현대자동차그룹 · 고려대학교 · 서울대학교 · 연세대학교 · 이화여자대학교 · 한국과학기술원

신기술 리서치를 위한 산학 클러스터 세미나 발표논문집 (I)

일시 **2020년 6월 26일 (금)**

장소 **한국과학기술회관**

CONTENTS

신기술 리서치를 위한 산학 클러스터 세미나 발표논문집(Ⅰ)

◈ Quality-discriminative localization for a root cause analysis of multisensor signal data ········· 3

발표자: 조윤상(고려대학교)

◈ PDF-GAN: Evading PDF Malware Classifiers using Generative Adversarial Networks ········· 31

발표자: 배호(서울대학교)

◈ Patient similarity learning via Bi-CNN with attention module ······ 47

발표자: 강승연(연세대학교)

◈ AMH-CNN: An Automatized Multi-step Hierarchical Convolutional Neural Networks Based on Data Inheritance ········· 61

발표자: 서이안(이화여자대학교)

◈ Reference-Based Sketch Image Colorization using Augmented-Self Reference and Dense Semantic Correspondence ··· 85

발표자: 이준수(한국과학기술원)

Quality-discriminative localization for a root cause analysis of multisensor signal data

발표자: 조윤상(고려대학교)

Quality-discriminative localization for a root cause analysis of multisensor signal data

Yoon Sang Cho, Seoung Bum Kim
School of Industrial Management Engineering
Korea University
{yscho187, sbkim1}@korea.ac.kr

Abstract

Root cause analysis (RCA) methods for effectively identifying critical causes of abnormal processes have drawn attention because manufacturing processes have become larger in scale and more complicated. However, existing methods for building automatic RCA models suffer from the disadvantage of typically requiring an expert’s knowledge. In addition, without a dataset representing the causal relationship of multivariate processes, it is difficult to provide useful information for an RCA. Although data-driven RCA methods have been presented, most are based on classification models. Considering that product quality is defined as a continuous variable in many manufacturing industries, classification models are limited in deriving root causes affecting the quality level of the product. In this study, we propose a regression model-based RCA method, named quality-discriminative localization, which consists of a convolutional neural network (CNN)-based activation mapping of multisensor signal data. In our proposed method, the CNN predicts the product quality of a continuous variable. Activation mapping then extracts causal maps that highlight significant sensor signals for each product. To identify the root causes, we generate a root cause map from the weighted sum of quality and causal maps. We consider root causes as locations of the abnormal process and processing time from localized activation scores on the root cause map. We demonstrate the usefulness of the proposed method in experiments with real data from a steel manufacturing process. Our results show that the proposed method successfully identifies root causes with distinct sensor signal patterns.

Keywords: root cause analysis, multisensor signal data, convolutional neural network, activation mapping, quality-discriminative localization, steel manufacturing

1. Introduction

Root cause analysis (RCA) plays a key role in maintaining stable manufacturing processes. An RCA aims to determine the critical causes leading to abnormal processes and product defects (Mahadevan & Shah, 2009). Recently, RCA methods have drawn attention because modern manufacturing processes have become more automatic, with processes linked to each other (Jia, Lei, Guo, Lin, & Xing, 2018). Once a process has an abnormal event with an unknown root cause, it adversely affects other processes. In particular, in large-scale and complex manufacturing systems, failing to detect root causes leads to the recurrence of problems, which can, unchecked, engender machine breakdowns, and decrease productivity (Weidl, Madsen, & Israelson, 2005). Therefore, an RCA model that appropriately explains the relationship between process states and product quality is required.

The purpose of this study is to propose an RCA model for understanding root causes that consider the following issues: First, abnormal signals, such as noisy symptoms, should be detected because they are directly associated with abnormal process states and decreased product quality. Second, the RCA model has to identify multilevel causes since the problems occur in relation to multivariate processes and processing times. Third, RCA methods need to detect not only temporary causes but also the unknown root causes that intrinsically lead to abnormal processes.

With advanced sensor technology, real-time multisensor signal data can be collected in many industries (Ronao & Cho, 2016; Wang, Chen, Hao, Peng, & Hu, 2019; Jiang, Hu, Liu, Yu, & Wu, 2016; Gong et al., 2019). Such sensor data represent sequential processes and processing time information that determines product quality. When the datasets include the causal factor, an RCA basically involves two steps: (1) construction of a predictive model and (2) identification of the root cause. The prediction model explains the relationship between process states and product quality, then detects the root cause from the sensor data with the highlighted parameters of the predictive model (Mahadevan & Shah, 2009; Chien, Hsu, & Chen, 2013).

In general, probabilistic and deterministic models are two approaches for RCA. In probabilistic model-based RCA, Bayesian networks that can achieve probabilistic reasoning have been widely used. These Bayesian networks derive the posterior probabilities and can detect the changes in sensor data (Weidl, Madsen, & Israelson, 2005; Nawaz, Arshad, & Hong, 2014; Liu, Liu, Cai, & Zheng, 2015; Wee, Cheah, Tan, & Wee, 2015). Weidl, Madsen, and Israelson (2005) proposed an object-oriented Bayesian network, which is a probabilistic graphical model that performs reasoning under uncertainty. Nawaz, Arshad, and Hong (2014) and Liu, Liu, Cai, and Zheng (2015) considered the cause and effect relationships between root causes, equipment, and process parameters using a Bayesian network. Furthermore, Wee, Cheah, Tan, and Wee (2015) proposed a Bayesian belief network-based causal knowledge model that provides causal strength using a fuzzy cognitive map. However, these studies are knowledge-based RCA models that depend on an expert’s knowledge (Lee, Cheon, & Kim, 2017). Although they are useful for identifying immediate causes, requiring an expert’s knowledge is a disadvantage in building automatic RCAs in large-scale systems.

Data-driven RCA methods based on deterministic models have become more attractive in modern industries because they can derive root causes from observations without model uncertainty (Li, Qin, & Yuan, 2016). Chien, Hsu, and Chen (2013) used Hotelling’s T^2 for sensor variable selection and analyzed the association between faulty products and sensor variables using decision trees. Mahadevan and Shah (2009) proposed a one-class support vector machine to identify abnormal processes and used support vector machine recursive feature elimination to determine the root causes. Although these methods perform reasonably well within the industrial realms for which they were designed, there is still plenty of room for improvement. Many feature selection methods suffer from the computational complexity in large volume datasets. Furthermore, rule-based models, such as decision trees, facilitate the interpretation of results, but they cannot be readily used with raw sensor signals.

Recently, with the surging popularity of deep learning for its computational and predictive performance, deep learning-based RCA models have become the prominent methods in various fields (Zhang, Peng, Wu, Yao, & Guan, 2017; Zhang, Peng, Li, Chen, & Zhang, 2017; Jia, Lei, Guo, Lin, & Xing, 2018). Deep neural networks directly use multiple sensor signals as input and automatically learn the desired information from the input data. For the diagnosis of defect causes in manufacturing processes, convolutional neural networks (CNNs) have been used. For example, Lee, Cheon, and Kim (2017) proposed a CNN structure, in which a receptive field tailored to multisensor signals slides along the time axis, to extract fault feature maps providing abnormal process variables and time information. In addition, Yao, Zhang, Yang, and Gui (2020) attempted fault diagnosis with CNN and a temporal attention mechanism to extract meaningful temporal parts from the sensor signals. However, these RCA methods have usually been performed to classify whether a product is defective or not. Although such classification model-based RCAs show good performance, they cannot be applied in situations where product quality is represented by continuous values. For example, in semiconductor manufacturing processes, quality is determined by failure rates. In steelmaking processes, quality can be determined by numerical values such as the weight deviation between the target and output products.

Only a few RCA methods have been presented to handle regression problems. Xia, Xia, Wan, and Cai (2012) used spectral regression for extracting fault feature extraction based on multisensor signal data. Borchert, Suarez-Zuluaga, Sagmeister, Thomassen, and Herwig (2019) attempted to use a partial least squares regression model that can derive variable importance and compared the performances between when they used raw sensor signals and when they used features extracted by principal component analysis. However, when using high-dimensional sensor signals, such approaches require feature selection or an extraction step, which are cumbersome for users. In particular, such approaches experience difficulty in deriving observation-wise causes because they focus on selecting significant variables. Hence, an effective and efficient RCA method for regression problems is required.

In the present study, we propose a regression model-based RCA approach that combines a CNN model and activation mapping with multisensor signal data. The key advantage of the proposed method is it visually explains which parts of the sensor signal cause abnormal quality. Thus, we named the proposed method quality-discriminative localization. The main contributions of this study can be summarized as follows:

(1) We propose a quality-discriminative localization method that localizes sensor signals representing causes of abnormal quality. In particular, the proposed method derives a root cause map (RCM) that emphasizes locations of process and processing time that have the most impact on decreased product quality.

(2) We apply the proposed method to a steel manufacturing system that consists of sequential processes that can affect product quality. To the best of our knowledge, this is the first attempt to use regression model-based RCA based on discriminative localization for sensor signals in a steel manufacturing process.

The remainder of this paper is organized as follows. Section 2 introduces the components of CNN. Section 3 describes the details of the proposed method. Section 4 presents experiments that examine the performance of the proposed method using real data from the steel manufacturing process. Finally, Section 5 contains our concluding remarks.

2. CNN

2.1. Components of CNN

A CNN is composed of a feature extraction network and a prediction network (LeCun, Bottou, Bengio, & Haffner, 1998). The feature extraction network mainly consists of convolutional, batch normalization (BN), and pooling layers (Badrinarayanan, Kendall, & Cipolla, 2017).

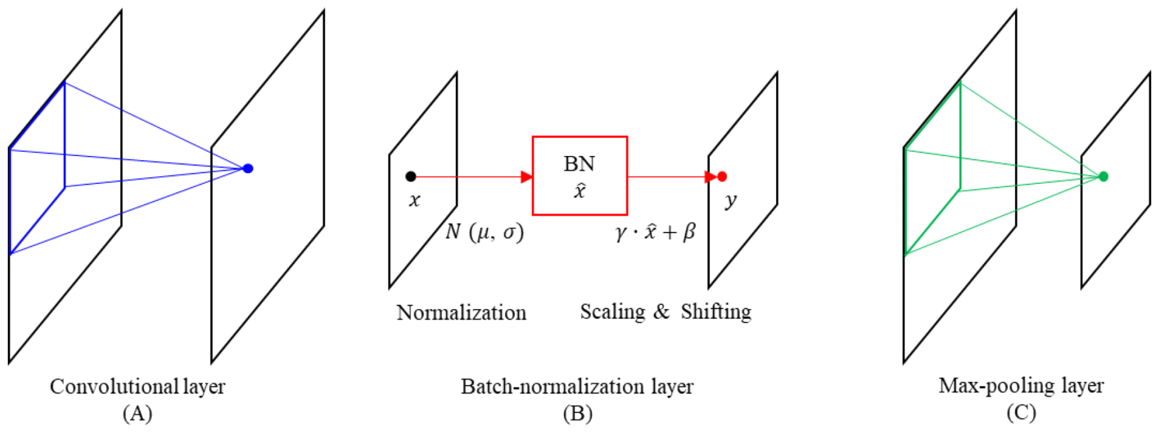


Figure 1. Illustration of components of the CNN: (A) convolutional layer, (B) BN layer, and (C) max-pooling layer.

Figure 1 illustrates the roles of each component of the CNN. Figure 1(A) shows the convolutional layer that extracts local features and constructs the feature maps, which are calculated by convolving the input with a learned convolutional kernel and applying a nonlinear activation function element-wise on the convolved results. The extracted feature maps contain distinctive features of input data. The convolution operation is followed by the activation functions, including the sigmoid, hyperbolic tangent, and rectified linear unit (ReLU) functions. These functions derive nonlinear features of the input that allow the network to make the learned features more dividable and are usually placed between two convolutional layers (Gu et al., 2018; Krizhevsky, Sutskever, & Hinton, 2012).

The BN layer is achieved through a normalization step that reduces the covariate shift and accelerates the CNN's training process (Ioffe, & Szegedy, 2015). As can be seen in Figure

1(B), the BN layer first normalizes the convolved features, then performs the linear transformation of the normalized features with scaling and shifting parameters to prepare them for the activation unit. The BN layer is usually added right after the convolutional layer and before the activation unit.

A pooling layer that reduces the spatial size of the feature maps to achieve shift-invariance is typically added after the convolutional layers and a BN layer (Badrinarayanan, Kendall, & Cipolla, 2017). Various pooling methods exist, including average pooling, max pooling, and global average pooling (GAP) (Gu et al., 2018). Among them, the max-pooling layer is widely applied in CNNs. As shown in Figure 1(C), the max-pooling layer is a downsampling layer that samples a maximum value from the feature maps and then contributes to reducing the number of parameters.

The GAP layer (Lin, Chen, & Yan, 2013) is also a downsampling operation, but it calculates the average values of each feature map, as follows:

$$F_k = \frac{1}{(W \cdot H)} \cdot \sum\nolimits_{x,y} f_k^l(x, y), \tag{1}$$

where $f_k(x,y)$ is the kth feature map in the lth layer, (x, y) represents the x- and y-axes information of the feature map, and W and H represent width and height feature maps. This gives the global average value F^k. The GAP layer is typically added after the feature extraction network and before a prediction network. Instead of adding a fully-connected layer for flattened features, the summarized vector F^k is fed directly into the activation function. One advantage of a GAP layer over a fully connected layer is that it makes it easier to construct an interpretable CNN by enforcing correspondences between feature maps and labels.

The prediction network comprises one or more dense fully-connected layers that are added with activation functions for each task. For a classification task, a soft-max function is used. In contrast, a linear activation function that directly passes the input value is used to build a regression model. The last dense layer has one neuron with a linear activation function that performs a linear combination of flattened feature values and learnable weight parameters (Gu et al., 2018).

2.2. Discriminative localization

A number of previous works have proposed visualizing the causality of CNN predictions by highlighting pixels that play an important role in prediction (Zhou, Khosla, Lapedriza, Oliva, & Torralba, 2016; Selvaraju et al., 2017). The most relevant to our study is the class activation mapping (CAM) approach to class-discriminative localization; Zhou, Khosla, Lapedriza, Oliva, and Torralba (2016) proposed a CAM method for identifying discriminative regions using a restricted class of image classification. They replaced fully-connected layers with convolutional layers and a GAP layer to produce class-specific feature maps. After training the CNN, the CAM derives an activation map by a weighted combination of the resulting feature values of the GAP and weights of a soft-max activation function. The activation map then explains which parts of an input image were looked at by the CNN for assigning labels. The activation map $M_n(x, y)$ of the nth observation is derived as follows:

$$M_n(x,y) = \sum_{k=1}^{K} F_k \cdot w_k^*, \quad (2)$$

where F_k and w_k^* are, respectively, the GAP value and learned weights of the last dense layer. In this study, we build a regression version of CNN-based activation mapping that can be used to predict the quality label of a continuous variable and to highlight the critical regions of two-dimensional multisensor signals data.

3. Quality-discriminative localization

Our goal is to derive root causes that can be visually explained for multisensor signal data. In this section, we present the architecture of the CNN, followed by the quality-discriminative localization method.

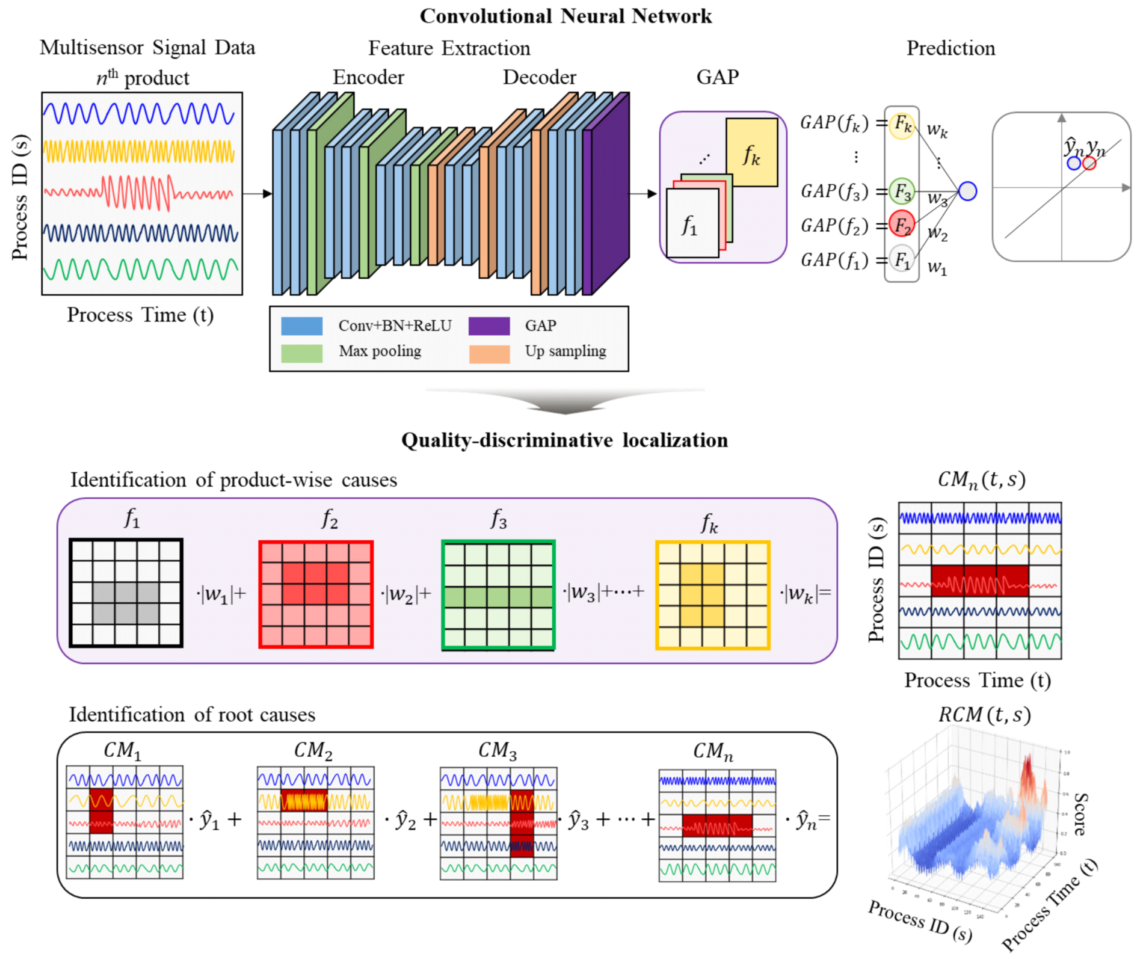


Figure 2. The proposed structure of the CNN and quality-discriminative localization.

Figure 2 shows the proposed structure of the CNN and quality-discriminative localization. For the feature extraction network of the CNN, we designed an encoder-decoder structure to generate output feature maps that are equal in size to the input data. The encoder is composed of three convolution blocks where each block includes two convolutional layers.

The output of each convolutional layer is fed into the BN with a ReLU activation function. A max-pooling layer then performs a downsampling operation. These layers form a block, and each block is repeatedly stacked three times. The decoder consists of a structure opposite the encoder's structure. To construct feature maps having the same size as the input, we add the upsampling layer instead of the max-pooling layer for each block. We make 32 feature maps with two-dimensional convolutional kernels of 3×3 size for all convolutional layers. Moreover, we use two-dimensional filters of 2×2 size for the max-pooling and upsampling layers. For the CNN's prediction network, a two-dimensional GAP layer is added. The GAP layer averages the last convolutional feature map $f_k^l(t,s)$, and the resulting values, F_k, are directly fed into the dense layer, which includes a linear activation function. The dense layer has one neuron, and the linear activation function enables the construction of the regression model. Thus, the formulation of the CNN regression can be summarized as follows:

$$\boldsymbol{Y} = \boldsymbol{W} \cdot \boldsymbol{F} = w_1 \cdot F_1 + w_2 \cdot F_2 + \cdots + w_k \cdot F_k, \tag{3}$$

where $\boldsymbol{Y}$ is a response variable, described by averaged feature maps $\boldsymbol{F}$ of extracted feature maps and weight parameters $\boldsymbol{W}$. Note that $\boldsymbol{W}$ can be considered the coefficients that allow us to identify the significant variables in the regression model. After training the model, we use the trained weight $\boldsymbol{W}^*$ as the importance for each F_k when deriving the activation map.

We train the CNN model such that cost function L is minimized. Cost function L can be defined as follows:

$$L = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2, \tag{4}$$

where *N* is the number of observations, y_n is the *n*th response variable, and $\hat{y}_n$ is the predicted value, which is the output of the CNN. The CNN is trained by an Adam optimizer, which is an algorithm for first-order gradient-based stochastic objective functions.

We now present the quality-discriminative localization for RCA. The purpose of the quality-discriminative localization is to visually emphasize sensor signals that are the root causes of abnormal quality. As shown in Figure 2, we first derive causal maps with activation mapping for all products and then identify the root causes with the localized sensor signals in the RCM. Having trained the CNN, we conduct the activation mapping to produce the following causal maps by the weighted sum of $|w_k^*|$ and $f_k(t,s)$:

$$CM_n(t,s) = \sum_{k=1}^{K} f_k(t,s) \cdot |w_k^*|, \tag{5}$$

where $|w_k|$ is an absolute value of w_k that can be considered the importance of each feature map $f_k(t,s)$. The score of $CM_n(t,s)$ highlights the important regions of the input data corresponding to the label. Considering that the regression model determines important variables based on the magnitude of weights, we use the absolute value of w_k. The $CM_n(t,s)$ explain the causes of the predicted quality of a product via localizing sensor signals. If a region contains high $CM_n(t,s)$ scores, the sensor signals of the corresponding region can be considered significant causes.

In the present study, the root causes are defined as sensor signals that represent abnormal processes and processing time. To identify the root causes, we obtain a quality-weighted activation map, named RCM. The RCM is calculated as follows:

$$RCM(t,s) = \sum_{n}^{N} CM_n(t,s) \cdot \hat{y}_n, \tag{6}$$

where $CM_n(t,s)$ is an activated map of the nth observation and $\hat{y}_n$ is a predicted value. Having constructed the RCM, we examine the localized positions that have scores exceeding percentile h. Thus, we interpret that the localized sensor signal is the indicator for the cause of $\hat{y}_n$. The causal maps show the causes of defects in the products, and the RCM explains the root causes accounting for all product quality. Algorithm 1 shows the procedure of the quality-discriminative localization.

Algorithm 1. Quality-Discriminative Localization

```
 1:  Input: Data X and y
 2:  Output: $RCM^*(t, s)$
 3:  ▷ Train the model
 4:  $w$ ← Initialize the parameters
 5:  Repeat
 6:          $\hat{y} \leftarrow \text{Model}(\mathbf{X}) = F_k \cdot w_k$
 7:          $L = \frac{1}{N} \cdot \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$
 8:          $w$ ← Update the parameters using the gradients of $L$
 9:  until the convergence of parameters using $L$
10:  ▷ Construct the RCM
11:  $RCM(t, s)$ ← Initialize the elements to zero of $t$ by $s$ matrix
12:  for n=1 to N do
13:          $CM_n(t, s) \leftarrow \sum_{k=1}^{K} F_k^n(t, s) \cdot |w_k^*|$
14:          $RCM(t, s) \leftarrow RCM(t, s) + CM_n(t, s) \cdot \hat{y}_n$
15:  end for
16:  ▷ Identify the root causes
17:  Let $h$ be a hyperparameter that indicates a threshold with percentile.
18:          If $RCM(t, s) > h$ then $RCM(t, s)$ = root cause's location
19:  $RCM^*(t, s)$ ← root cause's location
```

4. Experiments

4.1 Datasets

We used multisensor signal datasets obtained from a steel manufacturing process that produces steel from iron ore and scrap. The whole process is divided into four subprocesses: steelmaking, refining, continuous casting, and forming (Laha, Ren, & Suganthan, 2015). The steelmaking involves the input of raw materials being melted in a blast furnace. Refining reduces the impurities that can make the resulting molten steel brittle. Next, continuous casting places the molten steel into a cooled mold, causing it to solidify into a thin steel shell. It is then made into an intermediate-stage product such as a slab, bloom, or billet. The steel is formed into various shapes, often by hot rolling, a process that eliminates cast defects and achieves the required shape.

In this study, we focus on RCA for the continuous casting process, which plays an important role in determining the steel's quality. Figure 3 shows a procedure for the collection of multisensor signal datasets in continuous casting that consists of cooling, rolling, and cutting processes. Once molten steel enters the continuous casting process, the cooling process decreases its temperature. It is then rolled into the shape of an intermediate-stage product. In the cutting process, the molten steel is cut to the desired length. After the cutting process, we measure the difference between the weight of the intermediate-stage product and its target weight. This weight deviation is considered the quality of the final steel product.

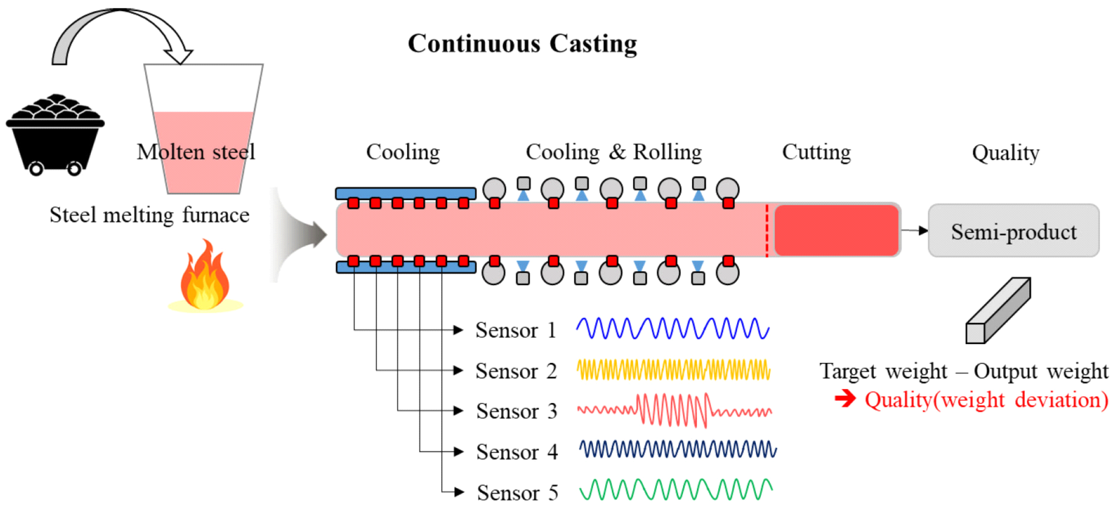


Figure 3. Illustration of data collection from the continuous casting process in the steel manufacturing process.

We collected 10 datasets from multiple sensors attached to each piece of the processing equipment. Table 1 shows a summary of the datasets. Each dataset was obtained from different product types and process paths to verify that the proposed method shows robustness and applicability under various process states. We set the time to 100 seconds for collecting signals for each sensor. To form the input data of the CNN model, we transformed the multiple sensor signals into two-dimensional data. Consequently, each of the products has a two-dimensional dataset that consists of the process ID of the y-axis and the processing time of the x-axis. The response variable is the weight deviation, which is measured after the cutting process, and its range is between 0 and 100. The larger the value of the response variable (weight deviation), the lower the quality.

Table 1. Summary of the two-dimensional multisensor signal datasets.

Dataset	Number of products	Number of sensors	Processing time (secs)
1	397	154	100
2	253	148	100
3	532	170	100
4	906	153	100
5	1,023	165	100
6	1,017	152	100
7	916	153	100
8	771	201	100
9	949	183	100
10	934	177	100

4.2 Evaluation of the predictive performance of the CNN

We evaluated predictive performance to demonstrate the usefulness of the proposed CNN for RCA. We performed a five-fold cross-validation using 80 percent of the data for training and 20 percent of the data for validation. We trained the CNN with a batch size of eight and epoch size of 100 for all datasets. We used the following performance measures with R^2, mean absolute error (MAE), and root mean squared error (RMSE):

$$R^2 = \frac{\sum_{n=1}^{N}(\hat{y}_n - \bar{y})^2}{\sum_{n=1}^{N}(y_n - \bar{y})^2}, \tag{7}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_n - \hat{y}_n|, \tag{8}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2}, \tag{9}$$

where y_n and $\hat{y}_n$ are the actual and predicted weight deviations of the n^{th} observation, respectively. R^2 represents the coefficient of determination calculated by the square of the correlation between y and $\hat{y}$. *MAE* is the average value over the validation data of the absolute differences between the actual and predicted values where all individual differences have equal weights, while *RMSE* is the square root of the mean squared difference between the actual and predicted values for the validation data. Table 2 presents the averages and standard deviations for the performances of the five-fold cross-validations. The average R^2, *MAE*, and *RMSE* of all datasets are listed in the table's last row. The average R^2 was 0.74, indicating that the proposed CNN structure has the robustness to explain the relationship between the sensor signals and product quality in various process states. The average *MAE* and *RMSE* were 5.77 and 8.75, respectively, indicating that the CNN correctly predicts the product quality where the quality variable is distributed between 0 to 100.

Table 2. The average of the predictive performances in five-cross validations. Standard deviations are presented in parentheses.

Dataset	R-squared	MAE	RMSE
1	0.68 (0.04)	5.60 (0.42)	7.90 (0.60)
2	0.64 (0.13)	5.29 (0.81)	7.42 (1.18)
3	0.69 (0.10)	4.84 (0.65)	6.73 (1.15)
4	0.75 (0.08)	4.48 (0.78)	5.86 (0.90)
5	0.76 (0.06)	5.24 (0.72)	6.86 (0.93)
6	0.70 (0.10)	4.78 (0.63)	6.29 (0.91)
7	0.88 (0.02)	3.58 (0.17)	4.84 (0.43)

8	0.70 (0.08)	9.72 (1.14)	12.96 (1.78)
9	0.82 (0.05)	7.21 (0.80)	9.41 (1.11)
10	0.76 (0.02)	6.97 (0.27)	9.21 (0.49)
Average	0.74 (0.07)	5.77 (0.64)	8.75 (0.95)

4.3 Evaluation of the localized sensor signals of the RCM

With the regression-trained CNN, we performed the quality-discriminative localization for RCA. We first extracted causal maps, which can provide temporary causes for each product, and then generated an RCM to identify the root causes.

Figure 4 shows two examples of the causal maps where the (A) causal map has the lowest weight deviation and the (B) causal map has the highest weight deviation in dataset 1. In the causal maps, the red highlighted regions exhibit more critical factors than the blue regions for predicting weight deviation. We consider that the red highlighted sensor signals represent the causal process states of predicted quality; thus, these regions allowed us to derive the location of abnormal sensor signals for process ID and processing time.

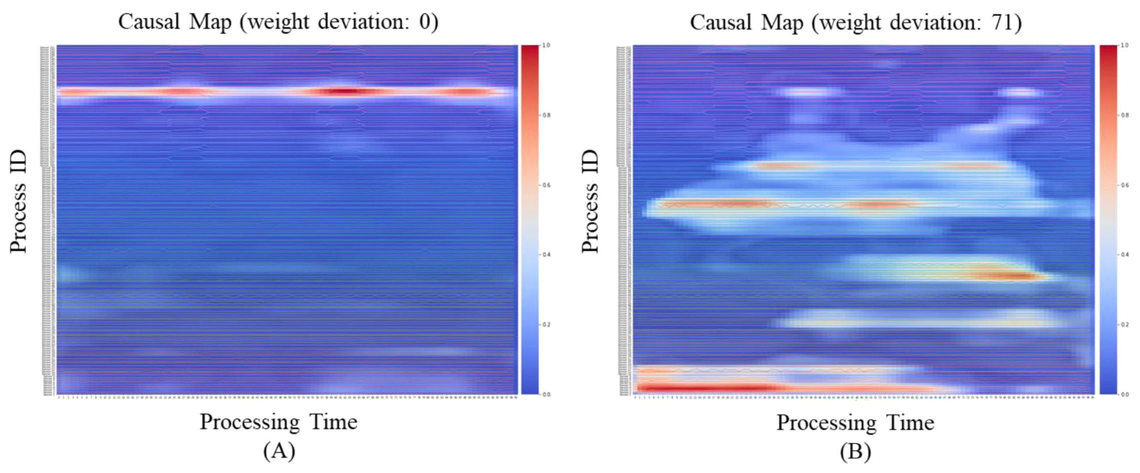


Figure 4. Causal maps: (A) causal map of a steel product with the lowest weight deviation and (B) causal map of the steel product with the highest weight deviation. The red regions indicate more important regions for predicting weight deviations.

As can be seen in Figure 5, we generated RCMs by the weighted sum of quality and causal maps and demonstrated three-dimensional activation maps for each dataset. In each RCM, the *x*-, *y*-, and *z*-axes indicate the process ID, processing time, and quality-weighted activation scores, respectively. The min-max normalization was performed to represent the quality-weighted activation scores between 0 and 1. The hyperparameter *h* was applied with a threshold of the 99th percentile to determine the root causes. The red highlighted regions of the RCMs indicate major causes of abnormal quality.

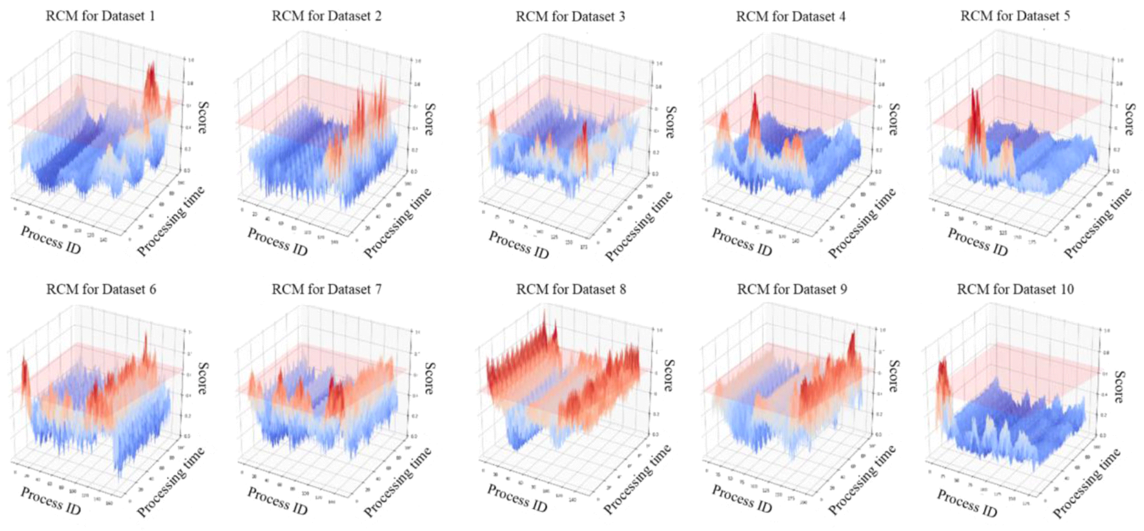


Figure 5. Results of the RCMs: The red plain indicates the threshold with the 99th percentile of quality-weighted activation scores.

To statistically verify that the localized regions of process ID and processing time exhibit significant causal sensor signals, we conducted a two-sample *t*-test for the comparison of sensor signals. We grouped the 100 signals into normal and abnormal classes, with 50 signals in each class. We summarized each of the sensor signals into seven statistical categories: mean, variance, min, max, median, range, area, and correlation coefficient. The area represents a summation of a signal for all time steps. We expected that, if each class has distinctive signal patterns, such statistics could represent the characteristics for the sensor signals. We also conducted a correlation analysis to derive a correlation coefficient for considering overall time information. The correlation analysis was performed between different sensor signals, from which we obtained a correlation coefficient matrix. We used an upper triangular matrix, excluding diagonal elements, as the input to the two-sample *t*-test and expected that the correlation coefficients of the sensor signals in the same class would be higher than the correlation coefficients from different classes. Using these summarized statistics, we performed a two-sample t-test to confirm the differences between the normal and abnormal groups. The *t* statistic is calculated by the following equation:

$$t = \frac{\bar{X}_{normal} - \bar{X}_{abnormal}}{\sqrt{\frac{S^2_{normal}}{N_{normal}} - \frac{S^2_{abnormal}}{N_{abnormal}}}}, \tag{10}$$

where N_{normal} and $N_{abnormal}$ are the sample sizes, S^2_{normal} and $S^2_{abnormal}$ are the sample variances, and $\bar{X}_{\text{normal}}$ and $\bar{X}_{\text{abnormal}}$ indicate the average values of the statistics from the sensor signals. We assumed the statistics have no equal variance.

Table 3 shows the resulting *p*-values of the two-sample t-tests. In each dataset, we listed three sensors that rendered high activation scores. The *p*-values, which are less than 0.05, are highlighted in bold. In some cases, we could not obtain the *p*-values because of the inflated *t* statistics caused by zero variance (indicated by the hyphens in Table 3). Most sensors exhibit *p*-values less than 0.05, implying that selected sensors have distinct features of the signals. However, we observed some cases with nonsignificant *p*-values (e.g., sensor 66 of dataset 4, sensor 119 of dataset 5, sensor 127 of dataset 6, and sensor 124 of dataset 7). We believe that these cases cannot be sufficiently explained based on the statistics used in this study. Nevertheless, the results showed that most localized signals exhibit significant differences.

Table 3. The *p*-values of the two-sample t-test for the significant sensors. Those in boldface represent *p*-values of less than 0.05.

Dataset	Sensor ID	*p*-value							
		Mean	Min	Max	Variance	Median	Range	Area	Correlation Coefficient
1	123	**0.00**	-	-	**0.00**	**0.00**	-	**0.00**	**0.00**
	149	**0.02**	-	**0.02**	**0.02**	-	**0.02**	**0.02**	-
	151	**0.00**	-	**0.00**	**0.00**	-	**0.00**	**0.00**	-
2	126	0.58	0.61	0.20	0.11	0.72	0.20	0.58	**0.00**
	116	**0.00**	-	-	**0.00**	**0.00**	-	**0.00**	**0.00**
	106	**0.00**	**0.00**	**0.00**	0.22	**0.00**	0.21	**0.00**	**0.00**
3	167	**0.00**	**0.00**	**0.00**	0.10	**0.00**	0.08	**0.00**	-
	158	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	157	**0.01**	**0.02**	-	**0.00**	0.32	**0.02**	**0.01**	-
4	66	0.14	-	0.15	0.13	0.08	0.15	0.14	-
	65	**0.00**	**0.00**	**0.00**	-	**0.00**	-	**0.00**	-
	64	**0.00**	**0.00**	**0.00**	0.15	**0.00**	0.17	**0.00**	-
5	121	0.06	-	-	0.07	0.17	-	0.06	**0.00**
	119	0.87	-	0.55	0.78	-	0.55	0.87	-
	137	**0.01**	0.30	0.99	0.83	**0.00**	0.79	**0.01**	**0.00**
6	125	0.13	-	**0.02**	**0.04**	-	**0.02**	0.13	-
	127	0.56	0.32	-	0.87	0.81	0.32	0.56	-
	141	**0.00**	0.23	**0.00**	**0.02**	**0.00**	**0.00**	**0.00**	**0.00**
7	2	0.16	**0.00**	**0.00**	**0.00**	0.10	**0.00**	0.16	0.59
	124	0.20	-	0.84	0.47	-	0.84	0.20	-
	4	0.88	**0.00**	**0.00**	**0.00**	0.71	**0.00**	0.88	0.39
8	180	0.62	**0.00**	**0.05**	0.15	0.08	0.07	0.62	**0.00**
	184	**0.04**	-	-	0.21	0.53	-	0.04	**0.00**
	182	0.27	1.00	-	0.52	1.00	1.00	0.27	**0.00**
9	58	**0.00**	**0.00**	**0.00**	0.42	**0.00**	0.43	**0.00**	0.30
	57	**0.00**	**0.00**	**0.00**	0.51	**0.00**	0.55	**0.00**	0.88
	67	**0.00**	**0.00**	**0.00**	0.42	**0.00**	0.36	**0.00**	0.99
10	15	**0.00**	**0.00**	**0.00**	0.66	**0.00**	0.53	**0.00**	0.69
	14	**0.00**	**0.00**	**0.00**	-	**0.00**	-	**0.00**	-
	13	**0.00**	**0.00**	**0.00**	0.41	**0.00**	0.98	**0.00**	0.81

5. Conclusions

We have proposed an RCA method called quality-discriminative localization. With multisensor signal data, the regression-trained CNN provides product-wise causal maps and generates an RCM to identify the most causal regions in multivariate processes. The proposed method allows us to interpret the multisensor signals by highlighting the distinctive patterns. In addition, through real-process sensor datasets, we demonstrate that the proposed method shows applicability and robustness with satisfactory predictive performance in the steel manufacturing process. The causal maps can be used for monitoring the causal process and processing time for all products. Furthermore, the RCM can be applied for discovering the root causes as processes that frequently cause abnormal quality. To verify the interpretability of the proposed method, we conducted a two-sample t-test for the time-series data and observed that the localized sensor signals of the two groups showed statistically significant differences. We believe that our study is the first attempt to demonstrate the applicability and usefulness of sensor signal regression-based discriminative localization to perform RCA in the steel manufacturing process.

Although the proposed method shows promising results, the activation mapping using the output of a GAP layer has a limitation that leads to information loss because the GAP layer summarizes the last feature maps. Nevertheless, CNN learns the parameters for predictions, even when using the summarized values of the GAP; verifying the predictive performance before conducting the activation mapping is essential. Moreover, it would be interesting to employ RCA with an alternating discriminative localization method, such as a gradient-based activation mapping that minimizes the information loss. The proposed CNN also has some limitations posed by the two-dimensional kernels in the convolution process. The two-dimensional kernel extracts a pixel-wise element of the local input. When we perform root cause identification, the pixel-wise activation scores involve risk because the element of each pixel can contain peripheral information from the previous input, i.e., the localized regions may represent not only the location of the target pixel but also the surrounding location. However, since the interaction relationship is the main factor in the CNN yielding powerful performance, we expect that if we show good predictive performance using a one-dimensional kernel, we can minimize the concerns.

Concerning how to use the discriminative localization via activation mapping, using the proposed method for identifying optimal processes is an interesting direction for future work. Despite the abnormal sensor signals, the detection of optimal sensor signals that explain the normal quality can be used for discovering the conditions of stable processes. We believe that the proposed method can be a cornerstone for using activation mapping in multiple sensor data-based root cause monitoring and be a useful tool for various manufacturing industries that require multilevel causal analysis for examining the characteristics of sequential processes.

References

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), 2481-2495.

Borchert, D., Suarez-Zuluaga, D. A., Sagmeister, P., Thomassen, Y. E., & Herwig, C. (2019). Comparison of data science workflows for root cause analysis of bioprocesses. *Bioprocess and biosystems engineering*, *42*(2), 245-256.

Chien, C. F., Hsu, C. Y., & Chen, P. N. (2013). Semiconductor fault detection and classification for yield enhancement and manufacturing intelligence. *Flexible Services and Manufacturing Journal*, *25*(3), 367-388.

Gong, W., Chen, H., Zhang, Z., Zhang, M., Wang, R., Guan, C., & Wang, Q. (2019). A novel deep learning method for intelligent fault diagnosis of rotating machinery based on improved CNN-SVM and multichannel data fusion. *Sensors*, *19*(7), 1693.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354-377.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jiang, P., Hu, Z., Liu, J., Yu, S., & Wu, F. (2016). Fault diagnosis based on chemical sensor data with an active deep neural network. *Sensors*, *16*(10), 1695.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Laha, D., Ren, Y., & Suganthan, P. N. (2015). Modeling of steelmaking process with effective machine learning techniques. *Expert systems with applications*, *42*(10), 4687-4696.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Lee, K. B., Cheon, S., & Kim, C. O. (2017). A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, *30*(2), 135-142.

Li, G., Qin, S. J., & Yuan, T. (2016). Data-driven root cause diagnosis of faults in process industries. *Chemometrics and Intelligent Laboratory Systems*, *159*, 1-11.

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Liu, Z., Liu, Y., Cai, B., & Zheng, C. (2015). An approach for developing diagnostic Bayesian network based on operation procedures. *Expert Systems with Applications*, *42*(4), 1917-1926.

Mahadevan, S., & Shah, S. L. (2009). Fault detection and diagnosis in process data using one-class support vector machines. *Journal of process control*, *19*(10), 1627-1639.

Nawaz, J. M., Arshad, M. Z., & Hong, S. J. (2014). Fault diagnosis in semiconductor etch equipment using Bayesian networks. *Journal of Semiconductor Technology and Science*, *14*(2), 252-261.

Ronao, C. A., & Cho, S. B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, *59*, 235-244.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618-626).

Wang, J., Chen, Y., Hao, S., Peng, X., & Hu, L. (2019). Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, *119*, 3-11.

Weidl, G., Madsen, A. L., & Israelson, S. (2005). Applications of object-oriented Bayesian networks for condition monitoring, root cause analysis and decision support on operation of complex continuous processes. *Computers & chemical engineering*, *29*(9), 1996-2009.

Wee, Y. Y., Cheah, W. P., Tan, S. C., & Wee, K. (2015). A method for root cause analysis with a Bayesian belief network and fuzzy cognitive map. *Expert Systems with Applications*, *42*(1), 468-487.

Xia, Z., Xia, S., Wan, L., & Cai, S. (2012). Spectral regression based fault feature extraction for bearing accelerometer sensor signals. *Sensors*, *12*(10), 13694-13719.

Yao, Y., Zhang, S., Yang, S., & Gui, G. (2020). Learning Attention Representation with a Multi-Scale CNN for Gear Fault Diagnosis under Different Working Conditions. *Sensors*, *20*(4), 1233.

Zhang, R., Peng, Z., Wu, L., Yao, B., & Guan, Y. (2017). Fault diagnosis from raw sensor data using deep neural networks considering temporal coherence. *Sensors*, *17*(3), 549.

Zhang, W., Peng, G., Li, C., Chen, Y., & Zhang, Z. (2017). A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. *Sensors*, *17*(2), 425.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2921-2929).

PDF-GAN: Evading PDF Malware Classifiers using Generative Adversarial Networks

발표자: 배호(서울대학교)

PDF-GAN: Evading PDF Malware Classifiers using Generative Adversarial Networks

Anonymous Author(s)

ABSTRACT

Recent research has shown that a small perturbation to an input may forcibly change the prediction of a machine learning (ML) model. Such perturbations are commonly referred to as *adversarial examples*. Early studies on adversarial examples have focused mostly on ML models for image processing, and continuously expanded to other applications including those for malware classification. In this paper, we are interested in the problem to find adversarial examples against ML-based PDF malware classifiers. We deem that our problem is more challenging than those against ML models for image processing because of the highly complex data structure of PDF in comparison to traditional image datasets and of an additional constraint that the generated PDF should exhibit malicious behavior. To resolve our problem, we propose a variant of *generative adversarial networks* (GANs) that generate evasive variant PDF malware (without any crash), which can be classified as benign by various existing classifiers, yet maintaining the original malicious behavior. Our model exploits the target classifier as the second discriminator to rapidly generate an evasive variant PDF with our new feature selection process that deliberately includes unique features extracted from malicious PDF files. We evaluate our technique against three representative PDF malware classifiers (Hidost '13, Hidost '16 and PDFrate-v2) and further test its effectiveness with AntiVirus engines from VirusTotal. To the best of our knowledge, our work is the first to analyze the performance against the commercial AntiVirus engines in addition to the public benchmarks. Our results are quite encouraging; our model finds, with extremely great speed, evasive variants for all selected seed against state-of-the-art PDF malware classifiers. We argue that our results against commercial AntiVirus engines may raise a serious security concern in the presence of adversaries.

1 INTRODUCTION

Machine learning (ML) has extensively adapted in a large number of application areas including speech recognition and image processing. One important such area is security, to which a variety of ML-based techniques have been applied in recent years. Several studies have empirically evinced a great potential and effectiveness of ML in solving certain security problems like malware detection [43, 53]. In particular, the ML techniques for malware detection have been developed for years diverging to many different security problem domains, such as clustering of malware families [15, 26], detection of malicious downloads [17, 41], detection of account misuse networks [19, 51], detection of commonly exploited file formats (e.g., Java archives [44], documents [27, 32]) and detection of PDF malware [47–50].

Not surprisingly, as ML becomes a dominant means for malware analysis, there is a growing temptation to find *adversarial examples* (AEs) that can diminish its effectiveness. Many latest studies of AE attacks on ML [14, 20, 23, 30] have demonstrated that a small perturbation to an input may forcibly change the prediction result of both ML and, newly surfacing, *deep learning* (DL) models. The studies suggest that the victim of AE attacks can be anyone from an entire spectrum of application areas where ML is applicable. As a result, this gloomy fact poses a daunting challenge to developers of ML models for malware detection [22]. Thus, it is crucial to build a robust malware detectors or classifiers that are resistant to AE attacks. One solution adopted by many in practice is to harden their model by training it with all possible AEs against it taken into account. For this, significant effort has been made to identify AEs against existing ML-models for various malware detectors. In this paper, we are interested in finding AEs that can be used to evade all the current (academic and commercial) ML-based PDF malware classifiers. Our interest originates from the fact [36] that as malicious PDF files have been known to be the most dangerous type of attack exploited by adversaries to date, ML-based techniques are being actively studied and developed to mitigate them even most recently.

Since its introduction, PDF has risen in great popularity and become the de facto standard for many different purposes of information sharing, such as text documents, data files and presentation materials. PDF includes not only static content (e.g., texts and styles) but also dynamic content (e.g., JavaScript code and action triggers). The versatility of PDF files comes in their capability of displaying such rich content on virtually all kinds of today's computer systems and platforms. The popularity and versatility have been capitalized on by adversaries in a way that the PDF format files are used to craft diverse attacks on viewer applications, inflicting extensive damage on countless victims. One key attribute of PDF exploited by adversaries is its high connectivity to other objects which facilitates modification of a PDF file, ultimately leading to an injection of a malicious load to the file. Another is the innate complexity of its file format, which facilitates malicious contents being concealed from the detectors. For example, JavaScript-based attacks deceive the detectors by injecting Javascript code in multiple objects at different locations inside the file. Such PDF malware is not only posing in the present but also likely to pose in the future, an immense threat to cybersecurity. It was reported by SonicWall [12], a private network security company, that more than 47,000 new attacks related to PDF files were discovered last year, and 73,000 PDF-based attacks were discovered in March, 2019 alone.

To prepare for the flooding of future zero-day PDF malware, much research has been done to improve the performance of PDF classifiers by employing ML techniques. As the first work in this direction, PDFrate-v1 [47] tackled the challenge with ML techniques by using metadata and *contents* of PDF documents. The approach characterized the documents' attributes by hand-crafting 202 features to train a random forest (RF) model for detection. PDFrate-v2 [48] further improved the performance by taking advantage of an ensemble training technique. Hidost [49, 50] attempted to extract the files into a *structural* map and used it as a feature set for their train model. Support vector machines (SVMs) and RF are used as classification models and both of them attain an impressive performance of detection.

As ML techniques for PDF classification become advanced and sophisticated, so do AE attack techniques for evading them. In principle, the purpose of these attack techniques is generating evasive PDF samples (i.e., AEs) against ML-based classifiers by picking and manipulating structural features that the classifiers utilize for detection. The early forms of AE attacks, which we collectively call *mimicry* attacks [21, 31, 45], aim to induce misclassifications of the classifiers by camouflaging malicious PDF files as benign ones. Unfortunately, mimicry attacks rely heavily on human expertise to understand a given malicious PDF file before finding fake structural features that will be added to the original file for aligning it with a known benign file. This implies that the success of their techniques would be strictly limited by human effort as well as their knowledge of a complex structure of the PDF file format.

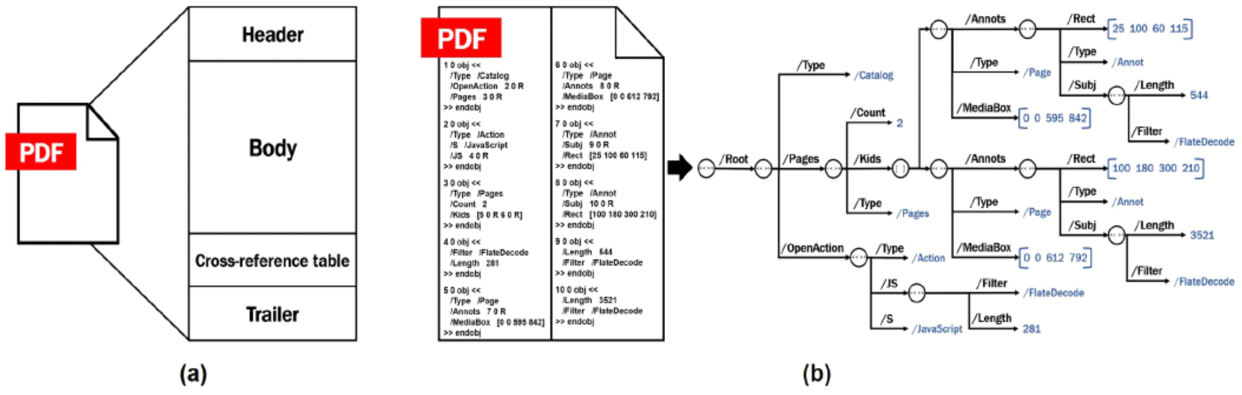
Researchers endeavored to overcome the limitation by minimizing human involvement. They proposed the evasion techniques that can automate the process of generating evasive samples for AE attacks. EvadeML [52] introduced a stochastic approach based on *genetic programming* (GP), which repeatedly performs feature manipulations based on a random mutation algorithm until an evasive, yet malicious PDF sample is successfully obtained as output. While the output sample maintains the input's original maliciousness, it, unlike the input, is guaranteed to be evasive as it will induce a misclassification of PDF malware classifiers. EvadeML exhibited its effectiveness by producing evasive samples of all 500 PDF malware files selected from Contagio malware archive [6]. A later work, EvadeHC [18], claimed to achieve the same performance as EvadeML; that is, succeeding in generating evasive samples for all 500 PDF malware files from Contagio. Moreover, they assumed a more restricted, realistic attack scenario where the attacker will only be given a binary prediction score from the PDF malware classifiers.

Despite the impressive success in their automated evasive sample generation, our analysis has revealed that the existing evasion techniques consume an excessively large amount of time to obtain each sample. Although EvadeHC has made some effort to speed up its generation time by applying a hill-climbing method to the *random mutation* algorithm, their numbers still have much room for improvement. According to our observation, the main factor that increases the total time taken to generate an evasive sample is the inherent difficulty of maintaining the original maliciousness even after several trials of operations being carried out to manipulate structural features, which often result a crash. To explain this, consider the PDF malware that is not originally evasive when being given as input to the evasion techniques like EvadeML or EvadeHC. In order to generate an evasive sample as output from the malicious file, they must transform the original file structure by manipulating (i.e., inserting, deleting and replacing) its structural features. Conceivably, it often occurs during the transformation that the original file loses its maliciousness if a certain feature manipulation happens to corrupt a file structure essential to maintain its maliciousness. Let hereby S denote a set of all structural features of our target PDF file, which can be manipulated to generate our evasive sample. We also define S', a subset of S, whose elements are *relevant* (essential for reconstructing PDF form) or crucial to maintaining the target's maliciousness, by which we mean that the maliciousness might be corrupted by changing any of them.

As briefly mentioned earlier, the existing evasion algorithms 'randomly' select one feature after another from S and 'mutate' the features until they obtain an evasive sample. Suppose that they select and mutate features from S' by chance. Then as been defined, it is likely that the maliciousness of our target file is lost by error. Upon recognizing the loss of maliciousness, the existing algorithms undo the mutation on the feature to restore the lost maliciousness and try to select another feature from S. We have observed in the existing techniques that they suffer from quite frequent trials and errors, each of which causes a waste of time, consequently all in all inducing a significant increase in the total time for sample generation.

A remedy for this problem would be to pinpoint the subset S' and transform the input PDF malware by manipulating features only from the complementary set of S' (i.e., $S - S'$) in search for an evasive malware sample. Sadly, none of the existing techniques listed above attempt to have knowledge of S' when they generate evasive samples. Our observation on previous work motivated us to develop a new solution where we drastically reduce the sample generation time by avoiding wasteful cycles of trials and errors during our PDF transformation. In our solution, we first strive to identify a set S' of structural features that are relevant to malicious behaviors of most PDF malware available today. Next, during the transformation phase, we continuously refer to the set in order to ensure that our algorithm should select candidates for mutation from the complementary set of S'.

Clearly, in order for our solution to work successfully, we must be able to determine the set S' for the PDF format files. To achieve this, we employ the *generative adversarial networks* (GANs), which,if appropriately trained, can learn to identify intrinsic properties (including structural features) of benign and malicious PDFs. The power of GANs that identifies the structural features belonging to S' comes from their innate characteristic, namely the adversarial interaction between their two components, the generator and discriminator, by which S' is formed. To avoid the time-consuming repetition of trials and errors, the generator constructs evasive samples by modifying features only in $S - S'$. Many existing GANs usually employ a single discriminator, through which a modified sample very similar to the original can be generated. To generate a modified sample structurally very similar to the original, GANs select features in $S - S'$, thereby conserving the original's malicious behavior as a result. However, the generated sample must not only maintain the desired maliciousness but also evade the targeted malware classifiers. To satisfy both the requirements, we have introduced a GAN variant that employs a target PDF malware classifier as the second discriminator to manipulate features only from $S - S'$ during our evasive sample generation. To adjust the dependency level of these two cooperative discriminators, we use an additional parameter that controls the balance between them. Let alone its speed in finding evasive malware samples, our solution has another advantage that it can operate even under a realistic black-box attack



```
1 0 obj <<
  /Type   /Catalog
  /OpenAction   2 0 R
  /Pages   3 0 R
>> endobj
2 0 obj <<
  /Type   /Action
  /S   /JavaScript
  /JS   4 0 R
>> endobj
3 0 obj <<
  /Type   /Pages
  /Count   2
  /Kids   [5 0 R 6 0 R]
>> endobj
4 0 obj <<
  /Filter   /FlateDecode
  /Length   281
>> endobj
5 0 obj <<
  /Type   /Page
  /Annots   7 0 R
  /MediaBox   [0 0 595 842]
>> endobj
6 0 obj <<
  /Type   /Page
  /Annots   8 0 R
  /MediaBox   [0 0 612 792]
>> endobj
7 0 obj <<
  /Type   /Annot
  /Subj   9 0 R
  /Rect   [25 100 60 115]
>> endobj
8 0 obj <<
  /Type   /Annot
  /Subj   10 0 R
  /Rect   [100 180 300 210]
>> endobj
9 0 obj <<
  /Length   544
  /Filter   /FlateDecode
>> endobj
10 0 obj <<
  /Length   3521
  /Filter   /FlateDecode
>> endobj
```

Figure 1: PDF structure (a), Tree representation of PDF file (b)

scenario, in which the classification score revealed from the malware classifiers is in binary form (i.e., benign or malicious) rather than a continuous classification score.

We have evaluated our solution against three PDF malware classifiers. First of all, we have found that it can generate evasive samples (without any crash) for all 500 unique PDF malware files selected from the Contagio archive. Our proposed model successfully evades the target PDF malware classifiers with the maximum number of 12 manipulating operations by 13 times faster than the previous approaches. In contrast, EvadeML required the maximum of 354 and 85 feature manipulating operations to complete the generation of evasive samples for PDFrate-v1 and Hidost '13 respectively. To further demonstrate the effectiveness of our approach, we include in our evasion seed all three types (e.g., JavaScript, ActionScript and File Embedding) of PDF malware as known by CVE-2018-9958 [11], CVE-2013-2729 [7] and CVE-2010-3654 [4]. The analysis reveals that our evasive samples are all generated without any crash exhibiting the same malicious behaviors as the original malware with minimum modification. Last of all, unlike previous work, we evaluate the evasiveness of our generated malware samples against AntiVirus engines from VirusTotal.

2 BACKGROUND

In this section, we describe the threat model and current state-of-the-art PDF malware classifiers and evasion attacks.

2.1 Threat Model

Depending on the different levels of knowledge held by an attacker, attack scenarios can be categorized into three different classes: white-box, gray-box and black-box. The less information available to an attacker, the darker the attack scenario is considered. The types of information that can be provided to an attacker are three-fold: (1) the training dataset and its labels, (2) the feature set and the feature extraction algorithm of the classifier with its extracted feature types and (3) the knowledge of the classification function and its hyper-parameters.

In the black-box scenario, an attacker is provided with minimal knowledge of the classifiers (e.g., the feature representation). EvadeML constructs evasive samples against Hidost '13 and PDFrate-v1 under a black-box attack scenario. They assumed that the classification score was revealed in a real number with many query attempts. EvadeHC also operates under a similar scenario but the main difference is that the classification score was given in the binary form.

Our approach, PDF-GAN, operates in the same black-box scenario as previous studies with an assumption that many submissions of files are allowed. Also, the classifiers only reveal the classification score in a binary form (i.e., benign or malicious). However, recently, many researchers managed to mitigate the evasion attack by limiting the number of queries as the current black-box attack requires many submissions. To this end, PDF-GAN was designed to also operate as a transfer-based attack [40]. For this, we trained a surrogate model that is a smaller network and evaluated the success rate of evasion with much fewer query attempts to the target classifiers. The details of the design and the experiments will be explained in the following sections.

2.2 Portable Document Format (PDF)

2.2.1 PDF Structure. A PDF file can be broken down into four parts: *header*, *body*, *cross-reference table* and *trailer*. Figure 1 (a) shows the structure of a PDF. A *header* contains rather simple information which includes the version number of the PDF specification (i.e., '%PDF-1.3'). The *body* section contains objects and holds all data of the document. There are eight different types of objects supported by a PDF (i.e., Boolean, integer and real numbers, arrays, strings, dictionaries, names, streams and null). A name object only contains unique values, whereas a dictionary object consists of a key and value pair.

Objects are identified by their given numbers, and they are either indirect objects or the direct objects constituting dictionaries. Indirect objects appear within the notation << >> and direct ones are denoted as follows:

```
6 0 obj  << /Type/Action/S/JavaScript/JS 7 0 R >>  endobj
7 0 obj  << /Length 231/Filter/FlateDecode >>
stream  · · ·  endstream endobj
```

For example, the object 6 with a keyword introduced by '/' will make an indirect jump to the object 7, which contains a sequence of direct objects with keywords and their values. The length of the stream is 231, which requires a FlateDecode filter. A *cross-reference table* stores the mapping information of random and direct access, allowing a specific object to be found without having to search throughout the entire file. Note that PDF readers start rendering the PDF from the bottom of the file, which is the trailer. The *trailer* specifies the offset value for the PDF reader to find the cross-reference table and helps the reader find a specific object more quickly (i.e., trailer << /Size 9 /Root 1 0 R >> startxref 9178 %EOF). In this case, the offset is 9178 bytes, '/Size' indicates the number of entries in the cross-reference table and '/Root' is the catalog dictionary for this file.

2.2.2 Types of PDF Malware. The three different types of PDF malware are briefly explained. (1) *JavaScript-based attacks* exploit a vulnerability using JavaScript code that can be embedded in one or several objects. Typical examples of such vulnerabilities are an API-based overflow and a Use-After-Free flaw. (2) *ActionScript-based attacks* capitalize on the fact that PDF files can visualize Flash content. This is usually achieved by embedding ShockWave Flash along with the ActionScript code such as memory corruption or corrupted file code. (3) *File-embedding attacks* take advantage of the fact that Adobe Reader can parse and read PDF files that are embedded with contents of different file types, such as images (e.g., bmp or tiff) and fonts (e.g., ttf). When reading a PDF file, embedded contents can lead to memory spraying to execute payloads with malicious activities.

2.3 PDF Malware Classifiers

2.3.1 Hidost. Hidost is implemented using two different types of classification models: support vector machine (SVM) [49] and random forest (RF) [50]. SVM is a supervised learning model that outputs an optimal hyperplane for separating two different labels. RF is a meta estimator, comprising several decision trees that are merged for more accurate classification. As the first step, Hidost utilizes Poppler [10] a PDF parser to dissect files into a structural multi-map in its structure extraction stage. These structural paths of objects in a PDF are used as features during classification. Since there are many semantically equivalent yet syntactically different structures, a structural path consolidation (SPC), which is based on rules that are manually created, is carried out. For feature selection, Hidost naively includes only paths that are occurring in more than a certain number of files to form a feature set. Hidost is provided as open-source, and the model was trained using 10,000 random files with a malicious-to-benign ratio of 1:1. The entire PDF dataset was composed of 407,037 benign and 32,567 malicious files. The results indicated that the Hidost model was the top detection tool compared to AntiVirus engines (VirusTotal).

2.3.2 PDFrate-v2. The PDFrate classifier is implemented using an RF algorithm that applies an ensemble learning model designed to improve the prediction accuracy [48]. It employs *metadata* and the *content* of the PDF files as classification features, which include the names of the authors of the files, the size of the file, the position and the number of specific keywords. The feature set is defined

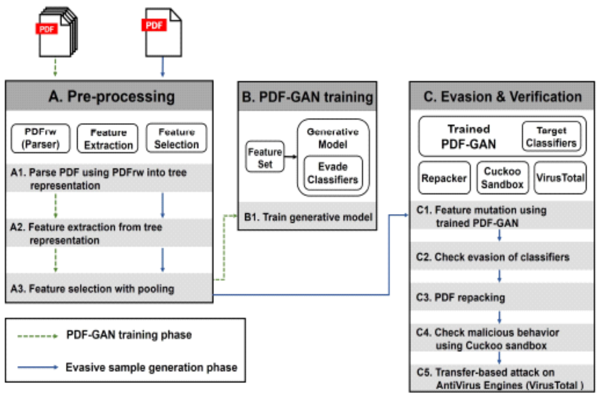

PDF
PDF
A. Pre-processing
PDFrw (Parser)
Feature Extraction
Feature Selection
A1. Parse PDF using PDFrw into tree representation
A2. Feature extraction from tree representation
A3. Feature selection with pooling
B. PDF-GAN training
Feature Set
Generative Model
Evade Classifiers
B1. Train generative model
C. Evasion & Verification
Trained PDF-GAN
Target Classifiers
Repacker
Cuckoo Sandbox
VirusTotal
C1. Feature mutation using trained PDF-GAN
C2. Check evasion of classifiers
C3. PDF repacking
C4. Check malicious behavior using Cuckoo sandbox
C5. Transfer-based attack on AntiVirus Engines (VirusTotal)
PDF-GAN training phase
Evasive sample generation phase

Figure 2: Flowchart of PDF-GAN framework

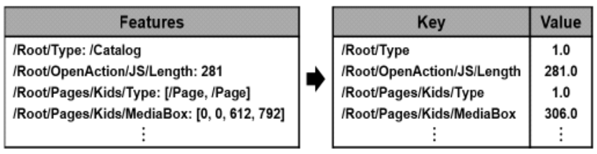
manually by the authors and the total number of features is 202. However, only 135 are publicly available in the Mimicus implementation of PDFrate, which claims to achieve a close approximation. The main difference between PDFrate-v1 and PDFrate-v2 lies in the ML model applied. PDFrate-v2 adopts an ensemble method by applying mutual agreement among the classifiers. It introduces the idea of 'uncertain' in the classifier votes, where rates of 25% to 50% are considered to be benign uncertainty, whereas rates of 50% to 75% imply malicious uncertainty. The effectiveness was tested against some known evasive attacks such as mimicry [31] and reverse mimicry [37], and impressive performance was demonstrated.

2.4 Evasion Attacks

2.4.1 Automatically Evading Classifiers. EvadeML presents a generic approach for evading the Hidost '13 and PDFrate-v1 classifiers through stochastic manipulations. It repeatedly mutates the original malicious PDFs to create evasive variants. It is an automated procedure in which evasive samples manufactured by random mutations are tested by the oracle to check the presence of maliciousness. If no maliciousness is present, the variant will be returned to the mutation stage. As for the reliable malware signature, only the network behavior of the malware samples is considered. A total of 500 sample seeds were selected from the Contagio PDF malware dataset and the proposed method successfully reached 100% evasion, which took approximately six days.

However, PDF-GAN is based on learning the difference in the feature sets between benign and malicious samples and modifying a malicious PDF with minimum effect on its original purpose, and hence achieving a 100% evasion success rate in noticeably less time and with fewer modifications.

2.4.2 Evading Classifiers by Morphing in the Dark. In this study, the authors focused on more restricted and realistic attack scenarios where the target classifiers will only reveal the final prediction regarding whether they are benign or malicious. Hence, a scoring mechanism, EvadeHC, was proposed to overcome the limited information. The intuition behind this is to measure the number of steps to overturn the result of the detector and derive the real-value score from it. The authors introduced the notion of malice-flipping distance, which is the number of mutations required for a malicious



Features
/Root/Type: /Catalog
/Root/OpenAction/JS/Length: 281
/Root/Pages/Kids/Type: [/Page, /Page]
/Root/Pages/Kids/MediaBox: [0, 0, 612, 792]
⋮

Key	Value
/Root/Type	1.0
/Root/OpenAction/JS/Length	281.0
/Root/Pages/Kids/Type	1.0
/Root/Pages/Kids/MediaBox	306.0
⋮	⋮

Figure 3: Feature abstraction [Dictionary (Key:Value)]

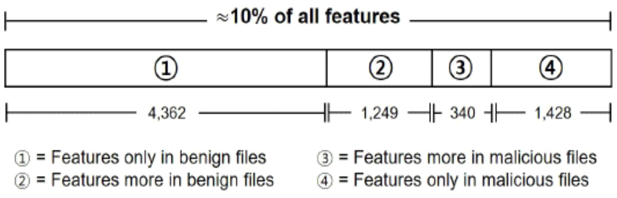

≈10% of all features
①
②
③
④
4,362
1,249
340
1,428
① = Features only in benign files
② = Features more in benign files
③ = Features more in malicious files
④ = Features only in malicious files

Figure 4: Feature selection by pooling

PDF to lose its maliciousness as determined by a tester. The reject-flipping distance is a comparable concept, which is the number of morphing steps required for a malicious sample to be classified as benign. A simple morphing technique is employed that performs the basic operations: insert, delete or replace. Their design consists of three components: a binary output detector, a tester to check the maliciousness of evasive samples, and a morpher that randomly mutates the PDF files. The target classifiers were Hidost '13 and PDFrate-v1 and its effects were evaluated with the 500 selected malware samples from Contagio archive.

Our approach operates under the strong assumption that the classifiers and testers only reveal their binary output results. Unlike this work, our PDF-GAN did not need a scoring function to convert the results into a real-value score and successfully evaded even more recent classifiers with the same seed samples.

3 DESIGN

In this section, the design of our approach will be explained in detail. First, how features are extracted from PDF files and selected to be used as a feature set for training PDF-GAN. The explanation of how we selected a seed file for our mutation and evading model architecture will be followed by PDF repacking process. Figure 2 shows an overview of our proposed method, which consists of three phases: 1) pre-processing of PDF, 2) PDF-GAN training and 3) detection. The details are presented in the following sections.

3.1 Feature Extraction

PDFs are parsed into the tree representation as shown in Figure 1 (b). We have utilized the PDFrw (i.e., PDF parser) provided by EvadeML [8] with few modification to correctly parse all objects. It is important that the parser do not omit any major objects (e.g., */Javascript* and */OpenAction*) as PDF-GAN would be unable to fully interpret the structural difference between benign and malicious PDFs, which in turn will lead to poor results in PDF-GAN's performance. Thus to avoid leaving out any potentially pivotal information, PDFrw has been modified to include all key values of */Root* while parsing PDFs into a tree representation (i.e., */Metadata*, */OpenAction*, */Javascript*, */AcroForm*, */PageLayout*, etc). Additionally, for higher speed computation, we ignore any paths containing an object with */Parent* or */Prev* as they are recursive.

From the tree representation, a feature set can be formed. Each path from the root to a leaf node and its value is considered as a feature as listed in Figure 3 (left). The feature abstraction is performed by converting features into a form of dictionaries (i.e., keys and values). Finally, similar to the previous work [49, 50], any values in a string type were converted to an integer value of 1 and a value given in the form of an array of values was transformed into the median of values in an array as shown in Figure 3 (right).

Description	No. of seed
Contagio dataset (excl. training set)	6,105
Cuckoo result with network activities	1,503
PDFrw parsing	1,502
Feature extraction	1,485
Unique files	712
True positive of SVM & RF & Ensemble	709
Randomly selected samples	500

Table 1: Mutation seed selection process

3.2 Feature Selection Process

The feature selection process plays a crucial role in determining the effectiveness of the ML. Since it is impractical to use all features extracted from the entire dataset, we must select a decent number of features that represent all features in a sufficient manner, to be included in the feature set. In previous studies, Hidost simply narrowed down the size of the feature set by only including features that occur in more than a certain number of files. Such a number is typically set to 1% of the training set size and the selection is performed "in hindsight" once for the entire dataset. However, this method failed to evenly include features from both benign and malicious files as malicious files tend to contain unique features. Hence, a huge portion of the feature set was occupied by features extracted from benign PDFs. Therefore, with the intention of including features extracted from malicious files, a novel feature selection process was applied. In short, the entire feature set was created by deliberately maneuvering the ratio between different pools of features to be used for training the target classifiers and PDF-GAN.

We segregated the entire features into four pools: (1) features found only in benign PDFs (2) features found more in benign PDFs than malicious ones. (3) features found more in malicious PDFs than benign ones. (4) features found only in malicious PDFs. Figure 4 shows the number of features from each pool. Any features that were found in less than that of either benign or malicious files were excluded. The main reason for applying the new feature selection process was to overcome the imbalance of dataset problem commonly found in DL that the existing mechanism failed to overcome as explained above. We gathered a set of 297 features with a balanced number of features from each pool. This method resulted

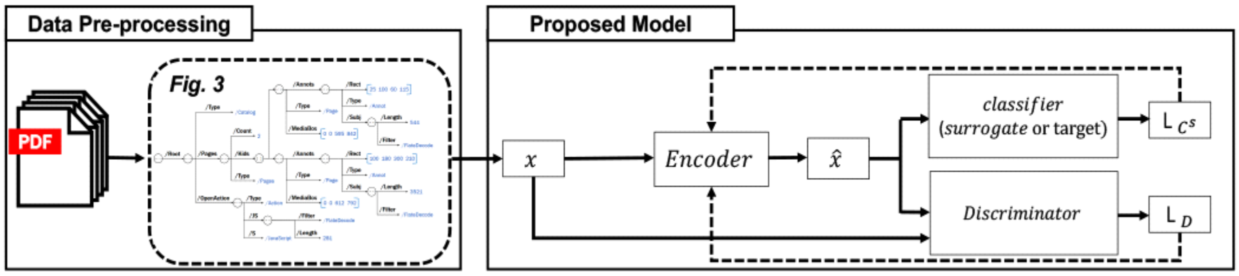


Figure 5: Model architecture.

in an improved detection performance as it will be illustrated in Section 4.2.

3.3 Seed Selection for Mutation

We have selected a total of 500 files after filtering out from Contagio malicious files. Table 1 shows how those files were chosen. Selecting seed PDF files to be fed into PDF-GAN was done with the dynamic analysis system (i.e., Cuckoo: the leading open-source automated malware analysis system). First, a primitive set of seeds was selected from the Conatgio data set, excluding files from the training set. After analyzing 6,105 files, it appeared that only 1,503 files indicated some malicious network activities. This result may have been caused by the inborn limitation of dynamic analysis. All of these files were parsed through PDFrw and the feature extraction process to validate their tree structure. A total of 1,485 files remained after this process. Upon close examination, we realized that many of the files shared the exact same value for the set of 297 features. Therefore, after filtering out homogeneous files, 712 remained. Finally, it was important that these files be classified as malicious by our target classifiers. The remaining files were put through all three classifiers sequentially and 99.6% of them were corrected classified as malicious, which demonstrates the high performance of our classifiers. Among 709 files, we randomly selected 500 for the evasion process to evaluate against previous studies.

3.4 Evading Model Architecture

Our training involved three parts: a generator, a discriminator and a surrogate classifier. The generator constructs a PDF close to the form of the input data and the surrogate classifier produces a prediction score of the malicious and benign PDF. The discriminator then predicts a confidence score on whether the data are from S' or not. The generator is trained with the original PDF to learn and create a variant version of the original PDF such that the prediction result of the generated data is a reverse of the original PDF.

The architecture of this model is illustrated in Figure 5. The generator takes an input x and gives an output $\hat{x}$, both of which are given to the discriminator and the surrogate classifier. The discriminator outputs the probability that $x = \hat{x}$ and the surrogate classifier outputs a prediction score given input $\hat{x}$. The learning objective of the generator is to minimize the prediction score of the surrogate classifier and the learning objective of the discriminator is to discriminate whether a generated PDF is the original x.

The generator accepts length n of a pre-processed PDF as input fed into the generator and constructs a variation of the PDF. The first layer consists of 64 filters with a length of 4; the second layer consists of 32 filters with a length of 2; the third layer consists of 16 filters with a length of 2. The fourth layer consists of 8 filters with a length of 2. Additional layers are then added in reverse order as the number of input length n. Batch normalization [25] is used at each layer and tanh [33] is used as the activation function at each layer, except for the final layer where ReLU [38] is used for the activation function. A sigmoid activation function is used to output the probabilities from the logits. The discriminator takes the output of the generator as input to determine whether the output is from S'. The first layer comprises n input feature size of filters; second $n*2$; third $n*4$; fourth $n*8$; fifth n input feature size of filters. Tanh is used at each layer as the activation, except for the final layer, where a sigmoid activation function is used to output probabilities from the logits. For the surrogate classifier, we constructed one layer network. The kernel size of each layer is 3 with stride 1 for all networks.

To define the learning objective, let L_G, L_D and L_{C^s} denote the loss of the generator, discriminator and surrogate classifier, respectively. The generator, G, aims to generate malicious PDF by learning data distribution close to the distribution of benign PDF. For each malicious input $x \in X$, G seeks a possible stochastic mapping to other representation, $\hat{x} = G(x; \theta_G) \in \hat{X}$ by conditional probability density function $p(\hat{x}|x)$, where θ_G denote the parameter for the generator. In original GANs, generator receives noise $z \sim p_z(z|\mathbb{Y})$, where $\mathbb{Y}$ is the class labels space. However, in our model, G receives noise z, which is computed by $x \times r \in \mathbb{R}^d$ where r, d are the random string and the feature dimension, respectively.

First discriminator, D, aims to distinguish malicious features by learning distribution of S', and the generator's input is required to retain the original form of maliciousness by computing reconstruction loss between generator's input and the output. Second discriminator (surrogate classifier), C^s, aims to evade the classifier by predicting prediction score given generator's output. Computed loss from both two discriminators are then back-propagated to the next step training step of the generator. For the white-box attack C^s can be replaced by the actual target classifier.

The generator then has the following objective functions:

$$\begin{aligned} \mathrm{L}_G &= d(x, G(x, \theta_G)) + ((1-\lambda_d) \cdot \mathrm{L}_D + \lambda_d \cdot \mathrm{L}_{C^s}) \\ &= d(x, \hat{x}) + ((1-\lambda_d) \cdot \mathrm{L}_D + \lambda_d \cdot \mathrm{L}_{C^s}), \end{aligned} \tag{1}$$

where $d(x, \hat{x})$ is the Euclidean distance between the generated and original data. In addition, λ_d is the weight parameter for the surrogate classifier that can be used to control the dependency level of the generation to maximize the diversity of the feature changes.

A discriminator has the sigmoid cross entropy loss of:

$$L_D = -y \cdot \log(D(x)) - (1-y) \cdot \log(1 - D(G(x, \theta_G))). \quad (2)$$

A surrogate classifier has an objective function of:

$$L_{C^s} = -||C^s(f(x)) - C^s(f(\hat{x}))||_2, \quad (3)$$

In the case where input x is classified as malicious by the surrogate classifier, $\hat{x}$ needs to be classified as benign. For this, we need to maximize the distance of the prediction score. Given the above equations, the generator optimizes a convex of Eq.1 finding Nash equilibrium of a min-max game between the G against both D and C^s.

3.5 PDF Repacking and Verification

Once PDF-GAN successfully evaded the classifiers with the mutated feature set, we must verify that the originally intended maliciousness was retained. For this verification, the mutated features must be applied to the original malicious PDF by the repacker. The repacker has three operations: *insert*, *replace* and *delete*. *insert* operation updates the dictionary according to the mutated feature set. A new key and value is inserted into PDF and the new value was in the form of real numeric value. For example, */Root/Pages/Rotate: None –> 0* means that the new path of */Root/Pages/Rotate* with a value of 0 is inserted into the tree representation. *replace* and *delete* operations are essentially operate in a similar manner. The former replaces the existing value with the new value and the latter deletes the key and the value pair (i.e., feature) from the dictionary hence deleting a path from the tree representation. All three operations are carried out while retaining the tree representation structure of PDF.

The most time-consuming aspect of finding evasive samples is repacking the mutated features back into proper PDF files. In addition to GANs reconstruction power, addition trick was considered to reduce the number of tries in repacking to reconstruct the PDF files. As explained in Section 3.1, if the dictionary was in an array from, the median value of elements was used instead. Therefore, in the repacking process, the modified feature value, which is a form of real numeric value, was reshaped into an original form (e.g., an array form). Consequently, it contributed in improving the evasion speed compared to the state-of-the-art evasion techniques, which is described in Section 5.4.

The reconstructed PDF file (i.e., repacked evasive sample) is tested in Cuckoo sandbox to verify that the maliciousness is maintained. The detail of this verification stage is explained in Section 4.4.

4 EXPERIMENT

For the experimental evaluation, a Cuckoo Sandbox 2.0.7 was arranged using 16 virtual machines (VMs) running Windows XP 32bit SP3 and Windows 10. Adobe Reader 8.1.1, PDF-XChange 2.5 and Foxit Reader 9.0.1 were installed in VMs. For the training, we used a Linux machine [Ubuntu 12.04, 2.6 GHz Intel Xeon E5-2690 (14 Cores) and 8 NVIDIA GTX TITAN V 12GB] without parallelization.

Classifier	Feature selection	Accuracy (%)	AUC	F1
SVM (Hidost '13)	Old	96.46	0.989	0.825
	New	**96.89**	**0.994**	**0.995**
Random Forest (Hidost '16)	Old	96.45	0.988	0.801
	New	**98.07**	**0.994**	**0.998**
Ensemble (PDFrate-v2)	Old	99.37	0.993	0.667
	New	**99.69**	**0.995**	**0.995**

Table 2: Detection accuracy of classifiers with new feature selection compared to old feature selection

4.1 Datasets and Model Training

We managed to gather PDF malware samples from VirusTotal. The dataset was collected on December 20, 2017 and on March 14 and June 19, July 17, 2018 and it consisted of 10,673 files in total. The Contagio dataset comprises a total of 9,109 benign files and 11,105 malicious files. In addition, CVE samples were collected from Exploit-db [13] where proof-of-concept (PoC) codes and files are uploaded. Six specific samples were used in the experiment.

For training PDF-GAN, we used an optimizer of the multi-class logarithmic loss function Adam [28] with a learning rate of 0.001, a beta rate of 0.5 and a mini-batch size of 16. The discriminator achieved optimal loss after 1,000 steps, whereas the generator required 1500 steps to generate original data similar to the sample. Most of these parameters and network structures were experimentally determined to achieve optimal performance. Randomly selected 5,000 benign and 5,000 malicious files from Contagio dataset were used for training classifiers and PDf-GAN.

4.2 Target Classifiers

Our target classifiers were Hidost '13 with a SVM model, Hidost '16 with a RF model both of which used Poppler as a parser and PDFrate-v2 with an ensemble method that used a customized parser. The feature extraction and selection process were reproduced according to an open-source code and each machine learning model was applied using scikit-learn. Well-known detection performance parameters such as accuracy, F1-score, and the area under the curve (AUC) were measured.

For Hidost, the test set comprised the Contagio dataset excluding the samples used for the training (i.e., 6,105 files) and PDF malware samples from VirustTotal (i.e., 10,673 files) that were not included in the training set also. For PDFrate-v2, we applied the same methodology as explained by the authors and the classifier was applied to the training set using 10-fold cross-validation. The results are presented in Table 2 under *Old*. However, with the help of our unique feature selection process we were able to achieve an even better detection performance as shown under the heading *New* in Table 2. The performance was measured with all varying factors including the training and test datasets fixed except for the feature selection process and the performance showed noticeable improvement across all fields. Hence, we can claim that, because our own classifiers performed better in terms of accuracy, AUC

CVE-ID	Type of PDF malware	Type of vulnerability	Application	Version	Exploited (Arbitrary code execution)
CVE-2008-2992 [1]	JavaScript	Buffer overflow	Adobe Acrobat & Reader	8.1.2	calc.exe & notepad.exe & message-box
CVE-2010-0188 [2]	File Embedding (.tiff)	Integer overflow	Adobe Acrobat & Reader	9.1	calc.exe
CVE-2010-2883 [3]	ActionScript	Buffer overflow (CoolType.dll)	Adobe Acrobat & Reader	9.3.4	calc.exe & notepad.exe
CVE-2010-3654 [4]	ActionScript	Flash (Memory corruption)	Adobe Acrobat & Reader	9.4	calc.exe
CVE-2011-2462 [5]	JavaScript	Flash (Buffer overflow)	Adobe Acrobat & Reader	9.4	calc.exe & message-box
CVE-2013-2729 [7]	File Embedding (.bmp)	Integer overflow	Adobe Acrobat & Reader	10.1.4	message-box
CVE-2017-13056 [9]	JavaScript	Improper validation of string	PDF-XChange	2.5	calc.exe
CVE-2018-9958 [11]	JavaScript	Use-After-Free	Foxit Reader	9.0.1	calc.exe

Table 3: Details of CVEs used in the experiment

Analysis Type	Description	Example
Network	Performs some HTTP requests	[GET http://www.deaf-video.de/3c55ea9320fcadfabb79d08f91bef510/.a1/load.php?e=2]
	DNS queries	[www.deaf-video.de]
	Transport IP addresses (UDP, TCP)	[UDP: 192.168.56.128:137 –> 192.168.56.1:137] [TCP:192.168.56.128:1292 –> 185.53.178.6:80]
	Network communications indicative of a potential or script payload download (API: URLDownloadToFileW)	[url: http://www.deaf-video.de/3c55ea9320fcadfabb79d08f91bef510/.a1/load.php?e=2] [filepath_r: C:\DOCUME 1\cuckoo\LOCALS 1\Temp\e.exe]
Behavior	One or more non-whitelisted processes were created	[C:\DOCUME 1\cuckoo\LOCALS 1\Temp\e.exe]
Static	The PDF open action contains JavaScript code	[<< /S /JavaScript /JS this.BXcfTYewQ() >>]

Table 4: Malicious signatures

and F1-score, it is reasonable to target the new classifiers instead of Hidost '13, Hidost '16 and PDFrate-v2.

4.3 CVEs for Various Types of PDF Malware

Common vulnerabilities and exposures (CVEs) provide publicly known security vulnerabilities in a reference-style. To widen the scope of capability in terms of evasion effectiveness, the CVEs described in Table 3 were included in the experiment in generating evasive samples. The set comprises three types of PDF malware, namely, JavaScript, ActionScript and File embedding. The types of vulnerability also varied widely from a buffer overflow to memory corruption and Use-After-Free. In addition, several different target applications were tested including Adobe Acrobat Reader and Foxit Reader. Adobe versions required for the successful attack range from 8.1.2 to 10.1.4. We implemented the attack using a few different codes to be executed when the PDF is parsed and viewed with the reader. It is important to note that these samples were not included in the training phase of the framework but only after PDF-GAN was fully trained these samples were fed into a trained PDF-GAN model for the purposes of generating evasive samples.

4.4 Malicious Signature

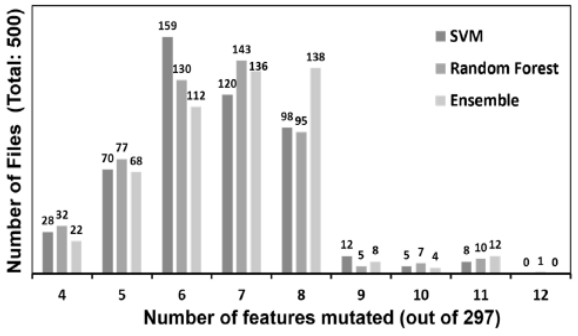
Maintaining the maliciousness of PDF files was an absolute necessity in confirming the evasive sample and completing the evasion of the classifiers. Had they lost maliciousness at any stage of the evasion process, the purpose of this study would have been negated. To check if mutated malicious PDF files still acted with malevolence, we leveraged Cuckoo sandbox. Cuckoo can analyze many different malicious files and trace API calls and the general behavior of files transformed into comprehensible signatures. Owning to the innate limitations of a dynamic analysis, the behavioral signatures varied even for the same file. Therefore, reliable malicious signatures were needed to confirm that the modified PDFs still maintained their maliciousness. There were three main types of analysis we paid special attention to network, behavior and static. Table 4 shows the types of signatures and their examples. We compared the analysis results between the original and modified versions. However, focusing only on the network behavior of the files would limit our work to malware related to network activities. Hence, unlike previous work, CVEs of all three types of PDF malware described in Section 2.2.2 were included in the experiment. After the modifications were made using trained PDF-GAN, the modified file was put through a tester stage. If it performed as originally designed as listed in Table 3, it was considered to be an evasive sample.

4.5 AntiVirus Engines (VirusTotal)

VirusTotal investigates submitted URLs or files with AntiVirus engines and reveals the detection result from each engine. Although PDF-GAN already proved its imposing capability in generating adversarial examples by evading open-source PDF malware classifiers, we further demonstrate its effectiveness by evading commercial AntiVirus engines. Tested AntiVirus engine version was the most recent update (i.e., 2020. April). The procedure for this attack consists of two stages: (1) Use generated AEs from Contagio dataset to check if any of them can evade AntiVirus engines (i.e., a transfer-based attack). (2) Generate variants of successful AEs to further improve the evasion rate for more AntiVirus engines. The result of stage 1 is illustrated in Section 5.5 and it shows that many AEs appeared to be also effective on numerous AntiVirus engines through a transfer-based attack (i.e., PDF-GAN is not trained to evade any of

Features	Contagio dataset						CVE-IDs					
	SVM		Random Forest		Ensemble		SVM		Random Forest		Ensemble	
	Operation	No.of files	Operation	No.of files	Operation	No.of files	Operation	No.of files	Operation	No.of files	Operation	No.of files
/Root/Type	['Insert', 'Change']	500	['Insert', 'Change']	500	['Insert', 'Change']	500	['Change']	14	['Change']	14	['Change']	14
/Root/Pages/Type	['Change']	500	['Change']	500	['Change']	500	['Change']	14	['Change']	14	['Change']	14
/Root/Pages/Rotate	['Insert']	500	['Insert']	500	['Insert']	500	['Insert']	14	['Insert']	14	['Insert']	14
/Root/Pages/Kids/Type	['Insert', 'Change']	500	['Insert', 'Change']	500	['Insert', 'Change']	500	['Change']	14	['Change']	14	['Change']	14
/Root/Pages/Kids/Resources/ProcSet	['Insert', 'Change']	396	['Insert', 'Change']	418	['Insert', 'Change']	442	['Insert', 'Change']	10	['Insert', 'Change']	13	['Insert', 'Change']	11
/Root/Pages/Kids/MediaBox	['Insert', 'Change']	269	['Change']	268	['Change']	268	['Change']	2	['Change']	5	['Change']	3
/Root/Pages/Kids/TrimBox	['Change']	234	['Change']	234	['Change']	234	-	-	-	-	-	-
/Root/Pages/Kids/Contents/Filter	['Insert', 'Change']	220	['Insert', 'Change']	204	['Insert', 'Change']	289	['Insert', 'Change']	5	['Insert', 'Change']	12	['Insert', 'Change']	6
/Root/Pages/MediaBox	['Change']	73	['Change']	73	['Change']	73	['Change']	2	['Change']	2	['Change']	2
/Root/Pages/Kids/CropBox	['Change']	18	['Change']	18	['Change']	18	-	-	-	-	-	-
/Root/Pages/Kids/BleedBox	['Change']	18	['Change']	18	['Change']	18	-	-	-	-	-	-
/Root/Pages/Kids/ArtBox	['Change']	18	['Change']	18	['Change']	18	-	-	-	-	-	-
/Root/Names/JavaScript/Names/JS/Length	['Change']	17	['Change']	17	['Change']	17	-	-	-	-	-	-
/Root/AcroForm/Fields/Kids/Kids/Rect	['Change']	8	['Change']	8	['Change']	8	-	-	-	-	-	-
/Root/Pages/Kids/Contents/Length	['Change']	7	['Change']	7	['Change']	7	-	-	-	-	-	-
/Root/Pages/Kids/Annots/Rect	['Change']	4	['Change']	4	['Change']	4	['Change']	2	['Change']	2	['Change']	2
/Root/OpenAction/Annots/Rect	['Change']	3	['Change']	3	['Change']	3	-	-	-	-	-	-
/Root/Pages/Kids/Annots/Subj/Length	['Change']	1	['Change']	1	['Change']	1	-	-	-	-	-	-

Table 5: Feature mutation result for Contagio dataset and CVEs



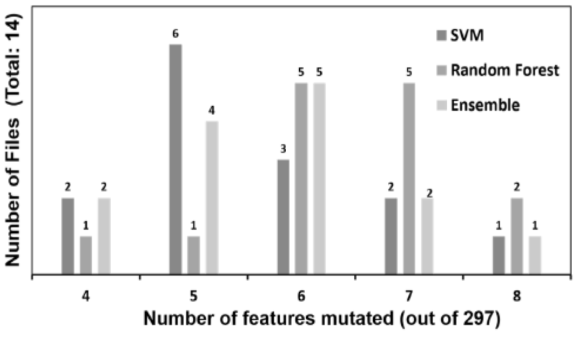


Figure 6: The number of features mutated to generate AEs for all 500 files selected from Contagio (top) and 14 CVE files (bottom)

AntiVirus engines). The variants of AEs were created by swapping malicious contents from other malware samples among detected as malicious by engines. This approach is rooted from the understanding that PDF-GAN successfully discovered PDF structure that can evade some engines and by only changing the contents, more AEs can be generated. The AntiVirus analysis result for 45 engines is shown in Section 5.5

5 RESULTS

In this section, the experimental result for evading PDF malware classifiers and maintaining maliciousness using both Contagio dataset and CVEs is described. Also, there was an drastic improvement in the time required to evade PDF malware classifiers for 500 samples. Finally, the result of an experiment to evade AntiVirus engines (Virustotal) is explained.

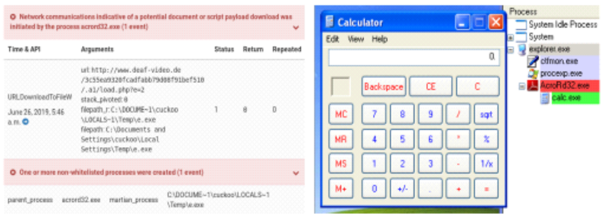
5.1 Feature Mutation Result for Contagio

The feature mutation results are shown in Table 5. All of 18 features that were manipulated in at least one file were from the set $S - S'$ as the desired maliciousness was maintained. There were four features listed at the top that needed to be altered in all of 500 files to evade the classifiers. In addition, none of the features were removed from the original files. We believe that this fact may have been crucial in retaining the maliciousness and passing the Cuckoo test phase on the first attempt. If any of the features from a set of features relevant to the malicious behavior (i.e., S') were modified, many malware files would have lost their original maliciousness.

As mentioned in Section 3.1, a value assigned as an integer value of 1 indicates that it was initially in a string type. Hence, if such a value is changed to any other value, the original meaning will be diminished. There were five cases in which the value was changed from 1 to 2:

{*/Root/Type*, */Root/Pages/Type*, */Root/Pages/Kids/Type*, */Root/Pages/Kids/Contents/Filter*, */Root/Pages/Kids/Resources/ProcSet*}

Another interesting observation is that to evade all classifiers, *insert* operation on */Root/Pages/Rotate* was required.


⊘ Network communications indicative of a potential document or script payload download was initiated by the process acrord32.exe (1 event)

Time & API	Arguments	Status	Return	Repeated
URLDownloadToFileW June 26, 2019, 5:46 a.m.	url: http://www.deaf-video.de /3c55ea9320fcadfabb79d08f91bef510 /.a1/load.php?e=2 stack_pivoted: 0 filepath_r: C:\DOCUME~1\cuckoo \LOCALS~1\Temp\e.exe filepath: C:\Documents and Settings\cuckoo\Local Settings\Temp\e.exe	1	0	0

⊘ One or more non-whitelisted processes were created (1 event)

parent_process	acrord32.exe	martian_process	C:\DOCUME~1\cuckoo\LOCALS~1 \Temp\e.exe

Figure 7: Cuckoo signature result (left), Arbitrary code execution result of CVE-2011-2462 (right)

Our approach, PDF-GAN, unlike EvadeML and EvadeHC, grasps the differences in the patterns of the features between benign and malicious files and opts only to modify the minimum number of features from the files. The number of mutations required for the files differed between learning algorithms, as shown in Figure 6 (top). Fewer than eight features were needed to be modified in more than 95% of the files.

To confirm that PDF-GAN truly deduced the distinction in the patterns of the features to incur the minimum number of feature manipulations, we partially modified the file according to PDF-GAN. For example, for a file that required five features to be modified, 31 partially modified variants were created. (i.e., 5 combination(C) 1 + 5C2 + 5C3 + 5C4+ 5C5 = 31). The result clearly showed that PDF-GAN provided the least number of modifications to complete finding evasive examples. For the case of PDFrate-v2 (Ensemble), generating evasive examples for all 500 original PDF malware at the point in which all features suggested by PDF-GAN were altered. Therefore, we can conclude that the our approach suggested all the features that were needed to be perturbed in order to evade the classifiers.

5.2 Feature Mutation Result for CVEs

Not surprisingly, the result of feature mutation for the CVE samples showed a significant similarity compared to that of the Contagio samples. Table 5 illustrates that all the modified features were among those in the results of the Contagio samples. The top four features that affected the entire Contagio samples were also affected by all 14 CVE samples. Also, no feature was deleted from the original malware. On top of this, the number of mutations requires to find evasive samples shows the same trend as illustrated in Figure 6 (bottom). This result confirms that PDF-GAN was trained to alter only those features that deceive the classifiers while preserving the intended maliciousness.

5.3 Malicious Signature Verification

The result of preserving a malicious signature was confirmed by Cuckoo and by manually running the CVE samples. An example of the results is shown in Figure 7. By analyzing the Cuckoo results, we confirmed that all signatures listed in Table 4 remained intact for all 500 seed samples, which verified the successful attempt of generating evasive samples. Moreover, all of the CVEs that contained all three types of PDF malware also operated according to the initial intention of the malware. The right side of Figure 7 shows that a calculator opened when the malicious PDF was read by Adobe

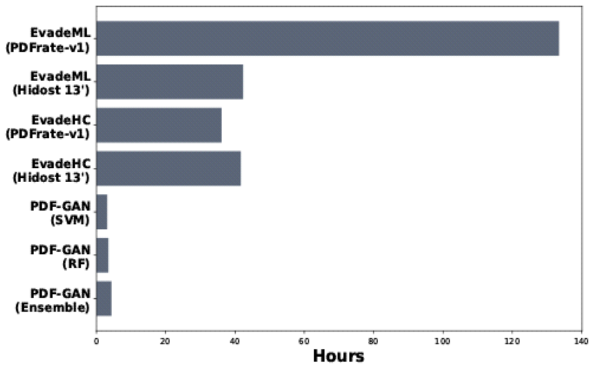


Figure 8: Time required to evade PDF malware classifiers for 500 selected malware files from Contagio

Reader. The attacker can customize the exploit to have any code executed at will.

5.4 Evasion Speed

As our approach tackles the problem by training PDF-GAN with the feature set, it was expected that the cost of execution in terms of evasion speed would be better than that of the previous work by EvadeML. Figure 8 illustrates the total time taken to identify an evasive variant for all 500 selected malware seeds. EvadeML employs a stochastic search based on a fitness function meaning that many possible variants are created to be tested on the oracle for their malicious signature. As the unit-cost of the Cuckoo sandbox testing was much higher than any other stages of the evasion, a huge portion of time spent by EvadeML was designated for oracle testing. However, our approach managed to avoid such overhead by utilizing PDF-GAN only to modify the non-relevant features of PDFs in maintaining malicious behavior. Thus, we needed to perform only the verification stage once to confirm that all variants maintained their malicious signature.

All stages of our evasion process are shown in Figure 8, including parsing of files, feature extraction, feature selection, training PDF-GAN, inference and finally testing the possible evasive sample with the Cuckoo sandbox. As expected, the stages that occupied the largest portion were the training and inference stages with PDF-GAN. A total of 102 minutes was spent on the classifier with the SVM model (e.g., Hidost '13), 127 minutes on the classifier with the RF model (e.g., Hidost '16) and 180 minutes on the classifier with the ensemble model (e.g., PDFrate-v2). They correspond to 55%, 61% and 69% respectively of the total time taken, respectively. The more robust the detector, the longer it took for PDF-GAN to be trained and to infer the modified version of PDFs.

In comparison to the total time taken for EvadeML to evade Hidost '13, our approach achieved the 100% evasion rate within about 3 hours, which is more than 13 times faster in evading a SVM model. Moreover, even when PDF-GAN aimed to evade a more advanced classifier with the ensemble model, it achieved the full evasion 30 times faster than EvadML targeting PDFrate-v1, which is merely a RF model. In addition, we compared the time required for full evasion against the EvadeHC algorithm. The total evasion time

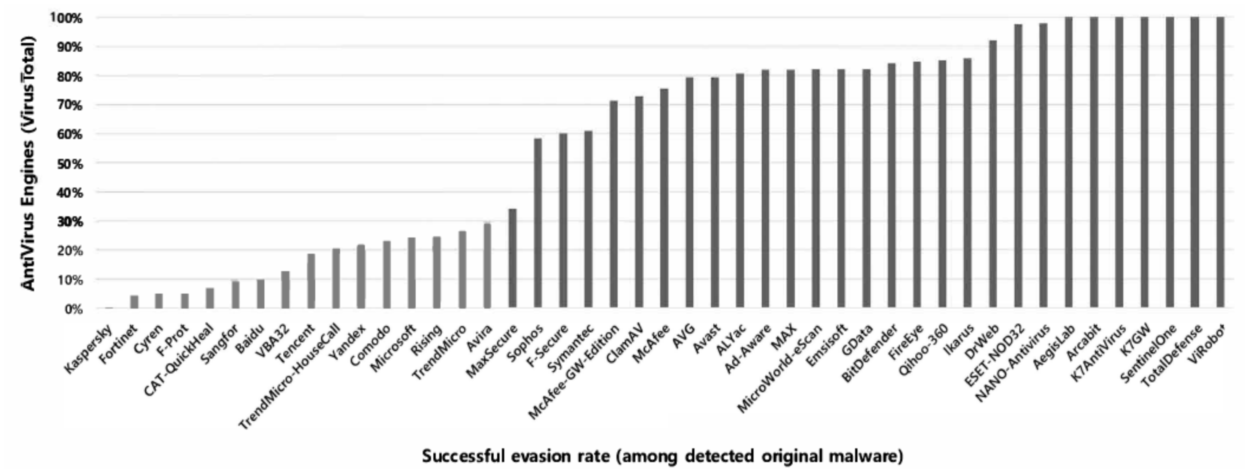


Figure 9: Evasion rate of AntiVirus engines by generating variants of AEs in Table 6

against Hidost '13 was measured according to the author explaining that the average time taken to generate a single evasive sample was 5 minutes. In relation to PDFrate-v1, the relative time taken was measured for evading all 500 seed samples. Surprisingly, a contrast to EvadeML, the time taken to generate all evasive samples for Hidost was greater than that of PDFrate-v1 with the EvadeHC algorithm. In comparison to our PDF-GAN model, it managed to achieve the full evasion more than 13 times faster for a SVM model. It is important to note that while in our experiment setup, only 16 virtual machines were implemented, EvadeHC utilized 216 virtual machines. This implies that PDF-GAN could achieve the full evasion in a much shorter time if the same number of virtual machines were used for the testing phase.

5.5 AntiVirus Engines (VirusTotal) Result

All 500 malware samples selected for previous experiments and AEs that PDF-GAN generated to evade three classifiers were uploaded to VirusTotal for analysis from AntiVirus engines. The result is summarized in Table 6 and it shows the number of malware files detected and AEs for each engine. It is important to notice that AEs discovered for each engine is from the AEs that PDF-GAN generated while evading Hidost '13, Hidost '16 and PDFrate-v2. We observed that generated AEs were still effective for the AntiVirus engines (i.e., a transfer-based attack) and a total of 19 engines appeared to be vulnerable to this transfer-based attack.

Furthermore, we created the variants of those AEs by swapping malicious contents from malware files and Figure 9 illustrates the successful evasion rate in 45 AntiVirus engines. Few engines were excluded as they did not support analysis for PDF format malware. As all 500 selected malware samples contain unique maliciousness, we defined that finding 500 AEs which collectively contains those maliciousness represents 100% evasion rate. 100% evasion rate was achieved in 7 AntiVirus engines and 60% for 45 engines in average. These results showed that if the experiment did not rely on a transfer-based attack (i.e., if PDF-GAN is trained to evade AntiVirus engines), even higher evasion rates can be achieved.

AntiVirus engines (VirusTotal)	No. of original files detected	No. of AEs
AegisLab	472	54
Arcabit	445	135
Avira	500	48
BitDefender	495	3
Comodo	472	49
DrWeb	279	1
ESET-NOD32	471	109
Emsisoft	492	1
F-Prot	483	4
F-Secure	495	78
GData	498	3
K7GW	138	5
McAfee-GW-Edition	498	1
MicroWorld-eScan	497	1
Microsoft	457	13
Sangfor	304	1
SentinelOne	500	2
TotalDefense	344	42
ViRobot	496	429

Table 6: Number of malware files detected (out of 500) by AntiVirus engines and adversarial examples (AEs) by a transfer-based attack

6 DISCUSSION

While PDF-GAN is effective under the black-box assumption that reflects its effectiveness, one can design a defensive mechanism for a more robust detection system. Similar to any other evasive techniques, multiple submissions to the detector is required to acquire a classification score. Therefore, if the defender decided to limit the number of submissions for a single peer, it would hinder the performance of the evasive technique. Moreover, the defender can opt to retrain the detector model with newly submitted files. Using

Type of Adversarial attacks	Fast Gradient Method (FGM)	Projected Gradient Descent (PGD)	Basic Iterative Method (BIM)	Carlini&Wagner (CW)
Is classifier evaded?	Yes	Yes	Yes	No
Is malicious signature maintained?	No	No	No	No
Is PDF repacking successful?	No	No	No	Yes
Is AntiVirus engines (VirusTotal) evaded?	No	No	No	No
Avg. No. of features mutated	126	120	121	29

Table 7: Different type of ML attacks

newly submitted files, the detector model can be retrained and can employ recent ML approaches for continual learning [29, 35, 42, 46] in which ML is used to continuously learn without loss of acquired knowledge on previous tasks.

In a strong black-box scenario where the number of queries is limited, it is imperative that PDF-GAN still shows a promising performance with extremely small number of queries to the classifier. With single layer surrogate model, we managed to achieve 9.6% transfer rates. This is an improvement from other query limited black-box attacks which showed 3.4% transfer rates [34]. If more queries are allowed to the target classifier, PDF-GAN can immensely improve the transfer rate.

In a white-box scenario, other types of adversarial examples may become applicable for generating evasive samples. This is, such as FGM or CW, for computing an AE in the features space and map it back to obtain an evasive document. However, most of the existing ML attacks were incapable of maintaining malicious signatures while easily evading classifiers, as shown in Table 7. Patterns that can easily be flipped by adversarial perturbations is known as non-robust features [24]. The aforementioned AEs can easily compute adversarial perturbations to evade classifiers by flipping non-robust features. However, none of these approaches use reconstruction loss to preserve to maintain original PDF behavior, which often resulted in a crash. Consequently, GANs reconstruction loss was necessary to conserve the original's PDF behavior.

The notion of robust and non-robust features are defined by [24]. Robust features correspond to patterns that are predictive of the true label even when input is adversarially perturbed. Conversely, non-robust features correspond to patterns that are also predictive but can easily be flipped by adversarial perturbations. ML models use both features to minimize the training loss; thus, flipping non-robust features will have a huge impact on their prediction accuracy.

To mitigate AEs, Goodfellow et al., [20] introduced an algorithm, called adversarial training, that is robust to AEs by retraining them. In the same sense, antivirus vendors can prevent such adversarial attacks by collecting mutated examples and updating their detectors. However, once the detectors have been updated, attackers can also retrain PDF-GAN that exploit target detectors. It was observed that new AEs remained undetected by the updated detectors. In our experiments, among 500 mutation seed files, some evasive PDFs were successfully uploaded to Gmail server. We believe that it is also possible to consider Gmail the target detector under the strong black-box scenario.

In the process of denoting PDFs in a tree structure form and in the process of extracting and selecting a feature set from the tree structure, we observed that there is considerable loss of information concerning the PDFs. For example, the most recent detectors select few features from a large group of features to be used in the training phase. We believe that the performance of feature extraction on the PDFs is directly associated with the generation of performance. Therefore, eliminating the handcraft of such a process can increase the performance of the PDF detectors. A similar story is also true for the evasion scenario. Once all information can be represented and abstracted without the application of any handcraft for training GANs, a considerable increase in the performance of evading malware classifiers can be observed.

DL has been applied to several forms of high-dimensional data to denote a low-dimensional Euclidean space and recently DL has been expanded even to non-Euclidean spaces such as graphs [16]. Word2Vec is a representative algorithm that generates a low dimensional embedded vector that corresponds to a high-dimensional word based on the mutual occurrence frequencies of words. Similarly, there are algorithms for embedding a graph [39] that maps a graph region to a low-dimensional region while preserving the adjacency and structure of the nodes. Graph2Vec is one of the representative algorithms used for graph data-driven learning approaches. Embedding algorithms such as Word2Vec and Graph2Vec may be suitable candidates for denoting all features of the PDF without any handcraft.

7 CONCLUSION

We introduced a novel approach for generating evasive PDF samples. In addition, we introduced a new technique for selecting a feature set that even includes unique features lurked in malicious files through pooling. As a consequence, we achieved a better detection performance than the state-of-the-art open source PDF malware classifiers. Finally, our approach was based on the black-box threat model in which the attacker is extremely restricted with information concerning the target classifier. We evaluated our approach using over 10,000 PDF documents from VirusTotal and over 20,000 PDF documents from Contagio and successfully discovered evasive samples for all unique 500 selected samples. We analyzed our model on commercial virus engines in addition to the traditional benchmark. By generating evasive sample through PDF-GAN, we identified significant flaws in some AntiVirus engines in the black-box scenario attack. We determined that one antivirus engine may outperform the rest, but this does not guarantee that it would be the best detector. While evasion attacks are still possible on commercial AntiVirus engines, we suggest that the use of multiple detectors will prevent such attacks.

REFERENCES

[1] 2008. MITRE. CVE-2008-2992. (2008). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-2992
[2] 2010. MITRE. CVE-2010-0188. (2010). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-0188
[3] 2010. MITRE. CVE-2010-2883. (2010). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2883
[4] 2010. MITRE. CVE-2010-3654. (2010). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-3654
[5] 2011. MITRE. CVE-2011-2462. (2011). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2462
[6] 2013. Milia Parkour "16,800 clean and 11,960 malicious files for signature testing and research". (2013). http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html
[7] 2013. MITRE. CVE-2013-2729. (2013). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-2729
[8] 2015. Weilin Xu, PDF-Malware-Parser: A fork of pdfrw aiming at parsing PDF malware. (2015). https://github.com/mzweilin/PDF-Malware-Parser
[9] 2017. MITRE. CVE-2017-13056. (2017). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-13056
[10] 2018. FreeDesktop.org. 2018. Poppler. (2018). https://poppler.freedesktop.org/
[11] 2018. MITRE. CVE-2018-9958. (2018). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-9958
[12] 2019. SONICWALL DETECTS, REPORTS DRAMATIC RISE IN FRAUDULENT PDF FILES IN Q1 2019. (2019). https://www.sonicwall.com/news/sonicwall-detects-reports-dramatic-rise-in-fraudulent-pdf-files-in-q1-2019/
[13] 2020. Exploit-Database. (2020). https://www.exploit-db.com/
[14] Scott Alfeld, Xiaojin Zhu, and Paul Barford. 2016. Data Poisoning Attacks against Autoregressive Models.. In *AAAI*. 1452–1458.
[15] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. 2009. Scalable, behavior-based malware clustering.. In *NDSS*, Vol. 9. Citeseer, 8–11.
[16] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).
[17] Marco Cova, Christopher Kruegel, and Giovanni Vigna. 2010. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceedings of the 19th international conference on World wide web*. ACM, 281–290.
[18] Hung Dang, Yue Huang, and Ee-Chien Chang. 2017. Evading classifiers by morphing in the dark. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 119–133.
[19] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2013. Compa: Detecting compromised accounts on social networks.. In *NDSS*.
[20] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[21] Mario Heiderich, Marcus Niemietz, Felix Schuster, Thorsten Holz, and Jörg Schwenk. 2012. Scriptless attacks: stealing the pie without touching the sill. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 760–771.
[22] Weiwei Hu and Ying Tan. 2017. Generating adversarial malware examples for black-box attacks based on GAN. *arXiv preprint arXiv:1702.05983* (2017).
[23] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).
[24] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*. 125–136.
[25] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).
[26] Jiyong Jang, David Brumley, and Shobha Venkataraman. 2011. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 309–320.
[27] Georgios Kakavelakis and Joel Young. 2011. Auto-learning of SMTP TCP Transport-Layer Features for Spam and Abusive Message Detection.. In *LISA*. 18–18.
[28] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.
[30] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).
[31] Pavel Laskov et al. 2014. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE symposium on security and privacy*. IEEE, 197–211.
[32] Pavel Laskov and Nedim Šrndić. 2011. Static detection of malicious JavaScript-bearing PDF documents. In *Proceedings of the 27th annual computer security applications conference*. ACM, 373–382.
[33] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer, 9–48.
[34] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2016. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770* (2016).
[35] David Lopez-Paz et al. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*. 6467–6476.
[36] DAVIDE MAIORCA, BATTISTA BIGGIO, and GIORGIO GIACINTO. 2018. Towards Adversarial Malware Detection: Lessons Learned from PDF-based Attacks. *arXiv preprint arXiv:1811.00830* (2018).
[37] Davide Maiorca, Igino Corona, and Giorgio Giacinto. 2013. Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 119–130.
[38] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.
[39] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).
[40] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).
[41] Moheeb Abu Rajab, Lucas Ballard, Noé Lutz, Panayiotis Mavrommatis, and Niels Provos. 2013. CAMP: Content-agnostic malware protection. (2013).
[42] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016).
[43] Joshua Saxe and Konstantin Berlin. 2015. Deep neural network based malware detection using two dimensional binary program features. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 11–20.
[44] Johannes Schlumberger, Christopher Kruegel, and Giovanni Vigna. 2012. Jarhead analysis and detection of malicious Java applets. In *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 249–257.
[45] Hovav Shacham et al. 2007. The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86).. In *ACM conference on Computer and communications security*. New York, 552–561.
[46] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*. 2990–2999.
[47] Charles Smutz and Angelos Stavrou. 2012. Malicious PDF detection using metadata and structural features. In *Proceedings of the 28th annual computer security applications conference*. ACM, 239–248.
[48] Stavrou Angelos Smutz, Charles. 2016. When a Tree Falls: Using Diversity in Ensemble Classifiers to Identify Evasion in Malware Detectors.. In *NDSS*.
[49] Nedim Šrndic and Pavel Laskov. 2013. Detection of malicious pdf files based on hierarchical document structure. In *Proceedings of the 20th Annual Network & Distributed System Security Symposium*. Citeseer, 1–16.
[50] Nedim Šrndić and Pavel Laskov. 2016. Hidost: a static machine-learning-based detector of malicious files. *EURASIP Journal on Information Security* 2016, 1 (2016), 22.
[51] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2013. Shady paths: Leveraging surfing crowds to detect malicious web pages. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 133–144.
[52] Weilin Xu, Yanjun Qi, and David Evans. 2016. Automatically evading classifiers. In *Proceedings of the 2016 network and distributed systems symposium*. 21–24.
[53] Zhenlong Yuan, Yongqiang Lu, Zhaoguo Wang, and Yibo Xue. 2014. Droid-sec: deep learning in android malware detection. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 371–372.

Patient similarity learning via Bi-CNN with attention module

발표자: 강승연(연세대학교)

PATIENT SIMILARITY LEARNING VIA BI-CNN WITH ATTENTION MODULE

As a fundamental topic in the healthcare field, discovering similar patient has relative positive effect on personalized healthcare scenarios. However, high complexity, irregular, hierarchical, sequential characteristics of EHR data cause difficulties for measuring similarity directly. In previous studies, most of them ignored the temporal information of EHR data. The purpose of this paper is to obtain a vector representation of patient historical records with consideration of temporal characteristics and to learn a method of patient similarity metrics. In this paper, we introduce an attention module into the BI-CNN framework to implement 'end-to-end' learning, which can simultaneously obtain vector representation and similarity metric for patient longitudinal historical records. The experiments are based on real world data and results show that introducing the attention module can improve the effectiveness of similarity learning. In the end, we also verify the similarity metric learned by the proposed model is reasonable in the actual situations.

Keywords: patient similarity learning; patient representation; BI-CNN; attention mechanism; CBAM

1. INTRODUCTION

Due to the maturity of big data technology and the increasing medical requirements, the similarity study of patients with general predictive ability is an important part of precision medicine, which has its increasingly broad application scenarios. Patient similarity analysis establishes a group of similar patients by measuring the distance between patients and predicts the target patient's condition by common phenotype of similar cohort. Specifically, patient similarity analysis refers to the selection of clinical concepts as features of patients in a designated medical environment, and quantitative analysis of the distance of a series of medical concepts, which could dynamically measure the distance between patients, determine patient cohort to target patient. In the medical big data scenario, the various features of the patient cohort can theoretically provide multiple predictions, which has better universality than the model for specific predictive goals.

EHR data is a complex collection of various medical concepts that can be broadly divided into structured data and unstructured data. Unstructured one has no fixed structure and usually composed of text, images, and signals, while structural data is stored in a table, which is more applicable for computer processing than unstructured data. EHR data is hierarchical, heterogeneous and sequential. 'Sequential' means that the patient's condition changes with time, that is, the observation results are different at different time. 'Hierarchical' means that there may be multiple medical concepts under each observation. For example, under a visit of a patient's profile, multiple medications and multiple procedure may occur. 'Heterogeneous' means that the difference between each patient's profile can be huge, as the frequency of each patient's visit to the hospital and the medical events under each visit may vary widely. These reasons described above often lead to difficulties in patient similarity analysis.

The study of patient similarity has a positive influence on the advancement of personalized medicine. For patients, the development of patient similarity can provide a more accurate personalized medical strategy. For doctors, it can provide supplementary for decision-making, providing case basis for more diverse treatment plans. For medical systems, accelerating the development of treatment from proficiency-based art to data-driven science. There are many previous studies on patient similarity analysis. Some work obtains the patient's vector representation through statistical values of the EHR data. This approach does not consider that the EHR data is sequential and irregular, and ignore the temporal property. Some work uses the method of data integration to achieve the effect of dimensionality reduction, and then analyzes the similarity of patients. These methods are not 'end-to-end 'for the analysis of similarity, and require domain knowledge on feature selection. This limits the ability to generalize the model. There are also some works based on supervised similarity learning. When defining supervision information, these tasks only consider whether the patient has the same or the same disease, but the patient's disease development is also a sequential data, this definition obviously loses temporal information.

In order to solve the mentioned problems and challenges, our main purpose is to measure the similarity of complex patient history records more accurate through models. In this work, by converting the similarity learning problem to a supervised binary classification problem, we propose a network framework for learning the similarity metric between patient pairs, which consists of two parts: the former part is based on the Bi-CNN structure to obtain the vector representation of the patient's historical information, and the latter part is measuring patient similarity based on the supervised metric learning method. The validation of the model is based on real data, and the results demonstrate that our approach is effective in improving the performance of subgroup the patient cohort. The main contributions of this work include: (1) Introducing an attention module into CNN structure to improve the representation of patient historical record. (2) Considering the temporal characteristics when defining supervised information, rather than just considering whether patients have same one or several diseases. (3) Based on the network structure of Bi-CNN and distance metric learning, the 'end-to-end' learning process is achieved, which means the patient similarity metric is obtained as well as the patient's representation.

2. RELATED WORK

2.1 Patient Representation

Converting complex, irregular, time-series EHR data into appropriate patient expression is the first step in the secondary use of EHR data. Due to lack of domain knowledge, statistics are often used to assess the information for each feature. For example, the TF-IDF method, TF means term frequency, and IDF means inverse document frequency, the product of which can be used to assess the importance of a clinical concept in a medical record in a data set. The importance of concept increases proportionally with the number of occurrences in the patient's medical record, but at the same time decreases inversely with the frequency of its occurrence in other patients' medical records. Roque *et al.,* (2011) used a vector consisting of ICD-10 codes for disease diagnosis to express individual patients, and used TF-IDF method to assign weights to ICD-10 codes: the greater weight, the higher the more important information is. Then they used the cosine value to calculate the distance between patients, and use the average connected hierarchical clustering method to divide the patients into 307 sub-groups, exploring the types and quantities of diseases included in each sub-group. Wang *et al.,* (2015) utilized the diagnosis, medication and lab tests as features to form the feature matrix, which is normalized by the TF-IDF method, using the unsupervised and the supervised method to dimensionality reduction, respectively. The patient similarity is determined by Euclidean distance. Wang *et al.,* (2012) Extracted features from diagnostic codes, drug components, and 1035 laboratory tests, using TF-IDF method to normalize feature matrix, Euclidean distance is used to define the degree of similarity between patients.

However, the above studies did not consider the temporal characteristics of EHR data when acquiring patient representation. With the development of deep learning in recent years, due to its superior performance in processing longitudinal data, some work use neural networks to obtain patient representation and apply it on the corresponding tasks. Miotto *et al.,* (2016) used a three-layer stack of denoising auto-encoders as a deep learning framework to obtain patient representation. The experiment was based on actual data and was evaluated by two different tasks, disease prediction and patient classification to validate of the model. Choi *et al.,* (2016) established vector representations for medical concepts in the longitudinal EHR data, and the vector representation of each visit is obtained from the sum of the vectors of the medical concepts occurring under that visit. In the work of Choi *et al.,* (2016), Med2vec is proposed to obtain the vector expression of each visit by using a multi-layer perceptron. This vector expression also considers two levels of embedding: medical concept level and visit level. Not only does this work enhance the effectiveness of predictive tasks, the learned interpretable representations are also meaningful to apply on other problems. Bajor *et al.,* (2018) used a patient-level embedding method to obtain the vector representation of the historical record of each patient, and the validity of the representation was validated on three prediction tasks. The advantage of this expression is that it is lightweight and can quickly get the expression of each patient. Baytas *et al.,* (2017) proposed a model that uses a new Time-Aware LSTM module to process irregularly spaced longitudinal EHR data and obtain effective patient representation by capturing independencies in the sequence. The patients were then divided into subgroups using K-means clustering. Zhang *et al.,* (2018) proposed a patient2vec framework, by adding an attention mechanism to the LSTM structure learning an interpretable, complete, personalized patient representation based on real data to predict whether the patient will be hospitalized within 6 months. The result of experiment indicates that the method is better than a series of baselines.

2.2 Patient Similarity Learning

Besides using existing distance metrics to directly calculate patient similarity, patient similarity problems can be also translated into supervised metric learning problems. The purpose of supervised patient similarity learning is to use feedback from expertise to learn a metric that more accurately measures the similarity between patients. Sun *et al.,* (2012) applies the idea of LMNN (Weinberger *et al.,* 2006) algorithm to patient similarity, and proposes a local supervised similarity learning algorithm LSML. The algorithm uses the label that expert giving for each pairwise patient as the supervisory information, which identifies the homogeneous and heterogeneous neighbors within a certain distance to the target patient by comparing the labels. Then, by narrowing the homogeneous neighbors and pushing away the heterogeneous neighbors, a generalized Mahalanobis distance is learned. Zhan *et al.,* (2016) proposed a framework of patient similarity learning, which obtains a low-dimensional sparse similarity matrix by introducing Group Lasso (Yuan *et al.,* 2006) into the objective function. This low-rank mapping avoids the shortcomings of the similarity learning algorithm that computational cost is relatively large.

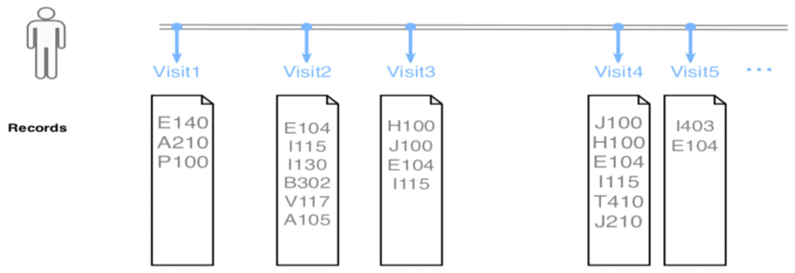


Figure 1. An illustration of EHR data

Huai *et al.,* (2018) proposed a patient similarity learning framework UnPSL, using maximum likelihood estimation to obtain the parameters in the similarity function. In addition to this, two regular items were introduced in order to implement sparse feature selection and uncorrelated feature selection. Suo *et al.,* (2018) proposed mtTMSL to measure the progression of patient similarity, which is a multi-task triple constrained sparse metric learning framework. Each task is a distance metric that learns at a future time point. The optimization is based on triple constrains, which achieves metric learning by bringing similar patients closer together and dissimilar patients far away from a margin.

However, though these studies improved the algorithm of metric learning, they ignored the temporal relationship of EHR data when establishing patient representation. Recently, there are some works implemented that learning the similarity measure of patients and obtaining patient representation with considering temporal property of longitudinal EHR data simultaneously. Zhu *et al.,* (2016) represented the longitudinal EHR data of each patient by a matrix, which is constructed by stacking equal-length embedded vector representations of each visit as the input to the network, which preserves the expression of the temporal characteristics of the longitudinal data. Based on the structure composed of BI-CNN and supervised metric learning, a matching matrix is used to calculate the similarity between the two embedded vectors which derived from deep network. The evaluation of the model is clustering based on the learned similarity metric, and the result is better than the baseline methods. Suo *et al.,* (2017) proposed a time-fusion CNN framework that not only preserves the local temporal relationship, but also obtains a global contribution of different time intervals for patient similarity measurement. The work also made predictions for target patient according to the phenotype of the k nearest patients. Suo *et al.,* (2018) proposed a CNN-based triple patient similarity learning framework to learn a margin that better separates similar and dissimilar patients. The experimental results show that the patient similarity learning framework is superior to the comparative distance metric method. Ni *et al.,* (2017) proposed a depth metric learning framework PSDML to achieve a fine-grained patient similarity metric by optimizing the quadruple loss function. The model is validated by multi-label KNN classification of the target patient on the real data set. The result shows that the method can slightly improve the performance of KNN. Since most of the work is aimed at exploring whether the patient pairs are similar or not, there were not sufficient studies on similarity of the patient's condition accounted for multiple diseases. Therefore, Zhao *et al.,* (2018) proposed a BI-CNN network structure to learn the similarity measure between patient pairs, and then determined whether the patients have the same phenotype on multiple diseases.

2.3 Attention Mechanism

The attention mechanism is derived from the human visual attention mechanism, and the purpose is to select information that is more critical to the corresponding task. Using limited attention to capture useful information greatly improves the efficiency and accuracy of visual information processing. Attention mechanism combined with CNN and RNN structure has achieved good performance in the field of natural language processing and computer vision (Yin *et al.,* 2016, Woo *et al.,* 2018, Luong *et al.,* 2015). With the development of deep learning in the medical field, the attention mechanism has also been applied to various medical tasks. Zhang *et al.,* (2018) introduced a hierarchical attention mechanism into the LSTM structure to predict patient hospitalization. The hierarchical attention mechanism consists of within-subsequence and subsequence-level attention. Furthermore, they also analyzed disease correlations from individual and population level. Suo *et al.,* (2017) segmented each patient matrix representation and applied convolution and pooling operations to obtain vector representations for each segmentation. Through learning the weight for each vector, the overall vector representation of each patient is obtained. The patient similarity learning process is based on this weighted representation. The experimental results showed that the patient representation obtained by adding the attention mechanism in network is superior than the ones without attention on classification and clustering problems.

3. MODEL AND METHOD

In this part, we will introduce how to use the irregular EHR data as the input of the supervised similarity learning network, how to define the supervision information, and introduce the overall network, an 'end-to-end' framework based on a BI-CNN architecture with attention module. The vector representation and similarity metric is obtained simultaneously.

3.1 Matrix representations of historical records

The goal of this section is to define the input to the model. As mentioned earlier, since the original EHR is hierarchical, time-series, irregular data, as shown in Figure 1. If a similarity analysis is to be performed based on the patient's history, the original EHR data needs to be transformed from a sequential patient medical record consisting of medical concept codes into a numerical form that is easily processed by the computer. The most primitive approach is to use one-hot representation, but this high-dimensional representation is difficult to reflect the implicit relationship between medical concepts. In addition, for each patient, since EHR is longitudinal data, the temporal property should be considered in when obtaining patient representation. Therefore, we use the matrix to represent the patient's historical medical records, which is obtained by stacking the same length of embedded vector for each visit.

The historical record of patient p can be represented by a matrix P with size $e \times v$, which e is the pre-defined embedding dimension. v is the number of visits associated to the patient historical profile. The vector representation for each visit is obtained by element-sum of the vector representations of all medical concepts under the that visit where the vector representation of each medical concept is obtained by using the embedding method in natural language processing However, what we are concerned about is representation of overall representation of patient profile, since patient historical records can be regarded as an ordered collection of visits and the sequential visit is fixed according to the access date, in order to get a representation which could capture the temporal information from EHR data, the equal-length representation of visits are concatenate along the visit order. For the sake of measuring patient similarity, there is a must to get a vector representation for each patient. There are some previous studies obtain the low-dimensional, dense vector representation for patient profile by regarding patient historical records as a document, and the visits can be regarded as words like in natural language process. Through the document-level embedding process (Bajor *et al.,* 2018, Le *et al.,* 2014), we can not only obtain the visit level embedding vector, but also same dimensional vector representations for patient historical records. In our work, each visit includes medical concepts, and also two demographic information, gender and age. Therefore, for each visit can be represented as a binary vector, the dimension of which is the total number of all medical events over the population in the dataset. Going along with the network structure of doc2vec in (Le *et al.,* 2014), any fixed-length representation for visit level as well as patient-level can be obtained simultaneously. So far, we can obtain a matrix representation of the patient historical profile, and also get a vector representation of each patient profile through document level embedding.

3.2 Vector representation of patients

3.2.1 CNN based framework

CNN has outstanding advantages in feature extraction because of its powerful representation ability. The feature extraction is performed on local information through convolution operation. The whole drama information is obtained by integrating local information. The feature detection layer of CNN learns through training data, avoiding explicit feature extraction but implicitly learning from training data. Furthermore, due to the sharing of local weights, the complexity of the network is reduced, which is also a major advantage of convolutional networks over fully connected networks. In this section we will show how to get the embedded vector representation of the patient's history matrix through the CNN structure.

Feature extraction is first performed through the convolutional layer. Since our goal is to explore the relationship between accesses in the patient matrix, a 1D filter is used here to perform a convolution operation on the access dimension of the patient's matrix representation. More specifically, the information in the matrix history map is extracted by using different filters, and the parameters in each filter are obtained through network optimization. For each filter $w \in \mathbb{R}^{h \times e}$, where h represents the size of the filter, which is the number of consecutive visits used to generate features in the convolution operation. Where e is the embedded dimension for each access defined in the previous work. By performing a feature extraction with temporal meaning through a convolution operation, we have (1), where $*$ represents the convolution operation, $b \in \mathbb{R}$ represents the error term, $P_{i:i+h-1}$ represents the concatenation of the i_th to the $(i+h-1)\_th$ visit vectors, and the result c_i represents a feature obtained after the convolution operation of $P_{i:i+h-1}$.

$$c_i = ReLU(w * P_{i:i+h-1} + b) \tag{1}$$

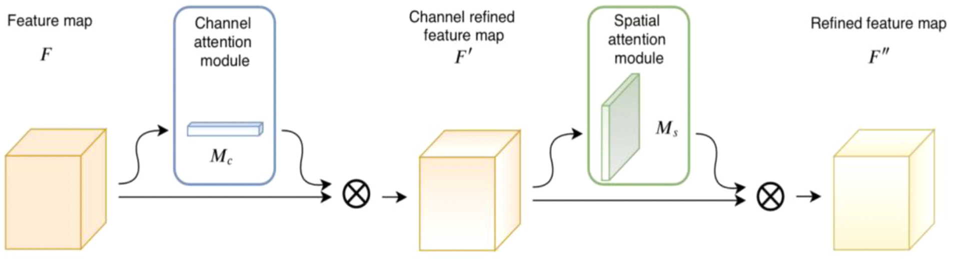


Figure 2. CBAM architecture

Applying the filter to the entire patient matrix representation with stride of 1, we can obtain $v - h + 1$ features, where v is the number of visits. Through these features we can get a feature map c:

$$c = [c_1, c_2, c_3, \dots, c_{v-h+1}] \tag{2}$$

The obtained feature map is reduced by the pooling layer. Here we use the max pooling, the purpose is to extract the most important information in the feature map to get a reduced feature map, which can reduce the computational complexity, and also make the feature representation more robust and avoid over-fitting.

3.2.2 Convolutional block attention module

Through the above method, the matrix representation of the patient history records can be converted into a vector by convolution and the pooling layer. However, although the basic convolution operation can capture the temporal information, there is a limitation that the information is equally treated during this process. In order to further improve the ability of patient vector representation, we introduce an attention mechanism in the convolutional layer in the network. CBAM (Woo *et al.*, 2018), which combined with the spatial level and channel level attention module, but not significantly increase the computation and parameters. Under the premise of quantity and parameter quantity, the feature extraction capability of the network can be improved for corresponding task.

The purpose of the channel attention module is to explore which channels in the feature map are meaningful. While the purpose of the spatial attention module is to determine the network to understand which parts of the feature map have a higher response at the spatial level. There are m different sizes of filters in the convolutional layer, each filter contains n channels, and after convolutional layer, m different feature maps can be obtained. Take one of the feature maps as an example to illustrate the process of CBAM, as shown in Figure 2. On the feature map, a channel level attention module and a spatial level attention module are sequential generated, which respective denotes as M_c, M_s. Then, the element-wise multiplication $\otimes$ is performed with the input feature map of each module to obtain the output feature map F', F'':

$$F' = M_c(F) \otimes F \tag{3}$$

$$F'' = M_s(F') \otimes F' \tag{4}$$

3.2.3 Supervised similarity learning

Patient similarity is derived by an appropriate metric in a specific medical scenario. Given the patient's vector representation, there are many distance metrics that can be used to calculate similarities between patients, e.g. Euclidean distance. However, due to the differences of different dimensions, using Euclidean distance means that the features of each dimension are of equal importance in the distance calculation. To cope with this problem, it is necessary to learn a distance metric for a specific task according to the data. Similarity learning can be divided into two categories, supervised and unsupervised distance metric learning. In this paper, the patient similarity problem is converted to a supervised distance metric learning problem. By optimizing an objective function, a metric matrix is obtained. In the new characteristic space, the distribution of similar patients become more closely and dissimilar ones further.

Given the vector representations of two patients $x_i, x_j \in \mathbb{R}^d$, using the method of calculating the similarity in (Zhan *et al.,* 2016), we have (5), where $M \in \mathbb{R}^{d \times d}$ is a symmetric semi-positive matching matrix, which is optimized during training process.

$$similarity(x_i, x_j) = x_i^T M x_j \tag{5}$$

The supervised distance metric learning problem usually solved as a binary classification problem. The patient pairs are classified as two categories, those have been diagnosed by same diseases are regarded as similar pairs, otherwise dissimilar. The model is optimized by solving the binary classification problem of whether the input patient pair is similar or not. For the classification problem, we use cross-entropy as the loss function (6), where N is the number of patient pair, $\hat{y}_k$ is the prediction of the k_th patient pair, y_k is the supervised information of the kth patient pair, which means they are similar or dissimilar.

$$\mathcal{L} = \frac{1}{N}\sum_{k=0}^{N}\left(y_k log(\hat{y}_k) + (1 - y_k) log(1 - \hat{y}_k)\right) \tag{6}$$

However, in the EHR data, there is no ground truth on the similarity between patients. Thus, how to obtain supervision information for supervised distance metric learning needs to be considered. In fact, supervision information on patient similarity should be given by experts according to the corresponding domain knowledge. However, for huge-volume EHR data, the workload on manual labeling is huge, and the similarity given by different experts also could cause bias. In order to label the patient pairs, previous studies indirectly obtained supervision information from the similarity of the physician's diagnostic information to the patient. Specifically, whether the patient pair has same one or multiple diseases or not. The reason for this is that the diagnostic information can be regarded as a summary of the patient's overall condition at that visit, moreover, the diagnosis information of a patient over a period of time can be regarded as a summary of the expert's evaluation on the patient. Hence, it is reasonable that obtain supervisory information from this. However, the diagnostic information is also sequential information with temporal characteristics. In practice, only by considering whether the patient pair has the same disease is not sufficient. Therefore, for the sake of sequential property, we use the longest common subsequence (Rivault *et al.,* 2017) as the basis for defining the supervision information y, which is based on the similarity of the disease trajectory over time. Where d_i, d_j denote as the diagnostic trajectory of patient i, j , LCS denotes the longest common subsequence, $|\cdot|$ denotes the number of elements in the sequence.

$$y = \begin{cases} 1, if \left|LCS(d_i, d_j)\right| \geq 2 \\ \quad 0, otherwise \end{cases} \tag{7}$$

3.2.4 Overall architecture

The network structure is shown in Figure. 4, the process of transforming embedded matrix representations which contain the temporal information of patient historical profiles into vector representations and performing similarity learning through a CNN structure with CBAM is implemented. The preceding part of this BI-CNN based network is able to accept the two characteristic matrices of patient pair, which is weight sharing. For each branch, we use two CNN blocks to extract the features from the original matrix. In each CNN block, we draw on the idea in the inception module (Szegedy *et al.,* 2015) which implement feature extraction by increasing the width of the network. As shown in Figure 5, the aim of different convolutional layer is extracting different features, and max pooling layers are used to reduce the intermediate representation. Then the feature maps derived from different branch is concatenate together as a feature map for next step processing. Besides, we introduce CBAM on each CNN block to enhance the capability of representations. The $1 * 1$ convolutional operation is used to aggregate the cross-channel information so that reduce the dimension of the feature map to obtain the vector representation. The patient similarity is calculated by a matching matrix based on the patient vector representations. In the end, the fully connected layer is used to derive whether the input patient pairs are similar or not. This 'end-to-end' network

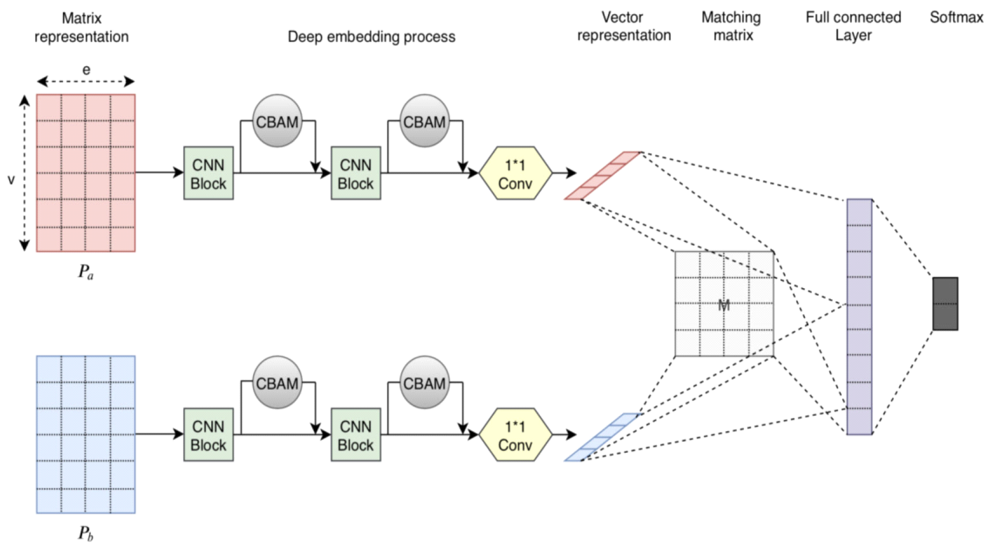


Figure 3. Overall framework of proposed method

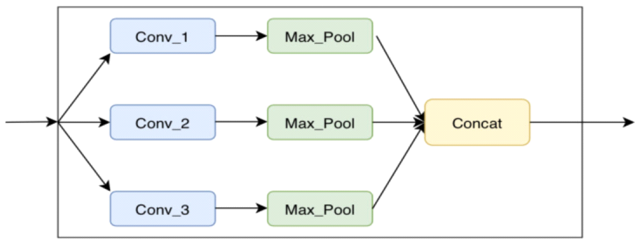


Figure 4. CNN Block

structure has many advantages: no need to intervene feature extraction process manually, optimize the parameters in the network for the specific task through the original input directly, and increase the consilience of the model.

4. EXPERIMENTS

There are 4.8 million people in South Korea having diabetes in 2018 (Bae *et al.*, 2018). As a prevalent chronic disease, diabetes and its complications have heavy threat to people's health. We conduct experiments on real word dataset, learning the similarity metric for diabetic patients, and validate the learned similarity metric is reasonable in real medical situation.

4.1 Datasets

The data is provided by Korea National Health Insurance Service and covers 12 years of EHR data. We filter the diabetes related visit records according to patient's diagnostic information, and sort visits with the same patient ID in order of date and then extracted gender, age, and medical concepts of medication, procedures, sub diseases as well as diagnostic information to reconstruct the EHR dataset for diabetic patients. In order to avoid bias caused by extreme data, those patients with fewer than 3 visits and visits with more than 12 drugs, 6 tests, and 10 sub diseases are dropped out.

4.2 Experiment setting

In this section we will introduce the implementation details of the model, the baseline methods to be compared with and the validation experiments.

4.2.1 Historical record embedding

In order to obtain a matrix representation of the patient's history, we first use the demographic information and medical events in the EHR data as input features to obtain equal-length vector representations of visits, including gender, age group, medication, procedure, and other diseases. Specifically, the closely-spaced visits are more diagnostic correlation along with patient's condition. We use doc2vec to obtain the distributed representations of visits. Experiments are performed on two models, distributed memory and distributed bag of words, and the embedding dimensions were set to 128 and 256, respectively. We choose distributed memory model with embedding dimension $e = 128$ which caused better performance. Thus, each patient's historical records can be converted from sequence of visits which represented by various medical codes in the raw dataset to a matrix representation by concatenate those equal-length visit vectors. In the real EHR data, the visit times for each patient is very likely to be different. For the reason of same size input characteristics of CNN, we use padding method to obtain a fixed-size matrix by zero vectors. This matrix based patient representation not only aggregates demographic information and historical records on variety of medical events, but also preserves the temporality of historical records.

4.2.2 Model implementation

Firstly, through training the model, we can obtain the optimized parameters in the network. The performance on the test set proves that our model can enhance the ability on patient similarity learning. We compare the performance of the proposed model with the one without the attention module. In addition, we also compare with the model using L2-norm for measuring similarity in the network instead of using the matching matrix for the purpose of validating our model that the ability to measure similarity can be enhanced. We randomly select 1.2 million patient pairs as experimental data, and the ratio of training set to test set was 9:1. The model is implemented by MXNET (Chen *et al.,* 2015) and optimized by adam (Kingma *et al.,* 2014).

4.2.3 Validation experiment

In order to more intuitively verify the validity of the proposed model for patient similarity measurement, we use the learned similarity measure method in two experiments, respectively.

The purpose of the experiment 1 is to determine the similarities between patients within and between groups of diabetic and non-diabetic patients. We select 10 diabetic patients as the sample of diabetic patients group and do the same to non-diabetic patients. Through the proposed model, the intra-group similarity calculation is conducted for the patients within the diabetic patient group and the non-diabetic patient group, and the similarity calculation is also conducted for the patients from two different groups. Experiment 2 includes two sections which respective represents as experiment 2-1 and 2-2. The purpose of experiment 2 is to investigate intra and inter-group similarities of diabetic patients in different age groups. We select 10 diabetic patients from the ages of 40 years old, 40 to 60 years old, and over 60 years old. The similarity of the patients within each group and between these three groups are calculated by the proposed method.

The rationality of learned similarity measure method is validated by the relationship between the similarities within and between these patient groups.

4.3 Results

4.3.1 Similarity learning results

Since we convert the similarity learning problem to a supervised binary classification problem, we compare the proposed model with the performance of baseline work under the binary classification task, and the results show that the proposed model is outperformed to baseline methods. We also train the network that with same structure of our proposed except the attention mechanism, CBAM. Furthermore, we also compare to the Bi-CNN networks which uses the L2-norm to measure the distance of with and without the CBAM. We choose recall, precision and F1 score as the performance indicators. The experimental result is shown in Table 1.

In Table 1, the models with using L2-norm for measuring the embedded patient vector representation distance is inferior than the performance which based on the matching matrix. Because the distance metric based on L2-norm is operated in the original space, whereas the matching matrix can map the data to a new space according to the similarity label. To optimize the parameters in the matrix, the matching matrix can map the data in the original space to a new one, in which the patient pairs can be classified more accurately. In addition, we have found that networks with attention mechanism perform better than those that do not, which indicates that the introduce the attention mechanism has a positive effect on feature extraction on specific tasks. In general, by solving the problem of patient similarity learning into a supervised binary classification problem, our proposed model can not only improve the representation ability of embedded vector through the attention mechanism, but also learning distance metric from supervised information through matching matrix, which also leads to a better performance.

Table 1. Performance of similarity learning process

	CNN_L2	CNN_CBAM_L2	CNN_matrix	CNN_CBAM_matrix
Recall	0.8735	0.8713	0.8802	0.9064
Precision	0.8735	0.8756	0.8821	0.9067
F1 score	0.8712	0.8688	0.8801	0.9064

4.3.2 Validation results

First, we calculate the similarity between the patients in the diabetic group, the similarity between the non-diabetic patient group, and the similarity between the patients from these 2 groups respectively. The similarity between patients with diabetes is generally higher than the similarity between non-diabetic patients. The similarities between diabetic patients and non-diabetic patients are extremely low. To quantify the results, the statistical values of the similarities of each set of experiments are shown in Table 2. The similarity between diabetic patients is relatively higher than that between non-diabetic patients and between diabetic and non-diabetic patients. This is because the patient historical records in the diabetic group show more correlations on medication, sub-diseases and procedures. Since the patients in the non-diabetic group were randomly selected from the whole dataset, it is obvious that they involved few similar medical events in their profile. In addition, the similarity between the diabetic patient group and the non-diabetic patient group is also relatively low, but higher than that in the non-diabetic patient group. This is due to the variety of complications caused by diabetes. These complications and their associated therapeutic drugs may coincide similar with the drugs for non-diabetic patients. For instance, diabetic patients with both high blood pressure and those with only high blood pressure will have similar conditions in the treatment of hypertension and cardiovascular diseases. However, these slight similarities could not make big contribution to the patient similarity measure based on the patient's entire historical profile, as the impact of these factors are minor to the overall history of patients of non-diabetic patients.

Table 2. Statistical values of diabetic patient and non-diabetic patient similarity (experiment 1)

	Diabetic patient	Non-diabetic patient	Diabetic & Non-diabetic patient
Mean	0.280	0.000	0.000
Max	0.998	0.000	0.005
Min	0.000	0.000	0.000

We also conduct similarity validation experiment for diabetic patients under different age groups. We selected 10 patients from each age group of the age under 40, 40-60 and over 60 as samples, and calculated the intra-group similarity of these three groups of diabetic patients. The statistical values are shown in Table 3. From the results that the overall differences of three groups is not significant. However, the mean value of the similarity within the group under 40 years old was slightly lower than that of the other two groups, and the difference in the average of intra-group similarity between 40 to 60 years old and over 60 years old is not significant. This is because that among diabetic patients under the age of 40, there are patients with type 1 diabetes, which is usually caused by hereditary factors. Although the difference of treatment for type 1 diabetes is small, the difference in sub-sick is quite large, resulting in a large difference in the therapeutic drugs for sub-sick. The average intra-group similarity of these three groups is generally slightly higher than that of the diabetes patient group in validation experiment 1, because age is an important factor for the general condition of diabetic patients. The similarity of patients in different age groups was slightly higher, which also validates that our proposed similarity measure can reflect the objective factor that age is related to the progression of diabetes.

Table 3. Statistical values of diabetic patient similarity in different age group (experiment 2-1)

	Diabetic patient under 40	Diabetic patient between 40-60	Diabetic patient over 60
Mean	0.367	0.470	0.469
Max	0.998	0.998	0.999
Min	0.000	0.000	0.000

In addition, we use the proposed method to measure the similarity between diabetic patients from three different age groups. There is little difference with the distribution of similarity within each age group. The statistical values of patient similarity between different age groups is shown in Table 4. In Table 4, the average similarity involved with the diabetic patients who are from under 40 years old age group is slightly lower than that between 40 to 60 and 60 years old. By removing type 1 diabetes patients in the group of under 40 years of age, we found that the average similarity between the group and the age group of 40-60 increased from 0.361 to 0.395, as well as the similarity of the group over 60 years old and under 40 years old increased from 0.360 To 0.440. Therefore, this slight change is also due to the fact that some patients with diabetes under the age of 40 are caused by type 1 diabetes, which is more dissimilar to the patients in other two groups. The relatively high degree of similarity between the groups of diabetic patients aged 40 to 60 and over 60 years old is due to the overlap of treatments for sub-sick in older patients with diabetes. Moreover, the mean of the average similarity between the three groups is 0.4096, which is basically consistent with the mean similarity between the diabetic patients which ignore the age difference in validation experiment 1.

Table 4. Statistical value of diabetic patient similarity between different age groups (experiment 2-2)

	Diabetic patient under 40 & Diabetic patient between 40-60	Diabetic patient under 40 & Diabetic patient over 60	Diabetic patient between 40-60 & Diabetic patient over 60
Mean	0.361	0.360	0.489
Max	1.000	0.750	0.997
Min	0.000	0.000	0.000

In validation experiments, we considering the distribution of similarity for patients in the diabetic and non-diabetic patient groups. The similarity of intra diabetic patient group is much higher than that of non-diabetic patients. When considering the similarity of different age groups of diabetic patients, the distribution of similarity slightly differed with age groups, but the average similarity is basically the same with the mean similarity value between the diabetic patients which ignore the age difference in validation experiment 1. Therefore, the proposed model is generally reasonable for measuring patient similarity in real situation.

5. CONCLUSION

Temporal information plays an important role in the patient's historical records, especially for the patients with chronical diseases. In order to capture the temporal information in the patient historical records and measure the similarity between patients, it is necessary to obtain a vector representation of the patient historical records, which contains temporal property. Patient similarity learning is based on pairwise constraint to learn a metric matrix that can effectively measure the similarity between patients. Compared with the basic distance metric, patient similarity learning can make the patient representation transform to a new characteristic space according to the learned matching matrix, which causes that the distances between similar patients are more compact, whereas dissimilar patients samples distributed far away. The matching matrix for measuring similarity between patient pairs is learned by converting the similarity learning problem to a supervised binary classification problem. The parameters in matching matrix is optimized through training process. In this work, we use matrix representation to preserve the temporal characteristics of longitudinal EHR data, and to improve the ability of patient vector representation to improve the similarity learning performance by introducing attention mechanisms in the BI-CNN based framework. In the end our experimental results validate that the proposed method can not only improve the accuracy of patient similarity learning process, it is also verified by validation experiments that the learned similarity measure is reasonable in the practical scenario. In future work, we will use the learned similarity metric to implement the task of subgrouping patients in real medical scenario.

REFERENCES

Bae, J. C. (2018). Trends of Diabetes Epidemic in Korea. Diabetes & metabolism journal, 42(5), 377-379.

Bajor, J. M., Mesa, D. A., Osterman, T. J., & Lasko, T. A. (2018). Embedding complexity in the data representation instead of in the model: A case study using heterogeneous medical data. arXiv preprint arXiv:1802.04233.

Baytas, I. M., Xiao, C., Zhang, X., Wang, F., Jain, A. K., & Zhou, J. (2017, August). Patient subtyping via time-aware LSTM networks. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 65-74). ACM.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., ... & Zhang, Z. (2015). Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274.

Choi, E., Bahadori, M. T., Searles, E., Coffey, C., Thompson, M., Bost, J., ... & Sun, J. (2016, August). Multi-layer representation learning for medical concepts. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1495-1504). ACM.

Choi, E., Schuetz, A., Stewart, W. F., & Sun, J. (2016). Medical concept representation learning from electronic health records and its application on heart failure prediction. arXiv preprint arXiv:1602.03686.

Huai, M., Miao, C., Suo, Q., Li, Y., Gao, J., & Zhang, A. (2018, May). Uncorrelated patient similarity learning. In Proceedings of the 2018 SIAM International Conference on Data Mining (pp. 270-278). Society for Industrial and Applied Mathematics.

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

Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196).

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Miotto, R., Li, L., Kidd, B. A., & Dudley, J. T. (2016). Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. Scientific reports, 6, 26094.

Ni, J., Liu, J., Zhang, C., Ye, D., & Ma, Z. (2017, November). Fine-grained patient similarity measuring using deep metric learning. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (pp. 1189-1198). ACM.

Rivault, Y., Le Meur, N., & Dameron, O. (2017, June). A Similarity Measure Based on Care Trajectories as Sequences of Sets. In Conference on Artificial Intelligence in Medicine in Europe (pp. 278-282). Springer, Cham.

Roque, F. S., Jensen, P. B., Schmock, H., Dalgaard, M., Andreatta, M., Hansen, T., ... & Jensen, L. J. (2011). Using electronic patient records to discover disease correlations and stratify patient cohorts. PLoS computational biology, 7(8), e1002141.

Sun, J., Wang, F., Hu, J., & Edabollahi, S. (2012). Supervised patient similarity measure of heterogeneous patient records. ACM SIGKDD Explorations Newsletter, 14(1), 16-24.

Suo, Q., Ma, F., Yuan, Y., Huai, M., Zhong, W., Gao, J., & Zhang, A. (2018). Deep patient similarity learning for personalized healthcare. IEEE transactions on nanobioscience, 17(3), 219-227.

Suo, Q., Ma, F., Yuan, Y., Huai, M., Zhong, W., Zhang, A., & Gao, J. (2017, November). Personalized disease prediction using a cnn-based similarity learning method. In 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 811-816). IEEE.

Suo, Q., Zhong, W., Ma, F., Ye, Y., Huai, M., & Zhang, A. (2018, November). Multi-task sparse metric learning for monitoring patient similarity progression. In 2018 IEEE International Conference on Data Mining (ICDM) (pp. 477-486). IEEE.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

Wang, F., Hu, J., & Sun, J. (2012, November). Medical prognosis based on patient similarity and expert feedback. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012) (pp. 1799-1802). IEEE.

Wang, F., & Sun, J. (2015). PSF: a unified patient similarity evaluation framework through metric learning with weak supervision. IEEE journal of biomedical and health informatics, 19(3), 1053-1060.

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research, 10(Feb), 207-244.

Woo, S., Park, J., Lee, J. Y., & So Kweon, I. (2018). Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 3-19).

Yin, W., Schütze, H., Xiang, B., & Zhou, B. (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. Transactions of the Association for Computational Linguistics, 4, 259-272.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68(1), 49-67.

Zhan, M., Cao, S., Qian, B., Chang, S., & Wei, J. (2016, December). Low-rank sparse feature selection for patient similarity learning. In 2016 IEEE 16th International Conference on Data Mining (ICDM) (pp. 1335-1340). IEEE.

Zhang, J., Kowsari, K., Harrison, J. H., Lobo, J. M., & Barnes, L. E. (2018). Patient2Vec: A Personalized Interpretable Deep Representation of the Longitudinal Electronic Health Record. IEEE Access, 6, 65333-65346.

Zhao, F., Xu, J., & Lin, Y. (2018, November). Similarity Measure for Patients via A Siamese CNN Network. In International Conference on Algorithms and Architectures for Parallel Processing (pp. 319-328). Springer, Cham.

Zhu, Z., Yin, C., Qian, B., Cheng, Y., Wei, J., & Wang, F. (2016, December). Measuring patient similarities via a deep architecture with medical concept embedding. In 2016 IEEE 16th International Conference on Data Mining (ICDM) (pp. 749-758). IEEE.

AMH-CNN: An Automatized Multi-step Hierarchical Convolutional Neural Networks Based on Data Inheritance

발표자: 서이안(이화여자대학교)

AMH-CNN: An Automatized Multi-step Hierarchical Convolutional Neural Networks Based on Data Inheritance

Yian Seo[a], Kyung-shik Shin[b]

[a] Dept. of Big Data Analytics, Ewha Womans University, 52 Ewhayeodae-gil, Seodaemun-gu, Seoul, 03760, Korea, yiansseo@gmail.com, +82-2-3277-3767

[b] Ewha School of Business, Ewha Womans University, 52 Ewhayeodae-gil, Seodaemun-gu, Seoul, 03760, Korea, ksshin@ewha.ac.kr, +82-2-3277-2799

Abstract — Multi-class image classification can be ambiguous to separate among similar classes as some classes share more similarities than others. This can be solved by using hierarchy, that is, by performing image classification reflecting hierarchical structure of categories. After arbitrarily defining hierarchy of the dataset in heuristic way based on human knowledge, the classifier using Convolutional Neural Networks (CNN) can classify the classes in hierarchical order by outputting multi-step classifying results. However, this vanilla hierarchical Convolutional Neural Networks model requires heuristic process prior to the CNN classifier, this model can be further developed by making the whole process automatized. This automation can be achieved by using data inheritance. Some of data consisting the dataset will share some feature, which can be denoted as data inheritance. By applying clustering method before classification, the model can first find hierarchy of the dataset in data-driven way, and then use the hierarchy to classify into levels of classes. This proposed model is Automatized Multi-step Hierarchical Convolutional Neural Networks (AMH-CNN). AMH-CNN is a single automatized model of finding shared hidden similarities in data, defining hierarchy by itself, and then outputting classification results reflecting hierarchical structure of the dataset. This study has contribution as a knowledge-embedded classifier outputting hierarchical information because this is the first study automatizing multi-step classification process by data-driven way to define hierarchy of training data categories based on data inheritance. Our implementation using VGGNet on Fashion-MNIST dataset has shown that AMH-CNN achieves better classification results than our benchmark results.

Keywords — Convolutional Neural Networks; hierarchy; multi-step; automation; image classification

1. INTRODUCTION

1.1. Motivation of Research

As large amount of image data has been surged with the help of search engines and social network services, fashion industry has been also flooded with considerable image usage, e.g. catalog images, fitting images, and product images on online shop, and therefore, there have been diverse necessity and application of image analyzing methodologies in the fashion field. First, it can be used in apparel segmentation (Hu et al. 2008). Second, it can be used in apparel recognition (Bossard et al. 2012), (Wang and Zhang 2011, Eshwar et al. 2016). Third, apparel classification and content-based retrieval can help users by product search and recommendation (Alzu'bi et al. 2017, Liu et al. 2012, Liang et al. 2015, Hara et al. 2016). Fourth, apparel classification can be carried out for tagging products (Bossard et al. 2012, Eshwar et al. 2016).

These applications can be implemented by conventional machine learning methods such as Support Vector Machine (SVM), Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) in the past studies (Taigman et al. 2014). In more recent studies, with large enough training data and the advanced computational power with multiple cores of CPU and the usage of GPU, deep learning methods such as Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN), are introduced to solve the issue of lack of capacity to handle large-scaled and unstructured data (Tsantekidis et al. 2017). These deep learning methods have been processing estimation and classification problems such as financial market prediction, speech recognition, hyperspectral image classification, medical image retrieval, image captioning, face recognition systems, objects detection, disaster management, age recognition, and adult content filtering (Karathanasopoulos et al., 2018, Iliukovich-Strakovskaia et al. 2016, Muhammad et al. 2018, Qayyum et al. 2017, Tsantekidis et al. 2017, Wehrmann et al. 2018, Yu et al. 2017).

For apparel classification tasks using CNN, several considerations need to be made for these applications. First, elaboration in the apparel classification algorithm is needed due to apparel property. Some apparel classes have similar characteristics and can be ambiguous such as pants and tights (Eshwar et al. 2016, Hara et al. 2016). And these apparel images can be taken in different conditions such as variations in the angle and light, cluttered backgrounds, occlusions made by other objects or subjects, and deformation by stretched or folded manners (Wang and Zhang 2011). Second, classification process should reflect the hierarchy of apparel categories, that is, to present multiple output results of hierarchical labels instead of presenting single final output label. Hierarchical structure has been recently discussed on image recognition in a few studies. As explained in the study of Zeiler and Fergus (2014), lower layers of CNN extract lower-level features, e.g. salient blobs and edges of the shape in the dog image, and higher layers extract sophisticated features, e.g. detailed characteristics of face and body in the dog image. In this sense, we can apply hierarchical categorization on layers composing CNN. Lower layers of CNN can represent coarse categories, while higher layers can do fine-grained subcategories. This idea of hierarchical classification has been suggested in earlier studies with MNIST and CIFAR datasets (Yan et al. 2015, Zhu and Bain 2017), and in recent study with Fashion MNIST dataset (Seo and Shin 2019).

1.2. Research Objectives

With consideration of above research contexts, multi-class classification has long been studied in previous studies (Malakooti and Zhou 1994) as it is tricky to separate among similar classes because some of classes share more similarities than others. Therefore, a novel approach is needed for multi-class apparel classification to separate far different classes first and then to discriminate among similar classes. In other words, it is to reflect hierarchical structure in classification process outputting multi-step classifying results. This multi-step hierarchical classifier is proposed as Hierarchical Convolutional Neural Networks (H-CNN) in the previous study of Seo and Shin (2019). This model requires prior process of defining the hierarchy. First, the hierarchy of apparel category needs to be defined in heuristic approach. The hierarchy can be arbitrarily defined following the general apparel category system used in apparel business, for instance, classifying into tops and bottoms, and so on. After this prior process, then the heuristic hierarchy is used to train H-CNN classifier.

However, in this study, we suggest that this multi-step hierarchical classification with prior process can be automatized into single algorithm by using the property of data inheritance. We cannot pinpoint which data groups share more similarities and can be clustered together, but some data groups are closer to each other, and to find this shared pattern, for example, Artificial Neural Networks (ANN) including CNN has been used to find hidden feature in data (Baesens et al. 2003). Based on this property of data inheritance, these groups of data can be clustered into hierarchical group by data-driven approach using clustering algorithm such as k-means. Therefore, in this study, we propose Automatized Multi-step Hierarchical Convolutional Neural Networks (AMH-CNN), which combines the process of defining hierarchy of apparel category and training the classifier based on this hierarchy into single automatized algorithm based on data-driven approach. Fig 1. describes the proposed concept of automation of defining the hierarchy and training the classifier.

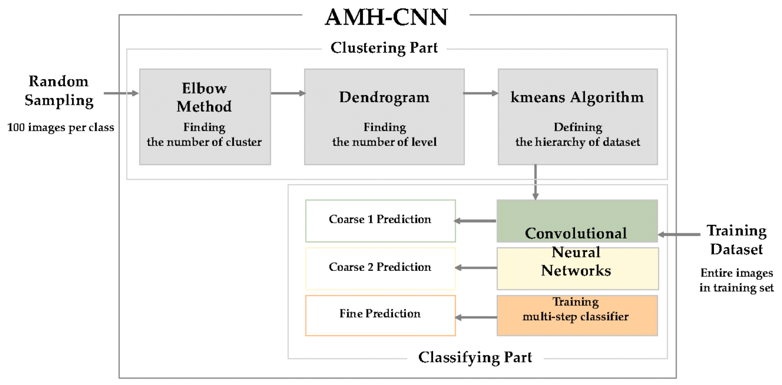


Fig. 1. Flowchart of Automatized Multi-step Hierarchical Convolutional Neural Networks (AMH-CNN).

Therefore, the proposed model no longer requires the prior process of heuristically defining the hierarchy of the dataset to train the CNN classifier as the model itself performs clustering the apparel category groups using k-means algorithm and defines the hierarchy of apparel dataset prior to train the classifier based on the clustered results. Furthermore, our hypothesis is that the classification results of hierarchical apparel classifier can be improved when the classifier is trained with hierarchy that is driven by clustering algorithm rather than hierarchy defined by human knowledge.

The contribution of this paper can be that as an extension study of applying hierarchical classification of apparel images using CNN, we are proposing knowledge embedded multi-class classification model rather than simple classification model outputting single value. Previous studies have been limited to vanilla classification resulting single outcome value for each class. However, this multi-step model presents multiple outcome values of three levels reflecting hierarchical structure to improve the error and inference related issues of multi-class classification. That is, by separating far different classes first, we can reduce the classification error and can also ease the interpretation of softmax values. Another contribution can be that, to our best knowledge, this is the first study automatizing the heuristic process of defining the hierarchy of training data groups prior to training the classifier. Moreover, we hypothesize that the accuracy of hierarchical classification model can be improved when the hierarchy to train the model is defined by clustering algorithm based on data-driven approach than heuristically defined based on human knowledge.

The remainder of this paper is organized as follows. Section 2 summarizes several related works. Section 3 gives an overview of proposed method. Section 4 introduces dataset and experiment design. Section 5 presents results of experiment and Section 6 concludes this paper.

2. RELATED WORK

2.1. Convolutional Neural Networks (CNN)

CNN is comprised of convolutional layer for generating feature maps, pooling layer for reducing the dimensionality of feature maps, and fully-connected layer for understanding and classifying the extracted features (Ferreira and Giraldi 2017). By stacking these layers, various CNN architectures can be formulated, and among architectures winning ImageNet Large Scale Visual Recognition Challenge (ILSVRC) are AlexNet (Krizhevsky et al. 2012) with top-5 error rate of 16.4%, ZFNet (Zeiler and Fergus 2014) with 11.7% error rate, VGGNet (Simonyan and Zisserman 2014) with 7.3% error rate, and so on. We implement our proposed model on VGGNet, which is composed 16 or 19 convolutional and fully-connected layers. It uses combination of two 3x3 filters on convolutional layers having effects of a receptive field with larger filter such as one 5x5 sized filter while decreasing the number of used parameters. In addition, VGGNet doubles the number of filters after each pooling layer, which enables shrinking its spatial dimensions while increasing its depth.

2.2. Apparel Classification

In the earlier studies of apparel image classification, image feature is extracted by Histograms of Oriented Gradients (HOG), Local Binary Pattern (LBP), color moment, and color histogram (Liu et al. 2012). In recent studies, CNN has been used to elaborate the process of extracting feature in apparel image classification (Eshwar et al. 2016). In more recent studies, however, fine-grained classification of apparel category is implemented based on sophisticated feature extraction of CNN classifier (Iliukovich-Strakovskaia et al. 2016). The research expands the classification scope by sub-dividing shoes category into 107 subcategories, however, classifying over 100 classes with a single plain classifier needs to be further speculated by conceiving hierarchical classification where classification is made by multiple levels.

2.3. Hierarchical Classification

Hierarchical image classification using deep learning methodology is first proposed in the study of Yan et al. (2015) demonstrating the idea that some classes are more confusing than others as it is relatively easy to tell apple from bus while telling apple from orange is harder. The proposed model, Hierarchical Deep Convolutional Neural Networks (HD-CNN), will first use an initial coarse classifier CNN to separate the easily separable classes, which denoted as coarse classes, and later the fine classes. HD-CNN is trained by a multinomial logistic loss and a novel temporal sparsity penalty. However, their proposed model is composed of two different classifier training separately with multiple stages of training. In another study of hierarchical image classification model for CNN classifier, Branch Convolutional Neural Network (B-

CNN), is introduced with BT-strategy for single process of training each level outcomes and phasing multiple levels of categories as first coarse-level, second-coarse level and finally fine-level hierarchy (Zhu and Bain 2017). However, in any of above studies, the way to define the hierarchy of categories is not suggested as they arbitrarily define the hierarchy based on human perception rather than based on data-driven approach. In addition, this process of defining the hierarchy was not included in their proposing models, rather it was considered as separate prior process by the researcher. This implies that the classification results of their proposing models can be affected by the researcher's definition of the hierarchy, and this whole process cannot be automatized into a single model.

2.4. Clustering

Clustering is to identify a set of objects into groups in the purpose of grouping data to make objects in the same cluster more similar to each other than to objects from other clusters (Saxena et al. 2017). Clustering analysis is a statistical multivariate analysis technique based on unsupervised learning with extensive applications in various domains such as financial fraud, medical diagnosis, image processing, information retrieval, and bioinformatics (Bai et al. 2017). This unsupervised learning process is also used as a preliminary step for data analytics such as identification of the patterns hidden in gene expression data, data summarization for big data to address the associated storage and analytical issues, and selection of representative insurance policies from a large portfolio to build metamodels (Gan and Ng 2017).

Clustering algorithms can be divided into four types hierarchical, partitional, density-based, and grid-based clustering (Bai et al. 2017). Among these types, hierarchical clustering and partitional clustering are mostly applied for data analytic purpose. Hierarchical clustering is to represent the nested grouping of patterns and similarity levels at which groupings change using dendrogram (Jain et al. 1999). Hierarchical clustering algorithm can be achieved by several methods including single-link and complete-link. Partitional clustering is to obtain a single partition of objects rather than dividing into hierarchical structure. This type of clustering is used when the number of target objects is large or when the construction of a dendrogram is computationally prohibitive. However, this type of clustering accompanies with the problem of deciding the number of desired clustering outputs. Among partitional clustering method, k-means is the simplest and most widely used method based on a squared error criterion.

These clustering methods can be applied in the areas of image segmentation, object and character recognition, document retrieval, and data mining (Jain et al. 1999). In one of previous studies, clustering has been done using PCA and k-means clustering to propose a new method to find significant features of the urban identity from public space (Chang et al. 2017). In the study of Kamper et al. (2017), the Embedded Segmental k-means (ES-KMeans) model is introduced for speech processing. In another study of Xing et al. (2017), clustering method has been deployed in the context of privacy preservation as a mutual privacy preserving using k-means clustering scheme, which neither discloses individual's private information nor leaks the community's characteristic data.

3. PROPOSED METHOD

3.1. Single-step Model

To compare the results of our suggesting AMH-CNN model, we implement single-step model without hierarchical structure. Moreover, to verify whether our experiment results are valid and can reach the same results with other networks, we conduct our proposing AMH-CNN structure on both VGG16 and VGG19, which are the most widely used networks among VGGNets (Simonyan and Zisserman 2014, Wang et al. 2015). For the base model, both VGG16 and VGG19 are composed of five building blocks as shown in Fig. 2 and Fig. 3. These models output single multi-class label at the end of each network.

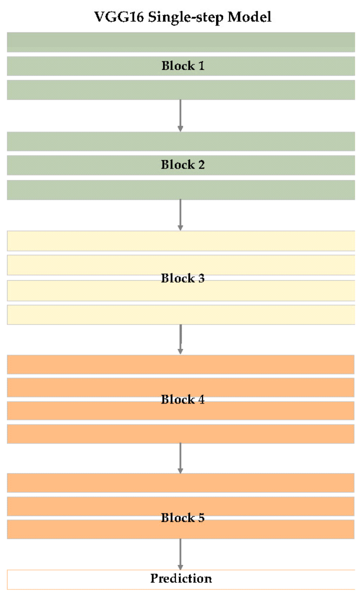


Fig. 2. CNN Architecture of VGG16 single-step model with single multi-class output.

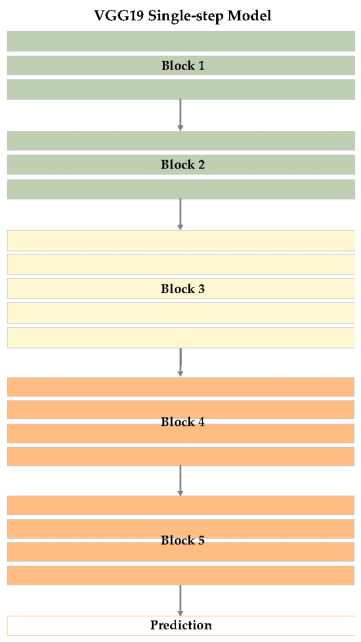


Fig. 3. CNN Architecture of VGG19 single-step model with single multi-class output.

3.2. Automatized Multi-step Hierarchical Convolutional Neural Networks (AMH-CNN) Model

AMH-CNN models follow same structural design of five building blocks as base models as shown in Fig. 4 and Fig. 5. However, these models have additional blocks below the five building blocks. We put three additional blocks followed by each prediction block outputting three levels of classified labels, which makes difference with the base models. The first underneath branch denotes the first coarse-level block, the second branch for the second coarse-level block, and the last branch for the fine-level block. All these additional blocks are composed of fully-connected neural networks. As input image goes through the AMH-CNN model, three prediction values of coarse 1 level, coarse 2 level, and fine level will be computed in the order. For example, when an input image of sweater is inserted, the first coarse level branch will indicate ‘clothes’, the second coarse level branch will indicate ‘tops’, and the final branch will indicate ‘pullover’ as output predictions.

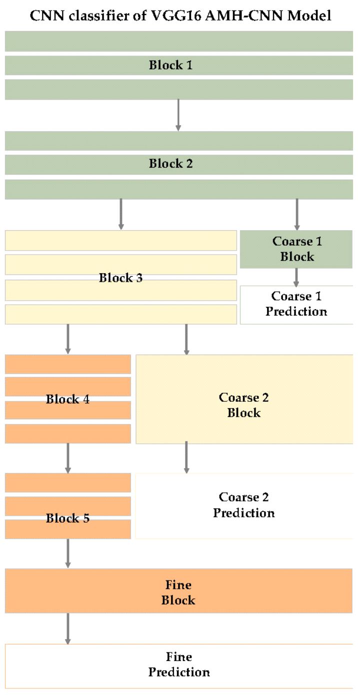


Fig. 4. CNN Architecture of VGG16 AMH-CNN model with three-levels of multi-class outputs.

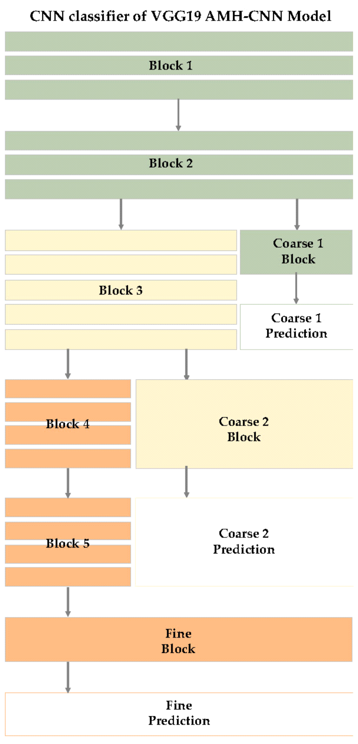


Fig. 5. CNN Architecture of VGG19 AMH-CNN model with three-levels of multi-class outputs.

4. Experiments

4.1. Dataset

For the input dataset of our proposed model, we use Fashion-MNIST dataset, which consists of 50,000 images of training set and 10,000 images of test set (Xiao et al. 2017). Each grayscale image is sized 28x28 pixels and classified into 10 classes including ‘t-shirt’, ‘trouser’, ‘pullover’, ‘dress’, ‘coat’, ‘sandals’, ‘bag’, ‘shirt’, ‘sneaker’, and ‘ankle boots’. However, to output the classification result into three-level hierarchy, we further develop the dataset into hierarchical structure, which will be discussed in Section 5.1.

4.2. Hierarchical Clustering

To define the hierarchy of the dataset, our proposed model first performs clustering with Fashion MNIST dataset by using k-Means clustering. The k-Means clustering algorithm can be implemented by following four steps (Jain et al. 1999):

1. The number of k cluster centers is chosen to either coincide with k randomly chosen patterns or k randomly defined points inside the hypervolume containing the pattern set.
2. Each pattern is assigned to the closest cluster center.
3. The cluster centers are recomputed using the current cluster memberships.
4. The process is repeated if a convergence criterion is not fulfilled.

To put these steps into equation (Xing et al. 2017), the number of participants is n with each participant A_i having sample data a_i and the participants should be clustered into k clusters of U_1, …, U_k with the j^{th} center cluster u_i. Each cluster center is randomly assigned. The sample data a_i belongs to its cluster U_i if the center u_i is the nearest according to Eq. (1) and the mean of the samples in U_i is computed following Eq. (2), where $I\{c_i = j\}$ equals 1 if $\{c_i = j\}$, else 0.

Eq. (1) $c_i := arg \min_j \|a_i - u_j\|^2$

Eq. (2) $u_j = \frac{\sum_{i=1}^{n} I\{c_i = j\} a_i}{\sum_{i=1}^{n} I\{c_i = j\}}$

Among wide varieties of criteria to measure the distance between a sample and the related cluster center, Euclidean distance is most widely used (Xing et al. 2017). For each iteration, the algorithm reassigns the data to the nearest centers according to Eq. (1) and recomputes the cluster centers following Eq. (2). This iteration is repeated until there is no or little change in the cluster centers. Note, however, that as presented in the study of Jain (2010), determining the number of clusters in the first step is one of the most difficult and important issues in clustering process.

4.3. Convolutional Neural Networks (CNN) Classifier

For single-step base models, the number of epochs is set to 60 times and the size of batch to 128. Different learning rates are applied as 0.001 in the initial stages, 0.0002 after the 42th epoch, and 0.00005 after the 52th epoch. Stochastic Gradient Descent (SGD) is used with 0.9 of momentum. For AMH-CNN models, same parameters are used as single-step models for number of epochs, batch size, learning rates, and gradient descent. However, to reflect the differences in the importance of each level of classes, different loss weights are applied on each level. In the initializing stage, we assign higher value than the later stages as the low-level feature extraction affects a lot on the result. Therefore, loss weights change as [0.98, 0.01, 0.01] in the first epoch, [0.10, 0.80, 0.10] in the 15th epoch, [0.1, 0.2, 0.7] in the 25th epoch, [0, 0, 1] epoch in the 35th epoch.

5. RESULTS

5.1. Hierarchical Clustering

Before defining the hierarchy of Fashion MNIST dataset using k-means algorithm, the model first needs to decide on how many levels and how many clusters per each level to allocate for the dataset. Elbow method is a method which finds the percentage of variance explained as a function of the number of clusters (Bholowalia and Kumar 2014). This method is used before applying k-means clustering algorithm so that to define the number of clusters by finding the minimized total intra-cluster variation or total within-cluster sum of square (WSS). In other words, the method aims to find the number of clusters where adding another cluster does not give much better modelling of the data. Elbow method first initializes the number of clusters with $k = 1$. Then, it increments the value and measures the cost of the optimal quality solution until the point when the cost of the solution drops dramatically.

Elbow method is performed on Fashion MNIST dataset to find the legitimate number of clusters and the result is plotted in Fig.6. Before the test, the model first randomly samples 100 images per each class of Fashion MNIST dataset and uses these sampled data for clustering. It then computes clustering algorithm for different values of k from 1 to 10 clusters. And for each k, it calculates the total WSS and draws the curve of WSS according to the number of clusters k. After this test, as shown in Fig.6., we can see the first bent area, which is denoted as elbow, when the number of clusters k is 2, and we can also find the second elbow between 4 to 6.

These elbow parts in the plot can be considered as indicators of the appropriate number of clusters. The rationale behind this finding is that after reaching the point where the cost drops dramatically and showing plateau, even if one increases the number of clusters, the cost will not change or improve.

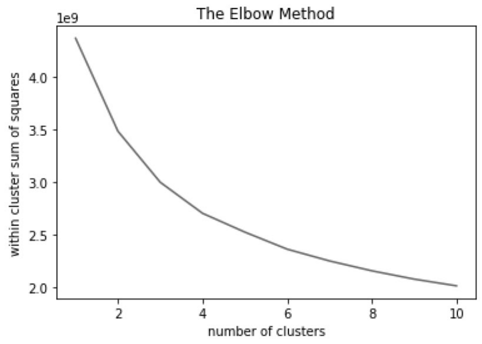


Fig. 6. Elbow method result to determine optimal number of clusters for k-means clustering.

Furthermore, the model performs additional prior test to determine right number of levels within the hierarchy of our used dataset. From the dendrogram result shown in Fig. 7, we can infer that for the Fashion MNIST dataset, the total 10 classes can be first grouped into two clusters and then further divided into the second-level of coarse clusters. The dendrogram in Fig. 7 shows the result of k-means to visualize optimal level of hierarchy. The model can first divide into two groups of leftmost and middle + rightmost. And it can further divide into two groups from leftmost group and three groups from middle + rightmost group. Here, we can infer that for the Fashion MNIST dataset, the total 10 classes can be first grouped into two clusters and then further divided into second level of coarse clusters. Therefore, we can conclude that 10 fine-levels of Fashion MNIST dataset classes can be clustered into first coarse-level groups and second coarse-level groups, which, in other words, the dataset has the hierarchical structure of two coarse-level and final fine-level clusters. Therefore, the model can conclude that 10 fine-levels of Fashion MNIST dataset classes can be clustered into first coarse-level groups and second coarse-level groups, which, in other words, the dataset has the hierarchical structure of two coarse-level and final fine-level categories.

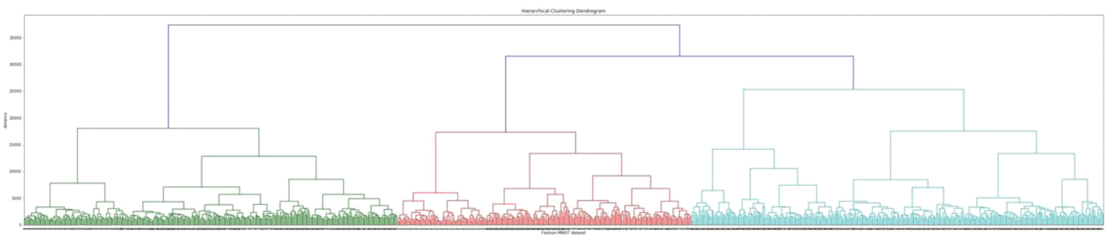


Fig. 7. Dendrogram result of k-means to visualize optimal level of hierarchies.

Based on above results from Elbow method and hierarchical clustering using dendrogram, the model finds the adequate number of clusters and levels to define the hierarchy of Fashion MNIST dataset, that is, the right value of *k* to apply k-means clustering algorithm. The model starts the clustering process setting *k* as 2, which results in t-shirt, trouser, dress, shirt, pullover, and coat into one cluster and sandal, sneaker, bag, and ankle boot into the other cluster. Then for the first cluster, it further performs clustering with $k = 3$. T-shirt, trouser, and dress belong to one group, shirt forms another group by itself, and pullover and coat are the other group. For the second cluster of the first clustering result, *k* is set to 2 and the result comes out with sandal and sneaker into one group while bag and ankle boot into the other group. Table 1 summarizes the results of k-means algorithm on Fashion MNIST dataset. The numbers indicate the number of images classified into the cluster.

TABLE 1 SUMMARY OF K-MEANS ALGORITHM RESULT ON FASHION MNIST DATASET

	k=2		**k=3**		
	cluster 0	**cluster 1**	**cluster 0**	**cluster 1**	**cluster 2**
T-Shirt	**72**	28	**57**	37	6
Trouser	**81**	19	**80**	19	1
Dress	**73**	27	**69**	30	1
Shirt	**68**	32	12	**50**	38
Pullover	**75**	25	2	35	**63**
Coat	**89**	11	15	18	**67**
			k=2		
			cluster 0	**cluster 1**	
Sandal	0	**100**	**96**	4	
Sneaker	0	**100**	**96**	4	
Bag	46	**54**	24	**76**	
Ankle Boot	22	**78**	16	**84**	

From the clustering results, the proposed model can define the hierarchy of the dataset. Fig.8. shows the hierarchical classes of dataset based on heuristic approach, which was used in the previous study of Seo and Shin (2019), while Fig.9. shows hierarchical classes of dataset based on data-driven clustering approach, which we are proposing in this paper. Hierarchy with heuristic approach has been labeled with apparel-related taxonomy by human knowledge, while hierarchy with clustering approach is labeled with number as it is resulted from data-driven approach. Slight changes have been made between these two hierarchies.

Coarse 1 level	Clothes				Goods	
Coarse 2 level	Tops	Bottoms	Dresses	Outers	Accessories	Shoes
Fine level	T-Shirt	Trouser	Dress	Coat	Bag	Sandal
	Pullover					Sneaker
	Shirt					Ankle boot

Fig. 8. Hierarchical classes of dataset based on heuristic approach.

Coarse 1 level	0			1	
Coarse 2 level	0	1	2	3	4
Fine level	T-Shirt	Shirt	Pullover	Sandal	Bag
	Trouser		Coat	Sneaker	Ankle boot
	Dress				

Fig. 9. Hierarchical classes of dataset based on data-driven clustering approach.

5.2. Convolutional Neural Networks (CNN) Classifier

We first compare the results of AMH-CNN models using clustering method for defining hierarchy with single-step base models and vanilla H-CNN models using heuristic knowledge to define the hierarchy. Table 2 shows final loss and accuracy of each of these models. The VGG16 single-step model has 0.0005 of training loss and 0.9999 of training accuracy. The loss reduces to 0.0002 and the accuracy improves to 1.0000 in H-CNN model of same architecture, while the AMH-CNN model achieves 0.0003 of training loss and 0.9999 of training accuracy. During test set, VGG16 single-step model has 0.4644 of loss and 0.9289 of accuracy, which improves to 0.3781 of loss and 0.9352 of accuracy with H-CNN model, and even more into 0.3782 of loss and 0.9366 of accuracy with AMH-CNN model.

In another comparison among different experiments with VGG19 architecture, we aim to observe whether the results are consistent with another architecture. The VGG19 single-step model has 0.0006 of loss and 0.9999 of accuracy on the training set, which get improved to 0.0002 of loss and 1.0000 of accuracy with both VGG19 H-CNN and AMH-CNN models. For the results of test set of VGG19 architecture, single-step model has 0.4356 of loss and 0.9290 of accuracy, H-CNN model shows better results of 0.4102 of loss and 0.9333 of accuracy, and finally, AMH-CNN model achieves even better results of 0.4043 of loss and 0.9349 of accuracy. We can observe that during the test set, the loss of single-step model can be decreased using H-CNN model and can be reduced even more using AMH-CNN model. As well as the loss, the accuracy of single-step model can be improved using H-CNN model and can be increased even more using AMH-CNN model.

TABLE 2 FINAL LOSS AND ACCURACY ON VGG16 SINGLE-STEP MODEL, VGG19 SINGLE-STEP MODEL, AND BOTH H-CNN AND AMH-CNN MODELS ON VGG16 AND VGG19

			Single-step Model	**H-CNN Model**	**AMH-CNN Model**
VGG16	**Train**	**Loss**	**0.0005**	**0.0002**	**0.0003**
		Accuracy	**0.9999**	**1.0000**	**0.9999**
	Test	**Loss**	**0.4644**	**0.3781**	**0.3782**
		Accuracy	**0.9289**	**0.9352**	**0.9366**
VGG19	**Train**	**Loss**	**0.0006**	**0.0002**	**0.0002**
		Accuracy	**0.9999**	**1.0000**	**1.0000**
	Test	**Loss**	**0.4356**	**0.4102**	**0.4043**
		Accuracy	**0.9290**	**0.9333**	**0.9349**

Moreover, we observe that both H-CNN and AMH-CNN models converge faster than base single-step models. Fig.10 shows plots of accuracy for each model during 60 epochs, while Fig.11 shows the plots of loss. Numbers of specific loss and accuracy value per epoch are also suggested in Table 3. We can notice that as well as the previous H-CNN model, our proposing AMH-CNN model can reach lower loss and higher accuracy faster than single-step models as the graphs go steeper in both VGG16 and VGG19 AMH-CNN models in the earlier stage. This signifies that AMH-CNN model with data-driven hierarchy can also contribute the speed of convergence.

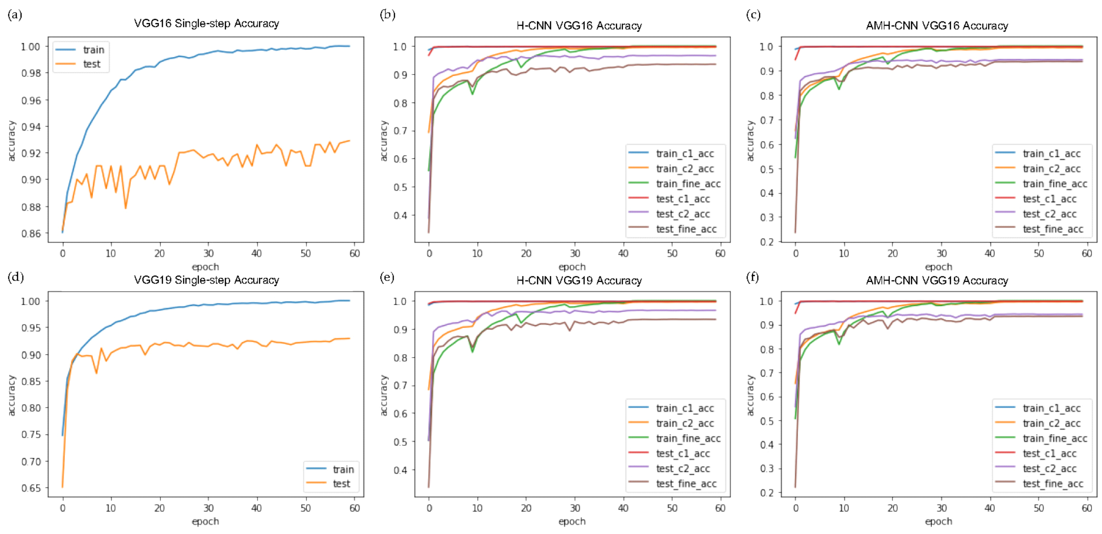


Fig. 10. Accuracy per epoch in each model.

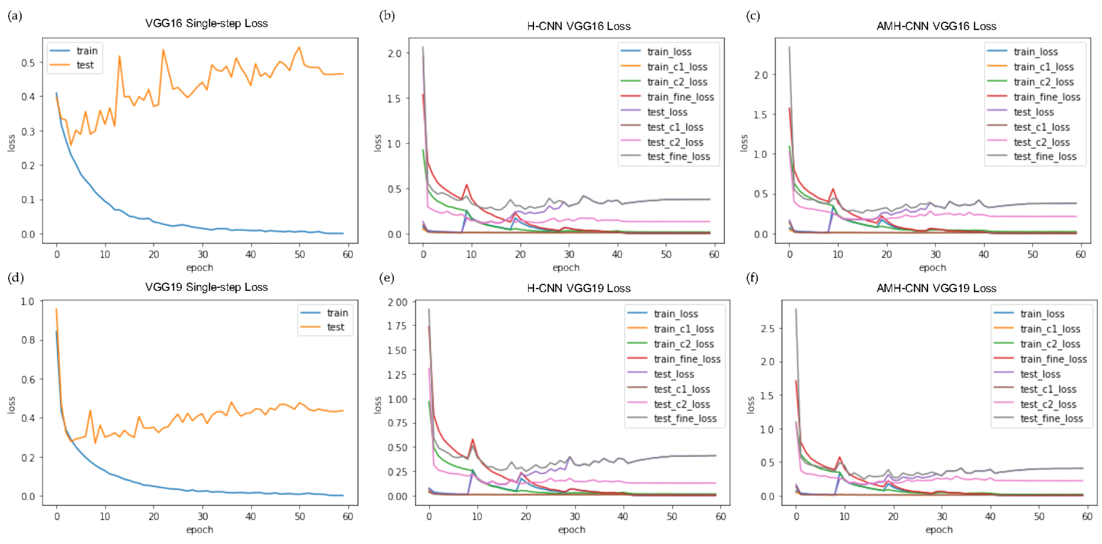


Fig. 11. Loss per epoch in each model.

TABLE 3 LOSS AND ACCURACY PER EPOCH ON VGG16 SINGLE-STEP MODEL, VGG19 SINGLE-STEP MODEL, AND BOTH H-CNN AND AMH-CNN MODELS ON VGG16 AND VGG19

	VGG16 Single-step Model				**VGG19 Single-step Model**			
	Train		**Test**		**Train**		**Test**	
Epoch	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
1	**0.4087**	**0.8600**	**0.3966**	**0.8620**	**0.8404**	**0.7473**	**0.9565**	**0.6503**
5	**0.2022**	**0.9261**	**0.3006**	**0.8960**	**0.2498**	**0.9112**	**0.2898**	**0.8954**
10	**0.1070**	**0.9606**	**0.3584**	**0.8930**	**0.1401**	**0.9501**	**0.3612**	**0.8866**
15	**0.0604**	**0.9778**	**0.3976**	**0.9000**	**0.0840**	**0.9701**	**0.3350**	**0.9152**
20	**0.0442**	**0.9842**	**0.4195**	**0.9000**	**0.0522**	**0.9810**	**0.3458**	**0.9195**
25	**0.0214**	**0.9923**	**0.4198**	**0.9200**	**0.0338**	**0.9878**	**0.3894**	**0.9163**
30	**0.0170**	**0.9939**	**0.4266**	**0.9160**	**0.0215**	**0.9927**	**0.4066**	**0.9147**
35	**0.0141**	**0.9953**	**0.4719**	**0.9100**	**0.0194**	**0.9928**	**0.4308**	**0.9138**
40	**0.0098**	**0.9965**	**0.4605**	**0.9100**	**0.0132**	**0.9957**	**0.4218**	**0.9241**
45	**0.0085**	**0.9970**	**0.4535**	**0.9260**	**0.0099**	**0.9968**	**0.4486**	**0.9218**
50	**0.0043**	**0.9985**	**0.5178**	**0.9210**	**0.0075**	**0.9974**	**0.4427**	**0.9215**
55	**0.0056**	**0.9983**	**0.4831**	**0.9200**	**0.0077**	**0.9977**	**0.4438**	**0.9237**
60	**0.0005**	**0.9999**	**0.4644**	**0.9289**	**0.0006**	**0.9999**	**0.4356**	**0.9290**

	VGG16 H-CNN Model				**VGG19 H-CNN Model**			
	Train		**Test**		**Train**		**Test**	
Epoch	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
1	**0.0683**	**0.5565**	**0.1319**	**0.3361**	**0.0753**	**0.5026**	**0.0657**	**0.3362**
5	**0.0169**	**0.8378**	**0.0197**	**0.8542**	**0.0188**	**0.8341**	**0.0164**	**0.8563**
10	**0.2537**	**0.8274**	**0.1776**	**0.8542**	**0.2646**	**0.8163**	**0.2252**	**0.8330**
15	**0.0876**	**0.9250**	**0.1346**	**0.9072**	**0.0915**	**0.9236**	**0.1224**	**0.9087**

20	0.1707	0.9233	0.2390	0.9038	0.1762	0.9203	0.2275	0.9049
25	0.0505	0.9771	0.2302	0.9199	0.0627	0.9724	0.2319	0.9175
30	0.0628	0.9783	0.3482	0.9063	0.0649	0.9770	0.3999	0.8925
35	0.0321	0.9895	0.3912	0.9149	0.0375	0.9873	0.3553	0.9154
40	0.0138	0.9953	0.3573	0.9263	0.0208	0.9929	0.3833	0.9210
45	0.0007	0.9999	0.3494	0.9328	0.0010	0.9998	0.3686	0.9333
50	0.0003	1.0000	0.3717	0.9342	0.0003	1.0000	0.3989	0.9333
55	0.0002	1.0000	0.3768	0.9348	0.0002	1.0000	0.4072	0.9334
60	0.0002	1.0000	0.3781	0.9352	0.0002	1.0000	0.4102	0.9333

	VGG16 AMH-CNN Model				VGG19 AMH-CNN Model			
	Train		Test		Train		Test	
Epoch	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
1	0.0698	0.5432	0.1699	0.2363	0.0708	0.5065	0.1658	0.2194
5	0.0195	0.8328	0.0191	0.8569	0.0188	0.8362	0.0166	0.8591
10	0.3357	0.8220	0.2507	0.8556	0.3360	0.8162	0.2510	0.8468
15	0.1266	0.9265	0.1718	0.9100	0.1255	0.9263	0.1602	0.9194
20	0.1727	0.9267	0.2563	0.9092	0.1732	0.9286	0.3119	0.8819
25	0.0510	0.9792	0.3042	0.9101	0.0524	0.9788	0.2519	0.9245
30	0.0624	0.9788	0.3861	0.9086	0.0589	0.9797	0.3220	0.9195
35	0.0288	0.9901	0.3644	0.9256	0.0350	0.9887	0.4075	0.9132
40	0.0212	0.9928	0.4188	0.9181	0.0218	0.9928	0.3688	0.9266
45	0.0007	0.9999	0.3464	0.9358	0.0007	0.9999	0.3644	0.9341
50	0.0003	1.0000	0.3723	0.9361	0.0003	1.0000	0.3951	0.9341
55	0.0003	1.0000	0.3754	0.9367	0.0002	1.0000	0.4014	0.9344
60	0.0003	0.9999	0.3782	0.9366	0.0002	1.0000	0.4043	0.9349

To compare our results with the existing classification results on Fashion MNIST dataset, we put several classification accuracy results of other data mining methods and other single-step CNN configuration in Table 4. Above four models in Table 4 are suggested value in the work of Bhatnagar et al. (2017). In their study, they propose a state-of-the-art model to classify images in the Fashion-MNIST dataset based on deep learning architectures. Their proposed model consists of two convolutional and max-pooling layers, which is denoted as CNN2, trained by batch normalization, denoted as BatchNorm, with residual skip connections, denoted as Skip. They compared their result with previously used classifier such Support Vector Classifier (SVC) and Evolutionary Deep Learning (EDEN). From the test accuracy suggested in Table 4, H-CNN models in both VGG16 and VGG19 achieve higher accuracy than previously used classifier of SVC and EDEN and even better than the state-of-the-art classification result using convolutional configuration of CNN2 + BatchNorm + Skip. Furthermore, our proposing AMH-CNN models with both VGG architectures outperform the results of H-CNN models.

TABLE 4 CLASSIFICATION RESULTS COMPARISON OF FASHION MNIST DATASET WITH LITERATURES

Model	Test Accuracy
SVC	0.8970
EDEN	0.9060
CNN2	0.9117
CNN2 + BatchNorm + Skip	0.9254
VGG16	0.9289
VGG19	0.9290
VGG16 H-CNN	0.9352
VGG19 H-CNN	0.9333
VGG16 AMH-CNN	**0.9366**
VGG19 AMH-CNN	**0.9349**

6. CONCLUSION

With current development in deep learning methodologies, image recognition using CNN is widely applied on fashion images for human detection, apparel classification, apparel retrieval, and automatic apparel tagging. However, even though those datasets have complicated hierarchical categories and their category labeling systems follow the hierarchical structure, hierarchy has not been considered in image classification process. To handle multi-class apparel classification, multi-step classification can reflect hierarchical structure of apparel classes on the classifying process of apparel image. Hierarchical Convolutional Neural Networks model trains the classifier with arbitrarily defined hierarchy of apparel category. However, as this hierarchical model requires prior process of defining the hierarchy of apparel category in heuristic approach, we therefore automatize this prior process and propose a single multi-step hierarchical classification model, called Automatized Multi-step Hierarchical Convolutional Neural Networks (AMH-CNN).

AMH-CNN clusters the apparel category groups using k-means algorithm, and following the clustered results, it defines the new hierarchy of apparel dataset to train the classifier. This classifier is based on VGGNet which consists of five building blocks of multiple convolutional, max-pooling and fully-connected layers. Proposed architecture is applied on Fashion-MNIST dataset which is 28x28 sized grayscale images of 10 classes consisting of 50,000 training images and 10,000 test images. Following the result of k-means algorithm, AMH-CNN reclassifies the dataset into three levels clusters. Whenever input image goes through AMH-CNN model, three prediction values of coarse 1 level, coarse 2 level, and fine-level will be computed in the order.

To compare the performance, we suggest experiment results of both VGG16 and VGG19 architectures in base single-step models, vanilla Hierarchical Convolutional Neural Networks (H-CNN) models, and our proposing AMH-CNN models. Results have shown that when using both multi-step H-CNN and AMH-CNN models have better classifying results than the base single-step model without hierarchical structure, and among those multi-step results, the data-driven AMH-CNN model achieves higher accuracies. We conclude that AMH-CNN brings better performance in classifying apparel images.

Our proposed classifier has several contributions first as a knowledge embedded classifier by providing additional information of hierarchical structure after multiple staged output learning strategy rather than providing single label output, and second as an automation of multi-step both reflecting data inheritance and hierarchical structure. Moreover, proposed model achieves higher classification accuracy than benchmark models. Furthermore, as with growing interest in Explainable Artificial Intelligence and its diverse approaches, our proposing model can contribute as a cue to ease and justify the inference and interpretation from the softmax classification results.

Further developments can be made for the future study. In relation to enhancing the performance of the proposed model, one can pre-trained the AMH-CNN classifier with ImageNet dataset to reduce time spent on training the classifier as transfer learning allows improvement in accuracy and enables the classifier to be trained well regardless of the size of target dataset. Or AMH-CNN can be implemented on different architectures such as ResNet, DenseNet or other CNN architectures and one can compare the time spent on training and computational cost. Meanwhile, one can also compare the classification accuracy to figure out the best suited CNN architecture for AMH-CNN model.

ACKNOWLEDGEMENTS

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

REFERENCES

Alzu'bi A, Amira A, Ramzan N (2017) Content-based image retrieval with compact deep convolutional features. *Neurocomputing* 249: 95-105.

Baesens B, Setiono R, Mues C, Vanthienen J (2003) Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science* 49(3): 312-329.

Bai L, Cheng X, Liang J, Shen H, Guo Y (2017) Fast density clustering strategies based on the k-means algorithm. *Pattern Recognition* 71: 375-386.

Bhatnagar S, Ghosal D, Kolekar MH (2017) Classification of fashion article images using convolutional neural networks. *Proc. International Conference on Image Information Processing*, 1-6.

Bholowalia P, Kumar A (2014) EBK-means: A clustering technique based on elbow method and k-means in WSN. *International Journal of Computer Applications* 105(9): 17-24.

Bossard L, Dantone M, Leistner C, Wengert C, Quack T, Van Gool L (2012) Apparel classification with style. Proc. Asian Conference on Computer Vision, 321-335.

Chang MC, Bus P, Schmitt G (2017) Feature Extraction and K-means Clustering Approach to Explore Important Features of Urban Identity. *Proc. IEEE International Conference on Machine Learning and Applications*, 1139-1144.

Eshwar SG, Rishikesh AV, Charan NA, Umadevi V (2016) Apparel classification using convolutional neural networks. *Proc. IEEE International Conference on ICT in Business Industry & Government*, 1-5.

Ferreira A, Giraldi G (2017) Convolutional Neural Network approaches to granite tiles classification. *Expert Systems with Applications* 84: 1-11.

Gan G, Ng MKP (2017) K-means clustering with outlier removal. Pattern Recognition Letters 90: 8-14.

Hara K, Jagadeesh V, Piramuthu R (2016) Fashion apparel detection: the role of deep convolutional neural network and pose-dependent priors. *Proc. IEEE Winter Conference on Applications of Computer Vision*, 1-9.

Hu Z, Yan H, Lin X (2008) Clothing Segmentation Using Foreground and Background Estimation Based on the Constrained Delaunay Triangulation. *Pattern Recognition* 41(5): 1581-1592.

Iliukovich-Strakovskaia A, Dral A, Dral E (2016) Using pre-trained models for fine-grained image classification in fashion field. *Proc. International Workshop on Fashion and KDD*, 31-40.

Jain AK (2010) Data clustering: 50 years beyond K-means. Pattern Recognition Letters 31(8): 651-666.

Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. *ACM Computing Surveys* 31(3): 264-323.

Kamper H, Livescu K, Goldwater S (2017) An embedded segmental k-means model for unsupervised segmentation and clustering of speech. *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 719-726.

Karathanasopoulos A, Osman M (20180 Forecasting the Dubai financial market with a combination of momentum effect with a deep belief network. *Journal of Forecasting*.

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* 1097-1105.

Liang X, Liu S, Shen X, Yang J, Liu L, Dong J, Yan S (2015) Deep human parsing with active template regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(12): 2402-2414.

Liu S, Song Z, Liu G, Xu C, Lu H, Yan S (2012) Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. *Proc. IEEE Computer Vision and Pattern Recognition*, 3330-3337.

Malakooti B, Zhou YQ (1994) Feedforward artificial neural networks for solving discrete multiple criteria decision making problems. *Management Science* 40(11): 1542-1561.

Muhammad K, Ahmad J, Baik SW (2018) Early fire detection using convolutional neural networks during surveillance for effective disaster management. *Neurocomputing* 288: 30-42.

Qayyum A, Anwar SM, Awais M, Majid M (2017) Medical image retrieval using deep convolutional neural network. *Neurocomputing* 266: 8-20.

Saxena A, Prasad M, Gupta A, Bharill N, Patel OP, Tiwari A, Lin CT (2017) A review of clustering techniques and developments. *Neurocomputing* 267: 664-681.

Seo Y, Shin KS (2019) Hierarchical convolutional neural networks for fashion image classification. *Expert Systems with Applications* 116: 328-339.

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv ahead of print, arXiv:1409.1556.

Taigman Y, Yang M, Ranzato MA, Wolf L (2014) Deepface: Closing the gap to human-level performance in face verification. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1701-1708.

Tsantekidis A, Passalis N, Tefas A, Kanniainen J, Gabbouj M, Iosifidis A (2017) Forecasting stock prices from the limit order book using convolutional neural networks. *Proc. Conference on Business Informatics*, 7-12.

Wang L, Xiong Y, Wang Z, Qiao Y (2015) Towards good practices for very deep two-stream convnets. arXiv ahead of print, arXiv:1507.02159.

Wang X, Zhang T (2011) Clothes search in consumer photos via color matching and attribute learning. *Proc. ACM International Conference on Multimedia*, 1353-1356.

Wehrmann J, Simões GS, Barros RC, Cavalcante VF (2018) Adult content detection in videos with convolutional and recurrent neural networks. *Neurocomputing* 272: 432-438.

Xing K, Hu C, Yu J, Cheng X, Zhang F (2017) Mutual privacy preserving k-means clustering in social participatory sensing. *IEEE Transactions on Industrial Informatics* 13(4): 2066-2076.

Yan Z, Zhang H, Piramuthu R, Jagadeesh V, DeCoste D, Di W, Yu Y (2015) HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. *Proc. IEEE International Conference on Computer Vision*, 2740-2748.

Yu S, Jia S, Xu C (2017) Convolutional neural networks for hyperspectral image classification. *Neurocomputing* 219: 88-98.

Zeiler M D, Fergus R (2014) Visualizing and understanding convolutional networks. *Proc. European Conference on Computer Vision*, 818-833.

Zhu X, Bain M (2017) B-CNN: branch convolutional neural network for hierarchical classification. arXiv a head of print, arXiv:1709.09890.

[dataset] Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv a head of print preprint. arXiv:1708.07747.

Reference-Based Sketch Image Colorization using Augmented-Self Reference and Dense Semantic Correspondence

발표자: 이준수(한국과학기술원)

Reference-Based Sketch Image Colorization using Augmented-Self Reference and Dense Semantic Correspondence

Junsoo Lee[*,1], Eungyeup Kim[*,1], Yunsung Lee[2], Dongjun Kim[1], Jaehyuk Chang[3], Jaegul Choo[1]
[1]KAIST, [2]Korea University, [3]NAVER WEBTOON Corp.
{junsoolee93,eykim94,rassilon,jchoo}@kaist.ac.kr,
swack9751@korea.ac.kr, jaehyuk.chang@webtoonscorp.com

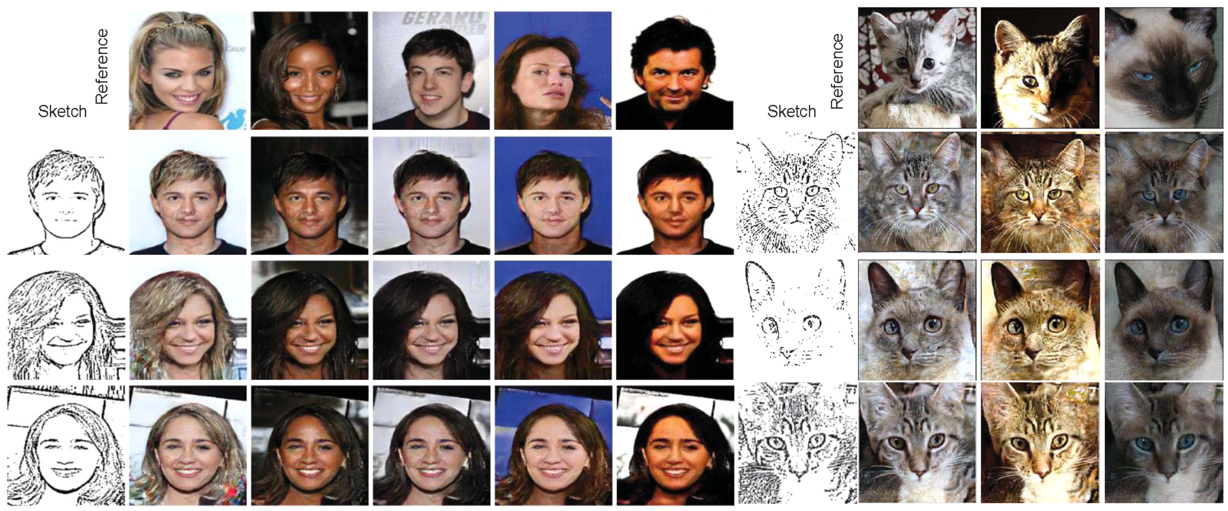


Figure 1: Qualitative results of our method on the CelebA [27] and ImageNet [36] dataset respectively. Each row has the same content while each column has the same reference.

Abstract

This paper tackles the automatic colorization task of a sketch image given an already-colored reference image. Colorizing a sketch image is in high demand in comics, animation, and other content creation applications, but it suffers from information scarcity of a sketch image. To address this, a reference image can render the colorization process in a reliable and user-driven manner. However, it is difficult to prepare for a training data set that has a sufficient amount of semantically meaningful pairs of images as well as the ground truth for a colored image reflecting a given reference (e.g., coloring a sketch of an originally blue car given a reference green car). To tackle this challenge, we propose to utilize the identical image with geometric distortion as a virtual reference, which makes it possible to secure the ground truth for a colored output image. Furthermore, it naturally provides the ground truth for dense semantic correspondence, which we utilize in our internal attention mechanism for color transfer from reference to sketch input. We demonstrate the effectiveness of our approach in various types of sketch image colorization via quantitative as well as qualitative evaluation against existing methods.

1. Introduction

Early colorization tasks [48, 21, 22] have focused on colorizing a grayscale image, which have shown great progress so far. More recently, the task of colorizing a given sketch or outline image has attracted a great deal of attention in both computer vision and graphics communities, due to its significant needs in practice. Compared to a grayscale im-

* indicates equal contribution

age, which still contains the pixel intensity, a sketch image is information-scarce, making its colorization challenging in nature. To remedy this issue, generally two types of approach of imposing additional conditions to the sketch image have been explored: user hints and reference image.

As explained in Section 2.2, there are previous works utilizing a reference or already-colored image, which shares the same semantic object of the target image. It requires an ability for the model to establish visual correspondences and inject colors through the mappings from the reference to the target. However, due to the huge information discrepancy between the sketch and reference, the sketch colorization guided by the reference is still under-explored compared to other sketch-based tasks (Section 2.1). Moreover, there are few datasets containing the labels of the correspondence between the two images, and the cost of generating a reliable matching of source and reference becomes a critical bottleneck for this task over a wide range of domains.

In this work, we utilize an augmented-self reference which is generated from the original image by both color perturbation and geometric distortion. This reference contains the most of the contents from original image itself, thereby providing a full information of correspondence for the sketch, which is also from the same original image. Afterward, our model explicitly transfers the contextual representations obtained from the reference into the spatially corresponding positions of the sketch by the attention-based pixel-wise feature transfer module, which we term the spatially corresponding feature transfer (SCFT) module. Integration of these two methods naturally reveals groundtruth spatial correspondence for directly supervising such an attention module via our similarity-based triplet loss. This direct supervision encourages the network to be fully optimized in an end-to-end manner from the scratch and does not require any manually-annotated labels of visual correspondence between source-reference pairs. Furthermore, we introduce an evaluation metric which measures how faithfully the model transfers the colors of the reference in the corresponding regions of sketch.

Both qualitative and quantitative experiments indicate that our approach exhibits the state-of-the-art performance to date in the task of information-scarce, sketch colorization based on a reference image. These promising results strongly demonstrate its significant potentials in practical applications in a wide range of domains.

2. Related work

2.1. Sketch-based Tasks

Sketch roughly visualizes the appearances of a scene or object by a series of lines. Thanks to its simple, easy-to-draw, and easy-to-edit advantages, sketch has been utilized in several tasks including image retrieval [20], sketch recognition [25], sketch generation [3, 29], and image inpainting [33]. However, due to the lack of texture and color information in sketch image, the research on sketch-based image colorization, especially reference-based colorization, is quite challenging and still under-explored.

2.2. Conditional Image Colorization

The automatic colorization has a limitation that users cannot manipulate the output with their desired color. To tackle this, recent methods come up with the idea of colorizing images with condition of the color given by users, such as scribbles [38], color palette [49, 28, 45], or text tags [18]. Even though these approaches have shown the impressive results in terms of the multi-modal colorization, they unavoidably require both precise color information and the geometric hints provided by users for every step.

To overcome the inconvenience, an alternative approach, which utilizes an already colored image as a reference, has been introduced. Due to the absence of geometric correspondence at the input level, early studies [17, 1, 26, 4, 7, 2] utilized low-level hand-crafted features to establish visual correspondence. Recent studies [10, 47, 40] compose the semantically close source-reference pairs by using features extracted from the pre-trained networks [10, 47] or color histogram [40] and exploit them in their training. These pair composition techniques however tend to be sensitive to domains, thereby limit their capability in a specific dataset.

Our work presents a novel training scheme to learn visual correspondence by generating augmented-self reference in the self-supervised manner at the training time, and then demonstrates it's scalability on various type of datasets.

3. Proposed method

In this section, we present our proposed model in detail, as illustrated in Fig. 2. We first describe overall workflow of the model and its two novel components called (1) Augmented-Self Reference Generation (Section 3.2) and (2) Spatially Corresponding Feature Transfer Module (Section 3.3). We then present our loss functions in detail.

3.1. Overall Workflow

As illustrated in Fig. 2, given a color image I in our dataset, we first convert it into its sketch image I_s using an outline extractor. Additionally, we generate an augmented-self reference image I_r by applying the thin plate splines (TPS) transformation. Taking these two images I_s and I_r as inputs, our model first encodes them into activation maps f_s and f_r using two independent encoders $E_s(I_s)$ and $E_r(I_r)$, respectively.

To transfer the information from I_r to I_s, we present a SCFT module inspired by a recently proposed self-attention mechanism [41], which computes dense correspondences

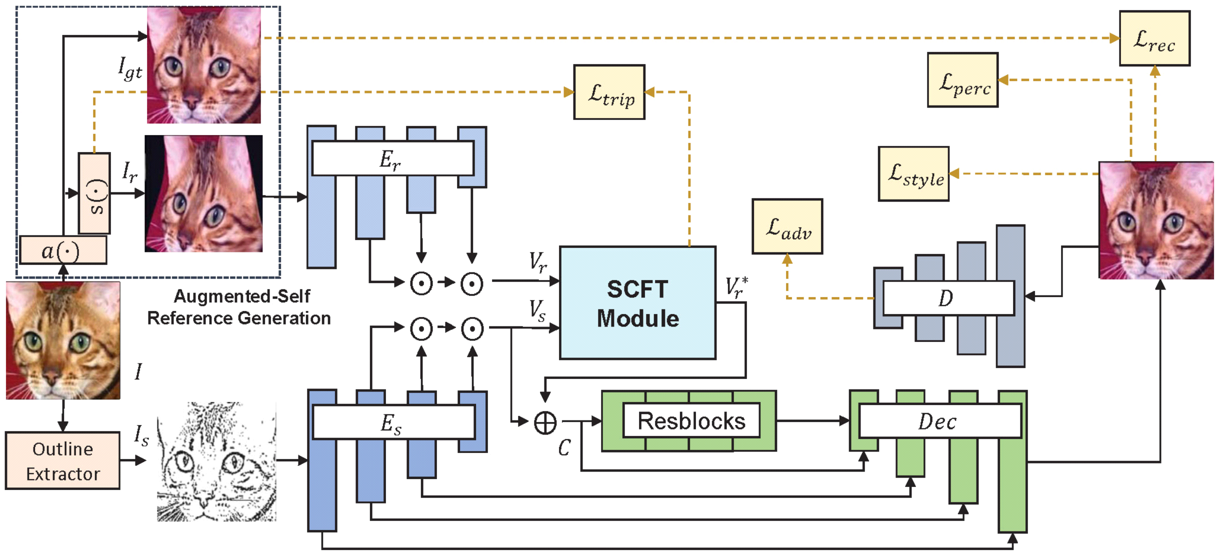


Figure 2: An overall workflow of our self-augmented learning process.

between every pixel pair of I_r to I_s. Based on the visual mappings from SCFT, context features fusing the information between I_r and I_s passes through several residual blocks and our U-net-based decoder [35] sequentially to obtain the final colored output.

3.2. Augmented-Self Reference Generation

To generate a reference color image I_r for a given sketch image I_s, we apply to original color image I two nontrivial transformations, appearance and spatial transformation. Since I_r is essentially generated from I, these processes guarantee that the useful information to colorize I_s exists in I_r, which encourages the model to reflect I_r in the colorization process. The details on how these transformations operate are described as follows. First, the appearance transformation $a(\cdot)$ adds a particular random noise per each of the RGB channel of I. The resulting output $a(I)$ is then used as the ground truth I_{gt} for the colorization output of our model. The reason why we impose color perturbation for making reference is to prevent our model from memorizing color bias, which means that a particular object is highly correlated with the single ground truth color in train data (i,e., a red color for apples). Given different reference in every iteration, our model should reconstruct different colored output for the same sketch, by leveraging I_r as the only path to restore I_{gt}. In other words, it encourages the model to actively utilize the information from E_r not just from E_s and generates reference-aware outputs at test time. Afterwards, we further apply the TPS transformation $s(\cdot)$, a non-linear spatial transformation operator to $a(I)$ (or I_{gt}), resulting in our final reference image I_r. This prevents our model from lazily bringing the color in the same pixel position from I_r, while enforcing our model to identify semantically meaningful spatial correspondences even for a reference image with a spatially different layout, e.g., different poses.

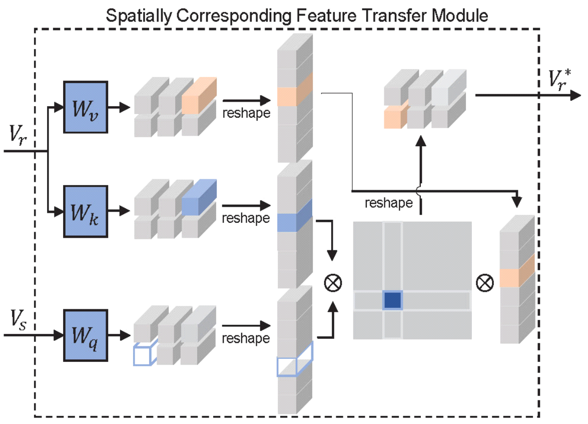


Figure 3: An illustration of spatially corresponding feature transfer (SCFT) module. SCFT establishes the dense correspondence mapping through attention mechanism.

3.3. Spatially Corresponding Feature Transfer

The goal of this module is to learn (1) which part of a reference image to bring the information from as well as (2) which part of a sketch image to transfer such information to, i.e., transferring the information from where to where. Once

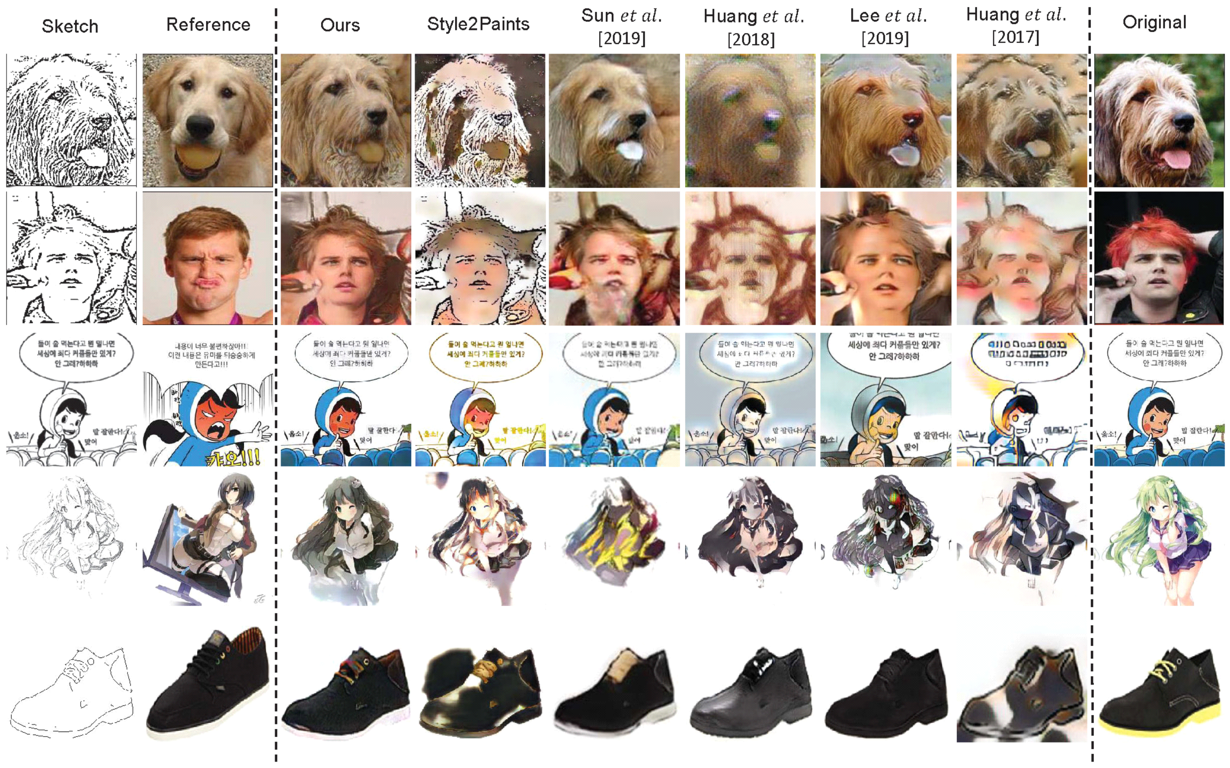


Figure 4: Qualitative comparison of colorize results with the baselines trained on the wide range of datasets. Note that the goal of our task does not reconstruct the original image. All results are generated from the unseen images. Please refer to the supplementary material for details.

obtaining this information as an attention map, our model transfers the feature information from a particular region of a reference to its semantically corresponding pixel of a sketch.

To begin with, each of the two encoders E_r and E_s consists of L convolutional layers, producing L activation maps $(f^1, f^2, \cdots, f^L)$ including intermediate outputs. Now we downsample each of them to match the spatial size of f^L and concatenate them along the channel dimensions, forming the final activation map V, i.e.,

$$V = \left[\varphi(f^1); \varphi(f^2); \cdots; f^{l_p}\right] \tag{1}$$

where φ denotes a spatially downsampling function of an input activation map $f^l \in \mathbb{R}^{h_l \times w_l \times c_l}$ to the size of $f^{l_p} \in \mathbb{R}^{h_p \times w_p \times c_p}$. ";" denotes the channel-wise concatenation operator. In this manner, we capture all the available low- to high-level features simultaneously.

Now we reshape V into $\bar{V} = [v_1, v_2, \cdots, v_{hw}] \in \mathbb{R}^{d_v \times hw}$, where $v_i \in \mathbb{R}^{d_v}$ indicates a feature representation of the i^{th} region of the given image and $d_v = \sum_{l=1}^{L} c_l$. We then obtain v_i^s of $\bar{V}_s$ and v_j^r of $\bar{V}_r$ from the outputs of the sketch encoder E_s and the reference encoder E_r, respectively. Given v_i^s and v_j^r, our model computes an attention matrix $\mathcal{A} \in \mathbb{R}^{hw \times hw}$ whose element α_{ij} is computed by the scaled dot product [41], followed by a softmax function within each row, i.e.,

$$\alpha_{ij} = \underset{j}{\text{softmax}}\left(\frac{(W_q v_i^s) \cdot (W_k v_j^r)}{\sqrt{d_v}}\right), \tag{2}$$

where $W_q, W_k \in \mathbb{R}^{d_v \times d_v}$ represent the linear transformation matrix into a query and a key vector, respectively, in the context of a self-attention module, and $\sqrt{d_v}$ represents a scaling factor. α_{ij} is a coefficient representing how much information v_i^s should bring from v_j^r. Now we can obtain the context vector v_i^* of the position i as

$$v_i^* = \sum_j \alpha_{ij} W_v v^{r_j}, \tag{3}$$

where $W_v \in \mathbb{R}^{d_v \times d_v}$ is the linear transformation matrix into a value vector containing the color feature in a semantically related region of a reference image.

Finally, v_i^* is added to the original feature v_i^s of a sketch image to form the feature vector enriched by the information of the corresponding region in the reference image, i.e.,

$$c_i = v_i^s + v_i^* \tag{4}$$

c_i is then fed into the decoder to synthesize a colored image.

3.4. Objective Functions

Similarity-Based Triplet Loss. When applying the spatial transformation $s(\cdot)$, each pixel value in the output image is represented as a weighted average of pixels in the input image, revealing the spatial correspondences of pixel pairs between I_s and I_r. In other words, we can obtain the full information of the weight w_{ij}, which represents how much the i^{th} pixel position of the input image, or a query, is related to the j^{th} pixel position of the output, or a key. Then, the value of w_{ij} can be considered as the pixel-to-pixel correspondence, which can work as the groundtruth for supervising how semantically related the pixel of the reference to a particular pixel of sketch image.

Utilizing this pixel-level correspondence information, we propose a similarity-based triplet loss, which is a variant of triplet loss [39], to directly supervise the affinity between the pixel-wise query and key vectors used to compute the attention map $\mathcal{A}$ in Eq. (2). The proposed loss term is computed as

$$\mathcal{L}_{tr} = \max(0, [-S(v_q, v_k^p) + S(v_q, v_k^n) + \gamma]), \tag{5}$$

where $S(\cdot,\cdot)$ computes the scaled dot product. Given a query vector v_q as an anchor, v_k^p indicates a feature vector sampled from the positive region, and v_k^n is a negative sample. γ denotes a margin, which is the minimum distance $S(v_q, v_k^p)$ and $S(v_q, v_k^n)$ should maintain. $\mathcal{L}_{tr}$ encourages the query representation to be close to the correct (positive) key representation, while penalizing to be far from the wrong (negatively sampled) one. This loss plays a crucial role in directly enforcing our model to find the semantically matching pairs and reflect the reference color into the corresponding position.

The reason we adopt triplet loss instead of commonly used losses such as L_1-loss is that the latter can overly penalize the affinities between semantically close but spatially distant query and key pixel pairs. This misleading result can be mitigated by only penalizing two cases: the semantically closest pair (positive sample) and randomly-sampled except it (negative sample), which is basically a triplet loss.

We further conduct a user study to compare the effects of our triplet loss to another possible loss, i.e., L_1-loss and no supervision. Details about the experimental settings and results are explained in Section 6.2 in the supplementary material.

L1 Loss. Since the groundtruth image I_{gt} is generated as Section 3.2, we can directly impose a reconstruction loss to penalize the network for the color difference between the output and the ground truth image as below:

$$\mathcal{L}_{rec} = \mathbb{E}\left[\| G(I_s, I_r) - I_{gt} \|_1\right]. \tag{6}$$

Adversarial Loss. The discriminator D, as an adversary of the generator, has an objective to distinguish the generated images from the real ones. The output of real/fake classifier $D(X)$ denotes the probability of an arbitrary image X to be a real one. We adopt *conditional GANs* which use both a generated sample and additional conditions [34, 44, 15]. In this work, we leverage the input image I_s as a condition for the adversarial loss since it is important to preserve the content of I_s as well as to generate a realistic fake image. The loss for optimizing D is formulated as a standard cross-entropy loss as

$$\begin{aligned} \mathcal{L}_{adv} = \; & \mathbb{E}_{I_{gt}, I_s}\left[\log D(I_{gt}, I_s)\right] \\ & + \mathbb{E}_{I_s, I_r}\left[\log(1 - D(G(I_s, I_r), I_s))\right]. \end{aligned} \tag{7}$$

Perceptual Loss. As shown in previous work [33], perceptual loss [16] encourages a network to produce an output that is perceptually plausible. This loss penalizes the model to decrease the semantic gap, which means the difference of intermediate activation maps between the generated output $\hat{I}$ and the ground truth I_{gt} from the ImageNet [36] pre-trained network. We employ a perceptual loss using multi-layer activation maps to reflect not only high-level semantics but also low-level styles as

$$\mathcal{L}_{perc} = \mathbb{E}\left[\sum_l \| \phi_l(\hat{I}) - \phi_l(I_{gt}) \|_{1,1}\right], \tag{8}$$

where ϕ_l represents the activation map of the l'th layer extracted at the *relul*_1 from the VGG19 network.

Style Loss. Sajjadi *et al.* [37] has shown that the style loss which narrow the difference between the covariances of activation maps is helpful for addressing checkerboard artifacts. Given $\phi_l \in \mathbb{R}^{C_l \times H_l \times W_l}$, the style loss is computed as

$$\mathcal{L}_{style} = \mathbb{E}\left[\| \mathcal{G}(\phi_l(\hat{I})) - \mathcal{G}(\phi_l(I_{gt})) \|_{1,1}\right], \tag{9}$$

where $\mathcal{G}$ is a gram matrix.

In summary, the overall loss function for the generator G and discriminator D is defined as

$$\begin{aligned} \min_G \max_D \mathcal{L}_{total} = \; & \lambda_{tr}\mathcal{L}_{tr} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv} \\ & + \lambda_{perc}\mathcal{L}_{perc} + \lambda_{style}\mathcal{L}_{style}. \end{aligned} \tag{10}$$

3.5. Implementation Details

We implement our model with the size of input image fixed in 256×256 on every datasets. For training, we set the coefficients for each loss functions as follows: $\lambda_{adv} = 1$, $\lambda_{rec} = 30$, $\lambda_{tr} = 1$, $\lambda_{perc} = 0.01$, and $\lambda_{style} = 50$. We

	ImageNet			Human Face	Comics		Hand-drawn
Methods	Cat	Dog	Car	CelebA	Tag2pix	Yumi's Cells	Edges→Shoes
Sun *et al.* [40]	160.65	168	192.00	75.66	122.14	72.45	124.98
Huang *et al.* [13]	281.44	271.47	258.36	173.12	76.00	132.90	86.43
Lee *et al.* [24]	151.52	172.22	70.07	68.43	91.65	63.34	109.29
Huang *et al.* [12]	257.39	268.69	165.84	160.22	97.40	148.52	190.16
(a) Ours w/o $\mathcal{L}_{tr}$	77.39	109.49	54.07	53.58	47.68	51.34	79.85
(b) Ours full	**74.12**	**102.83**	**52.23**	**47.15**	**45.34**	**49.29**	**78.32**

Table 1: Quantitative comparisons over the datasets with existing baselines by measuring FID [11] score: a lower score is better.

set the margin of the triplet loss $\gamma = 12$ for overall data. We use Adam solver [19] for optimization with $\beta_1 = 0.5$, $\beta_2 = 0.999$. The learning rate of generator and discriminator are initially set to 0.0001 and 0.0002 for each. The detailed network architectures are described in Section 6.5 of supplementary material.

4. Experiments

This section demonstrates the superiority of our approach on wide range of domain datasets (Section 4.1) including real photos, human face and anime (comics). We newly present an evaluation metric, named SC-PSNR described in Section 4.2, to measure the faithfulness of reflecting the style of the reference. Afterwards, we compare our method against the several baselines of related tasks quantitatively as well as qualitatively (Section 4.3). An in-depth analysis of our approach is described across Section 4.4-4.5.

4.1. Datasets

Tag2pix Dataset. We use Tag2pix dataset [18], which contains large-scale anime illustrations filtered from Danbooru2017 [8], to train our model for comic domain. Although there are various tag labels on this dataset, we only utilize images to train the model owing to our self-supervised training scheme. It consists of one character object with white background images. We partition into 54,317 images for train, 6036 images for test and then combine source-reference pairs by randomly sampled from the test set for evaluation.

Yumi Dataset. Like Yoo *et al.* [46], we collect images from the online cartoon named *Yumi's Cells* for the outline colorization of the anime domain. The dataset contains repeatedly emerging characters across 329 episodes. With this limited variety of characters, the network is required to find the correct character matching even if there is no explicit character supervisions. We randomly split into a train set of 7,014 images and test set of 380 images, and then manually construct source-reference pairs from the testset to evaluate the performance of the models.

	SC-PSNR (dB)		
Methods	Cat	Dog	Car
Sun *et al.* [40]	9.65	11.19	9.42
Huang *et al.* [13]	10.33	12.67	8.45
Lee *et al.* [24]	11.54	12.08	9.94
Huang *et al.* [12]	9.25	9.49	7.77
(a) Ours w/o $\mathcal{L}_{tr}$	12.76	13.73	10.56
(b) Ours full	**13.23**	**14.37**	**11.34**

Table 2: Quantitative comparisons over the SPair-71k with existing baselines by measuring SC-PSNR (dB) score: a higher score is better.

SPair-71k Dataset. SPair-71k dataset [31], which is manually annotated for a semantic correspondence task, consists of total 70,958 pairs of images from PASCAL 3D+ [43] and PASCAL VOC 2012 [6]. We select two non-rigid categories (cat, dog) and one rigid category (car), of which we can gather sufficient data points from ImageNet [36]. Note that this dataset is used to measure SC-PSNR (Section. 4.2) score only for the evaluation purpose.

ImageNet Dataset. As above-mentioned, we collect subclasses that correspond to three categories (i.e., cat, dog, car) from ImageNet [36] dataset and use them for training data. Images in each class are randomly divided into two splits with an approximate ratio of 9:1 for training and validation.

Human Face Dataset. Our method can be applied to colorize a sketch image of human face domain as well. To support this claim, we leverage CelebA [27] dataset, which have commonly been used for image-to-image translation or style transfer tasks. Training and validation sets are composed as the ImageNet dataset are.

Edges→Shoes Dataset. We use Edges→Shoes dataset, which contains pairs of sketch-color shoes images that have been widely used in image-to-image translation tasks [23, 13] as well. This enables a valid evaluation between our method and existing unpaired image-to-image translation

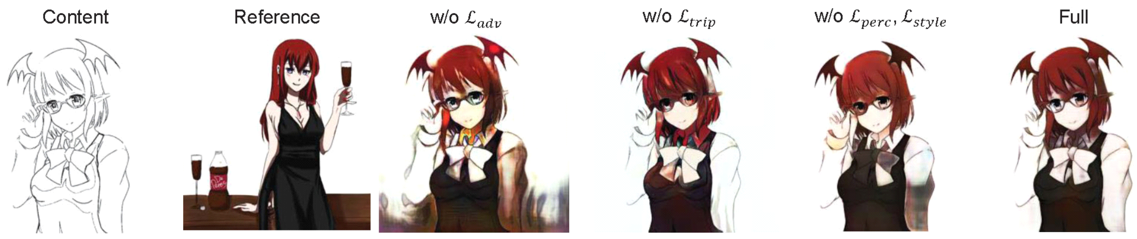


Figure 5: A qualitative example presenting the effectiveness of different loss functions.

	ImageNet			Human Face	Comics		Hand-drawn
Loss Functions	Cat	Dog	Car	CelebA	Tag2pix	Yumi's Cells	Edges→Shoes
$\mathcal{L}_{rec}$	82.10	143.76	68.45	77.70	58.00	52.86	91.10
$\mathcal{L}_{rec} + \mathcal{L}_{adv}$	78.56	110.86	56.54	54.75	48.71	51.96	82.55
$\mathcal{L}_{rec} + \mathcal{L}_{adv} + \mathcal{L}_{perc} + \mathcal{L}_{style}$	77.39	109.49	54.07	53.58	47.68	51.34	79.85
$\mathcal{L}_{rec} + \mathcal{L}_{adv} + \mathcal{L}_{perc} + \mathcal{L}_{style} + \mathcal{L}_{tr}$	**74.12**	**102.83**	**52.23**	**47.15**	**45.34**	**49.29**	**78.32**

Table 3: FID scores [11] according to the ablation of loss function terms described in Section 4.4. A lower score is better.

approaches.

4.2. Evaluation Metrics

Semantically Corresponding PSNR. This work proposes a novel evaluation metric to measure how faithfully the model transfers the style of reference in the corresponding regions. In the traditional automatic colorization setting where a groundtruth image is available, pixel-level evaluation metric, such as peak signal-to-noise ratio (PSNR), has been widely used. In reference-based colorization setting, however, there is no ground truth that have both the shape of the content and the style of the reference.

The key idea behind the semantically corresponding PSNR (SC-PSNR) is leveraging the datasets created for keypoint alignment tasks [6, 43, 31], thereby providing patch-level groundtruth. We use SPair-71k dataset [31] which contains semantically corresponding annotation pairs between two different images. Only the pixel values in a certain size of patch surrounding the corresponding keypoints of two images are used instead of the whole pixels for computing mean square error (MSE), and then PSNR is computed with the MSE. We refer to this measurement as the SC-PSNR.

Fig. 6 shows first and last two examples of images queried by the leftmost image. The list of images are retrieved in a decreasing order of the value of SC-PSNR being computed with query. This figures demonstrates that this metric captures perceptually plausible distance of the pixel values between the keypoint regions of two images.

***Fréchet Inception Distance* (FID) [11].** FID is a well-known metric for evaluating the performance of a generative model by measuring the Wasserstein-2 distance between the feature space representations of the real images and its generated outputs. A low score of FID indicates that the model generates the images with quality and diversity close to real data distribution.

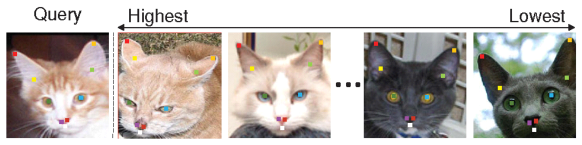


Figure 6: Different colors of points denote different keypoint annotations on cat face, e.g., eyes and noses.

4.3. Comparisons to Baselines

We compare our method against recent deep learning-based approaches on the various types of datasets both qualitatively and quantitatively. The baselines are selected from not only the colorization task [28, 40] but also the related problems tackling multi-modal image generation, such as exemplar-guided image translation [13, 24] and style transfer [12].

Fig. 4 shows the overall qualitative results of our model and other baselines on 5 different datasets. Datasets vary from real image domain like ImageNet or Human face dataset to sketch image domain like Edges→Shoes, Yumi's Cells, and Tag2pix. The leftmost and second column are sketch and reference, respectively. On every dataset our model brings the exact colors from the reference image and injects them into the corresponding position in the sketch.

For example, our model colorizes the character's face in third row with red color from the reference, while baselines tend not to fully transfer it. Likewise, in fifth row, inner side of the shoes and shoe sole are elaborately filled with the color exactly referencing the exemplar image.

We report on Table 1 the FID score calculated over the 7 different datasets. Our method outperforms the existing baselines by a large margin, demonstrating that our method has the robust capability of generating realistic and diverse images. Improved scores of our model with triplet loss indicates that $\mathcal{L}_{tr}$ plays a beneficial role in generating realistic images by directly supervising semantic correspondence.

Table 2 presents the other quantitative comparisons in regard to the SC-PSNR scores as described in Section 4.2. We measure SC-PSNR only over cat, dog and car dataset which are subclasses belonging to both ImageNet and SPair-71k [31]. Our method outperforms all the baseline models, demonstrating that our model is superior at establishing visual correspondences, and then generating suitable colors.

We conduct a user study for human evaluation on our model and other existing baselines, as shown in Fig. 8. The detailed experimental setting is described in Section 6.2 in the supplementary material. Our model occupies a large percentage of Top1 and Top2 votes, indicating that our method better reflects the color from the reference and generates more realistic outputs than other baselines.

4.4. Analysis of Loss Functions

We ablate the loss functions individually to analyze the effects of the functions qualitatively, as shown in Fig. 5 and quantitatively, as shown in Table 3. When we remove $\mathcal{L}_{adv}$, output image contains inaccurate colors emerging in the background and dramatically appears unrealistic. Without $\mathcal{L}_{tr}$, character's back hair, forehead and ribbon tail are colorized with wrong color or even not colorized. The FID score in Table 3 third row also represents that model generates unrealistic output. This degraded performance is due to the absence of supervision which encourages to match the semantically close regions between content and reference. When we remove $\mathcal{L}_{perc}$ and $\mathcal{L}_{style}$, the colorization tends to produce color bleeding or visual artifacts since there is no constraint to penalize the model for the semantic difference between the model output and the ground truth. Image generated with full losses have exact colors in its corresponding regions with fewer artifacts.

4.5. Visualization of Attention Maps

Fig. 7 shows an example of an attention map $\mathcal{A}$ learned by our SCFT module. In this module, each pixel from the sketch is used as a query to retrieve the relevant local information from the reference. In the case of left-eye region as a query (red square in (a)), we visualize the top three, highly-attentive regions in the reference image (a highlighted region in (b)). Based on this attention pattern, our model properly colorizes the left eye of a person in a sketch image (c) with blue color. For additional examples of visualizing attention maps for different sketch and reference images, we strongly encourage the readers to check out the Fig. 14 in the supplementary material for details.

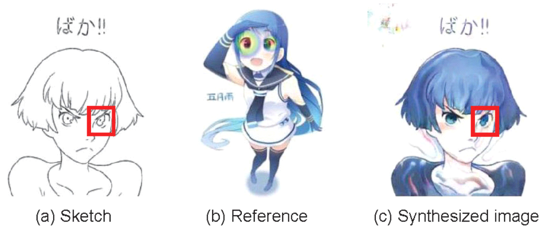


Figure 7: Visualization of our attention mechanism.

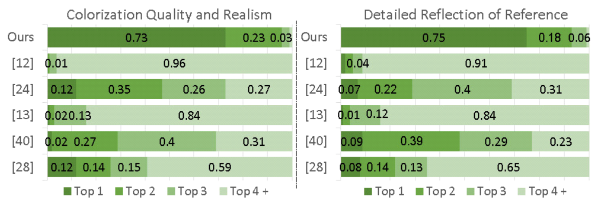


Figure 8: User study results. Percentage values are averaged over every datasets we experimented. Individual results are presented in Section 6.2 in supplementary material.

5. Conclusions

This paper presents a novel training scheme, integrating the augmented-self reference and the attention-based feature transfer module to directly learn the semantic correspondence for the reference-based sketch colorization task. Evaluation results demonstrate that our SCFT module exhibits the state-of-the-art performance over the diverse datasets, which demonstrates the significant potentials in practice. Finally, SC-PSNR, a proposed evaluation metric, effectively measures how the model faithfully reflects the style of the exemplar.

Acknowledgements. This work was partially supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)), the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (No. NRF-2019R1A2C4070420), and the National Supercomputing Center with supercomputing resources including technical support (KSC-2019-CRE-0133). Finally, we thank all researchers at NAVER WEBTOON Corp.

References

[1] Aurélie Bugeau, Vinh-Thong Ta, and Nicolas Papadakis. Variational exemplar-based image colorization. *IEEE Transactions on Image Processing*, 23(1):298–307, 2013. 2

[2] Guillaume Charpiat, Matthias Hofmann, and Bernhard Schölkopf. Automatic image colorization via multimodal predictions. In *ECCV*, pages 126–139, 2008. 2

[3] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *CVPR*, pages 9416–9425, 2018. 2

[4] Alex Yong-Sang Chia, Shaojie Zhuo, Raj Kumar Gupta, Yu-Wing Tai, Siu-Yeung Cho, Ping Tan, and Stephen Lin. Semantic colorization with internet images. *TOG*, 30(6):156:1–156:8, 2011. 2

[5] Geirhos et al. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *ICLR*, 2019. 13

[6] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. 6, 7

[7] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong. Image colorization using similar images. In *MM*, pages 369–378, 2012. 2

[8] Aaron Gokaslan Gwern Branwen. Danbooru2017: A large-scale crowdsourced and tagged anime illustration dataset. `https://www.gwern.net/Danbooru2017`, 2018. [Online; accessed 22-03-2018]. 6

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 12

[10] Mingming He, Dongdong Chen, Jing Liao, Pedro V Sander, and Lu Yuan. Deep exemplar-based colorization. *TOG*, 37(4):47, 2018. 2

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, pages 6626–6637, 2017. 6, 7, 12

[12] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1501–1510, 2017. 6, 7, 11, 12

[13] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, pages 172–189, 2018. 6, 7, 12

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, page 448–456, 2015. 12

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017. 5, 12, 15

[16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016. 5

[17] Hemant B Kekre and Sudeep D Thepade. Color traits transfer to grayscale images. In *2008 First International Conference on Emerging Trends in Engineering and Technology*, pages 82–85, 2008. 2

[18] Hyunsu Kim, Ho Young Jhoo, Eunhyeok Park, and Sungjoo Yoo. Tag2pix: Line art colorization using text tag with secat and changing loss. In *ICCV*, pages 9056–9065, 2019. 2, 6, 11, 12, 13, 14

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6, 12

[20] Sasi Kiran Yelamarthi, Shiva Krishna Reddy, Ashish Mishra, and Anurag Mittal. A zero-shot framework for sketch based image retrieval. In *ECCV*, 2018. 2

[21] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, pages 577–593, 2016. 1

[22] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, pages 6874–6883, 2017. 1

[23] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, pages 35–51, 2018. 6

[24] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations. *International Journal of Computer Vision*, 2020. 6, 7, 12

[25] Fang Liu, Xiaoming Deng, Yu-Kun Lai, Yong-Jin Liu, Cuixia Ma, and Hongan Wang. Sketchgan: Joint sketch completion and recognition with generative adversarial network. In *CVPR*, pages 5830–5839, 2019. 2

[26] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. Intrinsic colorization. *TOG*, 27(5):152:1–152:9, 2008. 2

[27] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, 2015. 1, 6, 13, 14

[28] Illyasviel. style2paints. `https://github.com/lllyasviel/style2paints`, 2018. [Online; accessed 22-03-2018]. 2, 7, 11, 12

[29] Yongyi Lu, Shangzhe Wu, Yu-Wing Tai, and Chi-Keung Tang. Image generation from sketch constraint using contextual gan. In *ECCV*, pages 205–220, 2018. 2

[30] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, pages 2794–2802, 2017. 12

[31] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019. 6, 7, 8

[32] NaverWebtoon. Yumi's cells. `https://comic.naver.com/webtoon/list.nhn?titleId=651673`, 2019. [Online; accessed 22-11-2019]. 11, 13, 19

[33] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. Edgeconnect: Structure guided image inpainting using edge prediction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2, 5

[34] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, pages 2642–2651. JMLR. org, 2017. 5

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 1, 5, 6, 15, 18

[37] Mehdi SM Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *ICCV*, pages 4491–4500, 2017. 5

[38] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, pages 5400–5409, 2017. 2

[39] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 5

[40] Tsai-Ho Sun, Chien-Hsun Lai, Sai-Keung Wong, and Yu-Shuen Wang. Adversarial colorization of icons based on contour and color conditions. In *MM*, pages 683–691, 2019. 2, 6, 7, 12

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 2, 4

[42] Holger WinnemöLler, Jan Eric Kyprianidis, and Sven C Olsen. Xdog: an extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*, 36(6):740–753, 2012. 12

[43] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *2014 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 75–82, 2014. 6, 7

[44] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, pages 1316–1324, 2018. 5

[45] Taizan Yonetsuji. Paintschainer. `https://paintschainer.preferred.tech/index_en.html`, 2017. [Online; Accessed 22-03-2018]. 2

[46] Seungjoo Yoo, Hyojin Bahng, Sunghyo Chung, Junsoo Lee, Jaehyuk Chang, and Jaegul Choo. Coloring with limited data: Few-shot colorization via memory augmented networks. In *CVPR*, pages 11283–11292, 2019. 6

[47] Bo Zhang, Mingming He, Jing Liao, Pedro V Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *CVPR*, pages 8052–8061, 2019. 2, 12, 13

[48] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, pages 649–666, 2016. 1

[49] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. Real-time user-guided image colorization with learned deep priors. *TOG*, 36(4), 2017. 2

A. Supplementary Material

This supplementary document presents additional details of the paper. Section A.1 discusses the effects of our spatially corresponding feature transfer mechanism with quantitative results. Section A.2 demonstrates the human evaluation results that compare ours against baseline methods. Afterwards, Section A.3 reports implementation details including network architectures, the processes of generating augmented-self reference images, and other training details. Comparisons to an existing study which shares similar network architectures are described in Section A.4. Lastly, Section A.5 addresses the case where a reference image does not exist. Qualitative results generated by our method are also shown throughout the document.

A.1. Effects of Aggregation Methods

The key assumption behind SCFT is that integrating spatially aligned reference features with content features would help reflect the exact color from the reference into corresponding positions. To prove this assumption, we compare our SCFT with two simple types of aggregation methods as shown in Fig. 9. Methods are as follows: (a) representations of the reference are simply added to the features of the content. (b) AdaIN [12] is utilized to transfer the style of reference by aligning the channel-wise mean and variance of content to match those of reference. (c) our SCFT module.

Qualitative comparison over three methods is shown in Fig. 10. The leftmost column contains sketch and reference, while next three columns contain colorized images from (a), (b) and (c), respectively. Method (a) tends not to perfectly locate the corresponding regions and results in colorizing car with overly yellowish color, which is mainly background color in the exemplar. Method (b) totally ignores the spatially varying color information, thus colorizing with dominant color from the reference. (c) is superior to other methods in terms of color transferability to the corresponding position.

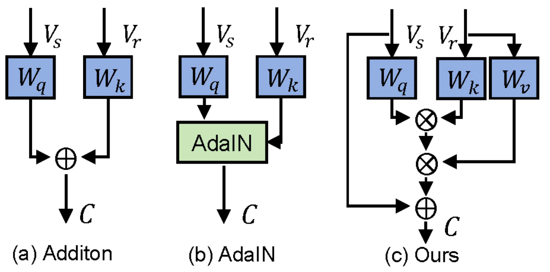


Figure 9: Diagram of three types of aggregation methods. (a) Addition block, (b) AdaIN [12] block, (c) Ours (SCFT)

Quantitative results comparing these methods are represented in Table 4. The network with SCFT module produces the most realistic results over most of the datasets. This is because the SCFT module properly aligns the corresponding local regions between the sketch and the reference image by using the attention matrix $\mathcal{A}$. On the other hand, the method (a) and (b) are not capable of aligning the local features of the reference with those of the sketch, resulting in low FID scores.

In Yumi's Cells [32] dataset, however, the SCFT module produced worse FID score than the others. The potential reason we assume is that the sketch and the reference we randomly pair for the inference time often contain different types of objects, e.g., Yumi (a human) and cells (non-human), which may have negatively impacted the colorization output.

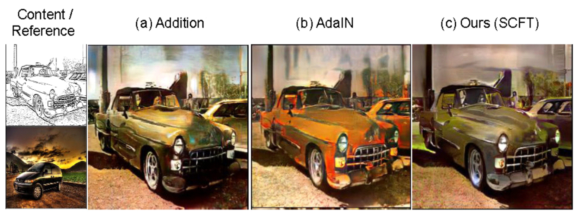


Figure 10: A qualitative example obtained from three different aggregation methods as shown in Figure 9.

A.2. User Study

We conduct two different human evaluation on the colorization outputs over various datasets. First, we randomly select ten sets of images per dataset, which contain the generated images from our method and other baselines. Second, we also randomly select ten sets of images for every dataset, and those contain the images obtained from the model trained with triplet loss, L_1-loss and no supervision for correspondence, respectively. For both cases, participants with no prior knowledge in this work are asked to rank them in terms of two types of questions sequentially as follows:

- **Overall Colorization Quality and Realism**
  How natural does the colorized image look? This question requires users to evaluate the overall quality of the generated colorization given an input sketch. The generated image should be perceptually realistic without any artifacts or color bleeding across sketches.
- **Detailed Reflection of Reference**
  How well is the colors of the reference image is reflected to a given sketch part by part? This question asks users to determine whether the particular color from a reference is injected into the corresponding regions in the sketch. For example, given an comics character image with green hair wearing a blue shirt as a reference, the generated output is expected to contain these colors at its corresponding hair and clothing part, respectively.

As seen in Fig. 23 and 24, superior measures indicate that our approach generates both more realistic and more faithfully colorized image than other methods. For both question type 1 and 2, it can be observed that our approach achieves the rank 1 votings more than 50% over all the dataset we adopt for user study. When asked the first question on Comics domain dataset including Tag2pix [18] and Yumi's Cell [32], Style2Paints [28] perform realistic generation quality comparable to our method with a small gap in top 1 rate. This notable measure is obtained as Style2Paints [28] is a adept baseline especially on comic domain. However, the difference in top 1 rate increases as the users are asked to choose based on faithful colorization performance. The results demon-

	ImageNet			Human Face	Comics		Hand-drawn
Aggregation Method	Cat	Dog	Car	CelebA	Tag2pix	Yumi's Cells	Edges2Shoes
(a) Addition	78.47	103.73	55.80	51.94	47.72	47.67	117.15
(b) AdaIN	75.17	105.72	52.85	50.61	52.81	**45.36**	88.46
(c) SCFT (ours)	**74.12**	**102.83**	**52.23**	**47.15**	**45.34**	49.29	**78.32**

Table 4: FID scores [11] according to different aggregation methods.

strate that our model utilizes the right color from the reference, which results in both realistic and exquisitely colorized output.

The results in Fig. 25 demonstrates that the model trained with triplet loss obtains more realistic and faithfully colorized outputs than with L_1-loss or no loss. Furthermore, along with the explanation of similarity-based triplet loss in Section 3.4 of the paper, these results support that the supervision for semantic correspondence with the L_1-loss leads to the inferior colorization performance even compared to the model without any supervision.

A.3. Implementation Details

This section provides the implementation details of our model, complementary to Section 3.5 of the paper.

Augmented-Self Reference Generation To automatically generate a sketch image from an original color image, We utilize a widely-used algorithm called XDoG [42]. The outputs, however, often involves superfluous edges, so in order to suppress them, we apply Gaussian blurring ($\sigma = 0.7$) to the original images before extracting sketches. The appearance transformation $a(\cdot)$ adds randomly sampled value from a uniform distribution on [-50, 50] to each of the RGB channels of the original image.

Encoder Our generator G contains two types of encoder, E_s and E_r. Both of them share the same architecture shown in Table 5, except for the number of input channels of the first layer, where E_s takes a single-channel, binarized sketch input while E_r takes a three-channel, RGB reference image. We utilize the an average pooling function for downsampling φ in Section 3.3 of the paper.

Layer	Encoder
L1	Conv(I:C,O:16,K:3,P:1,S:1), Leaky ReLU:0.2
L2	Conv(I:16,O:16,K:3,P:1,S:1), Leaky ReLU:0.2
L3	Conv(I:16,O:32,K:3,P:1,S:2), Leaky ReLU:0.2
L4	Conv(I:32,O:32,K:3,P:1,S:1), Leaky ReLU:0.2
L5	Conv(I:32,O:64,K:3,P:1,S:2), Leaky ReLU:0.2
L6	Conv(I:64,O:64,K:3,P:1,S:1), Leaky ReLU:0.2
L7	Conv(I:64,O:128,K:3,P:1,S:2), Leaky ReLU:0.2
L8	Conv(I:128,O:128,K:3,P:1,S:1), Leaky ReLU:0.2
L9	Conv(I:128,O:256,K:3,P:1,S:2), Leaky ReLU:0.2
L10	Conv(I:256,O:256,K:3,P:1,S:1), Leaky ReLU:0.2

Table 5: The network architecture of Encoder E. Conv denotes a convolutional layer. I, O, K, P, and S denote the number of input channels, the number of output channels, a kernel size, a padding size, and a stride size, respectively.

Resblocks We place four stacked residual blocks [9] with a kernel size of 3 and a stride of 1. Batch normalization [14] follows each convolutional block, and ReLU is used as the activation function.

Discriminator We adopt our discriminator architecture as Patch-GAN [15]. We utilize the LSGAN [30] objective for the stable training.

Training Details For all the experiments, our network is trained using Adam optimizer [19] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We set an initial learning rate for the generator as 0.0001 and that for the discriminator as 0.0002. We train the model for the first 100 epochs using the same learning rate, and then we linearly decay it to zero until the 200 epochs. We set the margin value $\gamma = 12$ for our triplet loss (Eq. 5 in the paper). The batch size is set as 16. The parameters of all our models are initialized according to the normal distribution which has a mean as 0.0 and a standard deviation as 0.02.

Baselines We exploit Sun [40] and Style2Paints [28] as the sketch image colorization methods, Huang [2018] [13], and Lee [24] as the image translation methods and Huang [2017] [12] as the style transfer method as our baselines. For Style2Paints [28], we generate the images based on the publicly available Style2Paints V3 in a similar manner to Tag2pix [18]. For the other methods, we utilize the officially available codes to colorize images after training them on our datasets.

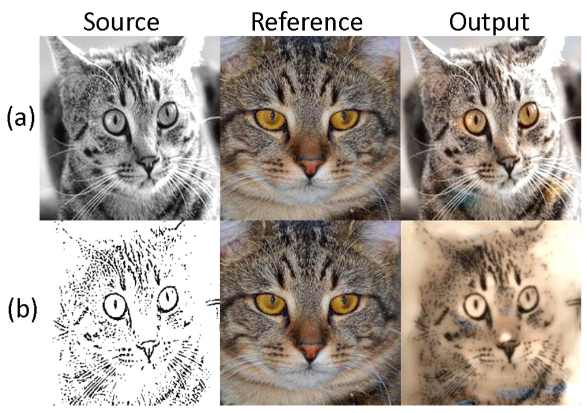


Figure 11: Qualitative results of our Zhang *et al.* [47] given gray-scale source image (row (a)) and sketch image (row (b)). In contrast to the output in the row (a), output in (b) fails to colorize the eyes with the color from the reference and spreads the yellow color over the face.

A.4. Comparison to Zhang *et al.* (2019) [47].

In this section, we discuss the detailed comparison between our method and Zhang *et al.*. These two works have similarity in that they both exploit geometric distortion for data augmentation and semantic correspondence module for color guidance. However the significant difference of our model against Zhang *et al.* lies in (1) direct supervision of semantic correspondence and (2) generalized attention module.

Direct supervision Our model directly supervises the attention module via a triplet loss, which enables the optimization of the attention module in an end-to-end manner. This fully trainable encoder encourages to generate plausible results over a wide range of datasets from real-world photos to comic images, as show in Fig. 4 of the paper and Fig. 17. In contrast, Zhang *et al.* requires a pre-trained, already reliable attention module, which is only indirectly supervised via a so-called contextual loss. According to Geirhos *et al.* (2019) [5], the features extracted from the ImageNet pre-trained encoder may be severely degraded for a sketch image due to large domain shifts. In this sense, Zhang *et al.*'s work may not be easily applicable to sketch image colorization tasks, and the examples of failure case are shown in Fig. 11. We reimplemented the code of Zhang *et al.*, trained and tested the model over cat dataset. As this baseline exploits the ImageNet pre-trained encoder, row (a) shows that it produces the plausible colorized output given gray-scale source image. However, when given information scarce sketch image (row (b)), it fails to obtain the dense correspondence with the reference image, resulting in degraded output.

Generalized attention module Inspired by the self-attention module in the Transformer networks, our attention module involves different query, key, and value mappings for flexibility, while Zhang *et al.* use a relatively simple module. More importantly, in terms of value vectors, Zhang *et al.* uses only raw color values, but ours uses all the available low- to high- level semantic information extracted from multiple layers. In this respect, ours is capable of transferring significantly richer contextual information than just low-level color information.

A.5. Colorization without reference.

Our main scope is focused on the colorization task with a reference available, but we can easily extend our method for no-reference cases by occasionally providing a zero-filled image as a reference to the networks during the training time. We feed the zero-filled image to our model as a reference with a ratio of 9:1 at the training time. As shown in Fig. 12, we confirm that our network still generates a reasonable quality of colorization output at test time. In this case, the zero-filled reference image does not have any information to guide. Therefore, the model is encouraged to synthesize an output image with colors that often appear in trainset conditioned on the sketch image. We recall that the main goal of this work is not restricted to generating the original image.

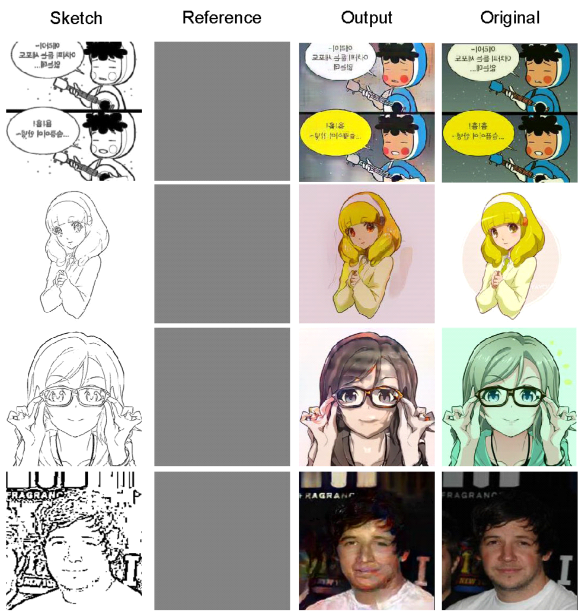


Figure 12: A qualitative example when there is no reference image. Our model takes the first column image (sketch) as a target and the second column image (zero-filled reference) to synthesize the third column image (output). The results of first row, second-to-third rows, last row are obtained from our model trained for Yumi's Cells [32], Tag2pix [18], and CelebA [27], respectively.

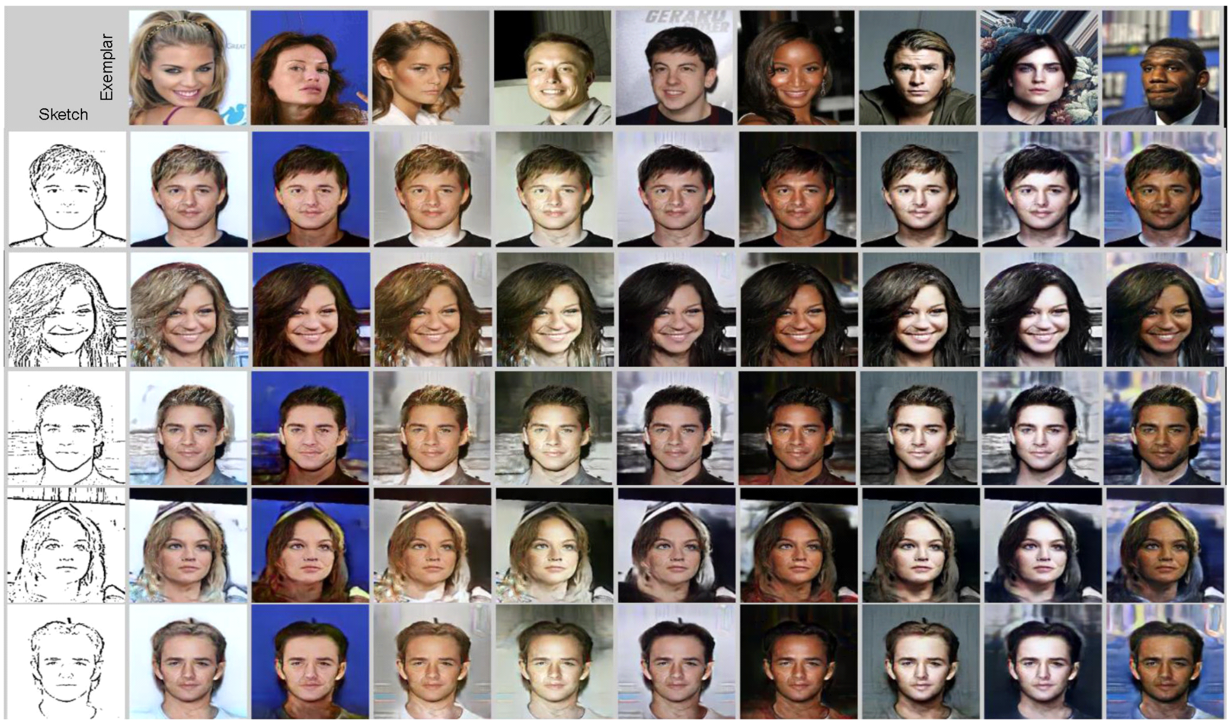


Figure 13: Qualitative results of our method on the CelebA [27] dataset.

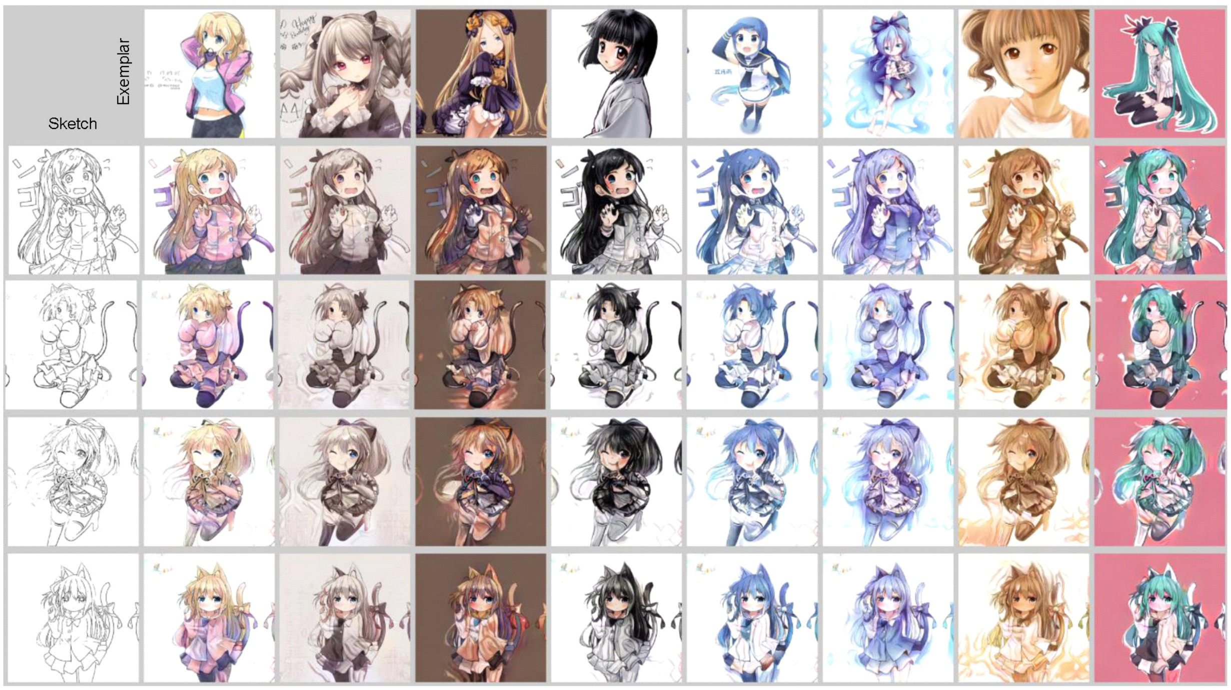


Figure 14: Qualitative results of our method on the Tag2pix [18] dataset.

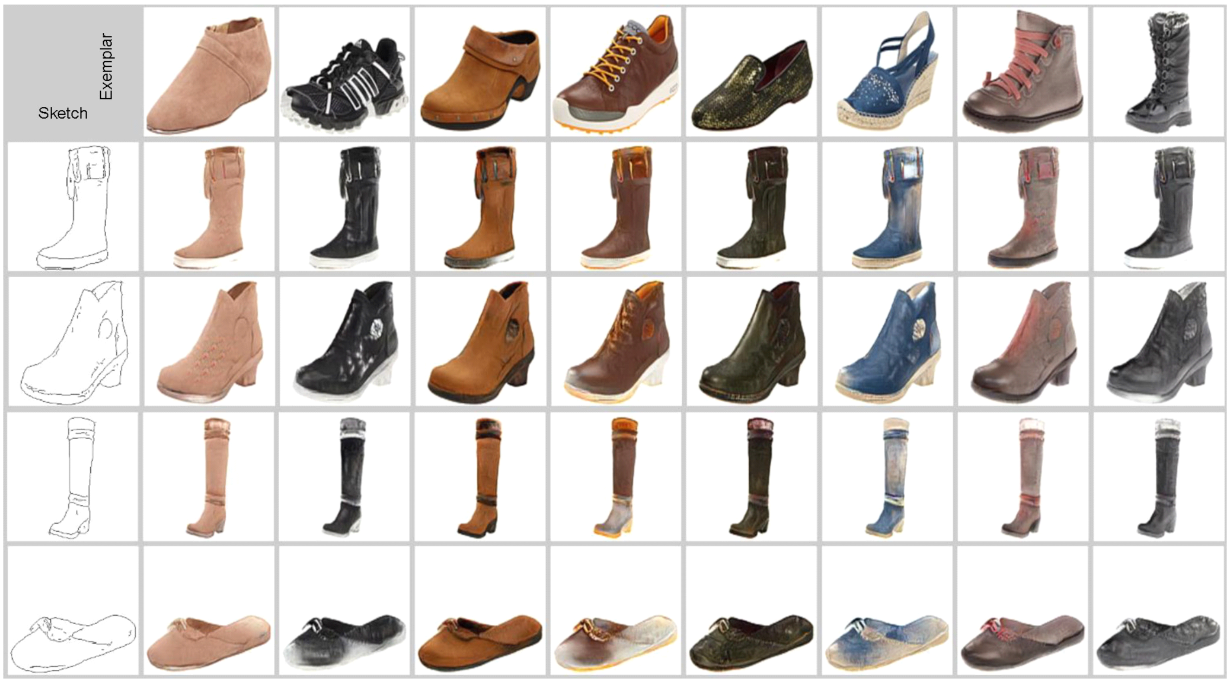


Figure 15: Qualitative results of our method on the Edges→Shoes [15] dataset.

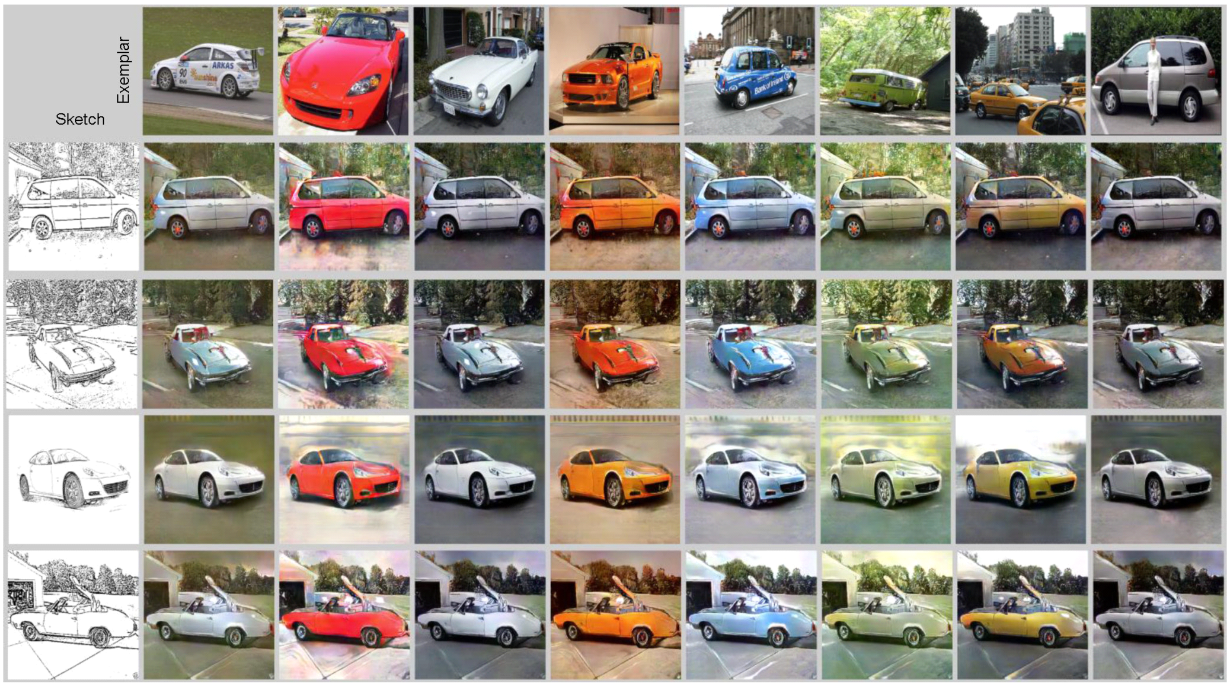


Figure 16: Qualitative results of our method on the ImageNet [36] dataset.

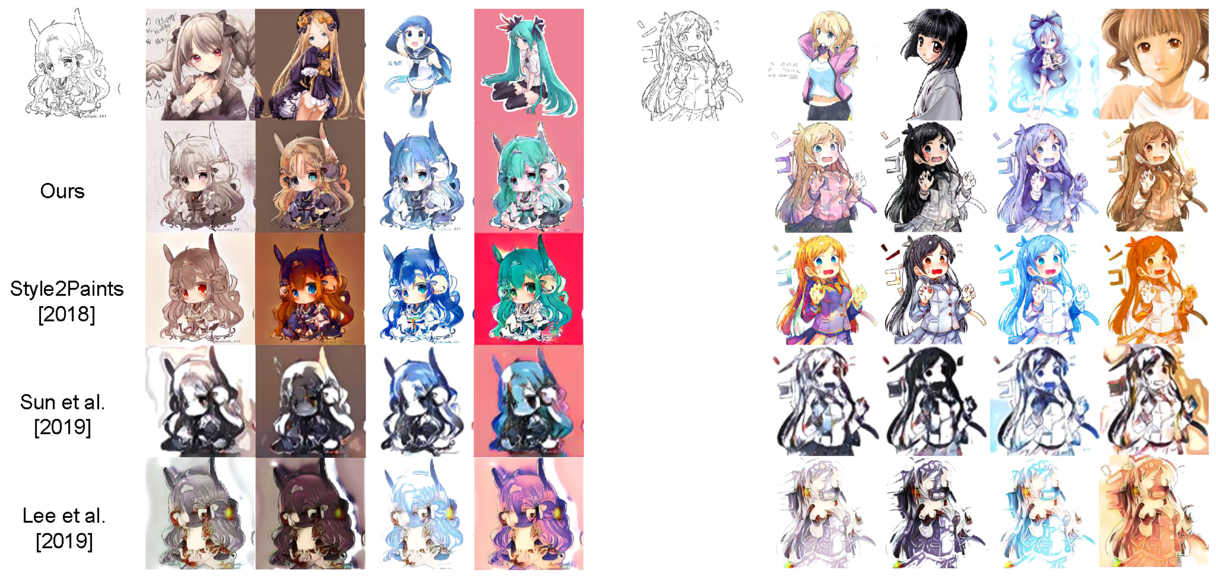


Figure 17: Qualitative comparisons with the baselines on the Tag2pix dataset.

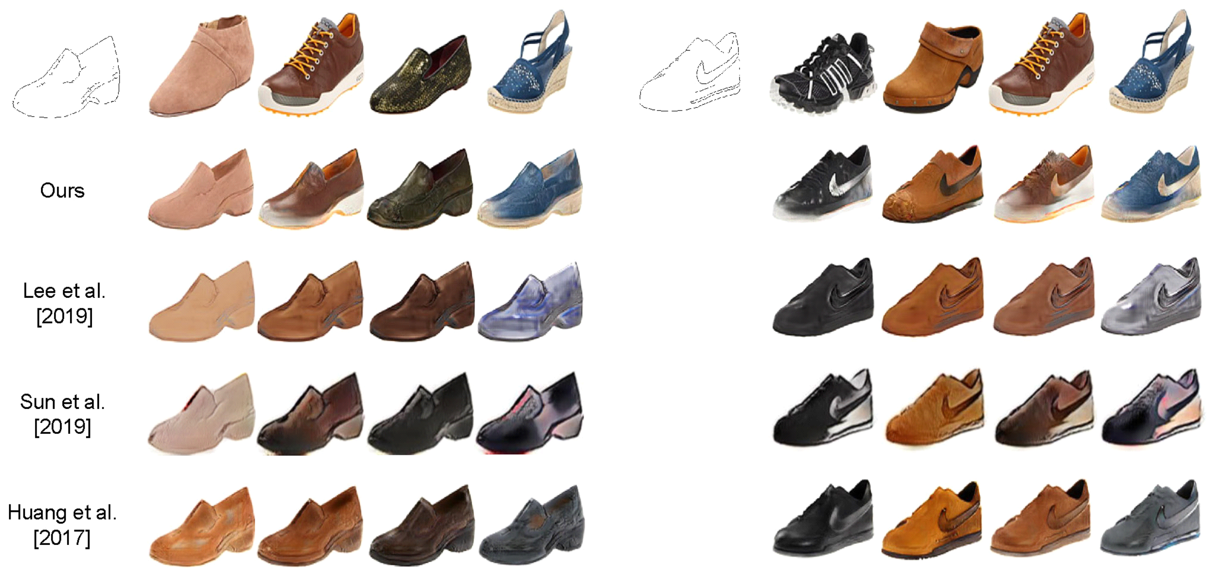


Figure 18: Qualitative results of our method on the Edges→Shoes dataset.

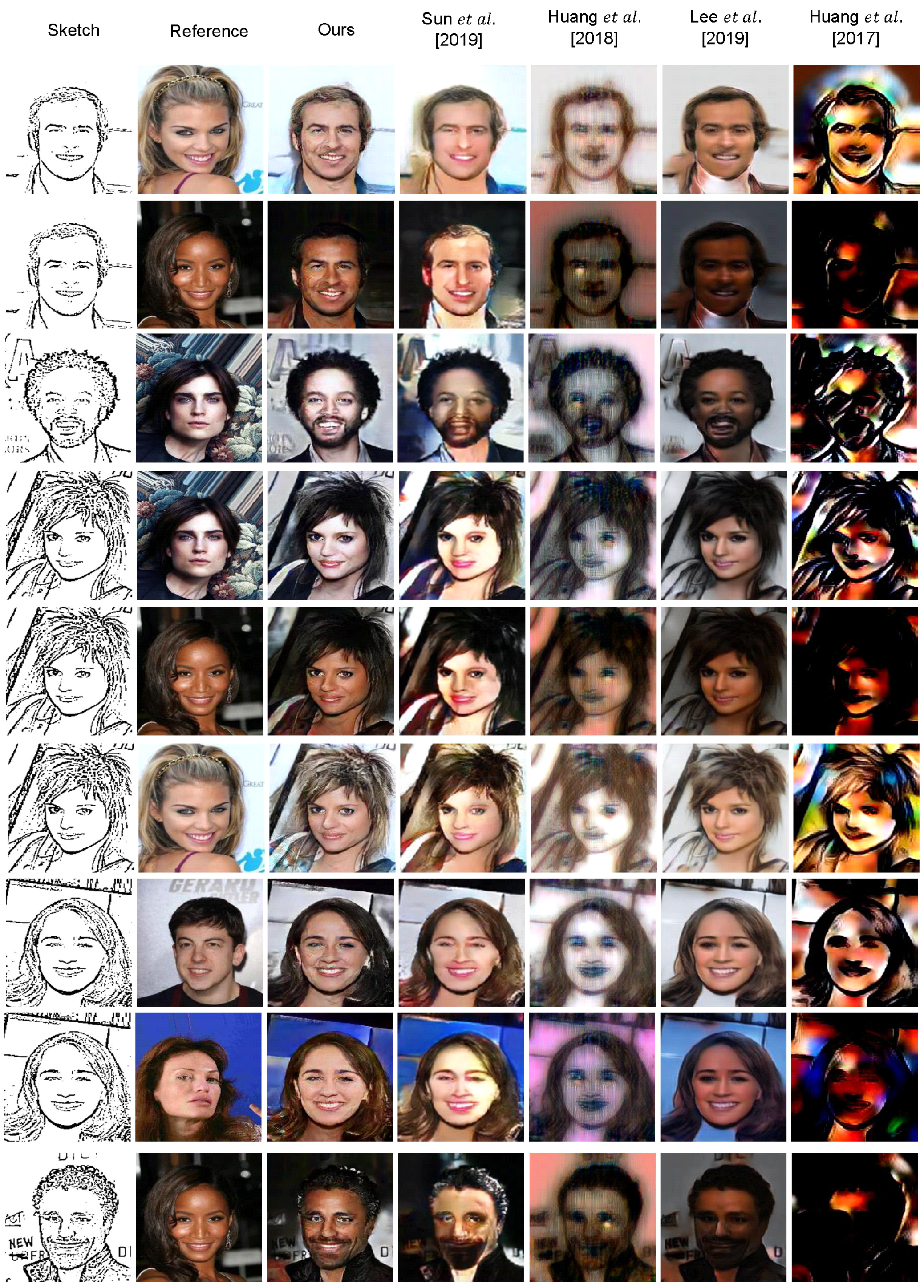


Figure 19: Qualitative comparisons with baselines on the CelebA dataset.

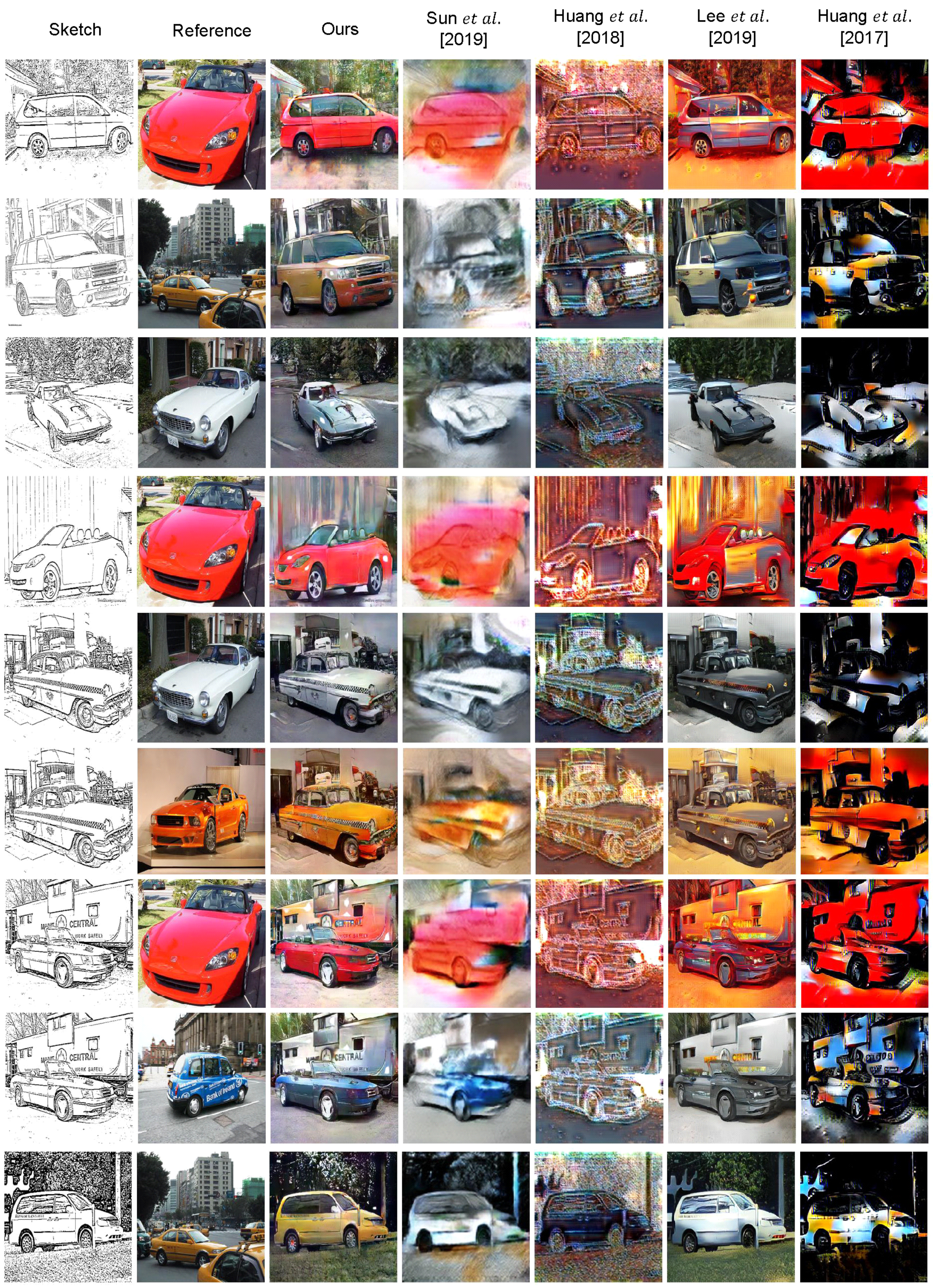


Figure 20: Qualitative comparisons with baselines on the ImageNet [36] dataset.

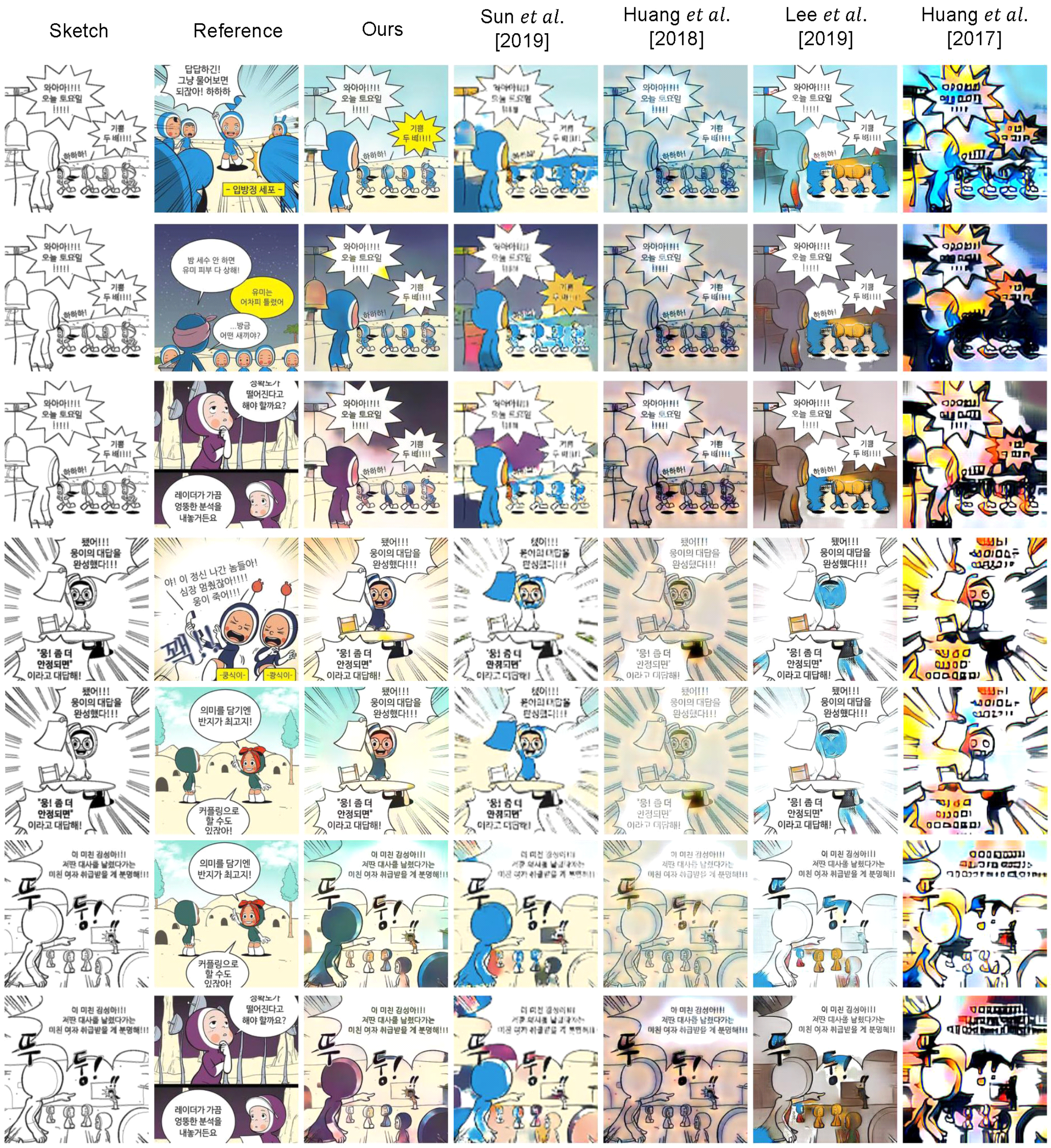


Figure 21: Qualitative comparisons with baselines on the Yumi's Cells [32] dataset.

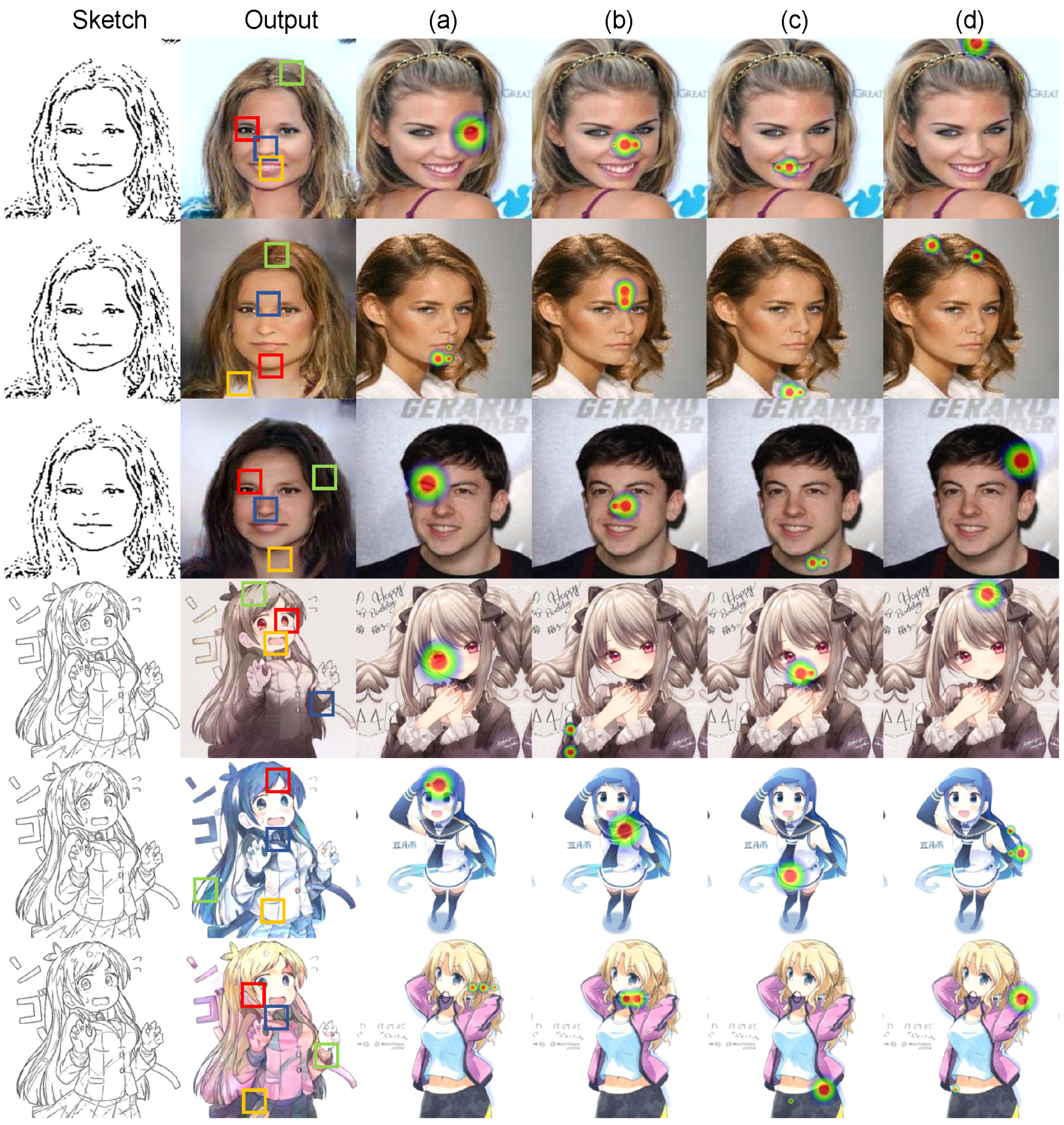


Figure 22: The visualization of attention maps on CelebA and Tag2pix dataset. The colored squares on the second column indicate the query region and corresponding key regions are highlighted in the next four columns. The different color of square means the different query region, and each red, blue, yellow, and green corresponds with the column (a), (b), (c), and (d), respectively.

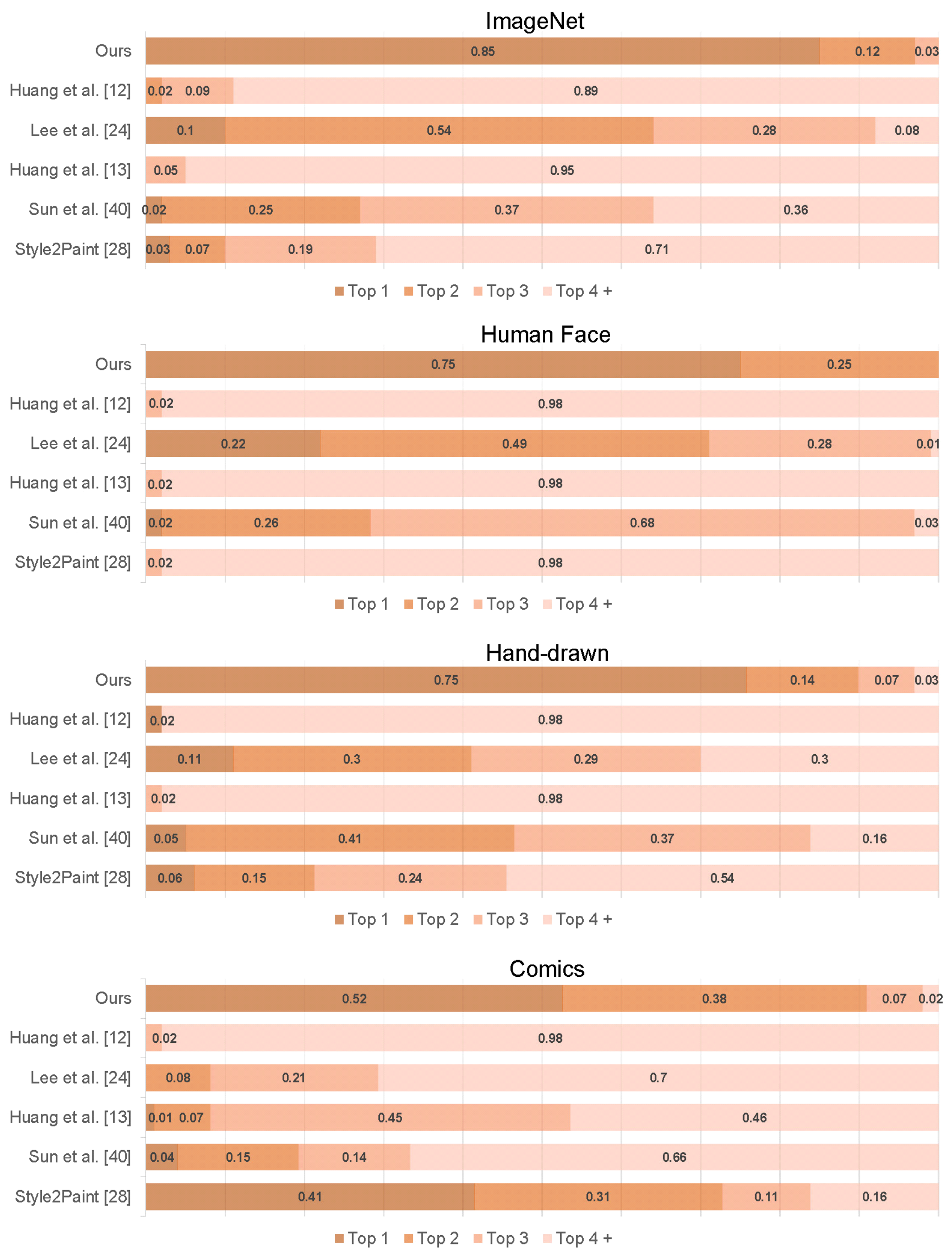


Figure 23: The results of the user study for comparison between our model and existing baselines. Question type 1: Overall Colorization Quality and Realism.

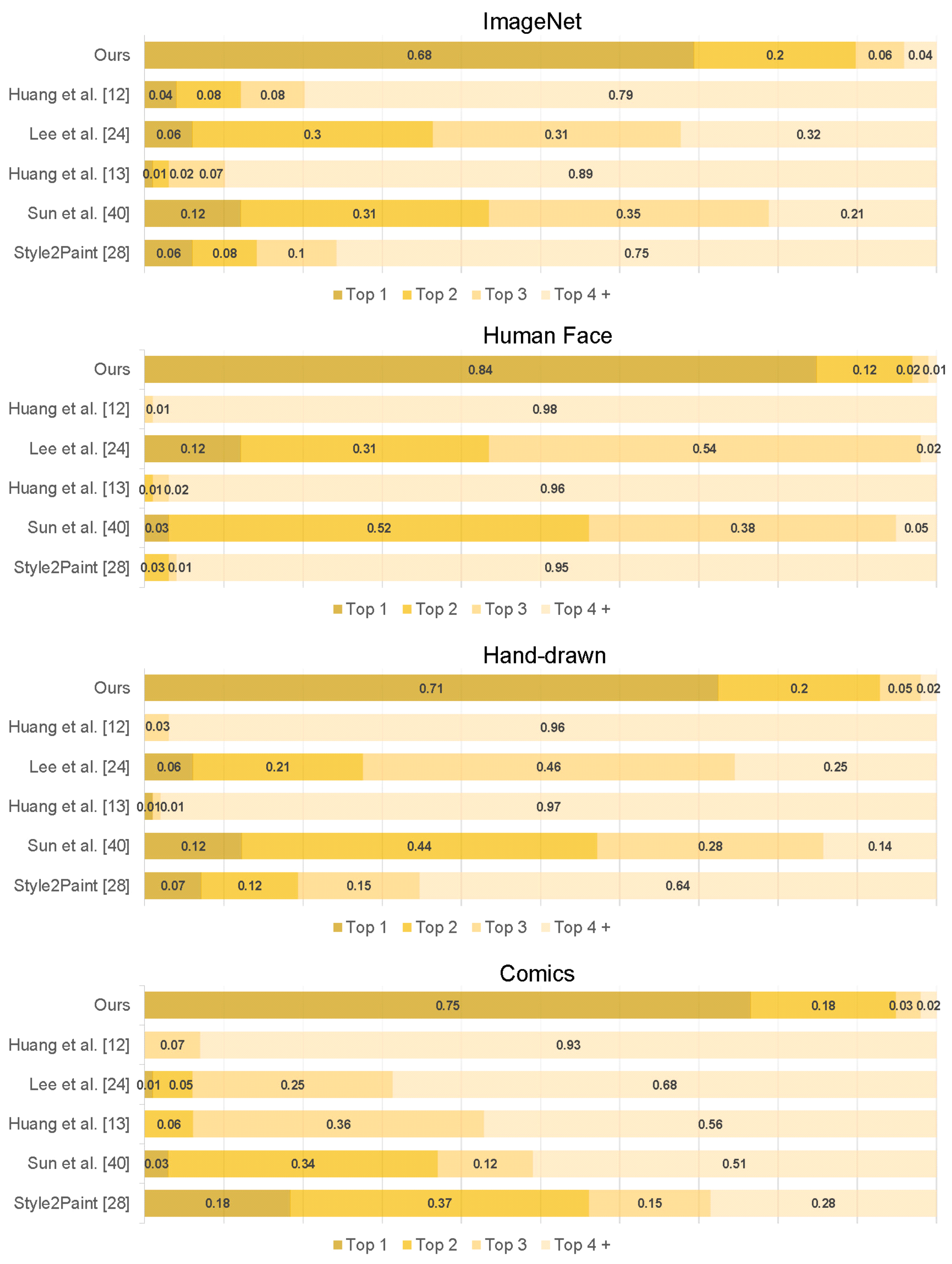


Figure 24: The results of the user study for comparison between our model and existing baselines. Question type 2: Detailed Reflection of Reference.

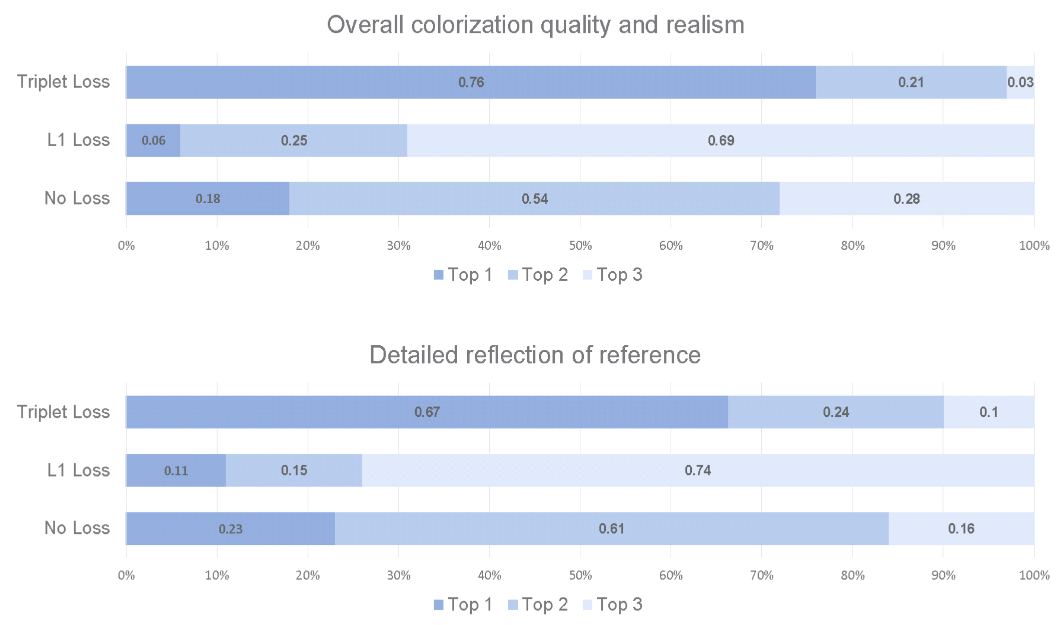


Figure 25: The results of the user study for comparison between model with triplet loss, L_1-loss and no loss. The percentages are averaged over all the datasets.