*electronics*

*Article*

# Feature Activation through First Power Linear Unit with Sign

**Boxi Duan [1], Yufei Yang [2] and Xianhua Dai [1,*]**

[1] School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China; duanbx3@mail2.sysu.edu.cn
[2] Faculty of Engineering and Information Technology, The University of Melbourne, Parkville 3010, Australia; yufeiy4@student.unimelb.edu.au
* Correspondence: issdxh@mail.sysu.edu.cn

**Abstract:** The activation function represents a crucial component in the design of a convolutional neural network (CNN). It enables the efficient extraction of multiple features from visual patterns, and introduces systemic non-linearity to data processing. This paper proposes a novel and insightful activation method termed FPLUS, which exploits mathematical power function with polar signs in form. It is enlightened by common inverse operations while endowed with an intuitive meaning of bionics. The formulation is derived theoretically under conditions of some prior knowledge and anticipative properties. Subsequently, its feasibility is verified through a series of experiments using typical benchmark datasets. The results indicate that our approach bears superior competitiveness among numerous activation functions, as well as compatible stability across many CNN architectures. Furthermore, we extend the function presented to a more generalized type called PFPLUS with two parameters that can be fixed or learnable, so as to augment its expressive capacity. The outcomes of identical tests serve to validate this improvement. Therefore, we believe the work in this paper holds a certain value in enriching the family of activation units.

**Keywords:** deep learning; convolutional neural network; activation function

**Citation:** Duan, B.; Yang, Y.; Dai, X. Feature Activation through First Power Linear Unit with Sign. *Electronics* **2022**, *11*, 1980. https://doi.org/10.3390/electronics11131980

Academic Editor: Maciej Ławryńczuk

Received: 30 May 2022
Accepted: 21 June 2022
Published: 24 June 2022

## 1. Introduction

As a core component embedded in CNN, the activation unit models the synapse in the way of non-linear mapping, transforming different inputs into valid outputs, respectively, before transmitting to the next node. This kind of simulation is partly motivated by neurobiology, such as the M–P model of McCulloch and Pitts [1], Hodgkin and Huxley's H–H Equation [2], Dayan and Abbott's firing rate curve [3], etc. As a consequence, a neural network is inherently non-linear owing to the contribution of hidden layers, in which the activation function is the source providing such a crucial property. Thus, an appropriate setup of non-linear activation should not be neglected for neural networks, and the option of this module usually varies according to the scenarios and requirements.

Over the years, a variety of activation functions have come into being successively, and some of them attain excellent performance. Sigmoid was applied in the early days but soon deprecated due to the severe gradient vanishing problem, which receives considerable alleviation from ReLU [4,5]. However, the risk of dying neurons renders ReLU vulnerable and hence many variants such as Leaky ReLU [6] attempt to fix this shortcoming. Subsequently, ELU [7] is proposed to address the bias shift effect while its limitation is relatively settled by supplementary strategies such as SELU [8]. Meanwhile, the implementation of combinative modality provides another way to merge different activation functions, e.g., Swish [9,10] and Mish [11]. As for works including Maxout [12] and ACON [13], they attempt to unify several approaches into the same paradigm. Moreover, a few methods approximate the sophisticated function through piecewise linear analog for reduction of complexity, such as hard-Swish [14].

The information mentioned above presents a rough outline of the remarkable progress regarding activation units achieved in recent decades. Given that most of them are based on the transform, recombination, or amendment of linear and exponential models, sometimes logarithmic, but rarely based on mathematical power form, a curious question arises: What if activation is conducted via a certain power function rather than the conventional familiar ones? Additionally, few of the methods take bionic characteristics into account and lack neurobiological implications, providing a weak approach to symbolizing the mechanisms underlying a neural synapse.

Drawing inspiration from the M–P model [1] and perceptron theory [15], as well as binary connect algorithm [16] and symmetric power activation functions [17], this paper proposes a new approach of activation which considers sign function as the switch factor and applies distinct first power functions to positive and negative inputs, respectively, thus to some degree a bionic meaning is imparted to it. We name this approach first power linear unit with sign (FPLUS). In terms of what has been discussed in [18] by Bengio et al., we theoretically derive the subtle representation in the form of a power function under some designated premises, which are in accordance with the attributes that a reasonable activation unit requires. Furthermore, following in the spirit of PReLU [19], PELU [20], and other parallel works, we generalize the expression to an adjustable form with two parameters which can be either manually set or learned from the training process, regulating both amplitude and flexure of the function shape. This helps with optimization when fine-tuning. Naturally, we call this approach parametric version (PFPLUS).

Of course, we conduct a sequence of experiments to verify the feasibility and robustness of the proposed FPLUS and PFPLUS. We then compare them with popular activation units in classification tasks to evaluate the performance of our methods, which demonstrates that the approach we come up with is capable of achieving comparable and steady results. Meanwhile, in order to figure out a preferable dynamic scope of fluctuation intervals, we explore the setting influence of two controllable parameters by various initialization or assignment, and a preliminary optimization tendency is obtained.

The main contributions of our work can be briefly summarized as follows: (1) We theoretically propose a new activation formulation FPLUS which contains bionic allusion, and extend it to a generalized form PFPLUS with tunable parameters; (2) We conduct experiments diversely to validate the performance of our methods, and the results demonstrate their advantages of competence; (3) The effect of parameter variation is explored through tests that analyse the dynamic property of PFPLUS.

The remainder of this paper is divided into five sections as follows. Section 2 reviews some representative activation functions in recent years. Section 3 describes our method's derivation in detail, while its characteristics and attributes are analyzed in Section 4. Section 5 includes experimental results and an objective discussion. Finally, a conclusive summary is presented in Section 6.

## 2. Related Works

As is widely known, there exists a number of activation functions, some of which obtain significant breakthroughs. We illustrate part of them and our work FPLUS with corresponding derivatives in Figure 1, and also list formulae of the prominent ones below.

The conventional activation function sigmoid was utilized in earlier stages, but Xavier demonstrated its tendency to result in the gradient vanishing phenomenon [21] for it is bounded in the positive domain, a tendency shared by the tanh function. However, LeCun's article [22] pointed out tanh leads to faster convergence for the network than sigmoid since the former one has zero-mean outputs, which are more in accord with the condition of natural gradient expounded in [23], and thus iteration can be reduced. Nevertheless, both produce saturated outputs when given large positive inputs, making the parameter's updating probably stagnate.

$$\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

Subsequently, ReLU [4,5] became widely employed as the activation function in CNN. Not only does it have easy shape and concise expression, but also alleviates the problem of gradient diffusion or explosion through identity mapping in the positive domain. Still, it forces the output of negative values to be zero, and thereby introduces the distribution of sparsity to the network. Although this sort of setting seems plausible referring to the neurobiological literature [24,25], it also brings about the disadvantage called dying neurons, which refer to the units that are not activated since their gradients reach zero, and in which case the parameters cannot be updated. Therefore to overcome this drawback, Leaky ReLU [6] assigns a fixed slope of non-zero constant to the negative quadrant. However, the value specified is extremely sensitive, so PReLU [19] makes it trainable. However, this approach inclines to cause overfitting. Then, RReLU [26] changes it to be stochastically sampled from a probability distribution, resembling random noise thus generating regularization. On the other hand, instead of focusing on the activation function itself, some methods seek assistance from auxiliary features distilled additionally, such as antiphase information for CReLU [27], and spatial context for FReLU [28], DY-ReLU [29]. However, this means more overhead on pretreatment.

$$\mathrm{ReLU}(x) = \begin{cases} x, & \text{if } x \geqslant 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{3}$$

$$\mathrm{LReLU}(x) = \begin{cases} x, & \text{if } x \geqslant 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \quad (\alpha > 0) \tag{4}$$

In contrast with ReLU, another classic alternative is ELU [7], which replaces the output of the negative domain with an exponential function. In this way, ELU avoids the predicament of dead activation from zero derivative, while mitigating the effect of bias shift by pushing the mean of outputs closer to zero. Furthermore, it appears robust to noise due to the existence of saturation in the negative quadrant. Later, CELU [30] remedied its flaw of not maintaining differentiable continuously in some cases, while PELU [20] gives it more flexibility with learnable parameters. In addition, SELU [8] is proved to possess a self-normalizing attribute with respect to the fixed point which is supposed after allocating scale factors, and FELU [31] provides a peculiar perspective to realize accelerating the exponential activation by equivalent transformation.

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x \geqslant 0 \\ \alpha(e^x - 1), & \text{if } x < 0 \end{cases} \quad (\alpha > 0) \tag{5}$$

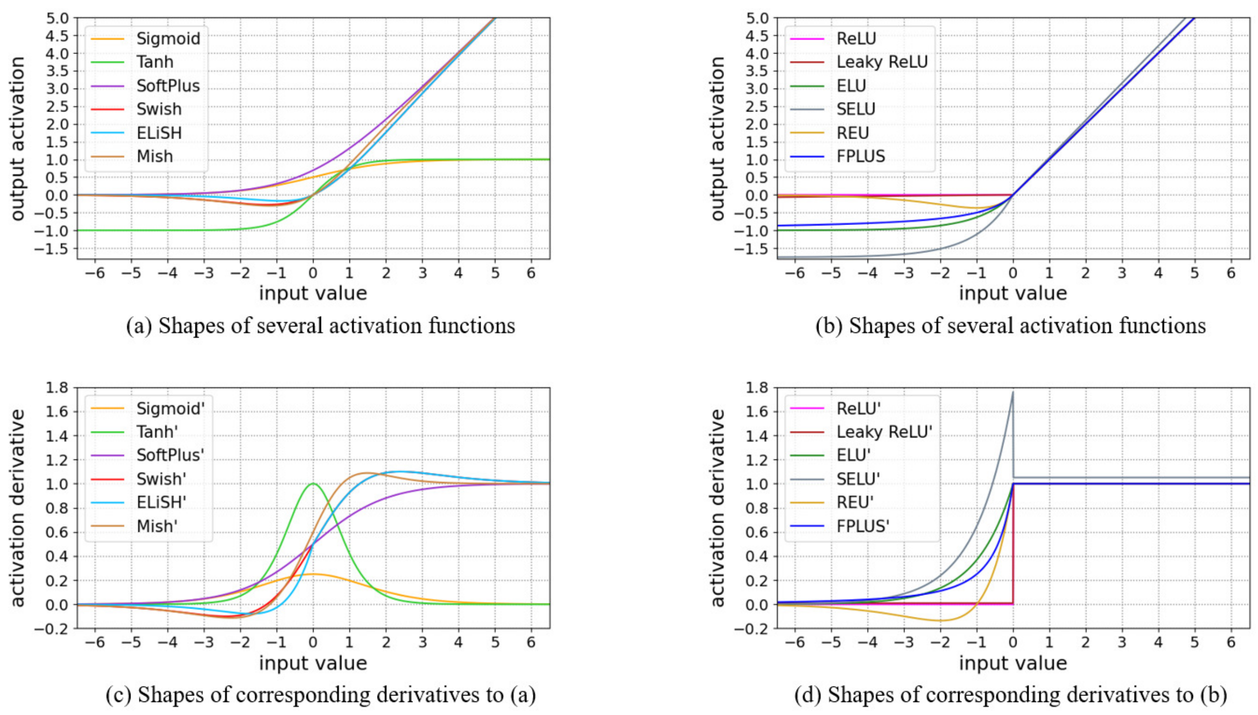
$$\mathrm{SELU}(x) = \lambda \cdot \begin{cases} x, & \text{if } x \geqslant 0 \\ \alpha(e^x - 1), & \text{if } x < 0 \end{cases} (\lambda > 1, \alpha > 0) \tag{6}$$

Moreover, the nested or composite mode can present an innovative scheme to generate new activation functions. SiLU [9] for instance, also known as Swish [10], exactly multiplies the input by sigmoidal weight, and ELiSH [32] makes a slight alteration on this basis. GELU [33] adopts a similar operation, regarding the cumulative distribution function of Gaussian distribution as the weight, while Mish [11] combines tanh and SoftPlus [34] before weighting the input. As for REU [35], it directly takes an exponential function as the multiplier resembling a hybrid of ReLU and ELU, whereas RSigELU [36] involves extra sigmoid at the cost of further complicating the expression.

$$\mathrm{Swish}(x) = x \cdot sigmoid(\beta x) \quad (\beta > 0) \tag{7}$$

$$\mathrm{GELU}(x) = x \cdot \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} \mathrm{d}t \tag{8}$$

$$\text{Mish}(x) = x \cdot tanh[ln(1 + e^x)] \tag{9}$$



(a) Shapes of several activation functions

(b) Shapes of several activation functions

(c) Shapes of corresponding derivatives to (a)

(d) Shapes of corresponding derivatives to (b)

**Figure 1.** Shapes of typical activation functions and their corresponding derivatives.

It is likewise noticed that the piecewise linear approximation of a curve is usually requested in mobile devices restricted by computing resources. In response, hard-series fitting was put forward, such as hard-sigmoid [16], hard-Swish [14]. Moreover, some other works such as Maxout [12] and ACON [13] managed to unify multiple types of activation methods into one single paradigm so that each of them becomes a special case to the whole family. However, this means many more parameters have to be learned to fulfill such an adaptive conversion, and hence preprocessing efficiency might not be desired.

From what has been mentioned above, we observe that progress and advancement of activation methods have proceeded for years, due to different application scenarios or performance requirements. Furthermore, it is not only a unit of transferring in artificial neural networks but also a component of gating when applied in practical microcircuits such as [37,38].

## 3. Proposed Method

In this section, we first explain the motivation of our approach more specifically, and the deduction process is described concretely in the next paragraph.

### *3.1. Source of Motivation*

There is no doubt neurobiology offers probative bionic fundamentals to the design of an imitated neural system. For example, CNN represents a tremendous milestone that has greatly promoted the advancement of computer vision, and can be traced back to Hubel and Wiesel's work [39], whose discovery provided inspiration for the conception of CNN framework. Hence, bionics-oriented practice might provide a way to find clues or hints of creation [40], to which lots of documents attach great importance.

The M–P model [1] follows the principle of a biological neuron, where there exists an excitatory or inhibitory state depending on the polarity of input type, and a threshold determines whether the neuron is activated or not. Based on this foundation, primitive perceptron [15] utilizes sign function to produce a binary output of type 0 or 1, representing different activation states, so as to abstract out effective and notable information from features generated by lower-levels and pass it on to the deep, although this is a fairly rudimentary method. Another recent research work called binary connect [16] inherits the idea alike, not with regard to coping with output but rather binarizing the weight via 1 and $-1$, intended at decreasing resource consumption on mobile facilities. In addition, this setup could be reckoned as a similar strategy of regularization.

On the other hand, advancement has witnessed multiple activation methods come into being and evolve consecutively, and plenty of mathematical functions have been employed to act as transfer units in neural networks. Still, in comparison to piecewise linear, exponential, and logarithmic formulae, power type functions seem to attract little attention, and are thus rarely seen in the field. Under the impact of these famous pioneers, only a few works focus on this subject, e.g., [17] changes the positive representation of ReLU by using a couple of power function and its inverse, which are symmetric to the linear part of ReLU. However, it lacks further validation on more datasets and networks.

As a consequence, there are many potentialities we can explore and exploit in power type activation theory, and our work probes into how to construct a refined activation function of this kind with simplicity as well as effectiveness, and pays more attention to the connection with bionic meanings.

### *3.2. Derivation Process*

Clearly, the positive part $y = x$ of ReLU is identical mapping, however, it can also be regarded as $y = x^1$, i.e., the output is one power over the input, and this is for positive situation, which reflects the excitatory state of a neuron. Associating with implicit bionic allusion, we ponder the question: Can we represent the inhibitory state in a contrary manner based on a certain power type function, corresponding to the negative part of the same united expression through inverse operation?

As the reverse procedure of positive first power, negative first power is doubtlessly supposed to be allocated to the opposite position in the formula, which will involve 1 and $-1$ two constant exponents simultaneously in this case, so a sign function is taken into consideration to play the role of switching between them. Meanwhile, the first-order power is relatively common and simpler than the higher-order ones, consuming less computational cost while enabling itself to handle bilateral inputs via two distinct branches.

For more generality, we firstly bring in some coefficients $\alpha$, $\beta$, $\omega$, $\theta$ which are finite real numbers and unequal to zero. Moreover, we give the definition of our formula $f(x) = \alpha[\omega x + \beta]^{sgn(x)} + \theta$, where $\mathrm{sgn}(x)$ is the sign function shown as Equation (10), while a piecewise form of our expression is given in Equation (11).

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geqslant 0 \\ -1, & \text{if } x < 0 \end{cases} \tag{10}$$

$$f(x) = \begin{cases} \alpha\omega x + \alpha\beta + \theta, & \text{if } x \geqslant 0 \\ \dfrac{\alpha}{\omega x + \beta} + \theta, & \text{if } x < 0 \end{cases} \tag{11}$$

With reference to the usual characteristics and prior knowledge of those typical activation methods, we design our function $f(x)$ to have such expected properties enumerated as follows:

(I). Defined on the set of real numbers:

$$x|_{f(x)=\infty} > 0 \tag{12}$$

(II). Through the origin of coordinate system:

$$f(0) = 0 \tag{13}$$

(III). Continuous at the demarcation point:

$$\lim_{x\to 0^+} f(x) = \lim_{x\to 0^-} f(x) = f(0) \tag{14}$$

(IV). Differentiable at the demarcation point:

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x=0^+} = \left.\frac{\partial f(x)}{\partial x}\right|_{x=0^-} \tag{15}$$

(V). Monotone increasing:

$$\forall x \in \mathbb{R}\,,\ \frac{\partial f(x)}{\partial x} \geqslant 0 \tag{16}$$

(VI). Convex down:

$$\forall x \in \mathbb{R}\,,\ \frac{\partial^2 f(x)}{\partial x^2} \geqslant 0 \tag{17}$$

Condition (I) requires any value on the set of real numbers ought to be valid as input, but we see there exists an issue when the negative part of $f(x)$ expression achieves infinity since the denominator $\omega x + \beta$ of the fraction is not allowed to be zero. As a consequence, considering the piecewise attribute, we can just confine the pole point to the opposite interval and this latent defect can be indefinitely avoided. Assuming that $\exists x_i \in (-\infty, 0)$, which makes for

$$\omega x_i + \beta = 0 \implies x_i = -\frac{\beta}{\omega}. \tag{18}$$

To cause a paradox, we need

$$x_i > 0 \implies \frac{\beta}{\omega} < 0 \tag{19}$$

With regard to Condition (II), it is clear that point $(0, 0)$ of the orthogonal plane coordinate system must be a solution to our activation function, and thus we have

$$f(0) = 0 \implies \alpha\beta + \theta = 0 \tag{20}$$

Condition (III) calls for continuity, so limits on both sides of the demarcation point must exist, while they equal each other and the value of that point. Hence, we list

$$\left.\begin{aligned} \lim_{x\to 0^+} f(x) &= \alpha\beta + \theta \\ \lim_{x\to 0^-} f(x) &= \frac{\alpha}{\beta} + \theta \end{aligned}\right\} \implies \alpha\beta + \theta = \frac{\alpha}{\beta} + \theta \tag{21}$$

Because $\alpha, \beta \neq 0$ stated above, it comes out

$$\beta = \pm 1 \tag{22}$$

From Formula (20) it is known that $f(0) = 0$ , and solving these simultaneous equations we can obtain

$$\alpha = \begin{cases} -\theta, & \text{if } \beta = 1 \\ \theta, & \text{if } \beta = -1 \end{cases} \tag{23}$$

As for Condition (IV), it demands taking the derivatives on both sides of the demarcation point and they should be identical, too. Thereby, we get

$$\alpha\omega|_{x=0^+} = \left.\frac{-\alpha\omega}{(\omega x + \beta)^2}\right|_{x=0^-} \implies \alpha\omega = \frac{-\alpha\omega}{\beta^2} \tag{24}$$

Since $\alpha, \beta, \omega \neq 0$ as previously mentioned, it turns out

$$\beta = i \tag{25}$$

Now that an outcome of imaginary number $i$ occurs, which disobeys the hypothesis that $\beta$ is a real number, the precondition thus needs to be amended.

Observing the graphic curves of a system for the first power functions, shown in Figure 2 for example, it can be recognized that line $y = \omega x$ is capable of being tangent to curve $y = -\omega x^{-1}$ somehow, but this will never happen to curve $y = \omega x^{-1}$ because they are always of intersection, and vice versa for line $y = -\omega x$. This fact indicates that the former combination has the potentiality to construct a smooth and derivable function if splicing appropriate portions severally, whereas the latter one does not. The shadow region in the figure implies a possible scope for line $y = \omega x$ to achieve tangency with curve $y = -\omega x^{-1}$, while we can observe that it crosses curve $y = \omega x^{-1}$ under any circumstance.

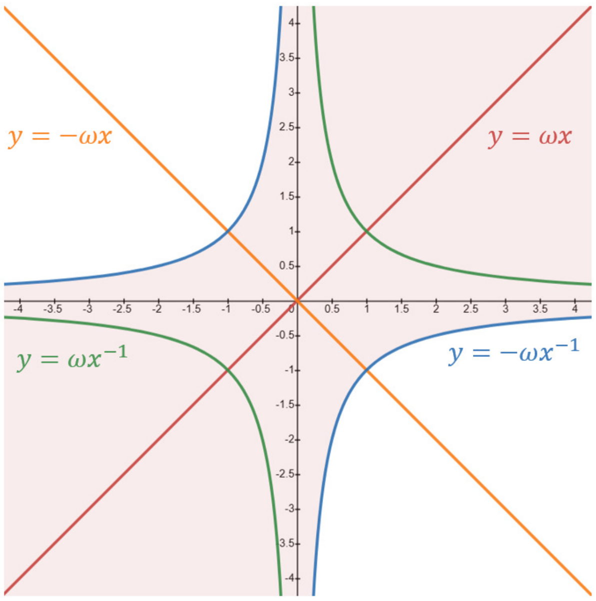


**Figure 2.** Graphic curves of power functions in a pair of opposite signs with 1 and $-1$ being exponents. Shadow region signifies a general orientation for $y = \omega x$ to reach tangency with $y = -\omega x^{-1}$.

Based on the rules mentioned above, we are aware that for reciprocal types such as $y = \omega_1 x$ and $y = \omega_2 x^{-1}$, only if the coefficients $\omega_1$ and $\omega_2$ have contrary signs, can they be differentiable at the demarcation point in piecewise combination when carrying out fundamental functional transformation, such as flip, scaling, translation, etc. Therefore, we know

$$\omega_1 \neq \omega_2 \tag{26}$$

and the original definition of our function has to be adjusted to

$$f(x) = \begin{cases} \alpha\omega_1 x + \alpha\beta + \theta, & \text{if } x \geqslant 0 \\ \dfrac{\alpha}{\omega_2 x + \beta} + \theta, & \text{if } x < 0 \end{cases} \tag{27}$$

where $\alpha, \beta, \omega_1, \omega_2, \theta \neq 0$ and all are finite real numbers. In this way, we evaluate the differentiability of the boundary point again

$$\alpha\omega_1\big|_{x=0^+} = \frac{-\alpha\omega_2}{(\omega_2 x + \beta)^2}\bigg|_{x=0^-} \implies \omega_1 = \frac{-\omega_2}{\beta^2} \tag{28}$$

Substituting Formula (22) into the equation, we are able to calculate the qualitative and quantitative relation

$$\omega_1 = -\omega_2 \tag{29}$$

In the meantime, inequality (19) is hence changed into

$$\frac{\beta}{\omega_2} < 0 \tag{30}$$

which can be inferred that

$$\omega_2 \begin{cases} < 0, & \text{if } \beta = 1 \\ > 0, & \text{if } \beta = -1 \end{cases} \tag{31}$$

As far as Condition (V) is concerned, it involves property of first-order derivative, and owing to the premise that no coeffiecient can be zero, we accordingly expect

$$(i)\ \forall x \geqslant 0,\ \frac{\partial f(x)}{\partial x} = \alpha\omega_1 > 0 \tag{32}$$

$$(ii)\ \forall x < 0,\ \frac{\partial f(x)}{\partial x} = \frac{-\alpha\omega_2}{(\omega_2 x + \beta)^2} > 0 \implies \alpha\omega_2 < 0 \tag{33}$$

Consulting Formula (29), it is clear the above two restrictions are equivalent and meeting one of them, e.g., inequality (32) is sufficient.

Finally, Condition (VI), which asks for the function's being convex down and that refers to second-order derivative, thus we hold

$$(i)\forall x \geqslant 0,\ \frac{\partial^2 f(x)}{\partial x^2} = 0 \geqslant 0 \tag{34}$$

$$(ii)\forall x < 0,\ \frac{\partial^2 f(x)}{\partial x^2} = \frac{2\alpha\omega_2^2}{(\omega_2 x + \beta)^3} > 0 \implies \frac{\alpha}{\omega_2 x + \beta} > 0 \tag{35}$$

Subordinate Condition (i) is apparently self-evident, while for (ii) it can be found out from the formulae (29), (31) and (32) that

- When $\beta = 1$, then $\omega_2 < 0,\ \alpha > 0 \implies \forall x \in (-\infty, 0)$ subordinate condition (ii) is tenable.
- When $\beta = -1$, then $\omega_2 > 0,\ \alpha < 0 \implies \forall x \in (-\infty, 0)$ subordinate condition (ii) is tenable.

As a result, this means condition (VI) is inevitable on the basis of previous constraints.

To summarize, any group of real numbers $\alpha, \beta, \omega_1, \omega_2, \theta$ which are not equal to zero and satisfy formulae (22), (23), (29), (31) and (32), can be valid coefficients for our function, and we summarize them as two cases.

- $\beta = 1,\ \omega_2 < 0,\ \omega_1 = -\omega_2 > 0,\ \alpha > 0,\ \theta = -\alpha$
- $\beta = -1,\ \omega_2 > 0,\ \omega_1 = -\omega_2 < 0,\ \alpha < 0,\ \theta = \alpha$

Having case 1 and case 2 substituted into the latest definition of $f(x)$ separately and simplifying them, hence there comes

$$\text{case 1: if } \beta = 1,\ \text{then } f(x) = \alpha\omega_1 \cdot \begin{cases} x, & \text{if } x \geqslant 0 \\ \dfrac{x}{1 - \omega_1 x}, & \text{if } x < 0 \end{cases} \tag{36}$$

in which $\alpha > 0,\ \omega_1 > 0$;

$$\text{case 2: if } \beta = -1,\ \text{then } f(x) = \alpha\omega_1 \cdot \begin{cases} x, & \text{if } x \geqslant 0 \\ \dfrac{x}{1 + \omega_1 x}, & \text{if } x < 0 \end{cases} \tag{37}$$

in which $\alpha < 0$, $\omega_1 < 0$.

Observing the two situations acquired above, it is not difficult to notice that $\alpha\omega_1 > 0$ always stands, and the coefficient of x laying in the denominator of $f(x)$'s piecewise representation within $(-\infty, 0)$, is always true to be negative. Therefore, we can unify the two cases that $\beta = \pm 1$ for the following new form:

$$f(x) = \lambda \cdot \begin{cases} x, & \text{if } x \geq 0 \\ \dfrac{x}{1 - \mu x}, & \text{if } x < 0 \end{cases} \quad (\lambda > 0, \mu > 0) \tag{38}$$

Here, $\lambda$ is termed as amplitude factor, which controls the degree of saturation for the negative zone and the magnitude of slope for positive zone, while $\mu$ is called scale factor, which regulates the speed of attenuation for the negative domain only. The derivative of our formulation is given below:

$$f'(x) = \lambda \cdot \begin{cases} 1, & \text{if } x \geqslant 0 \\ \dfrac{1}{(1 - \mu x)^2}, & \text{if } x < 0 \end{cases} \quad (\lambda > 0, \mu > 0) \tag{39}$$

To date, it can be seen that Equation (38) represents probably the core of our theory, which resembles SELU [8] in conception when taken as a counterpart since they both have linear and decayed areas with two controlled parameters that facilitate their shifting in shapes.

## 4. Analysis of the Method

After demonstrating the deduction process of our means and giving the ultimate definition in segmented form, in this section we further delve into more traits and their properties.

### 4.1. Characteristics and Attributes

In order to merge piecewise activation into one united paradigm for the response, we introduce the Heaviside step function known as Equation (40), i.e., unit step function, enabling it to classify the polarity of the input's sign, and react similarly to a switch in neural circuit, subsequently transferring the signal with instant gain or gradual suppression.

$$H(x) = \begin{cases} 1, & \text{if } x \geqslant 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{40}$$

As long as conversion is applied to Equation (38), we hereby formally set forth parametric first power linear unit with sign (PFPLUS):

$$\text{PFPLUS}(x) = \lambda x \cdot (1 - \mu x)^{H(x) - 1} \tag{41}$$

where $H(x)$ is the Heaviside step function, and $\lambda, \mu > 0$.

Undisputedly, with parameters $\lambda$ and $\mu$ changing, our proposed function will present different shapes in the graph. As shown in Figure 3, it is similar to ReLU when $\lambda = 0.2$ and $\mu = 10$, while approximating linear mapping as $\lambda = 5$ and $\mu = 0.1$. Of course, they can be either learned from training procedure or fixed by initial setting, depending on the intended purpose.

With regard to the implementation of trainable PFPLUS, we choose established gradient descent as the iterative approach to updating and learning these two coefficients. Further, we adopt strategies such as PReLU [19] to avoid overfitting as much as possible. For every activation layer of PFPLUS, parameters $\lambda$ and $\mu$ are completely identical across all channels, in other words, they are channel-shared or channel-wise. As a result for the whole network, the increment of parameters is merely negligible to the total quantity of weights.

Further, Ramachandran et al. have conducted a lot of work searching for good activation units based on ReLU [41], including unary functions and binary functions. They discovered that those which perform well tend to be concise, often consisting of no more than two elements, and usually comprise the original input, i.e., in the form of $f[x, g(x)]$. The study also perceived that most of them are smooth and equipped with linear regions. However, they missed inquiring into power ones with negative exponents when researching on unary types, whereas our work provides some supplementation in a sense.

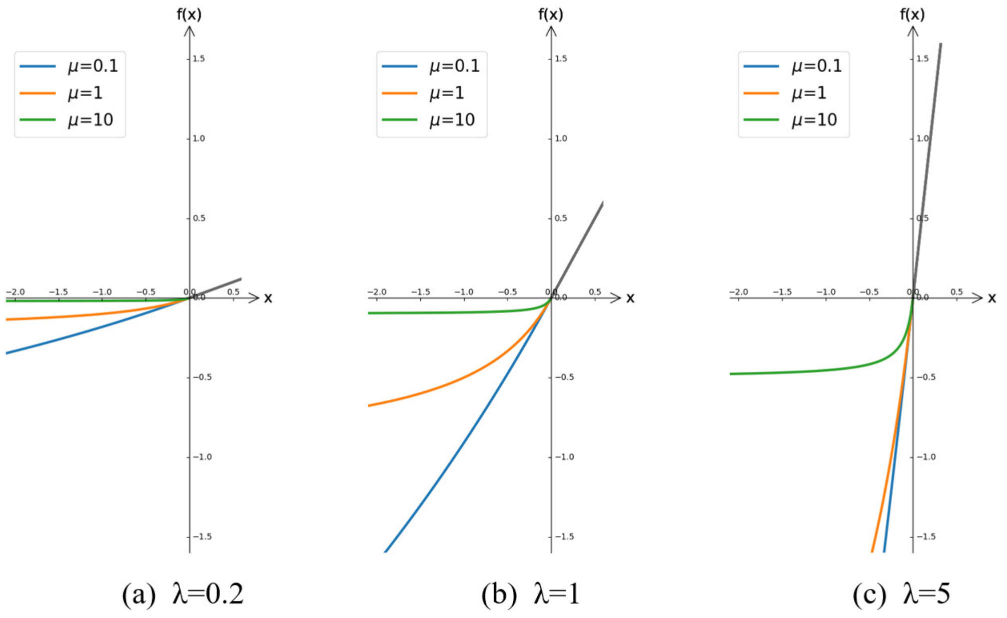


**Figure 3.** Different shapes of PFPLUS when $\lambda$ and $\mu$ vary.

Particularly, as the parameters $\lambda$ and $\mu$ are both equal to one, the function becomes a special form such as Equation (42), which is the simplest pattern of its whole family. In addition, considering the views discussed by Bengio et al.in [18], we theoretically reckon the advantages for our way as follows:

1. For positive zones, identity mapping is retained for the sake of avoiding the gradient vanishing problem.
2. For negative zones, as the negative input deepens, the outcome will show a tendency to gradually reach saturation, which offers our method robustness to the noise.
3. Entire outputs' mean of the unit is close to zero, since the result yielded for negative input is not directly zero, but exists a relative minus response to neutralize the holistic activation, so that bias shift effect can be reduced.
4. When bearing the corresponding formulation of the negative part processed in Taylor expansion, seen as Equation (43) , the operation carried out in the negative domain is equivalent to dissociating each order component of the input signal received, and thus more abundant features might be attained up to a point.
5. From an overall perspective, the shape of the function is unilateral inhibitive, and this kind of one-sided form could facilitate the introduction of sparsity to output nodes, making them similar to logical neurons.

$$f(x) = \begin{cases} x, & \text{if } x \geqslant 0 \\ \dfrac{x}{1-x}, & \text{if } x < 0 \end{cases} \tag{42}$$

$$\frac{x}{1-x} = x + x^2 + x^3 + \cdots + x^{n-1} + x^n + O(x^n) \tag{43}$$

### *4.2. Implications and Examples*

Moreover, our method can be further associated with a bionic meaning, as long as we implant sign function listed in Equation (10) to play the role of neuron switch, and we therefore officially put forward the first power linear unit with sign (FPLUS), shown as Equation (44).

$$\mathrm{FPLUS}(x) = [\mathrm{sgn}(x) \cdot x + 1]^{\mathrm{sgn}(x)} - 1 \tag{44}$$

For the two sign functions applied in this definition, the one outside bracket is an exponent for the whole power, and can be regarded as a discriminant factor to the input's polarity, while the function inside the bracket is a coefficient, which can be seen as a weight to the input, operating in a bipolar manner somewhat similar to that mentioned by [16] .

Figure 4 shows a signal flow diagram when an input passes through the FPLUS activation unit. In the figure, breakpoints signify the process of switch selection according to the polarity of the input signal. Rectangle components denote weight coefficients while the circle components denote adding biases. Moreover, diamond components symbolize the conversion operations including first power and negative first power.

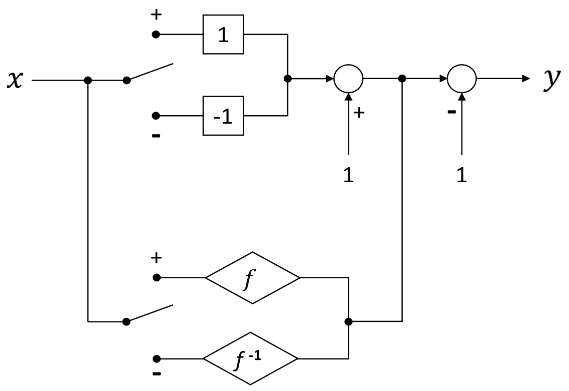


**Figure 4.** Signal flow diagram of the input passing through the FPLUS activation unit. There are two switches in the diagram, in which the upper one chooses the branch of weight, and the lower one decides to apply negative first power or not.

Therefore, once our method is employed in the neural network, it can be regarded that every neuron in the convolutional layer contains an excitation part of FPLUS, taking the place of the widely used ReLU. Figure 5 illustrates this concept by way of a system chart.

$$Y = f(WX + b) \tag{45}$$

As widely known, Formula (45) interprets the relationship between input $X$ and output $Y$ for a neural node, and there is no doubt our method meets the situation as well. For example, assuming that the weight tensor of a filter is $W = [2, -1, -2,\ 1]$ with bias $b = 0.5$, and the normalized input vector transmitted to a neuron is $X = [-0.8, -0.6,\ 0.4, -0.2]^{\mathrm{T}}$, we can easily reach that the outcome produced by the convolutional kernel is $-1.5$. However, diverse excitation approaches lead to different final results. The output comes to $-0.6$ after being activated by FPLUS, while the output comes to 0 if we use ReLU. Now we see FPLUS activation retains part of the negative values to some extent, but ReLU completely erases them all, and the latter one is bound to miss a lot of latent information when extracting features, which is a disadvantage during forward propagation.

In conclusion, as expounded upon above, we demonstrate our method is in possession of rigorous rationality and adequate novelty. At the same time, there are still numerous other methods we can develop to comprehend its ample connotation.

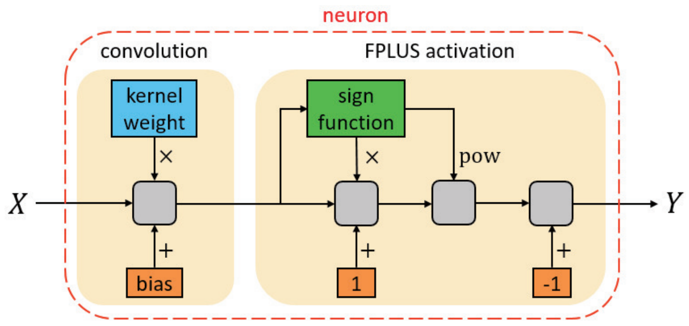


**Figure 5.** A neuron in the convolutional layer with FPLUS unit embedded as the activation gate. $X$ indicates the input, which may be normalized before, and $Y$ denotes convolutional output after indispensable excitation.

## 5. Experiments and Discussion

In this section, we conduct a series of experiments to probe into feasibility, capability, and universality of our method. Moreover, we validate those inferences we discussed before by classification task, covering controlled trials and ablation study. In order to explore its influence on the performance of extracting features in neural networks, we test it on different CNN architectures across many representative datasets, including MNIST & Fashion-MNIST, Kaggle's Dogs Vs. Cats, Intel Image Classification, CIFAR-10 and CIFAR-100, and ImageNet-ILSVRC2012.

This section is organized by different experiments with each dataset listed above, thus it is divided into 5 main segments.

### *5.1. Influence of Two Alterable Factors $\lambda$ and $\mu$*

MNIST is an old but classic dataset created by LeCun et al. in the 1990s, which is comprised of handwritten digits 0∼9 with a training set of 60K examples and a test set of 10K examples. These morphologically diverse digits have been self-normalized and centered in grayscale images, each with 28 × 28 pixels. Similarly, Fashion-MNIST is a popular replacement emerging in recent years, and shares the same image size, sample quantity, structure of training and testing splits much the same as the original MNIST. As the name implies, this dataset is a version of fashion and garments, containing labels from 10 classes.

Seeing as there are two mutable factors in our parametric function, we plan to preliminarily explore their effects on the performance of activation as the values vary. To facilitate direct observation of dynamic change and prominent differences, we choose LeNet-5 [42] to learn from the data due to its simplicity and origin, and its architecture is shown as Figure 6.

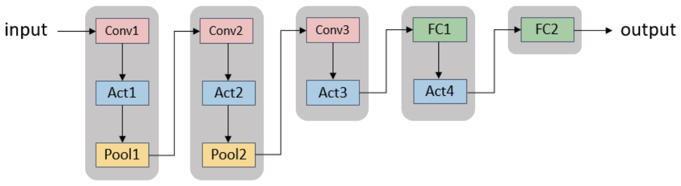


**Figure 6.** Architecture of LeNet-5. There are four activation units lying in four layers correspondingly, illustrated by blue boxes.

Arriving first is an experiment with fixed configurations of $\lambda$ and $\mu$, set with four orders of magnitude for each, which are 0.01, 0.1, 1, and 10. We train the model with MNIST and Fashion-MNIST for 5 epochs separately but follow the same training setups. The batch size is 64 and the learning rate is 0.001, with cross-entropy loss being the loss function and Adam being the optimizer.

The results of training loss and test accuracy on both datasets are listed in Table 1, grouped by different settings of $\lambda$ and $\mu$. Items in blue color are the best outcomes of loss and accuracy relevant to the corresponding configuration of $\lambda$ and $\mu$, which shows that our method achieves superior results when the two parameters are equal to 1, obtaining the lowest loss and highest accuracy when compared with any other combinations for both MNIST and Fashion-MNIST. In addition, factor $\lambda$ seems to be more sensitive than $\mu$ and exhibits a more intense response as the value changes. For each dataset, only considering $\lambda$ with the same $\mu$, results will get better when $\lambda$ grows from 0.01, especially by a large margin for the interval of 0.01 to 0.1, and reaches the optimum around 1, then begins to fall back slightly. A similar situation happens to $\mu$ basically, but not as conspicuous as the former one. Thus, this study provides us with a tentative scope of proper $\lambda$ and $\mu$ for favorable activation performance, and such regularity implies that an applicable assignment of $\lambda$ and $\mu$ should be neither too large nor small.

**Table 1.** Training loss and test accuracy on MNIST & Fashion-MNIST.

| Dataset | Factor | Loss | | | | Accuracy | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ \ $\mu$ | **0.01** | **0.1** | **1** | **10** | **0.01** | **0.1** | **1** | **10** |
| MNIST | 0.01 | 1.573 | 1.485 | 1.441 | 1.534 | 39.93% | 43.85% | 44.89% | 41.20% |
| | 0.1 | 0.147 | 0.114 | 0.139 | 0.158 | 96.21% | 97.01% | 96.63% | 96.05% |
| | 1 | 0.054 | 0.046 | 0.027 | 0.028 | 98.40% | 98.31% | 98.97% | 98.92% |
| | 10 | 0.567 | 0.430 | 0.111 | 0.079 | 96.21% | 96.47% | 97.11% | 97.45% |
| Fashion-MNIST | 0.01 | 1.091 | 1.072 | 1.065 | 1.343 | 52.88% | 57.25% | 59.45% | 52.30% |
| | 0.1 | 0.630 | 0.568 | 0.534 | 0.579 | 76.49% | 78.84% | 79.87% | 77.97% |
| | 1 | 0.309 | 0.307 | 0.253 | 0.267 | 87.75% | 88.18% | 89.62% | 88.98% |
| | 10 | 0.496 | 0.636 | 0.371 | 0.341 | 83.49% | 84.99% | 86.53% | 86.71% |

For each dataset, the best results of loss and accuracy are denoted in blue color.

Based on this, we narrow the value range and continue to investigate its preference when $\lambda$ and $\mu$ are capable of being learned dynamically from data. We keep other training setups alike but increase the epoch to 10, and convert our activation function to learnable mode. Because there are only four activation layers in the official structure of LeNet-5 as shown in Figure 6, we can easily record the variation tendency for each parameter after learning. More elaborately, we select five grades to start initialization for both $\lambda$ and $\mu$, which are 0.2, 0.5, 1, 2, and 5. Finally, according to the updated results of two parameters after learned by 10 epochs, Figure 7 shows the variation for each activation layer legibly.

It can be found that there seems to be some kind of gravity in the vicinity of 1 that attracts $\lambda$ to update along, and those which initialized from both ends, such as 0.2 and 5, tend to evolve towards the central zone. In contrast, $\mu$ appears not much interested in similar renewal but keeps itself fluctuating around the initial value of every layer.

From another perspective, this sort of discrepancy is sensible. Looking back on our PFPLUS formula in Equation (38), $\lambda$ plays the role of amplitude factor affecting both positive and negative domains. If it is too large, a gradient exploding problem will occur and the capability of resisting noise will be damaged. On the contrary, if it is quite small, the trouble of gradient vanishing will impede the learning process. As for $\mu$, it only manipulates the speed of attenuation for negative domain, and the activation level will decay to saturation sooner or later, which is just a matter of iteration time. So, we know $\lambda$ has more impact on the quality of our function compared to $\mu$, but that does not mean the latter one is neglectable since it likewise determines the optimal state of activation performance.

Therefore, we see the phenomenon that $\lambda$ inclines to converge on 1 nearby after training, albeit its degree differs in the light of layer, whereas $\mu$ shows no apparent iteration tendency but oscillates around where it is initiated.

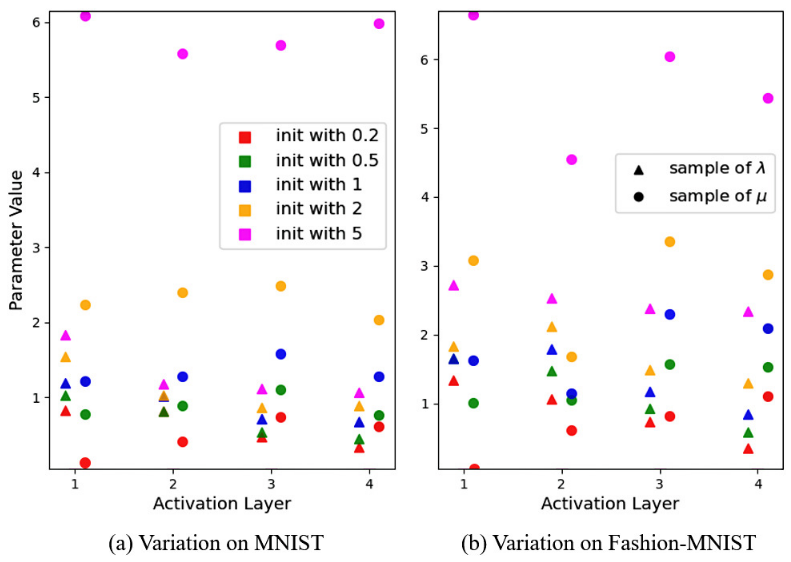


**Figure 7.** Parameter variation from different initialization on MNIST and Fashion-MNIST after learning. Triangular points are samples of $\lambda$ and circular ones are samples of $\mu$. Initial values are distinguished by different colors.

### *5.2. Intuitive Features of Activation Effect*

Kaggle's Dogs Vs. Cats is an adorable dataset released by Kaggle that contains 25K images in RGB format with arbitrary sizes, and specimens of dogs and cats account for each half. We sample 10% from every category randomly, which has 1250 examples, respectively, and then distribute them together to the validation set, so the remaining part automatically constitutes the training set.

Furthermore, we need a deeper network to verify the potentiality of our approach, so we choose the legendary masterpiece ResNet [43] with construction of 18 layers as the framework. By reason of matching the input size to that required by the network, we adjust every image to 224 $\times$ 224 pixels before being fed in. The training epoch is 10 in this experiment with batch size 32, and the learning rate is 0.001 but decays half every two epochs. Customarily, cross-entropy loss is the cost function and Adam is designated as the optimizer.

For typical comparison, we couple the network with our unit FPLUS and the widely-used ReLU separately, and the result of ours achieves 91.77% accuracy, outperforming ReLU's 90.23%.

Furthermore, to obtain an intuitive impression of the activation layer's effect, we visualize the feature maps from output of the first activation layer in ResNet18, and illustrate them in contrast, grouped by two different activation ways, as seen in Figure 8.

We can recognize that the feature maps generated by FPLUS have more extracted details and are better than the other's. There are two advantages for FPLUS activation judging from the illustration. Firstly, our method activates more regions of characteristics that are prominent in object's profile than ReLU. Secondly, foreground and background are distinguished effectively in our work, while being not so clear for ReLU and most of its channels exhibit ambiguous features. We know ReLU will map it to 0 when there comes a negative excitation, but our FPLUS will always produce non-zero response according to the type of stimulus. As a consequence, we can see the feature maps of ReLU are perceptually darker on the whole than ours.

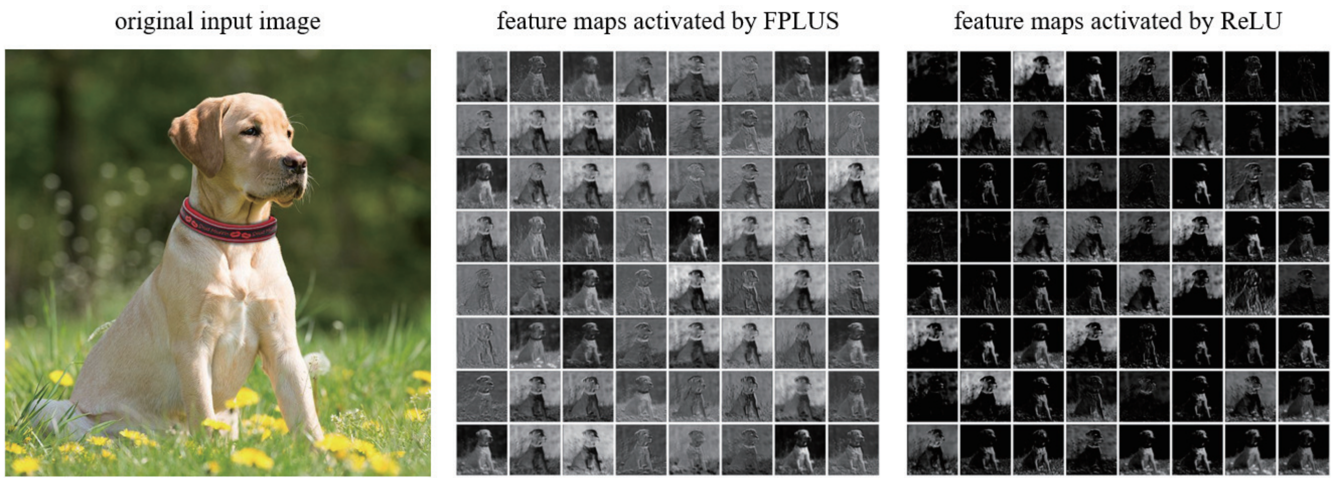

original input image
feature maps activated by FPLUS
feature maps activated by ReLU

**Figure 8.** Visualization of feature maps produced by ResNet18's first activation layer, whose dimension is 64 channels generated by previous convolutional layer with batch normalization. They are presented in 8 × 8 grids and the format is in original grayscale. Comparison displays the impact of two different activation approaches.

### *5.3. Explication of Robustness and Reliability*

Intel hosts an image classification challenge and releases one scenery dataset, consisting of approximately 17K RGB images of size 150 × 150 pixels subordinate to 6 categories, which are buildings, forest, glacier, mountain, sea, and street. The dataset splits in training part of 14K examples and test part of 3K, whose ratio is about 82%:18%.

As widely known, the iid condition is a substantial key to resolving the trouble where a training process becomes harder as the network deepens, and this kind of weakness is called bias shift effect [7] or internal covariate shift [44]. Throughout all those reformative activation theories, batch normalization provides such a solution driving inputs of each layer in the network to remain independently and identically distributed.

Taking ReLU [4,5] as an example, it directly prunes the activation of negative domain for every hidden neuron, making the entire distribution of input data deviate to limit regions of the mapping function, so that gradient of lower level in the network probably disappears during backward propagation, and batch normalization exactly serves for dragging it back to a normal distribution with 0 mean and 1 variance.

Nevertheless, unlike ReLU's bottom truncation, our function FPLUS retains mild activation for negative zone in a gradually saturated way, so as to counteract the shift influence in some degree. Therefore, this means ReLU essentially relies on batch normalization theoretically, while FPLUS remains more tolerable even if without being affected by less impact than ReLU.

In order to validate the aforementioned hypothesis, we conduct an ablation study in this subsection, mainly researching the influence of batch normalization to non-linear activation, so we compare the responses of ReLU and our FPLUS under two situations whether batch normalization is utilized or not.

As shown in Figures 9 and 10, we opt for ResNet-50 [43] to be the framework, but further introduce the regularization mechanism dropout [45] and attention module CBAM [46], to explore their mutual effects. Furthermore, we simultaneously try two attenuation styles for learning rate, including step decay and exponential decay, to investigate which one is more suitable.

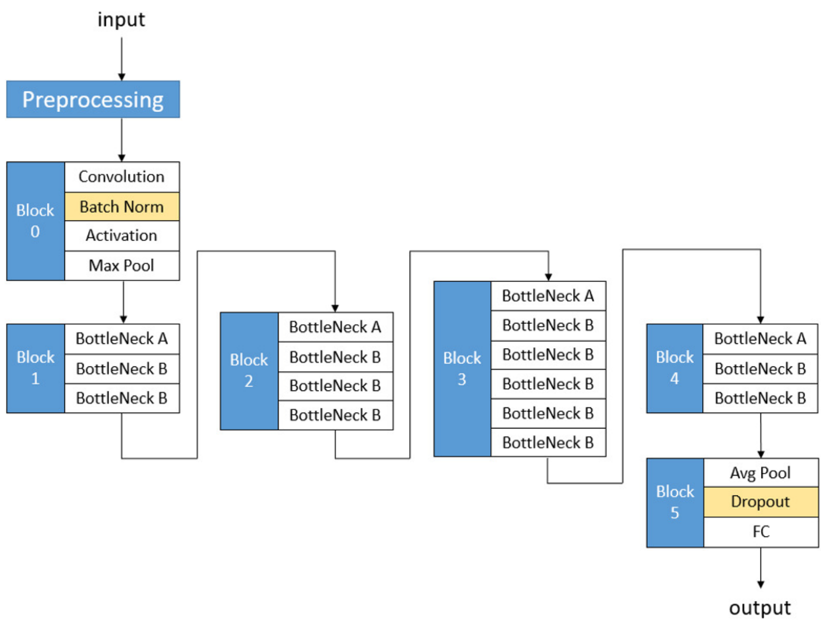


**Figure 9.** Architecture for traditional ResNet-50 with addition of dropout layer. Details in bottleneck A and B are shown below.

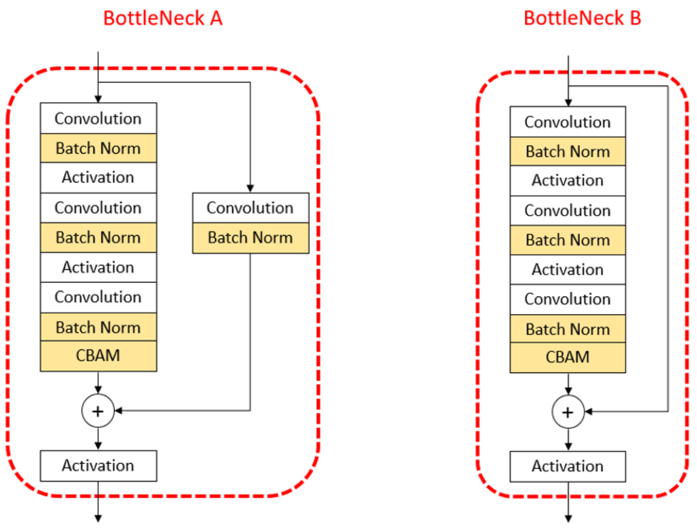


**Figure 10.** Structures of BottleNeck A and BottleNeck B in Figure 9's construction, embedded with CBAM modules specially.

We train the model for 30 epochs with batch size 64 and choose cross-entropy loss as before. The optimizer is SGD with 0.9 being momentum, and the learning rate starts from 0.001, multiplied by 0.1 every 10 epochs in the event of step decay, and making 0.98 the base if it employs exponential decay.

From the results compared in Table 2 we know, on the condition that applying the same decay mode of learning rate, settings with batch normalization are able to produce quite close outcomes for ReLU and FPLUS, while trials with step decay achieve slightly better grades than exponential decay if using identical activation function. On the contrary, if banning batch normalization, both decaying ways for learning rate will generate fallen performance, which is greatly apparent for exponential attenuation. However, our method FPLUS appears more solid and reliable, because it obtains considerably better results than ReLU when they both confront the lack of batch normalization, and this can be seen from the contrast of entries with cyan color in Table 2.

**Table 2.** Comparison of loss and accuracy for training Intel image classification dataset on ResNet-50 with multiple network configurations, grouped by two ways of learning rate decay.

| Architecture Configuration | | | Step Decay for lr | | | | Exponential Decay for lr | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Testing Loss | | Testing Accuracy | | Testing Loss | | Testing Accuracy | |
| **BatchNorm** | **CBAM** | **Dropout** | **ReLU** | **FPLUS** | **ReLU** | **FPLUS** | **ReLU** | **FPLUS** | **ReLU** | **FPLUS** |
| | | | 0.674 | 0.484 | 75.24% | 83.18% | 1.098 | 0.822 | 55.39% | 66.95% |
| ✓ | | | 0.682 | 0.544 | 87.28% | 88.11% | 0.678 | 0.669 | 86.21% | 86.40% |
| | ✓ | | 0.802 | 0.551 | 69.98% | 80.19% | 1.335 | 1.133 | 44.25% | 55.75% |
| | | ✓ | 0.722 | 0.499 | 72.57% | 82.66% | 1.123 | 0.835 | 53.19% | 67.19% |
| ✓ | ✓ | | 0.563 | 0.571 | 86.90% | 86.71% | 0.607 | 0.609 | 85.79% | 86.31% |
| ✓ | | ✓ | 0.846 | 0.490 | 87.08% | 87.44% | 0.683 | 0.684 | 86.03% | 86.51% |
| | ✓ | ✓ | 0.846 | 0.617 | 67.47% | 77.53% | 1.306 | 1.151 | 46.14% | 54.58% |
| ✓ | ✓ | ✓ | 0.586 | 0.496 | 87.50% | 88.50% | 0.651 | 0.666 | 85.71% | 85.75% |

Accuracies in cyan color correspond to the results of trial groups that do not employ batch normalization layers. Check mark ✓ means that the relevant module is added in the network.

According to the ablation study, separate utilization of dropout or CBAM without batch norm seems to degrade the quality of activation, and such a fact manifests the important position of batch norm. This is deemed to lay the foundation for the other two if we want improved performance through combining them.

Figure 11 shows the different responses of ReLU and FPLUS under two contrastive circumstances whether batch normalization is used or not, and both feature maps and heat maps are given to intuitively illustrate their reactions generated by distinct activation layers.

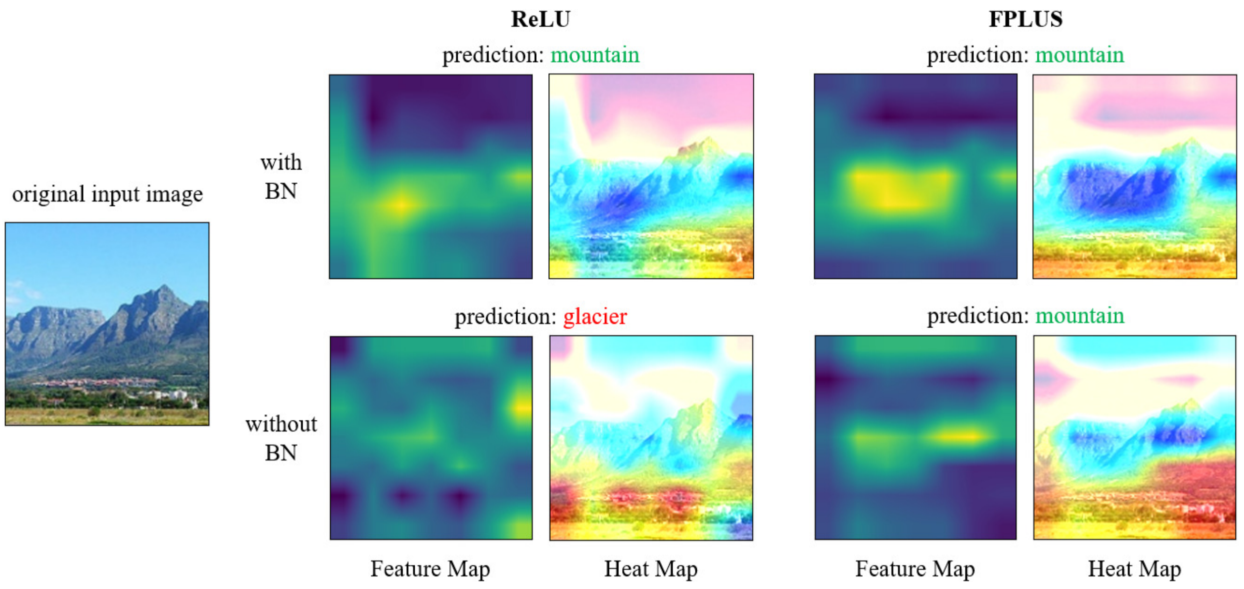


**Figure 11.** Visualization of feature maps and heat maps produced by model's final convolutional layer, under two circumstances whether batch normalization is employed or not. Original input image as well as corresponding predictive results for ReLU and FPLUS are given in the figure to check.

When the network enables batch normalization, models embedded, respectively, with ReLU and FPLUS can both output correct prediction, but feature map of ReLU is much more dispersive than that of FPLUS. For heat maps, the response area of FPLUS precisely lies on the key object, whereas ReLU's does not.

On the other hand, when batch normalization is disabled, models activated by ReLU and FPLUS all produce degenerative feature maps, among which ReLU's is more ambiguous while that of FPLUS is slightly better. With regard to heat maps, except for their common flaws at the top, the one belonging to FPLUS concentrates relatively more on valid targets, yet ReLU's is interfered with by something else in the background. Moreover, it is worth noting that classifier of ReLU misjudges the image category, whereas prediction of FPLUS still hits the right answer.

All things considered, the capacity of batch normalization is absolutely pivotal, on which ReLU relies very much, but our method comparatively shows less dependency and frangibility, notwithstanding it would certainly be better if batch normalization could be exploited at the same time.

### *5.4. Comparison of Performance on CIFAR-10 & CIFAR-100*

Datasets of CIFAR comprise two prominent versions CIFAR-10 and CIFAR-100, which have equal quantities of 60K samples for each and share the same image size of $32 \times 32$ pixels in RGB format. Moreover, they both split into a training set of 50K instances and a validation set of 10K. The sole difference is as their names suggest, CIFAR-10 has 10 categories yet CIFAR-100 has 100, and this means the former one gets 6K examples for every class while the latter one receives only 600, also training set and validation set have covered each category proportionally for both.

Having discussed the intrinsic values of our formulation in previous experiments, now we plan to explore its compatibility with various network architectures and compare it with other more typical activation functions.

Firstly, we select 4 kinds of networks from lightweight to heavyweight as the foundation frameworks to train on CIFAR-10, and their numbers of parameters as well as the amount of calculation are listed in Table 3. Varieties of activation ways also include ReLU's improvers LReLU [6] and PReLU [19], as well as ELU [7] and its variant SELU [8].

**Table 3.** Networks trained on CIFAR-10 and their parameter quantities as well as calculation overhead.

| Network | # Params | FLOPs |
|---|---|---|
| SqueezeNet [47] | 0.735 M | 0.054 G |
| NASNet [48] | 4.239 M | 0.673 G |
| ResNet-50 [43] | 23.521 M | 1.305 G |
| InceptionV4 [49] | 41.158 M | 7.521 G |

For 4 groups of training, the setup of the hyperparameters remain the same, and we still use SGD with 0.9 momentum to optimize cross-entropy loss. The iterative epoch is 50 with batch size being 128, and the learning rate starts by 0.01, multiplied by 0.2 every time at 15th, 30th, 40th epoch for decay, which represent the checkpoints of 30%, 60% and 80% in training process. For the purpose of data augmentation, we take some measures to transform the training set, such as random cropping, flipping, and rotation.

Loss and accuracy on validation set for each network with distinct activation ways are presented in Tables 4 and 5, in which colored items stand for the best results of corresponding networks. We can see our method FPLUS and PFPLUS clearly outperform the others on ResNet-50 and InceptionV4, while this is not so evident on SqueezeNet or NASNet.

**Table 4.** Loss on validation set of CIFAR-10 for each network with different activation functions.

| Activation | SqueezeNet | NASNet | ResNet-50 | InceptionV4 |
|---|---|---|---|---|
| ReLU | 0.350 | 0.309 | 0.335 | 0.442 |
| LReLU | 0.345 | 0.307 | 0.322 | 0.412 |
| PReLU | 0.365 | 0.366 | 0.370 | 0.305 |
| ELU | 0.360 | 0.280 | 0.259 | 0.234 |
| SELU | 0.447 | 0.307 | 0.295 | 0.307 |
| FPLUS | 0.349 | 0.287 | 0.255 | 0.220 |
| PFPLUS | 0.344 | 0.323 | 0.251 | 0.230 |

Entries in red color indicate the best results for each network under validation set.

**Table 5.** Accuracy rate (%) on validation set of CIFAR-10 for each network with different activation functions.

| Activation | SqueezeNet | NASNet | ResNet-50 | InceptionV4 |
|---|---|---|---|---|
| ReLU | 88.67 | 91.55 | 91.94 | 87.95 |
| LReLU | 88.63 | 91.61 | 92.20 | 88.18 |
| PReLU | 88.91 | 91.52 | 92.09 | 92.95 |
| ELU | 87.82 | 91.70 | 92.71 | 92.59 |
| SELU | 84.83 | 90.35 | 91.10 | 90.53 |
| FPLUS | 88.70 | 91.46 | 92.75 | 93.47 |
| PFPLUS | 89.09 | 91.62 | 93.13 | 93.41 |

Entries in magenta color indicate the best results for each network under validation set.

In addition, we completely record their validating processes of loss and accuracy, as shown in Figures 12 and 13. It can be noticed that for heavyweight models such as ResNet-50 and InceptionV4, our activation units display swifter loss descending as well as more prominent accuracy rising that enable them to be distinguished from others.

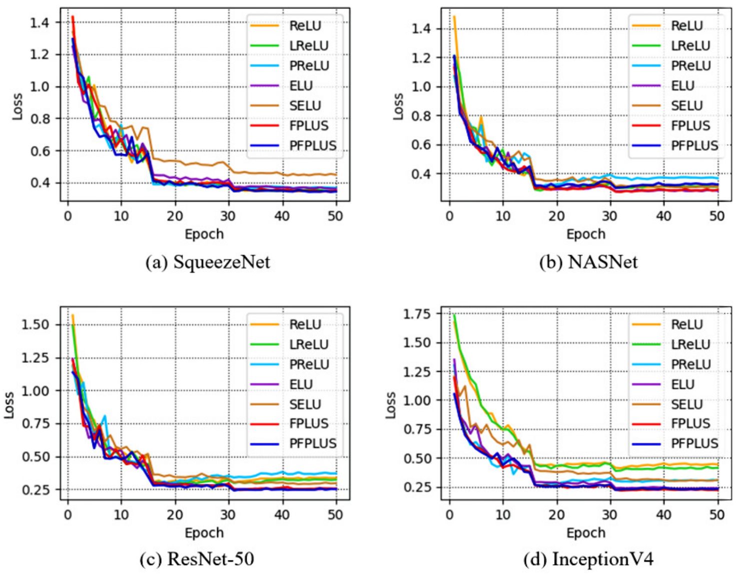


**Figure 12.** Curves of validating loss for each network applying diverse activation methods.

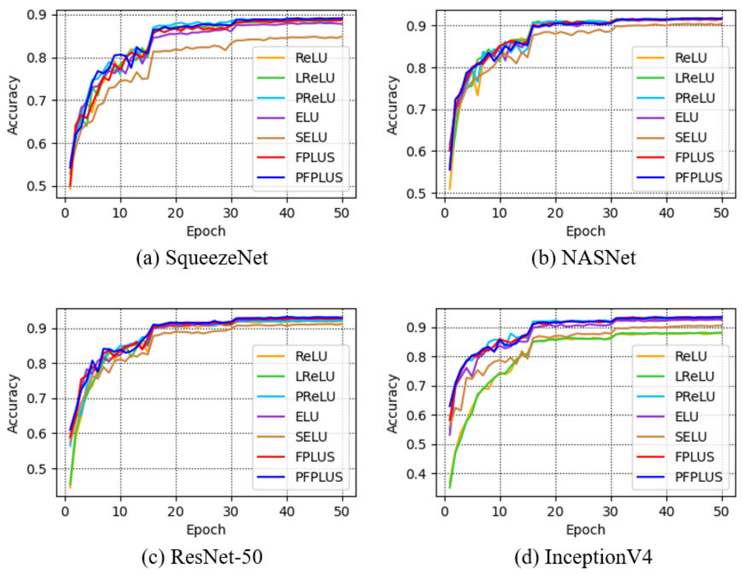


**Figure 13.** Curves of validating accuracy for each network applying diverse activation methods.

Next, we examine our formulae on CIFAR-100, involving 9 types of networks from pioneering classical types to recent lightweight ones. Their magnitude for parameter quantities and complexity for floating-point operations are detailed in Table 6. In the meantime, we additionally try two more non-monotonic activation methods in the test, and they are foregoing Swish [10] and Mish [11].

**Table 6.** Networks trained on CIFAR-100 and their parameter quantities as well as calculation overhead.

| Network | # Params | FLOPs |
|---|---|---|
| AlexNet [50] | 36.224 M | 201.056 M |
| GoogLeNet [51] | 6.403 M | 534.418 M |
| VGGNet-19 [52] | 39.328 M | 418.324 M |
| ResNet-101 [43] | 42.697 M | 2.520 G |
| DenseNet-121 [53] | 7.049 M | 898.225 M |
| Xception [54] | 21.014 M | 1.134 G |
| ShuffleNetV2 [55] | 1.361 M | 45.234 M |
| MobileNetV2 [56] | 2.369 M | 67.593 M |
| EfficientNetB0 [57] | 0.807 M | 2.432 M |

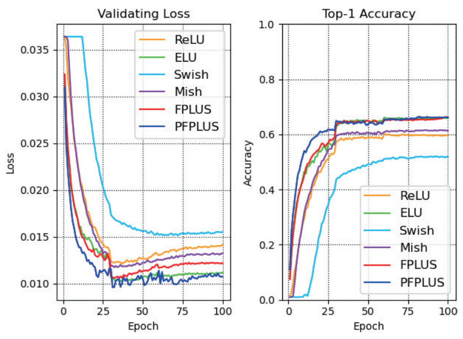
We follow all of the training setups configured in CIFAR-10 experiment, except make 1st epoch the warmup for early period of training schedule and double the rounds of iteration to 100 epochs. Attenuation points for learning rate are still located at 30%, 60% and 90% of the entire process, which correspond to 30th, 60th and 90th epoch.

In terms of dataset scale, the number of categories for CIFAR-100 is 10 times that for CIFAR-10, yet they bear the same volume of examples, which means every class has fewer samples for the model to learn but workload in total is increased, thus the difficulty of the task definitely becomes more challenging. Because of this, we at the same time use top-1 and top-5 accuracy rate two indices to measure the performance of models transferred by different activation functions, as shown in Table 7. Likewise, illustrations of loss descending and accuracy rising for every network are displayed in Figure 14.

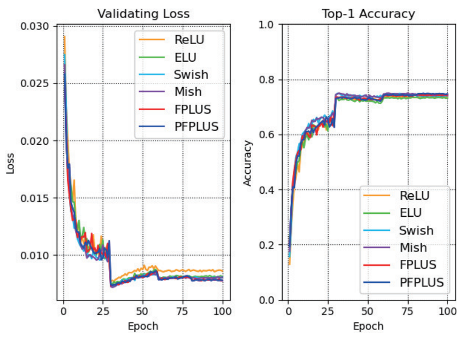
**Table 7.** Top-1 and top-5 accuracy rates (%) on validation set of CIFAR-100 for multiple networks with different activation implementations.

| Network | Top-1 Accuracy | | | | | | Top-5 Accuracy | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ReLU | ELU | Swish | Mish | FPLUS | PFPLUS | ReLU | ELU | Swish | Mish | FPLUS | PFPLUS |
| AlexNet [50] | 59.73 | 65.84 | 51.99 | 61.30 | 66.28 | 66.08 | 85.19 | 89.14 | 79.21 | 86.19 | 89.14 | 89.07 |
| GoogLeNet [51] | 73.69 | 73.17 | 74.01 | 74.10 | 74.38 | 74.69 | 92.50 | 92.36 | 92.39 | 92.81 | 92.77 | 92.85 |
| VGGNet-19 [52] | 69.00 | 65.49 | 68.30 | 68.10 | 68.03 | 68.24 | 87.86 | 84.25 | 88.58 | 88.69 | 89.72 | 89.36 |
| ResNet-101 [43] | 74.42 | 74.99 | 74.48 | 74.97 | 75.20 | 75.51 | 93.06 | 93.18 | 92.57 | 93.13 | 93.29 | 93.26 |
| DenseNet-121 [53] | 74.74 | 72.77 | 75.36 | 75.16 | 73.72 | 74.63 | 92.59 | 93.20 | 93.30 | 93.28 | 93.32 | 93.30 |
| Xception [54] | 72.62 | 72.34 | 72.81 | 72.86 | 72.83 | 73.21 | 91.37 | 91.56 | 91.14 | 91.53 | 91.80 | 92.04 |
| ShuffleNetV2 [55] | 65.32 | 68.01 | 66.73 | 67.48 | 69.01 | 67.89 | 88.58 | 90.53 | 89.46 | 89.72 | 91.23 | 90.17 |
| MobileNetV2 [56] | 64.09 | 65.57 | 65.73 | 65.85 | 65.47 | 66.52 | 88.52 | 89.25 | 88.52 | 89.06 | 89.34 | 90.01 |
| EfficientNetB0 [57] | 62.33 | 63.62 | 63.20 | 63.45 | 64.40 | 64.31 | 86.72 | 88.20 | 85.97 | 86.81 | 88.61 | 89.02 |

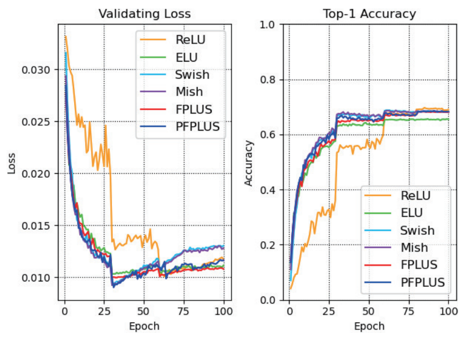
Items in red color indicate the best scores of top-1 accuracy for each network, while those in blue color are the best ones of top-5.



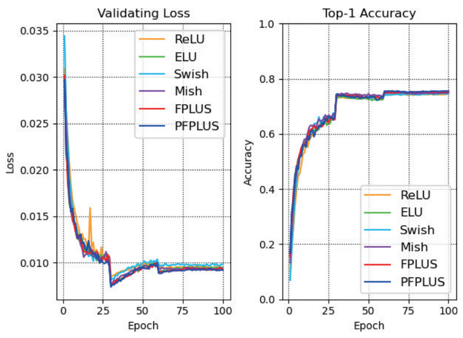
(a) AlexNet



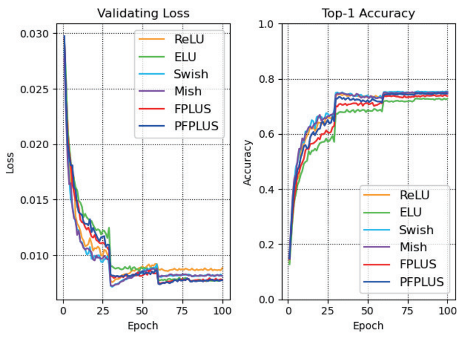
(b) GoogLeNet



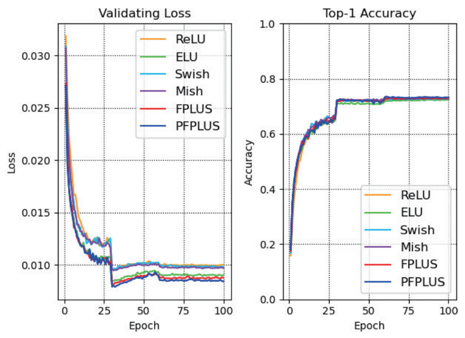
(c) VGGNet-19



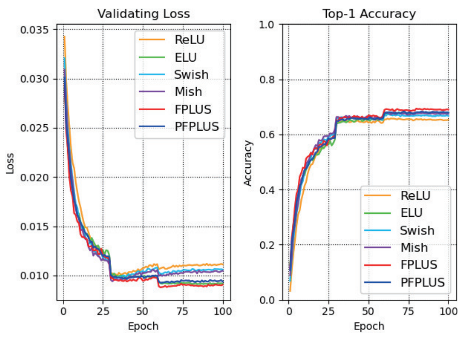
(d) ResNet-101



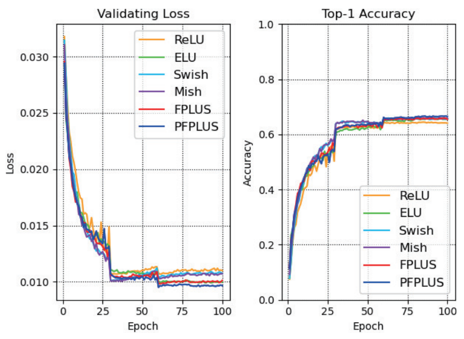
(e) DenseNet-121



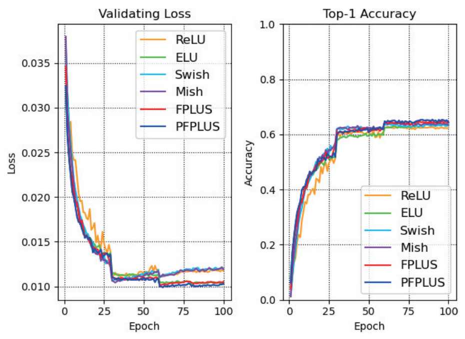
(f ) Xception



(g) ShuffleNetV2



(h) MobileNetV2



(i) EfficientNetB0

**Figure 14.** Curves of validating loss and top-1 accuracy for each model activated by different methods. Subfigures (**a**–**i**) are organized by the type of networks. Every subfigure contains a loss descent record on the left, and an illustration of top-1 accuracy ascent on the right.

According to the statistics contrasted in Table 7, it can be found that our method outperforms the others among seven of nine selected networks in terms of top-1 accuracy rate, either FPLUS or PFPLUS. For models of AlexNet, ShuffleNetV2 and EfficientNetB0, FPLUS achieves good results by a clear margin, while its parametric version PFPLUS gets better marks on GoogLeNet, ResNet-101, Xception and MobileNetV2. On the other hand, we are aware that our method is not always the optimal choice for all networks, so it is not perfect when embedded with an inappropriate model that they cannot coordinate. Therefore, structural characteristics of a network perhaps ought to be also considered, to match up with a suitable activation unit if screen trials are allowed.

As far as top-5 accuracy rate is concerned, the approach we propose obtains all the best scores on every network without exception. As a rule of thumb, this maybe suggests the method still exists more potentialities to be developed in future works.

Moreover, from Figure 14 we are able to notice that an obvious delay phenomenon happens to AlexNet's process at the beginning when using Swish activation, and loss as well as accuracy start to vary perceptibly only after about 10 epochs. Furthermore, even though ReLU comes out on top in VGGNet-19 for top-1 accuracy, its training process is full of drastic oscillation, jittering acutely when recorded in dynamic curves, and so does it in the procedures of ResNet101, MobileNetV2 and EfficientNetB0, but not that severely.

On balance, the proposed FPLUS and PFPLUS leave us the impression of robustness and consistency. Their curves show almost no anomalous vibration or abrupt fluctuation, with loss falling normally and accuracy ascending steadily. Neither sharp crest nor deep trough emerges in every model's record, which signifies our approach is capable of keeping the training procedure intact rather than vulnerable.

At any rate, in comprehensive consideration of the performance on CIFAR-10 and CIFAR-100, we hold that our proposed method shows comparable competitiveness and invariable stability, which make for the promotion of accommodating to as many integrated networks as possible.

### *5.5. Experimental Results on ImageNet-ILSVRC2012*

ImageNet dataset dates from 2009 by Fei-Fei Li et al. [58], and then an open competition named ImageNet Large Scale Visual Recognition Challenge was established. ILSVRC has been held for seven consecutive sessions until Kaggle took it over and proceeded with maintenance in 2017.

Quite literally, due to its impeccable status as a benchmark, ImageNet is one of the excellent masterpieces in the field of computer vision that nobody neglects. ImageNet is famous for the tremendously massive data samples and incredibly abundant categories, containing more than 14 million images and 20 thousand classes that no other datasets transcend.

Generally speaking, peers tend to utilize a subset of the whole, so we choose the popular one that most people do, and that is ILSVRC2012, also known as ImageNet-1000. It comprises 1K categories more than 1.3M images, of which training set owns 1,281,167 examples while validation set has 50,000. Therefore we can imagine it would be an enormously formidable challenge for training hardware, requiring accelerative devices of high quality.

Considering the time cost of training, we opt for representative ResNet-101 to be the typical framework, and apply ReLU, ELU and our FPLUS as the activation unit, respectively, to test their underlying limits when dealing with such a huge workload.

Restricted by available resources, we train the model of each group for 100 epochs with batch size 64, and the learning rate commences with 0.1 while decays by 0.1 at 30th, 60th and 90th epoch likewise. Cross-entropy loss is still the target, and SGD with momentum 0.9 is the optimizer.

Tables 8 and 9 show the final results of three activation ways after a long time of training. It can be found that ELU does not perform that well, although it proposes to solve the problem of bias shift effect, which is a disadvantage of ReLU. However, this can be also alleviated by another way of batch normalization, and ResNet-101 is originally equipped with batch norm layer itself, so we see ReLU achieves the best results on training set. As for validation set, our FPLUS reaches a commensurate level with ReLU, even a little bit better. As a consequence, we think our activation function is an effective and reliable approach that is capable of standing the trial, even if confronting a large workload of high intensity such as ImageNet.

**Table 8.** Training loss of each activation method applied on ResNet-101 for ILSVRC2012.

| **Activation Method** | **ReLU** | **ELU** | **FPLUS** |
|---|---|---|---|
| Training Loss | 1.10 | 1.26 | 1.16 |

**Table 9.** Top-1 and top-5 accuracy rates (%) of each activation method applied on ResNet-101 for ILSVRC2012, grouped by training set and validation set.

| Activation Method | Training Set | | Validation Set | |
|---|---|---|---|---|
| | Top-1 Accuracy | Top-5 Accuracy | Top-1 Accuracy | Top-5 Accuracy |
| ReLU | 73.98 | 90.76 | 73.55 | 91.37 |
| ELU | 70.23 | 88.01 | 72.15 | 90.36 |
| FPLUS | 72.20 | 89.23 | 73.64 | 91.64 |

## 6. Conclusions

Taking inspiration from conceptual bionics and inverse operation, this paper proposes a novel and subtle activation method, which involves mathematical first power and switch-like sign function. We theoretically derive the formula under certain specified prior knowledge, and obtain a concise expression form that owns two adjustable parameters. Either of them regulates the shape of our function, while they can be fixed or learnable. We name our parametric formulation PFPLUS, and FPLUS is one particular case. Furthermore, we investigate the influence of initialization variety for trainable parameters, and find out the limited interval near 1 is a preferable option with interpretability to achieve good performance. Meanwhile, we conduct an ablation study to demonstrate our activation unit is not that vulnerable to the lack of batch normalization, showing less dependency but more robustness. Furthermore, when compared with other typical activation approaches in the same task, our new approach manifests not only appreciable competence but also consistent stability, providing considerable advantages to its undeniable competitiveness. On the whole, we hold the opinion that our work to some extent enriches the diversity of activation theories.

Future works can be continued in other advanced tasks such as object detection and semantic segmentation, exploring its mechanism of function when the activation method is changed under these circumstances.

**Author Contributions:** Conceptualization, B.D.; Data curation, Y.Y.; Formal analysis, B.D.; Funding acquisition, X.D.; Investigation, Y.Y.; Methodology, B.D.; Project administration, X.D.; Resources, Y.Y.; Supervision, X.D.; Validation, B.D.; Writing—original draft, B.D.; Writing—review and editing, X.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by National Natural Science Foundation of China (Grant No. NSFC-61872396).

**Data Availability Statement:** The implementation code of our proposed method can be found at https://github.com/Lamborxhini/FPLUS-activation (accessed on 23 June 2022). Publicly available datasets were analyzed in this study. These datasets can be found via the URLs below: MNIST Dataset: http://yann.lecun.com/exdb/mnist/ (accessed on 23 June 2022). Fashion-MNIST Dataset: https://github.com/zalandoresearch/fashion-mnist (accessed on 23 June 2022). Kaggle's Dogs Vs Cats Dataset: https://www.kaggle.com/biaiscience/dogs-vs-cats (accessed on 23 June 2022). Intel Image Classification Dataset: https://www.kaggle.com/puneet6060/intel-image-classification (accessed on 23 June 2022). CIFAR-10 & CIFAR-100 Datasets: http://www.cs.toronto.edu/~kriz/cifar.html (accessed on 23 June 2022). ImageNet Dataset: https://image-net.org/ (accessed on 23 June 2022).

**Acknowledgments:** This work was supported by School of Electronics and Information Technology, Sun Yat-sen University; Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai); School of Computing and Information Systems, The University of Melbourne.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
2. Hodgkin, A.L.; Huxley, A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **1990**, *52*, *117*. [CrossRef]
3. Dayan, P.; Abbott, L.F. *Theoretical Neuroscience: Computational & Mathematical Modeling of Neural Systems*; The MIT Press: Cambridge, MA, USA, 2001.
4. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; pp. 807–814.
5. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), Ft. Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
6. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In Proceedings of the 30 th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
7. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv* **2015**, arXiv:1511.07289.
8. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-Normalizing Neural Networks. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 972–981.
9. Elfwing, S.; Uchibe, E.; Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw.* **2018**, *107*, 3–11. [CrossRef] [PubMed]
10. Ramachandran, P.; Zoph, B.; Le, Q.V. Searching for Activation Functions. *arXiv* **2017**, arXiv:1710.05941.
11. Misra, D. Mish: A Self Regularized Non-Monotonic Neural Activation Function. *arXiv* **2019**, arXiv:1908.08681v2.
12. Goodfellow, I.; Warde-Farley, D.; Mirza, M.; Courville, A.; Bengio, Y. Maxout Networks. In Proceedings of the 30th International Conference on International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
13. Ma, N.; Zhang, X.; Liu, M.; Sun, J. Activate or Not: Learning Customized Activation. *arXiv* **2021**, arXiv:2009.04759.
14. Howard, A.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.C.; Tan, M.; Chu, G.; Vasudevan, V.; Zhu, Y.; Pang, R.; et al. Searching for MobileNetV3. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 1314–1324. [CrossRef]
15. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. [CrossRef]
16. Courbariaux, M.; Bengio, Y.; David, J.P. BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations. *arXiv* **2015**, arXiv:1511.00363.
17. Berradi, Y. Symmetric Power Activation Functions for Deep Neural Networks. In Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications, Rabat, Morocco, 2–5 May 2018; pp. 1–6. [CrossRef]
18. Gulcehre, C.; Moczulski, M.; Denil, M.; Bengio, Y. Noisy Activation Functions. *arXiv* **2016**, arXiv:1603.00391.
19. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034. [CrossRef]
20. Trottier, L.; Giguère, P.; Chaib-draa, B. Parametric Exponential Linear Unit for Deep Convolutional Neural Networks. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 207–214. [CrossRef]
21. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
22. Lecun, Y.; Boser, B.; Denker, J.; Henderson, D.; Howard, R.; Hubbard, W.; Jackel, L. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
23. Amari, S.I. Natural Gradient Works Efficiently in Learning. *Neural Comput.* **1999**, *10*, 251–276. [CrossRef]
24. Attwell, D.; Laughlin, S.B. An Energy Budget for Signaling in the Grey Matter of the Brain. *J. Cereb. Blood Flow Metab.* **2001**, *21*, 1133–1145. [CrossRef] [PubMed]
25. Lennie, P. The Cost of Cortical Computation. *Curr. Biol. CB* **2003**, *13*, 493–497. [CrossRef]
26. Xu, B.; Wang, N.; Chen, T.; Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv* **2015**, arXiv:1505.00853.
27. Shang, W.; Sohn, K.; Almeida, D.; Lee, H. Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units. *arXiv* **2016**, arXiv:1603.05201.
28. Ma, N.; Zhang, X.; Sun, J. Funnel Activation for Visual Recognition. In Proceedings of the ECCV, Glasgow, UK, 23–28 August 2020. [CrossRef]
29. Chen, Y.; Dai, X.; Liu, M.; Chen, D.; Yuan, L.; Liu, Z. Dynamic ReLU. In Proceedings of the ECCV, Glasgow, UK, 23–28 August 2020. [CrossRef]
30. Barron, J.T. Continuously Differentiable Exponential Linear Units. *arXiv* **2017**, arXiv:1704.07483.
31. Zheng, Q.; Tan, D.; Wang, F. Improved Convolutional Neural Network Based on Fast Exponentially Linear Unit Activation Function. *IEEE Access* **2019**, *7*, 151359–151367.

32. Basirat, M.; Roth, P.M. The Quest for the Golden Activation Function. *arXiv* **2018**, arXiv:1808.00783.
33. Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv* **2016**, arXiv:1606.08415.
34. Dugas, C.; Bengio, Y.; Bélisle, F.; Nadeau, C.; Garcia, R. Incorporating Second-Order Functional Knowledge for Better Option Pricing. In Proceedings of the 13th International Conference on Neural Information Processing Systems, Denver, CO, USA, 1 January 2000; pp. 451–457.
35. Ying, Y.; Su, J.; Shan, P.; Miao, L.; Peng, S. Rectified Exponential Units for Convolutional Neural Networks. *IEEE Access* **2019**, *7*, 2169–3536 . [CrossRef]
36. Kiliarslan, S.; Celik, M. RSigELU: A nonlinear activation function for deep neural networks. *Expert Syst. Appl.* **2021**, *174*, 114805. [CrossRef]
37. Pan, J.; Hu, Z.; Yin, S.; Li, M. GRU with Dual Attentions for Sensor-Based Human Activity Recognition. *Electronics* **2022**, *11*, 1797. [CrossRef]
38. Tedesco, S.; Alfieri, D.; Perez-Valero, E.; Komaris, D.S.; Jordan, L.; Belcastro, M.; Barton, J.; Hennessy, L.; O'Flynn, B. A Wearable System for the Estimation of Performance-Related Metrics during Running and Jumping Tasks. *Appl. Sci.* **2021**, *11*, 5258. [CrossRef]
39. Hubel, D.H.; Wiesel, T.N. Receptive Fields of Single Neurons in the Cat's Striate Cortex. *J. Physiol.* **1959**, *148*, 574–591. [CrossRef]
40. Bhumbra, G.S. Deep learning improved by biological activation functions. *arXiv* **2018**, arXiv:1804.11237.
41. Ramachandran, P.; Zoph, B.; Le, Q. Swish: A Self-Gated Activation Function. *arXiv* **2017**. arXiv.1710.05941.
42. Lecun, Y.; Bottou, L. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
43. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
44. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.
45. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
46. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. In Proceedings of the Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018; pp. 3–19. [CrossRef]
47. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv* **2016**, arXiv:1602.07360.
48. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. *arXiv* **2017**, arXiv:1707.07012.
49. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv* **2016**, arXiv:1602.07261.
50. Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
51. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9. [CrossRef]
52. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
53. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269. [CrossRef]
54. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807. [CrossRef]
55. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In Proceedings of the Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018; Springer: Cham, Switzerland, 2018; Volume 11218.
56. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520. [CrossRef]
57. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.
58. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255. [CrossRef]