A Trainable Multiplication Layer for Auto-correlation and Co-occurrence Extraction

Anonymous ACCV 2018 submission

Paper ID 42

Abstract. In this paper, we propose a trainable multiplication layer (TML) for a neural network that can be used to calculate the multiplication between the input features. Taking an image as an input, the TML raises each pixel value to the power of a weight and multiplies them, thereby extracting the higher-order local auto-correlation from the input image. The TML can also be used to extract co-occurrence from the feature map of a convolutional network. The training of the TML is formulated based on backpropagation with constrains to the weights, enabling us to learn discriminative multiplication patterns in an end-to-end manner. In the experiments, the characteristics of the TML are investigated by visualizing learned kernels and the corresponding output features. The applicability of the TML for classification and neural network interpretation was also evaluated using public datasets.

1 Introduction

Typified by the success of convolutional neural networks (CNNs) in recent computer vision research, studies using deep learning-based methods succeeded in various fields [1], [2], [3], [4], [5]. Different from the traditional machine learning techniques based on hand crafted features, such deep learning-based methods automatically extract features from raw input data via end-to-end network learning.

To take advantage of the end-to-end learning capability of deep neural networks, many studies have investigated to develop a network layer that represents a certain function by incorporating a model into a layer structure [6], [7], [8], [9]. For example, Wang *et al.* [7] proposed a trainable structural layer named a global Gaussian distribution embedding network, which involves global Gaussian [10] as an image representation.

In spite of such challenging efforts, most of the layer structures are based on inner products of the input features and the weight coefficients. There has been little work attempting to introduce multiplication of input features. Shih *et al.* [8] proposed a network layer that detects co-occurrence between the feature maps calculated from a CNN. In classical pattern recognition techniques, however, multiplication of input features is important because it represents auto-correlation or co-occurrence of the input features.

This paper proposes a trainable multiplication layer (TML) for a neural network that can be used to calculate the multiplication between the input

features. Taking an image as an input, the TML raises each pixel value to the power of a weight and multiplies them, thereby extracting the higher-order local auto-correlation from the input image. The TML can also be used to extract co-occurrence from the feature map of a convolutional network. The training of the TML is formulated based on backpropagation with constrains to the weights, enabling us to learn discriminative multiplication patterns in an end-to-end manner.

The contributions of this work are as follows:

- A trainable multiplication layer is proposed.
- The learning algorithm of the TML based on constrained optimization is formulated.
- Applicability to classification and network interpretation is shown via experiments.

2 Related work

2.1 Multiplication in a neural network layer

As most of the existing neural network layers focus on the multiplication of the input features and weight coefficients, there are only few studies about the network layer that calculates the multiplication of input features. Classically, Sigma-Pi Unit based on the summation of multiplication is applied to a self-organizing map [11], [12], [13]. Incorporation of probabilistic models such as a Gaussian mixture model into the neural network structure consequently leads to the development of a network layer based on multiplication [9], [14], [15].

Although such multiplication-based layers are not actively studied, they have potential to make an efficient network structure because they can express correlation and co-occurrence directly. This paper therefore focuses on the development of the multiplication-based neural network layer.

2.2 Higher-order local auto-correlation

Higher-order local auto-correlation (HLAC) is a feature extraction method proposed by Otsu [16]. In particular, HLAC is frequently used in the field of image analysis [17], [18], [19], [20].

For L displacements $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_L$, the Lth-order autocorrelation function is defined as

$$R_L(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_L) = \sum_D f(\boldsymbol{r}) f(\boldsymbol{r} + \boldsymbol{a}_1) \cdots f(\boldsymbol{r} + \boldsymbol{a}_L), \tag{1}$$

$$D = \{\boldsymbol{r} | \boldsymbol{r} + \boldsymbol{a}_l \in D, \forall l \in \{1, 2, \ldots, L\}\}, \tag{2}$$

where $f(\boldsymbol{r})$ is the pixel value of the input image at a coordinate $\boldsymbol{r}$, D is a set of coordinates of the input image.

In HLAC, the patterns of displacement should be prepared manually. The number of the displacement patterns explosively increases according to the mask size and the order; hence they are limited practically.

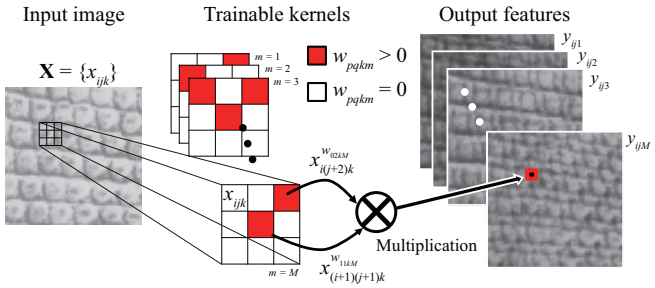


Fig. 1. Overview of the forward calculation conducted in the TML. This figure shows a simple case where the number of input channels K is 1 and the kernel size is 3×3.

3 Trainable multiplication layer

3.1 Layer structure

Fig. 1 illustrates the overview of the forward calculation conducted in the TML. The main idea behind this layer is to achieve multiplication of input values chosen by kernels, which is different from the well-known convolutional layer that conducts multiplication of input values and kernels.

Given an input image or feature map of a convolutional layer $\mathbf{X} \in \mathbb{R}^{N_1 \times N_2 \times K}$ (N_1, N_2, and K are the number of rows, columns, and channels, respectively), the forward pass of the proposed layer is defied as follows:

$$y_{ijm} = \prod_{k=0}^{K-1} \prod_{p=0}^{H-1} \prod_{q=0}^{W-1} x_{(i+p)(j+q)k}^{w_{pqkm}} \tag{3}$$

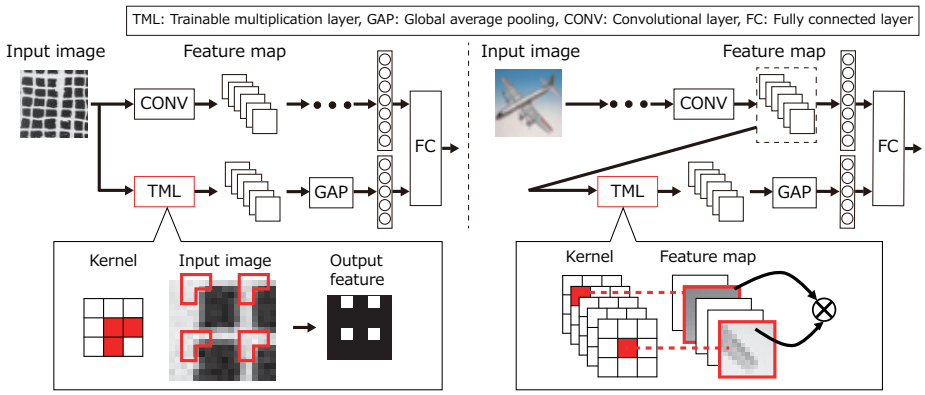
where x_{ijk} ($i = 0, \ldots, N_1 - 1, j = 0, \ldots, N_2 - 1, k = 0, \ldots, K - 1$) is the (i, j, k)-th element of $\mathbf{X}$ and $w_{pqkm} \geq 0$ is the (p, q)-th element of the m-the kernel for channel k ($p = 0, \ldots, H - 1, q = 0, \ldots, W - 1, m = 0, \ldots, M - 1$; M is the number of kernels). Since any number to the zero power is 1 (we define the value of zero to the power of zero is also 1), the forward pass of the proposed layer is regarded as the multiplication of input values where the value of kernel at the corresponding coordinate is greater than zero.

Practically, (3) is calculated in the logarithmic form to prevent under/overflow. Assuming that $x_{ijk} > 0$ since $\mathbf{X}$ is an image or a feature map passed through a ReLU function, (3) is rewritten as follows:

$$z_{ijmpqk} = \log\left(x_{(i+p)(j+q)k} + \epsilon\right), \tag{4}$$

$$y_{ijm} = \exp\left(\sum_{k=0}^{K-1} \sum_{p=0}^{H-1} \sum_{q=0}^{W-1} w_{pqkm} z_{ijmpqk}\right), \tag{5}$$

where $\epsilon > 0$ is a minute value to avoid $\log(0)$.



(a) Use as a discriminative higher-oder local auto-correlation extractor

(b) Use as a co-occurrence extractor

Fig. 2. Two usage methods of the TML in a CNN: (a) the use as a discriminative higher-order local auto-correlation (DHLAC) extractor and (b) the use as a co-occurrence extractor.

3.2 Usage of the layer in a convolutional neural network

The TML is supposed to be used as a part of a CNN-based deep neural network. Fig. 2 shows two different usage methods of the TML in a CNN. One is (a) the use as a discriminative higher-order local auto-correlation (DHLAC) extractor, and the other is (b) the use as a co-occurrence extractor.

The former usage method is motivated to achieve the discriminative learning of displacement patterns in HLAC. In this usage, the TML is inserted into just behind the input image. Output features calculated by the TML is then passed through global average pooling (GAP) [21]. GAP takes the average of each feature map and the resulting vector is fed into the next layer. This GAP is placed to simulate the integral computation of HLAC. Since the displacement patterns are represented by the power functions to be differentiable, they are trainable as kernel patterns.

The latter usage method is supposed to calculate the co-occurrence between feature maps. In the deeper layer of a CNN, feature maps calculated by a convolutional layer involve abstracted information of the input. In this usage, the TML finds out co-occurrence from such abstracted information by calculating the multiplication between feature maps. This calculation allows the extraction of non-Markovian co-occurrence that is useful for classification. Furthermore, we can use the TML to extract not only co-occurrence between features of a single network, but also co-occurrence between two or more networks.

3.3 Learning algorithm

Given a set of image $\mathbf{X}^{(n)}$ $(n = 1, \ldots, N)$ for training with the teacher vector $\boldsymbol{t}^{(n)}$, the training process of the network into which the TML is incorporated

Algorithm 1 Algorithm of the weight updating

Require: Parameters C_1, C_2, and λ, training image set $\mathbf{X}^{(n)}$, teacher vector $\boldsymbol{t}^{(n)}$.
Ensure: Trained network.
1: Initialize the kernels $\boldsymbol{w}$ and other network weights $\boldsymbol{\theta}$.
2: **while** $\boldsymbol{w}$ and $\boldsymbol{\theta}$ have not converged **do**
3: Calculate E.
4: Calculate gradients of E with respect to $\boldsymbol{w}$ and $\boldsymbol{\theta}$.
5: Update $\boldsymbol{w}$ and $\boldsymbol{\theta}$ using gradient-based updating.
6: $\boldsymbol{w} \leftarrow \mathrm{clip}(\boldsymbol{w}, 0, C_2)$
7: $\boldsymbol{w}_m \leftarrow C_1 \boldsymbol{w}_m / \mathrm{sum}(\boldsymbol{w}_m)$ for $m = 0, \ldots, M-1$
8: **end while**

involves minimizing the energy function E defined as

$$\text{minimize } E = \frac{1}{N} \sum_{n=1}^{N} \mathrm{Loss}(\boldsymbol{O}^{(n)}, \boldsymbol{t}^{(n)}) + \lambda \|\boldsymbol{w}\|_1 \tag{6}$$

$$\text{subject to } \sum_{k=0}^{K-1} \sum_{p=0}^{H-1} \sum_{q=0}^{W-1} w_{pqkm} = C_1, \quad \text{for } m = 0, \ldots, M-1, \tag{7}$$

$$0 \le w_{pqkm} \le C_2, \quad \text{for } p = 0, \ldots, H-1, q = 0, \ldots, W-1, \quad k = 0, \ldots, K-1, m = 0, \ldots, M-1, \tag{8}$$

where $\mathrm{Loss}(\boldsymbol{O}^{(n)}, \boldsymbol{t}^{(n)})$ is the loss function defined based on the final network output vector $\boldsymbol{O}^{(n)}$ corresponding to $\mathbf{X}^{(n)}$ and the teacher vector $\boldsymbol{t}^{(n)}$. $\| \cdot \|_1$ indicates the L_1 norm, $\boldsymbol{w}$ is all of the kernels shown in a vectorized form, and λ is a constant. The L_1 normalization is recruited expecting to obtain sparse kernels based on the concept of the TML. The first constraint shown in (7) fixes the total value of each kernel and prevents overflow. The second constraint shown in (8) keeps each kernel from being too sparse because each kernel should have two or more non-zero elements to be a meaningful kernel. It might seem that the L_1 normalization in (6) is redundant since the L_1 norm is fixed to a constant value due to both constraints. However, the L_1 normalization influences the gradient vectors to make a sparse solution during the network learning.

Algorithm 1 shows the procedure to solve the above optimization, where $\boldsymbol{w}_m$ is the vectorized m-th kernel. In this algorithm, gradient-based weight updating to decrease the energy function E and weight modification to keep the constraints are calculated alternately. Although this is an approximated approach different from the well-known Lagrange multiplier-based approach, we employ this approach to strictly satisfy the constraints during the network learning.

To calculate backpropagation, the partial derivative of E with respect to each kernel w_{pqkm} is required. Since the TML consists of multiplication and power

functions, the partial derivative is simply calculated as follows:

$$\begin{aligned}
&\frac{\partial E}{\partial w_{pqkm}} \\
&= \frac{\partial E}{\partial y_{ijm}} \frac{\partial y_{ijm}}{\partial w_{pqkm}} \\
&= \frac{\partial E}{\partial y_{ijm}} \prod_{k=0}^{K-1} \prod_{p=0}^{H-1} \prod_{q=0}^{W-1} x_{(i+p)(j+q)k}^{w_{pqkm}} \log x_{(i+p)(j+q)k} \\
&= \frac{\partial E}{\partial y_{ijm}} y_{ijm} \log x_{(i+p)(j+q)k}, \qquad (9)
\end{aligned}$$

where the form of $\frac{\partial E}{\partial y_{ijm}}$ depends on the layer connected between the TML and the output.

As outlined above, the TML incorporated in the deep network calculates DHALC of the input image or co-occurrence between the feature maps. The kernels are trained in the end-to-end manner via backpropagation with some constraints.

4 Investigation of the layer characteristics

Before going into classification experiments, we investigated the characteristics of the TML by training a relatively shallow network with the TML inserted, and visualizing learned kernels and corresponding feature maps.

4.1 Relationships between hyper parameters and learned kernels

The TML has some important hyper parameters, such as C_1, C_2 for learning constraints, and the kernel size. We investigate the changes in the learned kernels and the corresponding features according to the parameter variation.

In this trial, LeNet-5 [22] is used as the basic structure. The TML is inserted just behind the input. The Relu activation function [23] is used for convolution layers and the sigmoid function is used for the fully connected layer.

As the training data, we used the MNIST dataset [24]. This dataset comprised 10 classes of handwritten binary digit images with a size of 28×28, and contains 60,000 training images and 10,000 testing images. The network is trained with whole training data for each hyper parameter.

After network training, we visualized the learned kernels and the responses for the testing images. First, we observed the changes in learned kernels according to the parameters for constraints. Since the ratio of C_1 to C_2 will affect the training results, we varied the value of C_2 as $C_2 = 1.0, 0.5, 0.33, 0.25$ for fixed C_1 as 1.0. The kernel size and λ are set as 3×3 and $\lambda = 0.01$, respectively.

Fig. 3 shows the changes in learned kernels according to C_2. In this figure, values of the learned kernels normalized by C_2 were shown in gray scale heat map, and therefore black and white pixels show 0 and C_2, respectively. The

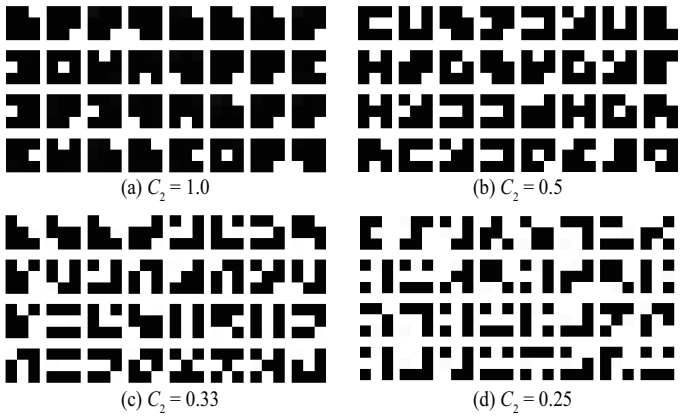


Fig. 3. Changes in learned kernels according to C_2. The kernel size and other parameters are fixed as 3×3, $C_1 = 1.0$, and $\lambda = 0.01$.

number of non-zero elements in each kernel increases according to the decrease of the value of C_2. Since the total value of elements in each kernel is fixed to C_1 and the upper limit of each element is suppressed by C_2, the number of non-zero elements approximates to C_1/C_2 if the L_1 normalization sufficiently worked. This results show that the number of non-zero elements, which is equivalent to the order in HLAC, can be controlled by changing the ratio of C_1 to C_2.

Second, the kernel size was varied as 3×3, 5×5, 7×7, and 9×9 for the fixed value of $C_1 = 1.0$, $C_2 = 0.5$, and $\lambda = 0.01$. Fig. 4 shows learned kernels and the corresponding output features according to the kernel size. The kernel pattern and the output feature pattern placed the same location correspond to each other. The learned kernel values shown in the left panels are also normalize in the same manner as Fig. 3. The values of the output features shown in the right panels are rescaled from [0, 1] to [0, 255] before visualization.

In terms of kernel size variation, the characteristics of learned kernels and the output features changed depending on the kernel size. In Fig. 4(a), the number of non-zero elements is two in each kernel and such elements adjoins each other in the most of kernels. Kernels with neighboring non-zero elements extract quite local correlation, and the output features are almost same as input images. However, kernels with non-zero elements apart from each other extract various output features according to the pattern of the kernel. In Fig. 4(b), the number of kernels with non-zero elements apart from each other increased and hence richer variation of output features was obtained than 4(a). In Fig. 4(c) and (d), there are some kernels that have gray pixels. This means that three or more values are multiplied in one calculation and therefore high-order local auto-correlation is extracted. The above results indicated that the number of non-zero elements in each kernel can be controlled to some extent by the ratio of C_1 to C_2, and the variety of output features changes according to the kernel size.

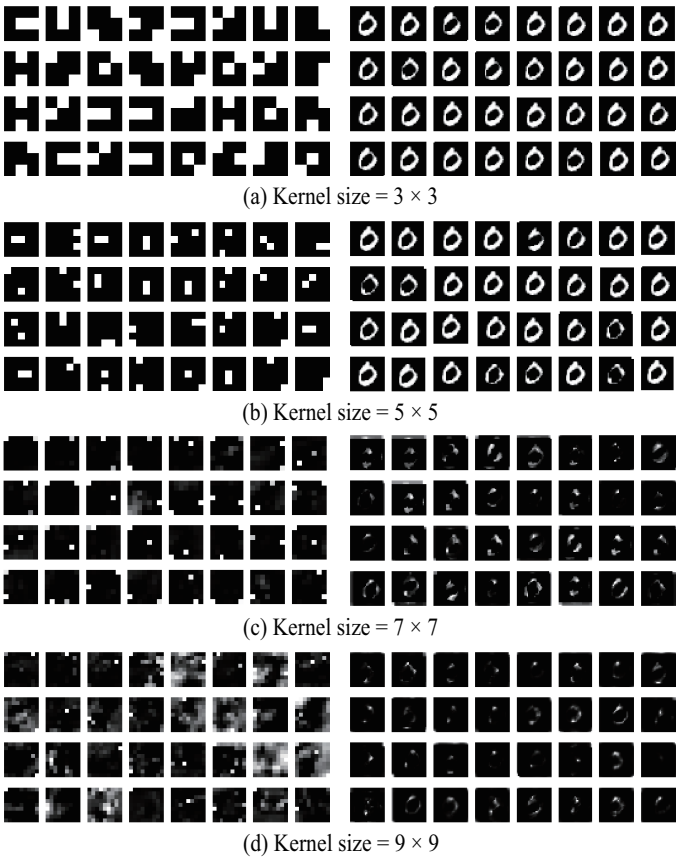


Fig. 4. Learned kernels (left panels) and the corresponding output features (right panels) for each kernel size. Other parameters are fixed as $C_1 = 1.0$, $C_2 = 0.5$, and $\lambda = 0.01$.

4.2 Characteristics as a DHLAC extractor

We verify whether the TML can make discriminative responses by observing the learned kernel patterns and the corresponding output features for the synthetic texture patterns.

The texture dataset used in this experiment contains six classes of 32×32 artificially generated images as following procedure (examples are shown on the leftmost panels in Fig. 5): First, six stripe patterns different for each class are drown on 1024×1024 black image. Second, uniform random noise in the range of [0, 1] is added to the images. Finally, a set of randomly cropped 32×32-sized images are treated as the dataset. We generated 100 training images for each class (600 training images in total). After training, we observed the learned kernels and the corresponding output features to the testing samples that is generated independently from training images.

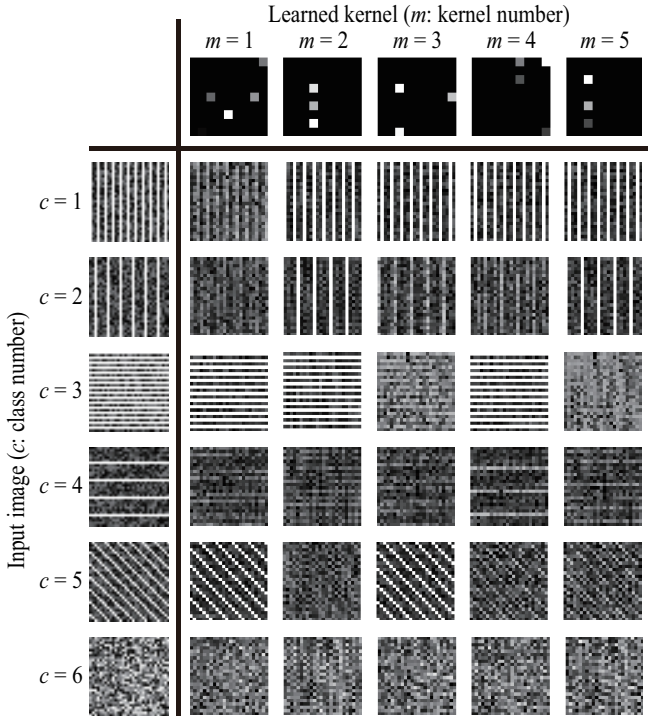


Fig. 5. Output features for each combination of the input image and the learned kernel.

The network structure used in this experiment is LeNet-5 as with Section 4.1. The parameters were set as $C_1 = 1.0$, $C_2 = 0.5$, $\lambda = 0.01$ and 9×9 for the kernel size.

Figure 5 shows the output features for each combination of the input image and the learned kernel. The row and column correspond to the class number and the kernels number, respectively. As with Fig. 4, the features are rescaled [0, 1] to [0, 255].

Fig. 5 shows that different output features are obtained according to the combination of the class and kernel. For example, for the input image of $c = 5$, slanting white stripe remains only in the output features from $m = 1$ and $m = 3$. This is because the kernels $m = 1$ and $m = 3$ have non-zero elements apart diagonally from each other. For other classes, the pattern remains in the output features if the direction or interval of the pattern correspond to those of the kernel patterns. This means that the TML learned kernels that can extract discriminative features from the input image.

4.3 Characteristics as a co-occurrence extractor

To investigate the capability of the TML for extracting co-occurrence features, we visualized the co-occurring regions detected by the TML.

Fig. 6. Visualization of the co-occurrence features. The top panels show the examples of the original MNIST images. The bottom panels show the corresponding co-occurrence features highlighted over the original images.

In the use as a co-occurrence extractor, the TML is connected to the fully connected layer as shown in Fig. 2(b). In this experiment, the TML is inserted between the second convolutional layer and the fully connected layer of the LeNet-5. Since this fully connected layer maps the input vector into the likelihood for each class, the weights of this layer represent the relevance between each dimension of the input vector and each class. Based on this fact, we extracted the most relevant kernel to the target category. We then defined the co-occurrence features as the input features to the TML that are activated by the kernel relevant to the target category.

Fig. 6 shows visualization results of the co-occurrence features. In this figure, it seems that distinctive features for each category are highlighted. For instance, bilateral lines of "0" are highlighted. In "7" the upper left edge and the vertical line are highlighted. These results shows the capability of the TML for extracting co-occurrence features that are useful for classification from CNN feature maps.

5 Classification experiments

To evaluate the applicability of the TML, we conducted classification experiment using public datasets.

5.1 Dataset

We used the following datasets in this experiment.

MNIST: As described in the previous section, this dataset includes 10 classes of handwritten binary digit images with a size of 28×28. We used 60,000 images as training data and 10,000 images as testing data.

Fashion-MNIST: Fashion-MNIST [25] includes 10 classes of binary fashion images with a size of 28×28. It includes 60,000 images for training data and 10,000 images for testing data.

CIFAR10: CIFAR10 [26] is labeled subsets of the 80 million tiny images dataset. This dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.

Kylberg: Kylberg texture dataset [27] contains unique texture patches for each class. We used its small subset provided by the author which includes 6

Table 1. Comparison of the recognition rates (%).

	MNIST	Fashion	CIFAR10	Kylberg	Brodatz
Baseline CNN	99.27	92.44	81.43	99.12	91.69
Baseline CNN + HLAC	99.31	92.09	**82.23**	99.31	91.33
Baseline CNN + TML (DHLAC)	**99.39**	92.45	81.50	99.02	**92.51**
Baseline CNN + TML (Co-occurrence)	99.27	**92.54**	81.49	**99.35**	91.56

classes with 40 samples each. We divided original 576×576 patched into 9 64×64 non-overlapping patches and treated each patch as one sample; thus 2,160 samples are available in total. We conducted 10-fold cross validation and calculated classification accuracy.

Brodatz: The Brodatz texture dataset [28] contains 112 texture classes with a size of 640×640. We divided each image into 100 64×64 non-overlapping patches, and treated each patch as one sample; thus 11,200 samples are available in total. We conducted 10-fold cross validation to calculate accuracy.

5.2 Experimental setup

As the baseline, we used a simple CNN (called as "baseline CNN" hereinafter) to clarify the effect of the TML. The baseline CNN consists of five convolutional layers with four max pooling layers between them and two fully connected layers (the structure is illustrated in the supplemental file). In the network, dropout was conducted after the first and second max pooling and the first fully connected layer.

The effectiveness of the TML was then examined by connecting the TML to the baseline CNN, both as a DHLAC extractor and a co-occurrence extractor. The kernel size of the TML was set as 5×5. We also compared with the results using HLAC instead of the TML.

5.3 Results

Table 1 shows the recognition rates for each dataset. In all the datasets, applying the TML to the CNN improved the classification performance. In particular, use as DHLAC showed remarkable performance for the Brodatz. This is because texture is based on repeated structure and hence auto-correlation information is effective for the classification, and discriminative training of HLAC features conducted by the TML worked efficiently. These results showed the applicability of the TML for classification.

6 Applications

6.1 Interpretation of the network

In this section, we demonstrate that the TML can be used to interpret which part in the input image the CNN focuses on. In the usage as a co-occurrence

Fig. 7. Visualization of the network interpretation results using the TML. The areas having strong influence for classification extracted by the TML are highlighted.

extractor, the TML is connected between the feature maps and the fully connected layer (Fig. 2(b)). By tracing this structure in the reverse direction in the trained network, it is possible to extract features having strong influence for classification. It should be emphasized that the purpose of this experiment is not to improve classification accuracy, but to improve interpretability.

Specifically, we interpret a trained network in the following procedure: 1. Perform forward calculation for a certain input image; 2. Calculate the largest weight among the weights of the fully connected layer between the unit which outputs the posterior probability of the target class and the features calculated by the TML; 3. Extract a kernel of the TML connecting the largest weight calculated in the previous step; 4. Extract CNN feature maps connecting to non-zero elements of the kernel; and 5. Visualize the extracted feature maps by up-sampling them to the size of the input image and over-lapping them on the input image. The property of the TML that the learned kernel acquires sparsity allows this calculation.

We used the Caltech-UCSD Birds-200-2011 dataset (CUB-200-2011) [29]. Each species of birds has unique characteristics such as feather patterns and body shapes. This fact makes the interpretation of visualization results easer. This dataset contains 200 species of bird images, and consists of 5,994 images for training data and 5,794 images for test data. Although this dataset is often used for fine-grained classification and segmentation, we use this dataset just for visualization in this experiment.

The basic network structure used in this experiment is the LeNet-5. The TML is inserted between the second convolutional layer and the fully connected layer. The parameters were set as $C_1 = 1.0$, $C_2 = 0.5$, $\lambda = 0.01$ and 1×1 for the kernel size.

Fig. 10 shows examples of visualization results. In Fig. 10, images of the same species are arranged in each column. The feather patterns or body shapes unique for each species are highlighted. For example, on the far left panels, yellowish

green patterns on the head and the wing are highlighted. These results indicates the applicability of the proposed TML for interpreting a neural network.

6.2 Co-occurrence extraction between two networks for multi-modal data

In this experiment, we apply the TML to multi-modal data classification. As mentioned in Section 3.2, the TML can also be used to extract co-occurrence between two networks. By extracting the co-occurrence between features of two CNNs that take input data from different modalities, it is expected that two networks complement each other.

The problem we tackle in this experiment is tumor detection in magnetic resonance (MR) images. In MR imaging, there are some different modalities depending on a setting of pulse sequences. Since each modality has a different response, extracting co-occurrence between multiple modalities has a possibility to improve tumor detection accuracy.

We prepared a MR image dataset containing two modalities (Flair and T1c) from the multimodal brain tumor image segmentation benchmark (BraTS) [30]. This dataset consists of 3D brain MR images with tumor annotation. We made 2D axial (transverse) slice images and separated them into tumor and non-tumor classes based on the annotation. The dataset contains 220 subjects, and we randomly divided them into 154, 22, and 44 for training, validation, and testing, respectively. Since about 60 images were extracted from each subject, we obtained 8,980, 1,448, and 2,458 images for training, validation, and testing, respectively. We resized the images from 240×240 pixels to 224×224 to fit the networks' input size.

Fig. 8 illustrates the network architecture used in this experiment. The network is constructed based on VGG-16 [31]. VGG-16 has three fully connected layers following five blocks consisting of some convolutional layers and a max pooling layer. We applied the TML to CNN features after the b-th max pooling layer from the top. For comparison, we calculated classification accuracy for a single VGG-16 with each modality and two VGG-16's concatenated at the first fully connected layer.

Table 2 shows the results of the MR image classification. It is confirmed that the TML is effective for improving classification performance. In particular, the improvement is largest for $b = 3$. One possible explanation of this is that the information necessary for classification is extracted to the extent that position information is not lost in the middle of the network, and extracting co-occurrence among such information is effective for multi-modal data classification. This result shows the effectiveness of co-occurrence extraction using the TML.

7 Conclusion

In this paper, we proposed a trainable multiplication layer (TML) for a neural network that can be used to calculate the multiplication between the input

Table 2. Comparison of the classification accuracies for the MRI dataset.

Method	Modality	Accuracy (%)
Single VGG	Flair	94.22
Single VGG	T1c	89.86
Concatenated two VGGs	Flair and T1c	94.47
Ours ($b = 1$)	Flair and T1c	95.16
Ours ($b = 2$)	Flair and T1c	95.57
Ours ($b = 3$)	Flair and T1c	**96.14**
Ours ($b = 4$)	Flair and T1c	95.85
Ours ($b = 5$)	Flair and T1c	95.77

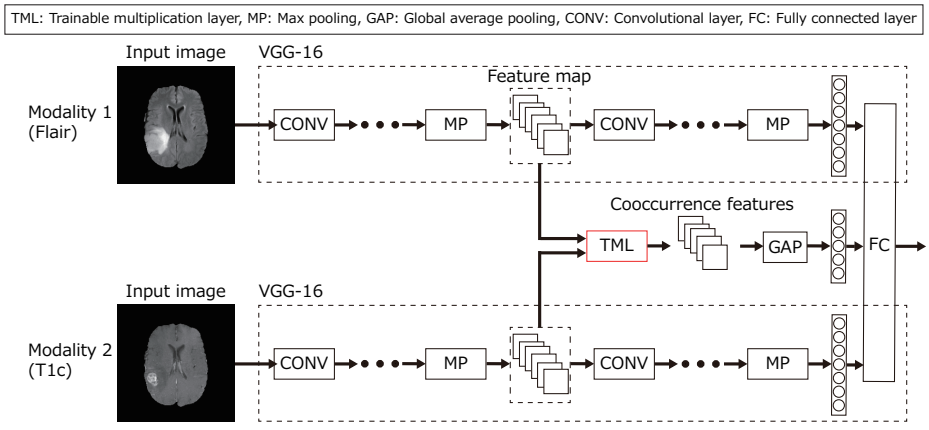


Fig. 8. Network architecture for MRI classification. Each VGG-16 model takes different modality's images as input. Co-occurrence features between two networks are extracted by the proposed TML.

features. Taking an image as an input, the TML raises each pixel value to the power of a weight and multiplies them, thereby extracting the higher-order local auto-correlation from the input image. The TML can also be used to extract co-occurrence from the feature map of a convolutional network. The training of the TML is formulated based on backpropagation with constrains to the weights, enabling us to learn discriminative multiplication patterns in an end-to-end manner. In the experiments, the characteristics of the TML were investigated by visualizing learned kernels and the corresponding output features. The applicability of the TML for the classification was also evaluated using public datasets. Applications such as network interpretation and co-occurrence extraction between two neural networks were also demonstrated.

In future work, we plan to investigate practical applications of the proposed TML. We will also expand the layer to the 3D structure in the same way as cubic HLAC [32]. Application to the time-series data analysis is also expected by constructing 1D structure.

References

1. Ma, L., Lu, Z., Li, H.: Learning to answer questions from image using convolutional neural network. In: AAAI. Volume 3. (2016) 16
2. Greenspan, H., van Ginneken, B., Summers, R.M.: Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. IEEE Transactions on Medical Imaging **35** (2016) 1153–1159
3. Zhang, T., Kahn, G., Levine, S., Abbeel, P.: Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In: IEEE International Conference on Robotics and Automation (ICRA), IEEE (2016) 528–535
4. Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al.: Deep speech 2: End-to-end speech recognition in english and mandarin. In: International Conference on Machine Learning. (2016) 173–182
5. Majumder, N., Poria, S., Gelbukh, A., Cambria, E.: Deep learning-based document modeling for personality detection from text. IEEE Intelligent Systems **32** (2017) 74–79
6. Oliva, J.B., Sutherland, D.J., Póczos, B., Schneider, J.: Deep mean maps. arXiv preprint arXiv:1511.04150 (2015)
7. Wang, Q., Li, P., Zhang, L.: G2denet: Global gaussian distribution embedding network and its application to visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 2730–2739
8. Shih, Y.F., Yeh, Y.M., Lin, Y.Y., Weng, M.F., Lu, Y.C., Chuang, Y.Y.: Deep co-occurrence feature learning for visual object recognition. In: Proc. Conf. Computer Vision and Pattern Recognition. (2017)
9. Hayashi, H., Shibanoki, T., Shima, K., Kurita, Y., Tsuji, T.: A recurrent probabilistic neural network with dimensionality reduction based on time-series discriminant component analysis. IEEE transactions on neural networks and learning systems **26** (2015) 3021–3033
10. Nakayama, H., Harada, T., Kuniyoshi, Y.: Global gaussian approach for scene categorization using information geometry. In: Conference on Computer Vision and Pattern Recognition, IEEE (2010) 2336–2343
11. Weber, C., Wermter, S.: A self-organizing map of sigma–pi units. Neurocomputing **70** (2007) 2552–2560
12. Valle-Lisboa, J.C., Reali, F., Anastasía, H., Mizraji, E.: Elman topology with sigma–pi units: An application to the modeling of verbal hallucinations in schizophrenia. Neural networks **18** (2005) 863–877
13. Courrieu, P.: Solving time of least square systems in sigma-pi unit networks. arXiv preprint arXiv:0804.4808 (2008)
14. Tsuji, T., Fukuda, O., Ichinobe, H., Kaneko, M.: A log-linearized gaussian mixture network and its application to eeg pattern classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **29** (1999) 60–72
15. Tsuji, T., Bu, N., Fukuda, O., Kaneko, M.: A recurrent log-linearized gaussian mixture network. IEEE Transactions on Neural Networks **14** (2003) 304–316
16. Otsu, N., Kurita, T.: A new scheme for practical flexible and intelligent vision systems. In: MVA. (1988) 431–435
17. Uehara, K., Sakanashi, H., Nosato, H., Murakawa, M., Miyamoto, H., Nakamura, R.: Object detection of satellite images using multi-channel higher-order local autocorrelation. arXiv preprint arXiv:1707.09099 (2017)

18. Fujino, K., Mitani, Y., Fujita, Y., Hamamoto, Y., Sakaida, I.: Liver cirrhosis classification on m-mode ultrasound images by higher-order local auto-correlation features. Journal of Medical and Bioengineering Vol **3** (2014)
19. Nosato, H., Sakanashi, H., Takahashi, E., Murakawa, M.: An objective evaluation method of ulcerative colitis with optical colonoscopy images based on higher order local auto-correlation features. In: IEEE 11th International Symposium on Biomedical Imaging, IEEE (2014) 89–92
20. Hu, E., Nosato, H., Sakanashi, H., Murakawa, M.: A modified anomaly detection method for capsule endoscopy images using non-linear color conversion and higher-order local auto-correlation (hlac). In: Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE, IEEE (2013) 5477–5480
21. Lin, M., Chen, Q., Yan, S.: Network in network. CoRR **abs/1312.4400** (2013)
22. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86** (1998) 2278–2324
23. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). (2010) 807–814
24. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
25. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)
26. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. (2009)
27. Kylberg, G.: The kylberg texture dataset v. 1.0. External report (Blue series) 35, Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden (2011)
28. Brodatz, P.: Textures: a photographic album for artists and designers. Dover Pubns (1966)
29. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Technical report (2011)
30. Menze, B.H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., et al.: The multimodal brain tumor image segmentation benchmark (brats). IEEE Transactions on Medical Imaging **34** (2015) 1993
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations. (2015)
32. Kobayashi, T., Otsu, N.: Action and simultaneous multiple-person identification using cubic higher-order local auto-correlation. In: Proceedings of the 17th International Conference on Pattern Recognition. Volume 4., IEEE (2004) 741–744

Supplemental materials

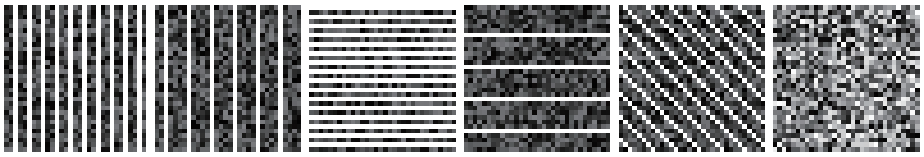


Fig. 9. Examples of texture images used for the layer response observation

The patterns of each class are as follows:

Class 1: Vertical white stripe with two pixels of intervals
Class 2: Vertical white stripe with four pixels of intervals
Class 3: Horizontal white stripe with one pixel of intervals
Class 4: Horizontal white stripe with six pixels of intervals
Class 5: Slanting white stripe with two pixels of intervals
Class 6: Nothing

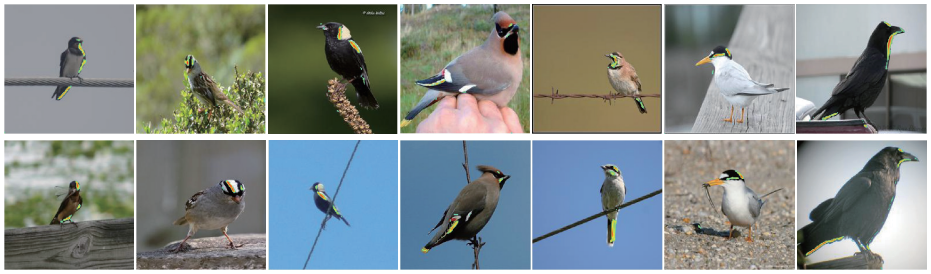


Fig. 10. Other results of the network interpretation using the TML.

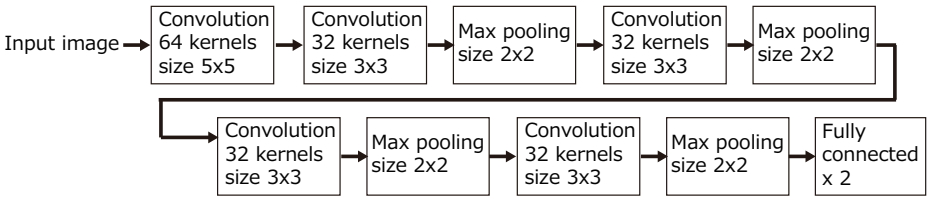


Fig. 11. Structure of the baseline CNN.