Pacific Graphics 2018
H. Fu, A. Ghosh, and J. Kopf
(Guest Editors)

*Volume 37 (2018), Number 7*

# Modeling Fonts in Context: Font Prediction on Web Designs

Nanxuan Zhao, Ying Cao, and Rynson W.H. Lau

City University of Hong Kong

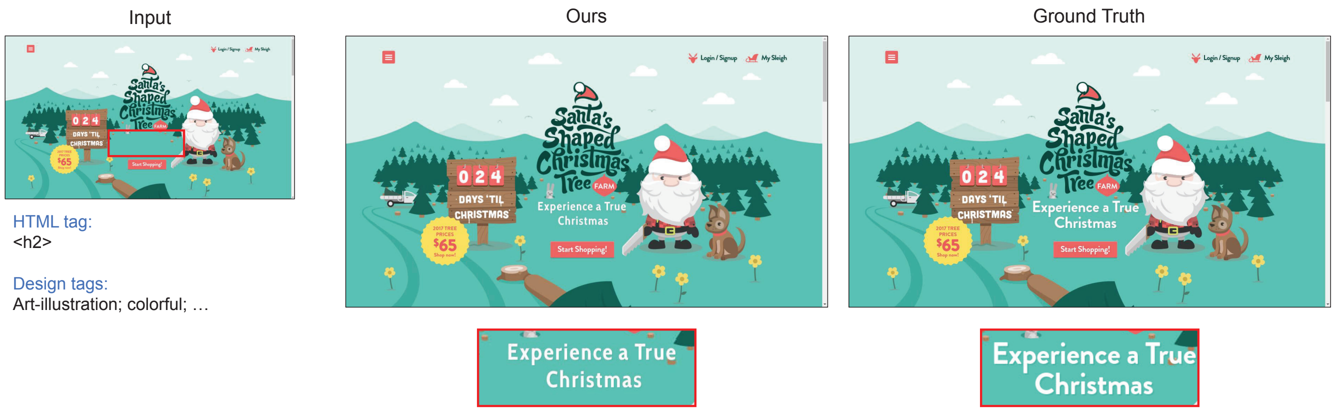

Input
Ours
Ground Truth
HTML tag:
<h2>
Design tags:
Art-illustration; colorful; …
Experience a True Christmas
Experience a True Christmas

**Figure 1:** *Given a text element (outlined in red) on a web design for suggestion, together with some semantic tags (i.e., HTML and Design tags) on the left, our method automatically selects font properties for the element (including font face, color, and size) that best fit the input web design. We show the original web design (Ground Truth) on the right for reference.*

**Abstract**

*Web designers often carefully select fonts to fit the context of a web design to make the design look aesthetically pleasing and effective in communication. However, selecting proper fonts for a web design is a tedious and time-consuming task, as each font has many properties, such as font face, color, and size, resulting in a very large search space. In this paper, we aim to model fonts in context, by studying a novel and challenging problem of predicting fonts that match a given web design. To this end, we propose a novel, multi-task deep neural network to jointly predict font face, color and size for each text element on a web design, by considering multi-scale visual features and semantic tags of the web design. To train our model, we have collected a CTXFont dataset, which consists of 1k professional web designs, with labeled font properties. Experiments show that our model outperforms the baseline methods, achieving promising qualitative and quantitative results on the font selection task. We also demonstrate the usefulness of our method in a font selection task via a user study.*

**CCS Concepts**

•***Computing methodologies*** $\rightarrow$ ***Perception;*** *Neural networks;*

## 1. Introduction

Font is fundamental to web designs. Just like how the body language and tone of a person can affect the perception of what the person says, the font properties of the text on a webpage can convey feelings and reactions that the text alone cannot. To make the designs both aesthetically pleasing and effective in information communication, web designers often select fonts carefully to fit their designs, in terms of theme, mood, readability, etc.. For example, the webpage of a hotel (Figure 2 top) uses delicate and serif font faces and a tan font color to help convey a feeling of dignity and elegance to the customers.

However, selecting proper fonts to fit the context of a web design is a challenging problem, as one needs to explore a large search

Ying Cao is the corresponding author. This work was led by Rynson Lau.

space covering many font properties, such as font face, color and size. There are some commercial tools (e.g., MyFonts and Typekit) and a previous work [OLAH14] that help users select fonts by modeling font similarity and attributes (e.g., serif and formal). Unfortunately, they only model fonts *in isolation* without considering how fonts are affected by the context where they are used. Therefore, even with those state-of-the-art font selection techniques, users still need a tedious trial-and-error process and a lot of expertise to put fonts in context (i.e., selecting matching fonts for web designs), in order to properly express the vision and mood of the whole designs.

In this paper, we take an initial step towards modeling fonts *in context*, by predicting font properties for the text in web designs. We consider three main properties of fonts defined in HTML: font face, color, and size. We propose a novel multi-task deep neural network (DNN) that, given a web design with a selected text element, automatically predicts font properties for the text element to match the web design. Our model simultaneously performs three tasks: font face, color and size predictions, based on multi-scale visual context, the semantic tags of the design and the HTML tag of the text element. For the font face prediction task, instead of casting it as a classification problem as in previous works on font face recognition [ZTW01,WYJ*15], we formulate it as a regression problem to predict a point in a lower-dimensional embedding space of font faces, for more efficient learning and better generalization capability. The embedding space is learned in an unsupervised way using an autoencoder network. To further improve the performance of font face prediction, we adopt adversarial learning [GPAM*14] and a novel data augmentation method during training.

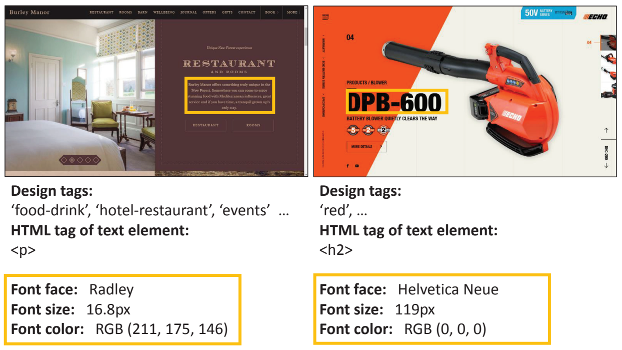
Since there are no existing datasets for our in-context font prediction problem, we have created our own CTXFont (Context Font) dataset from *awwwards.com*, a website with a large repository of professional web designs. We obtain the annotations of text elements by automatically analyzing HTML source files. We train and evaluate our model on CTXFont. Experimental results suggest that our model gives more accurate predictions than the baseline methods. We also conduct a user study to demonstrate the usability of our method in a font selection task.

In summary, the contributions of our paper are three-fold: (i) We made an initial effort to predict fonts in context, by modeling the dependency of fonts upon the designs; (ii) We propose a novel multi-task DNN for predicting font properties (i.e., font face, color, and size) for text elements in web designs; (iii) For training and evaluation of the in-context font prediction problem, we have collected the CTXFont dataset, which contains web designs with meta-data and annotations on font properties.

## 2. Related Work

To our knowledge, no prior works have directly studied automatic font selection for graphic designs. We discuss the most relevant previous works in this section.

**Font modeling.** As font is one of the core design factors, there is a growing research interest on fonts in recent years. O'Donovan et al. [OLAH14] proposed an approach to estimate attribute values (e.g., angular, artistic) and visual similarity of font face, which enable various interfaces for font face selection. Wang et al. [WYJ*15] proposed a stacked convolutional autoencoder network with domain adaption techniques for font face recognition. Some recent works put more efforts on font synthesis. Campbell and Kautz [CK14] presented the first generative model for font faces, which learns a manifold in a completely unsupervised manner. To ensure the generated fonts are realistic and precise, earlier works [CK14,PFC15] focused on geometric modeling of glyph outlines, which limits the topology of the generated font faces. In these two years, with the development of Generative Adversarial Networks (GANs) [GPAM*14], direct synthesis of new realistic font faces in pixel format [LBY*17,AFK*17] with various topologies and styles has become possible. Unfortunately, all the existing works consider fonts in isolation, without modeling their relationship to the context where they are used. In contrast, our work aims to put fonts into context, by predicting the font properties of the text that best match with the web design where the text resides.



**Design tags:**
'food-drink', 'hotel-restaurant', 'events' ...
**HTML tag of text element:**
<p>

**Font face:** Radley
**Font size:** 16.8px
**Font color:** RGB (211, 175, 146)

**Design tags:**
'red', ...
**HTML tag of text element:**
<h2>

**Font face:** Helvetica Neue
**Font size:** 119px
**Font color:** RGB (0, 0, 0)

**Figure 2:** *Web designs with annotated text elements. For each web design, we show its design tags given by designers and HTML tags for its annotated text element (outlined in orange). The text element has several font properties, including font face, color and size.*

**Assisting web design.** Given the wealth of web designs that are available online, many works adopt a data-driven method to assist the web design process. Kumar et al. [KTAK11] introduced an algorithm for web design retargetting, which can transfer design and content between web designs. By acquiring and managing a large repository of crawled web designs [KST*13], applications such as finding relevant design elements for inspiration become possible. Pang et al. [PCLC16] developed a framework to allow designers to direct user attentions on web designs by optimizing the properties of design elements. While sharing the same objective of helping the web design process, we focus on the problem of predicting fonts on web designs, which has not been explored so far.

**Multi-task learning in deep neural networks.** Multi-Task Learning (MTL) [Car98] is a general approach to learn related tasks jointly while using a shared representation. These multiple tasks usually have inherent relations and thus can benefit each other during training. MTL has been used successfully in many applications, from natural language processing [CW08], speech recognition [DHK13], to computer vision [HGDG17]. We refer readers to the survey in [Rud17] for a detailed discussion on MTL in DNNs. Our work builds upon this general learning framework to jointly predict font face, color, and size of a text element in a web design.

## 3. CTXFont Dataset

The CTXFont (Context Font) dataset is a collection of web designs with meta-data and text element annotations. To ensure the fonts of the text elements in all the web designs are chosen in a professional way, the web designs are taken from *awwwards.com*, a website competition platform where uploaded web designs are rated by the professionals in the design community. For each website, we first captured screenshots of the webpages as our web designs using a Chrome browser of $768 \times 1366$ in resolution, which is the most widely used screen resolution [w3s17]. We then obtained the properties of the text elements on each webpage by analyzing its HTML source file. Note that not all the fonts shown on a webpage can be obtained, as some of them may be in image format. In total, there are 1,065 web designs, with 4,893 text elements and 492 unique font faces. Figure 2 shows some examples in the dataset. In CTXFont, we provide the following annotations. For each web design, we have the URL, design tags given by the designer to describe the main characteristics of the design, country of the designer and the number of votes by viewers. For each text element, we have the font face, color, size, position, HTML tags, text content, margin, padding and bounding box.

## 4. Our Approach

### 4.1. Problem formulation

Given a text element in a given web design, our goal is to predict a font face, color, and size for it. As these three tasks are correlated with each other, we propose to jointly estimate the three properties with a multi-task learning framework. In particular, given a text element $e$ on a web design $d$, we want to learn a function $f$ to predict its font face $e_f$, size $e_s$ and color $e_c$: $(e_f, e_s, e_c) = f(d, e)$ as follows:

- **Font color prediction.** We do it in the RGB color space. A straightforward solution is to directly predict the color values using an $L_2$ loss [ISSI16]. However, with the multimodal nature of the color prediction problem, this loss is not appropriate, as it may suffer from the regression-to-mean problem. We thus treat this problem as a multi-class classification.
- **Font size prediction.** We formulate it as a regression problem due to its continuous nature.
- **Font face prediction.** Unlike previous work [WYJ*15] on font face recognition, which directly classifies font face names, we assume that all font faces lie in a low-dimensional embedding space, as in [CK14], and regress an input font face to a feature point in this space. Our regression-to-feature formulation has several advantages. First, unlike the classification-based model that involves many class-specific parameters (i.e., a softmax layer with many units) due to a large number of possible font faces, our lower-dimensional output space results in fewer parameters in our model, enabling more efficient learning. Second, our model can be easily extended to handle new font faces outside of the training data by projecting new font faces into the embedding space. In our work, we use an autoencoder to automatically discover the embedding space of font faces, as described in Section 4.2.

**Table 1:** *The architecture of our autoencoder. (C - convolutional layer, BN - BatchNorm layer, T - Tanh layer, and N - Normalization layer.)*

| | Type | Kernel size | Stride size | Output size |
|---|---|---|---|---|
| Encoder | C-BN-T | 11x11 | 4x4 | 64x56x56 |
| | C-BN-T | 3x3 | 4x4 | 128x14x14 |
| | C-BN-T | 3x3 | 4x4 | 256x3x3 |
| | C-BN-T | 3x3 | 2x2 | 256x1x1 |
| | C-N | 1x1 | 1x1 | 40x1x1 |
| Decoder | DC-BN-T | 7x7 | 1x1 | 256x7x7 |
| | DC-BN-T | 4x4 | 4x4 | 128x28x28 |
| | DC-BN-T | 4x4 | 4x4 | 64x112x112 |
| | DC-T | 2x2 | 2x2 | 3x224x224 |

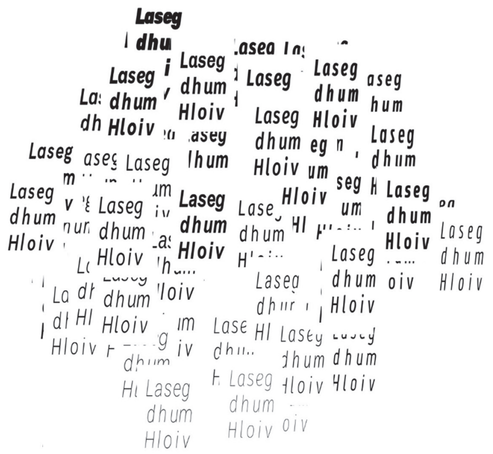


**Figure 3:** *2D visualization of our font face embedding space.*

### 4.2. Learning the Font Face Embedding Space

Our goal is to learn a latent embedding space of font faces, where similar font faces are put closer to each other while the dissimilar ones are pulled far apart. We leverage an autoencoder to automatically learn the embedding space in an unsupervised way, inspired by its recent success on feature learning [HBL*07]. To better encode visual characteristics of font faces, we represent each font face as an image. In particular, for each font face, we create a $224 \times 224$ image by rendering the same set of letters using the font face as illustrated in Figure 3.

Our autoencoder consists of an encoder that maps a font face image to a latent feature vector, and a decoder that reconstructs the font face image from the feature vector. Table 1 shows its architecture. We extract the output of the encoder as the feature vectors that represent a 40-dimensional font face embedding space. Note that we add a normalization layer at the end of the encoder to ensure that all feature vectors are unit vectors. To train the network, we use $L2$ loss and crawled 35,364 TrueType font faces from the web. In Figure 3, we show a 2D visualization of random font faces from Google Fonts, which are not present in our training dataset. Note how the distance between different font faces in the embedding space reflects the similarity of their appearances.

## 4.3. Our Network

Figure 4 illustrates the architecture of our multi-task font network. For a given text element $e$, to make our prediction aware of both visual and semantic contexts, our network takes as input the whole design image $I_d$, local image patch $I_p$ centered at $e$, the designer-given tags $t_d$ and HTML tags $t_h$ of $e$. Each input is sent to a separate encoder to produce a high-level feature representation. They are then concatenated and sent to the font decoder to output three font properties: font face ($e_f$), color ($e_c$), and size ($e_s$). To improve the quality of font face prediction, we add an additional discriminator to the font face prediction branch, as inspired by the recent success of adversarial learning approaches [IZZE17, WWXT17]. We describe each part of our model below.

### 4.3.1. Global and Local Visual Context Encoders

Selecting suitable fonts for a web design requires considering visual content and structure of the design at different scales (e.g., the color theme and layout of the design, the contents of images, and the neighborhood of the text element). To account for these factors, we introduce global and local visual context encoders to capture multi-scale visual semantics. The input of the global visual context encoder network $I_d$ is the entire input web design $d$ at a resolution of $168 \times 300$, to keep the aspect ratio of the original screenshot. Before resizing the image to the input size, we exclude the region covered by the bounding box of $e$ by filling it with the mean colors of the pixels on the boundary (10-pixel wide) of the bounding box. The input to the local visual context encoder $I_p$ is cropped from $I_d$ at the center of $e$, at a resolution of $90 \times 150$ with zero padding for the region outside of $d$. Both encoders have 4 convolutional layers with one fully-connected layer. Each convolutional layer is followed by a BatchNorm layer and a leakyReLU layer (slope 0.2).

### 4.3.2. Tag Encoders

When creating a web design, designers may label the design with some high-level semantic tags to describe the main characteristics and objectives of the design (e.g., the design objective and the mood of the design). These tags can play an important role in determining what fonts better fit the design. For example, when "business" tags are provided, the fonts to be used may exhibit a higher degree of "formal" and lower degree of "artistic". In addition, the HTML tags enclosing the text elements (e.g., "button", "h1", "a", and "p".) are also strong indicators of the font properties. For example, the use of "h1" would mean a text element is a title and thus should have a larger font size and more eye-catching font face and color. To model these, we introduce two tag encoders: design tag encoder and HTML tag encoder. Since there may exist multiple tags for a design or a text element, we encode a tag in our encoders with a binary Bag-of-Words (BoW) representation, i.e., $t_d \in \{0,1\}^M$ and $t_h \in \{0,1\}^N$, where $M$ and $N$ are the numbers of design tags and HTML tags, respectively. The $i$-th value of the BoW vector is set to 1 if the $i$-th tag is applied to a design/text element. Each of the tag encoders has three fully-connected layers with leakyReLU activation (slope 0.2), resulting in a 64D feature vector.

### 4.3.3. Font Decoder

The font decoder concatenates the feature vectors from the encoders to generate three outputs: font face, color, and size. We describe a font face $e_f$ as a point in the learned embedding space (Section 4.2). We measure a font size $e_s$ in pixel, and normalize it to [0, 1] by dividing the font height by the height of the design screenshot (i.e., 768). For font color $e_c$, we first quantize each RGB channel into 26 bins with an interval of 10, and then represent each channel as a 26D one-hot vector $(e_c^R, e_c^G, e_c^B)$. The bin containing the target color value is set to 1, while others are set to 0. With three different outputs, our font decoder first feeds the concatenated feature vector to two shared fully connected layers of 512 hidden units to obtain a shared feature vector, and is then split into three branches.

For font color, three softmax layers of 26 hidden units are added to predict three color channels. A 40D (or 1D ) fully-connected layer with tanh activation is used to regress font face (or size). All the fully-connected layers, except the last one, for each branch use the leakyReLU activation (slope 0.2). For font face, we add a normalization layer after the last layer, as stated in Section 4.2.

During prediction, for font face, we use the predicted feature vector to retrieve a list of ranked font faces from the local font face dataset, measured by the cosine distance in the font-face embedding space. To predict a RGB font color, for each color channel, we compute a sum of bin center colors, weighted by the predicted bin probabilities.

### 4.3.4. Loss Function

We use a multi-task loss $L$ to train our network for each text element. Let $e$ and $\hat{e}$ be the ground truth and prediction, respectively. Our loss is defined as:

$$L(e, I_d, I_p, t_d, t_h) = \lambda_1 L_f(\hat{e_f}, e_f) + \lambda_2 L_c(\hat{e_c}, e_c) + \lambda_3 L_s(\hat{e_s}, e_s), \quad (1)$$

where $L_f$, $L_c$, $L_s$ are losses for font face, color and size predictions, respectively. $\lambda_i$ controls the balance among the three losses.

For font face prediction, we use a $L_2$ loss to minimize the distance between the ground truth and predicted feature vectors as:

$$L_{rec}(\hat{e_f}, e_f) = ||e_f - \hat{e_f}||_2^2. \quad (2)$$

However, for a given input (i.e., a text element on a web design), there may be multiple font faces that are consistent with the input. To handle multiple modes in the output, we introduce an additional adversarial loss, which can pick a particular mode from a distribution. Our adversarial loss is based on Generative Adversarial Networks (GANs) [GPAM*14], where our font face branch is used as generator G and a font face discriminator D is introduced. Note that, in our context, instead of mapping a random noise to a sample, our generator maps the inputs to a font feature. The learning procedure is a two-player game, where $D$ aims to distinguish samples $\hat{\mathcal{X}}_f$ generated by $G$ from real samples $\mathcal{X}_f$ in our dataset, and $G$ tries to confuse $D$ by producing predictions as "real" as possible:

$$\min_G \max_D \mathbb{E}_{e_f \in \mathcal{X}_f}[log(D(e_f))] + \mathbb{E}_{\hat{e_f} \in \hat{\mathcal{X}}_f}[log(1 - D(\hat{e_f}))], \quad (3)$$

where $\hat{e_f} = G(e, I_d, I_p, t_d, t_h)$.

The total loss for font face prediction is then defined as:

$$L_f(\hat{e_f}, e_f) = w_{f_1} L_{rec}(\hat{e_f}, e_f) + w_{f_2} log(1 - D(\hat{e_f})), \quad (4)$$

and the loss of the font face discriminator is defined as:

$$-log(D(e_f)) - log(1 - D(\hat{e_f})). \quad (5)$$

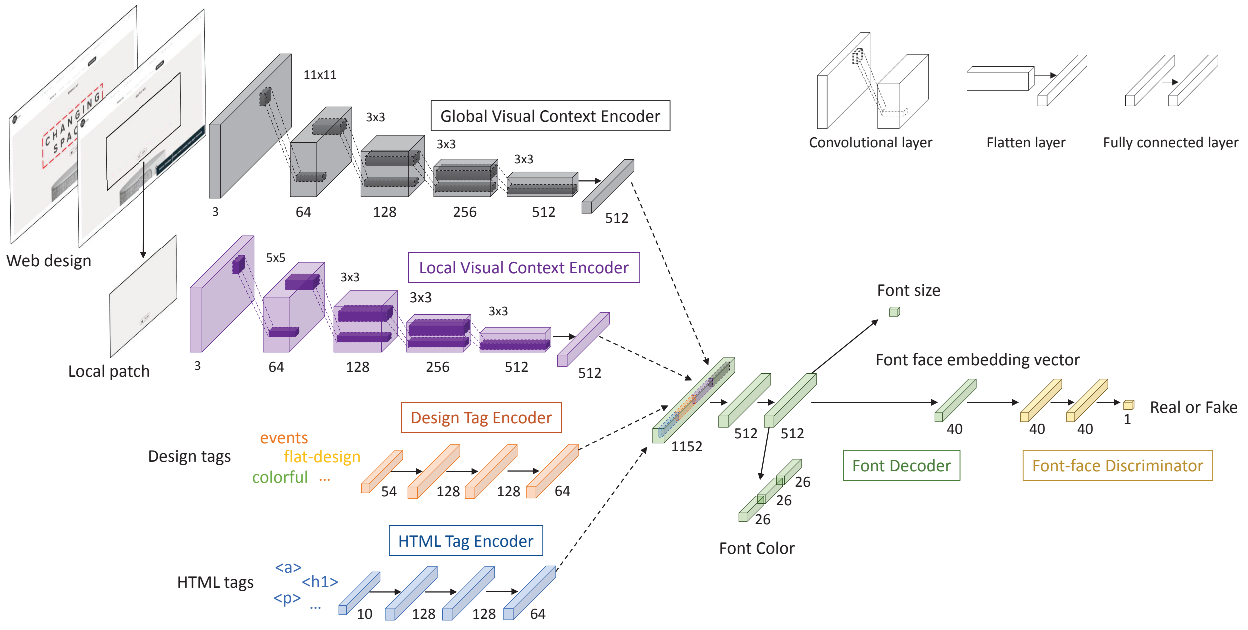


**Figure 4:** *The architecture of our network.*

For font color prediction, we use the cross entropy loss. As the font color is highly imbalanced (around 70% of the text elements in our dataset with colors in white, black, or different levels of gray), we adopt a re-weighting loss by assigning a weight based on color-class rarity [ZIE16].

$$L_c(\hat{e_c}, e_c) = -[w_c(e_c) \sum_{i \in (R,G,B)} \sum_{q \in Q} e^i_{c,q} log(\hat{e^i_{c,q}})]/3, \quad (6)$$

where Q is the number of color bins (i.e., 26), and $w_c$ is the rebalanced weight [ZIE16].

For font size prediction, we use the $L_1$ loss as:

$$L_s(\hat{e_s}, e_s) = ||e_s - \hat{e_s}||_1. \quad (7)$$

## 5. Implementation Details

To extract design tags used in our model, we searched over our dataset and only kept those used in at least 30 different web designs. We then manually removed some tags related to the design technique, such as "java" and "CSS". There are $M = 54$ design tags left at the end. For the HTML tags of text elements, we use the $N = 10$ most frequently used HTML tags: "a", "button", "h1", "h2", "h3", "h4", "h5", 'h6", "li", "p".

To train our network, we randomly split the CTXFont dataset into a training set with 915 web designs (4,268 text elements) and a testing set with 150 web designs (625 text elements). We determine the hyperparameters through 5-fold cross validation on the training set. We use the Adam optimizer [KB14] for optimization with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We use a learning rate of $1e-3$. We alternately minimize the loss functions defined in Equation 1 and 5 to update the parameters of our network, as in [GPAM*14]. We use a dropout rate of 0.2 for all layers, except the input and the output layers. We train the network using a batch size of 64 for 24,000 iterations. We set $\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 1, w_{f_1} = 1$, and $w_{f_2} = 1$ so as to balance the scale of losses for different tasks. This approach is commonly used in multi-task learning. Our current setting is found to give the best performance. An improper setting of these parameters may cause the model to be biased towards certain tasks and thus decrease the overall performance.

**Table 2:** *Accuracy and F1 score of various methods on font color prediction. (The best scores are in bold. L = local visual context encoder, G = global visual context encoder, D = design tag encoder, H = HTML tag encoder)*

| Method | Accuracy (%) | F1 score (%) |
|---|---|---|
| Random | 0.02 | 0.00 |
| Retrieval | 22.88 | 23.85 |
| L | 40.64 | 29.80 |
| L+G | 39.36 | **32.85** |
| L+H | **43.84** | 31.17 |
| L+D | 42.08 | 30.17 |
| L+D+H | 38.40 | 30.42 |
| L+G+D | 38.40 | 31.31 |
| L+G+H | 35.68 | 31.54 |
| Single | 42.72 | 31.95 |
| All (no adv) | 37.12 | 31.11 |
| All (no aug) | 30.88 | 29.29 |
| All | 39.68 | 31.72 |

**Data augmentation.** During training, we adopt a data augmentation on font faces, in order to increase the diversity of font faces

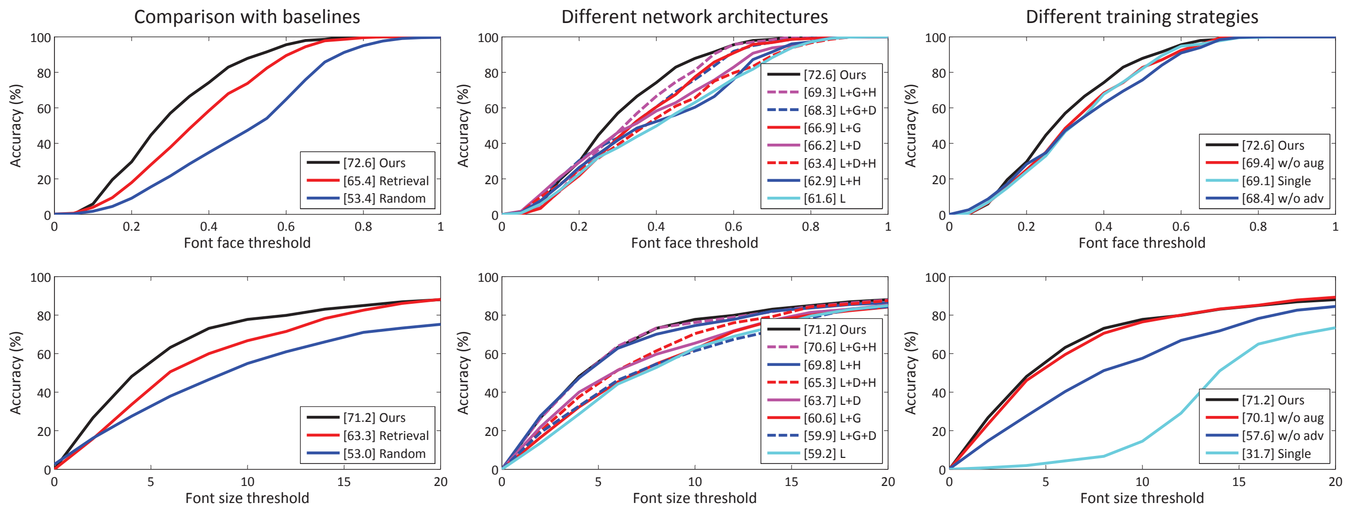


**Figure 5:** *Accuracy versus distance threshold for font face prediction (the first row) and font size prediction (the second row). We use cosine distance for font face and $l_1$ distance for font size. In the legend, we report the average accuracies. Left column: we compare our method (Ours) with two baselines (Random, Retrieval). Middle column: we compare different architectures of our network (L = local visual context encoder, G = global visual context encoder, D = design tag encoder, H = HTML tag encoder). Right column: we compare our model trained with the proposed learning strategy (Ours), our model trained without adversarial loss (w/o adv), our model without data augmentation (w/o aug), and single-task models (Single).*

that the model sees. For each font, we further find 5 nearest neighbors of its font face from the font face dataset (the font face space in Section 4.2) as its target font faces. The affinity between font faces are measured by cosine distance in the font face embedding space.

## 6. Results and Evaluations

In this section, we first show the qualitative and quantitative results of our method, as compared with several baselines. We then validate various design choices for our network architecture and learning strategies. Finally, we evaluate the effectiveness of our method in the font selection tasks via a user study.

### 6.1. Comparison with Baseline Methods

#### 6.1.1. Baselines

We compare our method with two baselines:

**Random.** The random baseline is used to show whether font choices on web designs are arbitrary or not. For font face, we randomly select a font face from the training set. For font color, we randomly select a RGB color from our quantized color space. For font size, we randomly select an integer between 0 and the height of the selected text element's bounding box.

**Retrieval.** We use a retrieval-based method as baseline, to mimic designers' commonly used strategy of using similar example designs for inspiration. For both font face and color predictions, given a text element, we take the average font face and color of its $k$ nearest neighbors (NNs) in the training set as the predicted values. The NNs are returned by measuring the cosine distance between the visual context features of the text elements, which are obtained by concatenating a global VGG feature and a local VGG feature. The VGG features are computed in the same way as in our visual context encoders, but are taken from the penultimate layer of a 16-layer VGG [SZ14] pre-trained on ImageNet [DDS*09]. We resize the inputs to the VGG to $224 \times 224$. We have empirically found that $k_{face} = 3$ and $k_{color} = 1$ obtain good performance on these two tasks, and use them in our experiments. Since the HTML tags largely determine the setting of the font size, for the font size of each query text element, we take the average font size of the text elements in the training set that have the same HTML tag vectors as the query one to be the prediction.

#### 6.1.2. Evaluation Metrics

**Metrics for font face and size.** For these two tasks, we compute the accuracy versus distance threshold. In particular, for each prediction, we compute the distance between the predicted value and the ground truth. The predicted value is regarded as correct if the distance is smaller than a threshold. For font face prediction, we use the cosine distance with a threshold ranging from 0 to 1. For font size prediction, we use $L_1$ distance with a threshold ranging from 0 to 20 (the average difference of font sizes in our dataset), and convert the normalized outputs back to the pixel-wise values for evaluation. To summarize the performance of a method, for font face, we report the average accuracy of thresholds ranging from 0.05 to 1, with an interval of 0.05. For font size, we report the average accuracy of thresholds ranging from 2 to 20, with an interval of 2.

**Metrics for font color.** The color prediction task as a classification problem is evaluated using accuracy and F1 score. For each of the three color channels in our prediction, we take the color bin with the highest probability. Accuracy is the fraction of predicted colors that match with the ground-truth color on all three R,G,B channels. To calculate the F1 score, we first calculate the precision

**Table 3:** *Results of pairwise comparisons in our user study. We show the raw percentages that our results are preferred over those by the baseline methods (Random and Retrieval), human manually (Manual) and ground truth (Ground Truth). As we have preference votes from multiple participants for each comparison, we also show the percentages of comparisons where our results have more or the same votes (Tie), compared with the others. All preference are statistically significant, according to a chi-squared test with $p < 0.05$.*

| | Raw (Ours) | Majority (Ours / Tie) |
|---|---|---|
| Random | 85.5% | 96.3% / 3.7% |
| Retrieval | 66.3% | 70.4% / 3.7% |
| Manual | 55.8% | 54.9% / 9.9% |
| Ground Truth | 35.5% | 22.2% / 11.1% |

$\mathcal{P}$ and recall $\mathcal{R}$ for each RGB bin triplet $(q_R, q_G, q_B)$:

$$\mathcal{P}_{q_R,q_G,q_B} = \frac{\mathcal{N}^c_{q_R,q_G,q_B}}{\mathcal{N}^p_{q_R,q_G,q_B}}, \mathcal{R}_{q_R,q_G,q_B} = \frac{\mathcal{N}^c_{q_R,q_G,q_B}}{\mathcal{N}^g_{q_R,q_G,q_B}}, \tag{8}$$

where $q_R \in Q_R, q_G \in Q_G, q_B \in Q_B$, $\mathcal{N}^c$ is the number of correct predictions for triplet $(q_R, q_G, q_B)$, $\mathcal{N}^p$ is the number of total predictions for triplet $(q_R, q_G, q_B)$, and $\mathcal{N}^g$ is the number of ground-truth triplets $(q_R, q_G, q_B)$. Simply aggregating $\mathcal{P}$ and $\mathcal{R}$ with the same weight for each triplet $(q_R, q_G, q_B)$ is inappropriate, which will make the F1 score dominated by frequent triplets. Thus, we use a weighted F1 score:

$$F1 = \sum_{q_R,q_G,q_B} \frac{\mathcal{N}^g_{q_R,q_G,q_B}}{\sum \mathcal{N}^g_{q_R,q_G,q_B}} \frac{2\mathcal{P}_{q_R,q_G,q_B}\mathcal{R}_{q_R,q_G,q_B}}{(\mathcal{P}_{q_R,q_G,q_B} + \mathcal{R}_{q_R,q_G,q_B})}. \tag{9}$$

### 6.1.3. Results

Figure 5 and Table 2 show the quantitative results. Our method outperforms the baseline methods by a large margin on all three tasks. Figure 6 shows some qualitative results of our method, compared with those from the baselines and the ground truth. In comparison with the baseline methods, our method can predict font properties that are more visually aesthetic (i.e., more compatible font colors and faces) and functionally valid (i.e., better readability and visibility). Our predicted font properties look rather similar to those in the ground truth most of the time. Even though our method does not suggest the same fonts as the ground truth since there may be multiple compatible fonts with a single input design, our suggested fonts still look plausible and favorably fit the input designs. For example, as shown in the example in the middle column of Figure 6, our method suggests a green color for the text element in the navigation bar, which is visually consistent with the background color of the logo and button.

## 6.2. Ablation Study

### 6.2.1. Evaluation of the Network Design

To investigate how the different encoders in our network affect the prediction performance, we use the network with only the local visual context encoder as our basic model, and add other encoders to observe the performance change. We have tested all possible combinations. In each case, the model is trained from scratch with the revised architecture. For fair comparison, the training setting is the same as our full model. Results are shown in Figure 5 and Table 2. Both the global visual context encoder and design tag encoder can help improve the performance of font face prediction by a large margin. One possible reason is that some global visual and semantic characteristics, such as the mood and theme, are indispensable for font face selection. For font size, unsurprisingly, the HTML tag encoder is the key to high performance. However, for font color, adding more encoders can only achieve marginal performance gain, indicating that the local encoder may have sufficient features for font color prediction. We have also tested other possible influential inputs to our work, i.e., semantics and number of words in a text element, but do not observe any performance improvement.

### 6.2.2. Evaluation of Learning Strategies

We finally test the effect of different learning strategies. The results are shown in Figure 5 and Table 2.

**The role of adversarial learning.** We test the importance of the adversarial loss $L_{adv}$ by removing it from our loss function (**w/o adv**). We can observe that, without the adversarial loss, a significant performance drop occurs for font face prediction and the performances for font size and color predictions are also decreased accordingly. This implies the importance of using adversarial loss in our network.

**The role of data augmentation on font face.** When training our model without data augmentation for font face (**w/o aug**), we also see a slightly worse performance for the three tasks. This suggests that our model can benefit from this simple data augmentation.

**The role of joint learning.** To verify whether joint learning can improve the performance for each task. We train our model separately for each of the three tasks with their respective losses (**Single**). From the results, we can see that our multi-task model significantly outperforms the single-task models on font face and size predictions, although with a slightly worse performance on font color prediction. This demonstrates the effectiveness of our multi-task joint learning framework.

## 6.3. User Studies

We finally test the usefulness of our method in real-world font selection tasks via a user study.

**Methodology.** In the study, we use 20 web designs with diverse appearances from the testing set. For each design, we first randomly select a text element and use its original font properties as the ground truth. We then generate font suggestions by our method (Ours), the random baseline (Random), the retrieval-based baseline (Retrieval) and the manual process (Manual). To generate manual results, we recruited 20 graduate students without any prior experiences on web design. Given a web design with a target text element, the participants were asked to select the font properties (font face, color and size) for the text element that best match the given design. We choose PowerPoint as our font selection tool, since it is more intuitive to use for average people as compared to editing the HTML source code directly. For font face selection, the participants were allowed to use either the font list in PowerPoint or the

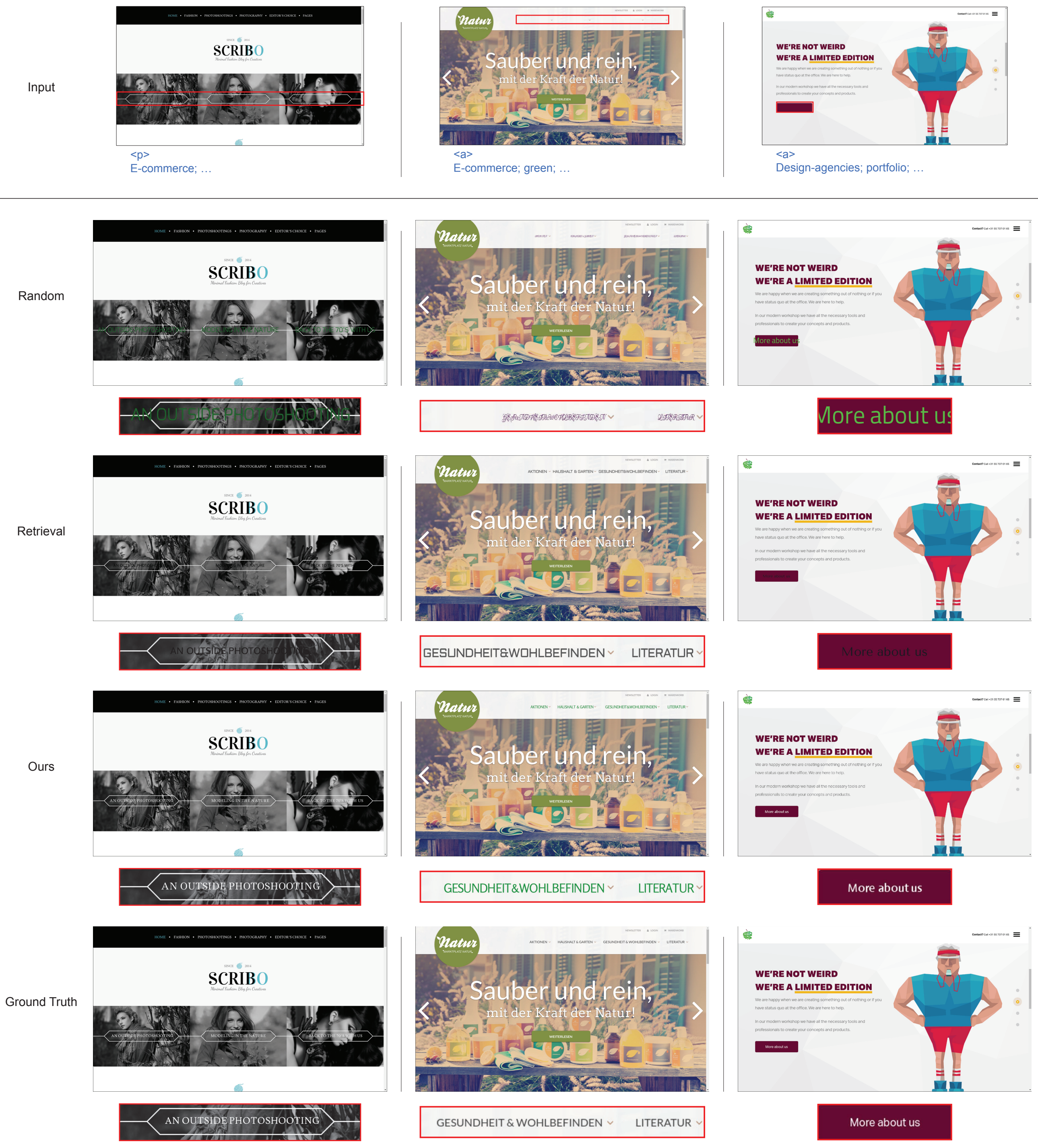


**Figure 6:** *Visual comparison of the results by our method (Ours) and the baselines (Random and Retrieval), against the ground truth.*

font selection interface proposed by [OLAH14]. To be consistent with [OLAH14], we use a total of 1,278 font faces in our method as well as the two baseline methods. Each participant was asked to complete the font selection for 5 different designs.

We use Amazon Mechanical Turk (AMT) to evaluate the font selections by different methods. In particular, we asked AMT workers to compare our results with those from other methods through pairwise comparisons in the two-alternative forced choice (2AFC)

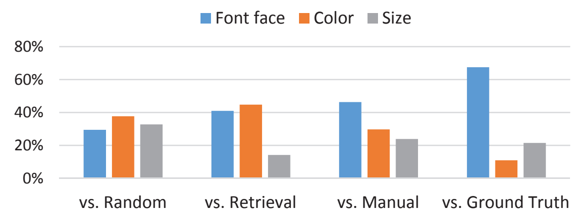


**Figure 7:** *The distributions of influential font properties when comparing our results with other results in the user study.*

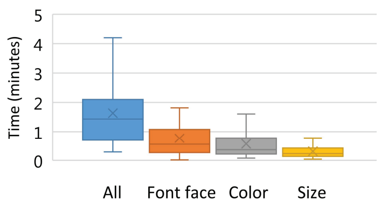


**Figure 8:** *Average time spent by participants on selecting all or individual font properties for a single text element.*

manner (i.e., they are forced to choose one of the two presented options), in order to better measure small differences [OLAH14]. For each of the 20 designs, we create 4 comparison pairs, each of which consists of two results, one by our method and the other from one of the 4 sources (Random, Retrieval, Manual and Ground Truth). For each comparison, two designs with the selected fonts (the two designs only differ in the font used for the target text element) were displayed side-by-side in random order. The participants were asked to select the font which is more suitable for the design. The design and HTML tags were also displayed beside the designs for reference. Meanwhile, the participants were required to choose the font properties (i.e., font face, color and size) that most affect their choices (termed as *influential font properties*) for each comparison. Each comparison was evaluated by 20 different workers. Each HIT consists of 20 different comparisons. For each worker, we further duplicated five randomly chosen comparisons and swap the presentation order of the two designs in each comparison to reject unreliable workers. Workers need to answer four of them consistently for their data to be accepted.

**Results.** Table 3 shows results of our user study. Our results are significantly preferred over those by the baselines and manual method, although worse than the ground truth. This implies that our method is more effective than the other alternatives in the font selection task, and can suggest font properties that better match the target web designs. To understand what font properties contribute most to the perceived compatibility of fonts with web designs, we plot in Figure 7 the distributions of influential font properties when comparing our results with other results during the study. As shown in Figure 7, the influential properties vary with different methods that our method is compared against. When comparing our results with the baseline results, color plays a more important role in determining the quality of the font selection than font face and size. When comparing our results with the manual results, font face plays a more important role. When comparing our results with the ground truth, font face determines the participants' choices for almost 70% of the time, which in turn indicates that our method is on par with the ground truth, in terms of color and size.

Figure 8 shows the average time taken by the participants to manually complete a single selection task for different font properties. In general, the participants spent more time on font face (more than 0.5 min on average) than font color and size. On average, the participants need more than 1.5 min to select all the three font properties for a single text element, whereas our method can generate font suggestions in real time. Besides, during the user study, we noticed that the participants tended to use a few font faces and colors that they had used before, even though they were given a much larger font and color space to choose from. This reduces the selection time but limits the diversity of the results. In contrast, our method can generate diverse suggestions. Figure 9 and the supplemental show the results by our method, the baselines and the participants.

## 7. Conclusion and Future Work

In this paper, we have studied the problem of modeling fonts conditioned on the context where they are used. To this end, we have presented a novel multi-task deep neural network, which can predict a font face, color, and size for a text element in a web design. We have conducted extensive qualitative and quantitative evaluations to show the effectiveness of our proposed model on a newly collected benchmark dataset, and run a user study to demonstrate the usability of our method in a font selection task.

This work is just an initial step towards addressing the problem of context-aware font prediction in web designs. We believe that there are still many exciting problems worth exploring in the future. First, we have only considered three common and important font properties in this work. We would like to explore other typography-related properties (e.g., space between words or text rows), which may also be set under the context of a design. Second, we have only considered the prediction of existing font faces. It would be interesting to investigate how to generate new font faces for web designs. This will allow us to customize different font faces to different graphic designs, instead of using the same set of pre-defined font faces for all the designs. Third, although we have only focused on web designs in this paper, our framework is generic and can be easily adapted to deal with other types of graphic designs (e.g., posters or magazine covers). We show some rudimentary results of directly applying our model (without any retraining or finetuning) to some posters. While our results are somewhat different from the ground truth, they still look plausible and visually consistent with the input posters. We envision that better results can be obtained by re-training our model on a poster dataset and/or modifying our model to accommodate the characteristics of posters, which is an interesting future work.

**Acknowledgement.** We thank the anonymous reviewers for their insightful comments. We also thank NVIDIA for donation of a Titan X Pascal GPU card.

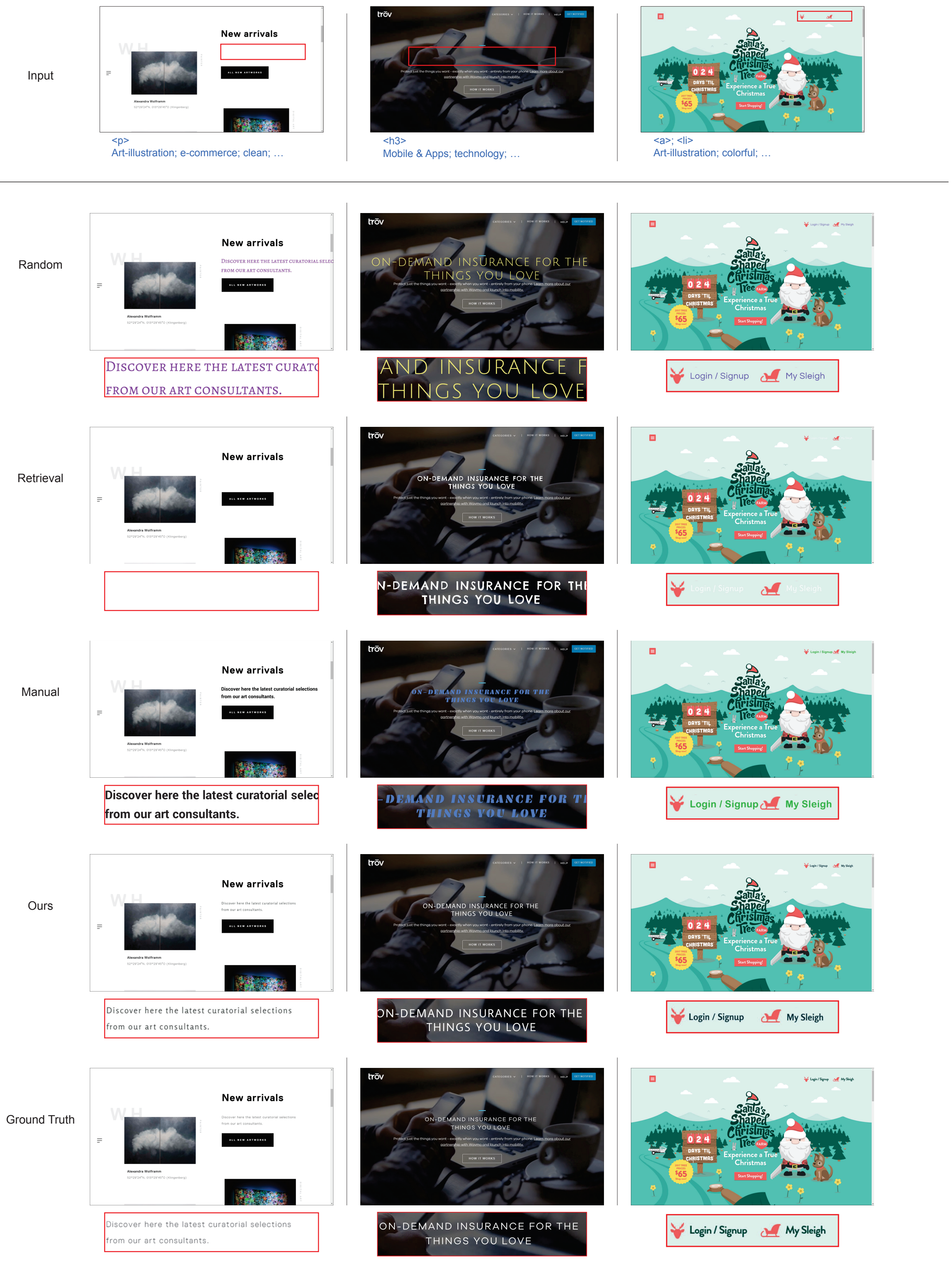

Input
<p>
Art-illustration; e-commerce; clean; …
<h3>
Mobile & Apps; technology; …
<a>; <li>
Art-illustration; colorful; …
Random
Retrieval
Manual
Ours
Ground Truth
Discover here the latest curatorial selections from our art consultants.
ON-DEMAND INSURANCE FOR THE THINGS YOU LOVE
Login / Signup
My Sleigh

**Figure 9:** *Example results used in the user study. Top to bottom: input, baseline (Random), baseline (Retrieval), manual method (Manual), our method (Ours), and the ground truth. Readers are suggested to zoom-in for a better viewing of the results.*

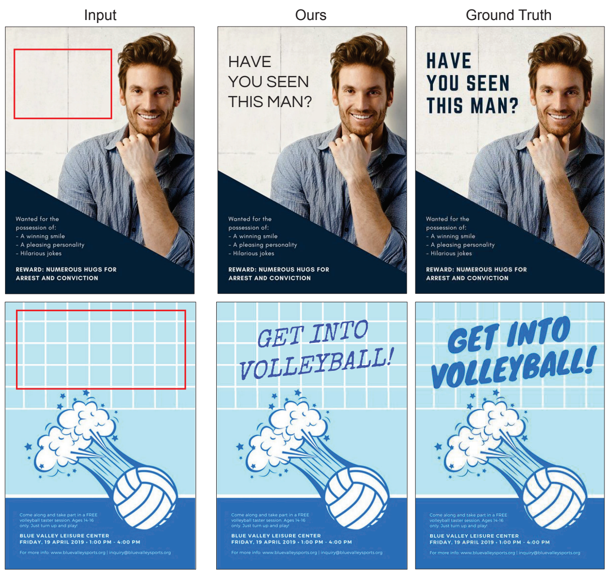


**Figure 10:** *Font prediction results of applying our model to two posters. In each case, we set the HTML tag and design tag to be "h1" and empty, respectively.*

## References

[AFK*17] Azadi S., Fisher M., Kim V., Wang Z., Shechtman E., Darrell T.: Multi-content gan for few-shot font style transfer. *arXiv:1712.00516* (2017). 2

[Car98] Caruana R.: Multitask learning. In *Learning to Learn*. Springer, 1998, pp. 95–133. 2

[CK14] Campbell N. D., Kautz J.: Learning a manifold of fonts. *ACM TOG 33*, 4 (2014), 91. 2, 3

[CW08] Collobert R., Weston J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. ACM ICML* (2008), pp. 160–167. 2

[DDS*09] Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L.: Imagenet: A large-scale hierarchical image database. In *Proc. IEEE CVPR* (2009), pp. 248–255. 6

[DHK13] Deng L., Hinton G., Kingsbury B.: New types of deep neural network learning for speech recognition and related applications: An overview. In *Proc. IEEE ICASSP* (2013), pp. 8599–8603. 2

[GPAM*14] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.: Generative adversarial nets. In *Proc. NIPS* (2014), pp. 2672–2680. 2, 4, 5

[HBL*07] Huang F. J., Boureau Y.-L., LeCun Y., et al.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. IEEE CVPR* (2007), pp. 1–8. 3

[HGDG17] He K., Gkioxari G., Dollár P., Girshick R.: Mask r-cnn. *arXiv preprint:1703.06870* (2017). 2

[ISSI16] Iizuka S., Simo-Serra E., Ishikawa H.: Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM TOG 35*, 4 (2016), 110. 3

[IZZE17] Isola P., Zhu J.-Y., Zhou T., Efros A. A.: Image-to-image translation with conditional adversarial networks. In *Proc. IEEE CVPR* (2017), pp. 5967–5976. 4

[KB14] Kingma D., Ba J.: Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014). 5

[KST*13] Kumar R., Satyanarayan A., Torres C., Lim M., Ahmad S., Klemmer S. R., Talton J.: Webzeitgeist: design mining the web. In *Proc. ACM SIGCHI* (2013), pp. 3083–3092. 2

[KTAK11] Kumar R., Talton J., Ahmad S., Klemmer S.: Bricolage: example-based retargeting for web design. In *Proc. ACM SIGCHI* (2011), pp. 2197–2206. 2

[LBY*17] Lyu P., Bai X., Yao C., Zhu Z., Huang T., Liu W.: Auto-encoder guided gan for chinese calligraphy synthesis. *arXiv:1706.08789* (2017). 2

[OLAH14] O'Donovan P., Lībeks J., Agarwala A., Hertzmann A.: Exploratory font selection using crowdsourced attributes. *ACM TOG 33*, 4 (2014), 92. 2, 8, 9

[PCLC16] Pang X., Cao Y., Lau R. W., Chan A.: Directing user attention via visual flow on web designs. *ACM TOG 35*, 6 (2016), 240. 2

[PFC15] Phan H. Q., Fu H., Chan A.: Flexyfont: Learning transferring rules for flexible typeface synthesis. In *CGF* (2015), vol. 34, pp. 245–256. 2

[Rud17] Ruder S.: An overview of multi-task learning in deep neural networks. *arXiv:1706.05098* (2017). 2

[SZ14] Simonyan K., Zisserman A.: Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556* (2014). 6

[w3s17] w3schools.com: Browser display statistics. `https://www.w3schools.com/browsers/browsers_display.asp`, Jan 2017. 3

[WWXT17] Wang C., Wang C., Xu C., Tao D.: Tag disentangled generative adversarial network for object image re-rendering. In *Proc. IJCAI* (2017), pp. 2901–2907. 4

[WYJ*15] Wang Z., Yang J., Jin H., Shechtman E., Agarwala A., Brandt J., Huang T.: Deepfont: Identify your font from an image. In *Proc. ACM MM* (2015), pp. 451–459. 2, 3

[ZIE16] Zhang R., Isola P., Efros A.: Colorful image colorization. In *Proc. ECCV* (2016), Springer, pp. 649–666. 5

[ZTW01] Zhu Y., Tan T., Wang Y.: Font recognition based on global texture analysis. *IEEE TPAMI 23*, 10 (2001), 1192–1200. 2