Parametric Variational Linear Units (PVLUs) in Deep Convolutional Networks

Shikhar Ahuja*
Montgomery High School
Skillman, NJ

ahujashikhar314@gmail.com

Aarush Gupta*
Montgomery High School
Skillman, NJ

aarush.g75@gmail.com

Abstract

The Rectified Linear Unit is currently a state-of-the-art activation function in deep convolutional neural networks. To combat ReLU's dying neuron problem, we propose the Parametric Variational Linear Unit (PVLU), which adds a sinusoidal function with trainable coefficients to ReLU. Along with introducing nonlinearity and non-zero gradients across the entire real domain, PVLU allows for increased model generalization and robustness when implemented in the context of transfer learning. On a simple, non-transfer sequential CNN, PVLU led to relative error decrease of 16.3% and 11.3% without and with data augmentation, relative to ReLU. PVLU is also tested on transfer learning problems. The VGG-16 and VGG-19 models experience relative error reductions of 9.5% and 10.7% on CIFAR-10, respectively, after the substitution of ReLU with PVLU. When training on Gaussian-filtered CIFAR-10 images, similar improvements are noted for the VGG models. Most notably, PVLU fine tuning allows for relative error reductions up to and exceeding 10% on near state-of-the-art ResNet models for both CIFAR-10 and CIFAR-100.

1. Introduction

The Rectified Linear Unit, known as ReLU, has become a standard function in deep neural networks [Nair and Hinton, 2010]. Sigmoidal activation functions saturate as inputs deviate from zero, meaning that their gradients vanish and significant training fails to occur. This follows from noting that $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ and $\sigma(x) \to 1$ as $x \to \infty$ or $x \to -\infty$.

However, ReLU suffers from its own key issue known as the dying neuron problem. Due to its derivative being zero for negative inputs, the use of ReLU may sometimes zero gradients and completely stop local weight updates during training [Gu et al., 2017]. The issue becomes even more prevalent for deeper networks — just a few dying neurons

can zero the long chain rule expressions used to compute training gradients. It has been mathematically proven that the use of ReLU will necessarily lead to the dying neuron problem in the limit of an infinitely deep network [Lu et al., 2020].

Variants of ReLU have been created in the past to address its various issues. Three popular variants are the Exponential Linear Unit (ELU) [Clevert et al., 2016], the Leaky Rectified Linear Unit (Leaky ReLU) [Maas et al., 2013], and the Parametric Rectified Linear Unit (PReLU) [He et al., 2015a]. All three have gotten results in a variety of datasets but have their respective issues. ELU's gradients saturate while Leaky ReLU and PReLU remain unbounded for negative domains.
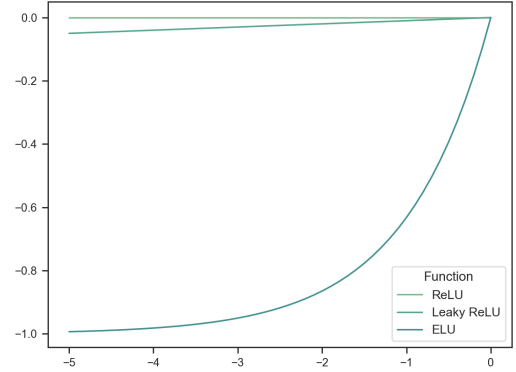

Figure 1. Behavior of ReLU, Leaky ReLU, and ELU for $x < 0$.

While less common, sinusoidal aspects have been incorporated in the past into activation functions [Sitzmann et al., 2020]. In the context of our work, one relevant example is the SineReLU, available in Keras's contrib module [Chollet et al., 2015]. The function is written as the following (assuming ϵ is a hyperparameter):

$$\text{SineReLU}(x) = \begin{cases} x & \text{if } x \leq 0 \\ \epsilon(\sin x - \cos x) & \text{if } x > 1 \end{cases}$$

The SineReLU activation can assist with the dying neuron problem and we further extend on its findings. In do-

ing so, we propose the Parametric Variational Linear Unit (PVLU), which is built up from first considering a simpler Variational Linear Unit (VLU). While it shares similarities to the SineReLU activation discussed before, PVLU introduces a variety of novel features that will be discussed further in our paper. In our paper, we first outline the VLU's theoretical basis and definition. Then, we extend our proposal to a parametrized VLU, or PVLU — applications in fine tuning and transfer learning are also discussed. Empirically, we demonstrate that PVLU performs better than ReLU on a variety of models running on the CIFAR-10 and CIFAR-100 dataset. These include everything from sequential, non-transfer CNNs to deep resiudal network models.

2. Variational Linear Unit

We first propose the Variational Linear Unit (VLU), which is defined as

$$\text{VLU}(x) = \text{ReLU}(x) + \alpha \sin\left(\beta x\right)$$
$$= \max(0, x) + \alpha \sin\left(\beta x\right)$$

where α and β are hyperparameters that can be chosen based on the model. Larger values for both of these parameters inject more periodic variation into the ReLU function. For $\alpha = 0$ or $\beta = 0$, no variation is present.

Note that there are significant differences between the previously seen SineReLU and the currently proposed VLU. VLU introduces a sinusoidal variation throughout the entire real domain $x \in \mathbb{R}$. This differs from not only SineReLU, but also a significant number of previously seen ReLU variants which only alter outputs for domain $x < 0$. In doing so, we add nonlinearity throughout the entire activation function while also preserving one of ReLU's most important properties, which is its unbounded nature for positive values. Similar to SineReLU, adding non-monotonicity to the ReLU activation function may be beneficial as was observed with the Swish activation function [Ramachandran et al., 2017].

The graphs below illustrate VLU's behavior for selected values of α and β.

We can also analyze VLU's properties more rigorously. Let $\mathbf{a}^\ell \in \mathbb{R}^{N_\ell}$ be the set of activations $a_1, a_2, a_3, \ldots, a_{N_\ell}$ in layer ℓ, where N_ℓ is the number of neurons in that layer. Then, we define $z^\ell \in \mathbb{R}^{N_\ell}$ as $W^{\ell-1}\mathbf{a}^{\ell-1} + \mathbf{b}^{\ell-1}$, where $\mathbf{b}^\ell \in \mathbb{R}^{N_\ell}$ is the bias vector. $W^{\ell-1} \in \mathbb{R}^{N_{\ell-1} \times N_\ell}$ is the weight matrix connecting layers $\ell - 1$ and ℓ.

Assuming weight W_{ij}^ℓ connects the j-th neuron in layer ℓ and the i-th neuron in layer $\ell + 1$, recall that

$$z_i^{\ell+1} = \sum_{j=1}^{N_\ell} \left(W_{ij}^\ell a_j^\ell + b_j^\ell\right)$$
$$a_i^{\ell+1} = \alpha \sin\left(\beta z_i^{\ell+1}\right) + \text{ReLU}(z_i^{\ell+1})$$
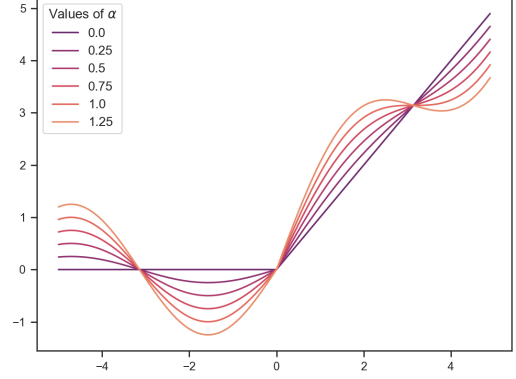
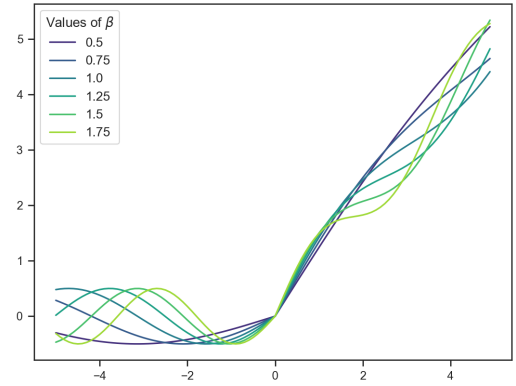Figure 2. Different values of α in $f(x) = \text{ReLU}(x) + \alpha \sin\left(\beta x\right)$.

Figure 3. Different values of β in $f(x) = \text{ReLU}(x) + \alpha \sin\left(\beta x\right)$.

Then, the partial derivatives of the cost function $\mathcal{C}$ with respect to the network weights and biases are:

$$\frac{\partial \mathcal{C}}{\partial W_{ij}^\ell} = \frac{\partial \mathcal{C}}{\partial a_i^{\ell+1}} \frac{\partial a_i^{\ell+1}}{\partial W_{ij}^\ell}$$
$$= \frac{\partial \mathcal{C}}{\partial a_i^\ell} a_j^\ell \left(\alpha\beta \cos\left(\beta z_i^{\ell+1}\right) + H(z_i^{\ell+1})\right)$$
$$\frac{\partial \mathcal{C}}{\partial b_i^\ell} = \frac{\partial \mathcal{C}}{\partial a_i^\ell} \frac{\partial a_i^\ell}{\partial b_i^\ell}$$
$$= \frac{\partial \mathcal{C}}{\partial a_i^\ell} \left(\alpha\beta \cos\left(\beta z_i^{\ell+1}\right) + H(z_i^{\ell+1})\right)$$

where

$$H(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

Note that these gradients are similar to that of ReLU except for the addition of the trigonometric terms. Since sine and cosine do not go to zero for large negative inputs, they help to alleviate the dying neuron problem present in ReLU. However, unlike ReLU, the derivative of our new activation function is no longer static for positive inputs.

3. Parametric VLU

Now we extend the discussion of the VLU to the Parametric VLU, or PVLU. Rather than static constants across the entire network, the parameters α and β are now channelwise and are trained during backpropogation.

Then, our new activation function $A^\ell : \mathbb{R}^{N_\ell} \to \mathbb{R}^{N_\ell}$ in layer ℓ is defined by the channel-wise parameters α and β like so:

$$
A^\ell(\mathbf{z}^\ell) = \begin{pmatrix} \alpha_1^\ell \sin\left(\beta_1^\ell z_1^\ell\right) + \mathrm{ReLU}(z_1^\ell) \\ \vdots \\ \vdots \\ \alpha_{N^\ell}^\ell \sin\left(\beta_{N^\ell}^\ell z_{N^\ell}^\ell\right) + \mathrm{ReLU}(z_N^\ell) \end{pmatrix}
$$

Using the chain rule, we can thus compute the gradients of the cost function $\mathcal{C}$.

$$
\frac{\partial \mathcal{C}}{\partial \alpha_i^\ell} = \frac{\partial \mathcal{C}}{\partial a_i^\ell} \frac{\partial a_i^\ell}{\partial \alpha_i^\ell} = \frac{\partial \mathcal{C}}{\partial a_i^\ell} \sin\left(\beta_i^\ell z_i^\ell\right)
$$

$$
\frac{\partial \mathcal{C}}{\partial \beta_i^\ell} = \frac{\partial \mathcal{C}}{\partial a_i^\ell} \frac{\partial a_i^\ell}{\partial \beta_i^\ell} = \frac{\partial \mathcal{C}}{\partial a_i^\ell} z_i^\ell \cos\left(\beta_i^\ell z_i^\ell\right)
$$

As for the partial derivatives of the cost function with respect to weights and biases, we simply replace α and β from our original expressions with their respective channel-wise parameters.

$$
\frac{\partial \mathcal{C}}{\partial W_{ij}^\ell} = \frac{\partial \mathcal{C}}{\partial a_i^{\ell+1}} \frac{\partial a_i^{\ell+1}}{\partial W_{ij}^\ell}
$$
$$
= \frac{\partial \mathcal{C}}{\partial a_i^\ell} a_j^\ell \left(\alpha_i^{\ell+1} \beta_i^{\ell+1} \cos\left(\beta_i^{\ell+1} z_i^{\ell+1}\right) + H(z_i^{\ell+1}) \right)
$$
$$
\frac{\partial \mathcal{C}}{\partial b_i^\ell} = \frac{\partial \mathcal{C}}{\partial a_i^\ell} \frac{\partial a_i^\ell}{\partial \beta_i^\ell}
$$
$$
= \frac{\partial \mathcal{C}}{\partial a_i^\ell} \left(\alpha_i^{\ell+1} \beta_i^{\ell+1} \cos\left(\beta_i^{\ell+1} z_i^{\ell+1}\right) + H(z_i^{\ell+1}) \right)
$$

3.1. Fine Tuning

One of the major applications of PVLU in transfer learning is fine tuning. Given a pretrained deep model, all occurrences of ReLU can be replaced with PVLU where α is initialized to zero. Similar work with the already existing PReLU activation function (and even linear activations) has already been done in the past [Dong et al., 2017]. This application allows PVLU to enhance models which are already performing at state-of-the-art accuracies.

Specifically, the use of PVLU can be thought of as adding a layer of trainable periodic noise to a network when applied in the context of transfer learning. Noisy activation functions have been shown to improve model performance [Gulcehre et al., 2016]. What seperates this proposal from previous literature relating to noisy activation functions is that the added noise is not purely entropic.

4. Empirical Results

Initially, we tested shallow architectures on various simple datasets such as MNIST and Fashion-MNIST. To test our proposal, we interchanged ReLU activations with PVLU. However, there was no significant difference or improvement after making this change. This result is logical as the dying neuron problem is not evident or impactful for networks that are not significantly deep. Thus, PVLU may add unnecessary complexity without real results for simple computer vision problems — the function's real applications are present in deeper and more complex networks.

4.1. Initial Tests on CIFAR-100

For the CIFAR-100 dataset, we utilized a deeper architecture where the dying neuron problem would be prevalent.

For our first tests, we did not use data augmentation in assessing performance performance. To initially investigate the plausibility of PVLU, we test it on a relatively simple convolutional neural network. Our model is adapted from

`github.com/andrewkruger/cifar100_CNN`.

The model has six convolutional layers, each followed by a ReLU activation layer. The model also includes maxpooling, dropout, and dense layers. As a starting point, we run CIFAR-100 without data augmentation. In addition to testing ReLU as a baseline along with PVLU, we compare our proposal to a variety of existing variants (Leaky ReLU, ELU, PreLU). For the sake of consistency and reproducibility, we set random seeds using the Tensorflow function `tf.random.set_seed` during each of our trials. As a starting point, the values of α and β are set to 0.5 and 1, respectively. This ensures that the activations maintain a notable magnitude of sinusoidal nonlinearity.

Our peak test accuracies for five trials conducted are shown in the table below.

Seed	ReLU	Leaky	PReLU	PVLU (ours)
0	0.7%	50.1%	56.0%	56.6%
1	49.7%	50.5%	55.5%	56.6%
2	0.8%	50.5%	55.6%	57.8%
3	48.7%	50.7%	54.9%	56.7%
4	47.2%	50.3%	55.0%	56.7%
Mean	**48.5%**	**50.4%**	**55.5%**	**56.9%**
Std. Err.	**0.7%**	**0.1%**	**0.2%**	**0.2%**

Table 1. Peak test accuracies of ReLU, PVLU, Leaky ReLU, and PReLU on simple non-transfer CNN without data augmentation.

Note: ReLU "dies" several times in our trials. To fairly compare the activation functions, we only consider trials in which ReLU performs reasonably.

Examining the mean peak test accuracies of all activation functions along with their respective standard errors, PVLU performs the best with PReLU more than a percent in accuracy behind. These results can be clearly interpreted as statistically significant from comparing the standard errors in accuracies to their differences. As can be seen from the data, ReLU unexpectedly leads to a dying model for some trials — this may be reflective of its volatile nature as it relates to model initialization.

We can also examine the curves which show test accuracy as a function of epochs for each of the different activations. These curves are generated from the best trials of each function.
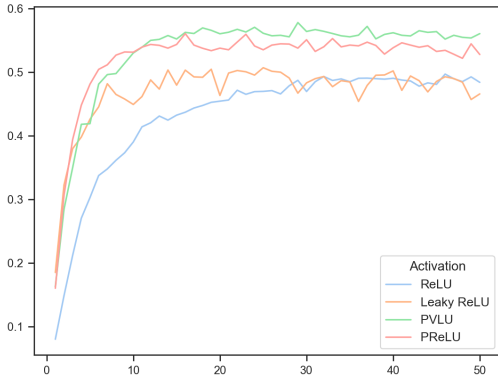

Figure 4. Test accuracy vs. epochs trained for ReLU, Leaky ReLU, PReLU, and PVLU on a non-transfer CNN without data augmentation.

PVLU and PReLU can be seen to converge far quicker than ReLU and its other variants. This fact can most likely be attributed to their parametrization and thus faster training.

While the above accuracies are expected for a relatively simple model, we also test PVLU's effects on more performant models. To do this, data augmentation is now applied while training on the same model. We use three trials instead of five due to computational limits. Additionally, Leaky ReLU is not included in our testing due to its subpar performance as compared to PReLU in previous tests. Our results are summarized below.

With data augmentation, ReLU converged at 62 percent while PVLU and PReLU converged at 67% — this model performance is on the higher side for a sequential model that does not employ transfer learning. Based on the standard error intervals, PVLU and PReLU performances were statistically similar. However, PVLU performed better when comparing top trials while PReLU had a higher mean accuracy.

Seed	ReLU	PReLU	PVLU (ours)
0	61.9%	67.2%	67.4%
1	61.9%	66.6%	65.9%
2	61.7%	66.9%	64.9%
Mean	**61.8%**	**66.9%**	**66.1%**
Std. Error	**0.1%**	**0.7%**	**0.1%**

Table 2. Peak test accuracies of ReLU, Leaky ReLU, PReLU, and PVLU (ours) on simple non-transfer CNN without data augmentation.

Examining the trend of test accuracies for the best trial of each of the activation functions, it is seen that both PVLU and PReLU converge slower than ReLU (unlike our tests without data augmentation). Even approaching the hundredth epoch, PVLU and PReLU experience slow improvements in test accuracies.
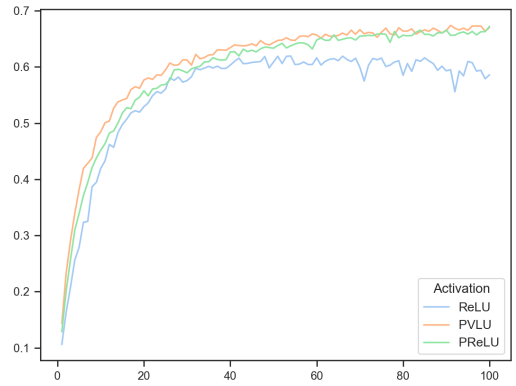

Figure 5. Test accuracy vs. epochs trained for ReLU, Leaky ReLU, PReLU, and PVLU on a non-transfer CNN with data augmentation.

4.2. VGG-16 and VGG-19 Models

VGG models are a powerful class of sequential convolutional networks that have achieved notable results on ImageNet and other deep computer vision problems [Simonyan and Zisserman, 2015]. With over one hundred million parameters, VGG models are perfect for transfer learning since complex representations are already prebuilt.

To test the performance of PVLU fine tuning, we first took the existing VGG models and trained their final layers on CIFAR-10 data. After obtaining the final convergence accuracies, we froze all model weights and replaced all instances of ReLU with PVLU ($\alpha = 0$, $\beta = 1$). This ensures that all improvements are solely the result of PVLU fine tuning. Our results are summarized in the table below.

The significant relative error reductions observed for both the VGG-16 and VGG-19 models highlight the effectiveness of PVLU fine tuning in transfer learning problems.

Model	ReLU	PVLU (ours)	Rel. Error Dec.
VGG-16	86.95%	88.19%	**9.5%**
VGG-19	89.98%	91.06%	**10.7%**

Table 3. VGG peak test accuracies for ReLU and PVLU on CIFAR-10. Relative error decrease is defined as $(e_i - e_f)/e_i$.

4.2.1 Noise Adaptability

An important feature of a robust model is an ability to adapt to noise. To test PVLU in this context, a Gaussian filter is added to both training and testing CIFAR-10 data. Gaussian filters are used to blur images by performing convolution operations with a filter whose kernel coefficients follow the form

$$G(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{\sigma^2}\right)$$

where x and y are the horizontal and vertical distances from the point at which the convolution is being performed. We set $\sigma = 1$ when applying the Gaussian filter to CIFAR-10 image data.

Figure 6. CIFAR image before and after Gaussian filter is applied

First, we fine tune the existing VGG-19 model with ReLU activations on CIFAR-10. Then, this same model is evaluated on the noisy data to find a baseline accuracy. When testing PVLU, we freeze existing model weights while keeping the channelwise values of α and β as trainable. Allowing all layers to train in a noisy setting leads to sharp gradient updates that can interfere with and destroy representations in the existing transferred model. The only exception to this was for batch normalization layers — setting these to trainable allows for robust model generalization and adaptability [Ioffe and Szegedy, 2015].

Model	ReLU	PVLU (ours)	Rel. Error Dec.
VGG-16	84.89%	86.30%	**9.3%**
VGG-19	82.34%	84.05%	**9.7%**

Table 4. VGG peak test accuracies for ReLU and PVLU on CIFAR-10 noised using a Gaussian filter. Relative error decrease is defined as $(e_i - e_f)/e_i$.

From the magnitude of relative error reductions observed, it is clear that PVLU can be used to adapt already trained and accurate models to new, noisy conditions. This approach, along with batch normalization, is far more practical than attempting to train an entirely new model on a noisy dataset.

4.3. Residual Neural Networks

In nearly all current state-of-the-art computer vision models, residual neural network architectures are used. Resiudal neural networks, or ResNets, use skip connections between non-adjacent layers to improving network training [He et al., 2015b]. In our tests, we fine-tune a pretrained ResNet-50 model on CIFAR-100, keeping only the batch normalization layers as trainable. We also test the ResNet-101 v2 architecture and ResNet-152 v2 architecture, which use identity mappings as residual connections [He et al., 2016]. To ensure quick and manageable processing times, the original models containing ReLU were trained for 15 epochs. PVLU was then substituted and trained for another 15 epochs. When testing on ResNets, we use the cutout data augmentation technique as a method of regularization for our networks [DeVries and Taylor, 2017].

Model	ReLU	PVLU (ours)	Rel. Error Dec.
R50	95.46%	95.82%	**7.9%**
R101v2	95.71%	96.20%	**11.2%**
R152V2	96.39%	96.49%	**2.8%**

Table 5. ResNet peak test accuracies on CIFAR-10. Relative error decrease is defined as $(e_i - e_f)/e_i$.

Model	ReLU	PVLU (ours)	Rel. Error Dec.
R50	79.9%	80.7%	**4.1%**
R101v2	80.66%	81.29%	**3.3%**
R152v2	81.66%	82.52%	**4.7%**

Table 6. ResNet peak test accuracies on CIFAR-100. Relative error decrease is defined as $(e_i - e_f)/e_i$.

The results are especially significant because they demontrate PVLU's capacity for improvement on models which achieve near state-of-the-art results. It is worth noting that the original time of training (15 epochs) was enough time for the models to converge or come extremely close to converging, because all layers were frozen except for batch normalization. In the tests that we ran, the relative error decreases on ResNet models indicates that PVLU has significant potential in transfer models.

5. Conclusion

The Parametric Variational Linear Unit, or PVLU, leads to notable improvements across a wide variety of computer

vision problems. By introducing a sinusoidal nonlinearity across the ReLU function, PVLU diminishes the dying neuron problem and also increases activation nonlinearity. In our empirical work, we tested PVLU across shallow non-transfer convolutional neural networks, VGG models, and residual neural networks. We also demonstrated that PVLU can lead to increased robustness when a previously-trained model is introduced to a noisy environment. Observed relative error reductions of roughly 10% or more indicate that PVLU is worth further exploring in different contexts.

While this work primary centered around computer vision problems, PVLU may also be investigated in other domains of machine learning problems in the future. Within computer vision, PVLU can be tested on other transfer learning models for training deep networks on even more complex datasets.

References

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *ICML*, pages 807–814. Omnipress, 2010.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Li Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks, 2017.

Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples, 2020. ISSN 1991-7120.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015a.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020.

Francois Chollet et al. Keras, 2015. URL https://github.com/fchollet/keras.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

Xuanyi Dong, Guoliang Kang, Kun Zhan, and Yi Yang. Eraserelu: A simple way to ease the training of deep convolution neural networks, 2017.

Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy activation functions, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.

Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout, 2017.