How Does a CNN Manage Different Printing Types?

Shota Ide, Seiichi Uchida
Kyushu University, Fukuoka, Japan 819–0395. email: uchida@ait.kyushu-u.ac.jp

***Abstract*—In past OCR research, different OCR engines are used for different printing types, i.e., machine-printed characters, handwritten characters, and decorated fonts. A recent research, however, reveals that convolutional neural networks (CNN) can realize a universal OCR, which can deal with any printing types without pre-classification into individual types. In this paper, we analyze how CNN for universal OCR manage the different printing types. More specifically, we try to find where a handwritten character of a class and a machine-printed character of the same class are "fused" in CNN. For analysis, we use two different approaches. The first approach is statistical analysis for detecting the CNN units which are sensitive (or insensitive) to type difference. The second approach is network-based visualization of pattern distribution in each layer. Both analyses suggest the same trend that types are not fully fused in convolutional layers but the distributions of the same class from different types become closer in upper layers.**

I. Introduction

Universal OCR is a character recognition system that can recognize arbitrary printing types of characters. Figure 1 shows digit images in three different printing types, machine printed (MP), handwritten (HW), and multiple fonts (MF), and a universal OCR needs to recognize all of them. In past research (e.g., [1]), characters are pre-classified into individual printing types and then recognized by a type-specific OCR engine. In contrast, universal OCR recognizes characters without pre-classification and thus requires high classification performance to deal with type variations.

In a recent paper [2], it is proven that a convolutional neural network (CNN) easily realizes a universal OCR if it is trained with a sufficient number training samples from all printing types. Figure 2 shows a typical CNN, called LeNet [3], which was also used in [2]. In the experimental setup of the literature [2], recognition accuracy by LeNet was 99.69% for a mixture of MP, HW, and MF and 99.91% for a mixture of MP and HW.

The purpose of this paper is to conduct deeper analyses of the universal OCR realized by CNN. More specifically, we will analyze skeptically how a CNN manages different printing types in it while achieving a very high recognition accuracy. The analyses will be worthy from the following viewpoints. First, we can understand how a CNN can deal with pattern distributions which are more complex than typical ones due to existence of multiple types. Second, we can observe whether there is some synergy among multiple types for training a CNN. Since many successful results by transfer learning have been reported, it might be possible that MP helps recognition of HW and vice versa. Third, our analysis results give a hypothesis for understanding how human beings perceive characters with different printing styles, since CNN somehow mimics human neural networks and also achieves human-level recognition performance.

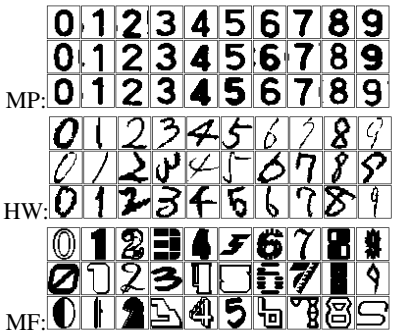

Fig. 1. Digit image samples in three different printing types. From top to bottom, machine printed (MP), handwritten (HW), and multiple fonts (MF).

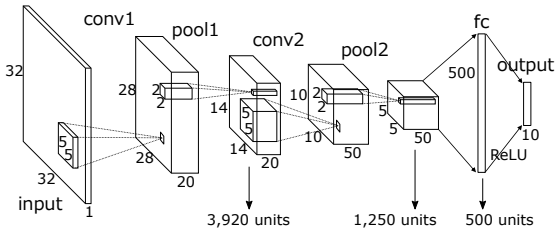

Fig. 2. A typical CNN (LeNet).

A key idea for our analyses is *type-fusion* at each layer. As shown in the bottom of Fig. 3, output values from units at the k-th layer comprise a vector $\mathbf{x}^k, k = 1, 2, 3$. Type-fusion is defined as the overlap between the distribution of $\mathbf{x}^k$ for a type and that of another type. If types are fused (i.e., the distributions of individual types are overlapped with each other) at a layer, it suggests that CNN for universal OCR minimizes the difference by the printing types there for, probably, being robust to the difference. Since the above almost perfect accuracy by the universal OCR proves that types are fused in CNN, our main concern is *how and which layers* are fusing the different printing types.

We can have two hypotheses about type-fusion in CNN, as shown in Fig. 3.

- The first hypothesis is that types are fused gradually through the layers of CNN, like Fig. 3 (a). The distributions of MP-"3" and HW-"3" are fused in a deeper layer even though they are not overlapped in an earlier layer. A version of this hypothesis is abrupt fusion at a certain layer instead of gradual fusion.
- The second hypothesis is that types are not fused in CNN like Fig. 3 (b). CNN treats MP-"3" and HW-"3"

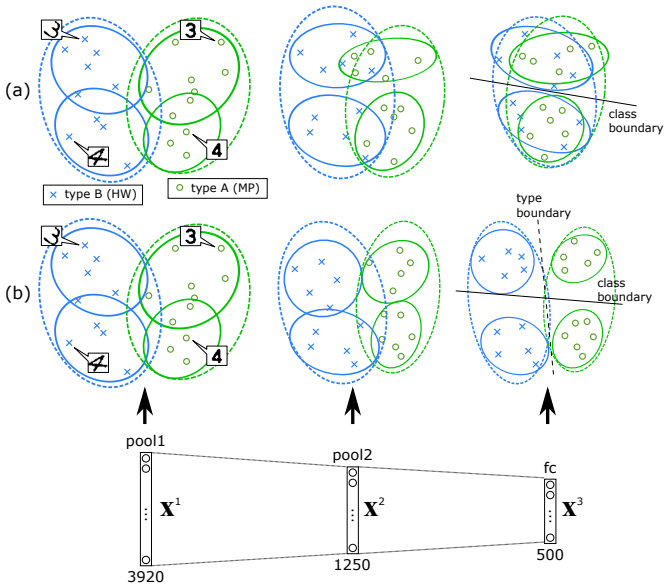

Fig. 3. Are different printing types fused in CNN? — (a) A hypothesis that types are gradually fused in deeper layers. (b) A hypothesis that types are not fused even in deeper layers. Each dot or cross represents an output vector from each layer ($\mathbf{x}^k, k = 1, 2, 3$) to an input. Note that classes should become separable in both cases because it is known that the universal OCR by CNN achieves a high recognition accuracy.

as different classes until `fc`. Then, CNN will finally identify MP-"3" and HW-"3" as the same "3" at the discrimination step just before the output layer.

We will see that our analyses mostly support the second hypothesis — no type-fusion is made in CNN.

For type-fusion analysis, we try two different approaches, statistical approach and network-based visualization approach. In the statistical approach, we classify each unit into two classes, fusing unit and non-fusing unit. Roughly speaking, if a unit is insensitive (sensitive) to type difference, it is a fusing (non-fusing) unit. If fusing units dominate at a layer, types are fused at the layer. Like [9], this unit classification is formulated as a regression problem with sparse regularization.

In the network-based visualization approach, we visualize the distribution of input patterns at each layer. In CNNs, an input pattern is converted into different feature vectors at different layers. We, therefore, observe the feature vector distribution at every layer. If distributions of MP, HW, and MF are overlapped at a layer, types are fused there. For visualizing the distribution of high-dimensional feature vectors, we use relative-neighborhood graph (RNG) representation.

A. Related work

Needless to say, CNNs are a very important tool for pattern recognition research. Character recognition have been a good target of CNNs from their inception [3], [4]. CNNs also have significant impacts on scene text recognition tasks [5], [6]. A recent paper [2] showed that a CNN can recognize machine-printed digits, handwritten digits, and multi-font digits with near perfect accuracies, given $1 \sim 100$ thousands of training samples per class. In the paper, it is also shown a CNN can realize a universal OCR which recognizes digit images in arbitrary printing types.

Besides these performance improvements by CNN, several "skeptic" research have also started to understand the behavior of CNNs. In fact, as a recent activity, "Workshop on Visualization for Deep Learning" was held in conjunction with ICML2016. We cannot refer to all the papers in the workshop here and readers can visit their homepage[1]. The first research direction for behavior understanding of CNN is to find difficult samples for CNNs. Adversarial images [7] and Fooling images [8] are famous trials. The second direction is to use some statistical approach to measure an internal reaction of a CNN. In [9], units which react to a specific visual attributes are extracted by regression framework.

The third and the most common direction to understand the behavior of CNNs is visualization. Deconvolutional neural networks [10] are a traditional way for visualizing an internal reaction of CNNs to input images. A recent trend is so-called deep visualization, such as [11], which also generates synthetic images explaining reaction of units. DeCAF [12] visualizes the distribution of input images at each layer in a projection-based strategy.

This paper focuses behavior of CNNs for universal OCR. Specifically, we want to reveal how a CNN manages difference of printing types inside it. For this purpose, we applied the method of [9] to detect units that are sensitive or insensitive to type difference. Furthermore, we propose a novel method for visualizing the distribution of patterns at individual CNN layers. The method utilizes relative-neighborhood graph (RNG) [13], which is very suitable for analyze how samples from different types are neighboring to each other in the feature space at a layer. This method has a strong merit that, it does not damage actual neighboring relationship, whereas DeCAF [12] damages the relationship due to the projection into a lower-dimensional space.

B. Contributions

Our main contributions are summarized as follows.

1) We analyze how a single CNN recognizes digit images in various printing types and prove experimentally that the CNN does not fuse the types in its internal feature spaces. For example, the distribution of machine-printed "3" is never mixed up with that of handwritten "3" in any layer of a CNN.
2) For the above type-fusion analysis, we employ a statistical approach and a visualization approach and conclusions drawn by both approaches coincide with each other.
3) More importantly, the visualization method is technically novel and can be utilized for analyzing any recognition task by CNN. The method utilizes RNG and gives an exact representation of the neighboring relationship among patterns.

[1] `http://icmlviz.github.io/home/`

II. Digit Datasets with Different Printing Types

We use three original digit datasets with different printing types, i.e., MP, HW, and MF. Samples shown in Fig. 1 were actual examples. Differences between MP and MF are that MF contains more designed (even fancy) fonts. In addition, MP contains scanned images and MF does born-digital. All digit images are 32×32 pixels and binary. MP, HW, and MF contain 822,714, 622,678, and 66,470 samples, respectively. Note that MP and HW are almost 10-times larger than MNIST. Each sample of MF has a different font style. MF, therefore, is comprised of 6,647 different fonts.

III. Convolutional Neural Network

As the CNN, LeNet shown in Fig. 2 is used for all experiments in this paper. More precisely, it is the version implemented in *Caffe* and slightly different from the original LeNet-5 [3] at ReLU and max-pooling as its activation function and subsampling, respectively. The network is initialized with random values (i.e., without any pre-training) and then trained with back-propagation for 30 epochs.

For realizing a universal OCR engine, a CNN is trained by combining the three datasets in various ways. Specifically, the CNN is trained by a couple of datasets (MP+HW, MP+MF, HW+MF) or all datasets (MP+HW+MF). In any case, 90% samples of each dataset is randomly selected and used for training.

To check the performance of the universal OCR by the CNN, its recognition accuracy was evaluated by using the remaining 10% samples [2]. In case of MP+HW and MP+HW+MF, 99.91% and 99.69% were achieved, respectively. When CNN is trained by a single type, its accuracy was 99.99%, 99.88%, and 95.7% for MP, HW, and MF, respectively. This result suggests that no significant degradation is caused by dealing multiple types in a single CNN-based OCR engine and thus CNN has a sufficient capacity for realizing a universal OCR.

IV. Type-fusion analysis by statistical approach

A. Methodology

Figure 4 illustrates the idea of the statistical approach for type-fusion analysis. For each input pattern, each unit in a trained CNN outputs some value. In the figure, the output x_a from the a-th unit has different values for different types and x_b from the b-th unit has no difference. Hereafter, we call a unit like a a *non-fusing unit* and a unit like b a *fusing unit*. If a layer in a trained CNN is dominated by fusing units, types are fused at the layer.

Let $\mathbf{x}_i$ denote the high-dimensional vector comprised of the output values from all units for the i-th input pattern ($i \in [1, N]$). In this paper, we consider the three layers, `pool1`, `pool2`, and `fc`, in LeNet of Fig. 2 and thus the dimensionality of $\mathbf{x}_i$ becomes $3,920 + 1,250 + 500 = 5,670$. In addition, $l_i \in \{0, 1\}$ denote the class of its printing type; note that in our statistical approach, only two types can be assumed, whereas more types can be assumed in the network-based visualization approach in Section V.

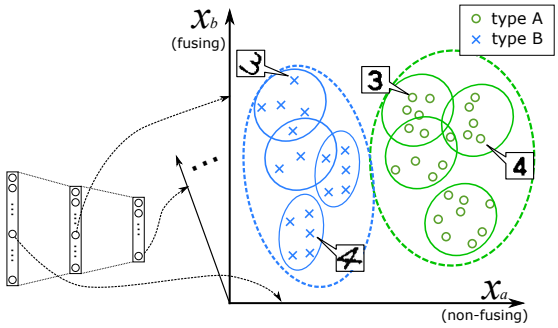

Fig. 4. Type-fusion analysis using statistical approach.

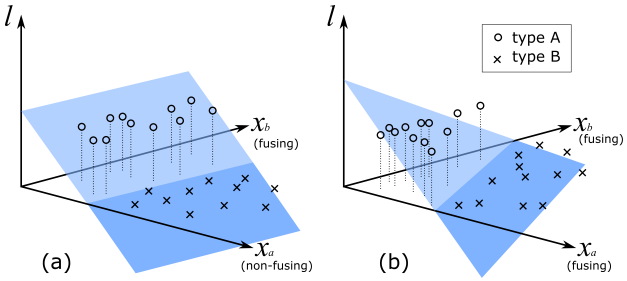

Fig. 5. (a) x_a correspond to a non-fusing unit whereas x_b does to a fusing unit. (b) Both of x_a and x_b correspond to fusing units.

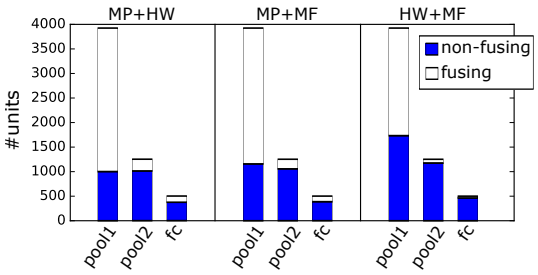

Fig. 6. The number of non-fusing and fusing units at each layer. If fusing units dominate a layer, types are fused there.

According to [9], we classify all units into fusing units and non-fusing units by regression framework. The basic idea of our linear regression problem is to find $\mathbf{w}$ that satisfies $l_i \sim \mathbf{w}^T \mathbf{x}_i$ for all i. Let w_m denote the m-th elements of $\mathbf{w}$. If x_b corresponds to a fusing unit, w_b will be zero so that x_b should not give a large effect in the regression problem between $\mathbf{x}_i$ and its type l_i. Hereafter if $w_m = 0$, its corresponding unit is considered as a fusing unit. Otherwise, a unit is considered as a non-fusing unit.

As shown in Fig. 5, the regression problem can be interpreted as a two-class classification problem to discriminate two types by a linear classifier specified by $\mathbf{w}$. If two types are clearly distinguishable on x_a like Fig. 5 (a), the discrimination boundary will be perpendicular to the axis of x_a. Consequently, w_a tends to be non-zero and x_a corresponds to a non-fusing unit. On the other hand, w_b tends to be zero and x_b corresponds to a fusing unit. In the case of Fig. 5 (b), both of x_a and x_b correspond to fusing units.

The regression problem is formulated as follows with

elastic-net regularization:

$$\min_{\mathbf{w}} \frac{1}{2N} \sum_{i=1}^{N} \|l_i - \mathbf{w}^T \mathbf{x}_i\|_2^2 + \mu \left\{ \alpha \|\mathbf{w}\|_1 + (1-\alpha)\frac{1}{2}\|\mathbf{w}\|_2^2 \right\}, \tag{1}$$

where μ and α are non-negative constant. Elastic-net regularization is employed for a sparse solution of $\mathbf{w}$, i.e., $\mathbf{w}$ with many zero elements, which correspond to fusing units.

B. Result of Type-Fusion Analysis

Figure 6 shows type-fusion analysis results at $\mu = 0.1$ and $\alpha = 0.1$. Since the analysis method is designed for two-class problems, three results are shown for all type pairs, i.e., "MP+HW", "MP+MF", and "HW+MF." In all cases, 90% of the entire samples were used for training the CNN. For evaluating the degree of type-fusion, 5,000 samples randomly chosen from the remaining 10% of each type were used as input patterns $\{\mathbf{x}_i\}$. Namely, we use the same number of samples for each type for a fair evaluation. Note that we performed many experiments with different parameter values of μ and α, but they result in almost the same conclusion.

All of those results in Fig. 6 suggest that the number of fusing units becomes less at deeper layers (`pool2` and `fc`). From these results, it is possible to say that types are not fused in the CNN. In addition, it is even possible to say that the CNN tries to remove "original" overlaps. Namely, input patterns from a certain type are overlapped with those from another type in their distribution, the overlap is dissolved in deeper layers. It is also interesting to note that there are few fusing units in the `fc` layers. This means types are clearly separated in after the `fc` layer. Considering that CNNs always give very high recognition accuracies, we can say the distributions of 20 classes (= two types $\times$ 10 digit classes) are well-separated by the CNN.

V. Type-fusion analysis by Network-Based Visualization Approach

A. Methodology

In this approach, we will visualize the distribution of feature vectors at each layer like Fig. 3 and then observe how the distributions from different printing types are overlapped or separated. This approach, therefore, is a more direct way to analyze type-fusion than the previous approach. Let $\mathbf{x}_i^1$, $\mathbf{x}_i^2$, and $\mathbf{x}_i^3$ denote feature vectors, each of which is comprised of the outputs from the units at `pool1`, `pool2`, and `fc`, respectively, for the i-th input pattern. For `pool1`, N feature vectors, $\{\mathbf{x}_1^1, \ldots, \mathbf{x}_N^1\}$, are distributed in the 3,920-dimensional space. If the feature vectors of MP, HW, and MF are overlapped in their distribution, they are fused in `pool1`.

Since it is impossible to visualize the distribution of high-dimensional vectors directly, we use an RNG-based visualization at each layer. RNG is similar to nearest neighbor graph but different. Nearest neighbor graph is a directed graph, where two nodes are connected by a directed edge if one node is the nearest neighbor of the other node. In contrast, RNG is an undirected graph, where two nodes are connected by an undirected edge if there is no other node between them, as shown in Fig. 7. Thus, RNG can represent bi-directional neighboring relationship among nodes in a more intuitive way[2]. RNG is a connected graph and has been utilized for visualizing pattern distribution (e.g., [13], [14]).

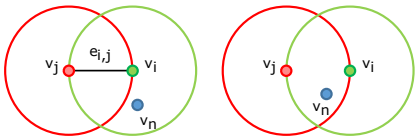

Fig. 7. Condition of edge assignment in RNG. Left: Two nodes (v_i, v_j) are connected. Right: They are not connected by the existence of v_n.

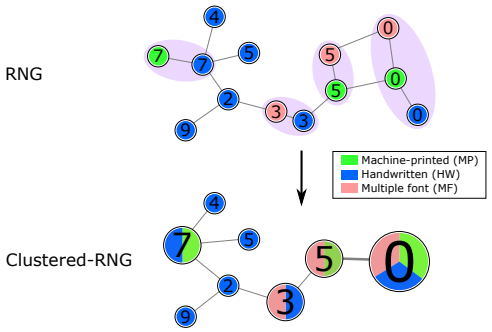

Fig. 8. Clustered-RNG [13] customized for type-fusion analysis.

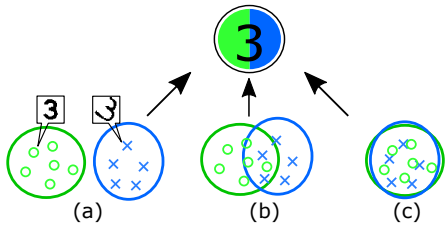

Fig. 9. In clustered-RNG, a cluster node with multiple types does not always mean that types are fused. Types are not fused in (a); however, it results in the same cluster node representation as (b) and (c), where types are fused.

RNG for the k-th layer ($k = 1, 2, 3$) is denoted as an undirected graph $\mathcal{G}^k = (\mathbf{V}^k, \mathbf{E}^k)$, where $\mathbf{V}^k$ is the set of N nodes, $\mathbf{V}^k = \{v_1^k, \ldots, v_i^k, \ldots, v_N^k\}$, and v_i^k corresponds to the vector $\mathbf{x}_i^k$ of the i-th input pattern. As noted above, a pair of nodes (v_i^k, v_j^k) are connected by an edge if there is no other node between them. More formally, (v_i^k, v_j^k) are connected by an edge $e_{i,j}^k$ iff the condition

$$\|\mathbf{x}_i^k, \mathbf{x}_j^k\| \leq \max(\|\mathbf{x}_i^k, \mathbf{x}_n^k\|, \|\mathbf{x}_n^k, \mathbf{x}_j^k\|) \tag{2}$$

is satisfied for all $n \in [1, N], n \neq i, n \neq j$.

Since it is still not feasible to visualize an RNG for large N, we use *clustered-RNG* [13] instead. Clustered-RNG is a coarse visualization of RNG for patterns with class labels. Figure 8 illustrates how to build a clustered-RNG from an RNG. If RNG has a connected subgraph whose nodes have the same class label, they are united into one node, called *cluster node*, in the clustered-RNG. When visualizing clustered-RNG, the

[2]If two nodes (v_i, v_j) are connected in a nearest neighbor graph, they are also connected in RNG. Therefore, the nearest neighbor graph is a subgraph of RNG, if we ignore the edge direction of the nearest neighbor graph.

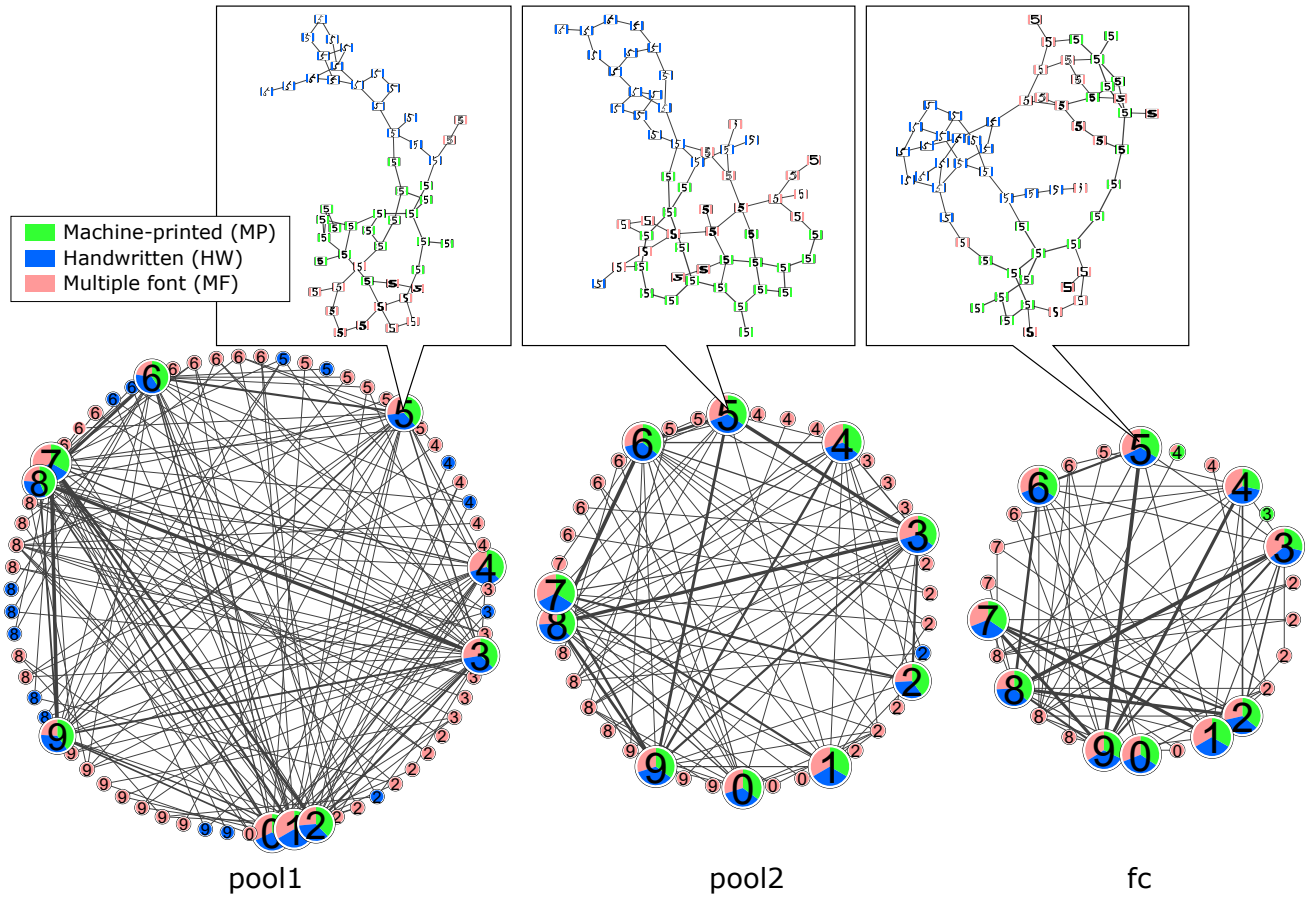

Fig. 10. Bottom: Clustered-RNG visualization of vector distribution at each layer. Top: RNG that forms the main cluster of "5".

size of each cluster node is relative to the number of united RNG nodes. Similarly, the thickness of each edge is relative to the number of RNG edges between those cluster nodes. In the clustered-RNG of Fig. 8, the edge between class "0" and "5" is thicker than the edge between "3" and "5", because there are two RNG edges between "0" and "5". Note that clustered-RNG is a coarse visualization but still represents inter-class relationship exactly.

In this paper, clustered-RNG is customized for type-fusion analysis. As shown in Fig. 8, each cluster node is a pie chart showing the ratio of three types in the node. In this example, the cluster node for class "0" contains all three types because three RNG nodes of different types are united. It should be noted that existence of cluster nodes containing multiple types is just a necessary condition for proving that types are fused. In other words, a cluster node containing multiple types does not always imply that types are not fused in the cluster node. As shown in Fig. 9, all three cases result in the same node visualization, although they have different degrees of type-fusion. On the other hand, a cluster node containing a single type proves that types are not fused at all in the cluster node.

B. Result of Type-Fusion Analysis

Figure 10 shows the clustered-RNG obtained at each layer by using all input patterns. Each clustered-RNG is visualized in a circular layout; starting from the bottom, in a counter-clockwise order, cluster nodes are arranged from class "0" to "9." Like the former experiment in IV-B, 90% of the entire samples were used for training the CNN. Then, 200 samples randomly chosen from the remaining 10% of each type were used as input patterns. Therefore, the number of all input samples is $N = 600$.

It can be observed that deeper layers have less cluster nodes[3]. Many small cluster nodes at `pool1` disappear at `pool2`. Especially, cluster nodes of HW almost disappear; this means the HW samples in those cluster nodes at `pool1` become closer to the main cluster node of their class at `pool2`. This is reasonable by considering that CNNs try to make class distributions more separable at deeper layers. In other words, each class distribution becomes less scattered, or more compact. At `fc`, classes become more separable. Especially, class "1" and "9" only has a single large cluster node. This means there is no pattern from other classes around the distribution of "1" at `fc`.

Since most cluster nodes contain multiple types, one might expect that type is more fused in a deeper layer. However, we need to be careful of the fact explained in V-A with Fig. 9.

[3]Clustered-RNG at the input layer (i.e., Clustered-RNG for the original input images) has more nodes but no significant difference from the clustered-RNG at `pool1`.

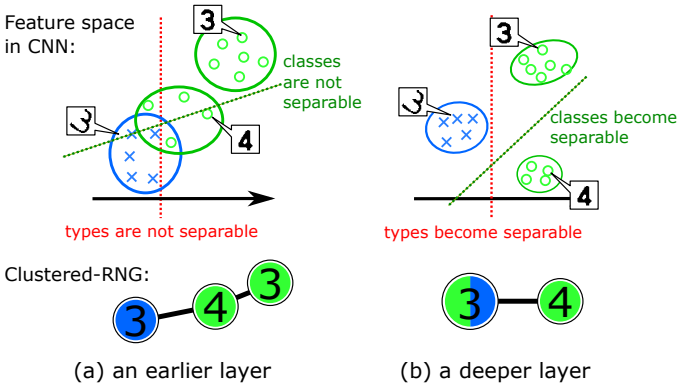

Fig. 11. How does a CNN manage different printing types? – A conclusion drawn from all analysis results.

Even if the distribution of each class becomes more compact, it does not always mean that types are fused. It is still possible that distributions of three types get closer but do not overlap like Fig. 9(a). Consequently, the clustered-RNGs of Fig. 10 satisfy the necessary condition for type-fusion but we need to make a further inspection to prove type-fusion.

In fact, it is proven by the upper part of Fig. 10 that types are not fused. This figure shows RNGs, each of which contains the nodes and edges clustered into the main cluster node of "5" in the clustered-RNG. In RNGs, three printing types are not fused. More specifically, HW is not fused at all with other types. MP and MF are slightly fused; this is simply because MF contains fonts similar to MP. In other words, there is no positive force to fuse MP and MF.

VI. Summarizing the Results from Two Approaches and Conclusion

Results from two type-fusion analysis approaches, i.e., statistical approach at Section IV and network visualization approach at Section V, coincide with each other — they say that *no type-fusion is made at any layer of a CNN*. In a sense, CNN takes a very rational strategy; if most input patterns are recognized correctly, CNNs do not need to fuse types. Moreover, the statistical analysis result says that *type is rather separated than fused in deeper layers.* Thus, roughly speaking, CNNs try to treat MP-'3"s, HW-"3"s, and MF-"3"s are different classes in its internal feature spaces and identifies them at the very final discrimination step.

Although types are not fused, we need to remember that they become closer to each other in a deeper layer. This was proven by the fact that clustered-RNG of a deeper layer has less cluster nodes, as shown in Fig. 10.

Considering all those analysis results, we can summarize that the distribution changes like Fig. 11. In an earlier layer like `pool1`, types are rather not separable in many units. In deeper layers, the distribution of patterns of a certain class and a certain type becomes more compact and thus types are more separable in many units of the layers. At the same time, as shown by the clustered-RNGs, the distributions of a certain class and different types get closer and the distributions of different classes get further. These translations of the distributions are, of course, reasonable for better discrimination.

The above conclusion gives a negative view for transfer learning between types; this is because the conclusion says that CNNs try to make different types more independent. This consideration coincides with the experiment done in [2], which reports that pre-training by MF gives almost no positive effect on the training of CNNs for HW.

To the authors' best knowledge, there is no research how human beings fuse printing types when they recognize characters. Thus, we cannot say anything about difference or similarity between CNNs and human beings in their type-fusion mechanism. We, however, make a hypothesis that human beings also do not fuse printing types in their feature extraction steps and fuse them in their final discrimination step. This is because our experiment proves that it is possible (for at least CNNs) to extract class-discriminative features while keeping types separated. If human beings have a similar feature extraction step, they also do not need to fuse types before the final discrimination step.

Acknowledgment

This research was partially supported by MEXT-Japan (Grant No. 26240024 and 17H06100) and Kakihara Foundation.

References

[1] K. Zagoris, I. Pratikakis, A. Antonacopoulos, B. Gatos, and N. Papamarkos, "Handwritten and Machine Printed Text Separation in Document Images Using the Bag of Visual Words Paradigm,", *ICFHR*, 2012.
[2] S. Uchida, S. Ide, B. K. Iwana, and A. Zhu, "A Further Step to Perfect Accuracy by Training CNN with Larger Data," *ICFHR*, 2016.
[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient- Based Learning Applied to Document Recognition," *Proc. of the IEEE*, 86(11):2278-2324, 1998.
[4] K. Fukushima. "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics*, 36(4):193-202, 1980.
[5] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud and V. Shet, "Multi-Digit Number Recognition from Street View Imagery Using Deep Convolutional Neural Networks," *arXiv*, 2013.
[6] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, "Reading Text in the Wild with Convolutional Neural Networks," *IJCV*, 116(1):1-20, 2016.
[7] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *ICLR*, 2015.
[8] A. Nguyen, J. Yosinski and J. Clune, "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images," *CVPR*, 2015.
[9] V. Escorcia, J. C. Niebles and B. Ghanem, "On the Relationship between Visual Attributes and Convolutional Networks," *CVPR*, 2015.
[10] M. Zeiler, G. Taylor and R. Fergus, "Adaptive Deconvolutional Networks for Mid and High Level Feature Learning," *ICCV*, 2011.
[11] A. Nguyen, J. Yosinski, J. Clune, "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks," *arXiv*, 2016.
[12] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng and T. Darrell, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition," *arXiv*, 2013.
[13] M. Goto, R. Ishida, Y. Feng and S. Uchida, "Analyzing the Distribution of a Large-scale Character Pattern Set Using Relative Neighborhood Graph," *ICDAR*, 2013.
[14] F. Rayar, T. Mondal, S. Barrat, F. Bouali, G. Venturini, "Visual Analysis System for Features and Distances Qualitative Assessment: Application to Word Image Matching," *DAS*, 2016.