a) Baseline (b) Dropout (c) ZeroLiers

Figure 9: The structure of fully connected layers for each experimental setting.

C.2 Effect of Optimizers

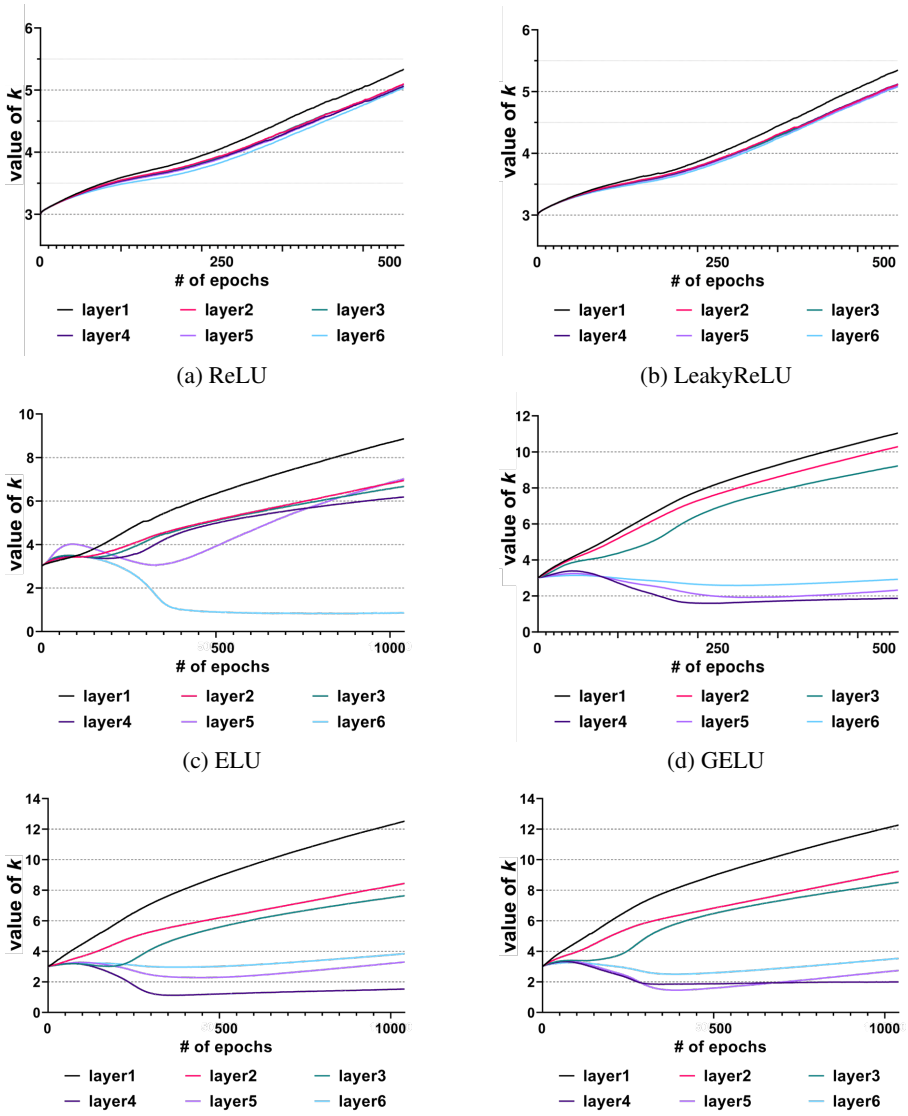
We first use Adam for the MLP, Autoencoder, and Denoising Autoencoder with an initial learning rate of 0.0001. Then, we train the MLP and Autoencoder with ZeroLiers-L-k using various stochastic optimization algorithms, such as SGD, SGD with momentum, and RMSprop (Tieleman et al., 2012), to investigate the effectiveness of ZeroLiers across the optimization algorithms. The learning rate of SGD and SGD with momentum is 0.01, and that of RMSprop is 0.001. The momentum coefficient is fixed to 0.9 when using SGD with momentum.

On the seven-layer MLP. Figure 11(a) compares the validation accuracy of the seven-layer MLP with ZeroLiers-L-k ($k_0 = 3$) for various gradient descent optimizers, such as SGD, SGD with momentum, and RMSprop. The original activation function used in the MLP is ReLU. The validation accuracy of the SGD with momentum optimizer increases more rapidly than those of others at the beginning. However, the validation accuracies of SGD and RMS prop optimizer increases more rapidly than those of others later. The SGD optimizer achieves the highest accuracy. The result indicates that the convergence speed and validation accuracy of the MLP using ZeroLiers-L-k highly depend on the optimizer used.

On the eight-layer AE. Figure 11(b) shows the validation loss of the eight-layer AE with ZeroLiers-L-k ($k_0 = 3$) for various gradient descent optimizers, such as SGD, SGD with momentum, and RMSprop. The original function used in the AE is ReLU. In this case, the validation loss of the RMSprop optimizer decreases most rapidly and achieves the lowest. Similar to the seven-layer MLP, the result indicates that the convergence speed and validation loss of the AE using ZeroLiers-L-k highly depend on the optimizer used.

C.3 Experiment with GPT-2

Figure 12 shows the training and validation losses of the one-layer GPT-2 in Section 4.4 on Wikitext-103_{v2} dataset. Unlike BERT, the best validation loss after applying Dropout is larger than that of the baseline. The same thing is true for ZeroLiers, and it is the worst. Moreover, ZeroLiers does not boost the convergence speed of GPT-2 either.

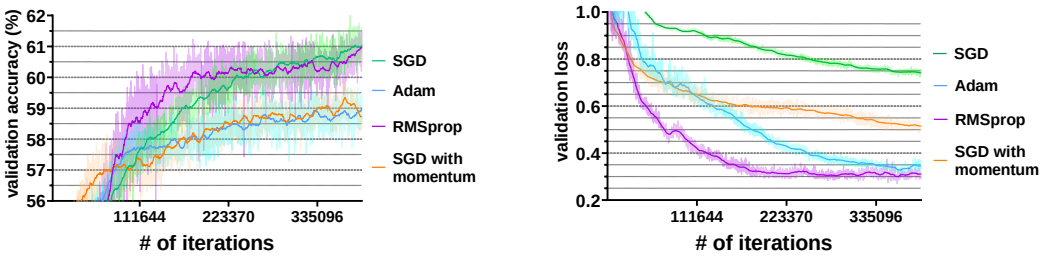


(a) ReLU (b) LeakyReLU

(c) ELU (d) GELU

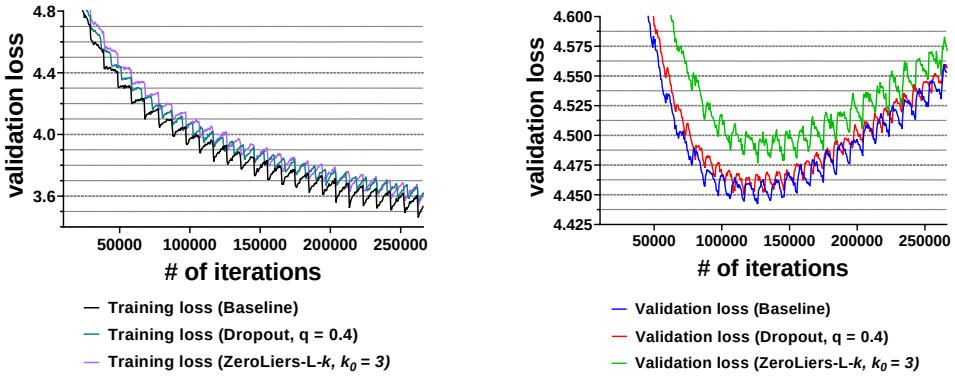
(e) SiLU (f) Mish

Figure 10: Change of the k value over epochs in ZeroLiers-L-k for the seven-layer MLP.



(a) The seven-layer MLP (b) The eight-layer AE

Figure 11: The validation accuracy (loss) of the seven-layer MLP and eight-layer AE using ZeroLiers-L-k ($k_0 = 3$) on CIFAR-10 for various optimizers. The original activation function is ReLU. The convergence speed and the validation accuracy (loss) are highly dependent upon the optimizers used.



(a) Training loss

(b) Validation loss

Figure 12: The training and validation losses of the one-layer GPT-2 on Wikitext-103_{v2}.