# STEFANN: Scene Text Editor using Font Adaptive Neural Network

Prasun Roy[1]*, Saumik Bhattacharya[2]*, Subhankar Ghosh[1]*, and Umapada Pal[1]
[1]Indian Statistical Institute, Kolkata, India
[2]Indian Institute of Technology, Kharagpur, India
https://prasunroy.github.io/stefann

## Abstract

*Textual information in a captured scene plays an important role in scene interpretation and decision making. Though there exist methods that can successfully detect and interpret complex text regions present in a scene, to the best of our knowledge, there is no significant prior work that aims to modify the textual information in an image. The ability to edit text directly on images has several advantages including error correction, text restoration and image reusability. In this paper, we propose a method to modify text in an image at character-level. We approach the problem in two stages. At first, the unobserved character (target) is generated from an observed character (source) being modified. We propose two different neural network architectures – (a)* ***FANnet*** *to achieve structural consistency with source font and (b)* ***Colornet*** *to preserve source color. Next, we replace the source character with the generated character maintaining both geometric and visual consistency with neighboring characters. Our method works as a unified platform for modifying text in images. We present the effectiveness of our method on COCO-Text and ICDAR datasets both qualitatively and quantitatively.*

## 1. Introduction

Text is widely present in different design and scene images. It contains important contextual information for the readers. However, if any alteration is required in the text present in an image, it becomes extremely difficult for several reasons. For instance, a limited number of observed characters makes it difficult to generate unobserved characters with sufficient visual consistency. Also, different natural conditions, like brightness, contrast, shadow, perspective distortion, complex background, etc., make it harder to replace a character directly in an image. The main motivation of this work is to design an algorithm for editing textual information present in images in a convenient way similar to the conventional text editors.


**(a)** **(b)**

Figure 1. Examples of text editing using STEFANN: **(a)** Original images from ICDAR dataset; **(b)** Edited images. It can be observed that STEFANN can edit multiple characters in a word (top row) as well as an entire word (bottom row) in a text region.

Earlier, researchers have proposed font synthesis algorithms based on different geometrical features of fonts [6, 24, 27]. These geometrical models neither generalize the wide variety of available fonts nor can be applied directly to an image for character synthesis. Later, researchers have addressed the problem of generating unobserved characters of a particular font from some defined or random set of observations using deep learning algorithms [4, 7, 31]. With the emergence of Generative Adversarial Network (GAN) models, the problem of character synthesis has also been addressed using GAN-based algorithms [2, 19]. Though GAN-based font synthesis could be used to estimate the target character, several challenges make the direct implementation of font synthesis for scene images difficult. Firstly, most of the GAN-based font synthesis models require an explicit recognition of the source character. As recognition of text in scene images is itself a challenging problem, it is preferable if the target characters can be generated without

*These authors contributed equally to this work.

a recognition step. Otherwise, any error in the recognition process would accumulate, and make the entire text editing process unstable. Secondly, it is often observed that a particular word in an image may have a mixture of different font types, sizes, colors, etc. Even depending on the relative location of the camera and the texts in the scene, each character may experience a different amount of perspective distortion. Some GAN-based models [2, 19] require multiple observations of a font-type to faithfully generate unobserved characters. A multiple observation-based generation strategy requires a rigorous distortion removal step before applying generative algorithms. Thus, rather than a word-level generation, we follow a character-level generative model to accommodate maximum flexibility.

**Contributions**: To the best of our knowledge, this is the first work that attempts to modify texts in scene images. For this purpose, we design a generative network that adapts to the font features of a single character and generates other necessary characters. We also propose a model to transfer the color of the source character to the target character. The entire process works without any explicit character recognition.

To restrict the complexity of our problem, we limit our discussion to the scene texts with upper-case non-overlapping characters. However, we demonstrate in Figs. 5 and 13 that the proposed method can also be applied for lower-case characters and numerals.

## 2. Related Works

Because of its large potential, character synthesis from a few examples is a well-known problem. Previously, several pieces of work tried to address the problem using geometrical modeling of fonts [6, 24, 27]. Different synthesis models are also proposed by researchers explicitly for Chinese font generation [19, 37]. Along with statistical models [24] and bilinear factorization [30], machine learning algorithms are used to transfer font features. Recently, deep learning techniques also become popular in the font synthesis problem. Supervised [31] and definite samples [4] of observations are used to generate the unknown samples using deep neural architecture. Recently, Generative Adversarial Network (GAN) models are found to be effective in different image synthesis problems. GANs can be used in image style transfer [10], structure generation [13] or in both [2]. Some of these algorithms achieved promising results in generating font structures [7, 19], whereas some exhibits the potential to generate complex fonts with color [2]. To the best of our knowledge, these generative algorithms work with text images that are produced using design software, and their applicability to edit real scene images are unknown. Moreover, most of the algorithms [2, 4] require explicit recognition of the source characters to generate the unseen character set. This may create difficulty in our problem as text recognition in scene images is itself a challenging problem [3, 11, 21] and any error in the recognition step may affect the entire generative process. Character generation from multiple observations is also challenging for scene images as the observed characters may have distinctively different characteristics like font types, sizes, colors, perspective distortions, etc.

Convolutional Neural Network (CNN) is proved to be effective in style transfer with generative models [10, 17, 18]. Recently, CNN models are used to generate style and structure with different visual features [9]. We propose a CNN-based character generation network that works without any explicit recognition of the source characters. For a natural-looking generation, it is also important to transfer the color and texture of the source character to the generated character. Color transfer is a widely explored topic in image processing [25, 28, 35]. Though these traditional approaches are good for transferring global colors in images, most of them are inappropriate for transferring colors in more localized character regions. Recently, GANs are also employed in color transfer problem [2, 16]. In this work, we introduce a CNN-based color transfer model that takes the color information present in the source character and transfer it to the generated target character. The proposed color transfer model not only transfers solid colors from source to target character, it can also transfer gradient colors keeping subtle visual consistency.

## 3. Methodology

The proposed method is composed of the following steps: (1) Selection of the source character to be replaced, (2) Generation of the binary target character, (3) Color transfer and (4) Character placement. In the first step, we manually select the text area that requires to be modified. Then, the algorithm detects the bounding boxes of each character in the selected text region. Next, we manually select the bounding box around the character to be modified and also specify the target character. Based on these user inputs, the target character is generated, colorized and placed in the inpainted region of the source character.

### 3.1. Selection of the source character

Let us assume that $I$ is an image that has multiple text regions, and $\Omega$ is the domain of a text region that requires modification. The region $\Omega$ can be selected using any text detection algorithm [5, 20, 36]. Alternatively, a user can select the corner points of a polygon that bounds a word to define $\Omega$. In this work, we use EAST [38] to tentatively mark the text regions, followed by a manual quadrilateral corner selection to define $\Omega$. After selecting the text region, we apply the MSER algorithm [8] to detect the binary masks of individual characters present in the region $\Omega$. However, MSER alone cannot generate a sharp mask for most of the

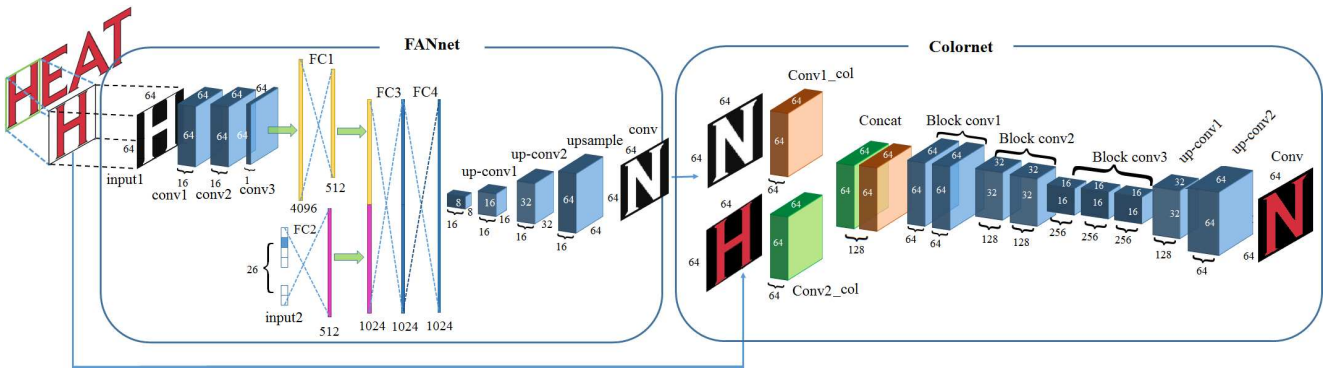


Figure 2. Architecture of FANnet and Colornet. At first, the target character ('N') is generated from the source character ('H') by FANnet keeping structural consistency. Then, the source color is transferred to the target by Colornet preserving visual consistency. Layer names in the figure are: *conv* = 2D convolution, *FC* = fully-connected, *up-conv* = upsampling + convolution.

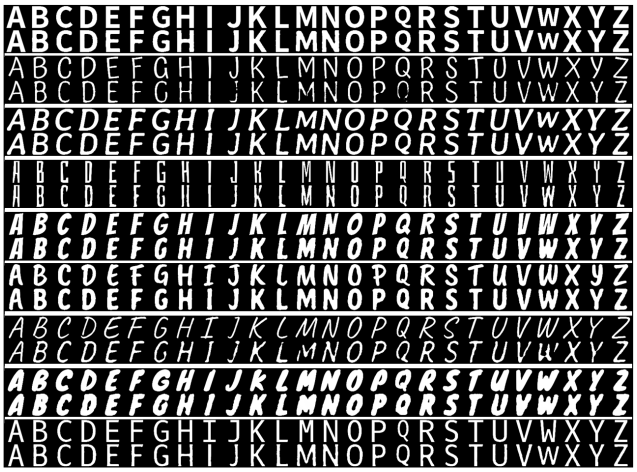

Figure 3. Generation of target characters using FANnet. In each image block, the upper row shows the ground truth and the bottom row shows the generated characters when the network has observed one particular source character ('A') in each case.

characters. Thus, we calculate the final binarized image $I_c$ defined as

$$I_c(\mathbf{p}) = \begin{cases} I_M(\mathbf{p}) \bigodot I_B(\mathbf{p}) & \text{if } \mathbf{p} \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

where $I_M$ is the binarized output of the MSER algorithm [8] when applied on $I$, $I_B$ is the binarized image of $I$ and $\bigodot$ denotes the element-wise product of matrices. The image $I_c$ contains the binarized characters in the selected region $\Omega$. If the color of the source character is darker than its background, we apply inverse binarization on $I$ to get $I_B$.

Assuming the characters are non-overlapping, we apply a connected component analysis and compute the minimum bounding rectangles of each connected component. If there are $N$ number of connected components present in a scene, $C_n \subseteq \Omega$ denotes the $n^{\text{th}}$ connected area where $0 < n \leq N$. The bounding boxes $B_n$ contain the same indices as the


Figure 4. Color transfer using Colornet: **(a)** Binary target character; **(b)** Color source character; **(c)** Ground truth; **(d)** Color transferred image. It can be observed that Colornet can successfully transfer solid color as well as gradient color.

connected areas that they are bounding. The user specifies the indices that they wish to edit. We define $\Theta$ as the set of indices that require modification, such that $|\Theta| \leq N$, where $|.|$ denotes the cardinality of a set. The binarized images $I_{C_\theta}$ associated with components $C_\theta$, $\theta \in \Theta$ are the source characters, and with proper padding followed by scaling (discussed in Sec. 3.2), they individually act as the input of the font generation network. Each $I_{C_\theta}$ has the same dimension with the bounding box $B_\theta$.

### 3.2. Generation of the binary target character

Conventionally, most of the neural networks take square images as input. However, as $I_{C_\theta}$ may have different aspect ratios depending on the source character, font type, font size etc., a direct resizing of $I_{C_\theta}$ would distort the actual font features of the character. Rather, we pad $I_{C_\theta}$ maintaining its aspect ratio to generate a square binary image $I_\theta$ of size $m_\theta \times m_\theta$ such that, $m_\theta = \max(h_\theta, w_\theta)$, where $h_\theta$ and $w_\theta$ are the height and width of bounding box $B_\theta$ respectively, and $\max(.)$ is a mathematical operation that finds the maximum value. We pad both sides of $I_{C_\theta}$ along $x$ and $y$ axes with $p_x$ and $p_y$ respectively to generate $I_\theta$ such that

$$p_x = \left\lceil \frac{m_\theta - w_\theta}{2} \right\rceil, \quad p_y = \left\lceil \frac{m_\theta - h_\theta}{2} \right\rceil$$

followed by reshaping $I_\theta$ to a square dimension of $64 \times 64$.

#### 3.2.1 Font Adaptive Neural Network (FANnet)

Our generative font adaptive neural network (FANnet) takes two different inputs – an image of the source character of size $64 \times 64$ and a one-hot encoding $\mathbf{v}$ of length 26 of the target character. For example, if our target character is 'H', then $\mathbf{v}$ has the value 1 at index 7 and 0 in every other location. The input image passes through three convolution layers having 16, 16 and 1 filters respectively, followed by flattening and a fully-connected (FC) layer FC1. The encoded vector $\mathbf{v}$ also passes through an FC layer FC2. The outputs of FC1 and FC2 give 512 dimensional latent representations of respective inputs. Outputs of FC1 and FC2 are concatenated and followed by two more FC layers, FC3 and FC4 having 1024 neurons each. The expanding part of the network contains reshaping to a dimension $8 \times 8 \times 16$ followed by three 'up-conv' layers having 16, 16 and 1 filters respectively. Each 'up-conv' layer contains an upsampling followed by a 2D convolution. All the convolution layers have kernel size $3 \times 3$ and ReLU activation. The architecture of FANnet is shown in Fig. 2. The network minimizes the mean absolute error (MAE) while training with Adam optimizer [14] with learning rate $lr = 10^{-3}$, momentum parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and regularization parameter $\epsilon = 10^{-7}$.

We train FANnet with 1000 fonts with all 26 upper-case character images as inputs and 26 different one-hot encoded vectors for each input. It implies that for 1000 fonts, we train the model to generate any of the 26 upper-case target character images from any of the 26 upper-case source character images. Thus, our training dataset has a total of 0.6760 million input pairs. The validation set contains 0.2028 million input pairs generated from another 300 fonts. We select all the fonts from Google Fonts database [12]. We apply the Otsu thresholding technique [22] on the grayscale output image of FANnet to get a binary target image.

### 3.3. Color transfer

It is important to have a faithful transfer of color from the source character for a visually consistent generation of the target character. We propose a CNN based architecture, named Colornet, that takes two images as input – colored source character image and binary target character image. It generates the target character image with transferred color from the source character image. Each input image goes through a 2D convolution layer, Conv1_col and Conv2_col for input1 and input2 respectively. The outputs of Conv1_col and Conv2_col are batch-normalized and concatenated, which is followed by three blocks of convolution layers with two max-pooling layers in between. The expanding part of Colornet contains two 'up-conv' layers followed by a 2D convolution. All the convolution layers have kernel size $3 \times 3$ and Leaky-ReLU activation with $\alpha = 0.2$. The architecture of Colornet is shown in Fig. 2. The network minimizes the mean absolute error (MAE) while training with Adam optimizer that has the same parameter settings as mentioned in Sec. 3.2.1.

We train Colornet with synthetically generated image pairs. For each image pair, the color source image and the binary target image both are generated using the same font type randomly selected from 1300 fonts. The source color images contain both solid and gradient colors so that the network can learn to transfer a wide range of color variations. We perform a bitwise-AND between the output of Colornet and the binary target image to get the final colorized target character image.

### 3.4. Character placement

Even after the generation of target characters, the placement requires several careful operations. First, we need to remove the source character from $I$ so that the generated target character can be placed. We use image inpainting [29] using $W(I_{C_\theta}, \psi)$ as a mask to remove the source character, where $W(I_b, \psi)$ is the dilation operation on any binary image $I_b$ using the structural element $\psi$. In our experiments, we consider $\psi = 3 \times 3$. To begin the target character placement, first the output of Colornet is resized to the dimension of $I_\theta$. We consider that the resized color target character is $R_\theta$ with minimum rectangular bounding box $B_\theta^R$. If $B_\theta^R$ is smaller or larger than $B_\theta$, then we need to remove or add the region $B_\theta \setminus B_\theta^R$ accordingly so that we have the space to position $R_\theta$ with proper inter-character spacing. We apply the content-aware seam carving technique [1] to manipulate the non-overlapping region. It is important to mention that if $B_\theta^R$ is smaller than $B_\theta$ then after seam carving, the entire text region $\Omega$ will shrink to a region $\Omega_s$, and we also need to inpaint the region $\Omega \setminus \Omega_s$ for consistency. However, both the regions $B_\theta \setminus B_\theta^R$ and $\Omega \setminus \Omega_s$ are considerably small and are easy to inpaint for upper-case characters. Finally, we place the generated target character on the seam carved image such that the centroid of $B_\theta^R$ overlaps with the centroid of $B_\theta$.

## 4. Results

We tested[1] our algorithm on COCO-Text and ICDAR datasets. The images in the datasets are scene images with texts written with different unknown fonts. In Fig. 5, we show some of the images that are edited using STEFANN. In each image pair, the left image is the original image and the right image is the edited image. In some of the images, several characters are edited in a particular text region, whereas in some images, several text regions are edited in a single image. It can be observed that not only the font features and colors are transferred successfully to the target characters, but also the inter-character spacing is main-

[1]Code: https://github.com/prasunroy/stefann


Figure 5. Images edited using STEFANN. In each image pair, the left image is the original image and the right image is the edited image. It can be observed that STEFANN can faithfully edit texts even in the presence of specular reflection, shadow, perspective distortion, etc. It is also possible to edit lower-case characters and numerals in a scene image. STEFANN can easily edit multiple characters and multiple text regions in an image. **More results are included in the supplementary materials.**

tained in most of the cases. Though all the images are natural scene images and contain different lighting conditions, fonts, perspective distortions, backgrounds, etc., in all the cases STEFANN is able to edit the images without any significant visual inconsistency.

**Evaluation and ablation study of FANnet:** To evaluate the performance of the proposed FANnet model, we take one particular source character and generate all possible target characters. We repeat this process for every font in the test set. The outputs for some randomly selected fonts are shown in Fig. 3. Here, we only provide an image of character 'A' as the source in each case and generate all 26 characters. To quantify the generation quality of FANnet, we select one source character at a time as input and measure the average structural similarity index (ASSIM) [34] of all 26 generated target characters against respective ground truth images over a set of 300 test fonts. In Fig. 6, we show the average SSIM of the generated characters for each different source character. It can be seen from the ASSIM scores that some characters, like 'I' and 'L', are less informative as they

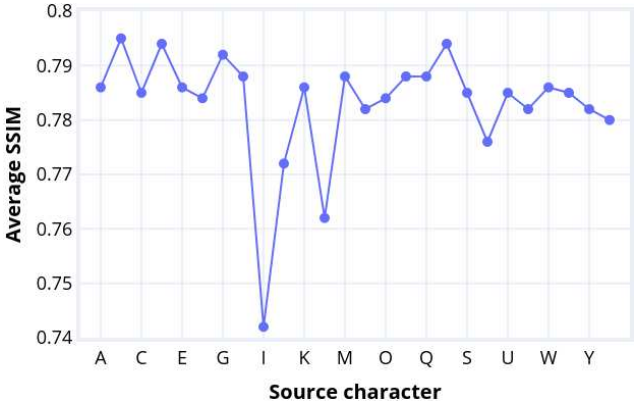


Figure 6. Average SSIM of the generated characters for each different source character.

Table 1. Ablation study of FANnet architecture. Layer names are similar to Fig. 2.

| # | Excluded layer(s) | ASSIM |
|---|---|---|
| 1 | up-conv1 + up-conv2 | 0.5664 |
| 2 | FC4 + up-conv2 | 0.6426 |
| 3 | FC1 + FC2 + up-conv2 | 0.6718 |
| 4 | None (Proposed FANnet) | **0.7712** |
| 5 | All (FCN) | 0.1332 |

produce lower ASSIM values, whereas characters, like 'B' and 'M', are structurally more informative in the generation process.

We perform ablation studies using ASSIM score to validate the proposed FANnet architecture. The results are shown in Table 1. As Fully Convolutional Networks (FCN) are often used in generative models, we tried to develop FANnet using FCN models at first. Unfortunately, none of the developed FCN architecture for FANnet worked properly. To demonstrate it, we have included an FCN architecture in Table 1 which is similar to ColorNet with two inputs – one is the source character image and another is the target character in standard 'Times New Roman' font. Table 1 clearly shows the motivation of the proposed FANnet model as it achieves the highest ASSIM score in the ablation study. In our ablation study, when the 'up-conv' layer is removed, it is replaced with an upsampling layer by a factor of 2 to maintain the image size. During the ablation study, replacement of ReLU activation with Leaky-ReLU activation gives an ASSIM score of 0.4819. To further analyze the robustness of the generative model, we build another network with the same architecture as described in Sec. 3.2.1, and train it with lower-case character images of the same font database. The output of the model is shown in Fig. 7(a) when only a lower-case character (character 'a') is given as the source image and all the lower-case characters are generated. As shown in Fig. 7(b) and Fig. 7(c), we also observe that the model can transfer some font features even when a lower-case character is provided as the input and we try to generate the upper-case characters or vice versa.

**Evaluation and ablation study of Colornet:** The performance of the proposed Colornet is shown in Fig. 4 for both solid and gradient colors. It can be observed that in both cases, Colornet can faithfully transfer the font color of source characters to target characters. As shown in Fig. 4, the model works equally well for all alphanumeric characters including lower-case and upper-case letters. To understand the functionality of the layers in Colornet, we perform an ablation study and select the best model architecture that faithfully transfers the color. We compare the proposed Colornet architecture with two other variants – Colornet-L and Colornet-F. In Colornet-L, we remove 'Block conv3' and 'up-conv1' layers to perform layer ablation. In Colornet-F,

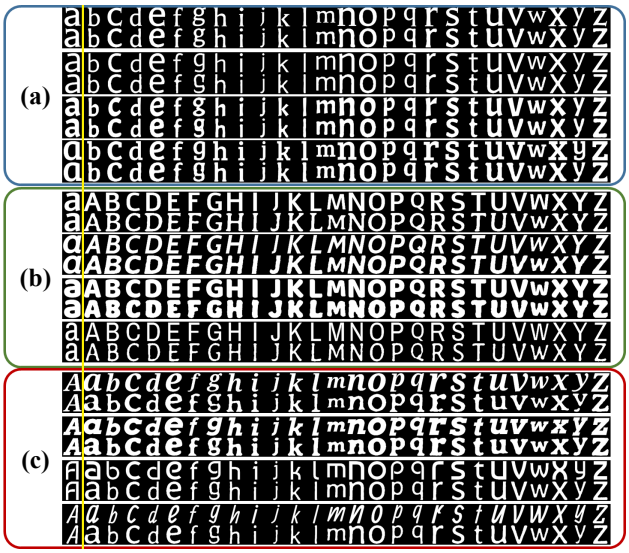


Figure 7. Additional generation results of target characters using FANnet: In each image block, the upper row shows the ground truth, and the bottom row shows the generated characters when the network observes only one particular character: **(a)** lower-case target characters generated from a lower-case source character ('a'); **(b)** upper-case target characters generated from a lower-case source character ('a'); **(c)** lower-case target characters generated from an upper-case source character ('A').

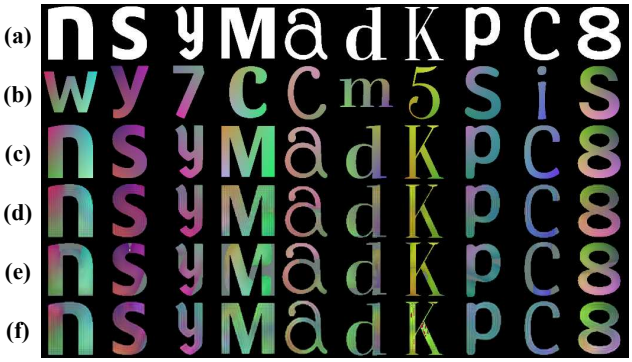


Figure 8. Color transfer results for different models: **(a)** Binary target character; **(b)** Color source character; **(c)** Ground truth; **(d)** Output of the proposed Colornet model; **(e)** Output of Colornet-L; **(f)** Output of Colornet-F. It can be observed that the Colornet architecture discussed in Sec. 3.3 transfers the color from source to target without any significant visual distortion.

we reduce the number of convolution filters to 16 in both 'Conv1_col' and 'Conv2_col' layers to perform filter ablation. The results of color transfer for all three Colornet variants are shown in Fig. 8. It can be observed that Colornet-L produces visible color distortion in the generated images, whereas some color information are not present in the images generated by Colornet-F.

**Comparison with other methods:** To the best of our knowledge, there is no significant prior work that aims to

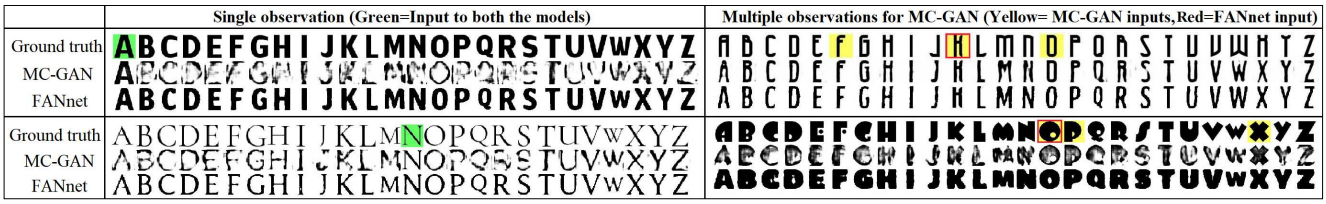


Figure 9. Comparison between MC-GAN and the proposed FANnet architecture. The green color indicates input to both the models when only one observation is available. Yellow colors indicate input to MC-GAN and the red box indicates input to FANnet when 3 random observations are available. The evaluation is performed on the MC-GAN dataset.

Table 2. Comparison of synthetic character generation between MC-GAN and FANnet.

| MC-GAN (1 observation) | | MC-GAN (3 random observations) | | FANnet (1 observation) | |
|---|---|---|---|---|---|
| nRMSE | ASSIM | nRMSE | ASSIM | nRMSE | ASSIM |
| 0.4568 | 0.4098 | 0.3628 | 0.5485 | 0.4504 | 0.4614 |

edit textual information in natural scene images directly. MC-GAN [2] is a recent font synthesis algorithm, but to apply it on scene text a robust recognition algorithm is necessary. Thus on many occasions, it is not possible to apply and evaluate its performance on scene images. However, the generative performance of the proposed FANnet is compared with MC-GAN as shown in Fig. 9. We observe that given a single observation of a source character, FANnet outperforms MC-GAN, but as the number of observations increases, MC-GAN performs better than FANnet. This is also shown in Table 2 where we measure the quality of the generated characters using nRMSE and ASSIM scores. The comparison is done on the dataset [2] which is originally used to train MC-GAN with multiple observations. But in this case, FANnet is not re-trained on this dataset which also shows the adaptive capability of FANnet. In another experiment, we randomly select 1000 fonts for training and 300 fonts for testing from the MC-GAN dataset. When we re-train FANnet with this new dataset, we get an ASSIM score of 0.4836 over the test set. For MC-GAN, we get ASSIM scores of 0.3912 (single observation) and 0.5679 (3 random observations) over the same test set.

We also perform a comparison among MC-GAN [2], Project Naptha [15] and STEFANN assisted text editing schemes on scene images as shown in Fig. 10. For MC-GAN assisted editor, we replace FANnet and Colornet with MC-GAN cascaded with Tesseract v4 OCR engine [26]. Project Naptha provides a web browser extension that allows users to manipulate texts in images. It can be observed that the generative capability of MC-GAN is directly affected by recognition accuracy of the OCR and variation of scale and color among source characters, whereas Project Naptha suffers from weak font adaptability and inpainting.

To understand the perceptual quality of the generated characters, we take opinions from 115 users for 50 different fancy fonts randomly taken from the MC-GAN dataset to evaluate the generation quality of MC-GAN and FANnet. For a single source character, 100% of users opine that the generation of FANnet is better than MC-GAN. For 3 random source characters, 67.5% of users suggest that the generation of FANnet is preferable over MC-GAN.


Figure 10. Comparison among MC-GAN, Project Naptha and STEFANN assisted text editing on scene images. **Top row:** Original images. Text regions to be edited are highlighted with green bounding boxes. OCR predictions for these text regions are shown in respective insets. **Middle row:** MC-GAN or Project Naptha assisted text editing. **Bottom row:** STEFANN assisted text editing.

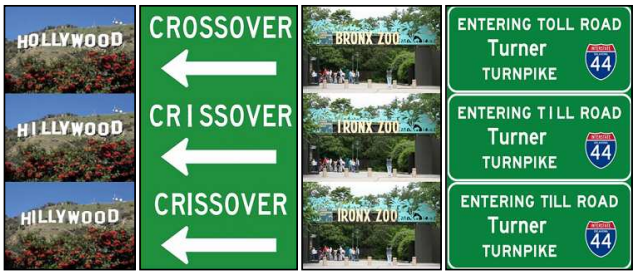


Figure 11. Effectiveness of seam carving. In each column, top row shows the original image, middle row shows the edited image without seam carving and bottom row shows the edited image with seam carving.

**Visual consistency during character placement:** Automatic seam carving of the text region is an important step to perform perceptually consistent modifications maintaining the inter-character distance. Seam carving is particularly required when the target character is 'I'. It can be seen from Fig. 11 that the edited images with seam carving look

visually more consistent than those without seam carving.

**Evaluation of overall generation quality:** To evaluate the overall quality of editing, we perform two opinion based evaluations with 136 viewers. First, they are asked to detect whether the displayed image is edited or not from a set of 50 unedited and 50 edited images shown at random. We get 15.6% true-positive (TP), 37.1% true-negative (TN), 12.8% false-positive (FP) and 34.4% false-negative (FN), which shows that the response is almost random. Next, the same viewers are asked to mark the edited character(s) from another set of 50 edited images. In this case, only 11.6% of edited characters are correctly identified by the viewers.

## 5. Discussion and Conclusion

The major objective of STEFANN is to perform image editing for error correction, text restoration, image reusability, etc. Few such use cases are shown in Fig. 12. Apart from these, with proper training, it can be used in font adaptive image-based machine translation and font synthesis. To the best of our knowledge, this is the first attempt to develop a unified platform to edit texts directly in images with minimal manual effort. STEFANN handles scene text editing efficiently for single or multiple characters while preserving visual consistency and maintaining inter-character spacing. Its performance is affected by extreme perspective distortion, high occlusion, large rotation, etc. which is expected as these effects are not present in the training data. Also, it should be noted that while training FANnet, we use Google Fonts [12] which contains a limited number of artistic fonts, and we train Colornet with only solid and gradient colors. Thus at present, STEFANN does not accommodate editing complex artistic fonts or unusual texture patterns. The proposed FANnet architecture can also be used to generate lower-case target characters with similar architecture as discussed in Sec. 3.2.1. However in the case of lower-case characters, it is difficult to predict the size of the target character only from the size of the source character. It is mainly because lower-case characters are placed in different 'text zones' [23] and the source character may not be replaced directly if the target character falls into a different text zone. In Fig. 13, we show some images where we edit the lowercase characters with STEFANN. In Fig. 14, we also show some cases where STEFANN fails to edit the text faithfully. The major reason behind the failed cases is an inappropriate generation of the target character. In some cases, the generated characters are not consistent with the same characters present in the scene [Fig. 14(a)], whereas in some cases the font features are not transferred properly [Fig. 14(b)]. We also demonstrate that STEFANN currently fails to work with extreme perspective distortion, high occlusion or large rotation [Fig. 14(c)]. In all the editing examples shown in this paper, the number of characters in a text region is not changed. One of the main limitations


Figure 12. Application of STEFANN. Misspelled words (bounded in Red) are corrected (bounded in Green) in scene images.


Figure 13. Some images where lower-case characters are edited using STEFANN.


(a) (b) (c)

Figure 14. Some images where STEFANN fails to edit text with sufficient visual consistency.

of the present methodology is that the font generative model FANnet generates images with dimension $64 \times 64$. While editing high-resolution text regions, a rigorous upsampling is often required to match the size of the source character. This may introduce severe distortion of the upsampled target character image due to interpolation. In the future, we plan to integrate super-resolution [32, 33] to generate very high-resolution character images that are necessary to edit any design or illustration. Also, we use MSER to extract text regions for further processing. So, if MSER fails to extract the character properly, the generation results will be poor. However, this can be rectified using better character segmentation algorithms. It is worth mentioning that robust image authentication and digital forensic techniques should be integrated with such software to minimize the risk of probable misuses of realistic text editing in images.

## Acknowledgements

We would like to thank NVIDIA Corporation for providing a TITAN X GPU through the GPU Grant Program.

## References


[1] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM SIGGRAPH*, 2007. 4

[2] Samaneh Azadi, Matthew Fisher, Vladimir G Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content GAN for few-shot font style transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 7

[3] Fan Bai, Zhanzhan Cheng, Yi Niu, Shiliang Pu, and Shuigeng Zhou. Edit probability for scene text recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[4] Shumeet Baluja. Learning typographic style: from discrimination to synthesis. *Machine Vision and Applications*, 2017. 1, 2

[5] Michal Busta, Lukas Neumann, and Jiri Matas. Deep TextSpotter: An end-to-end trainable scene text localization and recognition framework. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[6] Neill DF Campbell and Jan Kautz. Learning a manifold of fonts. *ACM Transactions on Graphics (TOG)*, 2014. 1, 2

[7] Jie Chang and Yujun Gu. Chinese typography transfer. *arXiv preprint arXiv:1707.04904*, 2017. 1, 2

[8] Huizhong Chen, Sam S Tsai, Georg Schroth, David M Chen, Radek Grzeszczuk, and Bernd Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *The IEEE International Conference on Image Processing (ICIP)*, 2011. 2, 3

[9] Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016. 2

[10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[11] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Learning to read by spelling: Towards unsupervised text recognition. *arXiv preprint arXiv:1809.08675*, 2018. 2

[12] Google Inc. Google Fonts. https://fonts.google.com/, 2010. 4, 8

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 4

[15] Kevin Kwok and Guillermo Webster. Project Naptha. https://projectnaptha.com/, 2013. 7

[16] Chongyi Li, Jichang Guo, and Chunle Guo. Emerging from water: Underwater image color correction based on weakly supervised color transfer. *IEEE Signal Processing Letters*, 2018. 2

[17] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[18] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. In *ACM SIGGRAPH*, 2017. 2

[19] Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. Auto-encoder guided GAN for Chinese calligraphy synthesis. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2017. 1, 2

[20] Pengyuan Lyu, Cong Yao, Wenhao Wu, Shuicheng Yan, and Xiang Bai. Multi-oriented scene text detection via corner localization and region segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[21] Lukas Neumann. *Scene text localization and recognition in images and videos*. PhD thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, 2017. 2

[22] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics (TSMC)*, 1979. 4

[23] U Pal and BB Chaudhuri. Automatic separation of machine-printed and hand-written text lines. In *International Conference on Document Analysis and Recognition (ICDAR)*, 1999. 8

[24] Huy Quoc Phan, Hongbo Fu, and Antoni B Chan. FlexyFont: Learning transferring rules for flexible typeface synthesis. In *Computer Graphics Forum*, 2015. 1, 2

[25] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 2001. 2

[26] Ray Smith. An overview of the Tesseract OCR engine. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2007. 7

[27] Rapee Suveeranont and Takeo Igarashi. Example-based automatic font generation. In *International Symposium on Smart Graphics*, 2010. 1, 2

[28] Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 2

[29] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 2004. 4

[30] Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural Computation*, 2000. 2

[31] Paul Upchurch, Noah Snavely, and Kavita Bala. From A to Z: Supervised transfer of style and content using deep neural network generators. *arXiv preprint arXiv:1603.02003*, 2016. 1, 2

[32] Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

[33] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN:

Enhanced super-resolution generative adversarial networks. In *The European Conference on Computer Vision Workshops (ECCVW)*, 2018. 8

[34] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 2004. 5

[35] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. In *ACM SIGGRAPH*, 2002. 2

[36] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao. Robust text detection in natural scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013. 2

[37] Baoyao Zhou, Weihong Wang, and Zhanghui Chen. Easy generation of personal Chinese handwritten fonts. In *The IEEE International Conference on Multimedia and Expo (ICME)*, 2011. 2

[38] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. EAST: An efficient and accurate scene text detector. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2