# Consideration on application of the concept of Saak transform to convolutional neural networks

Tomonori Maeda, Kiyoshi Nishikawa*
Tokyo Metropolitan University, Hino-shi, Tokyo, Japan
* E-mail: knishikawa@m.ieice.org Tel: +81-42-585-8423

***Abstract*—In this paper, we consider applying the concept of Saak (Subspace approximation with augmented kernels) transform to the convolutional neural networks (CNNs). In CNNs, the activation function known as ReLU (Rectified linear unit) is widely used for image or signal processing applications, e.g., image classification, image super resolution, etc. Activation functions including ReLU discards negative values of the input to achieve nonlinear input-output relations. In CNNs, therefore, ReLU discards negative values of filter outputs although those negative values may have equal importance as positive values in image processing. The Saak transform is proposed to utilize the information carried by the negative values. In this paper, we consider the CNN architectures to utilize the concept of Saak transform by introducing a modified ReLU which discards positive values. Then we show that we can construct several CNN architectures based on the concept of Saak and the modified ReLU. To see the possibility of the proposed architecture, we apply them to the image classification and consider the validity from the results.**

## I. Introduction

In this paper, we consider applying the concept of Saak (Subspace approximation with augmented kernels) transform [1] to the convolutional neural network (CNN) architectures in order to utilize the negative correlations.

Deep learning systems based on artificial neural networks are expected to be applied to a wide variety of applications, such as self-driving cars [2], automatic translation of foreign languages [3], etc. Besides, CNN architectures are widely used in the applications in image processing area, such as image classification [4], image super resolution [5], etc. There are several widely known architectures based on CNN, e.g., LeNET [6], VGG16 [7], or ResNet [8].

When we apply CNNs to image or signal processing applications, the activation function known as 'ReLU' (Rectified linear unit) is widely used. The activation function is applied to the output of the filter unit, and when the value becomes negative, the output is truncated to be zero. This means that the negative outputs, or negative correlations, are not processed in the CNNs although they have the equal importance in the image or signal processing as the positive correlations. This may cause problems, such as the increased number of required parameters, or longer learning time, etc.

The Saak transform [1] is an attempt to utilize those truncated negative correlations. The Saak transform employs the Karhunen-Loève transform (KLT) [9] to reduce the amount of data. After the KLT, to utilize the negative correlations, it performs the extension of the bases in the negative direction by augmenting kernels. Then, the ReLU function is applied to discard the negative values. At this point, however, because the bases are extended in the negative direction before applying the ReLU, the negative correlations are also processed separately, and forwarded to the next process. Using the Saak in combination with support vector machine (SVM) has been shown to be effective for handwriting recognition [1] to reduce the required amount of calculation in comparison with the CNN-based architectures.

In this paper, we consider applying the concept of the Saak transform to the CNN architectures. Although the advantage of the Saak concept used with the SVM is shown, it may still have importance to apply the concept to the CNN architectures. Because, if we could utilize the concept to improve the learning characteristics, or to shorten the learning time, etc, we can apply it to the wide variety of applications where the advantage of the CNN architectures had been proven. We show that we can construct several CNN architectures to incorporate the Saak concept. As the base architecture, we choose the ResNet which is shown to be effective in the image processing applications. The modification of the ResNet to utilize the negative correlations are proposed and, through the computer simulations, we evaluate the validity of the proposed method.

## II. Related Works

### A. ReLU

To consider CNN architectures to apply the concept of Saak transform, we first review the ReLU, and then, define its variant.

The input-output relation of the standard ReLU, (we refer it as sReLU in the following), is expressed as

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

where $x$ is the input to the unit. We show the input-output relation of the sReLU in Fig. 1(a). We note that there are modified versions of the ReLU, such as Leakey ReLU, or Parametric ReLU [10].

In addition to the sReLU, we could define the negative ReLU, nReLU in the following, whose input-output

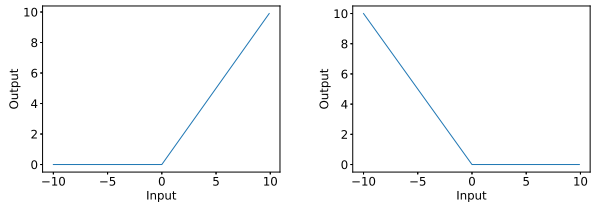
relation is given as below.

$$f(x) = \min(0, x) \tag{2}$$

In Fig. 1(b), we depict this relation.

In this paper, using these two types of ReLU, i.e., **sReLU** (standard ReLU) and **nReLU** (negative ReLU), we consider CNN architectures to utilize the information of negative correlations in addition to the positive ones. In those architectures, sReLU and nReLU are mixed, and the network are branched according to the positive and negative correlations as in the Saak transformation.

We show that various combinations could be constructed by the presence or absence of a branch in a block, and the merging methods after the branch. Then, we compare those CNN architectures.




(a) sReLU (b) nReLU

Fig. 1: Input-Output relations of the standard ReLU (sReLU), and negative ReLU (nReLU). Only positive values are processed and negative values are discarded by sReLU. nReLU, on the other hand, outputs negative values and discards positive values.

### B. Saak Transform

The Saak transform [1, 11] is derived based on the analysis of CNNs as the RECOS (REctified-Correlations on Sphere) transform [12]. It employs the KLT to obtain the eigenvectors of the correlation matrix of the input signal. Then, each kernel of this transformation is expanded in the negative direction (inverted sign) as kernel augmentation. This process is called sign-to-position format conversion (S/P Conversion).

Augmentation of the kernels in the negative direction is performed according to the following rules.

$$\mathbf{a}_{2k-1} = \mathbf{b}_k, \quad \mathbf{a}_{2k} = -\mathbf{b}_k \tag{3}$$

where $\mathbf{a}_k$ and $\mathbf{b}_k$ show the augmented kernels and transformation bases of the KLT respectively. In Fig. 2, we show a block diagram of the Saak transform.

In the following, based on the concept of the Saak, we consider extending the ResNet by using nReLU instead of kernel augmentation to utilize the negative correlations.

### C. ResNet

Generally, it is considered that advanced features could be extracted by using deeper neural networks. However, it is not an easy task to increase the number of layers because several difficulties arise as it increases, e.g., vanishing of gradient problem.

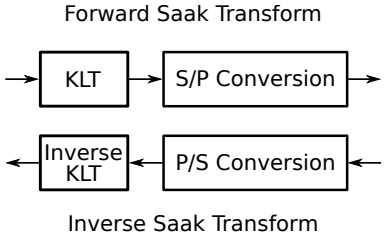



Fig. 2: Structure of the Saak transform

As an architecture that enables the construction of deeper networks, ResNet (Residual network) was proposed by He et al. [8] In the ResNet, layers learn using the residual by short-cutting the input signal and adding up with the outputs of the previous layer. Since the proposed method uses the ResNet to consider the application of the concept of the Saak transform, we describe a brief summary of the ResNet here.

*1) Residual Network:* A building block of the ResNet is called a Residual Block. The mathematical definition of Residual Block is as follows.

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, W_i) + \boldsymbol{x} \tag{4}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and the output vectors of the considered layers respectively; and $W_i$ show the weights to be determined.

The conventional neural networks update the weights in $\mathcal{H}(\boldsymbol{x}, W_i)$ so that the optimum output $\boldsymbol{y} = \mathcal{H}(\boldsymbol{x}, W_i)$ is obtained for the corresponding input. However, the ResNet defines the difference between the input and the output $\mathcal{F}(\boldsymbol{x}, W_i)$ as the following:

$$\mathcal{F}(\boldsymbol{x}, W_i) = \mathcal{H}(\boldsymbol{x}, W_i) - \boldsymbol{x}. \tag{5}$$

Then, the internal weights of $\mathcal{F}(\boldsymbol{x}, W_i)$ are adapted so that the optimum output $\boldsymbol{y} = \mathcal{H}(\boldsymbol{x}, W_i)$ is obtained by changing this equation as

$$\mathcal{H}(\boldsymbol{x}, W_i) = \mathcal{F}(\boldsymbol{x}, W_i) + \boldsymbol{x}. \tag{6}$$

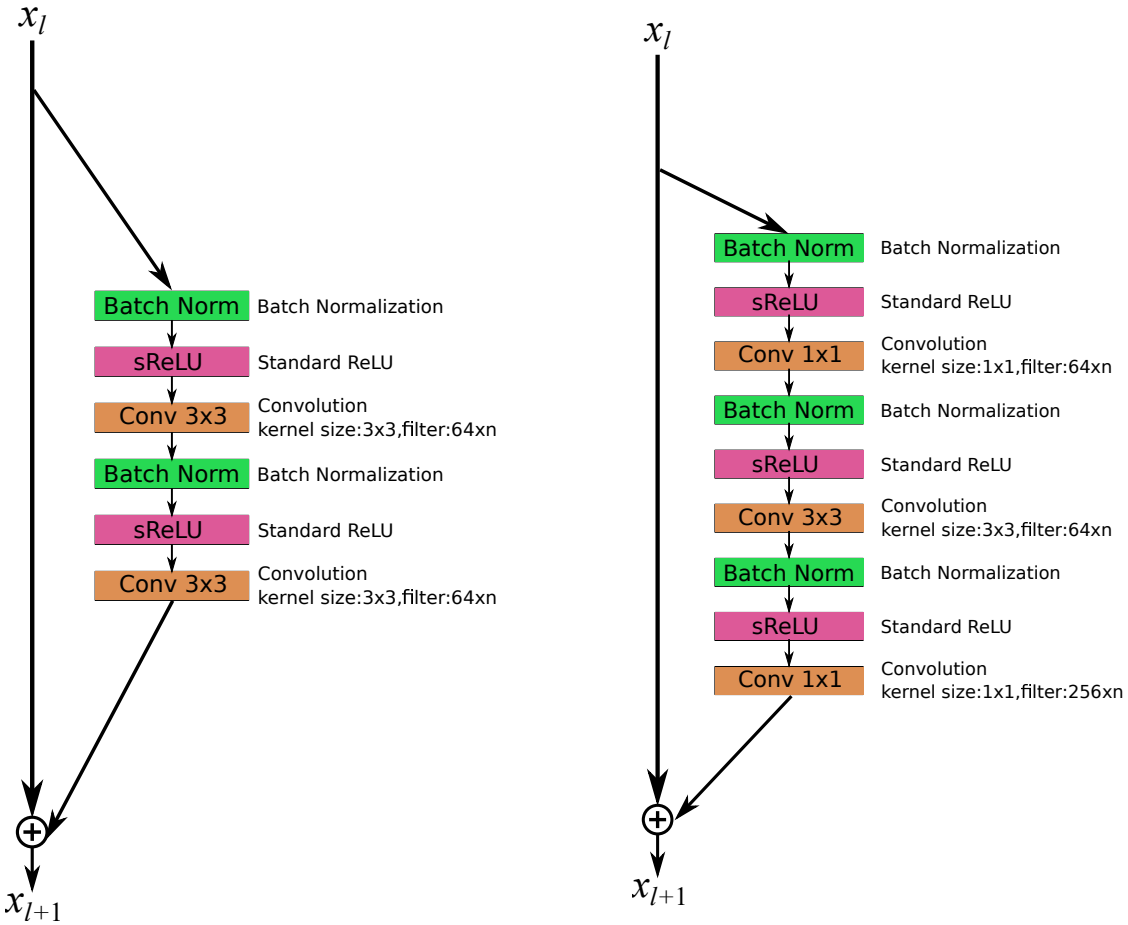
Note that we cannot define the addition when the dimensions of the output of $\mathcal{F}(\boldsymbol{x}, W_i)$ and the input $\boldsymbol{x}$ differ. Then, by introducing a transformation matrix $W_s$ that transforms the dimension of $\boldsymbol{x}$, equation (4) is modified as

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, W_i) + W_s\boldsymbol{x} \tag{7}$$

so that the dimensions of $\mathcal{F}(\boldsymbol{x}, W_i)$ and $W_s\boldsymbol{x}$ will become the same.

*2) Network Architecture:* There are two types of the Residual blocks, namely Basic and Bottleneck. Basic block has two convolutional layers as shown in Fig.3(a). On the other hand, the Bottleneck block has three convolutional layers as shown in Fig. 3(b).

By combining several Residual Blocks, different ResNet architectures can be constructed with different number of




(a)Basic Block (b)Bottleneck

Fig. 3: Construction of the Residual Blocks of ResNet. (a) Basic and (b) Bottleneck blocks are used in the conventional architecture. $n$ is stage of number of stage. It repeats this architecture 4 times.

layers. For example, ResNet 18 is an 18-layer network, which consists of 8 Basic blocks (16 layers), with the first convolutional layer and the final combined layer.

## III. Proposed Architecture

### A. Proposed architecture to implement the concept of Saak

Here, we consider applying the concept of the Saak transform to the ResNet. For that purpose, we first consider modifying the Residual blocks in Fig. 3 and branching the network to realize the functionality of nReLU.

The input-output relation of nReLU can be realized by inverting the input $x$ and then performing sReLU as shown in the following equation

$$y = \sigma(-x) \tag{8}$$

where $\sigma$ shows the output of sReLU. We can easily confirm that $y$ and the output of nReLU are identical.

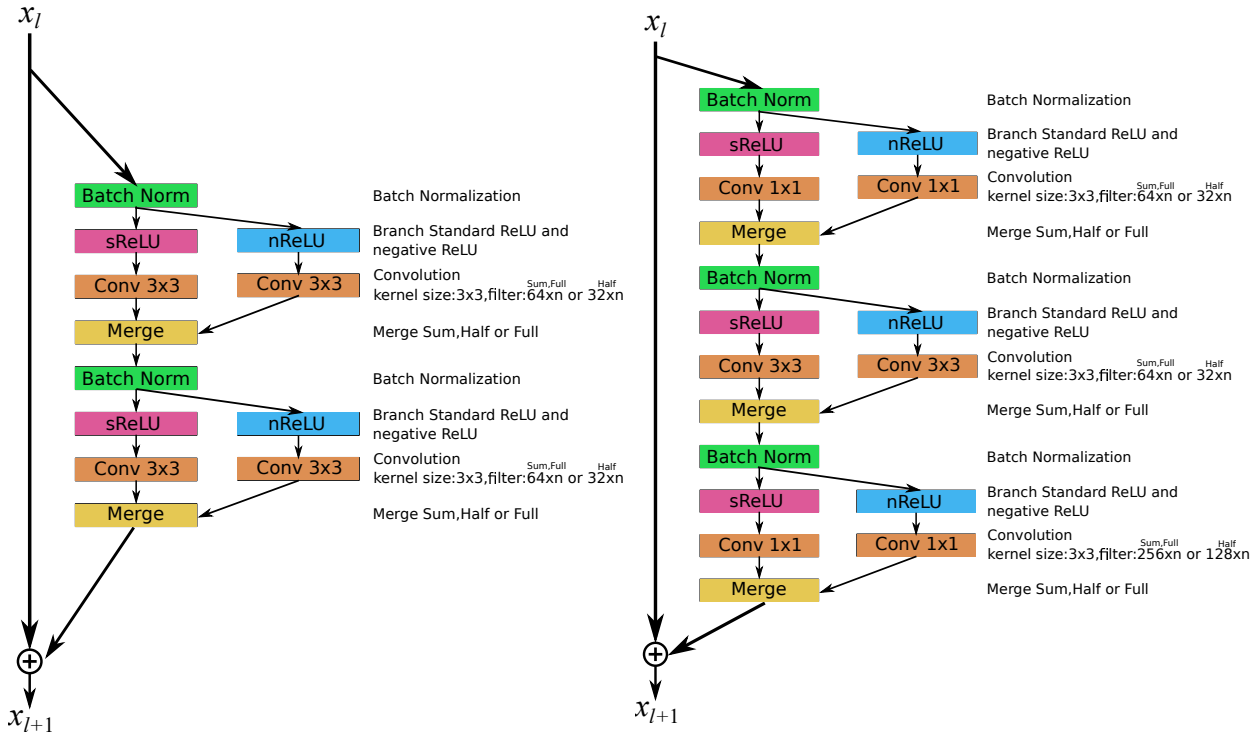
Here, we propose a modified Residual block, which we call the Dual Residual Block (DRB), in which the ReLU unit in Fig. 3 (a), (b) is branched as shown in Fig. 4 (a), (b) to utilize both the positive and the negative correlations.

In the conventional Residual block, only the rectification by sReLU exists after batch normalization. In contrast, in a DRB, we add a branch to realize the functionality of nReLU in which the negative rectification is performed. Hence, we have two paths to perform convolutions for calculating the positive and negative correlations, respectively.

The merging of positive and negative correlations after convolution is explained in section III-C.

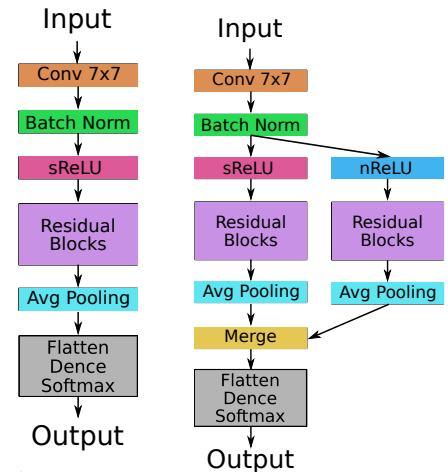
### B. Double ResNet

As mentioned before, the ResNet is constructed by combining several residual blocks. In the proposed method, we construct the network by using not only Basic and Bottleneck blocks but also Dual Basic and Dual Bottleneck blocks. Besides, we implement sReLU and nReLU as different branches as shown in Fig. 5. We call the network as **Double ResNet** when branching by sReLU and nReLU are used, and, otherwise, as **Single ResNet**.




(a)Dual Basic Block (b)Dual Bottleneck

Fig. 4: Construction of Residual Network proposed in this paper (a) Dual Basic and (b) Dual Bottleneck blocks are used in the conventional architecture. $n$ is stage of number of stage. It repeats this architecture 4 times like traditional ResNet.




Single ResNet Double ResNet

Fig. 5: Double ResNet is a network in which sReLU and nReLU are branched after batch normalization to form two parallel Residual Blocks.

### C. Merging Methods

In the proposed structure, the outputs of Dual Basic blocks should be merged to obtain the output of the blocks. For that, three merging methods are considered in the following.

The first method is called **Sum**, which adds the same components of the output of convolution. The second is called **Half**, which halves the number of filters in convolution and merges the output of nReLU at the end of the output of sReLU. The third is called **Full**, which does the same thing without reducing the number of filters of the second convolution in Half.

Let us explain these three methods using mathematical expression by defining the following variables.

$A[k]$: output of the unit $A$ in the $k$-th channel
$C^+[k]$: output of the $k$-th channel after convolution of sReLU
$C^-[k]$: output of the $k$-th channel after convolution of nReLU.

Using these variables, the result of merging using Sum can be expressed as

$$A[k] = C^+[k] + C^-[k]. \tag{9}$$

Besides, Half and Full have the following output:

$$A[k] = \begin{cases} C^+[k] & (k < n) \\ C^-[k-n] & (k \geqq n) \end{cases} \tag{10}$$

where $n$ shows the number of filters.

The number of filters $n$ is half of the conventional ResNet when 'Half' is used, and the same number of filters as ResNet when 'Full' is used.

| Mathematical notation | Network | Block | Merge | # params | Test accuracy[%] | Train Time [sec] |
|---|---|---|---|---|---|---|
| $B(3,3)$ | Single ResNet | Basic Block | — | 11.2M | 83.4 | 13978 |
| $B(3+3,3+3)$ | Single ResNet | Dual Basic | Sum | 22.1M | 82.9 | 24215 |
| $B(3\oplus 3,3\oplus 3)$ | Single ResNet | Dual Basic | Half | 11.2M | 83.0 | 16175 |
| $B(3\boxplus 3,3\boxplus 3)$ | Single ResNet | Dual Basic | Full | 44.5M | 83.4 | 43652 |
| $B(3,3)\boxplus B(3,3)$ | Double ResNet | Basic Block | — | 22.4M | 83.4 | 26275 |
| $B(3+3,3+3)\boxplus B(3+3,3+3)$ | Double ResNet | Dual Basic | Sum | 44.3M | 83.2 | 46375 |
| $B(3\oplus 3,3\oplus 3)\boxplus B(3\oplus 3,3\oplus 3)$ | Double ResNet | Dual Basic | Half | 22.4M | 83.2 | 30945 |
| $B(3\boxplus 3,3\boxplus 3)\boxplus B(3\boxplus 3,3\boxplus 3)$ | Double ResNet | Dual Basic | Full | 89.0M | 83.9 | 85529 |

TABLE I: Accuracy rate and learning time in Basic Block and its extended residual block

| Mathematical notation | Network | Block | Merge | # params | Test accuracy[%] | Train Time [sec] |
|---|---|---|---|---|---|---|
| $B(1,3,1)$ | Single ResNet | Bottleneck | — | 23.6M | 81.3 | 28309 |
| $B(1+1,3+3,1+1)$ | Single ResNet | Dual Bottleneck | Sum | 44.3M | 81.9 | 47144 |
| $B(1\oplus 1,3\oplus 3,1\oplus 1)$ | Single ResNet | Dual Bottleneck | Half | 24.6M | 82.6 | 36113 |
| $B(1\boxplus 1,3\boxplus 3,1\boxplus 1)$ | Single ResNet | Dual Bottleneck | Full | 93.9M | 83.2 | 80938 |
| $B(1,3,1)\boxplus B(1,3,1)$ | Double ResNet | Bottleneck | — | 47.2M | 82.1 | 53730 |
| $B(1+1,3+3,1+1)\boxplus B(1+1,3+3,1+1)$ | Double ResNet | Dual Bottleneck | Sum | 88.6M | 82.2 | 83026 |
| $B(1\oplus 1,3\oplus 3,1\oplus 1)\boxplus B(1\oplus 1,3\oplus 3,1\oplus 1)$ | Double ResNet | Dual Bottleneck | Half | 47.2M | 81.9 | 68962 |
| $B(1\boxplus 1,3\boxplus 3,1\boxplus 1)\boxplus B(1\boxplus 1,3\boxplus 3,1\boxplus 1)$ | Double ResNet | Dual Bottleneck | Full | 188M | 82.9 | 65555 |

TABLE II: Accuracy rate and learning time in Bottleneck and its extended residual block

### D. Network Combination

By selecting and combining the types of residual blocks and the merging methods, we can construct several architectures.

There are three points that we can select.

- Type of network $\cdots$ whether to use SingleResNet or DoubleResNet defined in Sec. III-B.
- Type of Residual block $\cdots$ Select from the 4 types of Residual blocks shown in Sec. III-A. That is, Basic, Bottleneck, Dual Basic, or Dual Bottleneck
- Merging method $\cdots$ The method for merging the outputs of Residual blocks can be selected, i.e., Sum, Half, or Full.

To show the combination of the selection, we introduce a rule for description as below.

Let $B(M)$ denotes a residual block structure where $M$ represents the kernel size in the convolution layer. For example, $B(3,3)$ represents a residual block structure formed of two $3\times 3$ convolution layers, and the ResNet Basic block is selected.

Then, we show the Sum, Half and Full of merging methods by $+$, $\oplus$, and $\boxplus$, respectively. For example, $B(3+3,3+3)$ indicates a ResNet structure in which a convolution layer with a kernel size of $3\times 3$ is branched by sReLU and nReLU and then merged by Sum method.

Other examples are expressed as below.

- $B(1,3,1)$ - Original Bottleneck Block
- $B(3\oplus 3,3\oplus 3)$ - Single ResNet in Dual Basic with merging Half
- $B(3,3)\boxplus B(3,3)$ - Double ResNet in Original Basic Block
- $B(3+3,3+3)\boxplus B(3+3,3+3)$ - Double ResNet in Dual Basic with merging Sum
- $B(1+1,3+3,1+1)$ - Single ResNet in Dual Bottleneck with merging Sum

## IV. Experimental Results

Here, we show the results of experiments of applying the proposed network architectures to Cifar-10 [13] image classification. It has 50,000 training and 10,000 testing data sets which are classified into 10 classes. Also, in order to increase the number of training in each epoch, we expanded the training data by image manipulations, i.e., rotation, horizontal, or vertical flipping, etc.

We evaluated the proposed networks in terms of the accuracy rate for test data and train time. Because the proposed Dual ResNet is a derivative of Basic and Bottleneck blocks, we evaluate them separately below.

### A. Experiment Environment

We used the following environment for the experiment.

- Computer:
  - CPU: Intel Xeon E5-1620v4
  - Memory: DDR4-2400 128GB RAM
  - GPU: NVIDIA GeForce GTX1080
  - OS: Ubuntu 18.04.2 LTS
- Software:
  - Python 3.7.3 [14]
  - TensorFlow 1.13.1 [15]
  - Keras 2.2.4 [16]

### B. Conditions

The architectures used in the experiments are based on ResNet18 (Basic Block) and ResNet50 (Bottleneck). The following architectures were compared:

- Basic and Dual Basic block
  - $B(3,3)$
  - $B(3+3,3+3)$
  - $B(3\oplus 3,3\oplus 3)$
  - $B(3\boxplus 3,3\boxplus 3)$
  - $B(3,3)\boxplus B(3,3)$
  - $B(3+3,3+3)\boxplus B(3+3,3+3)$

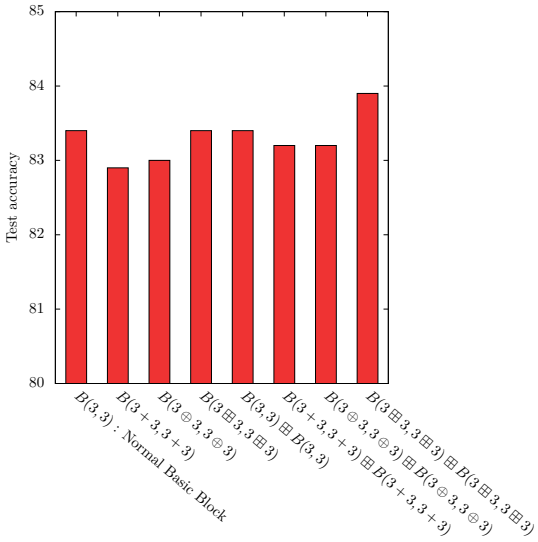



Fig. 6: Accuracy rate and train time in Basic Block and its extended residual block

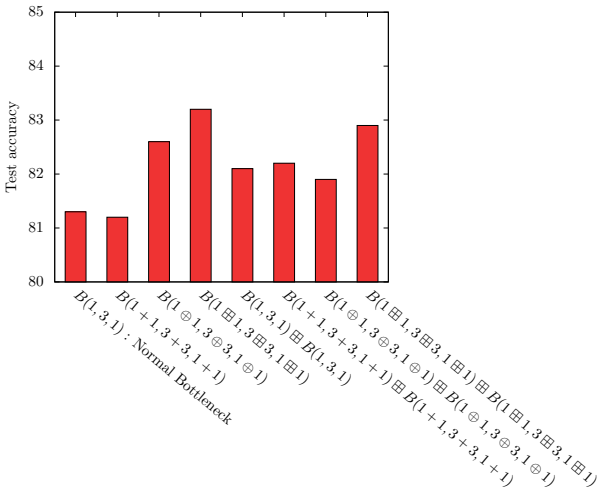



Fig. 7: Accuracy rate and train time in Bottleneck and its extended residual block

  - $B(3 \oplus 3, 3 \oplus 3) \boxplus B(3 \oplus 3, 3 \oplus 3)$
  - $B(3 \boxplus 3, 3 \boxplus 3) \boxplus B(3 \boxplus 3, 3 \boxplus 3)$

Figure 4: Accuracy rate and learning time of each model

- Bottleneck and Dual Bottleneck block
  - $B(1, 3, 1)$
  - $B(1 + 1, 3 + 3, 1 + 1)$
  - $B(1 \oplus 1, 3 \oplus 3, 1 \oplus 1)$
  - $B(1 \boxplus 1, 3 \boxplus 3, 1 \boxplus 1)$
  - $B(1, 3, 1) \boxplus B(1, 3, 1)$
  - $B(1 + 1, 3 + 3, 1 + 1) \boxplus B(1 + 1, 3 + 3, 1 + 1)$
  - $B(1 \oplus 1, 3 \oplus 3, 1 \oplus 1) \boxplus B(1 \oplus 1, 3 \oplus 3, 1 \oplus 1)$
  - $B(1 \boxplus 1, 3 \boxplus 3, 1 \boxplus 1) \boxplus B(1 \boxplus 1, 3 \boxplus 3, 1 \boxplus 1)$

Note that, among these architectures, $B(3, 3)$ and $B(1, 3, 1)$ correspond to the conventional ResNet18 and ResNet50, respectively.

In all the experiments, the number of epochs was 10 and the batch size was 32.

### C. Comparison of the merging methods

We applied the proposed architectures to the experiments using the Basic and the Bottleneck blocks with the three merging methods, i.e., Sum, Half and Full.

TABLES I and II show the results of the basic block and bottleneck. The accuracy rates are shown in Fig. 6 and Fig. 7.

From TABLE I and TABLE II, it is confirmed that the accuracy rate tends to become higher in the order 'Sum', 'Half', and 'Full' for the proposed structures except the Double ResNet case in TABLE II. Besides, we can see that from TABLE I, only the 'Double ResNet' with 'Full' merging achieves the higher accuracy rates than the conventional ResNet $B(3, 3)$. On the other hand, from TABLE II, the proposed architectures achieve the higher accuracy rate than the convention architecture $B(1, 3, 1)$.

### D. Comparison of Single and Double ResNet

Figure 5: Accuracy rate and learning time of each model

When Single ResNet and Double ResNet are compared in both Basic and Bottleneck blocks, there is a tendency for the accuracy rate to be higher when Double ResNet is used. However, it is also noticed that both the train times and the number of parameters were almost two times of those of the ResNet. This is due to the fact that Double ResNet is composed of two Residual Blocks in parallel.

### E. Train Time

Train Time shown in TABLE I and TABLE II are the total learning time for epochs (10 epochs). The shortest train time was achieved when $B(3, 3)$ was used which is the original Basic block. In view point of the train time, the proposed architectures requires the longer ones, because of the branching in the networks to implement sReLU and nReLU.

For the proposed architectures to be implemented in the actual applications, we need to consider more efficient implementation of the negative activation.

## V. Conclusions

In this paper, we proposed CNN architectures that split functionality of the ReLU used in ResNet into sReLU and nReLU and perform convolutions in parallel. The purpose of the proposed method is to utilize the negative correlations, which are discarded in the conventional architectures, in addition to the positive ones because both

correlations may contain equally important information in the image and signal processing applications. We applied the proposed architectures to Cifar-10 image classification for evaluating the performance, and it is shown that the architectures have an ability to achieve better accuracy rate than the conventional ResNet. However, for evaluating the true performance of the proposed architectures, we need to apply them to other applications and to consider adjusting the network structures. On the other hand, because of the cost of branching, the number of parameters and learning time have increased compared to the conventional architectures. As a future work, we would investigate more efficient architectures for achieving better results with less learning time.

## REFERENCES


[1] Yueru Chen, Zhuwei Xu, Shanshan Cai, Yujian Lang, and C. C. Jay Kuo. A saak transform approach to efficient, scalable and robust handwritten digits recognition, 2017.

[2] Zhilu Chen and Xinming Huang. End-to-end learning for lane keeping of self-driving cars. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1856–1860, June 2017.

[3] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] waifu2x. http://waifu2x.udp.jp/index.html.

[6] LeCun Yann, Haffner Patrick, Bottou Lenon, and Bengio Yoshua. Object recognition with gradient-based learning. In Forsyth Daivd., editor, *Feature Grouping*. Springer, 1999.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] Henry Stark and W. Woods John, editors. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.

[10] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *ArXiv*, abs/1505.00853, 2015.

[11] C. C. Jay Kuo and Yueru Chen. On data-driven saak transform, 2017.

[12] C. C. Jay. Kuo. The cnn as a guided multilayer recos transform [lecture notes]. *IEEE Signal Processing Magazine*, 34(3):81–89, May 2017.

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[14] Rossum Guido. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, and ... Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[16] François Chollet et al. Keras. https://keras.io, 2015.