L*ReLU: Piece-wise Linear Activation Functions for Deep Fine-grained Visual Categorization

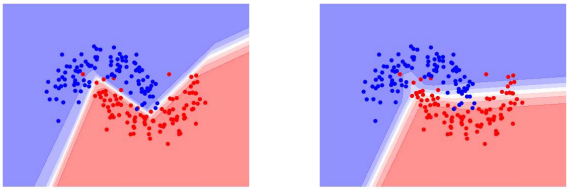
Mina Basirat Peter M. Roth
Institute of Computer Graphics and Vision
Graz University of Technology
{mina.basirat,pmroth}@icg.tugraz.at

Abstract

*Deep neural networks paved the way for significant improvements in image visual categorization during the last years. However, even though the tasks are highly varying, differing in complexity and difficulty, existing solutions mostly build on the same architectural decisions. This also applies to the selection of activation functions (AFs), where most approaches build on Rectified Linear Units (ReLUs). In this paper, however, we show that the choice of a proper AF has a significant impact on the classification accuracy, in particular, if fine, subtle details are of relevance. Therefore, we propose to model the degree of absence and the degree presence of features via the AF by using piece-wise linear functions, which we refer to as L*ReLU. In this way, we can ensure the required properties, while still inheriting the benefits in terms of computational efficiency from ReLUs. We demonstrate our approach for the task of Fine-grained Visual Categorization (FGVC), running experiments on seven different benchmark datasets. The results do not only demonstrate superior results but also that for different tasks, having different characteristics, different AFs are selected.*

1. Introduction

Deep Convolutional Neural Networks (*e.g.*, [1, 2] have recently shown to be very beneficial for a variety of applications in the field of Computer Vision. Thus, there has been, for instance, a considerable interest in designing new network architectures, introducing efficient data augmentation techniques, and improving the parameter optimization. In addition, to increase the robustness and the speed of training also different techniques for initialization (e.g., [3, 4]) and normalization (e.g., [5]) have been explored. However, one relevant and important parameter is mostly ignored, the proper choice of the non-linear activation function (AF).



(a) *L*ReLU* (proposed). (b) *ReLU*.

Figure 1: Binary classification for the two-moon dataset: Just by using a proper activation function a better decision boundary can be estimated.

In fact, recent works have demonstrated that introducing [6–10]) and learning [11–14] new AFs and their parameters are beneficial in terms of convergence speed and training stability, however, also that only minor improvements for the final tasks can be achieved. Thus, most deep learning approaches use Rectified Linear Units (*ReLU*) [15], which have proven to be reliable and allow for fast learning. In this paper, however, we show that the proper choice of the AF can significantly improve the performance of deep neural networks for applications like segmentation, tracking, or object retrieval where subtle visual differences are of relevance. In particular, we demonstrate these benefits for Fine-grained Visual Categorization (FGVC) [16, 17].

In contrast, to Coarse-grained Visual Categorization (CGVC), which aims at distinguishing well-defined categories (*e.g.*, dogs, birds, or man-made objects), the goal of FGVC [16, 17] is to differentiate between hard-to-distinguish classes (*e.g.*, classes belonging to the same category such as different species of birds). Thus, there is a high visual similarity between the classes, where even subtle differences are of relevance. This is illustrated in Fig. 1, where we trained a shallow network (2 layers) for a two-class toy problem (*i.e.*, the two-moon dataset) using different activation functions, namely *ReLU* and the proposed *L*ReLU* with $\alpha = 0.1$). In fact, using the same architecture a bet-

ter decision boundary can be estimated. In particular, the samples close to the true decision boundary, having a small distance to each other, can be classified significantly better.

In this paper, we address the FGCV problem by using a proper AF, modeling the degrees of presence and absence of features [7]. If a feature is present in the current sample, an AF returns a value greater than zero; on the other hand, if a feature is absent, a value smaller or equal to zero is returned. For example, *ReLU* maps the presence of the features via identity, whereas all missing features are mapped to zero. For FGCV, however, it is important also to retain the degree of absence, *i.e.*, the output of the AF in the deactivation state should not be zero.

To ensure the desired properties, we need an AF which is monotonically increasing and uniform-continuous. In other words, the negative values should not saturate, and similar inputs should produce similar outputs. We are ensuring these properties via a piece-wisely defined (on the positive and the negative domain) linear AFs, where the positive part is the identity function and the negative part a linear function with a data-dependent slope. Thus, we refer our method to as *L*ReLU*, indicating the similarities to *Leaky ReLU* (having a slope of 0.01) and the fact that the slope is set according to a data-dependent Lipschitz constant [18]. In this way, not only the desired properties are ensured but also the positive properties of *ReLU* are inherited. The experimental results on seven different datasets clearly show the benefits in terms of classification accuracy compared to the baselines, but also that for different tasks different AFs (*i.e.*, different parameterizations) are necessary.

Thus, the main contributions of this paper can be summarized as follows:

- We propose to model the degree of absence and the presence of features via a properly defined activation function (AF).
- We propose *L*ReLU*, a piece-wise linear AF, where the slope of the negative part is selected according to a properly defined, data-dependent Lipschitz constant.
- We run a thorough experimental analysis on five different Fine-grained Visual Categorization (FGCV) benchmarks and compared to both pre-defined as well as parametric AFs. The results clearly demonstrate the benefits of the approach in terms of improved classification accuracy and applicability for multiple tasks.

2. Related Work

In the following, we first give a short review on Fine-grained Visual Categorization (FGVC), then discuss in detail different AFs for deep network training, and finally summarize the idea of Lipschitz regularization.

2.1. Fine-grained Visual Categorization

To provide a sufficient discriminative capability for FGVC, several techniques have been explored over time. For instance, this can be achieved by learning discriminative features [19, 20], where recently, in particular, Bilinear-CNNs, which compute second-order bilinear features interactions using two symmetric CNNs [16, 21–24] have shown to work very well in practice. Alternative approaches are focusing on local information and model parts of objects [17, 25–30]. To identify and detect informative regions which include important local information, recently, attention models have been emerging [31–35]. In addition, also metric learning approaches (*e.g.*, [36]), which might be most similar to our approach, are applied. In contrast, in this work, we show that the FGVC problem can be addressed by applying more appropriate, task-specific AFs.

2.2. Pre-defined Activation Functions

Starting from simple thresholding functions, initially, the main focus when developing activation functions (AFs) was on squashing functions such as *Sigmoid* and *Tanh* [37]. In particular, as following the universal approximation theorem [37] any continuous real-valued function can be arbitrary well approximated by a feed-forward network with one hidden layer if the AF is continuous, bounded, and monotonically increasing. However, such functions are suffering from the vanishing gradient problem [38], which is, in particular, a problem if the networks are getting deeper.

To overcome this problem, various non-squashing functions were introduced, where, in particular, *ReLU* [15] paved the way for the success for deep learning. As the derivative of positive inputs of *ReLU* is one, the gradient cannot vanish. On the other hand, all negative values are mapped to zero, resulting in two main problems: (1) There is no information flow for negative values, which is known as dying *ReLU*. (2) The statistical mean of the activation values is still larger than zero, leading to a bias shift in successive layers. Moreover, all negative values are treated equally, which is not desirable for the FGVC task! To deal with the dying-*ReLU*-problem *Leaky ReLU* (*LReLU*) [39] introduces a very small negative slope ($\alpha = 0.01$) for the negative part. Even though showing better results for many tasks, the function is still suffering from the bias shift. Slightly differently *Randomized Leaky Rectified Linear Unit* (*RReLU*) [40] sets the slope for the negative part randomly.

Both shortcomings of *ReLU* can be avoided by using *Exponential Linear Unit* (*ELU*) [7], which is robust to noise and eliminates the bias shift in the succeeding layers by pushing the mean activation value towards zero. By returning a bounded exponential value for negative inputs *ELU* is saturated at a predefined threshold. The idea was later extended by introducing *Scaled Exponential Linear Unit*

(*SELU*) [6], showing that the proposed self-normalizing network converges towards a normal distribution with zero mean and unit variance. However, both *ELU* and *SELU* are bounded in the negative part, which is not a desired property for the FGVC task.

2.3. Learned and Parametric Activation Functions

To increase the flexibility, parametric AFs have been proposed, which learn parameters to tune themselves during the training. For instance, *Parametric ReLU* (*PReLU*) [41] builds on the ideas of *LReLU*, but learns the slopes for the negative part based on the training data. Moreover, *SReLU* [42] is defined via three piece-wise linear functions including four adaptive scalar values, forming a crude S shape. Having both convex and non-convex shapes are remarkable characteristics of *SReLU*.

Similarly, *Parametric ELU* [43] evades the vanishing gradient problem and allows for precisely controlling the bias shift by learning parameters from the data. More complex functions (*i.e.*, even non-convex ones) can be learned using *Multiple Parametric Exponential Linear Units* [14], which, in turn, leads to a better classification performance and preferable convergence properties.

The same goal can also be achieved by adopting ideas from reinforcement learning [11] and genetic programming [12], where complex search spaces are explored to construct new AFs. In particular, in [11] *Swish*, a combination of a squashing and a linear function, was found as the best solution for a variety of tasks. Moreover, *Parametric Swish* (*PSwish*) contains a trainable (scaling) parameter. Similar functions, also yielding slightly better results in the same application domains, were also found by [12]. Recently, a theoretic justification for these results have been given in [13], showing that *Swish*-like functions propagate information better than *ReLU*.

However, even though these functions yield good results for CGVC, they cannot cope very well with FGVC tasks. Thus, the goal of this paper is to define AFs better suited for more complex image classification tasks, capturing the small, subtle differences between the rather similar classes more effectively.

2.4. Lipschitz Regularization

Motivated by fact that even small perturbations on the input data can significantly change the output ("adversarial samples") [44, 45], there has been considerable interest in Lipschitz regularization for deep neural network training. Recent works show that given a constrained Lipschitz constant is meaningful for DNNs in terms of robustness and classification accuracy [18, 46, 47]. The Lipschitz constant bounds the ratio of output change to change in its input. In particular, for classification tasks, a small Lipschitz constant improves generalization ability [48, 49]. In this way, in [50] a L_2-nonexpansive neural network is introduced to control the Lipschitz constant and increase the robustness of classifier. Similarly, a robust deep learning model using Lipschitz margin is proposed for object recognition [51]. These works mainly intended to increase the robustness against adversarial samples to guarantee better convergence properties. In contrast, we adopt ideas that relate the separability of multiple classes to the choice of a proper Lipschitz constant for piece-wise linear AFs. This additionally ensures the desired properties to model the degrees of absence and presence of features very well.

3. L*ReLU: Lipschitz ReLU

In the following, in Sec. 3.1, we first discuss the problem of modeling the presence and the absence of features by via AFs. Then, we discuss technical preliminaries on continuity of functions in Sec. 3.2. Finally, in Sec. 3.3, we introduce *L*ReLU*, which copes with the similarity of samples in the FGVC problem much better.

3.1. Presence and Absence of Features

The output a_j of a single neuron j within a neural network is computed by

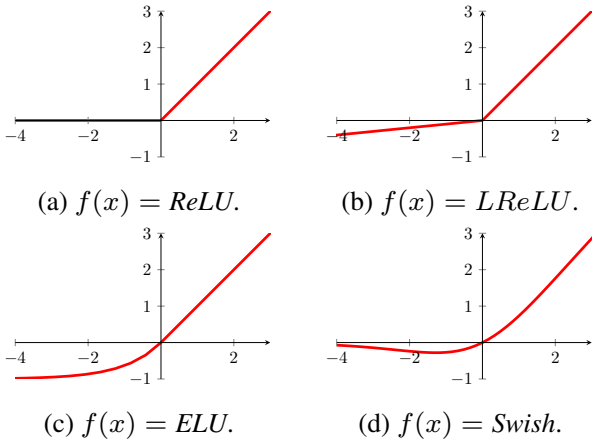
$$a_j = f\left(\sum_{i=0}^{n} w_{i,j} a_i\right), \tag{1}$$

where a_i are the outputs of the n connected neurons (from the previous layer), $w_{i,j}$ are the related weights, and $f(x)$ is a non-linear function referred to as an activation function (AF). In this way, $f(x)$ codes the degree of presence or absence of a feature in the input [7]: a feature is present if $f(x) > 0$ and absent if $f(x) \leq 0$.

The degree of presence is modeled very well for most existing AFs. This can be seen from Figure 2, where we show well-known and widely used AFs, which are defined in Table 1. In these cases, this is achieved via the positive part of the function (*i.e.*, for $x > 0$) being either a linear function (*ReLU*, *LReLU*, *ELU*) or a "quasi-linear" function (*Swish*).

	Name	Function
(a)	*ReLU*	$y(x) = \max(x, 0)$
(b)	*LReLU*	$y(x) = \max(x, 0) + \min(0.01x, 0)$
(c)	*ELU*	$y(x) = \max(x, 0) + \min(e^x - 1, 0)$
(d)	*Swish*	$y(x) = x \cdot \text{sigmoid}(\beta x)$

Table 1: Sample activation functions. For *Swish* the parameter $\beta = 1$ is fixed, however, it can be trained via *PSwish*.



(a) $f(x) = ReLU$. (b) $f(x) = LReLU$. (c) $f(x) = ELU$. (d) $f(x) = Swish$.

Figure 2: Activation functions as defined in Table 1.

On the other hand, the degree of absence of features is not captured very well. For instance, *ReLU* does define a clear "off-state" or "deactivation-state": $f(x) = 0$ for all negative values. In this way, the neuron does not model any information about the degree of absence which can be propagated to the next layer. Similar also applies to *ELU*. Even though $f(x)$ is getting smaller if x is decreased, the function saturates at -1, which represents the "off-state". As a result, the derivations for small values are getting smaller, thus, reducing the information that is propagated to the next layer. *Swish*, in contrast, models the degree of absence for small negative values well but also saturates at 0 if the values are further decreased.

Taking only the presence of features into account is sufficient for many applications including CGVC. If a feature is absent, it is sufficient to say "It's not there!". Therefore, AFs such as *ReLU*, *ELU*, *SELU*, and *Swish* are well suited for CGVC tasks. This is also revealed by our experimental results, where we show that for CGVC the choice of the AF has only a minor impact on the finally obtained accuracy. However, just modeling the presence is not sufficient for FGVC, where the different classes often share a similar appearance and often differ just in subtle visual differences. In these cases, also modeling the degree of absence is necessary. To define functions showing the necessary properties, we need first to review uniform- and Lipschitz-continuous functions.

3.2. Uniform- and Lipschitz-continuous Functions

Even though the following concepts are more general, for reasons of simplicity, we will restrict the discussion to functions in $\mathbb{R}$ [52, 53].

A function $f : \mathbb{R} \to \mathbb{R}$ is called *Lipschitz-continuous* if there exist a constant $L \geq 0$ such that

$$|f(x_i) - f(x_j)| \leq L|x_i - x_j| \tag{2}$$

for all $x_i, x_j \in \mathbb{R}$. Any L fulfilling the condition Eq. (2) is referred to as a *Lipschitz constant*. The minimum $\hat{L}$ of all Lipschitz constants L is often called the *minimal Lipschitz constant*. For $x_i \neq x_j$ we can re-write Eq. (2) to

$$\frac{|f(x_i) - f(x_j)|}{|x_i - x_j|} \leq L. \tag{3}$$

This means that the slopes of secants and tangents in an interval $I \in \mathbb{R}$ are bounded by L. In particular, we have

$$f'(z) \leq L \quad \text{for all } \; z \in I,$$

or in other words:

$$L = \sup_{x \in I} |f'(z)|.$$

In this way, the Lipschitz constant L measures the maximum change rate of function f within an interval I. If $0 \leq L < 1$, then f is called a *contraction mapping* on I.

Moreover, a Lipschitz-continuous function $f : \mathbb{R} \to \mathbb{R}$ is also *uniformly continuous*, that is, for every $\epsilon > 0$ there exits a $\delta > 0$ such that for all $x_i, x_j \in \mathbb{R}$ we have

$$|x_i - x_j| < \delta \Rightarrow |f(x_i) - f(x_j)| < \epsilon. \tag{4}$$

In other words, a uniform-continuous function ensures that $f(x_i)$ and $f(x_j)$ are close to each other if x_i and x_j are sufficiently close to each other.

3.3. Piece-wise Linear Activation Functions

From the discussion above, it is clear that for FGVC an AF is needed, which models both the degree of presence and the degree of absences of features. As these two aspects are related to positive and negative domain of $\mathbb{R}$, we propose to use a piece-wise function for the positive values $x > 0$ and the negative values $x \leq 0$:

$$f(x) = p(x > 0) + n(x \leq 0), \tag{5}$$

with

$$p(x) = \max(\phi(x), 0) \tag{6}$$

and

$$n(x) = \min(\eta(x), 0), \tag{7}$$

where $\phi(x)$ and $\eta(x)$ can be any (non-linear) function $f : \mathbb{R} \to \mathbb{R}$. In this way, we ensure that the positive and the negative part of the piece-wise function reside in the first and third quadrants of a Cartesian coordinate system. It is easy to see that we can easily re-write almost all popular AFs to such a form.

Modeling the presence of features is already realized very well by existing AFs using linear (*ReLU*, *ELU*) or quasi-linear (*Swish*) functions for the positive domain. Modeling the degree of absence of features, however,

is more difficult. In particular, we would need a non-saturating, monotonically increasing function with bounded change rate (*i.e.*, similar inputs should generate similar outputs). Given the definitions from Sec. 3.2 this means that we need contractive, unbounded Lipschitz- and thus also uniform-continuous functions.

However, as can be seen from the example shown in Sec. 4.6, defining such AFs is not trivial. Thus, we propose to use a piece-wise linear approximation:

$$p(x) = \max(x, 0) \tag{8}$$

and

$$n(x) = \min(\alpha x, 0), \tag{9}$$

where $\alpha \geq 0$ defines the slope of the linear function for the negative part.

Indeed, using the parameter setting $\alpha = 0.01$, we get the well-known *LReLU* activation function. However, as we also show in the experiments using such small slopes, which are typically working well for CGVC, fails for FGVC, as the degree of absence of features cannot be modeled very well. The same also applies to *RReLU*, where the slopes are chosen randomly. Thus, a valid choice would be to use *PReLU*, where the slope parameters are estimated from the data. However, the method is not flexible enough in practice, as the optimizer often gets stuck into local minima, which are not generalizing very well. Thus, an initialization close to the optimal solution is needed. We also demonstrate this behavior in the experimental results, where we show that using *PReLU* only gives reasonable results if the approach was initialized close to the optimal solution.

Thus, the critical question is, how to optimally set the slope α? Following the ideas of [12] and [18], we argue that there is no unique solution across different tasks. In particular, [12] shows that for classification tasks of different complexity different AFs are useful. In contrast, [18] proves that given any finite dataset where different classes are separated in the input space by at least a distance of c, there exists a function with Lipschitz constant $c/2$ that correctly classifies all points. In other words, different datasets might have different separability, raising the need for learning functions with different properties. This can be realized by choosing a proper slope according to the Lipschitz-properties of the data.

That is, we call our approach *L*ReLU*, indicating both that the used AF builds on the ideas of *Leaky ReLU* (and PReLU), but also that the slope parameter is chosen according to the Lipschitz-properties of the data. Indeed, our experiments also demonstrate that for the seven different datasets different parameters are needed, however, also that these are not critical. In practice, or each task we can identify a restricted range for the Lipschitz parameter (and thus the slope) [47], yielding stable and reliable classification results.

4. Experimental Results

To demonstrate the importance and the effect of using proper AFs, we run experiments on seven different benchmark datasets: (1) two coarse-grained datasets (*i.e.*, CIFAR-10 and CIFAR-100) [54] and (2) five fine-grained (*i.e.*, Caltech-UCSD Birds-200-2011 [55], Car Stanford [56], Dog Stanford [57], Aircrafts [58], and iFood [59]). We compared our approach to existing AFs, known to yield good results in practice, namely *ReLU* [15], *ELU* [7], *SELU* [6], and *Swish* [11]. Moreover, we also compare to parametric AFs, namely *PReLU* [43] and *PSwish* [11], showing that *L*ReLU* with a proper selected Lipschitz constant yields better results compared to all of these baselines.

In the following, we first describe the used benchmarks and the experimental setup and then discuss the results for both the coarse-grained and the fine-grained datasets in detail.

4.1. Benchmark Datasets

The benchmark datasets described below are illustrated in Figure 3. It can be seen that for the coarse-grained problem (Figure 3 (a)) the single classes are well defined, whereas for the fine-grained problem (Figs. 3 (b)–(f)), the differences are just subtle and often hard to see.

CIFAR-10 and CIFAR-100 The CIFAR-10 dataset consists of 60,000 32×32 colour images in 10 classes, with 6,000 images per class. CIFAR-100 is just like the CIFAR-10, except that it has 100 classes containing 600 images. The data is split into 5,0000 training images and 10000 test images.

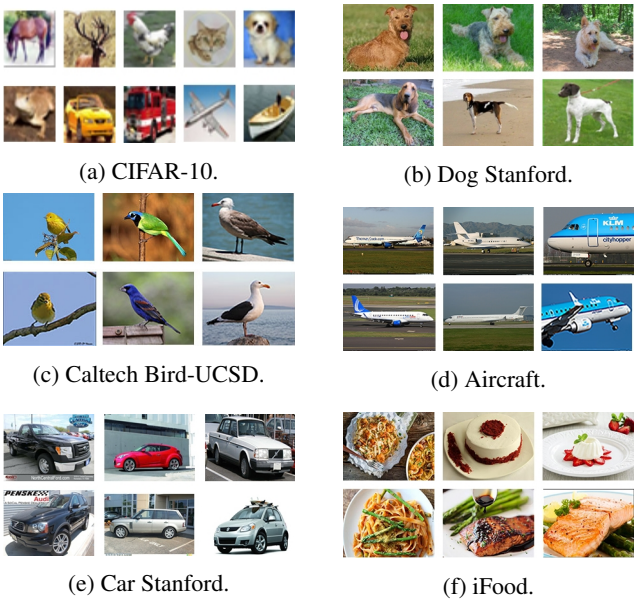
Caltech-UCSD Birds-200-2011 The dataset contains 11,788 images of 200 bird species. Each species is associated with a Wikipedia article and organized by scientific classification (order, family, genus, species). The data is split into 5,994 training images and 5,794 test images [55]).

Car Stanford The dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 test images, where each class has been split roughly in a 50-50 split. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe [56].

Dog Stanford The dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. The total number of images is 20,580. The data is split into 12,000 training images and 8,580 test images [57].

Air crafts The dataset contains 9,960 images of aircraft, with 100 images for each of the 102 different aircraft model variants [58]. The data is split into 3,216 training, 3,231 test and 3,231 validation images.

iFood Dataset This large data-set consist of 211 fine-grained (prepared) food categories with 101,733 training images collected from the web. Test set contains 10,323 images. images[1].



(a) CIFAR-10. (b) Dog Stanford. (c) Caltech Bird-UCSD. (d) Aircraft. (e) Car Stanford. (f) iFood.

Figure 3: Fine-grained visual categorization benchmark datasets (b)–(f) plus CIFAR-10 (a) used in our studies.

4.2. Experimental Setup

To allow for a fair comparison, for all experiments the same experimental setup was used. In particular, to keep the computational cots at a reasonable level (*i.e.*, one NVIDIA Titan XP GPU was used), we re-sized all images to a size of 120×120. We trained an architecture similar to *VGG* cosisting of eight convolutional layers plus two fully connected layers with 400 and 900 units, respectively. Moreover, we used a batch size of 70 and set the maximum pooling size to seven. For training an Adam optimizer with batch normalization was applied. Since we used a different image size and the weights are related to the used AF, we were not able to use pre-trained weights. Thus, we used a random initialization for each training, but—to ensure statistically fair results—we run all experiments three times, where the mean results (and the standard deviations) are shown, respectively.

[1]https://sites.google.com/view/fgvc5/ competitions/fgvcx/ifood

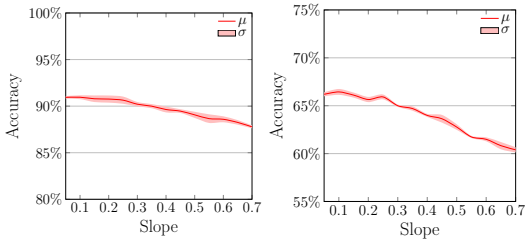
4.3. Coarse-grained Visual Categorization

First of all, we evaluated our approach for the coarse-grained datasets (*i.e.*, CIFAR-10 and CIFAR-100). In Table 2 we show the final averaged results for all methods. The same results also covering the standard deviation are shown in form of boxplots in Figure 6 (a). In addition, in Figure 4, we analyze the classification accuracy when varying the slope. Also here, in addition to the mean, the standard deviation is shown. For CIFAR-10 it can be seen that all AFs perform on par, where *Swish* slightly outperforms the others (which confirms previously published studies). In addition, as can be seen from Figure 4 (a), increasing the slope for *L*ReLU* decreases the classification accuracy, showing that the data is already well separable and that further enforcing the separability is not helpful.

In contrast, for CIFAR-100 we can observe slightly different results. Figure 4 (b) also shows a clear trend that increasing the slope decreases the classification accuracy of *L*ReLU*. However, a slight slope in the range of 0.1 to 0.25 (with a peak at 0.1) demonstrated to be beneficial (compared to *ReLU*) and allows to outperform the baselines. Due to the higher number of classes (*i.e.*, 100 instead of 10), the task is more complex as the classes are getting more similar. Thus, modeling the degree of absence is getting relevant.

Dataset	ELU	Swish	ReLU	SeLU	**L*ReLU**
CIFAR-10	90.81%	**91.23%**	90.83%	89.72%	*90.95%*
CIFAR-100	63.64%	64.36%	*65.32%*	63.29%	**66.44%**

Table 2: Mean accuracy for CGVC: The best result is in **boldface**, the runner up in *italic*.

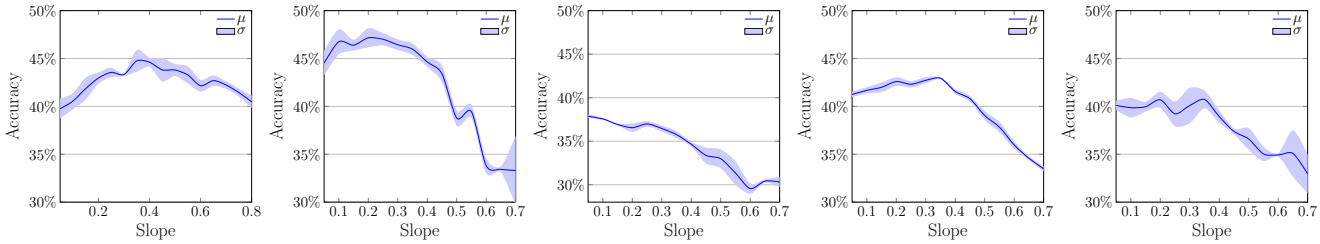


(a) CIFAR-10 (b) CIFAR-100

Figure 4: Class. accuracy of *L*ReLU* vs. slope for CGVC.

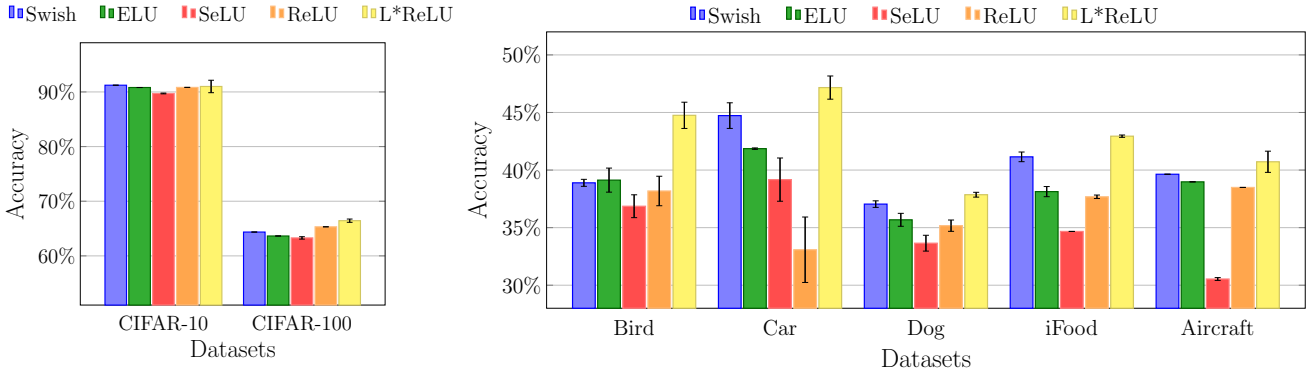
4.4. Fine-grained Visual Categorization

Next, we run the same experiments for the more complex FGVC task. As discussed above, the single classes are more similar as they represent the same general category, making it harder to distinguish the single instances and classes. The finally obtained averaged classification accuracy for all five datasets and all AFs are summarized in



(a) Bird. (b) Car. (c) Dog. (d) iFood. (e) Aircraft.

Figure 5: Class. accuracy of *L*ReLU* vs. slope for FGVC.



(a) (b)

Figure 6: Mean and standard deviation of accuracy for all datasets and AFs: (a) CGVC and (b) FGVC.

Table 3. The same results also covering the standard deviation are shown in form of boxplots in Figure 6 (b). The results show that for all datasets *L*ReLU* finally yields the best and *Swish* the second best results. Whereas the results are close ($+1\%$) for *Dogs* and *Aircraft* for the best and the second best results, the gap is larger for *iFood*, *Car*, and *Birds*: up to $+5\%$. Moreover, it is notable that compared to *ReLU*, which can be seen as a baseline, there is a significant gap for all datasets.

Dataset	ELU	Swish	ReLU	SeLU	**L*ReLU**
Birds 200	*39.12%*	38.89%	38.18%	36.86%	**44.75%**
Car	41.86%	*44.72%*	33.08%	39.17 %	**47.16%**
Dogs	35.67%	*37.04%*	35.17%	33.65%	**37.85%**
iFood	38.12%	*41.14%*	37.67%	34.67%	**42.94%**
Aircraft	38.97%	*39.63%*	38.49%	30.54%	**40.72%**

Table 3: Mean accuracy for FGVC: The best result is in **boldface**, the runner up in *italic*.

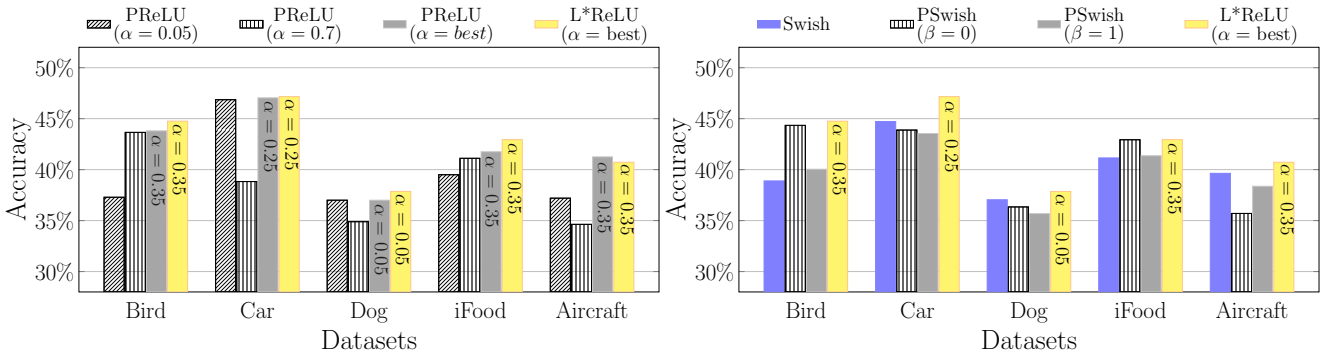
In addition, again we analyze the averaged classification accuracy (plus standard deviation) varying the slops in Figure 5. It can be seen that the different datasets define a different Lipschitz level, showing that applying task-specific AFs is meaningful. Moreover, also these curves show clear trends with clear peaks. Indeed, for all datasets there is constrained slope range of $[0.1, 0.4]$, where stable classification results are obtained. Thus, the selection of the Lipschitz constant for *L*ReLU*is important, but not critical.

4.5. Comparison to Parametric AFs

Next, we give a detailed comparison of *L*ReLU* to parametric AFs, which should better adapt to more complex problems due to trainable parameters? In particular, we compare our approach to *Parametric ReLU* (*PReLU*) [43] and *Parametric Swish* (*PSwish*) [11] using different initializations: for *PReLU* we used a small (0.05), a large (0.7), and the best slope for *L*ReLU* to initialize α; similarly, for *PSwish* a small (0.0) and large (1.0) value was used to initialize β (see Table 1). The corresponding results are shown in Figure 7, respectively. From Figure 7(a) it can be seen that the classification accuracy is highly varying depending on the initialization. Similar also applies to *PSwish*, even though the variation in the accuracy is smaller and thus less sensitive to the initialization.

4.6. Importance of Lipschitz Constant

Finally, we would like to demonstrate the importance of the proper selected Lipschitz constant, by comparing the ac-



(a) (b)

Figure 7: Accuracy for FGVC: *L*ReLU* vs. parametric AFs: (a) *PReLU* and (b) *PSwish*.

curacy of a more complex activation function [13],

$$f(x) = tanh(ax) + bx, \tag{10}$$

with our linear approximation

$$g(x) = \alpha x \tag{11}$$

based on the Lipschitz constant for the negative domain. For the positive domain, we used the identity function, respectively. However, in the following, we are only focusing on the negative domain! Both functions are illustrated in Figure 8. From Eq. (10) we compute the derivation

$$f'(x) = \frac{1}{cos^2(ax)} + b. \tag{12}$$

In this way, for $a = 0.1$ and $b = 0.15$ we compute the Lipschitz constant via $\sup_{x \leq 0}(f'(x)) = 0.25$. Similarly, setting $\alpha = 0.25$ the Lipschitz constant of $g(x)$ is 0.25, as $g(x)$ is a linear function having the same slope for all $x \leq 0$.

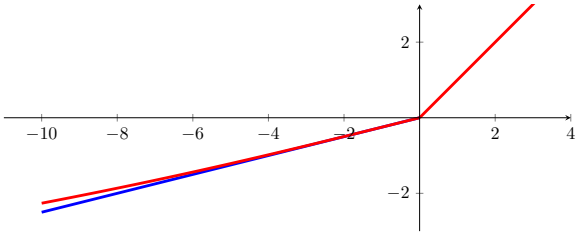


Figure 8: Two functions sharing the same Lipschitz constant $L = 0.25$ for the negative domain: $f(x) = tanh(0.1x) + 0.15x$ (red) and $g(x) = 0.25x$ (blue).

In this way, we set the parameters a and b of $f_{a,b}(x)$ such that the Lipschitz constant matches the best and the worst slope *L*ReLU* for each datasets, respectively. The thus obtained results are shown in Table 4, showing that for all datasets we get a similar classification accuracy, indicating that the Lipschitz constant covers essential information about the data.

Dataset	*L*ReLU* (best α)	$f_{a,b}(x)$ (best α)	*L*ReLU* (α=0.7)	$f_{0.4,0.3}(x)$ (α=0.7)
Car	47.16%	46.49% (a)	33.31%	36.89%
Dogs	37.85%	36.95% (b)	30.32%	31.06%
Birds 200	44.75%	43.16% (c)	42.23%	43.17%
iFood	42.94%	43.21% (c)	33.49%	35.24%
Aircraft	40.72%	40.08% (c)	32.95%	35.04%

Table 4: Activation functions sharing the same Lipschitz constant finally yield a similar classification result: (a) a=0.1, b=0.15; (b) a=0.05, b=0.05; (c) a=0.15, b=0.2.

5. Conclusion and Discussion

In this paper, we demonstrated that using a proper activation function can significantly improve the classification accuracy for the problem of Fine-grained Visual Categorization (FGVC), where subtle differences between similar images are of relevance. Thus, we propose to use activation functions, which model the degrees of presence and absence of features. Whereas the degree of presence is realized via an identity function, the degree of absence can be modeled via monotonically increasing uniform-continuous functions. In our case, we realized this by using piece-wise linear functions, where the slope of the negative part is set according to an optimal Lipschitz constant (given by the data). In this way, we outperform a wide range of fixed and parametric AFs for different FGVC benchmark datasets. Future work would include to automatically estimate the Lipschitz constant from the data and to explore the found properties for different, more complex AFs.

Acknowledgement This work was partially supported by FFG Bridge project SISDAL (21604365). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[3] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int'l Conf. on Machine Learning*, 2013.

[4] D. Mishkin and J. Matas, "All you need is a good init," in *Proc. Int'l Conf. on Learning Representations*, 2017.

[5] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *Proc. Int'l Conf. on Acoustics, Speech and Signal Processing*, 2016.

[6] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances on Neural Information Processing Systems*, 2017.

[7] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. Int'l Conf. on Learning Representations*, 2016.

[8] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, 2018.

[9] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int'l Conf. on Artificial Intelligence and Statistics*, 2011.

[10] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio, "Noisy activation functions," in *Proc. Int'l Conf. on Machine Learning*, 2016.

[11] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," in *Proc. Int'l Conf. on Learning Representations (Workshop track)*, 2018.

[12] M. Basirat and P. M. Roth, "Learning task-specific activation functions using genetic programming," in *Proc. Int'l Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2019.

[13] S. Hayou, A. Doucet, and J. Rousseau, "On the selection of initialization and activation function for deep neural networks," *arXiv:1805.08266*, 2018.

[14] Y. Li, C. Fan, Y. Li, Q. Wu, and Y. Ming, "Improving deep neural network with multiple parametric exponential linear units," *Neurocomputing*, vol. 301, pp. 11–24, 2018.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. Int'l Conf. on Machine Learning*, 2010.

[16] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. Int'l Conf. on Computer Vision*, 2015.

[17] J. Krause, T. Gebru, J. Deng, L.-J. Li, and L. Fei-Fei, "Learning features and parts for fine-grained recognition," in *Proc. Int'l Conf. on Pattern Recognition*, 2014.

[18] J. E. Cohen, T. Huster, and R. Cohen, "Universal lipschitz approximation in bounded depth neural networks," *arXiv:1904.04861*, 2019.

[19] J. Sánchez, F. Perronnin, and Z. Akata, "Fisher Vectors for Fine-Grained Visual Categorization," in *Proc. Workshop on Fine-Grained Visual Categorization (CVPRW)*, 2011.

[20] F. Perronnin and D. Larlus, "Fisher vectors meet neural networks: A hybrid classification architecture," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2015.

[21] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," *CoRR*, vol. abs/1511.06062, 2015.

[22] S. Kong and C. Fowlkes, "Low-rank bilinear pooling for fine-grained classification," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2017.

[23] S. Cai, W. Zuo, and L. Zhang, "Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization," in *Proc. Int'l Conf. on Computer Vision*, 2017.

[24] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie, "Kernel pooling for convolutional neural networks," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2017.

[25] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *Proc. European Conf. on Computer Vision*, 2014.

[26] S. Huang, Z. Xu, D. Tao, and Y. Zhang, "Part-stacked CNN for fine-grained visual categorization," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2016.

[27] K. J. Shih, A. Mallya, S. Singh, and D. Hoiem, "Part localization using multi-proposal consensus for fine-grained categorization," *CoRR*, vol. abs/1507.06332, 2015.

[28] D. Lin, X. Shen, C. Lu, and J. Jia, "Deep LAC: Deep localization, alignment and classification for fine-grained recognition," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2015.

[29] M. Simon and E. Rodner, "Neural activation constellations: Unsupervised part model discovery with convolutional networks," in *Proc. Int'l Conf. on Computer Vision*, 2015.

[30] J. Krause, H. Jin, J. Yang, and F. F. Li, "Fine-grained recognition without part annotations," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2015.

[31] Y. Yu, Z. Ji, Y. Fu, J. Guo, Y. Pang, and Z. Zhang, "Stacked semantic-guided attention model for fine-grained zero-shot learning," *CoRR*, vol. abs/1805.08113, 2018.

[32] M. Sun, Y. Yuan, F. Zhou, and E. Ding, "Multi-attention multi-class constraint for fine-grained image recognition," in *Proc. European Conf. on Computer Vision*, 2018.

[33] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Advances on Neural Information Processing Systems*, 2015.

[34] T. Chen, W. Wu, Y. Gao, L. Dong, X. Luo, and L. Lin, "Fine-grained representation learning and recognition by exploiting hierarchical semantic embedding," in *Proc. ACM Int'l Conf. on Multimedia*, 2018.

[35] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2017.

[36] J. Wang, Y. song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2014.

[37] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[38] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int'l Journal of Uncertainty, Fuzziness and Knowledge-Based System*, vol. 6, no. 2, pp. 107–116, 1998.

[39] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[40] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolution network," *arXiv:1505.00853*, 2015.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. Int'l Conf. on Computer Vision*, 2015.

[42] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep learning with s-shaped rectified linear activation units," in *Proceedings AAAI Conference on Artificial Intelligence*, 2016, pp. 1737–1743.

[43] L. Trottier, P. Giguère, and B. Chaib-draa, "Parametric exponential linear unit for deep convolutional neural networks," in *Int'l Conf. on Machine Learning and Applications*, 2017.

[44] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, and R. F. Dumitru Erhan, Ian Goodfellow, "Intriguing properties of neural networks," in *Proc. Int'l Conf. on Learning Representations*, 2014.

[45] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for ReLU networks," in *Proc. Int'l Conf. on Machine Learning*, 2018.

[46] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," in *Advances on Neural Information Processing Systems*, 2018.

[47] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. J. Pappas, "Efficient and accurate estimation of lipschitz constants for deep neural networks," *arXiv:1906.04893*, 2019.

[48] J. Sokolić, R. Giryes, G. Sapiro, and M. Rodrigues, "Robust large margin deep neural networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4265–4280, 2017.

[49] C. Anil, J. Lucas, and R. B. Grosse, "Sorting out lipschitz function approximation," in *Proc. Int'l Conf. on Machine Learning*, 2019.

[50] H. Qian and M. N. Wegman, "L2-nonexpansive neural networks," in *Proc. Int'l Conf. on Learning Representations*, 2019.

[51] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," in *Advances on Neural Information Processing Systems*, 2018.

[52] G. J. Gordon, "Stable function approximation in dynamic programming," in *Proc. Int'l Conf. on Machine Learning*, 1995.

[53] H. Sagan, *Advanced Calculus*. Houghton Mifflin, 1974.

[54] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep. 1 (4), 7, 2009.

[55] C. Wah, S. Branson, P. Welinder, P. Perona, and Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[56] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *Proc. Int'l Workshop on 3D Representation and Recognition*, 2013.

[57] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *Proc. Workshop on Fine-Grained Visual Categorization (CVPRW)*, 2011.

[58] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," *arXiv:1306.5151*, 2013.

[59] Kaggle, "https://sites.google.com/view/fgvc5/competitions/fgvcx/ifood," (last accessed: July 25, 2019).