# Detecting and Segmenting Adversarial Graphics Patterns from Images

Xiangyu Qu, Stanely H. Chan
School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana USA
{qu27, stanchan}@purdue.edu

## Abstract

*Adversarial attacks pose a substantial threat to computer vision system security, but the social media industry constantly faces another form of "adversarial attack" in which the hackers attempt to upload inappropriate images and fool the automated screening systems by adding artificial graphics patterns. In this paper, we formulate the defense against such attacks as an artificial graphics pattern segmentation problem. We evaluate the efficacy of several segmentation algorithms and, based on observation of their performance, propose a new method tailored to this specific problem. Extensive experiments show that the proposed method outperforms the baselines and has a promising generalization capability, which is the most crucial aspect in segmenting artificial graphics patterns.*

## 1. Introduction

Social media enables people to share their lives with the world through images and videos. However, some people use these platforms for more sinister purposes: spreading illegal content, selling drugs, and making illicit advertisements [35, 36, 55] as shown in Figure 1. Most of today's social media platforms use automated screening systems for filtering these inappropriate contents by using a variety of computer vision tools [22, 50]. However, these automated screening systems are not robust as the offenders have managed to breach them by altering the policy-violating content. Their approaches are sometimes simple, e.g, drawing patterns or adding texts. Although not as powerful as adversarial attacks in terms of minimizing the visual difference between the original and the altered images, adding artificial graphics patterns is easily doable by a layperson and these simple artificial graphics patterns can be effective. As a simple toy example, we conducted an experiments on ImageNet with pre-trained ResNet-101 and Mobilenet-v3large provided by PyTorch. The accuracy of each classifier drops from 76.9% to 54.6% and 72.1% to 41.8% respectively, although the randomly added graphics patterns occupy less than $4\%$ of the image area.


(a) cigar Ad (b) illicit medicine Ad

Figure 1. [Top row] Example unlawful/policy-violating advertisement images with adversarially added graphics patterns for bypassing screening systems[2]. [Bottom row] The masks generated by the proposed method. We use one network for all these types of perturbations. Detailed results can be found in the experiment section.

At the first glance, adversarial training would seem be to an effective solution. However, considering the set of all possible graphics patterns, deploying adversarial training at a large-scale is challenging. Inspired by [16], in which the authors show that a classifier can be trained and performs equally well during testing even if large connected regions are masked, we argue that a more practical solution is to disentangle the task of identifying added graphics patterns and the task of screening. This way, all screening classifiers can be trained on fixed masked images and one only needs to re-train or fine-tune the graphics pattern detector whenever a novel adversarial pattern appears. In this paper, we focus on detecting and segmenting adversarially added *artificial graphics patterns*, as illustrated in Figure 1. Given the context of the problem, we limit the scope of artificial graphics patterns to *any* relatively simple patterns that are drawn using computer graphics software (e.g. Adobe Photoshop, Microsoft Paint), such as stickers,

[2]Due to privacy of the real data, the examples shown here are created by the authors using Photoshop for illustration purposes only.

text, lines, shapes, etc. As we show in our experiments, by leveraging multi-scale modeling and mining hard samples (i.e. more complicated patterns with less foreground-background visual difference), models can generalize well to out-of-distribution samples even if the training set is tiny and the variety of patterns in the training set is limited.

Our contribution is summarized as following:

- We take a novel perspective in handling adversarial attacks by artificial graphics patterns that are prevalent in industry and establish baseline results for various SOTA segmentation systems.
- We propose a data synthesis scheme to systematically simulate training images. Our synthesis method adjusts the adversarial patterns so that the visual features are similar to the local neighborhood and enables hard sample mining.
- We demonstrate that utilization of multi-scale information and lower level features is crucial for generalizing to out-of-distribution patterns. Based on our observations, we propose a novel cascade network with custom training policy and show that it achieves better generalization to unseen patterns and more consistent performance across pattern size.

## 2. Related Work

The problem studied in this paper is about segmenting patterns created by human using computer graphics software. To our knowledge, previous literature does not have a solution to this specific problem. Subjects such as image forensics [2, 4, 20, 39, 42, 44] are traditionally framed under unsupervised learning or single-class classification settings. Splice detection methods such as [12] focus on detecting splicing between two natural images and therefore have a different data domain. Watermark and logo detection [13, 46] are related tasks but SOTA solutions that generate segmentation masks are similar to those in saliency detection and semantic segmentation.

### 2.1. Semantic Segmentation and Saliency Detection

Since the goal of this paper is about segmentation, the most natural comparisons would be semantic segmentation and saliency detection, summarized as follows.

**Semantic segmentation**. Semantic segmentation aims at giving each pixel a categorical label. Traditional approaches focus on using hierarchical graph models of superpixels [3, 25, 27]. Convolutional networks outperform these methods [45]. However, early works usually have high model capacity, only produce low-resolution output due to large receptive field, and do not use multi-scale information well. Improvements have been proposed by, for example, Farabet et al. [15], Chen et al. [5], Zhao et al. [58], and Ronneberger et al. [41]. Recent works leverage a combination of these improved techniques [6–8, 24, 31, 32, 37, 54, 56].

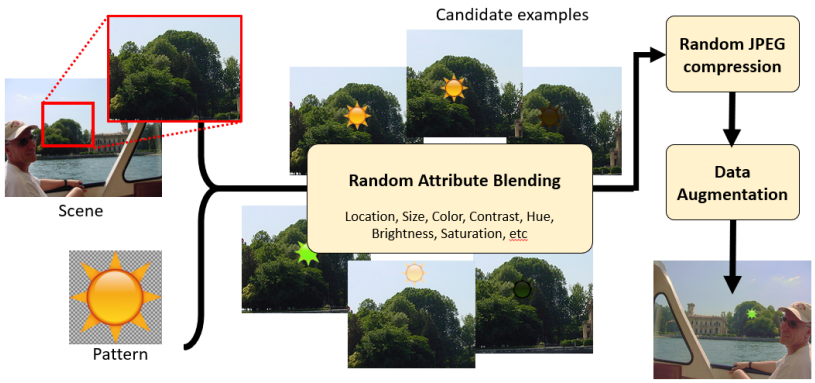
While semantic segmentation can be considered as a parent task of our problem, the diversity of the data domain makes our problem different. In general, data from the same class in a traditional semantic segmentation task (e.g. scene parsing and face parsing) share a common global context (e.g. shape) and have a relatively fixed feature-to-scale correspondence. Our problem does not share these characteristics, and therefore many generic semantic segmentation methods fail. In the experiment section, we will use DeepLabv3 (DLV3) [6] and UNet [41] as baselines for comparison with our proposed method.

**Saliency detection**. The goal of saliency detection is to locate the most informative region in a natural image. Arguably, one can translate the problem to ours by declaring that an artificial pattern region is important and natural regions are not. However, when framed in this context, to our knowledge, there is no off-the-shelf solution presented, although many lessons in saliency detection can be learned.

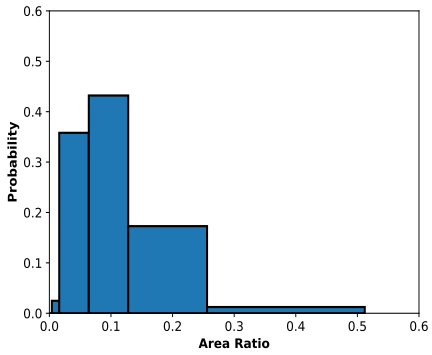
Existing saliency detection methods range from using hand-crafted features and graphical models [10, 14, 21] to the recent deep learning approaches [17, 19, 26, 28, 33, 34, 40, 51, 53, 57, 59, 60]. For example, modeling local and global context by a two-stream network [59], exploiting both pixel-level prediction and region-level information [28], making a coarse prediction and then gradually refine the prediction with recurrent network [33]. Recent works focus on combining hand-crafted features with deep network [26], better multi-scale and multi-level joint modeling [19, 28, 34, 57, 60], or more accurate segmentation along boundaries [40, 51, 53]. In the experiment section, we will compare with the BASNet [40] and PFANet [60], which are state-of-the-art saliency detection methods.

### 2.2. Cascade Models

The backbone of our proposed solution is a cascade network. This idea is inspired by the Viola-Jones detection framework [49]. In Viola-Jones, cascading is used for computational efficiency. In our case, we use cascading to induce implicit attention and save the model capacity to resolve regions that cannot be resolved at shallower layers. Previous CNN based works have explored similar ideas in other areas of computer vision such as face landmark prediction [47], pose estimation [48], face detection [29], classification [38], and perceptual edge detection [18]. For image segmentation, there are attempts for coarse-to-fine class segmentation [11], multi-stream fusion [61], and class-agnostic segmentation refinement [9]. Finally, our proposed method shares some common ideas with a work earlier by Li et al. [30]. The main differences are twofold: (1) our proposed architecture operates on multi-scale input with multiple entry points and is different from theirs. (2) We train our




(a) Proposed data synthesis pipeline




(b) Empirical distribution of sizes

Figure 2. (a) The proposed data synthesis pipeline: We first draw random images and artificial patterns from their respective datasets. For each random location we choose, we adjust the pattern until the attributes match with those in the local neighborhood. JPEG compression is added to improve robustness against compression artifacts. (b) Empirical distributions of the sizes of the artificial patterns in a dataset containing 100 images downloaded from popular social media websites. Following this analysis, we configure the size of our synthesized pattern to approximately 0.1%-25% of the image.

multi-scale cascade network bottom-up at multiple stages so that coarser-scale network guide the training of the finer-scale network, rather than training all sub-networks jointly in two stages.

### 2.3. Multi-scale Methods

The proposed method uses a multi-scale framework to model and extract features across scale variations. While this idea has been used in many prior work, e.g., [7, 15, 17, 24, 32, 37, 59], the specific usage of the scale information is quite different when we compare the two perspectives in Table 1: 1) At which stage is multi-scale modeling constructed: explicitly extract features from image at different scales (explicit) or treat feature maps from different levels of a network as multi-scale (implicit). 2) How information from different scales is used to generate final predictions: aggregate features or aggregate predictions.

| | feature aggreg. | prediction aggreg. |
|---|---|---|
| explicit | [17, 32, 37, 59] | [7, 15], ours |
| implicit | [28, 34, 58, 60] [33, 40, 54] | [19, 24, 57] |

Table 1. Taxonomy of multi-scale modeling by semantic segmentation and saliency detection methods

Most of prediction aggregation methods in the segmentation literature take the weight average of the predictions from multiple scales as the final prediction, where the weighting is implemented as a learn-able layer. In contrast, our model takes the conjunction of predictions from different scales for each pixel. This will be illustrated in the experiment section when we compare with the panoptic feature pyramid network (SFPN) by Kirillov et al. [24].

## 3. Method

We discuss the three ideas of this paper: a data synthesis pipeline, a multi-scale network, and a training scheme.

### 3.1. Data Synthesis

Since the actual process of adding artificial patterns is relatively simple to emulate, we synthesize the data during training on-the-fly.

**Procedure for synthesizing training data**. Our data generation pipeline is as illustrated in Figure 2(a). We identify four categories of artificial content: stickers (e.g. emoji), text, lines/stripes, and digital logos. We construct a set $\mathcal{A}$ of these canonical patterns, which consists of 381 samples of high-resolution patterns. The size of these added patterns consumes approximately from 0.1% to 25% of the size of the images, with a higher concentration in the bottom quantile. Figure 2(b) shows an empirical distribution of 100 randomly downloaded images from the internet.

Given an input image $X \in \mathbb{R}^{H \times W \times 3}$ drawn from a dataset $\mathcal{X}$ of natural images, we randomly draw $K$ artificial patterns $A_1, ..., A_K$ from the canonical pattern set $\mathcal{A}$. We resize these $K$ patterns so that the sum of the area of all $K$ patterns reach a target proportion, e.g., 10% of the image size. For each pattern $A_k$ ($k = 1, \ldots, K$), we randomly put it at a location in the image $X$. Afterward, we perform JPEG compression with a random quality factor between 70 and 100 to the resulting image. This step is to ensure that any compression artifacts are learned.

In order to make sure that the manipulated image does not have obvious features for saliency detection (so that the training data is not too easy), each pattern $A_k$ is pre-processed before added to the image. This observation is

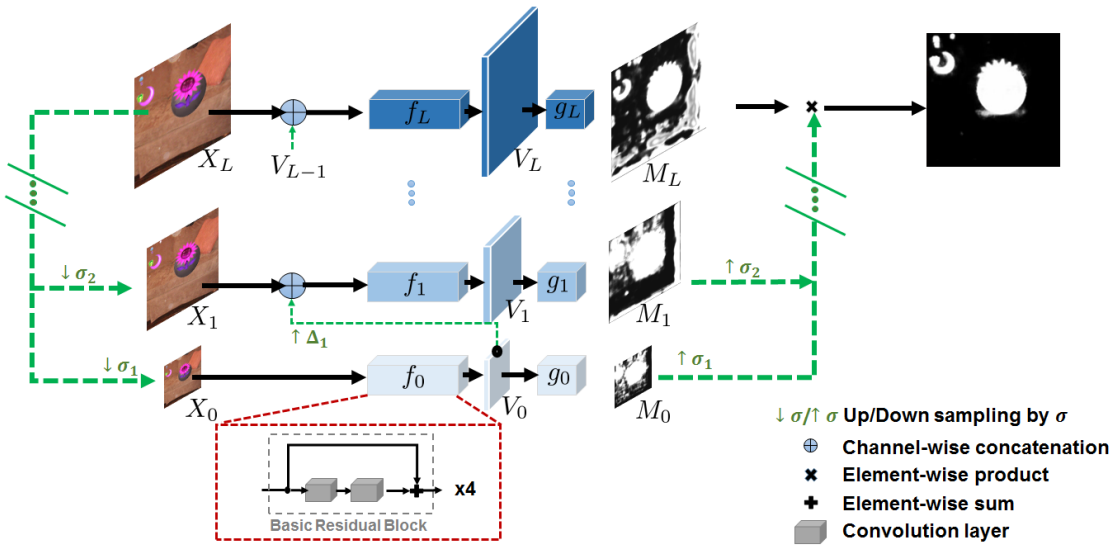



Figure 3. The proposed multi-scale cascade network. In our model, we use a cascade of $L$ levels of sub-networks. The goal of each sub-network is to segment the graphics patterns in their respective scales. Information at the current level is propagated to the next level, so that we can accumulate the prediction results. Under such a design, the final mask is generated by taking the intersection of the patterns of all the scales. Note how prediction from each sub-network complement each other and allows false positive predictions in other sub-networks

confirmed by Jiang et al. [14, 21] who showed that brightness, local and global contrast, hue, color saturation are important cues for saliency detection. Our pre-processing involves comparing these attributes (brightness, local and global contrast, hue, color saturation) of artificial patterns with the neighbors at the target location of $X$. We randomly adjust the attributes of the added patterns so they match or mis-match with those of the neighbors.

### 3.2. Multi-scale Cascade Network

**Proposed architecture**. We propose a multi-scale cascade network to encourage generalization to unseen patterns and explicitly handle the huge size variation across the patterns. The overall network is shown in Figure 3. The network consists of multiple sub-networks that operate on inputs at different scales. Each sub-network consists of two elements:

- **Backbone** for extracting features: Each backbone sub-network is composed of 1 convolution layer with $C$ 3-by-3 kernels that transforms the input feature to a $C$-channel feature, and 4 basic blocks from the Resnet.
- **Regression head** for generating segmentation mask: The regression head of each sub-network is composed of 1 convolution layer with 3-by-3 kernels that reduce the number of channels by half, 1 convolution layer with a 1-by-1 kernel that collapses channel number to 1, and a sigmoid function that normalizes the score map to the range $[0, 1]$.

We denote $\ell \in \{0, 1, ..., L\}$ as the scale indices from coarse to fine. Let $f_\ell$, $g_\ell$ be the backbone and regression head at scale $\ell$, respectively. Given an input image $X \in \mathbb{R}^{H \times W \times 3}$, we low-pass filter it and down-sample it to obtain coarser scale images $X_\ell \in \mathbb{R}^{\frac{H}{\sigma_\ell} \times \frac{W}{\sigma_\ell} \times 3}$, where $\sigma_\ell$'s are the scale factors (e.g., $\sigma_\ell = \sqrt{2}$). Features extracted at the $\ell$-th layer are denoted as $V_\ell$'s.

At the coarsest scale $\ell = 0$, the sub-network at that scale extracts features and makes a prediction

$$\underbrace{M_\ell}_{\text{mask}} = \underbrace{g_\ell}_{\text{regression}} \big( \underbrace{f_\ell(X_\ell)}_{\text{feature extract}} \big), \qquad \text{for } \ell = 0 \text{ only}, \tag{1}$$

where $M_\ell \in \mathbb{R}^{\frac{H}{\sigma_\ell} \times \frac{W}{\sigma_\ell} \times 3}$ is the predicted mask by the $\ell$-th sub-network.

In our proposed network, high-level features extracted from the coarser levels are re-used at the finer scales as a global context, and the predicted mask from the coarser scale is used to filter out regions that are difficult to make a correct prediction based on local context and features at the fine scale. Therefore, at scales $\ell = 1, \ldots, L$, the sub-network leverages extracted high-level features from the previous scale as well as the image at that scale

$$M_\ell = g_\ell\Big( f_\ell\Big( \overbrace{X_\ell}^{\substack{\text{concatenation of} \\ \text{image and feature}}} \oplus \overbrace{\big[\, V_{\ell-1} \,\big]_{\uparrow \Delta_\ell}}^{\substack{\text{upsample of} \\ (\ell-1)^{\text{th}} \text{ feature}}} \Big) \Big), \tag{2}$$

where $\oplus$ denotes concatenation of along channel dimension, $[\ \cdot\ ]_{\uparrow \Delta}$ denotes the bi-linear upsampling with a factor of $\Delta$. In our problem, $\Delta_\ell = \sigma_\ell / \sigma_{\ell-1}$.

The stage-wise output mask $\widetilde{M}_\ell \in \mathbb{R}^{H\times W\times 3}$ is the intersection of the previous stage masks. It is defined as the pixel-wise multiplication from all scales below $\ell$

$$\widetilde{M}_\ell = \underbrace{\prod_{i=0}^{\ell}}_{\substack{\text{intersection}\\ \text{of all stages}}} \underbrace{\left[\ M_i\ \right]_{\uparrow\sigma_i}}_{\substack{\text{upsample of}\\ \text{previous masks}}}. \tag{3}$$

Our cascade scheme allows sub-networks to complement each other and therefore, each sub-network is encouraged to be "forgiving"; a sub-network can make false positive predictions at regions that are rejected by other networks.

### 3.3. Customized Stage-wise Training

Training of the proposed multi-scale cascade network requires a stage-wise training scheme. The training strategy is illustrated in Figure 4. We discuss the rationale and the procedure as follows.

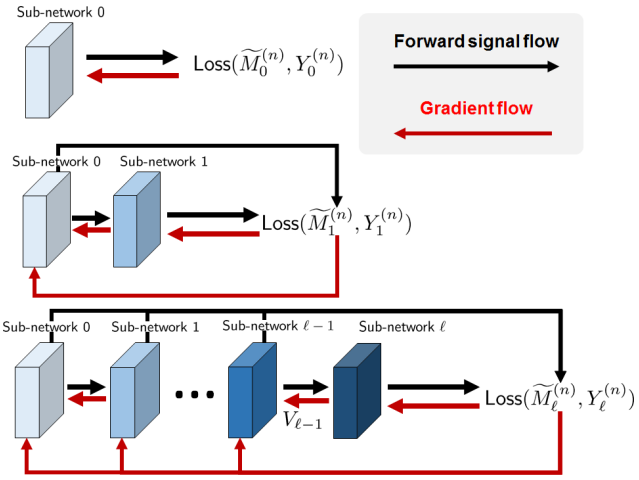



Figure 4. Multi-stage training for the proposed network. We train the sub-networks sequentially by initializing with the previous stage outputs.

**Why stage-wise training?** A stage-wise training strategy for the proposed architecture has two advantages:

(i) It allows us to explicitly specify the minimum precision ratio and the desired recall ratio for each cascading level. See our optimization below.

(ii) The learned coarser scale sub-networks can guide the finer scale networks to focus on regions where positive samples might exist, and neglect regions that are already considered as hard negative (gradient to finer scale networks is masked). This encourages fine-scale classifiers to look at different features.

If the entire network is trained jointly, the benefit of coarse-scale sub-network guidance is invalid, since the mask from a lower scale is not meaningful at that stage. We show the ablation study of multi-stage training and simultaneous joint training in supplementary material.

**Training procedure**. We denote $(\cdot)^{(n)}$ as the $n$-th training sample. At every scale $\ell$, we consider an estimate mask $\widetilde{M}_\ell$ and a ground truth binary mask $Y_\ell$. Using the cross-entropy as the training loss, we formulate the following optimization [3]:

$$\begin{aligned}
\underset{f_\ell,\, g_\ell}{\text{minimize}} \quad & -\frac{1}{N}\sum_{n=1}^{N}\Big\{Y_\ell^{(n)}\log\widetilde{M}_\ell^{(n)} \\
& \qquad + (1-Y_\ell^{(n)})\log(1-\widetilde{M}_\ell^{(n)})\Big\} \qquad (4)\\
\text{subject to} \quad & \underbrace{\frac{1}{N}\sum_{n=1}^{N}\text{Precision}\left(\widetilde{M}_\ell^{(n)}, Y_\ell^{(n)}, \tau\right)}_{\substack{\text{average precision using threshold } \tau}} \geq P_{\text{min}},\\
& \underbrace{\frac{1}{N}\sum_{n=1}^{N}\text{Recall}\left(\widetilde{M}_\ell^{(n)}, Y_\ell^{(n)}, \tau\right)}_{\substack{\text{average recall using threshold } \tau}} \geq R_{\text{min}}.
\end{aligned}$$

In this optimization problem, the functions Precision$(\cdot)$ and Recall$(\cdot)$ computes the precision-recall values of the predicted mask $\widetilde{M}_\ell$ relative to the true mask $Y_\ell$, at a given threshold $\tau$.

The parameters $P_{\text{min}}$ and $R_{\text{min}}$ are the minimum precision and recall levels we desire. A pixel is considered to be positive (artificial) in the final prediction by the entire network only if all sub-networks predict it to be positive. Therefore, the minimum recall $R_{\text{min}}$ at each stage must be high (so that we allow more false positives to go through). This is not a problem, because even if the false positive rate at each sub-network is high (e.g. 40% false positive rate or 60% precision), after cascading the final false positive rate will be low (e.g. $40\%$ false positive rate at each level will be $6.4\%$ after 3 levels of cascading). Therefore, we could have a relatively loose precision $P_{\text{min}}$ at each stage of training. In our implementation, we set $R_{\text{min}}$ and $P_{\text{min}}$ by checking final loss on validation set.

### 3.4. Data Balancing

Unlike a generic semantic segmentation problem where the sizes of the objects do not have a substantial impact on the performance [45], the same issue is significantly more important in our problem. Specifically, since the added artificial contents usually only occupy a small portion of the foreground, there is a significant data imbalance between the positive and the negative samples.

[3] For notation simplicity we omit the averaging over all pixels of an image in the constraints.

Addressing the data balancing issue is typically done by (1) drawing samples according to the desired percentage, or (2) adjusting the training loss. Our approach is based on the latter. Considering the cross-entropy loss in (4), we introduce a per-image weighing constant $\alpha_\ell^{(n)}$ so that the loss becomes

$$\underset{f_\ell,\, g_\ell}{\text{minimize}} \; -\frac{1}{N}\sum_{n=1}^{N}\Bigg\{\frac{1}{\alpha_\ell^{(n)}} Y_\ell^{(n)} \log \widetilde{M}_\ell^{(n)} + \frac{1}{(1-\alpha_\ell^{(n)})}(1-Y_\ell^{(n)})\log(1-\widetilde{M}_\ell^{(n)})\Bigg\}$$

Here, the weighing constant is defined according to the proportion of the added content relative to the image:

$$\alpha_\ell^{(n)} = \frac{1}{HW}\sum_{\text{all pixels}} Y_\ell^{(n)}, \tag{5}$$

where $H$ and $W$ are the number of rows and columns of the label mask $Y_\ell^{(n)}$.

# 4. Experiments

## 4.1. Implementation and Datasets

**Training and validation**. For the training data, we synthesize the perturbed images on-the-fly to minimize overfitting. As discussed in Section 3.1, the data synthesis procedure draws clean and natural images from a dataset. Then it renders artificial patterns and adds them to the images. The clean images we use are the PASCAL VOC 2012 training set. We randomly crop and resize them to $256 \times 256$ and send through the data synthesis pipeline. The artificial patterns we use are manually selected from the emoji dataset used by Apple's platform. We use this small collection of Apple platform emoji rather than a larger variety of artificial contents because the goal of our experiment is to test whether a model can generalize. We further break down the collected emoji images to the training set and validation set by the emoji category. Detailed training and validation set statistics are as shown in table 2.

We implemented in PyTorch. We used Adam [23] for optimizing sub-network parameters at each stage with initial learning rate 1e-3 and $\beta's$ in (0.9, 0.999).

**Testing set**. To quantitatively evaluate the generalization capability of each segmentation model, we create a fixed synthetic testing dataset using a large collection of commonly seen artificial patterns: **stickers/emoji, texts, lines/curves, and logos**. (Note that this is for testing. **We only train on a small collection of 156 Apple emoji**. Our goal is to show that our model generalizes.) Two large public emoji sets (noto-emoji by Google and twemoji by Twitter, a total of more than 6000 emoji) are used. These emoji have drastically different visual and textural appearances as compared to the small Apple emoji set. They are also widely adopted by various software/websites due to their open-source nature. For the logo patterns, we use Large Logo Dataset (LLD) by Sage et al. [43]. Since the size of these artificially added patterns has a significant impact on the segmentation quality, for each type of artificial content, we group the synthesized test images into three subsets (**small, medium, large**). In total, our test set has 12000 images (1000 for each category and size level). Detailed description and statistics of testing set is in supplementary material.

| Clean images | | Total = 5717 | |
|---|---|---|---|
| Training emoji | animals | 22 | 156 |
| | food | 36 | |
| | nature | 44 | |
| | objects | 54 | |
| Validation emoji | people | 79 | 225 |
| | face | 109 | |
| | hand signs | 37 | |

Table 2. Statistics of the emoji we use to train our model.

## 4.2. Evaluation Metrics

Following the semantic segmentation and the saliency detection literature, we evaluate the performance of the baseline methods and our proposed methods on the test set with the following metrics: (i) The **mean intersection-over-union (mIoU)**. The threshold for mIoU was chosen to maximize the mIoU value over the validation set. (ii) The **precision-recall** curve. (iii) The **mean absolute error (MAE)**. (iv) The $F_\beta$**-measure**, defined as

$$F_\beta = \frac{(1+\beta^2)\cdot \text{Precision}\cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \tag{6}$$

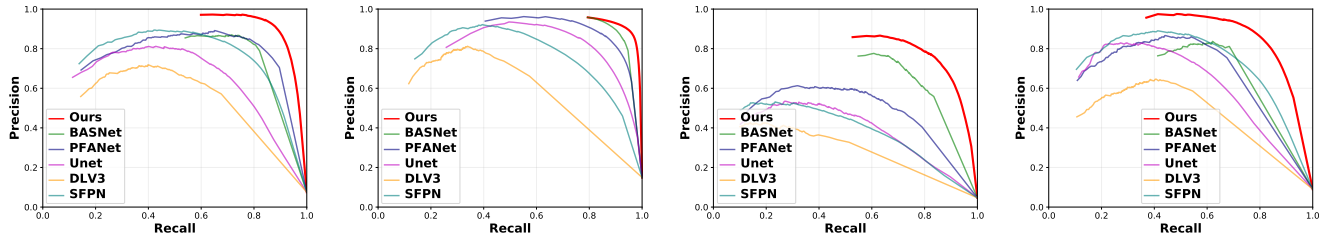
with $\beta$ set to 0.3 and 2 for emphasizing precision and recall, respectively. Saliency detection works usually report the $F_\beta$-measure value obtained by an adaptive threshold as suggested by [1]. However, since our problem scope is different and so the adaptive threshold is less relevant, we report the maximum $F_\beta$-measure obtained on each category of the test set, as suggested by [40].

## 4.3. Main Results

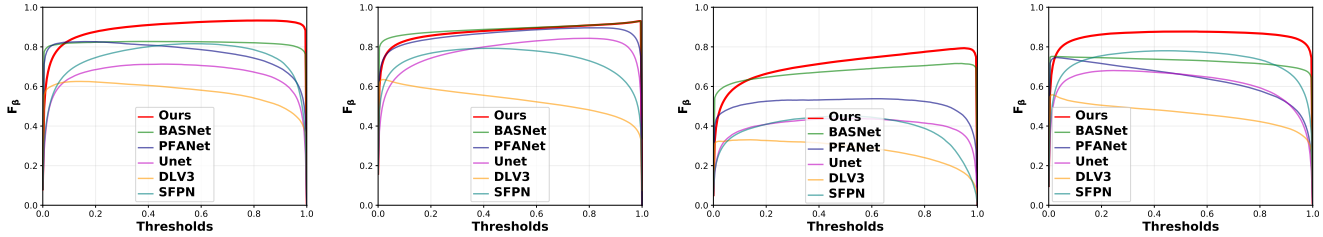
**Baseline models**. We use DeepLabV3 (DLV3) [6], Unet [41], BASNet [40], PFANet [60], and panoptic FPN [24] as the baseline models. The baseline models are trained with the same data synthesis process as our proposed model. Each baseline model is initialized with either a pre-trained model (if available) or random weights. All models are trained until (1) training loss converged, or (2) validation loss and training loss diverges, whichever comes first. To demonstrate the raw detection capability, we do not include

| | Stickers | | | | Lines | | | | Texts | | | | Logos | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | mIoU | MAE | $F_{0.3}$ | $F_2$ | mIoU | MAE | $F_{0.3}$ | $F_2$ | mIoU | MAE | $F_{0.3}$ | $F_2$ | mIoU | MAE | $F_{0.3}$ | $F_2$ |
| **Ours** | **0.818** | **0.011** | **0.933** | **0.894** | **0.846** | 0.022 | **0.930** | **0.942** | **0.645** | 0.017 | **0.793** | **0.826** | **0.673** | **0.038** | **0.878** | **0.822** |
| **Unet** | 0.512 | 0.035 | 0.713 | 0.691 | 0.690 | 0.046 | 0.843 | 0.828 | 0.321 | 0.040 | 0.440 | 0.536 | 0.398 | 0.061 | 0.680 | 0.625 |
| **DLV3** | 0.368 | 0.051 | 0.519 | 0.509 | 0.352 | 0.111 | 0.574 | 0.458 | 0.245 | 0.039 | 0.357 | 0.445 | 0.368 | 0.062 | 0.508 | 0.498 |
| **BASNet** | 0.702 | 0.015 | 0.827 | 0.800 | 0.819 | **0.020** | **0.930** | 0.908 | 0.589 | **0.013** | 0.716 | 0.740 | 0.579 | 0.044 | 0.752 | 0.684 |
| **PFANet** | 0.689 | 0.024 | 0.826 | 0.839 | 0.735 | 0.031 | 0.896 | 0.883 | 0.411 | 0.023 | 0.538 | 0.630 | 0.491 | 0.059 | 0.746 | 0.663 |
| **SFPN** | 0.630 | 0.033 | 0.816 | 0.778 | 0.599 | 0.064 | 0.793 | 0.770 | 0.315 | 0.042 | 0.451 | 0.531 | 0.562 | 0.048 | 0.780 | 0.720 |

Table 3. Quantitative comparison of our proposed model and baseline models across different image with artificial pattern test data




(a) Precision-recall curves




(b) $F$-score as a function of the threshold

Figure 5. The precision-recall curves and the $F$-scores as a function of the thresholds. The four columns in this figure correspond to the four classes of testing patterns outlined in Table 3.

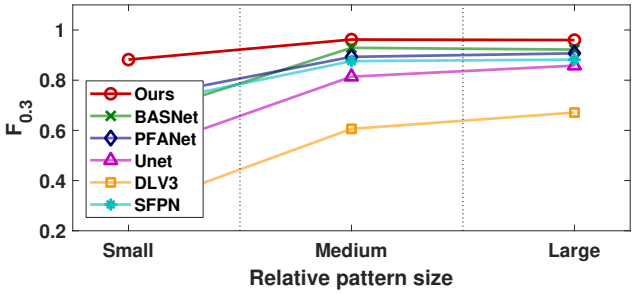



Figure 6. $F_{0.3}$ score as a function of the size of the added patterns. Despite all models are trained using the same training dataset, the proposed method is more resilient to the variation of the size.

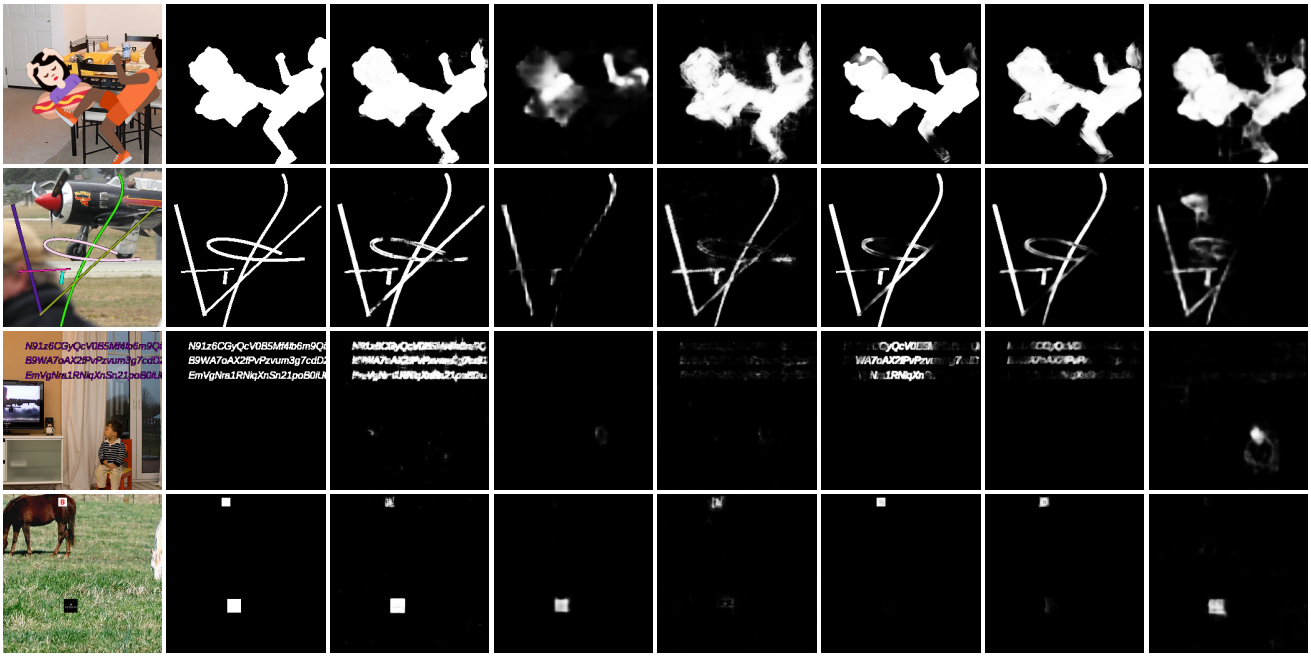
any post-processing steps such as conditional random field [5, 32] to refine the masks.

**Generalization to out-of-distribution patterns**. The mIoU, MAE, and the maximum $F$ scores of the competing methods are summarized in Table 3, whereas the precision-recall curve, and the $F$-score as a function of threshold are shown in Figure 5. We divide our testing scenarios into four categories of data: stickers, lines, texts and logos. Since the models are all trained on the same Apple platform emoji, the performance analysis will indicate the generalization capability of each method.

According to Table 3 and Figure 5, the proposed method demonstrates better generalization than all baselines in all testing scenarios using all the evaluation metrics. Particularly, we make following observations

- Regardless of the model capacity (shown in table 4), models that combine low level (shallower layer) features with high level context (Ours, Unet, BASNet, PFANet, SFPN) for making final prediction generalize better than those not (DLV3).
- Models that use residual unit as building block (Ours, BASNet) generate sharper boundaries

**Scale variations**. Another crucial aspect of solving graphics pattern segmentation problem is the ability to handle a wide range of scales. We show in Figure 6 the maximum $F$-score as function of the relative sizes of the artificial patterns. We consider three scales of the artificial patterns relative to the image size: large, medium, and small. Detailed descriptions of these categories can be found in the supplementary materials. As shown in the figure, the proposed method has the most consistent performance among the competing methods.



(a) Image (b) GT (c) Ours (d) DLV3 (e) UNet (f) BASNet (g) PFANet SFPN

Figure 7. Visual comparison of the proposed method and the competing methods. All models are trained using the same training dataset to ensure a fair comparison. Note that the proposed method can handle very large and very small perturbations, while some competing methods fail to do so.

| | Ours | DLV3 | UNet | BASNet | PFANet | SFPN |
|---|---|---|---|---|---|---|
| # para. | 1.0M | 61M | 0.5M | 87M | 16.4M | 48M |

Table 4. Number of parameters used by competing methods.

**Visual comparison**. In Figure 7 we show several visual comparisons for very large and very small perturbations. We can see that the proposed method produces the best masks across the scales. Additional example can be found in the supplementary material.

**Wild data and ablation study**. Additional results such as the performance on wild test data collected from popular social media websites and extensive ablation studies are presented in the supplementary material due to space limit.

## 5. Conclusion

We formulate the defense against adversarially added artificial graphics patterns as an image segmentation problem and propose a promising solution. The benchmark for such a task is that the method has to handle a wide range of scales and a variety of types, shapes, colors, contrast, brightness, etc. This paper shows that the proposed cascading scheme and explicit multi-scale modeling are effective method for the problem. To train the model, we propose a multi-stage training scheme where we can control the desired precision and recall levels per-stage.

The generalization capability of the proposed method has been tested over the set of graphics patterns we reported. One interesting observation is that majority of the artificial graphics patterns are rendered through a limited set of primitive 2D graphics directives, although the fused images can be stored in any image format. Due to the simplicity and limited options of the available primitive directives, it is recommended to train on the more complex examples (stickers and text). Skipping the simpler patterns such as lines and logos will likely not hurt the performance.

Current system is optimized for opaque graphics patterns under mild corruption (e.g. noise, blur, blocking artifacts introduced by JPEG compression) and is well-suited for majority of the threat so far. However, it is expected that hackers will improve their techniques and exploit vulnerability of current systems. In particular, future research along this direction may focus on improving the performance on translucent patterns (patterns alpha blended with the original content) as well as the robustness against heavy corruption.

**Acknowledgement** The work is funded in part by the Army Research Office under the contract W911NF-20-1-0179 and the National Science Foundation under grants CCF-1763896 and CCF-1718007.

## Supplementary material

This supplementary document summarizes the following experimental results:

- Detailed description of the synthetic test set (section S1).
- Ablation study of the proposed system (section S2).
- Evaluation on real testing downloaded from the internet (section S3).
- Additional visual comparisons (section S4).

## S1. Test Set Details

Our synthetic test set is composed of 12,000 images synthesized with clean natural images and graphics patterns drastically different from training set. We used 4 types of patterns: stickers, lines, text, and logos. For each type of pattern, we synthesize test images at 3 different size levels: large, medium, and small. Sizes are defined slightly different across different patterns as it is very difficult to stipulate a common meaningful definition of effective size taken by a pattern for all 4 patterns (e.g. number of pixels overtaken by the rendered pattern in an image is a good measure of size for stickers and logos but not for text, since to achieve same amount of area as stickers/logos, a text box would occupy much larger part of image). During test set synthesis, we do not perform random attribute alignment between graphics patterns and image. We list the exact definition of size for each category and statistics used during test set synthesis for each category and size level below. Example images are as shown in figure S1


Figure S1. Example test images from each category and size level

**Stickers/logos** The *effective size* of stickers/logos pattern is defined as the ratio between number of pixels overwritten by patterns in an image and the total number of pixels of that image. During synthesis, we render a random number of patterns onto image such that the total effective area is within range for that size level.

- **Small**: size ˜ Uniform(0.001, 0.016), number of patterns $\in [1, 2]$
- **Medium**: size ˜ Uniform(0.016, 0.064), number of patterns $\in [1, 4]$
- **Large**: size ˜ Uniform(0.064, 0.4), number of patterns $\in [1, 12]$

**Lines** The *effective size* of lines is defined as the ratio between width of a line and the length of shorter side of an image. Note although we use term line, the pattern rendered includes both line segment as well as free-form curves.

- **Small**: size ˜ Uniform(0.008, 0.02), number of patterns $\in [1, 10]$
- **Medium**: size ˜ Uniform(0.02, 0.06), number of patterns $\in [1, 10]$
- **Large**: size ˜ Uniform(0.06, 0.15), number of patterns $\in [1, 6]$

**Text** The *effective size* of text is dictated by both the size of a glyph (roughly, width of a single character relative to image width) and the total area of bounding box of text in the image since we could have a very long string of small font text that occupies entire image or a very large single character. We used following number during synthesis

- **Small**: glyph size ˜ Uniform(0.05, 0.1), bounding box size $\in [0.002, 0.016]$
- **Medium**: glyph size ˜ Uniform(0.1, 0.2), bounding box size $\in [0.016, 0.25]$
- **Large**: glyph size ˜ Uniform(0.15, 0.4), bounding box size $\in [0.25, 0.6]$

## S2. Ablation Study of Proposed Solution

We perform ablation study on each element of our proposed solution and show the effectiveness of each proposed element in boosting the performance.

### S2.1. Proposed Attribute Randomization

For data synthesis, we compare our random graphics pattern attribute blending scheme with 1) a simple and straightforward attribute random perturbation, 2) no attribute perturbation. In the simple perturbation scheme, we randomly adjust pattern attributes (e.g. make pattern brighter/darker, less/more saturated, etc.) irrespective of the corresponding local/global attribute at the target location in the image

| | **Overall Performance on Test Set** | | | |
|---|---|---|---|---|
| | mIoU | MAE | $F_{0.3}$ | $F_2$ |
| **Proposed** | **0.745** | **0.022** | **0.873** | **0.863** |
| **Simple** | 0.667 | 0.031 | 0.843 | 0.793 |
| **None** | 0.606 | 0.042 | 0.799 | 0.743 |

Table S1. Quantitative comparison of our proposed attribute randomization versus a simple randomization scheme and no randomization

### S2.2. Impact of JPEG Compression

Albeit already discussed in other scenarios (e.g. Photoshop face manipulation detection by Wang et al. [52]), models trained for our problem heavily relies on low-level image features, and therefore we emphasize the necessity of applying JPEG compression to training data for achieving reasonable generalization to real online images. To this end, we show in figure S2 the performance (F-measure) of different models, trained with/without JPEG compression, on test set under different JPEG compression settings. As can be seen, without JPEG compression during training, model performance is severely impaired by blocking artifacts introduced by JPEG compression even at quality level 90.

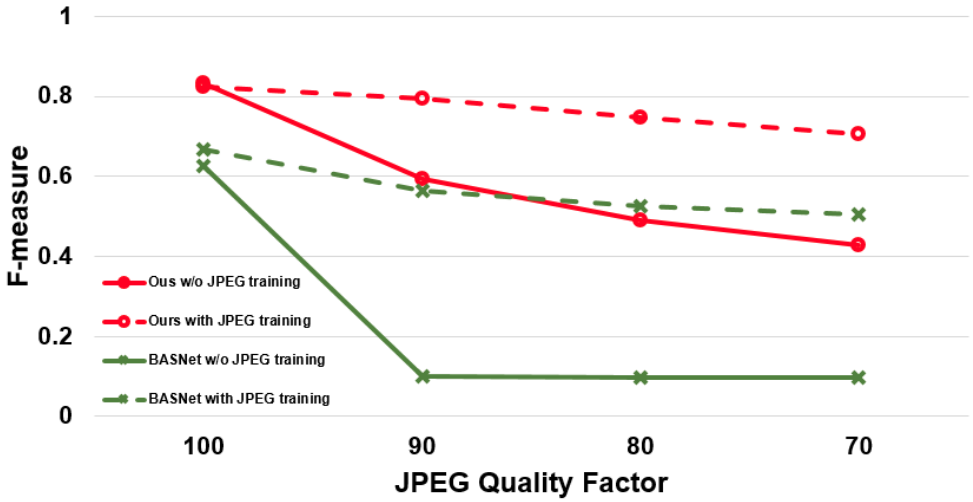



Figure S2. $F_{0.3}$ score as a function of the JPEG compression quality factor. A lower quality factor means less quantization levels on DCT coefficient and therefore stronger blocking artifacts and poorer image quality. Only performance degradation of our proposed model and BASNet is shown for figure clarity, but all models are heavily influenced.

## S2.3. Network Architecture

We investigate the effectiveness of cascading scheme by training variants of our multi-scale cascade network with 1-level (i.e. no cascade), multi-scale input 2/3/4-level cascade, and single scale input 3-level cascade (SS 3-level). For fair comparison, we keep total size (number of parameters) of feature extractors of each network roughly the same. Specifically, we used 12-resblocks for 1-level network feature extractor, 6-resblocks for each of two feature extractors of 2-level network, 4-resblocks for each of three feature extractors of 3-level network, and 3-resblocks for each of four feature extractors of 4-level network. As shown in table S2, although 1-level network demonstrates on-par/slightly better performance on large patterns, our usage of cascading scheme improves consistency of performance across different pattern sizes. The 3-level cascade network that only uses single scale input (input at original resolution of 256x256) achieves similar performance as compared to its multi-scale input counterpart. but it should be noted that: 1) multi-scale version still has better consistency across patterns sizes, 2) single scale version generates very large feature maps at all hidden layers and the computation budget required is much higher at both training and inference time. The extra computation budget limits the possibility of using a larger backbone at each cascade level or cascading more levels.

| | **Performance on Test Set** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Small | | Medium | | Large | | Gap between Large and Small | |
| | mIoU | $F_{0.3}$ | mIoU | $F_{0.3}$ | mIoU | $F_{0.3}$ | $\lvert\Delta mIoU\rvert$ | $\lvert\Delta F_{0.3}\rvert$ |
| **1-level** | 0.545 | 0.695 | 0.735 | 0.856 | 0.809 | 0.923 | 0.262 | 0.228 |
| **2-level** | 0.582 | 0.732 | 0.790 | 0.910 | 0.801 | 0.920 | 0.219 | 0.188 |
| **3-level** | **0.664** | **0.809** | **0.796** | **0.913** | 0.782 | 0.902 | 0.118 | 0.093 |
| **4-level** | 0.616 | 0.782 | 0.757 | 0.893 | 0.724 | 0.869 | **0.108** | **0.087** |
| **SS 3-level** | 0.645 | 0.788 | 0.781 | 0.906 | **0.820** | **0.926** | 0.175 | 0.138 |

Table S2. Quantitative comparison of models with different number of cascade levels. Note that 3-level model performs best on small and medium size patterns and has smallest performance gap between performance on large and small pattern

## S2.4. Multi-stage training

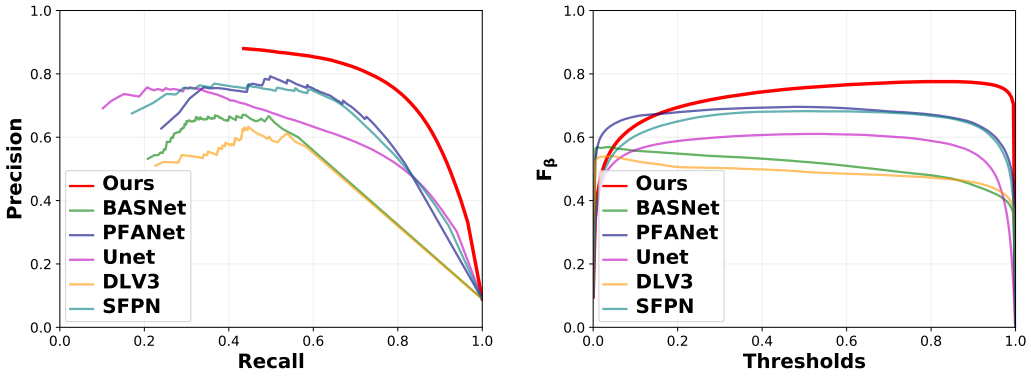
Deep supervision has been demonstrated to be effective in training segmentation networks and is widely adopted by numerous works [18, 19, 40, 51, 53]. Our multi-stage training can be perceived as a special form of deep supervision. Therefore, we compare our training scheme with training entire multi-scale cascade network jointly with supervision on each sub-network. Table S3 shows that our multi-stage training scheme enables our cascade network to achieve better performance.

| | Overall Performance on Test Set | | | |
|---|---|---|---|---|
| | mIoU | MAE | $F_{0.3}$ | $F_2$ |
| **Multi-stage** | **0.745** | **0.022** | **0.873** | **0.863** |
| **Joint** | 0.565 | 0.047 | 0.755 | 0.722 |

Table S3. Quantitative comparison of multi-stage training versus simultaneous joint training

## S3. Wild Data Test

We compare performance of our proposed model and competing models on 100 images collected from popular social media website using same metric as in the main text. Collected images are manually labeled. As shown in figure S3 (a, b) and table S4, our proposed model displays best performance on these wild images. Some visual comparison examples are shown in figure S4. Note that the overall performance is poorer as compared with synthetic test set because some images are heavily corrupted by blur and noise.




(a) Precision-Recall Curve (b) F-measure vs. Thresholds

Figure S3. Precision-recall and F-measure curve of each competing methods evaluated on images collected from popular social media websites. The models here are the same as the ones in the main result section of the paper (i.e. no extra training or additional data used)

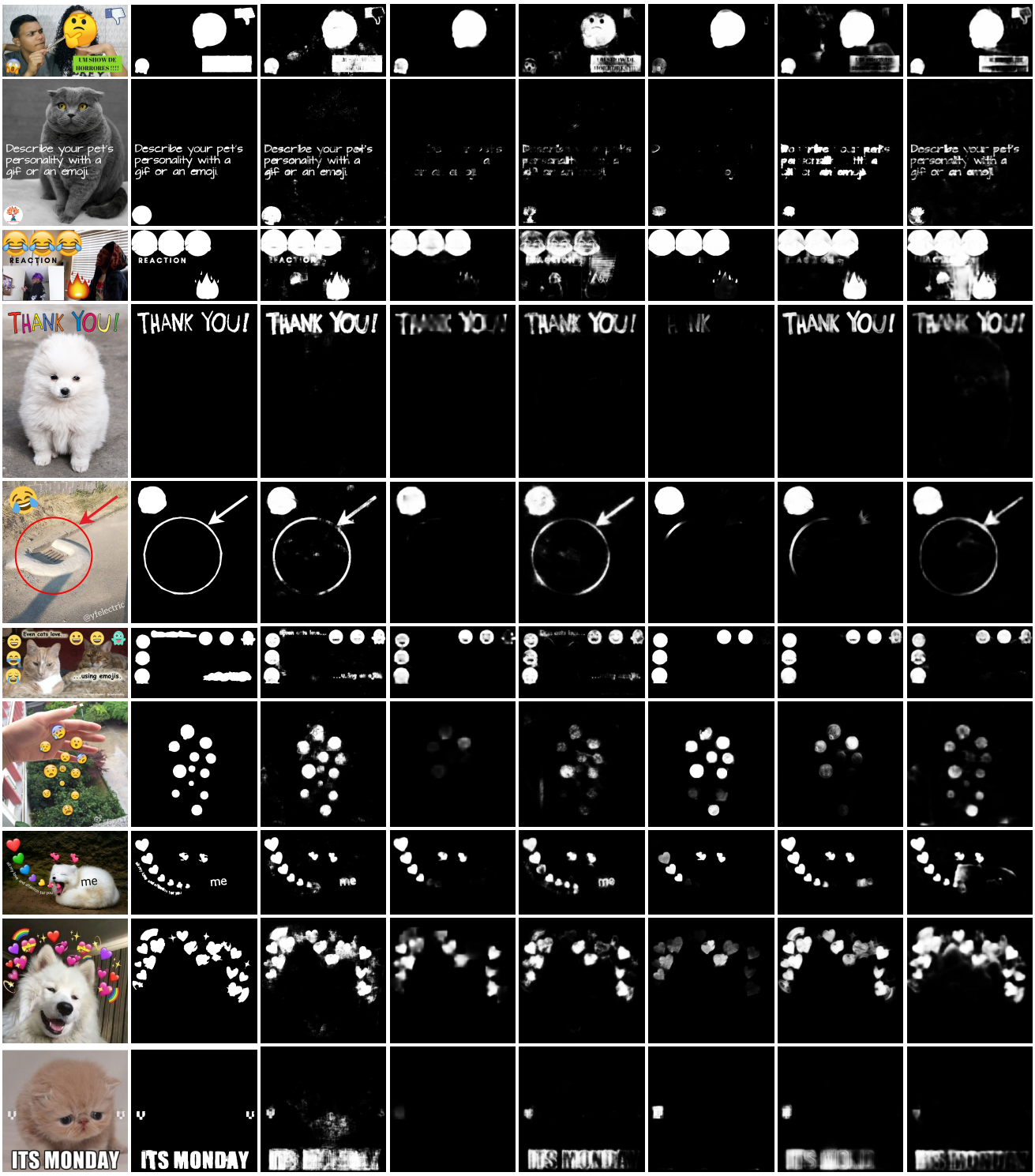
| | Overall Performance on Test Set | | | |
|---|---|---|---|---|
| | mIoU | MAE | $F_{0.3}$ | $F_2$ |
| **Ours** | **0.631** | **0.044** | **0.776** | **0.793** |
| **Unet** | 0.445 | 0.071 | 0.610 | 0.685 |
| **DLV3** | 0.410 | 0.052 | 0.541 | 0.546 |
| **BASNet** | 0.410 | 0.063 | 0.569 | 0.541 |
| **PFANet** | 0.541 | **0.044** | 0.696 | 0.707 |
| **SFPN** | 0.540 | 0.050 | 0.683 | 0.700 |

Table S4. Quantitative comparison of competing methods on data from wild

## S4. Additional Visual Comparisons

Additional results of the visual comparisons are shown in Figure S5, Figure S7, Figure S6, and Figure S8. For each category, we show the perturbations of different sizes, and compare the performance with the competing methods.

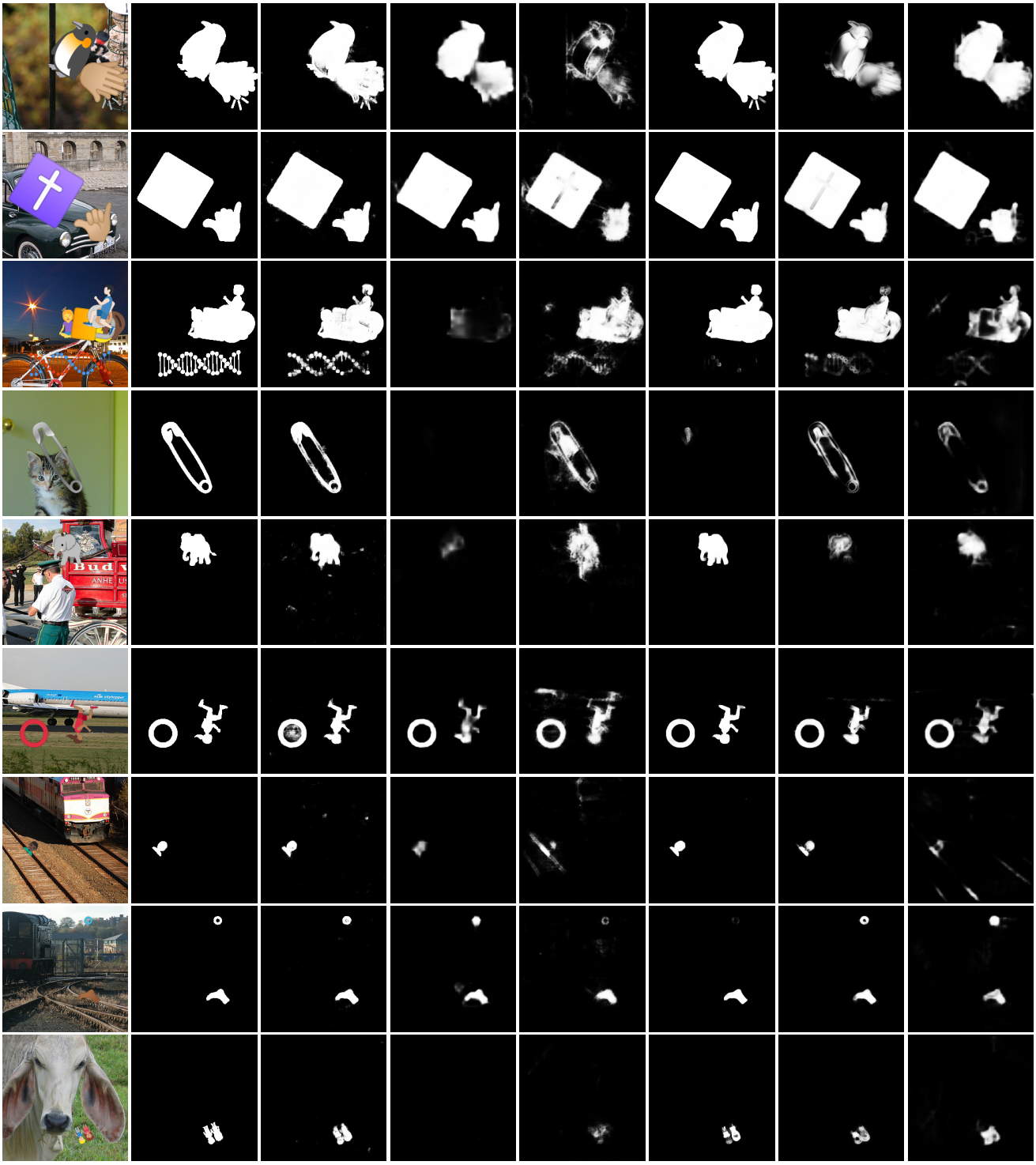
## Real images downloaded from the internet




(a) Image (b) GT (c) Ours (d) DLV3 (e) UNet (f) BASNet (g) PFANet (h) SFPN

Figure S4. Visual comparison of the proposed method and the competing methods on wild data collected from popular social media websites

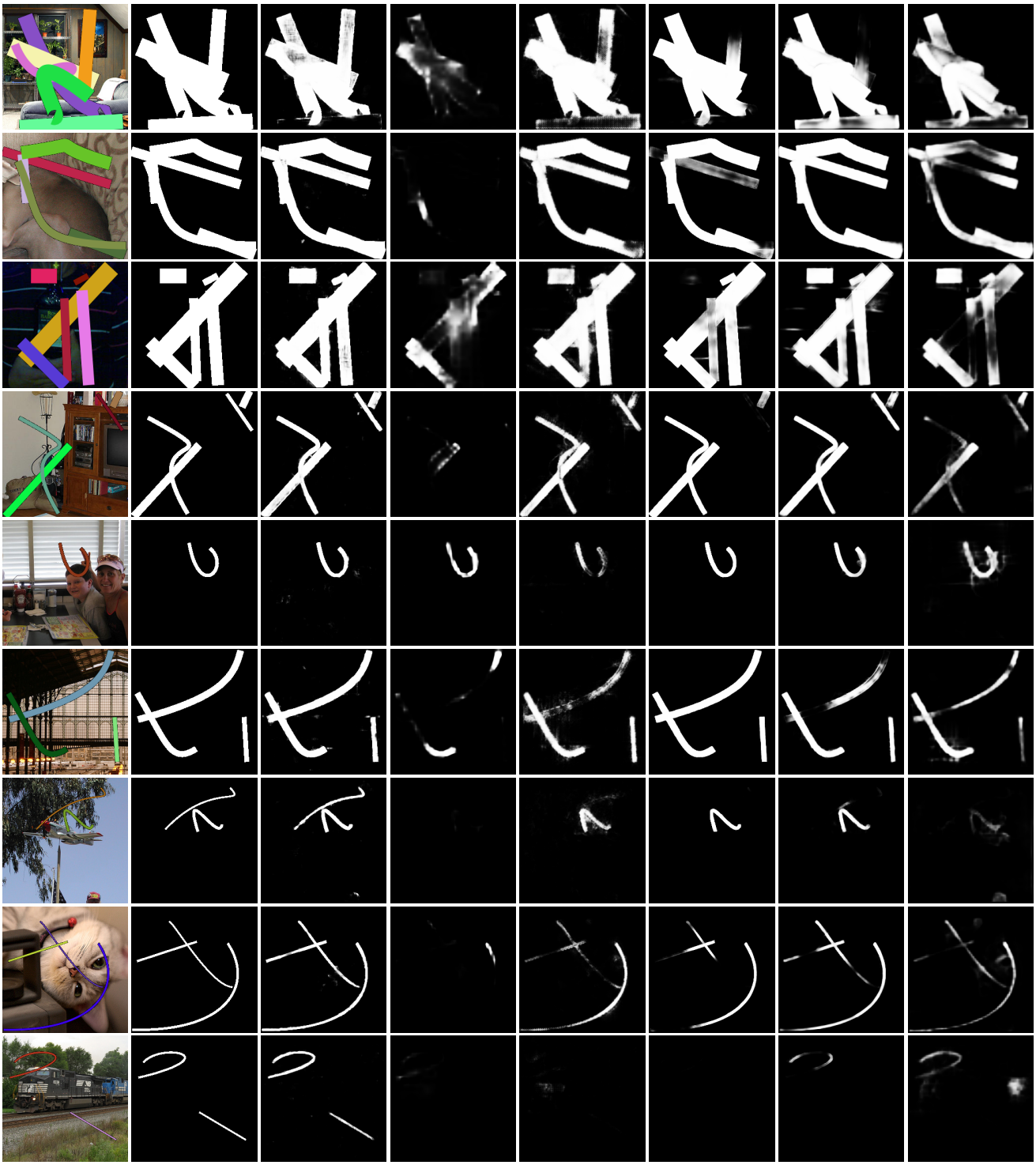
## More Test Set Visual Comparisons (1): Stickers



(a) Image (b) GT (c) Ours (d) DLV3 (e) UNet (f) BASNet (g) PFANet (h) SFPN

Figure S5. Visual comparison of the proposed method and the competing methods on stickers

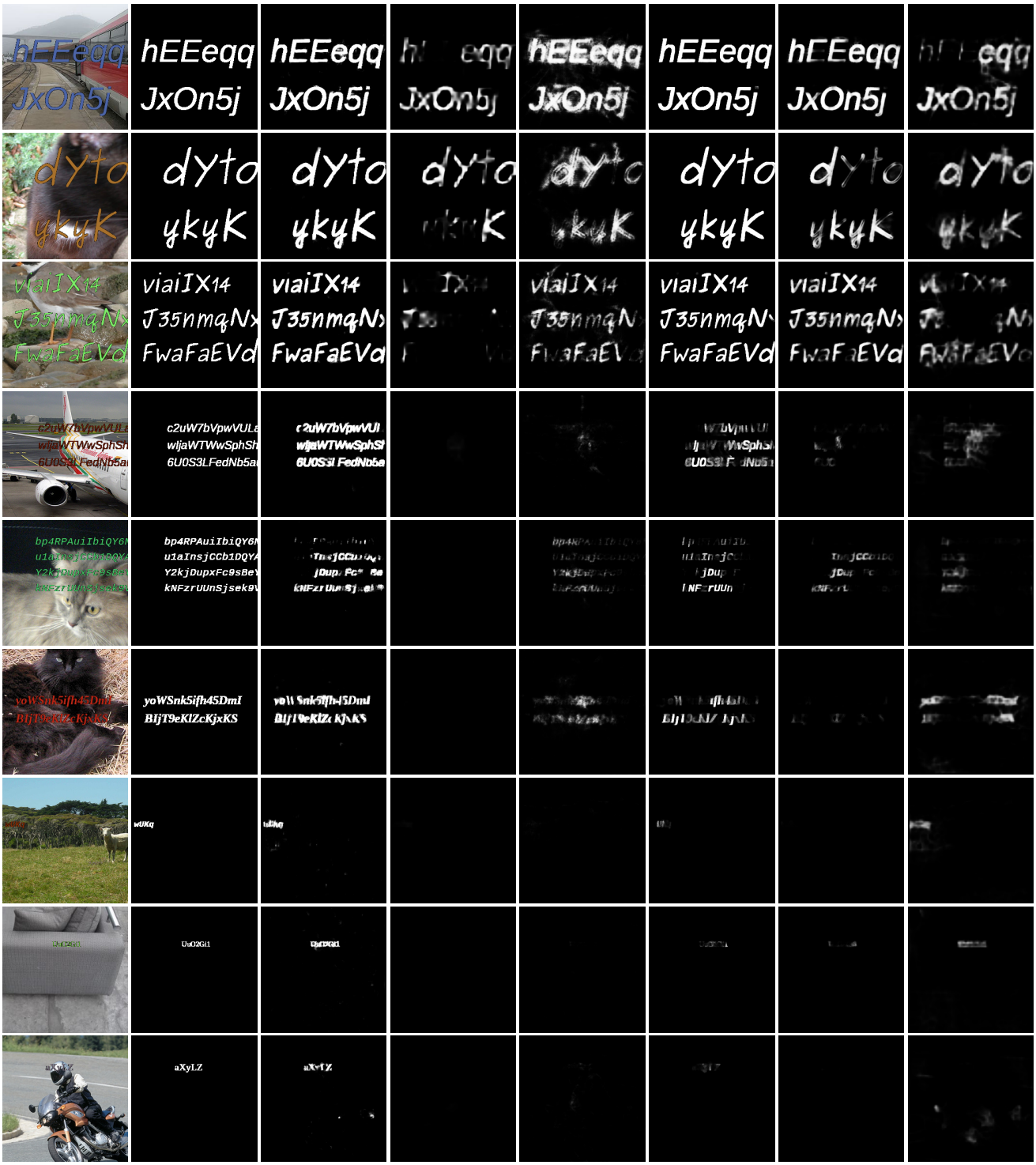
## More Test Set Visual Comparisons (2): Lines



(a) Image (b) GT (c) Ours (d) DLV3 (e) UNet (f) BASNet (g) PFANet (h) SFPN

Figure S6. Visual comparison of the proposed method and the competing methods on lines

## More Test Set Visual Comparisons (3): Text



(a) Image (b) GT (c) Ours (d) DLV3 (e) UNet (f) BASNet (g) PFANet (h) SFPN

Figure S7. Visual comparison of the proposed method and the competing methods on text

## More Test Set Visual Comparisons (4): Logo


(a) Image (b) GT (c) Ours (d) DLV3 (e) UNet (f) BASNet (g) PFANet (h) SFPN

Figure S8. Visual comparison of the proposed method and the competing methods on logos

## References


[1] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *CVPR*, 2009. 6

[2] Irene Amerini, Tiberio Uricchio, Lamberto Ballan, and Roberto Caldelli. Localization of JPEG double compression through multi-domain convolutional neural networks. In *CVPR Workshops*, 2017. 2

[3] Xavier Boix, Josep M. Gonfaus, Joost van de Weijer, Andrew D. Bagdanov, Joan Serrat Gual, and Jordi Gonzàlez. Harmony potentials - Fusing global and local scale for semantic image segmentation. *Int. J. Comput. Vis.*, 96(1):83–102, 2012. 2

[4] Luca Bondi, Silvia Lameri, David Güera, Paolo Bestagini, Edward J Delp, and Stefano Tubaro. Tampering detection and localization through clustering of camera-based CNN features. In *CVPR Workshops*, 2017. 2

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2017. 2, 7

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2, 6

[7] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016. 2, 3

[8] Wuyang Chen, Ziyu Jiang, Zhangyang Wang, Kexin Cui, and Xiaoning Qian. Collaborative global-local networks for memory-efficient segmentation of ultra-high resolution images. In *CVPR*, 2019. 2

[9] Ho Kei Cheng, Jihoon Chung, Yu-Wing Tai, and Chi-Keung Tang. CascadePSP: Toward class-agnostic and very high-resolution segmentation via global and local refinement. In *CVPR*, 2020. 2

[10] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):569–582, 2014. 2

[11] Patrick Ferdinand Christ, Mohamed Ezzeldin A Elshaer, Florian Ettlinger, et al. Automatic liver and lesion segmentation in CT using cascaded fully convolutional neural networks and 3D conditional random fields. In *Intl. Conf. Med. Image Comput. Comput. Assist. Interv.* Springer, 2016. 2

[12] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *IEEE Intl. Workshop Inf. Forensics Security*, pages 1–6, 2015. 2

[13] Tali Dekel, Michael Rubinstein, Ce Liu, and William T. Freeman. On the effectiveness of visible watermarks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[14] Cheng Deng, Xu Yang, Feiping Nie, and Dapeng Tao. Saliency detection via a multiple self-weighted graph-based manifold ranking. *IEEE Trans. Multimedia*, 22(4):885–896, 2019. 2, 4

[15] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1915–1929, 2012. 2, 3

[16] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 1

[17] Guanbin Li and Y. Yu. Visual saliency based on multiscale deep features. In *CVPR*, 2015. 2, 3

[18] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *CVPR*, 2019. 2, 11

[19] Qibin Hou, Ming-Ming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip HS Torr. Deeply supervised salient object detection with short connections. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(4):815–828, 2019. 2, 3, 11

[20] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A. Efros. Fighting fake news: Image splice detection via learned self-consistency. In *ECCV*, 2018. 2

[21] Zhuolin Jiang and Larry S. Davis. Submodular salient region detection. In *CVPR*, 2013. 2, 4

[22] Sasan Karamizadeh and Abouzar Arabsorkhi. Methods of pornography detection: Review. In *Proceedings of the 10th International Conference on Computer Modeling and Simulation*, ICCMS 2018, page 33–38, New York, NY, USA, 2018. Association for Computing Machinery. 1

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[24] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollar. Panoptic feature pyramid networks. In *CVPR*, 2019. 2, 3, 6

[25] Lubor Ladicky, Chris Russell, Pushmeet Kohli, and Philip H. S. Torr. Associative hierarchical CRFs for object class image segmentation. In *ICCV*, 2009. 2

[26] Gayoung Lee, Yu-Wing Tai, and Junmo Kim. Deep saliency with encoded low level distance map and high level features. In *CVPR*, 2016. 2

[27] Victor S. Lempitsky, Andrea Vedaldi, and Andrew Zisserman. Pylon model for semantic segmentation. In *NeurIPS*, 2011. 2

[28] Guanbin Li and Yizhou Yu. Deep contrast learning for salient object detection. In *CVPR*, 2016. 2, 3

[29] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *CVPR*, 2015. 2

[30] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *CVPR*, 2017. 2

[31] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017. 2

[32] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016. 2, 3, 7

[33] Nian Liu and Junwei Han. DHSNet: Deep hierarchical saliency network for salient object detection. In *CVPR*, 2016. 2, 3

[34] Zhiming Luo, Akshaya Mishra, Andrew Achkar, Justin Eichel, Shaozi Li, and Pierre-Marc Jodoin. Non-local deep features for salient object detection. In *CVPR*, 2017. 2, 3

[35] Tim Ken Mackey and Bryan A Liang. Global reach of direct-to-consumer advertising using social media for illicit online drug sales. *J Med Internet Res*, 15(5):e105, May 2013. 1

[36] Elizabeth M. Morgan, Chareen Snelson, and Patt Elison-Bowers. Image and video disclosure of substance use on social media websites. *Computers in Human Behavior*, 26(6):1405–1411, 2010. Online Interactivity: Role of Technology in Behavior Change. 1

[37] Mohammadreza Mostajabi, Payman Yadollahpour, and Gregory Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015. 2, 3

[38] Venkatesh N Murthy, Vivek Singh, Terrence Chen, R Manmatha, and Dorin Comaniciu. Deep decision network for multi-class image classification. In *CVPR*, 2016. 2

[39] Alin C Popescu and Hany Farid. Exposing digital forgeries by detecting traces of resampling. *IEEE Trans. Signal Process.*, 53(2):758–767, 2005. 2

[40] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. BASNet: Boundary-aware salient object detection. In *CVPR*, 2019. 2, 3, 6, 11

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Intl. Conf. Med. Image Comput. Comput. Assist. Interv.*, 2015. 2, 6

[42] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, and Reinhard Klette. Deep-cascade: Cascading 3D deep neural networks for fast anomaly detection and localization in crowded scenes. *IEEE Trans. Image Process.*, 26(4):1992–2004, 2017. 2

[43] Alexander Sage, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. LLD - Large Logo Dataset - version 0.1. https://data.vision.ee.ethz.ch/cvl/lld, 2017. 6

[44] Ronald Salloum, Yuzhuo Ren, and C.-C. Jay Kuo. Image splicing localization using a multi-task fully convolutional network (MFCN). *J. Vis. Commun. Image Represent.*, 51:201 – 209, 2018. 2

[45] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017. 2, 5

[46] Hang Su, Xiatian Zhu, and Shaogang Gong. Deep learning logo detection with data expansion by synthesising context. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 530–539, 2017. 2

[47] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, 2013. 2

[48] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 2

[49] Paul Viola and Michael Jones. Robust real-time object detection. *Intl. J. Comp. Vision*, 57:137–154, 01 2001. 2

[50] Paulo Vitorino, Sandra Avila, Mauricio Perez, and Anderson Rocha. Leveraging deep neural networks to fight child pornography in the age of social media. *Journal of Visual Communication and Image Representation*, 50:303–313, 2018. 1

[51] Linzhao Wang, Lijun Wang, Huchuan Lu, Pingping Zhang, and Xiang Ruan. Salient object detection with recurrent fully convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(7):1734–1746, 2019. 2, 11

[52] Sheng-Yu Wang, Oliver Wang, Andrew Owens, Richard Zhang, and Alexei A Efros. Detecting photoshopped faces by scripting photoshop. In *ICCV*, 2019. 10

[53] Tiantian Wang, Ali Borji, Lihe Zhang, Pingping Zhang, and Huchuan Lu. A stagewise refinement model for detecting salient objects in images. In *ICCV*, 2017. 2, 11

[54] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 2, 3

[55] Michele L. Ybarra and Kimberly J. Mitchell. Exposure to internet pornography among children and adolescents: A national survey. *CyberPsychology & Behavior*, 8(5):473–486, 2005. PMID: 16232040. 1

[56] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018. 2

[57] Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Xiang Ruan. Amulet: Aggregating multi-level convolutional features for salient object detection. In *ICCV*, 2017. 2, 3

[58] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2, 3

[59] Rui Zhao, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Saliency detection by multi-context deep learning. In *CVPR*, 2015. 2, 3

[60] Ting Zhao and Xiangqian Wu. Pyramid feature attention network for saliency detection. In *CVPR*, 2019. 2, 3, 6

[61] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017. 2