Maximum and Leaky Maximum Propagation

1st Wolfgang Fuhl
University Tübingen
Human Computer Interaction
Tübingen, Germany
wolfgang.fuhl@uni-tuebingen.de

2nd Enkelejda Jasneci
University Tübingen
Human Computer Interaction
Tübingen, Germany
enkelejda.kasneci@uni-tuebingen.de

***Abstract*—In this work, we present an alternative to conventional residual connections, which is inspired by maxout nets. This means that instead of the addition in residual connections, our approach only propagates the maximum value or, in the leaky formulation, propagates a percentage of both. In our evaluation, we show on different public data sets that the presented approaches are comparable to the residual connections and have other interesting properties, such as better generalization with a constant batch normalization, faster learning, and also the possibility to generalize without additional activation functions. In addition, the proposed approaches work very well if ensembles together with residual networks are formed. LinkToCodeBlind**

***Index Terms*—Neural Network, Deep Neural Network, Residual Block, Maximum Propagation, Maximum Selection, Residual Block Alternative, Ensemble Neural Networks**

I. Introduction

The development of deep neural networks [1]–[3] has undergone steady advancements over the past few decades. These advancements, especially through the introduction of convolutions [2], residual blocks [3], and memory functionality [4], have led to deep neural networks being the de facto standard approach for many areas of algorithm development today. This has led to major advances in the areas of computer vision [5], speech recognition [6], pattern recognition [7], human computer interaction [8], and big data processing [9]. The application areas of deep neural networks in modern times are Autonomous Driving [10], Gaze Estimation [11], Collision Detection [12], Industrial Algorithm Development [13], Tumor Detection [14], Person Identification [15], Text Translation [16], Image Generation [17], Quality Enhancement of Images, and many more.

The research areas of deep neural networks include activation functions [18], optimization methods [19], robustness of neural networks [20], explainability and definitional domains [21], unsupervised learning [22], reinforcement learning [23], application to graphs resp. dynamically structured data [24], computing metrics [25], autonomous learning or artificial intelligence [26], and many more. In this work, we deal with an alternative to residual blocks that optimize the function $R(x, y) = f(x)+y$. x and y are here two consecutive outputs of convolution layers.

Our approach builds on maxout nets [27], which were introduced as activation functions requiring k times the number of neurons. For $k = 2$ these maxout nets optimize the function $M(x, y) = max(x, y)$ where x and y are the outputs of two neuron or convolution layers which are parallel to each other (Receiving the same input). This concept can be directly integrated as a replacement for the residual function, thus not requiring any additional learnable parameter or neurons. This means we use the function $M(x, y) = max(x, y)$ whereby x and y are consecutive convolutions. In addition, we formulate the leaky max propagation (LMP) to combine the residual function with the maxout function. This combination optimizes the equation $LMP(x, y) = max(x, y) * \alpha + min(x, y) * \beta$ where x and y are again sequential convolutional layer outputs and α and β are weighting factors which are fixed before training and satisfy the condition $\alpha + \beta = 1$.

Difference to Maxout

- We do not need twice the amount of neurons
- We use the maximum as a combination or propagation layer
- We extended the formulation to leaky max propagation.

Contribution of this work

- We show that maximum propagation can be used effectively as a replacement for residual connections as well as an alternative to form ensambles.
- We extend the formulation of maximum propagation to allow a fraction to pass (leaky maximum propagation).
- Our proposed approaches generalize well without batch normalization and since batch normalization makes DNNs less robust [28], [29], we see this as an advantage.
- Our proposed approaches can be trained without additional activation function.

II. Method

Figure 1 shows a visual representation of the structure of residual blocks (a), maximum propagation blocks (b), and leaky maximum propagation blocks (c). There is no difference between the blocks except for the final combination of the block's input and the output of the internal layers. Formally, the residual blocks and the maximum propagation blocks optimize different functions where the leaky maximum propagation function is a generalization of both functions. For the input of a block $x \in \mathbb{R}^d$ and the function of the internal blocks $f(x)$, the function which is optimized in residual blocks is given by:

$$Residual(x) = f(x) + x \tag{1}$$

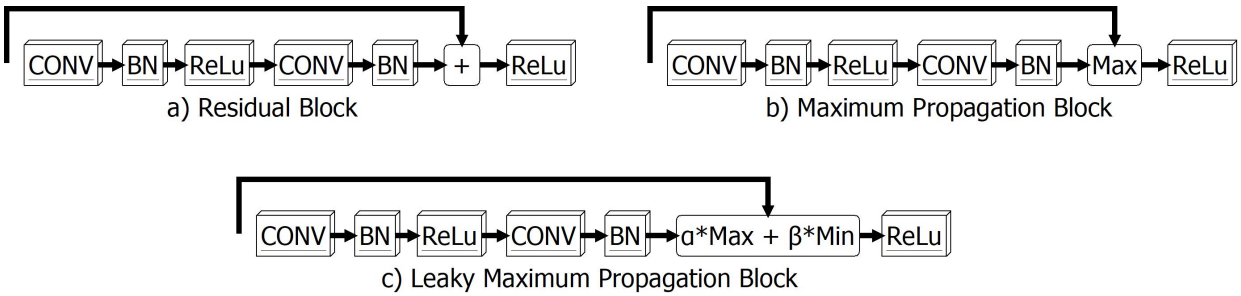

Fig. 1. Visual description of the architecture from residual blocks (a), maximum propagation blocks (b), and leaky maximum propagation blocks (c) for ResNet-34 (ResNet-50 uses two depthwise convolution before the 3×3 convolution block in the center but the concept is identical). All three share the same placement of the convolution (CONV), batch normalization (BN), and rectifier linear unit (ReLu) blocks. The only difference is the performed operation to combine the result of the block computation and the input to the block.

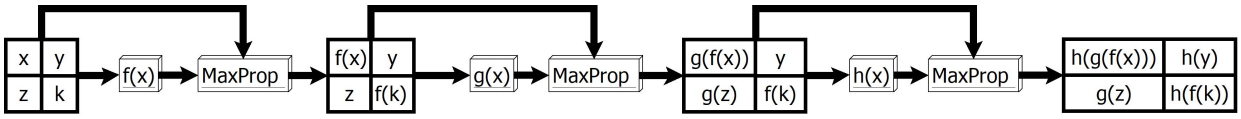

Fig. 2. Visual description of functions learned by consecutive maximum propagation blocks. On the left side is the input to the three blocks with the internal functions f(x), g(x), and h(x). Based on the maximum selection each input has a different output function.

The function 1 tries to find an optimization $f(x)$ for the input x. In the case of blocks connected in series, this means that each block produces an improvement for the previous output, which is passed from block to block. This results in a nested optimization that can be interpreted as an ensemble of many small networks [30]. The biggest advantage of residual blocks, however, is that they help circumvent the vanishing gradient problem [30].

$$MaxProp(x) = \begin{cases} f(x), & f(x) \geq x \\ x, & else \end{cases} \tag{2}$$

In equation 2, the maximum propagation function is shown, which is also used as the activation function in maxout nets [27]. Since we use this as a combining function for nested blocks, we do not need twice the amount of neurons as is the case in maxout nets [27]. Compared to the residual blocks, only the maximum value is forwarded. Hereby, the whole network optimizes different long and small nets in parallel. This means that whole blocks can be skipped internally. For the backpropagation of the error, both inputs receive the same error value, which is added to the input connection since it receives the error of the internal function additionally.

Figure 2 shows a visual representation of the function approximation of maximum propagation blocks. Each input value (x,y,z,k) is assigned a different output function based on the maximum selection. The functions f(x), g(x), and h(x) are computed by the internal convolutions, batch normalization, and activation functions in the intermediate layer as shown in Figure 1. Through this, the network now no longer learns an optimization to the previous layer in each layer, but different deep networks, which are assembled based on the maximum values. For the backpropagation, the error is given to the maximum value only, which means, that the lesser value does not receive an error and therefore, can be theoretically unused in the network. This brings us to our general formulation of the maximum propagation, which is the leaky maximum propagation function in equation 3.

$$MaxProp(x) = \begin{cases} f(x) * \alpha + x * \beta, & f(x) \geq x \\ x * \alpha + f(x) * \beta, & else \end{cases} \tag{3}$$

The leaky maximum propagation function has two additional parameters α and β, which have to be set in advance. These two parameters, specify how much weight is given to the maximum and the minimum value. For the parameters $\alpha = 1.0$ and $\beta = 1.0$ this would be the residual function. If we use $\alpha = 1.0$ and $\beta = 0.0$ we would have the maximum propagation function. For any other value, we receive a function, which optimizes both functions in parallel but weighted (With the exception of $\alpha = 0.0$ and $\beta = 0.0$). This means that the leaky maximum propagation function is a generalization of the residual and maximum propagation function. For the back propagation of the error we use the same values for α and β to assign each input the weighted error. The idea behind this formulation is to overcome a possible unused layer which can happen in the maximum propagation function.

Figure 3 shows further possible combinations of the residual and maximum propagation blocks as an alternative to the leaky maximum propagation formulation. In Figure 3 a) the results of the maximum propagation and the residual blocks are concatenated. Hereby, each layer has only half the depth (Or output channels) to allow a fair comparison with the same number of parameters. This is the use of the inception [31] technique which processes several small nets in parallel and concatenates their results at the end. Figure 3 b) shows the residual and maximum propagation blocks connected in series. Here the residual block optimizes the previous result and the maximum propagation block selects either the improvement or

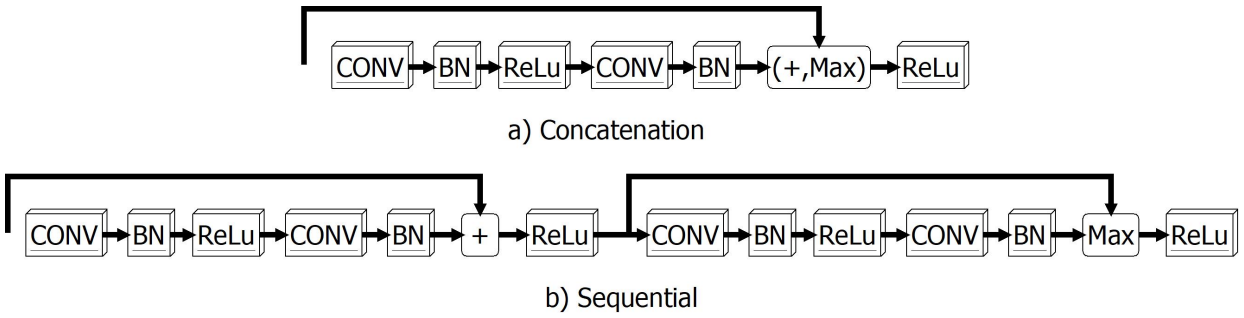

Fig. 3. Visual description of possible alternatives to combine the residual and maximum propagation layer. In a) the concatenation is shown and in b) the alternating interconnection of the layers.

another internal optimization. These nestings can be continued arbitrarily. However, this requires a large number of GPUs and a grid search over many possibilities, which is beyond the scope of this work. In this work, we consider only the alternating use of residual and maximum propagation blocks if "Alternating Blocks" is specified in brackets after a model.

Figure 4 shows the use of the residual (a), maximum propagation (b), and leaky maximum propagation (c) blocks without activation function. For this, the two internal convolutions must also be reduced to one, doubling the number of layers, in order to arrive at the same set of learnable parameters and allow for a fair comparison. The ReLu after each block was also removed and since we used global average pooling [32] and only one fully connected layer at the end, there is no activation function in the whole network. This is a very interesting property of maximum propagation, since maximum selection is already an activation function, as shown earlier in maxout nets [27]. For residual blocks (Figure 4 a)), this does ofcourse not work and the network will either learn nothing or end up in not a number. For maximum propagation this work surprisingly and even better than the leaky maximum propagation, which we will show in our results.

III. Evaluation

In this section we show the performance of maximum propagation and leaky maximum propagation in comparison to residual blocks, which are marked as (Addition) in the evaluations, on a variety of publicly available data sets. For the leaky maximum propagation, we always used $\alpha = 0.9$ and $\beta = 0.1$. The hardware to train all the models is an AMD Ryzen 9 2950X with 3.50 GHz and 64 GB DDR4 Ram. The used GPU is an NVIDIA 3090 RTX with 24 GB DDR6X Ram. As deep learning framework we used DLIB [33] version 19.21 in which the provided code can be copy pasted an directly executed. The NVIDIA driver version of the system is 461.09, the operating system is Windows 10 64Bit, the CUDA version is 11.2, and the cuDNN version used is 8.1.0. First we describe all used data sets and the training and test splits we used in this work.

SVHN [34] is a public data set with real world images from house numbers. Similar to MNIST [35] it contains all numbers from 0 to 9. The resolution of each image is 32×32 with all color channels (red, green, and blue). For training the authors provide 73,257 images and for validation 26,032 images. There is also a third set without annotations which we did not use in our evaluation.

FashionMNIST [36] is an image classification data set inspired by MNIST [35]. The images are gray scale and have a resolution of 28×28. The data set contains 60,000 images for training and 10,000 images for validation.

CIFAR10 [37] is a publicly available data set with RGB (red, green, and blue) images and 10 classes. The resolution of the images is 32×32 and the authors provided 50,000 images for training and 10,000 images for validation. The data set is balanced which means, that for each class the same amount of samples is provided.

CIFAR100 [37] is a similar data set to CIFAR10 but with 100 instead of 10 classes. The resolution of the images is 32×32 and each image has all three color channels (red, green, and blue). The training set contains 500 images per class which sums up to 50,000 images. The validation set has 100 images per class which is 10,000 in total. As CIFAR10 this data set is balanced.

ImageNet ILSVRC2015 [38] is one of the largest publicly available image classification data sets. It has one thousand classes and more than one million images for training. The images have different resolutions which contains an additional challenge. For validation 50,000 images are provided.

VOC2012 [39] is a publicly available data set for semantic segmentation. Semantic segmentation means that the object has to be extracted from the image with pixel accuracy and each pixel has to be classified into the object class. For training 1,464 images are provided with 3,507 segmented and classified objects. The validation set consists of 1,449 images with 3,422 segmented objects. In addition to the images with segmented objects, a third set is provided without annotations. In our evaluation we did not use the third set.

500kCloser [40], [41] is a publicly available data set with 866,069 images. It contains semantic segmentations for the pupil and the iris as well as eyeball parameters and gaze vectors. In our evaluation we only used the regression of the eyeball parameters as well as the gaze vector. The resolution of the images is 192×144 and the recordings are gray scale. The images are split in two videos, one from simulator driving

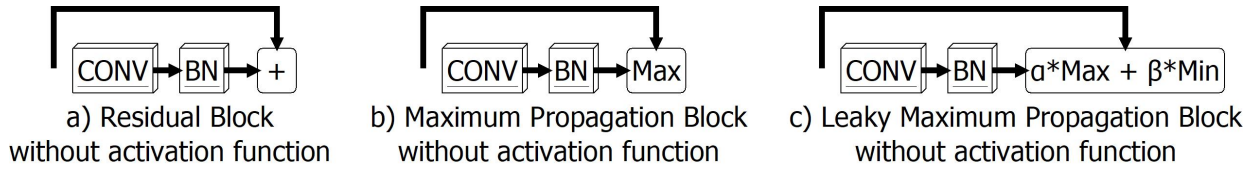

Fig. 4. Visual description of the architecture from residual blocks (a), maximum propagation blocks (b), and leaky maximum propagation blocks (c) without any actiavation function for ResNet-34 blocks (ResNet-50 would requier to pack each convolution block between a max propagation connection). All three share the same placement of the convolution (CONV) and batch normalization (BN) blocks. The only difference is the performed operation to combine the result of the block computation and the input to the block. There is also no activation function after the block and due to the global average pooling [32] no activation function in the entire model.

TABLE I

SHOWS THE CLASSIFICATION ACCURACY RESULTS FOR THE DATA SETS CIFAR10, CIFAR100, FASHIONMNIST, AND SVHN WITH DIFFERENT ARCHITECTURES AND **BATCH NORMALIZATION** AS AVERAGE OVER 5 RUNS WITH STANDARD DEVIATION AFTER $\pm$. THE TEXT IN THE BRACKETS AFTER THE ARCHITECTURE SPECIFIES THE PERFORMED COMBINATION OPERATION TO THE INPUT AND OUTPUT OF EACH BLOCK.

Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0005, Learning rate=0.1, Batch size=100, Training time=100 epochs, Learning rate reduction after each 20 epochs by 0.1

Data augmentation: Shifting by up to 4 pixels in each direction and padding with zeros. Horizontal flipping for the CIFAR datasets only. Mean (Red=122, Green=117, Blue=104) subtraction and division by 256.

Method	CIFAR10	CIFAR100	FashionMNIST	SVHN
ResNet-34 (Additon)	92.52 ± 0.25	73.16 ± 0.61	94.83 ± 0.22	96.1 ± 0.23
ResNet-34 (Maximum)	93.36 ± 0.08	73.13 ± 0.69	95.00 ± 0.18	$\mathbf{96.36 \pm 0.09}$
ResNet-34 (Leaky Max)	$\mathbf{93.88 \pm 0.09}$	$\mathbf{73.66 \pm 0.2}$	$\mathbf{95.05 \pm 0.14}$	96.19 ± 0.19
ResNet-50 (Additon)	93.13 ± 0.19	74.41 ± 0.41	95.44 ± 0.19	96.77 ± 0.19
ResNet-50 (Maximum)	94.02 ± 0.21	74.4 ± 0.41	95.77 ± 0.21	$\mathbf{97.11 \pm 0.21}$
ResNet-50 (Leaky Max)	$\mathbf{94.49 \pm 0.18}$	$\mathbf{74.91 \pm 0.4}$	$\mathbf{96.01 \pm 0.18}$	96.95 ± 0.21

TABLE II

SHOWS THE CLASSIFICATION ACCURACY RESULTS FOR THE DATA SETS CIFAR10, CIFAR100, FASHIONMNIST, AND SVHN WITH DIFFERENT ARCHITECTURES AND **FIXED BATCH NORMALIZATION (MEAN=0,STD=1,SCALE=1,SHIFT=0)** AS AVERAGE OVER 5 RUNS WITH STANDARD DEVIATION AFTER $\pm$. THE TEXT IN THE BRACKETS AFTER THE ARCHITECTURE SPECIFIES THE PERFORMED COMBINATION OPERATION TO THE INPUT AND OUTPUT OF EACH BLOCK.

Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0005, Learning rate=highest possible (0.01 or 0.001), Batch size=100, Training time=100 epochs, Learning rate reduction after each 20 epochs by 0.1

Data augmentation: Shifting by up to 4 pixels in each direction and padding with zeros. Horizontal flipping for the CIFAR datasets only. Mean (Red=122, Green=117, Blue=104) subtraction and division by 256.

Method	CIFAR10	CIFAR100	FashionMNIST	SVHN
ResNet-34 (Additon)	76.83 ± 0.72	35.63 ± 0.2	92.4 ± 0.15	93.16 ± 0.05
ResNet-34 (Maximum)	$\mathbf{88.76 \pm 0.05}$	$\mathbf{59.96 \pm 0.23}$	$\mathbf{93.45 \pm 0.09}$	$\mathbf{95.92 \pm 0.05}$
ResNet-34 (Leaky Max)	80.26 ± 0.1	44.54 ± 0.46	93.09 ± 0.15	94.45 ± 0.02

recordings (509,379 images) and one from real world driving recordings (356,690 images). Since there is no predefined train and test split we used the simulator recordings for training and the real-world recordings for validation.

Table I shows the results with batch normalization for the model ResNet-34 and ResNet-50. Averaged over five runs, the maximum propagation and the leaky maximum propagation seem to perform at least as well or even slightly better than ordinary residual connections. Between the leaky maximum propagation and the maximum propagation only the data set SVHN seems to make a difference, whereas the maximum propagation seems to work a bit better. Since we can't see much difference, although the maximum propagation networks learn a different function (See Figure 2), with the standard networks, we investigated why this is the case and found that batch normalization does a significant amount of the work for residual connections.

Table II shows the results with a constant batch normalization so the parameters Mean=0, Std=1, Scale=1, and Shift=0 are constant during the whole training process and also during the validation. Here it becomes clear that the maximum propagation generalizes best, which is especially the case for CIFAR100 and CIFAR10. Another interesting property of the constant batch normalization, is the reduction of the variance of the results for the maximum propagation. This clearly shows that the maximum propagation as well as the leaky maximum propagation learn different functions compared to residual connections. Of course, this also raises the question of why not use batch normalization. Here we would mention firstly the reduced robustness of networks caused by batch normalization [28], [29]. Since our approaches do not require batch normalization for generalization, we see this as an advantage of our approaches. Another point in favor of removing batch normalization is that it reduces the complexity of the networks and thus simplifies validation in future research.

Table III shows the results for different possible combinations of residual connections and maximum propagation. By

TABLE III

SHOWS THE CLASSIFICATION ACCURACY RESULTS FOR THE DATA SETS CIFAR10, CIFAR100, FASHIONMNIST, AND SVHN WITH DIFFERENT ARCHITECTURES AND **BATCH NORMALIZATION** AS AVERAGE OVER 5 RUNS WITH STANDARD DEVIATION AFTER $\pm$. THE TEXT IN THE BRACKETS AFTER THE ARCHITECTURE SPECIFIES THE PERFORMED COMBINATION OPERATION TO THE INPUT AND OUTPUT OF EACH BLOCK. FOR THIS EVALUATION WE FOCUSED ON DIFFERENT COMBINATION APPROACHES LIKE CONCATENATION AND THE USEAGE OF ALTERNATING BLOCKS IN THE SAME ARCHITECTURE. ALTERNATING IN THIS CONTEXT MEANS, ONE MAXIMUM PROPAGATION BLOCK IS ALWAYS FOLLOWED BY ONE RESIDUAL BLOCK.

Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0005, Learning rate=0.1, Batch size=100, Training time=100 epochs, Learning rate reduction after each 20 epochs by 0.1

Data augmentation: Shifting by up to 4 pixels in each direction and padding with zeros. Horizontal flipping for the CIFAR datasets only. Mean (Red=122, Green=117, Blue=104) subtraction and division by 256.

Method	CIFAR10	CIFAR100	FashionMNIST	SVHN
ResNet-34 (Leaky Max)	$\mathbf{93.88 \pm 0.09}$	$\mathbf{73.66 \pm 0.2}$	$\mathbf{95.05 \pm 0.14}$	$\mathbf{96.19 \pm 0.19}$
ResNet-34 (Concatenation)	92.56 ± 0.24	71.38 ± 0.54	91.52 ± 0.64	95.02 ± 0.37
ResNet-34 (Alternating)	93.04 ± 0.35	72.39 ± 0.4	95.06 ± 0.36	96.06 ± 0.1
ResNet-50 (Leaky Max)	$\mathbf{94.49 \pm 0.18}$	$\mathbf{74.91 \pm 0.4}$	$\mathbf{96.01 \pm 0.18}$	$\mathbf{96.95 \pm 0.21}$
ResNet-50 (Concatenation)	93.18 ± 0.2	72.65 ± 0.4	92.21 ± 0.21	95.63 ± 0.18
ResNet-50 (Alternating)	93.68 ± 0.21	73.67 ± 0.4	95.68 ± 0.21	96.75 ± 0.21

TABLE IV

SHOWS THE CLASSIFICATION ACCURACY RESULTS FOR THE DATA SETS CIFAR10, CIFAR100, FASHIONMNIST, AND SVHN WITH ENSAMBLES OF SIX RESNET-34 ARCHITECTURES. THE TEXT IN THE BRACKETS AFTER THE ARCHITECTURE SPECIFIES THE AMOUNT OF MODELS PER APPROCH IS IN THE ENSAMBLE (A=RESIDUAL BLOCKS OR ADDITON, M=MAXIMUM PROPAGATION, LM=LEAKY MAXIMUM PROPAGATION).

Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0005, Learning rate=0.1, Batch size=100, Training time=100 epochs, Learning rate reduction after each 20 epochs by 0.1

Data augmentation: Shifting by up to 4 pixels in each direction and padding with zeros. Horizontal flipping for the CIFAR datasets only. Mean (Red=122, Green=117, Blue=104) subtraction and division by 256.

Method	CIFAR10	CIFAR100	FashionMNIST	SVHN
ResNet-34 (6*A,0*M,0*LM)	93.64%	75.42%	95.18%	96.14%
ResNet-34 (0*A,6*M,0*LM)	94.97%	75.59%	95.46%	96.89%
ResNet-34 (0*A,0*M,6*LM)	94.44%	74.94%	95.19%	96.67%
ResNet-34 (3*A,3*M,0*LM)	95.36%	78.02%	95.61%	96.68%
ResNet-34 (3*A,0*M,3*LM)	94.65%	77.15%	95.49%	96.49%
ResNet-34 (0*A,3*M,3*LM)	95.39%	77.83%	95.81%	97.13%
ResNet-34 (2*A,2*M,2*LM)	**95.99%**	**78.55%**	**95.99%**	**97.96%**

TABLE V

SHOWS THE CLASSIFICATION ACCURACY RESULTS FOR THE DATA SETS CIFAR10, CIFAR100, FASHIONMNIST, AND SVHN WITH **BATCH NORMALIZATION AND NO ACTIVATION FUNCTION** AS AVERAGE OVER 5 RUNS WITH STANDARD DEVIATION AFTER $\pm$. THE TEXT IN THE BRACKETS AFTER THE ARCHITECTURE SPECIFIES THE PERFORMED COMBINATION OPERATION TO THE INPUT AND OUTPUT OF EACH BLOCK. THE USED BLOCKS CAN BE SEEN IN FIGURE 4.

Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0005, Learning rate=0.1, Batch size=100, Training time=100 epochs, Learning rate reduction after each 20 epochs by 0.1

Data augmentation: Shifting by up to 4 pixels in each direction and padding with zeros. Horizontal flipping for the CIFAR datasets only. Mean (Red=122, Green=117, Blue=104) subtraction and division by 256.

Method	CIFAR10	CIFAR100	FashionMNIST	SVHN
ResNet-34 (Additon)	nan	nan	nan	nan
ResNet-34 (Maximum)	$\mathbf{93.78 \pm 0.13}$	71.08 ± 2.29	94.82 ± 0.27	96.19 ± 0.11
ResNet-34 (Leaky Max)	87.69 ± 2.1	$\mathbf{74.04 \pm 0.25}$	$\mathbf{95.15 \pm 0.17}$	$\mathbf{96.39 \pm 0.16}$

"concatenation" we mean the concatenation of the addition and the maximum as shown in Figure 3 a). For these nets we used the inception model where each sub-block had only half the depth to get the same number of parameters as all other nets. By "alternating" we refer to the alternating use of residual blocks and maximum propagation as shown in Figure 3 b). Based on the results in Table III, it is evident that the leaky formulation is the most effective over all data sets.

Table IV shows the results for the formation of ensembles. Here, of course, it is important that networks learn different functions to benefit from the combination. This can be seen in the central section of Table IV in which almost all combinations are significantly better than combining the same nets (First section in Table IV). Of course, there is an exception here, which is the maximum propagation for the data set SVHN and CIFAR10. In these two cases, combining the same nets with the maximum propagation is slightly better in comaprison to some other combinations. In the last section in Table IV, all approaches are combined and also give the best results overall.

Figure 5 shows the loss (Solid line) and the accuracy (Dashed line) for the maximum propagation (Green) and residual connections (Red) over five runs as average value. As can be seen, the maximum propagation reduces the loss faster and therefore requires less time to learn. Note that after each 20 epochs the learning rate was reduced by 0.1. An exception here is the second segment (Epoch 20 to 40) for the CIFAR100 data set. This shows that the faster learning is not always the

TABLE VI

SHOWS THE MEAN ABSOLUTE DISTANCE FOR THE EYEBALL CENTER, EYEBALL RADIUS, AND OPTICAL VECTOR ON THE RESULTS FOR THE DATA SET 500KCLOSER. THE TEXT IN THE BRACKETS AFTER THE ARCHITECTURE SPECIFIES THE PERFORMED COMBINATION OPERATION TO THE INPUT AND OUTPUT OF EACH BLOCK.

Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0005, Learning rate=0.001, Batch size=10, Training time=100 epochs, Learning rate reduction after each 30 epochs by 0.1

Data augmentation: Random noise up to 20%. Random image shift of up to 20% of the image. Random image blurring with σ up to 1.5. Random image overlay with up to 30% of its intensity. Random color offset. Random color offset. Mean (Gray-scale=125) subtraction and division by 256.

Method	Center	Eyeball Radius	Optical Vector
ResNet-34 (Additon)	0.52 ± 2.20	$\mathbf{0.03 \pm 0.02}$	0.20 ± 0.10
ResNet-34 (Maximum)	0.66 ± 2.37	$\mathbf{0.03 \pm 0.02}$	$\mathbf{0.19 \pm 0.11}$
ResNet-34 (Leaky Maximum)	$\mathbf{0.52 \pm 2.10}$	0.04 ± 0.02	0.24 ± 0.12

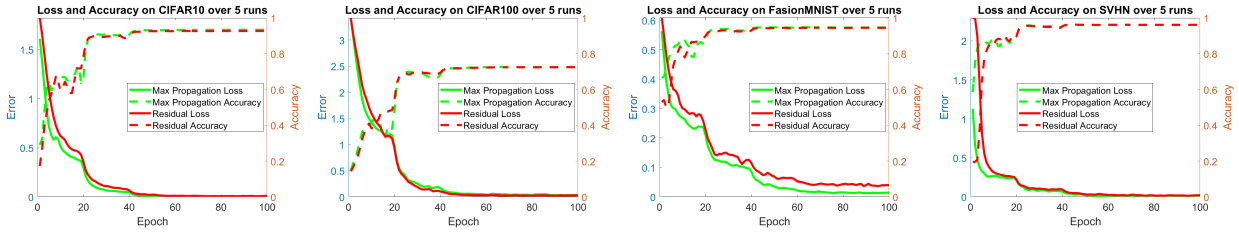

Fig. 5. Visual progress of the loss and accuracy as average over 5 runs for CIFAR10, CIFAR100, FashionMNIST, and SVHN. The solid lines are the loss values and the dashed lines are the accuracy. Red represents the values for the residual blocks and green the maximum propagation. The used architecture is ResNet-34 and every 20 epochs the learning rate was reduced by 0.1.

case and as described earlier, the maximum propagation and leaky maximum propagation can be seen as an alternative to residual connections but not as a replacement which works always better.

Table V shows the results without additional activation function in or after a residual, maximum, or leaky maximum connection. This is also visualized in Figure 4. For the residual connection this does not work since it ends up in not a number indicated by nan. For the maximum and leaky maximum propagation it still works but based on the random initialization of the network it can have a negative impact on the results as can be seen for CIFAR10 for the leaky maximum propagation as well as for CIFAR100 for the maximum propagation. In the following, we show that the presented maximum propagation and leaky maximum propagation also work for large data sets like ImageNet, and that they can also be used for semantic segmentation and regression.

TABLE VII

SHOWS THE CLASSIFICATION ACCURACY RESULTS FOR THE DATA SET ILSVRC2015 OR IMAGENET. THE TEXT IN THE BRACKETS AFTER THE ARCHITECTURE SPECIFIES THE PERFORMED COMBINATION OPERATION TO THE INPUT AND OUTPUT OF EACH BLOCK.

Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0001, Learning rate=0.1, Batch size=160, Training time=150 epochs, Learning rate reduction after each 50 epochs by 0.1

Data augmentation: Random cropping of 227×227 regions. Random flipping in each direction. Random color offset. Mean (Red=122, Green=117, Blue=104) subtraction and division by 256.

Method	Top-1	Top-5
ResNet-34 (Additon)	75.33%	92.43%
ResNet-34 (Maximum)	75.23%	92.31%
ResNet-34 (Leaky Maximum)	**75.55%**	**92.90%**

Table VII shows the Top-1 and Top-5 accuracy on ImageNet. The maximum propagation has the lowest results. The best results are obtained by the leaky maximum propagation. Since the training and evaluation on ImageNet consume a lot of time, we made only one training and evaluation run. Therefore, we think, that the results show that all three approaches are comparable since for each training and evaluation run the results change slightly as it is the case for all of the other data sets.

TABLE VIII

SHOWS THE PIXEL CLASSIFICATION ACCURACY OF THE SEMANTIC SEGMENTATION RESULTS FOR THE DATA SET VOC2012. THE TEXT IN THE BRACKETS AFTER THE ARCHITECTURE SPECIFIES THE PERFORMED COMBINATION OPERATION TO THE INPUT AND OUTPUT OF EACH BLOCK.

Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0005, Learning rate=0.1, Batch size=30, Training time=80 epochs, Learning rate reduction after each 40 epochs by 0.1

Data augmentation: Random cropping of 227×227 regions. Random flipping in each direction. Random color offset. Mean (Red=122, Green=117, Blue=104) subtraction and division by 256.

Method	Pixel Accuracy
U-Net (Additon)	85.15%
U-Net (Maximum)	85.34%
U-Net (Leaky Maximum)	**85.58%**

Table VIII shows the pixel accuracy on VOC2012 for the semantic segmentation task. As can be seen the maximum propagation and leaky maximum propagation work slightly better in comparison to the residual connection but since it is evaluated in a single run this cannot be seen as an evident result. We just want to show, that the proposed approaches work for semantic segmentation too.

Table VI shows the regression results on the 500kCloser data set. As can be seen the residual connections as well as the maximum and leaky maximum propagation deliver more

or less the same results. Since the data set is huge we also computed only a single run. Overall, Table VI shows, that the proposed approaches work for regression with competitive results to residual connections.

TABLE IX
SHOWS THE CLASSIFICATION ACCURACY RESULTS FOR THE DATA SET CIFAR100 WITH DIFFERENT LARGE DNN MODELS. THE **JOINTLY TRAINED ENSAMBLE (JTE)** CONSISTS OF THREE SUBNETWORKS WITH ADDITION, MAXIMUM PROPAGATION, AND LEAKY MAXIMUM PROPAGATION CONNECTIONS, WHICH ARE CONCATENATED. A DETAILED DESCRIPTION IS GIVEN IN THE SUPPLEMENTARY MATERIAL. THE NUMBER AFTER THE JOINTLY TRAINED ENSAMBLE STANDS FOR THE AMOUNT OF JTES WE HAVE COMBINED IN A MAJOR VOTING ENSAMBLE. THE DIFFERENCE BETWEEN JTE1 AND JTE2 IS ONLY THE DEPTH OF THE CONVOLUTIONS LAYERS WHICH OF COURSE ALSO EFFECTS THE SIZE OF THE CONCATENATED TENSOR BEFORE THE LAST FULLY CONNECTED LAYER.
Training parameters: Optimizer=SGD, Momentum=0.9, Weight Decay=0.0005, Learning rate=0.1, Batch size=50, Training time=150 epochs, Learning rate reduction after each 30 epochs by 0.1
Data augmentation: Shifting by up to 4 pixels in each direction and padding with zeros. Horizontal flipping and mean (Red=122, Green=117, Blue=104) subtraction as well as division by 256.

Method	CIFAR100	Params
ResNet-152 (Additon)	**76.17%**	60M
ResNet-152 (Maximum)	72.80%	60M
ResNet-152 (Leaky Max)	75.12%	60M
WideResNet-28-10 (Additon)	**78.04%**	36.5M
WideResNet-28-10 (Maximum)	74.57%	36.5M
WideResNet-28-10 (Leaky Max)	76.89%	36.5M
JTE1 $\times 1$	70.45%	1M
JTE1 $\times 3$	74.14%	3M
JTE1 $\times 6$	75.67%	6M
JTE1 $\times 9$	77.18%	9M
JTE2 $\times 1$	73.41%	3.5M
JTE2 $\times 3$	75.73%	10.5M
JTE2 $\times 6$	78.15%	21M
JTE2 $\times 9$	**79.89%**	31.5M

Table IX shows the results of the standard addition connections compared to the maximum and leaky maximum propagation. As can be clearly seen, the presented connections drastically reduce the accuracy of large models. This is especially true for the maximum propagation. In comparison, the ensambles of jointly trained ensambles achieve the same or even better results as the large networks with less parameters. This confirms the results in Table IV where the nets were trained independently and the individual nets were also significantly larger. A detailed architecture of the jointly trained ensambles can be found in our supplementary material.

IV. LIMITATIONS

A limitation of the use of the maximum propagation or the leaky maximum propagation is that the selection of the maximum requires an additional comparison in the implementation. Through this the calculation should be minimally slower than residual blocks. In the case of the leaky maximum propagation the multiplications with α and β are added. If however no activation functions are used, which permits the use of the maximum propagation, then the maximum propagation as well as the leaky maximum propagation should be minimally faster or equally fast to compute. In addition, the maximum and leaky maximum propagation, does not work as well as residual blocks for large networks but it outperforms the sole use of residual blocks in jointly trained ensambles with a similar sturucture as vision transformers.

V. CONCLUSION

In this work, we present an alternative to conventional residual connections which propagates the maximum value or a combination of the maximum and minimum value. We showed in several evaluations, that it is competitive to usually used residual blocks and can be effectively combined in neuronal network ensembles together with residual networks. In addition to the competitive results, the maximum propagation and leaky maximum propagation has some interesting properties. One is the better generalization with a fixed batch normalization (Or without) which should help in creating more robust networks since it has been shown, that batch normalization is vulnerable to adversarial attacks. Our proposed approaches can also be trained without additional activation function which could help in validation approaches since the network complexity reduces. This is especially true if the batch normalization is also removed. Another interesting property is that for most of the data sets, the proposed approaches learn faster in comparison to residual connections. Future research should investigate the robustness of maximum propagation and leaky maximum propagation blocks as well as the possibility of validating such networks. A further alternative is the usage of such blocks in architecture search.

REFERENCES

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] M. Hassaballah and A. I. Awad, *Deep learning in computer vision: principles and applications*. CRC Press, 2020.

[6] X. Lu, S. Li, and M. Fujimoto, "Automatic speech recognition," in *Speech-to-Speech Translation*. Springer, 2020, pp. 21–38.

[7] R. G. Bayrak, J. A. Salas, Y. Huo, and C. Chang, "A deep pattern recognition approach for inferring respiratory volume fluctuations from fmri data," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2020, pp. 428–436.

[8] D. Brodić and A. Amelio, "Human-computer interaction," in *The CAPTCHA: Perspectives and Challenges*. Springer, 2020, pp. 7–14.

[9] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, N. M. Akim, and M. Imran, "Deep learning and big data technologies for iot security," *Computer Communications*, vol. 151, pp. 495–517, 2020.

[10] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[11] Y. Yu and J.-M. Odobez, "Unsupervised representation learning for gaze estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7314–7324.

[12] R. Fan, H. Wang, P. Cai, J. Wu, J. Bocus, L. Qiao, and M. Liu, "Learning collision-free space detection from stereo images: Homography matrix brings better data augmentation," *IEEE/ASME Transactions on Mechatronics*, 2021.

[13] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4329–4341, 2020.

[14] T. Saba, A. S. Mohamed, M. El-Affendi, J. Amin, and M. Sharif, "Brain tumor detection using fusion of hand crafted and deep learning features," *Cognitive Systems Research*, vol. 59, pp. 221–230, 2020.

[15] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. Hoi, "Deep learning for person re-identification: A survey and outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[16] M. Popel, M. Tomkova, J. Tomek, Ł. Kaiser, J. Uszkoreit, O. Bojar, and Z. Žabokrtskỳ, "Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals," *Nature communications*, vol. 11, no. 1, pp. 1–15, 2020.

[17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[18] S. Hayou, A. Doucet, and J. Rousseau, "On the impact of the activation function on deep neural networks training," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2672–2680.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[21] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[22] G. E. Hinton, T. J. Sejnowski *et al.*, *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[23] M. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, 2012.

[24] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 968–977.

[25] B. Kulis *et al.*, "Metric learning: A survey," *Foundations and trends in machine learning*, vol. 5, no. 4, pp. 287–364, 2012.

[26] B. Goertzel and C. Pennachin, *Artificial general intelligence*. Springer, 2007, vol. 2.

[27] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International conference on machine learning*. PMLR, 2013, pp. 1319–1327.

[28] P. Benz, C. Zhang, A. Karjauv, and I. S. Kweon, "Revisiting batch normalization for improving corruption robustness," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 494–503.

[29] P. Benz, C. Zhang, and I. S. Kweon, "Batch normalization increases adversarial vulnerability: Disentangling usefulness and robustness of model features," *arXiv preprint arXiv:2010.03316*, 2020.

[30] A. Veit, M. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *arXiv preprint arXiv:1605.06431*, 2016.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[32] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[33] D. E. King, "Dlib-ml: A machine learning toolkit," *The Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[34] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[35] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[36] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[37] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[39] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012.

[40] W. Fuhl, W. Rosenstiel, and E. Kasneci, "500,000 images closer to eyelid and pupil segmentation," in *Computer Analysis of Images and Patterns, CAIP*, 11 2019.

[41] W. Fuhl, H. Gao, and E. Kasneci, "Neural networks for optical vector and eye ball parameter estimation," in *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.