OCR实战

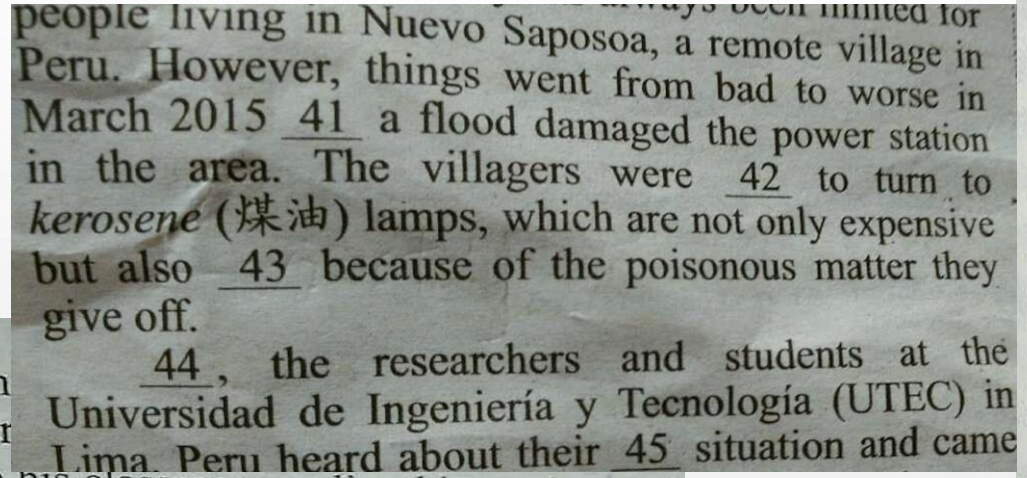
第六课 专题讨论

第六课 专题讨论

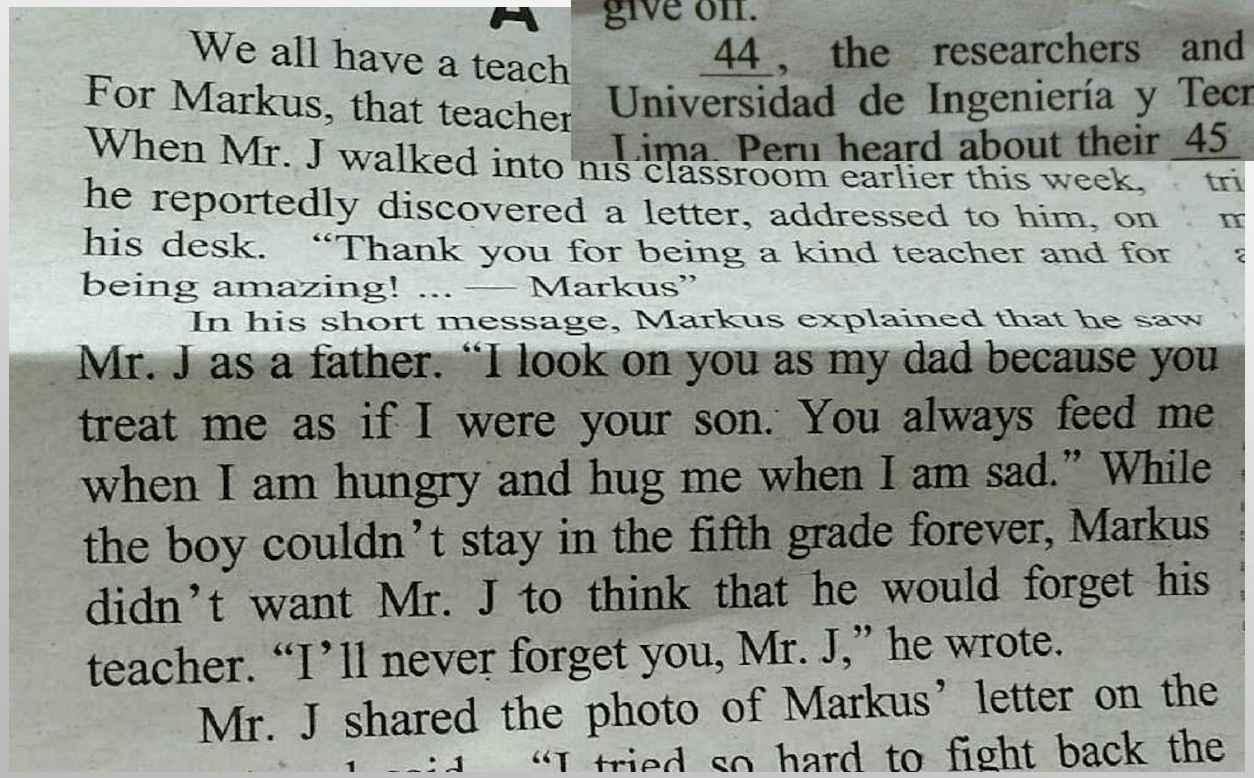
- 图像增加
  - UNWARPING
  - SUPER-RESOLUTION
- 自然场景文字检测（三）
  - PIXEL LINK
  - PSE NET

常见挑战

- 扭曲形变

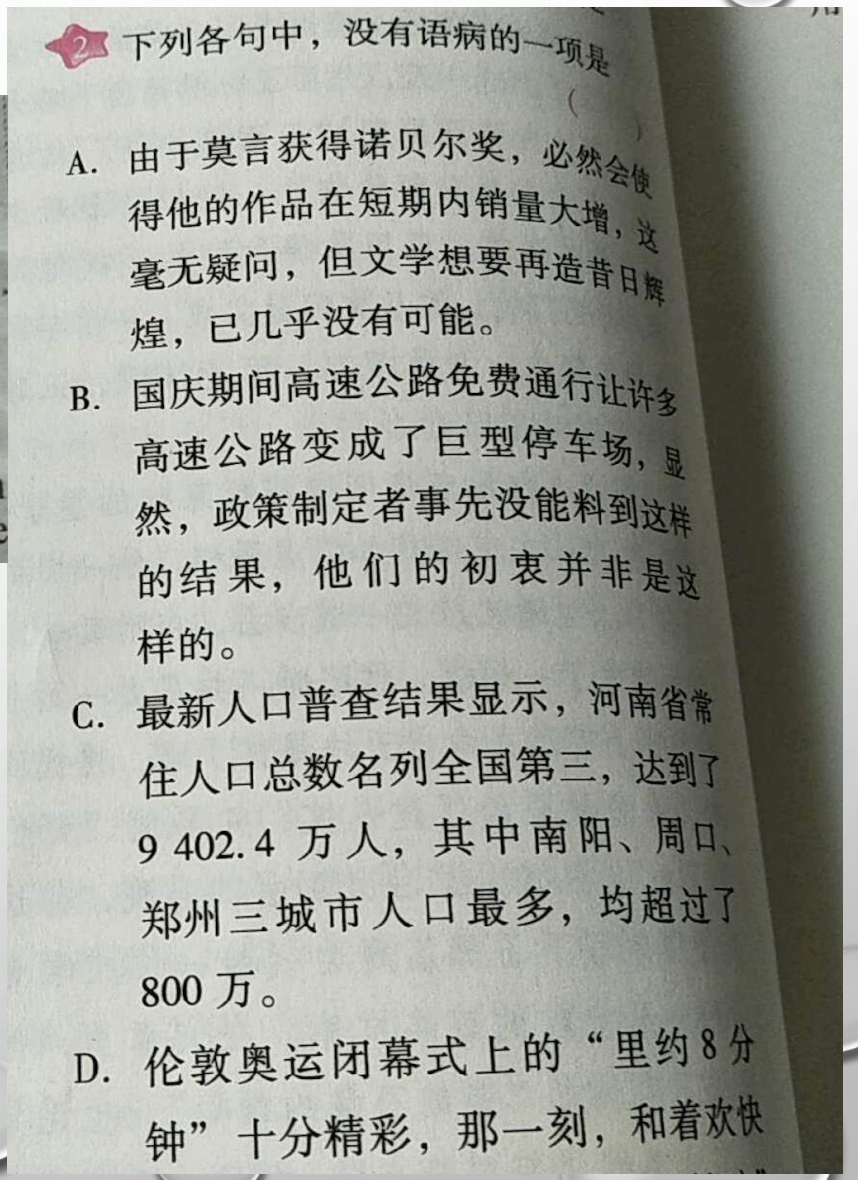

people living in Nuevo Saposoa, a remote village in Peru. However, things went from bad to worse in March 2015 41 a flood damaged the power station in the area. The villagers were 42 to turn to *kerosene* (煤油) lamps, which are not only expensive but also 43 because of the poisonous matter they give off.

44, the researchers and students at the Universidad de Ingeniería y Tecnología (UTEC) in Lima, Peru heard about their 45 situation and came


We all have a teach
For Markus, that teacher
When Mr. J walked into his classroom earlier this week, he reportedly discovered a letter, addressed to him, on his desk. "Thank you for being a kind teacher and for being amazing! ... — Markus"

In his short message, Markus explained that he saw Mr. J as a father. "I look on you as my dad because you treat me as if I were your son. You always feed me when I am hungry and hug me when I am sad." While the boy couldn't stay in the fifth grade forever, Markus didn't want Mr. J to think that he would forget his teacher. "I'll never forget you, Mr. J," he wrote.

Mr. J shared the photo of Markus' letter on the


2. 下列各句中，没有语病的一项是（　　）

A. 由于莫言获得诺贝尔奖，必然会使得他的作品在短期内销量大增，这毫无疑问，但文学想要再造昔日辉煌，已几乎没有可能。

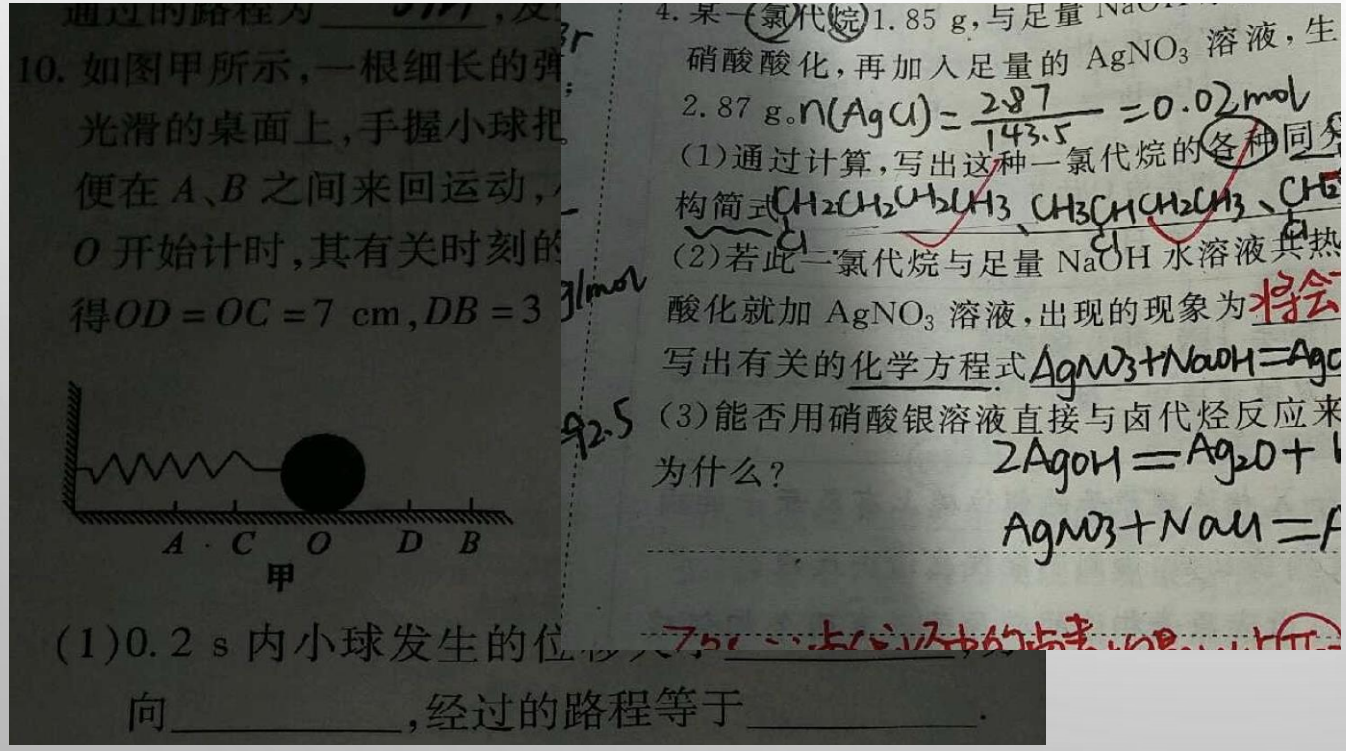
B. 国庆期间高速公路免费通行让许多高速公路变成了巨型停车场，显然，政策制定者事先没能料到这样的结果，他们的初衷并非是这样的。

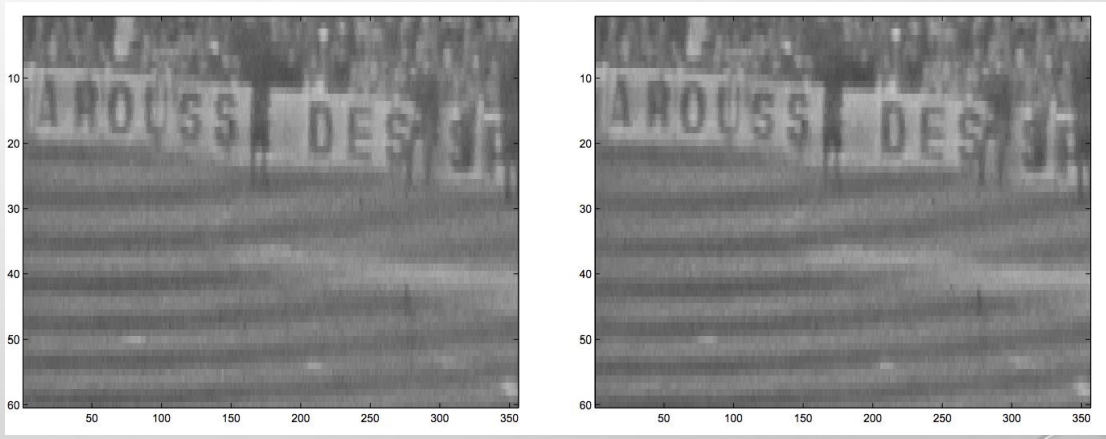
C. 最新人口普查结果显示，河南省常住人口总数名列全国第三，达到了9 402.4 万人，其中南阳、周口、郑州三城市人口最多，均超过了800 万。

D. 伦敦奥运闭幕式上的"里约 8 分钟"十分精彩，那一刻，和着欢快

常见挑战

- 不均匀光照 / 反光 / 弱光
- 失焦 / 运动模糊 / 摩尔纹





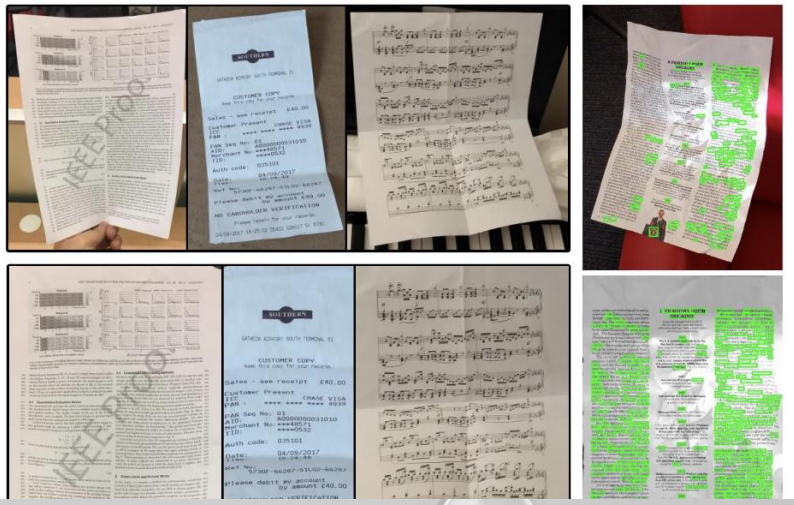
图像增强

DocUNet: Document Image Unwarping via A Stacked U-Net

Ke Ma[1] Zhixin Shu[1] Xue Bai[2] Jue Wang[2] Dimitris Samaras[1]
[1]Stony Brook University [2]Megvii Inc.
[1]{kemma, zhshu, samaras}@cs.stonybrook.edu [2]{baixue, wangjue}@megvii.com

Abstract

Capturing document images is a common way for digitizing and recording physical documents due to the ubiquitousness of mobile cameras. To make text recognition easier, it is often desirable to digitally flatten a document image when the physical document sheet is folded or curved. In this paper, we develop the first learning-based method to achieve this goal. We propose a stacked U-Net [25] with intermediate supervision to directly predict the forward mapping from a distorted image to its rectified version. Because large-scale real-world data with ground truth deformation



https://www3.cs.stonybrook.edu/~cvl/content/papers/2018/Ma_CVPR18.pdf

图像增强

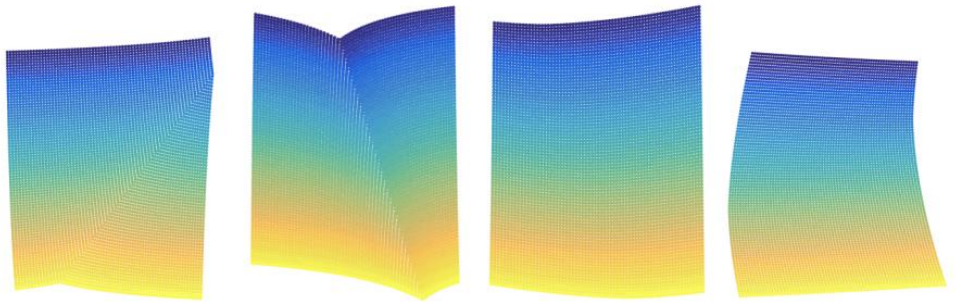
Perturbed mesh generation: Given an image I, we impose an $m \times n$ mesh M on it to provide control points for warping. A random vertex $\boldsymbol{p}$ is selected on M as the initial deformation point. The direction and strength of the deformation is denoted as $\boldsymbol{v}$ and is also randomly generated. Finally, based on observation i), $\boldsymbol{v}$ is propagated to other vertices by weight w. The vertices on the distorted mesh are computed as $\boldsymbol{p_i} + w\boldsymbol{v}, \forall i$.

It is crucial to define w. As $\boldsymbol{p}$ and $\boldsymbol{v}$ define a straight line, we first compute the normalized distance d between each vertex and this line, and define w as a function of d. Based on observation ii), we define a function for each distortion type. For folds:

$$w = \frac{\alpha}{d + \alpha}, \tag{2}$$

and for curves:

$$w = 1 - d^{\alpha}, \tag{3}$$



(a) (b) (c) (d)

Figure 4. **Effects of deformation functions.** (a) and (b) show the fold effect based on Eq. 2. (a) for large α, (b) for small α. (c) and (d) show the curve effect based on Eq. 3. (c) for large α and (d) for small α.

Figure 5. **Sample images in the synthetic dataset.**

图像增强

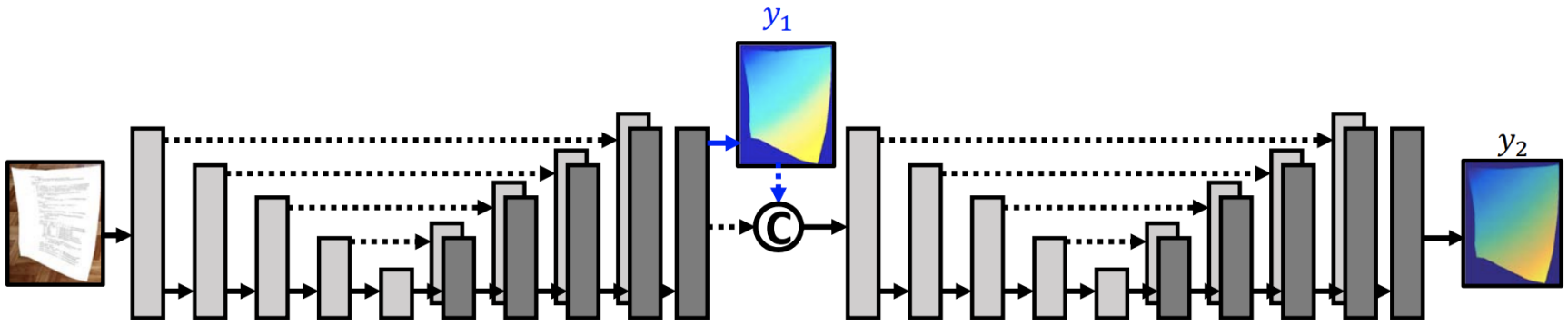


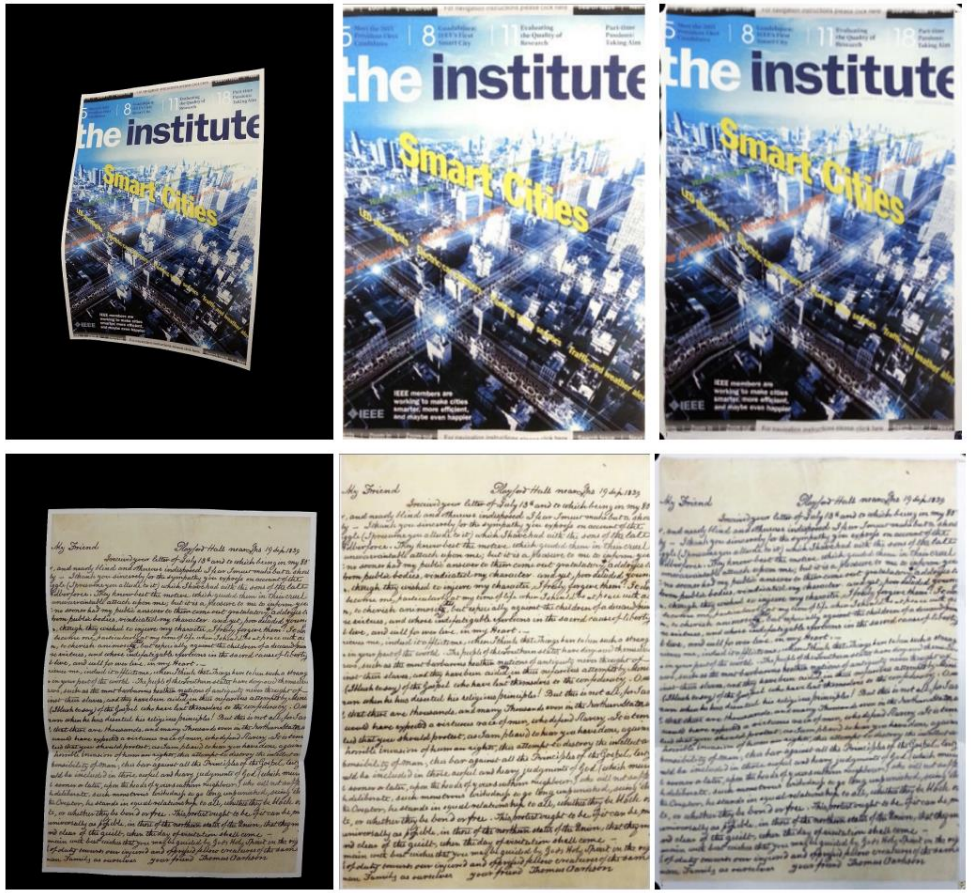
Figure 3. **Network architecture.** Our network is a stack of two U-Nets. The network splits and produces a forward mapping y_1 from the output of the first U-Net. The same loss applied at y_2 is also applied at y_1. Then y_1 is concatenated with the output feature map of the first U-Net and serves as the input for the second U-Net.© represents the concatenation operator. y_2 can be directly used to generate the rectified image.

We formulate this task as seeking the appropriate 2D image warping that can rectify a distorted document image. Our network predicts a mapping field that moves a pixel in the distorted source image $S(u, v)$ to (x, y) in the result image D:

$$D(x, y) = S(u, v). \quad (1)$$

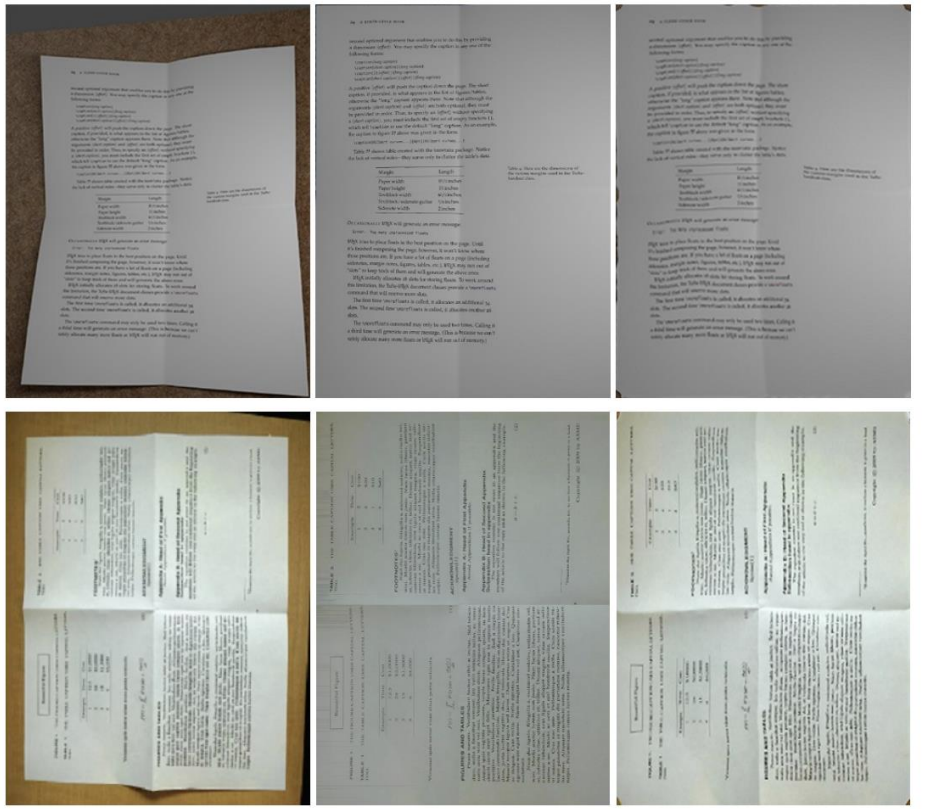
Formulating the problem in this way, we find this task shares some commonalities with semantic segmentation. For the latter, the network assigns a class label to each pixel. Similarly, our network assigns a 2-dimensional vector to each pixel. This inspires us to use U-Net [25] in our network structure, which is widely known for its success in semantic segmentation. To adapt it to our regression problem, we define a novel loss function to drive the network to regress the coordinate (x, y) in D for each pixel in S.

图像增强



distorted image result of [36] our result

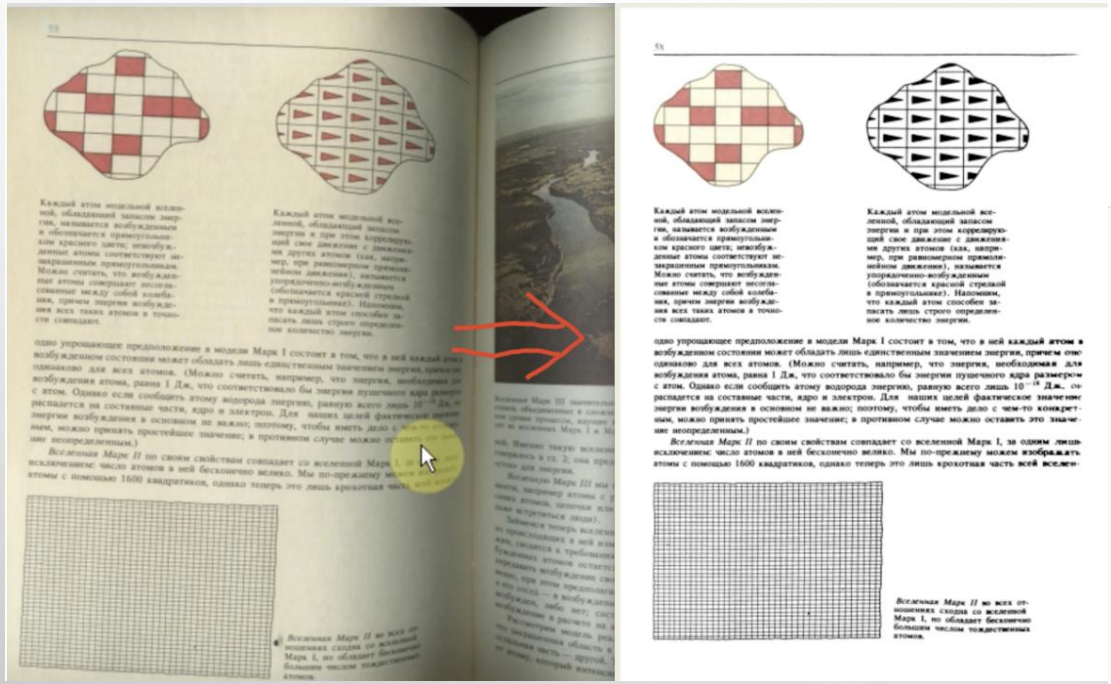
Figure 8. **Comparison with You et al. [36].** Even though [36] utilizes five to ten images, our method is still quite comparable.

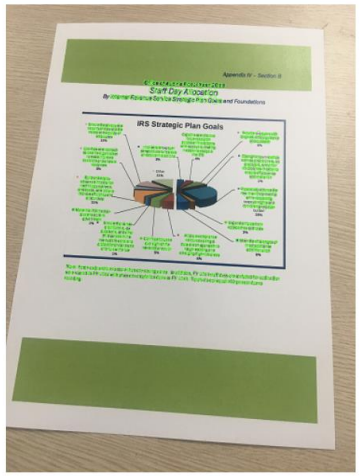


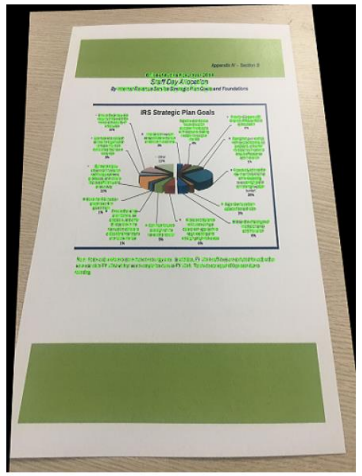
distorted image result of [6] our result

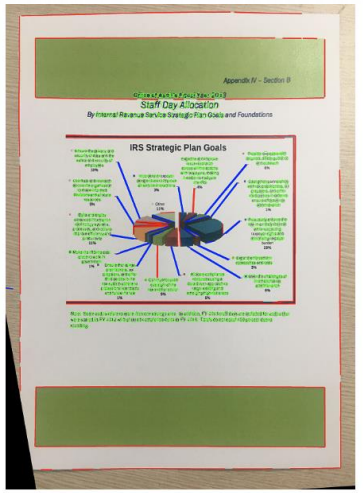
Figure 9. **Comparison with Das et al. [6].** The method in [6] is specially designed to work well for two folds condition. Our method also works well under this condition. The two images are from [6].

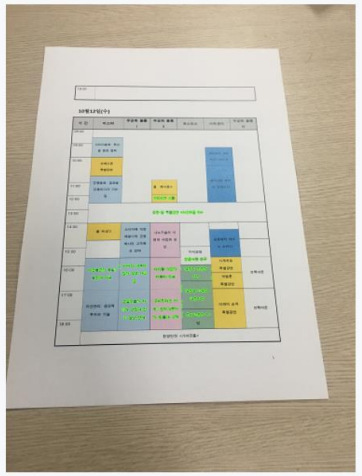
图像增强

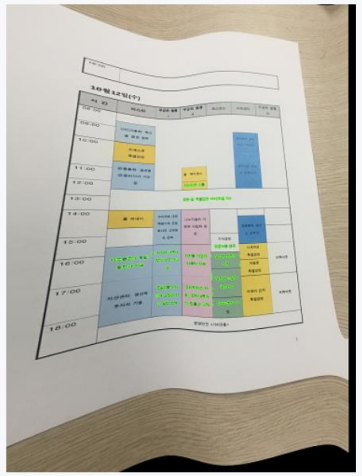


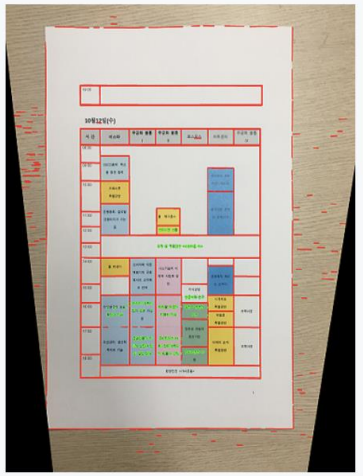












- HTTP://S1NH.ORG/POST/DOCUMENT-LAYOUT-ANALYSIS/
- HTTPS://GITHUB.COM/XELLOWS1305/DOCUMENT-IMAGE-DEWARPING

图像增强 -- SR

Deep Learning for Image Super-resolution: A Survey

Zhihao Wang, Jian Chen, Steven C.H. Hoi, Fellow, IEEE

Abstract—Image Super-Resolution (SR) is an important class of image processing techniques to enhance the resolution of images and videos in computer vision. Recent years have witnessed remarkable progress of image super-resolution using deep learning techniques. In this survey, we aim to give a survey on recent advances of image super-resolution techniques using deep learning approaches in a systematic way. In general, we can roughly group the existing studies of SR techniques into three major categories: supervised SR, unsupervised SR, and domain-specific SR. In addition, we also cover some other important issues, such as publicly available benchmark datasets and performance evaluation metrics. Finally, we conclude this survey by highlighting several future directions and open issues which should be further addressed by the community in the future.

Index Terms—Image Super-resolution, Deep Learning, Convolutional Neural Networks (CNN), Generative Adversarial Nets (GAN)

https://arxiv.org/pdf/1902.06068.pdf

图像增强 -- SR

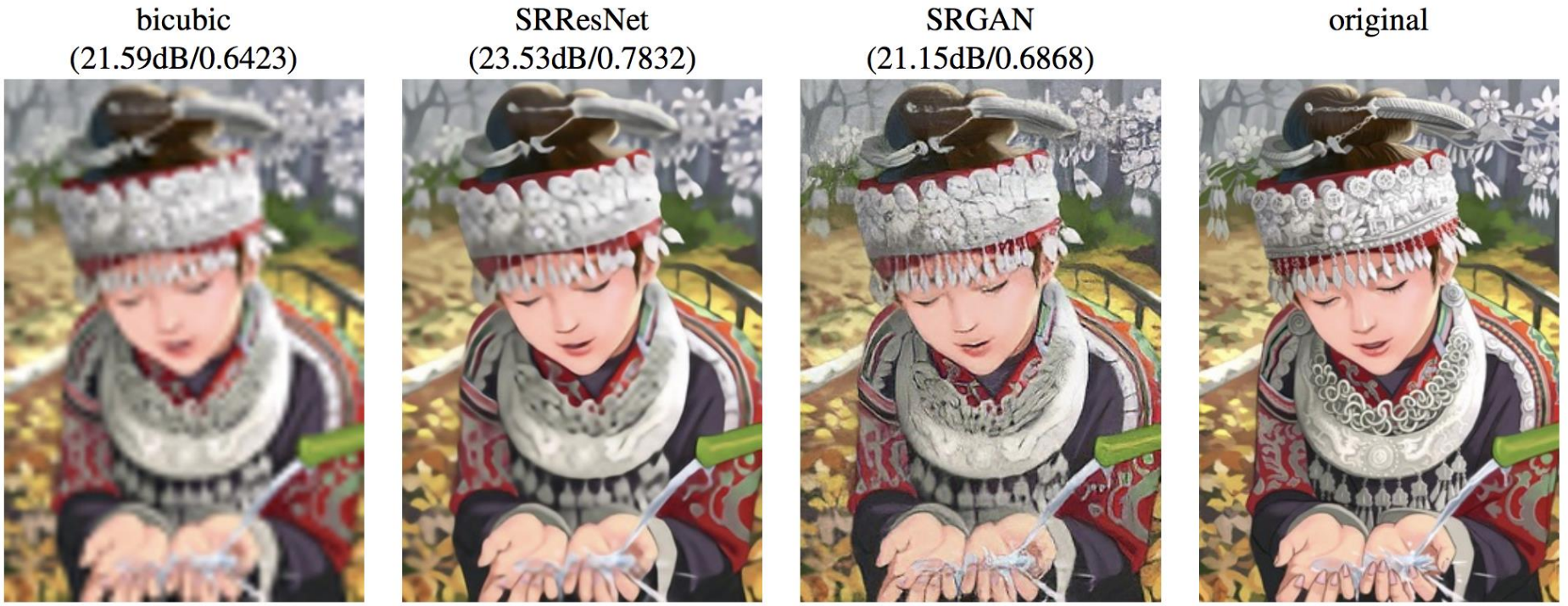


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

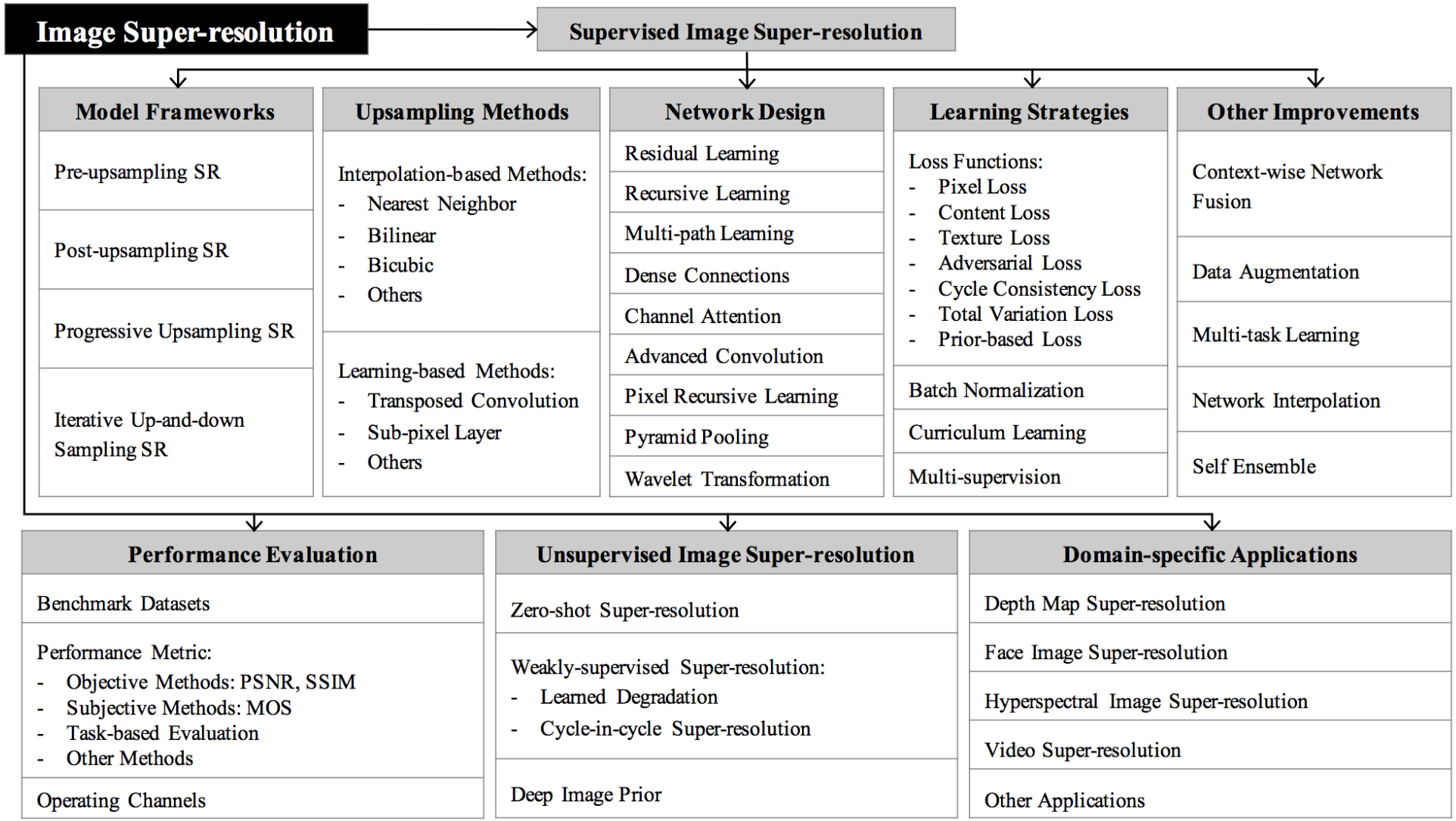
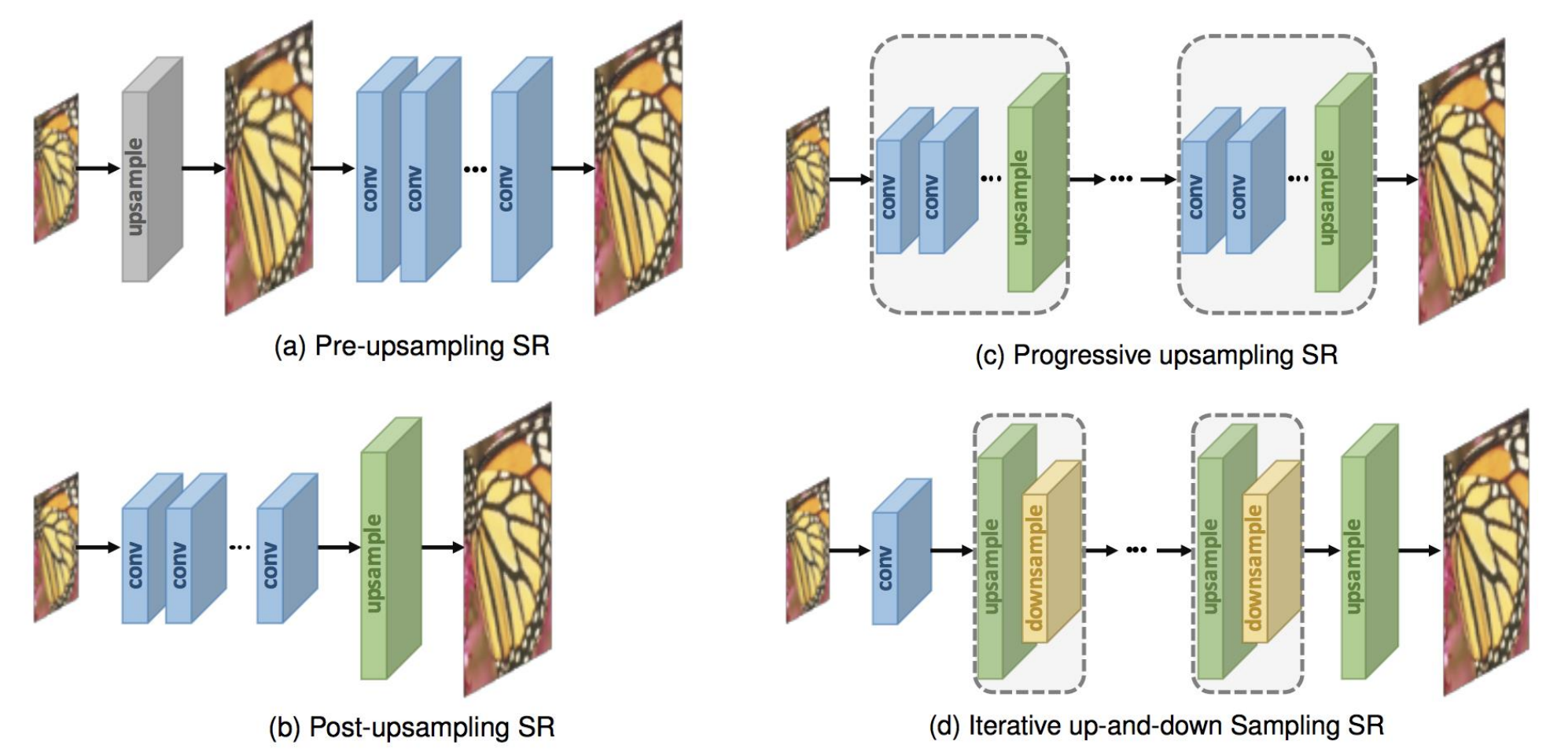


Fig. 1. Hierarchically-structured taxonomy of this survey.

图像增强 -- SR


upsample
conv
conv
conv
(a) Pre-upsampling SR
conv
conv
upsample
conv
conv
upsample
(c) Progressive upsampling SR
conv
conv
conv
upsample
(b) Post-upsampling SR
conv
upsample
downsample
upsample
downsample
upsample
(d) Iterative up-and-down Sampling SR

图像增强 -- SR

- UPSAMPLING METHODS
  - INTERPOLATION-BASED UPSAMPLING
    - NEAREST-NEIGHBOR INTERPOLATION
    - BILINEAR INTERPOLATION
    - BICUBIC INTERPOLATION

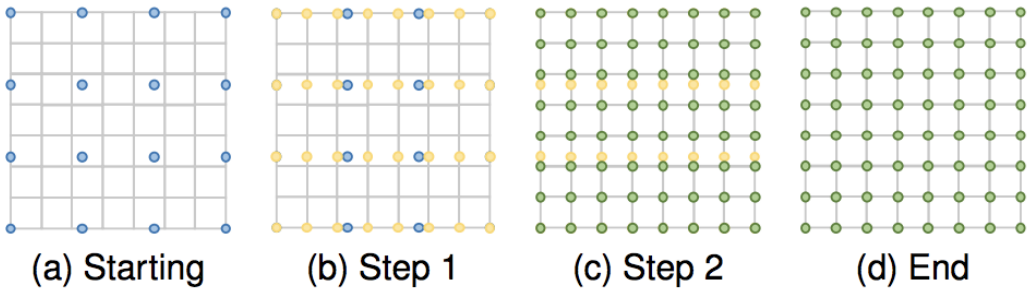


Fig. 3. Interpolation-based upsampling methods. The gray board denotes the coordinates of pixels, and the blue, yellow and green points represent the initial, intermediate and final pixels, respectively.

Bilinear Interpolation. The bilinear interpolation first performs linear interpolation on one axis of the image and then performs it again on the other axis. This two-step interpolation process is shown in Fig. 3. Although each step is linear in the sampled values and positions, it results in a quadratic interpolation with a receptive field sized 2×2, and shows much better performance than nearest-neighbor interpolation while keeping relatively fast speed.

Bicubic Interpolation. Similarly, the bicubic interpolation [12] performs a cubic interpolation on each of the two dimensions of the image, as Fig. 3 shows. Compared to bilinear interpolation, the bicubic interpolation takes 4×4 pixels into count, and thus generates smoother results with fewer interpolation artefacts and lower speed. In fact, the bicubic interpolation with anti-aliasing is currently the mainstream method for constructing SR datasets (i.e., degrading HR images to corresponding LR images), and is also widely used in pre-upsampling SR framework (Sec. 3.1.1).

图像增强 -- SR

- UPSAMPLING METHODS
  - LEARNING-BASED UPSAMPLING
    - TRANSPOSED CONVOLUTION LAYER
    - SUB-PIXEL LAYER

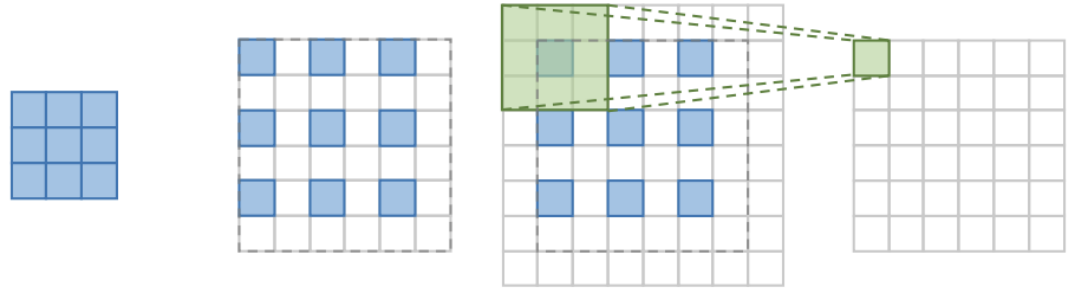


Fig. 4. Transposed convolution layer. The blue boxes denote the input, and the green boxes indicate the kernel and the output of the convolution operation.

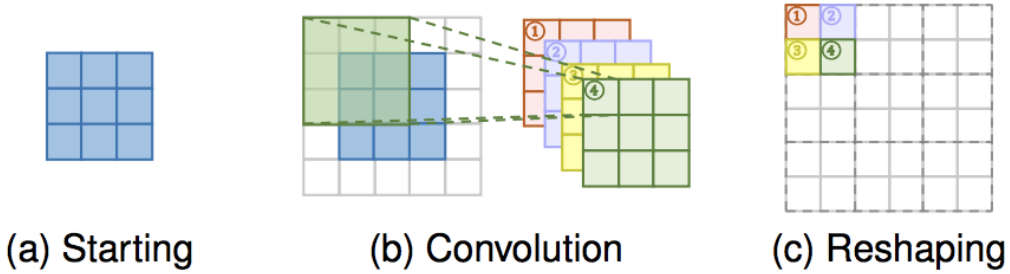
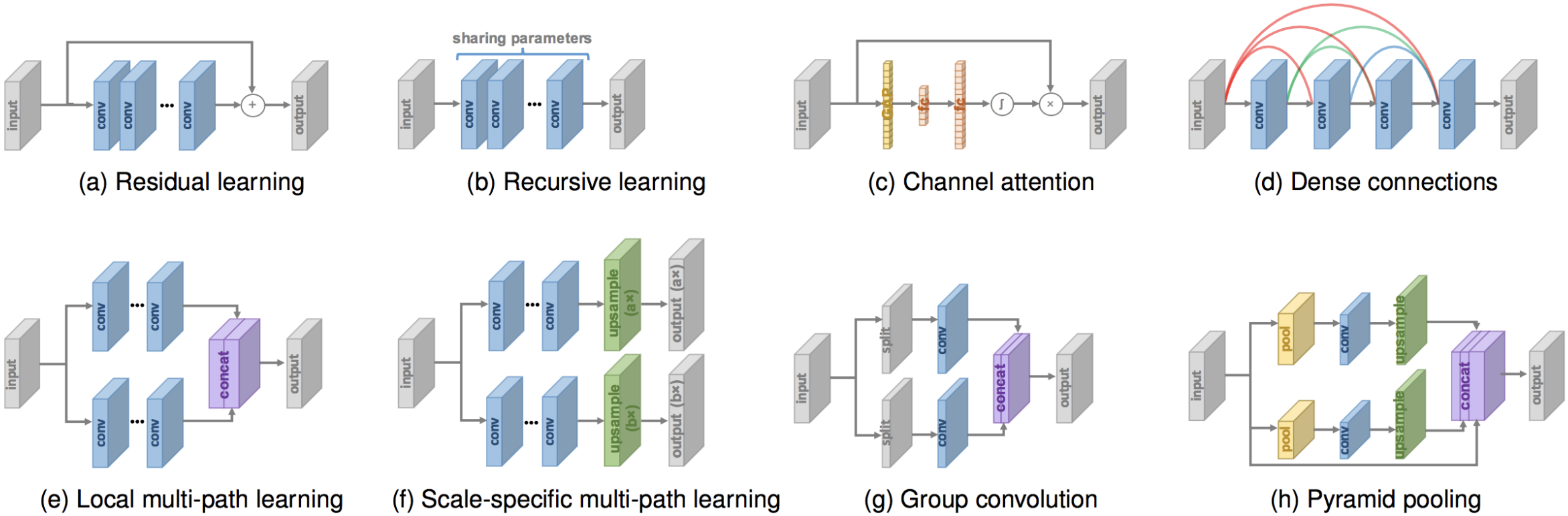


Fig. 5. Sub-pixel layer. The blue boxes denote the input, and the boxes with other colors indicate different convolution operations and different output feature maps.

Compared with transposed convolution layer, the greatest advantage of sub-pixel layer is the larger receptive field, which provides more contextual information to help generate more accurate details. Nevertheless, the distribution of the receptive fields of sub-pixel layers is uneven, blocky regions actually share the same receptive field, which may result in some artefacts near the boundaries of different blocks.

图像增强 -- SR



(a) Residual learning (b) Recursive learning (c) Channel attention (d) Dense connections

(e) Local multi-path learning (f) Scale-specific multi-path learning (g) Group convolution (h) Pyramid pooling

Fig. 6. Network design strategies.

图像增强 -- SR

- LOSS FUNCTIONS

Pixel Loss. Pixel loss measures pixel-wise difference between two images and mainly includes L1 loss (i.e., mean absolute error) and L2 loss (i.e., mean square error):

$$\mathcal{L}_{\text{pixel\_l1}}(\hat{I}, I) = \frac{1}{hwc} \sum_{i,j,k} |\hat{I}_{i,j,k} - I_{i,j,k}|, \tag{16}$$

$$\mathcal{L}_{\text{pixel\_l2}}(\hat{I}, I) = \frac{1}{hwc} \sum_{i,j,k} (\hat{I}_{i,j,k} - I_{i,j,k})^2, \tag{17}$$

where h, w and c are the height, width and number of channels of the evaluated images, respectively. In addition, there is a variant of the pixel L1 loss, namely Charbonnier loss [29], [120], given by:.

$$\mathcal{L}_{\text{pixel\_Cha}}(\hat{I}, I) = \frac{1}{hwc} \sum_{i,j,k} \sqrt{(\hat{I}_{i,j,k} - I_{i,j,k})^2 + \epsilon^2}, \tag{18}$$

where ϵ is a small constant (e.g., $1e-3$) for numerical stability.

图像增强 -- SR

- LOSS FUNCTIONS

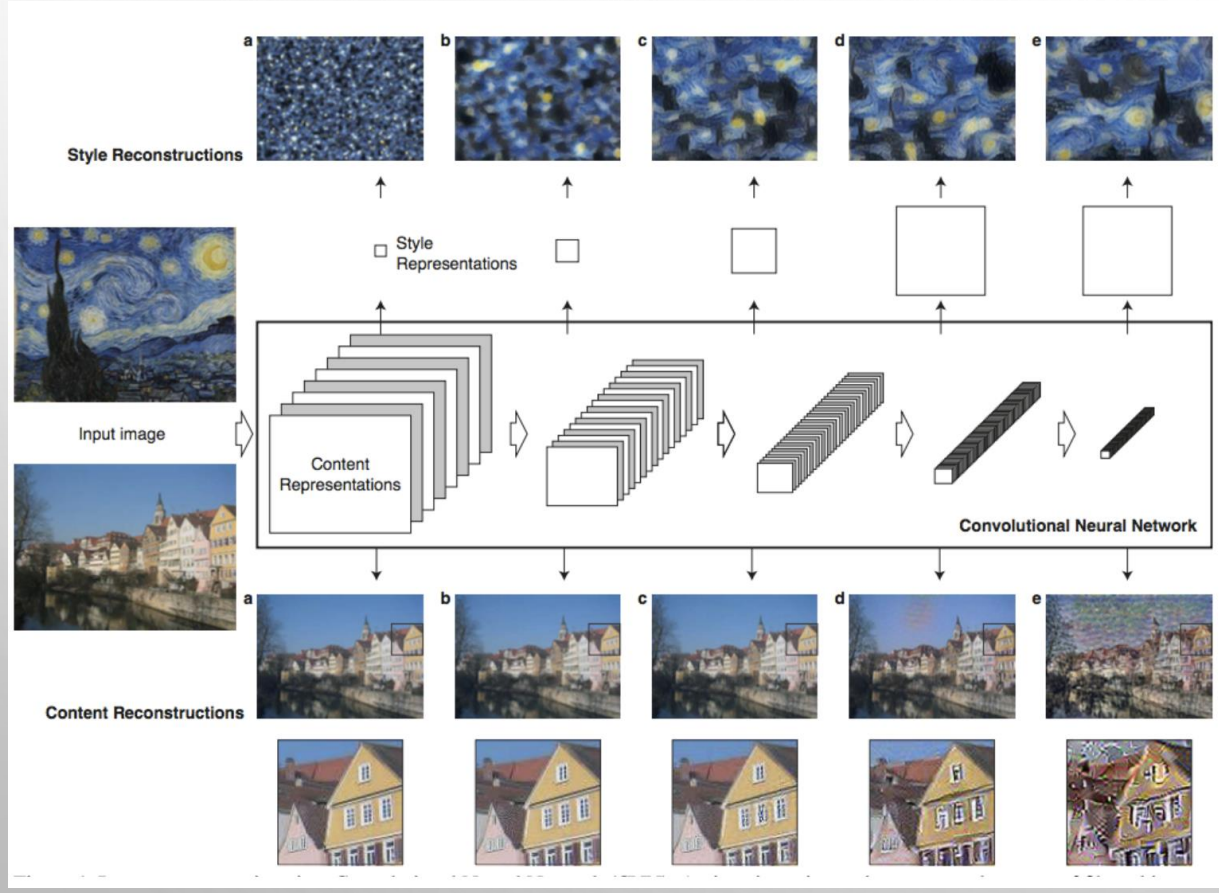
Essentially the content loss transfers the learned knowledge of hierarchical image features from the classification network ϕ to the SR network. In contrast to the pixel loss, the content loss encourages the output image $\hat{I}$ to be perceptually similar to the target image I instead of forcing them to match pixels exactly. Thus it produces visually more perceptible results and is also widely used in this field [10], [27], [31], [32], [48], [101], where the VGG [123] and ResNet [95] are the most commonly used pre-trained CNNs.

Content Loss. To evaluate image quality based on the perceptual quality, the content loss is introduced into super-resolution [31], [122]. Specifically, it measures the semantic differences between images using a pre-trained image classification network. Denoting this network as ϕ and the extracted high-level representations on l-th layer as $\phi^{(l)}(I)$, the content loss is indicated as the Euclidean distance between high-level representations between two images, as follows:

$$\mathcal{L}_{\text{content}}(\hat{I}, I; \phi, l) = \frac{1}{h_l w_l c_l} \sqrt{\sum_{i,j,k} (\phi^{(l)}_{i,j,k}(\hat{I}) - \phi^{(l)}_{i,j,k}(I))^2}, \tag{19}$$

where h_l, w_l and c_l are the height, width and number of channels of the extracted feature maps on layer l, respectively.

图像增强 -- SR



Texture Loss. On account that the reconstructed image should have the same style (e.g., colors, textures, contrast) with the target image, and motivated by the style representation by Gatys *et al.* [124], [125], the texture loss (a.k.a. style reconstruction loss) is introduced into super-resolution. Following [124], [125], the texture of an image is regarded as the correlations between different feature channels and defined as the Gram matrix $G^{(l)} \in \mathcal{R}^{c_l \times c_l}$, where $G^{(l)}_{ij}$ is the inner product between the vectorized feature maps i and j on layer l:

$$G^{(l)}_{ij}(I) = \text{vec}(\phi^{(l)}_i(I)) \cdot \text{vec}(\phi^{(l)}_j(I)), \tag{20}$$

where $\text{vec}(\cdot)$ denotes a vectorization operation, and $\phi^{(l)}_i(I)$ represents the i-th channel of the feature maps on layer l of image I. Based on the above definitions, the texture loss is given by:

$$\mathcal{L}_{\text{texture}}(\hat{I}, I; \phi, l) = \frac{1}{c_l^2} \sqrt{\sum_{i,j} (G^{(l)}_{i,j}(\hat{I}) - G^{(l)}_{i,j}(I))^2}. \tag{21}$$

feature_map的大小为$C_iH_iW_i$，可以看成是C_i个特征，这些特征两两之间的内积的计算方式如上

STYLE2PAINTS

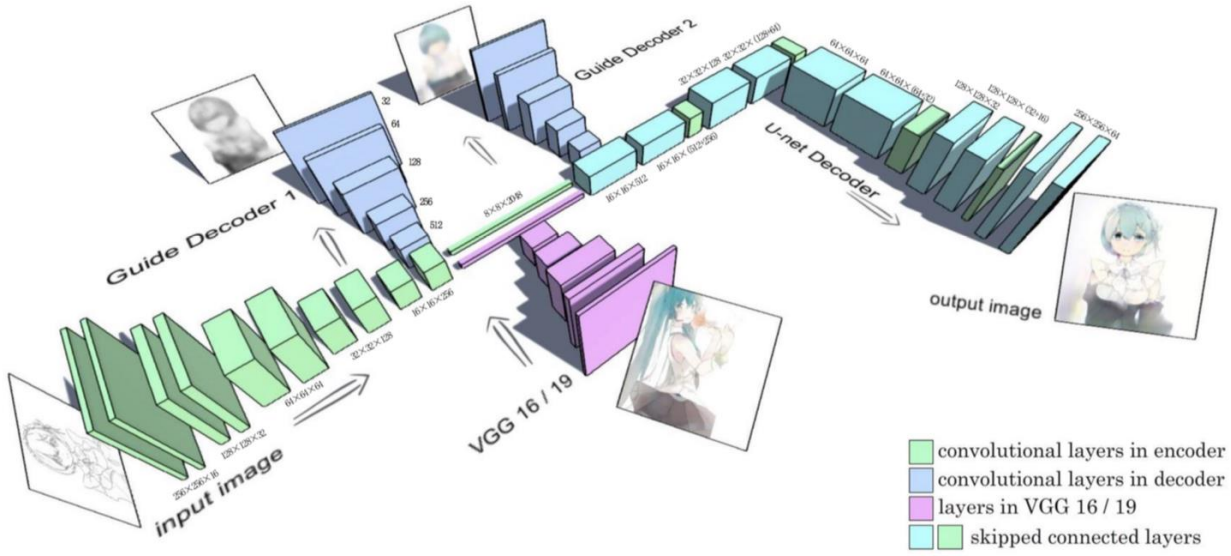
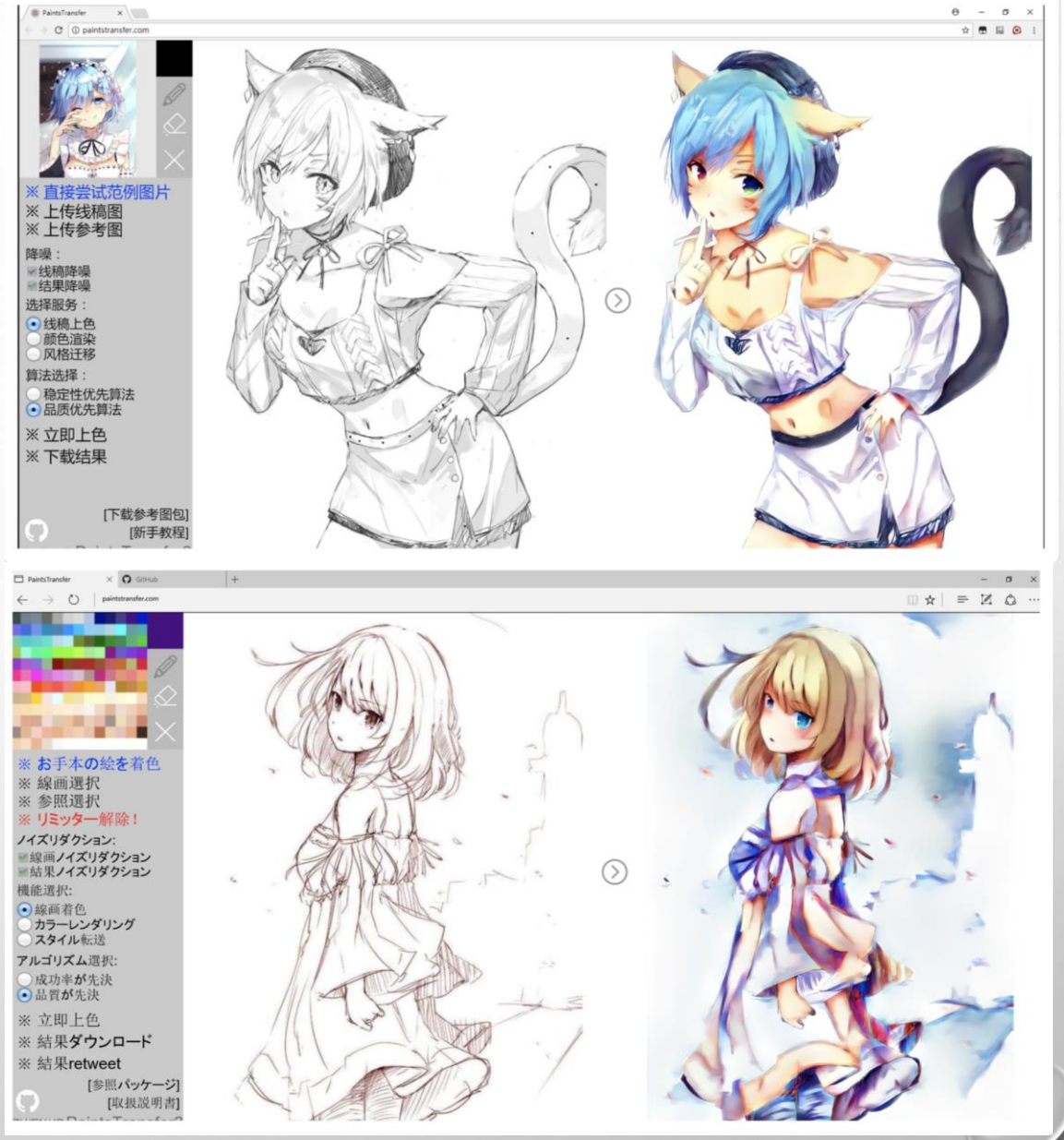
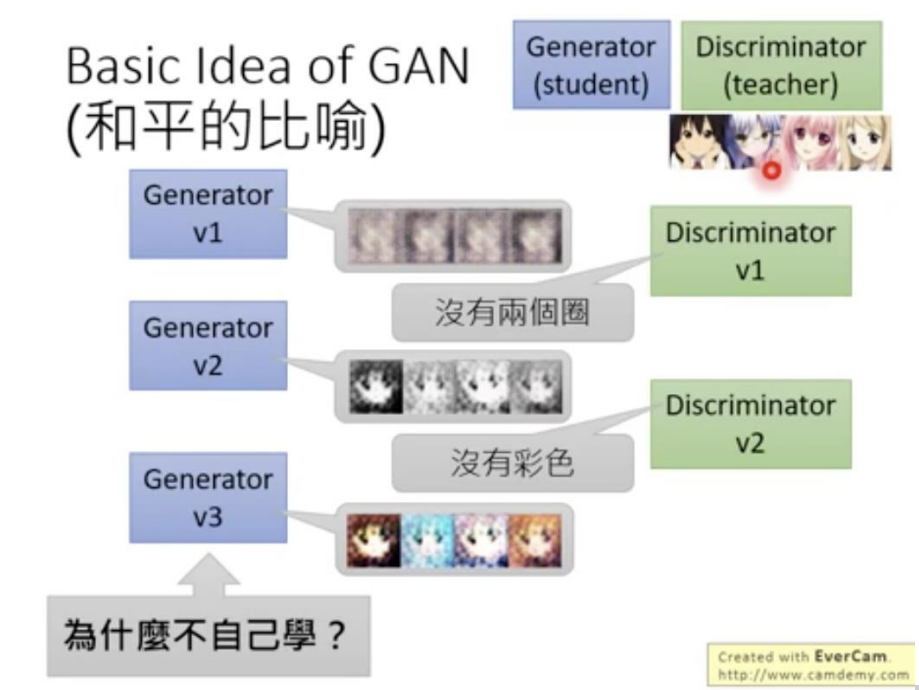


Figure 2: The architecture of the generator. The generator is based on U-net with skip connected layers. The weights of the VGG 16 or 19 [15] is locked in order to avoid being disturbed while training. The two "Guide Decoder" are located at the entry and exit of mid-level layers. The 4096 outputs of VGG's *fc1* are regarded as global style hint and added to the mid-level layers after a global normalization. For better performance in training, we add a dense 2048 layer above the *fc1*.



图像增强 -- SR

- LOSS FUNCTIONS

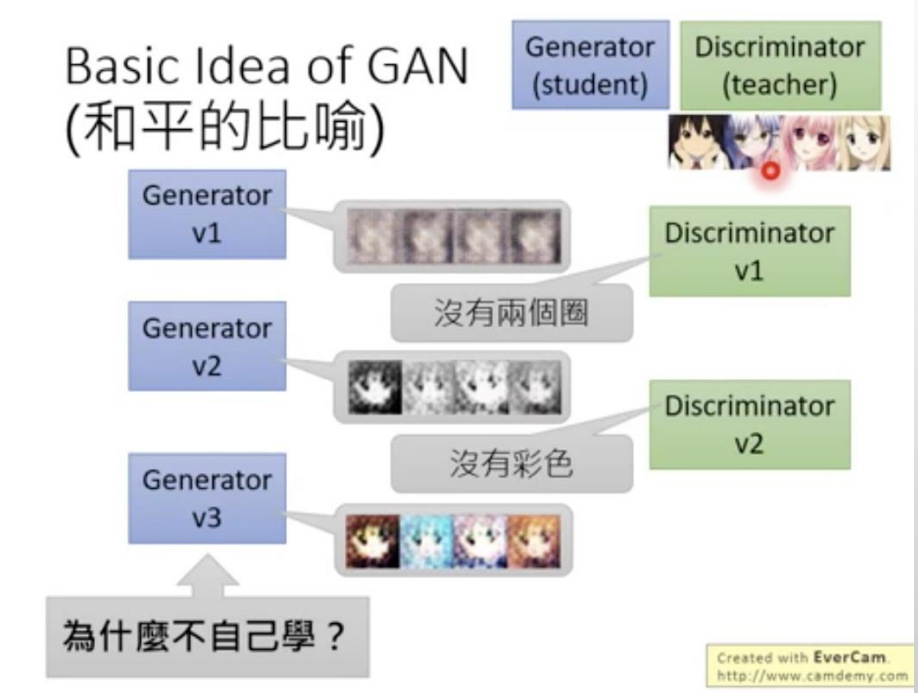


Adversarial Loss. In recent years, the GANs [26] have been more and more popular and introduced to various vision tasks. To be concrete, the GAN consists of a generator performing generation (e.g., text generation, image transformation), and a discriminator which takes the generated output and instances sampled from the target distribution as input and discriminates whether each input comes from the target distribution. During training, two steps are alternately performed: (a) fix the generator and train the discriminator to better discriminate, (b) fix the discriminator and train the generator to fool the discriminator. Through iterative adversarial training and after the model eventually converges, the resulting generator can produce outputs consistent with the distribution of real data, while the discriminator can't distinguish between the generated data and real data.

https://www.bilibili.com/video/av24011528

图像增强 -- SR

- LOSS FUNCTIONS

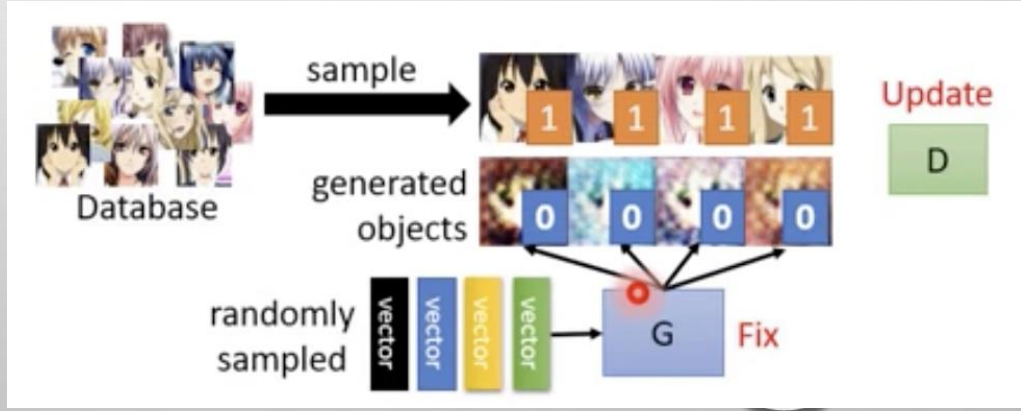


In the super-resolution field, it is straightforward to adopt adversarial learning, in which case we only need to treat the SR model as a generator, and additionally define a discriminator to judge whether the input image is generated or not. Ledig *et al.* [27] firstly introduce SRGAN using adversarial loss based on cross entropy, as follows:

$$\mathcal{L}_{\text{gan\_ce\_g}}(\hat{I}; D) = -\log D(\hat{I}), \tag{22}$$

$$\mathcal{L}_{\text{gan\_ce\_d}}(\hat{I}, I_s; D) = -\log D(I_s) - \log(1 - D(\hat{I})), \tag{23}$$

where $\mathcal{L}_{\text{gan\_ce\_g}}$ and $\mathcal{L}_{\text{gan\_ce\_d}}$ denote the adversarial loss of the generator (i.e., the SR model) and the discriminator D (i.e., a binary classifier), respectively. I_s represents randomly sampled data from ground truth HR images.



图像增强 -- SR

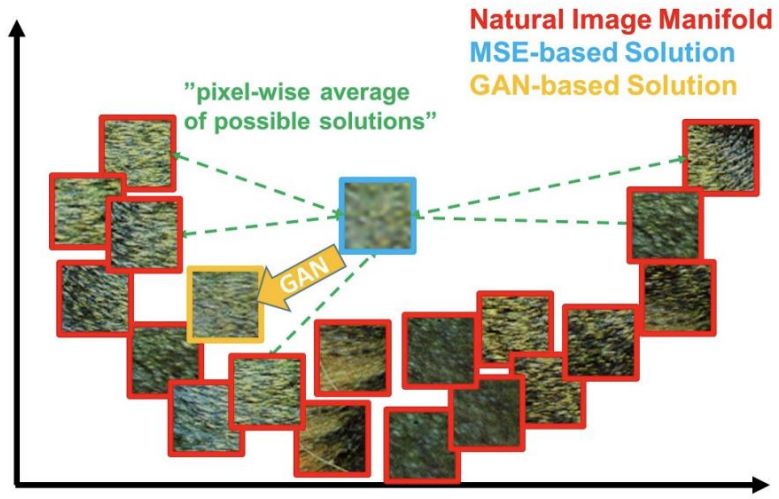


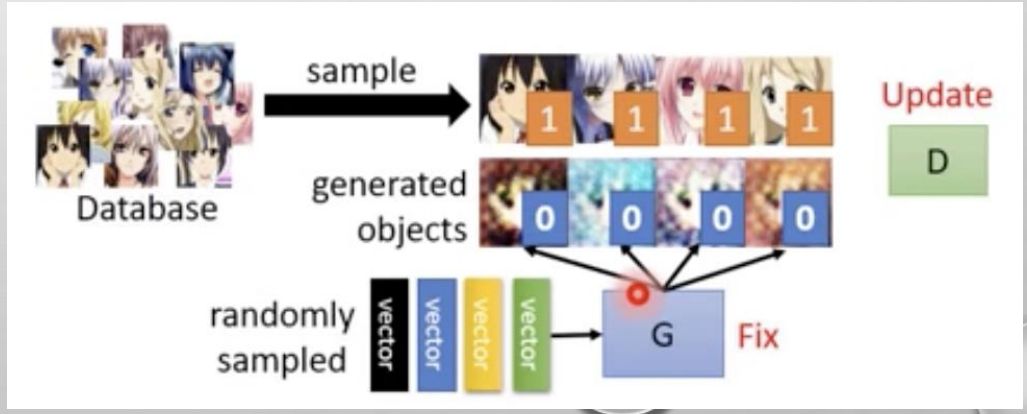
Figure 3: Illustration of patches from the natural image manifold (red) and super-resolved patches obtained with MSE (blue) and GAN (orange). The MSE-based solution appears overly smooth due to the pixel-wise average of possible solutions in the pixel space, while GAN drives the reconstruction towards the natural image manifold producing perceptually more convincing solutions.

In the super-resolution field, it is straightforward to adopt adversarial learning, in which case we only need to treat the SR model as a generator, and additionally define a discriminator to judge whether the input image is generated or not. Ledig *et al.* [27] firstly introduce SRGAN using adversarial loss based on cross entropy, as follows:

$$\mathcal{L}_{\text{gan\_ce\_g}}(\hat{I}; D) = -\log D(\hat{I}), \tag{22}$$

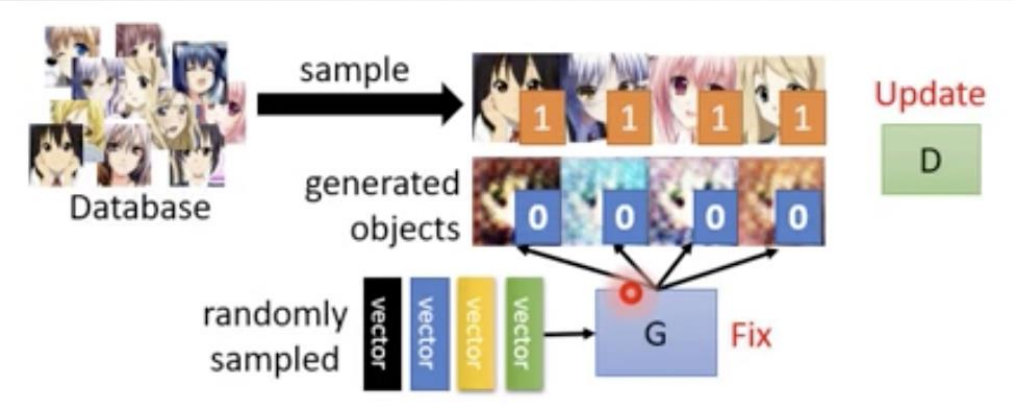
$$\mathcal{L}_{\text{gan\_ce\_d}}(\hat{I}, I_s; D) = -\log D(I_s) - \log(1 - D(\hat{I})), \tag{23}$$

where $\mathcal{L}_{\text{gan\_ce\_g}}$ and $\mathcal{L}_{\text{gan\_ce\_d}}$ denote the adversarial loss of the generator (i.e., the SR model) and the discriminator D (i.e., a binary classifier), respectively. I_s represents randomly sampled data from ground truth HR images.



图像增强 -- SR

- LOSS FUNCTIONS



In the super-resolution field, it is straightforward to adopt adversarial learning, in which case we only need to treat the SR model as a generator, and additionally define a discriminator to judge whether the input image is generated or not. Ledig *et al.* [27] firstly introduce SRGAN using adversarial loss based on cross entropy, as follows:

$$\mathcal{L}_{\text{gan\_ce\_g}}(\hat{I}; D) = -\log D(\hat{I}), \tag{22}$$

$$\mathcal{L}_{\text{gan\_ce\_d}}(\hat{I}, I_s; D) = -\log D(I_s) - \log(1 - D(\hat{I})), \tag{23}$$

Furthermore, Wang *et al.* [34] and Yuan *et al.* [126] use adversarial loss based on least square error for more stable training process and higher quality results [127], given by:

$$\mathcal{L}_{\text{gan\_ls\_g}}(\hat{I}; D) = (D(\hat{I}) - 1)^2, \tag{24}$$

$$\mathcal{L}_{\text{gan\_ls\_d}}(\hat{I}, I_s; D) = (D(\hat{I}))^2 + (D(I_s) - 1)^2. \tag{25}$$

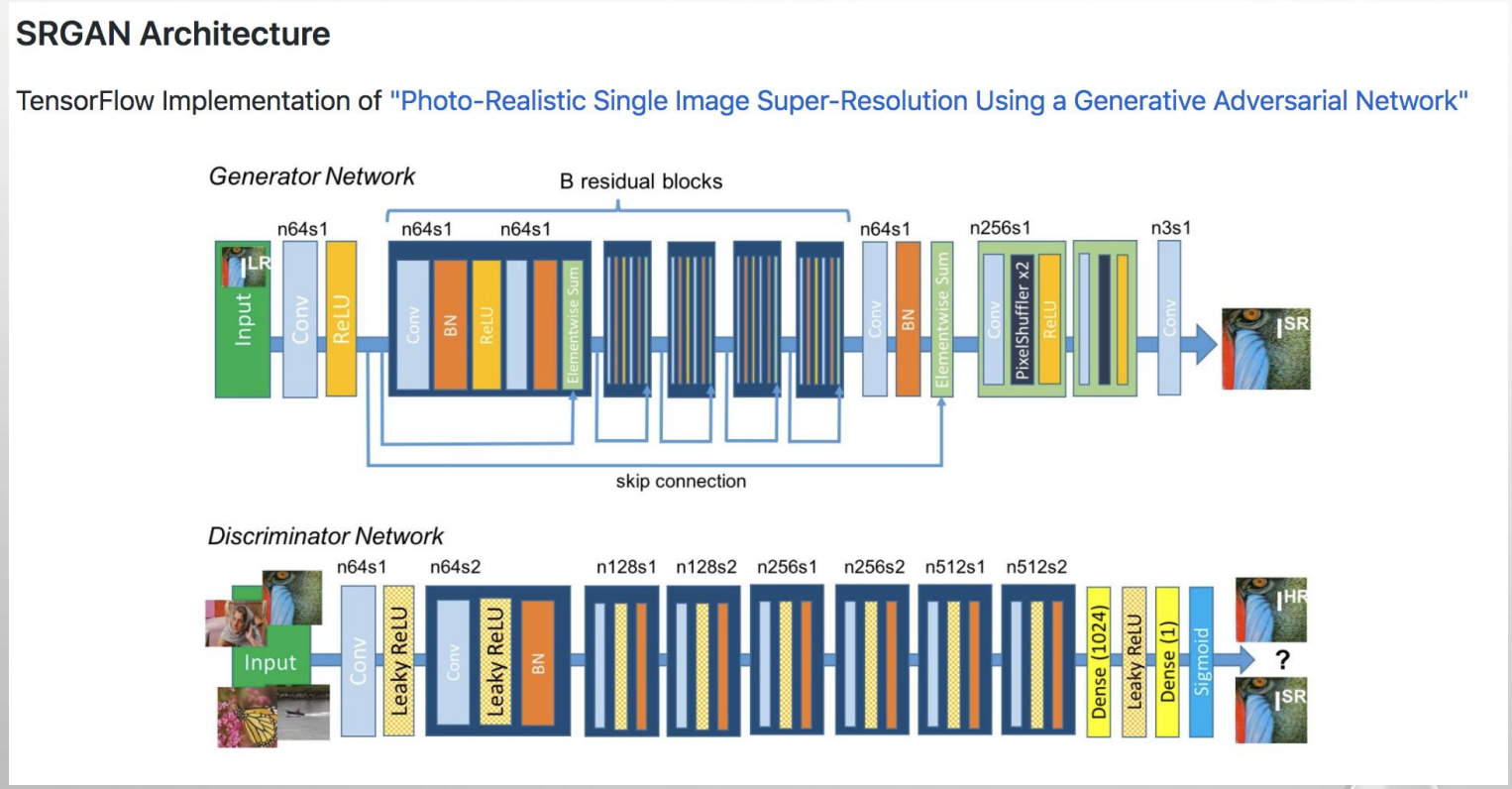
And Bulat *et al.* [128] adopt the hinge-format adversarial loss [129], as follows:

$$\mathcal{L}_{\text{gan\_hi\_g}}(\hat{I}; D) = -D(\hat{I}), \tag{26}$$

$$\mathcal{L}_{\text{gan\_hi\_d}}(\hat{I}, I_s; D) = \min(0, D(\hat{I}) - 1) + \min(0, -D(I_s) - 1).$$

SRGAN

- CODE: [HTTPS://GITHUB.COM/TENSORLAYER/SRGAN](https://github.com/tensorlayer/srgan)

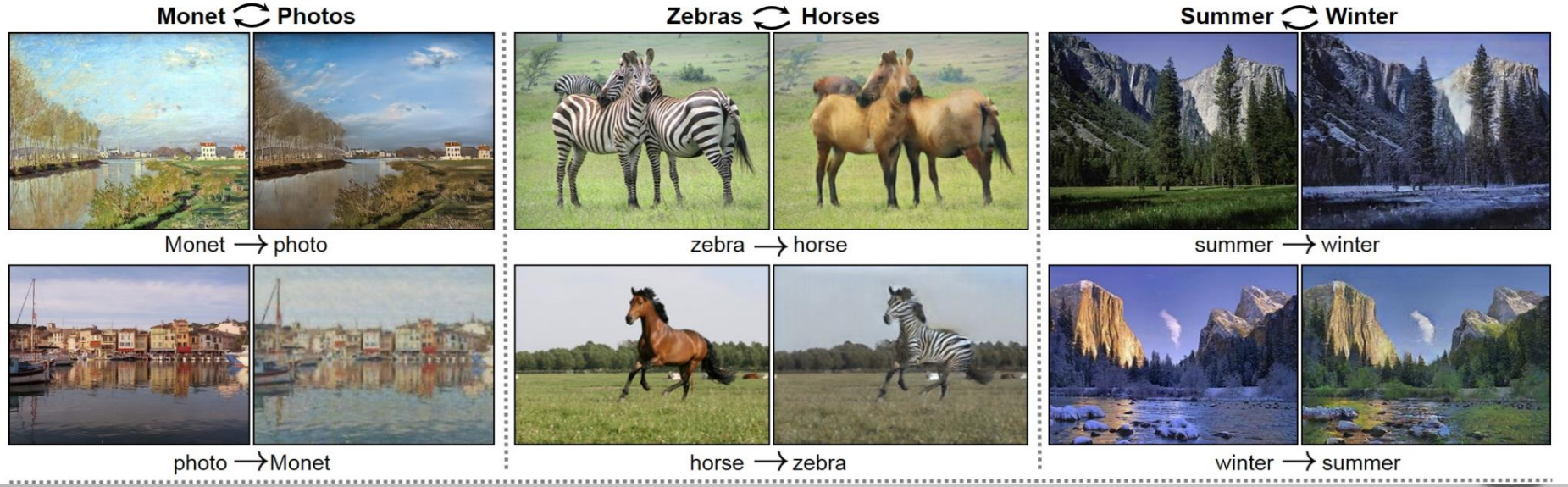


图像增强 -- SR

- LOSS FUNCTIONS

Cycle Consistency Loss. Motivated by the CycleGAN proposed by Zhu *et al.* [134] for image-to-image translation tasks, Yuan *et al.* [126] present a cycle-in-cycle approach for super-resolution. Concretely speaking, they not only super-resolve the LR image I to the HR image $\hat{I}$, but also downsample $\hat{I}$ back to another LR image I' through a CNN. The regenerated I' is required to be identical to the input I, thus the cycle consistency loss is introduced for constraining their pixel-level consistency:

$$\mathcal{L}_{\text{cycle}}(I', I) = \frac{1}{hwc}\sqrt{\sum_{i,j,k}(I'_{i,j,k} - I_{i,j,k})^2}. \tag{28}$$



图像增强 -- SR

- LOSS FUNCTIONS

Total Variation Loss. In order to suppress noise in generated images, the total variation (TV) loss [135] is introduced into the SR field by Aly *et al.* [136]. It is defined as the sum of the absolute differences between neighboring pixels and measures how much noise is in the images. For the generated HR image $\hat{I}$, the TV loss is define as:

$$\mathcal{L}_{\mathrm{TV}}(\hat{I}) = \frac{1}{hwc} \sum_{i,j,k} \sqrt{(\hat{I}_{i,j+1,k} - \hat{I}_{i,j,k})^2 + (\hat{I}_{i+1,j,k} - \hat{I}_{i,j,k})^2}. \tag{29}$$

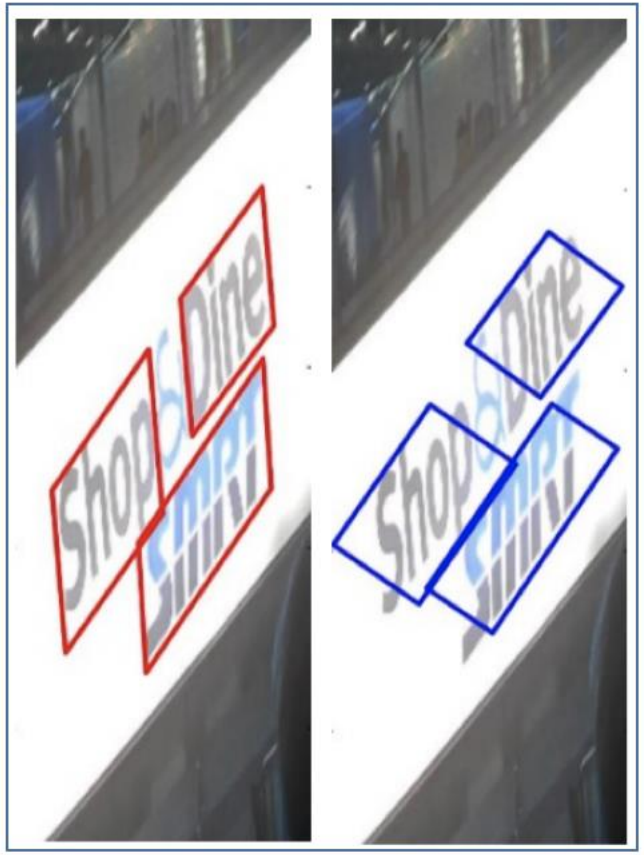
Lai *et al.* [27] and Yuan *et al.* [126] also adopt this TV loss for imposing spatial smoothness.

Prior-Based Loss. In addition to the above loss functions, external prior knowledge is also introduced to constrain the generation process. Bulat *et al.* [32] focus on face image SR and introduce a face alignment network (FAN) to constrain the consistency of facial landmarks detected from the original and generated images. The FAN is pre-trained and integrated for providing face alignment knowledge. In this way, the proposed Super-FAN improves performance both on low-resolution face alignment and face image super-resolution. As a matter of fact, the content loss and the texture loss, both of which introduce a classification network, essentially provide prior knowledge of hierarchical image features for SR. By introducing more prior knowledge, the performance of super-resolution can be further improved.

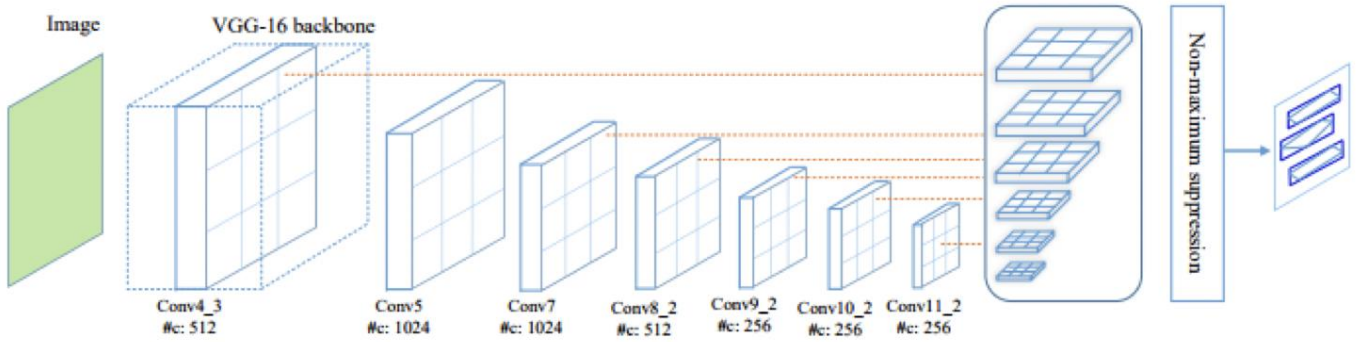
第六课 专题讨论

- 图像增加
  - UNWARPING
  - SUPER-RESOLUTION
- 自然场景文字检测（三）
  - PIXEL LINK
  - PSE NET

文本检测

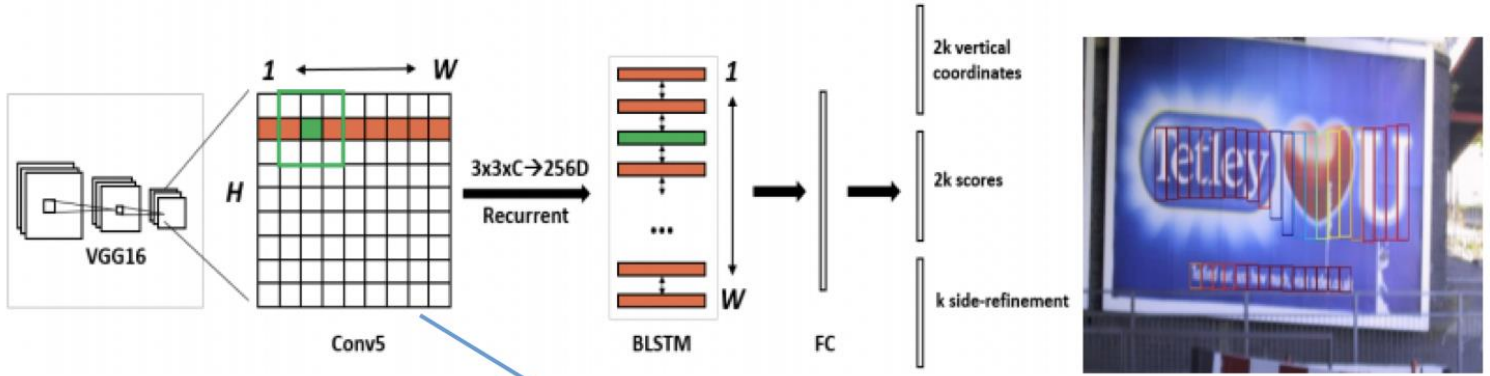


anchor-based检测算法 -> TextBoxes++ -> PIPELINE

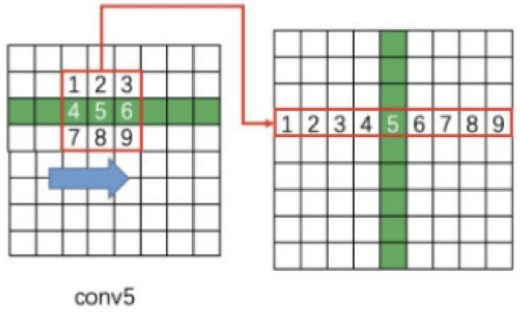


文本检测

anchor-based检测算法 -> CTPN -> PIPELINE

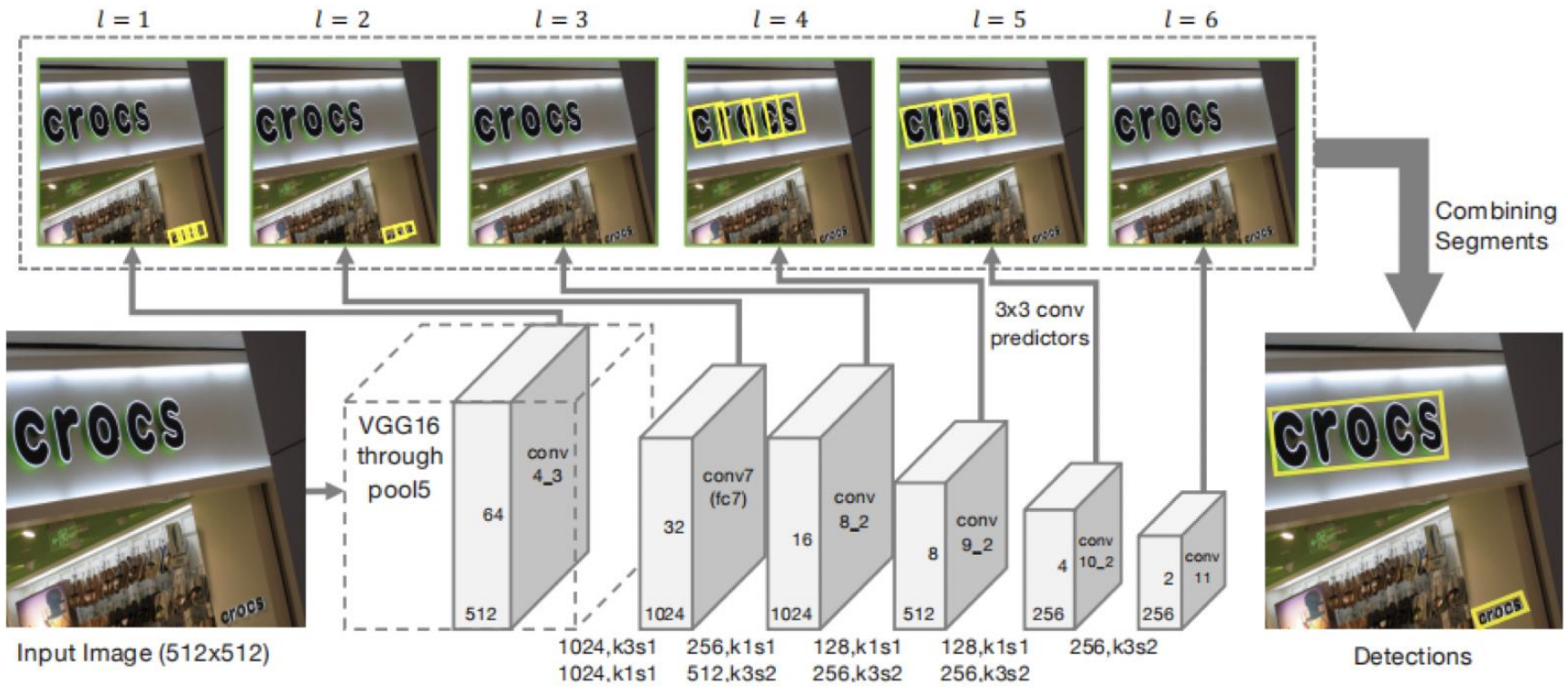


1. VGG16 -> Conv5 -> NxCxHxW
2. 3x3滑窗-> Nx9CxHxW->((NH)xWx9C)
3. Batch=NH、Tmax=W->LSTM->(NH)xWx256 ->Nx256xHxW(空间特征+序列特征)
4. Nx256xHxW->FC->Nx512xHxW->text proposals



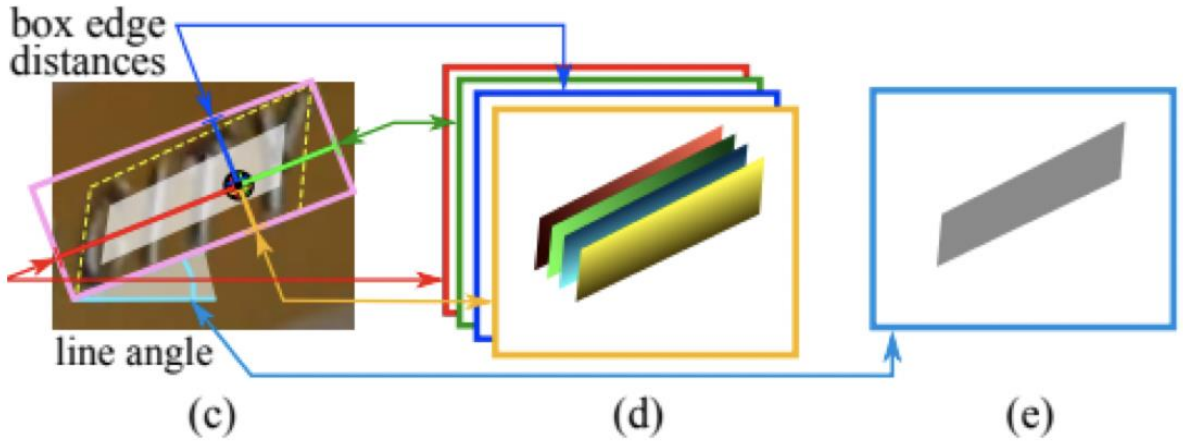
文本检测

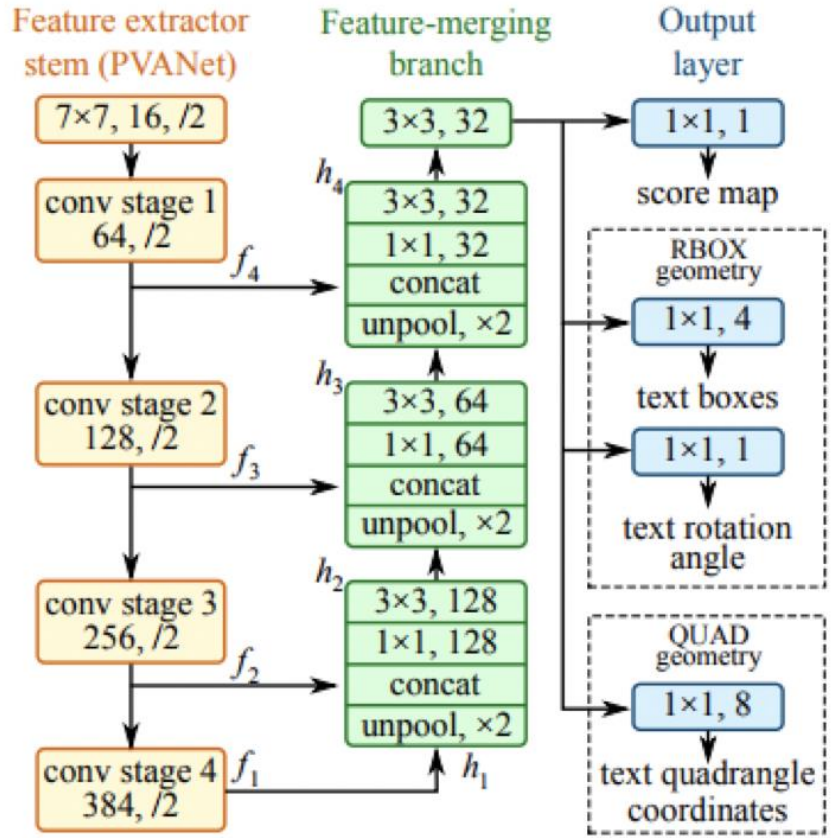
anchor-based检测算法 -> SEGLINK -> PIPELINE



文本检测

pixel-based检测算法 -> EAST -> PIPELINE





文本检测算法总结

类别	算法	优点	缺点	可用代码	实用性
pixel-based	EAST	结构简单，可处理多方向文本	无法处理过长文本	有	高(适合传统方法无法处理的场景)
	PSENet	可处理过长文本、曲线排列文本	复杂后处理	有但不完善	低(无成熟代码)
	FOTS	结合识别，性能优于EAST	无法处理过长文本	有但不完善	低(无成熟代码)
anchor-based	CTPN	可处理过长文本	仅能检测水平方向文本，复杂后处理	有	中(无法适应复杂场景)
	SegLink	可处理过长文本，多方向文本	较复杂后处理	有但不完善	低(无成熟代码)
	TextBoxes++	结构简单，可处理多方向文本	无法处理过长文本	有但不完善	低(无成熟代码)
pixel与anchor结合	pixel-anchor	可处理过长文本、小文本、多方向文本等	无法处理曲线排列文本	无	低(无成熟代码)
传统方法	形态学+联通域+文字特性	运行速度快、对硬件要求低，适合固定场景	无法处理复杂场景	有	中(适合处理无粘连的纯打印体)

PIXEL LINK

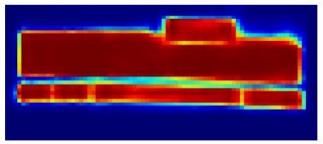
PixelLink: Detecting Scene Text via Instance Segmentation

Dan Deng[1,3*], **Haifeng Liu**[1], **Xuelong Li**[4], **Deng Cai**[1,2]
[1]State Key Lab of CAD&CG, College of Computer Science, Zhejiang University
[2]Alibaba-Zhejiang University Joint Institute of Frontier Technologies
[3]CVTE Research
[4]Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences
dengdan.zju@gmail.com {haifengliu,dcai}@zju.edu.cn xuelong_li@opt.ac.cn

Abstract

Most state-of-the-art scene text detection algorithms are deep learning based methods that depend on bounding box regression and perform at least two kinds of predictions: text/non-text classification and location regression. Regression plays a key role in the acquisition of bounding boxes in these methods, but it is not indispensable because text/non-text prediction can also be considered as a kind of semantic segmentation that contains full location information in itself. However, text instances in scene images often lie very close to each

original image



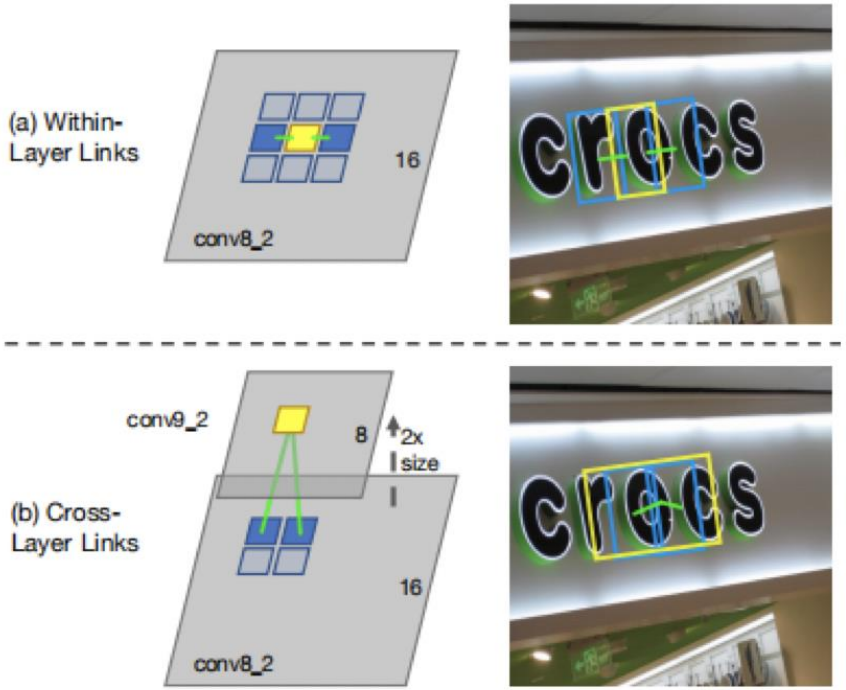
semantic segmentation

Figure 1: Text instances often lie close to each other, making them hard to separate via semantic segmentation.

- PAPER: HTTPS://ARXIV.ORG/ABS/1801.01315
- CODE: HTTPS://GITHUB.COM/ZJULEARNING/PIXEL_LINK

PIXEL LINK

anchor-based检测算法 -> SEGLINK -> Link检测



→ ＊ 同一个feature map层，每个feature map层仅预测一个segment，所以层内link仅考虑当前segment的8个领域，即判断每个segment与周围8个领域的连接情况(连接、断开)

→ ＊ 同一segment可能被不同feature map检测到，为了降低冗余，相邻feature map建立link，即cross-layer link，当前feature map的每个点与上一层feature map建立4个link(连接、断开)

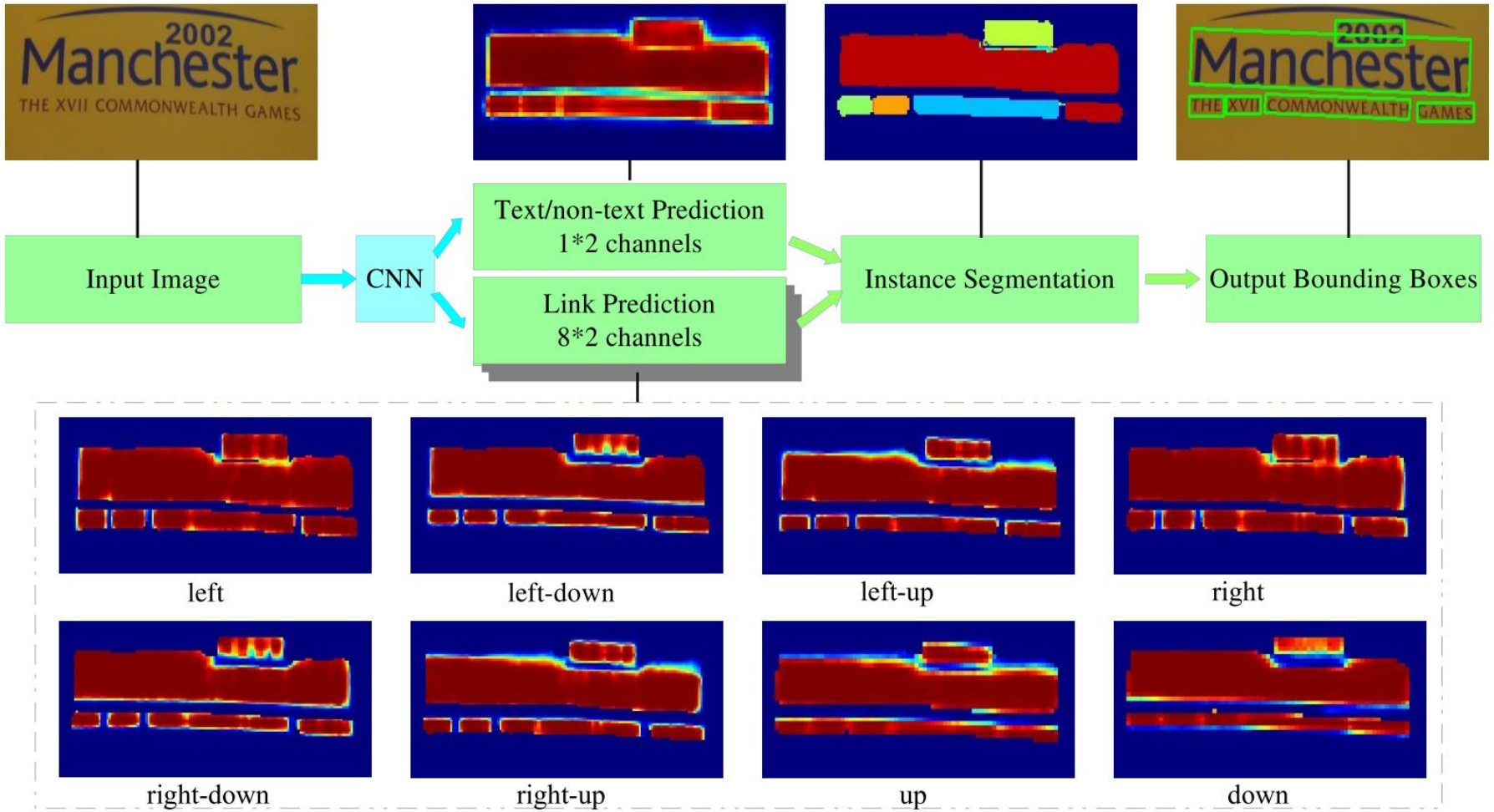


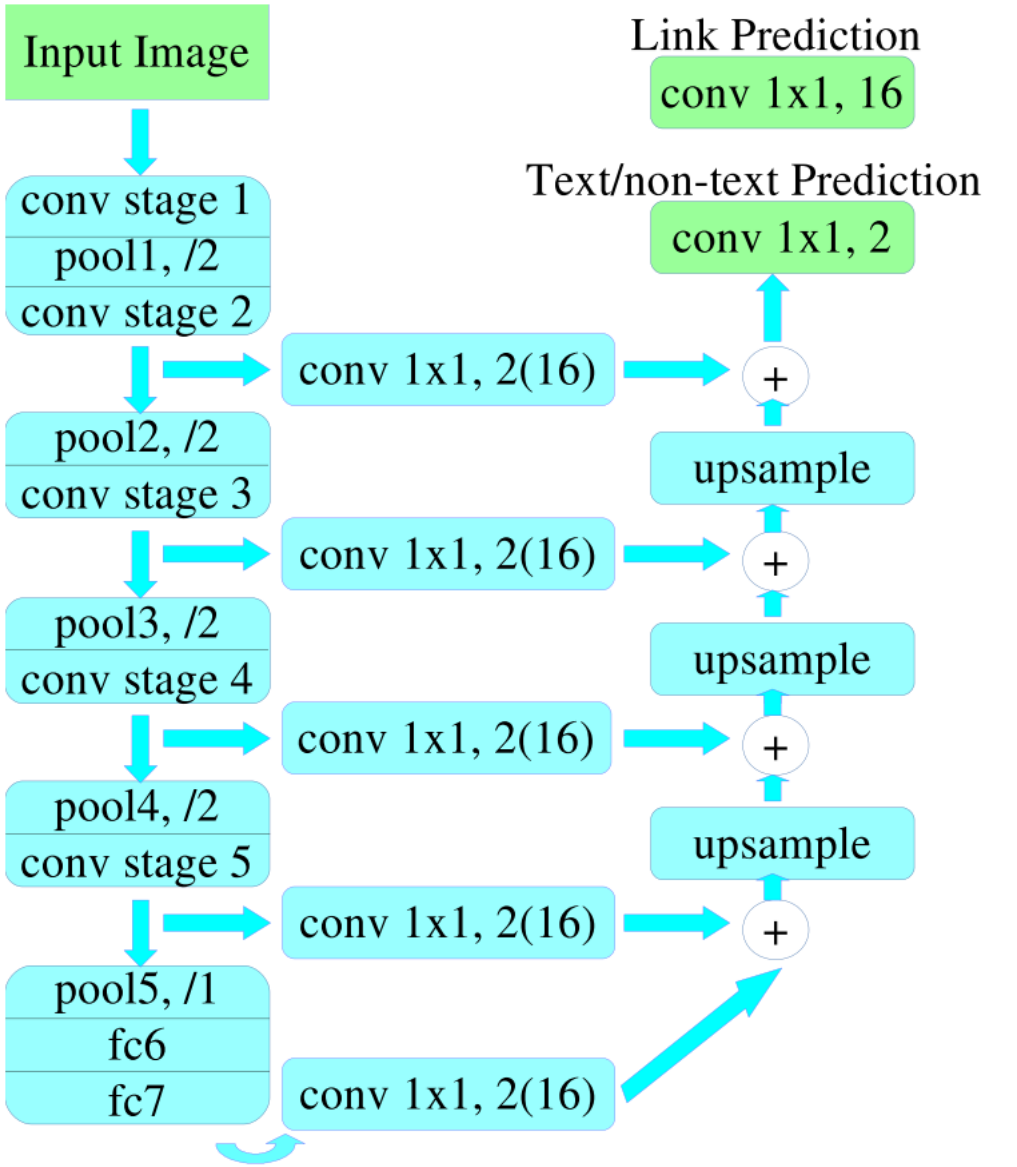
Figure 2: Architecture of PixelLink. A CNN model is trained to perform two kinds of pixel-wise predictions: text/non-text prediction and link prediction. After being thresholded, positive pixels are joined together by positive links, achieving instance segmentation. *minAreaRect* is then applied to extract bounding boxes directly from the segmentation result. Noise predictions can be efficiently removed using post-filtering. An input sample is shown for better illustration. The eight heat-maps in the dashed box stand for the link predictions in eight directions. Although some words are difficult to separate in text/non-text prediction, they are separable through link predictions.

PIXEL LINK

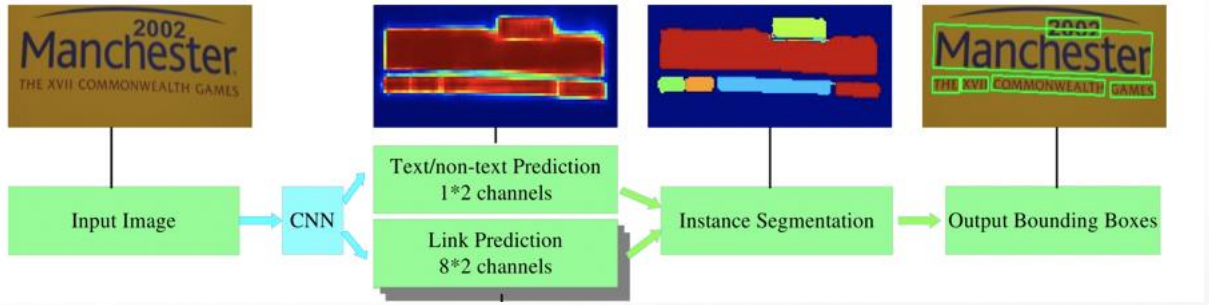
3.1 Network Architecture

Following SSD and SegLink, VGG16 (Simonyan and Zisserman 2014) is used as the feature extractor, with fully connected layers, *i.e.*, *fc6* and *fc7*, being converted into convolutional layers. The fashion of feature fusion and pixel-wise prediction inherits from (Long, Shelhamer, and Darrell 2015). As shown in Fig. 3, the whole model has two separate headers, one for text/non-text prediction, and the other for link prediction. Softmax is used in both, so their outputs have 1*2=2 and 8*2=16 channels, respectively.

Two settings of feature fusion layers are implemented: {*conv2_2, conv3_3, conv4_3, conv5_3, fc_7*}, and {*conv3_3, conv4_3, conv5_3, fc_7*}, denoted as **PixelLink+VGG16 2s**, and **PixelLink+VGG16 4s**, respectively. The resolution of 2s predictions is a half of the original image, and 4s is a quarter.



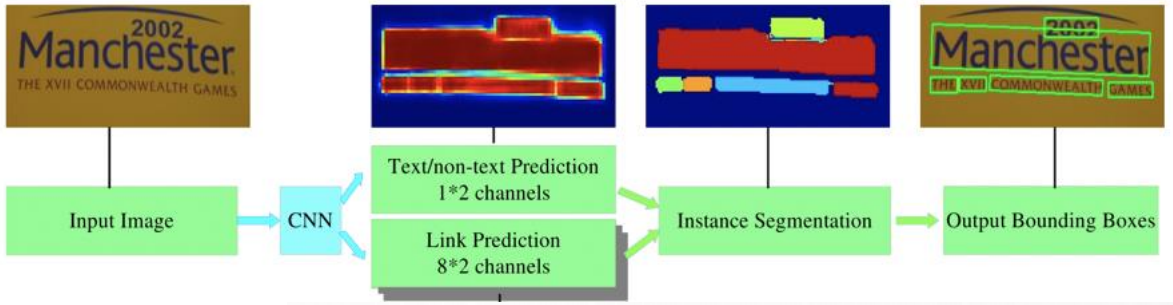
PIXEL LINK



3.2 Linking Pixels Together

Given predictions on pixels and links, two different thresholds can be applied on them separately. Positive pixels are then grouped together using positive links, resulting in a collection of CCs, each representing a detected text instance. Thus instance segmentation is achieved. It is worth noting that, given two neighboring positive pixels, their link are predicted by both of them, and they should be connected when one or both of the two link predictions are positive. This linking process can be implemented using disjoint-set data structure.

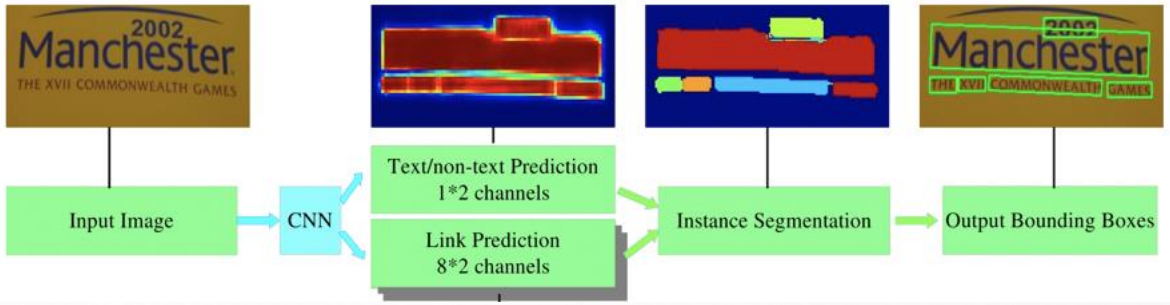
PIXEL LINK



3.4 Post Filtering after Segmentation

Since PixelLink attempts to group pixels together via links, it is inevitable to have some noise predictions, so a post-filtering step is necessary. A straightforward yet efficient solution is to filter via simple geometry features of detected boxes, *e.g.*, width, height, area and aspect ratio, *etc.* For example, in the IC15 experiments in Sec. 5.3, a detected box is abandoned if its shorter side is less than 10 pixels or if its area is smaller than 300. The 10 and 300 are statistical results on the training data of IC15. Specifically, for a chosen filtering criteria, the corresponding 99-th percentile calculated on **TRAINING** set is chosen as the threshold value. For example, again, 10 is chosen as the threshold on shorter side length because about 99% text instances in IC15-train have a shorter side $\geq$ 10 pixels.

PIXEL LINK



4 Optimization

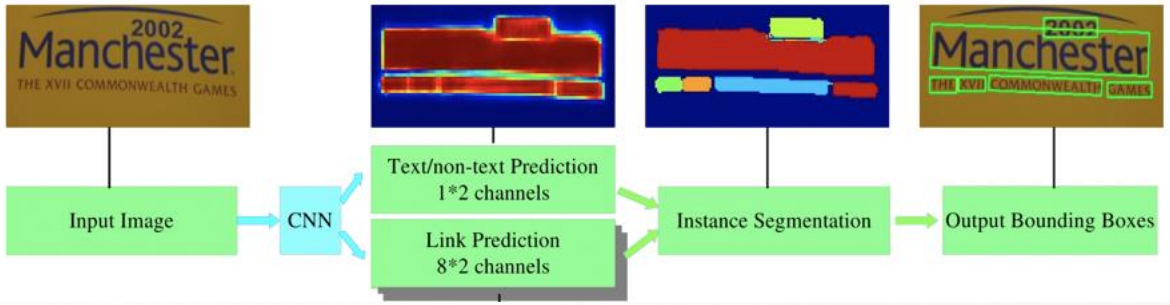
4.1 Ground Truth Calculation

Following the formulation in TextBlocks (Zhang et al. 2016), pixels inside text bounding boxes are labeled as positive. If overlapping exists, only un-overlapped pixels are positive. Otherwise negative.

For a given pixel and one of its eight neighbors, if they belong to the same instance, the link between them is positive. Otherwise negative.

Note that ground truth calculation is carried out on input images resized to the shape of prediction layer, *i.e.*, *conv3_3* for 4s and *conv2_2* for 2s.

PIXEL LINK



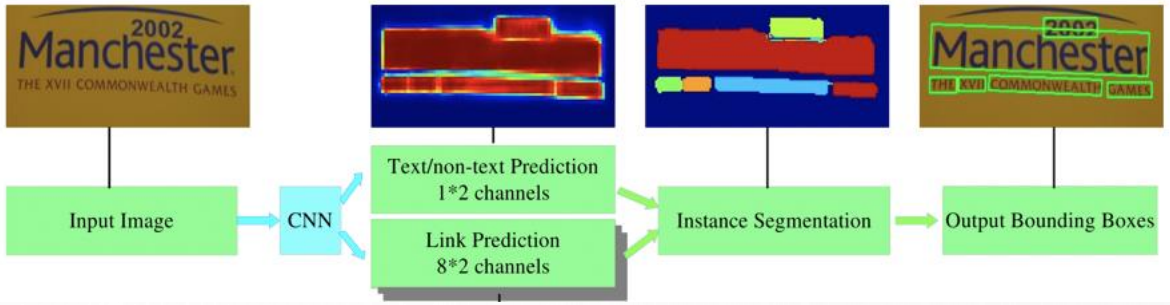
4.2 Loss Function

The training loss is a weighted sum of loss on pixels and loss on links:

$$L = \lambda L_{pixel} + L_{link}. \quad (1)$$

Since L_{link} is calculated on positive pixels only, the classification task of pixel is more important than that of link, and λ is set to 2.0 in all experiments.

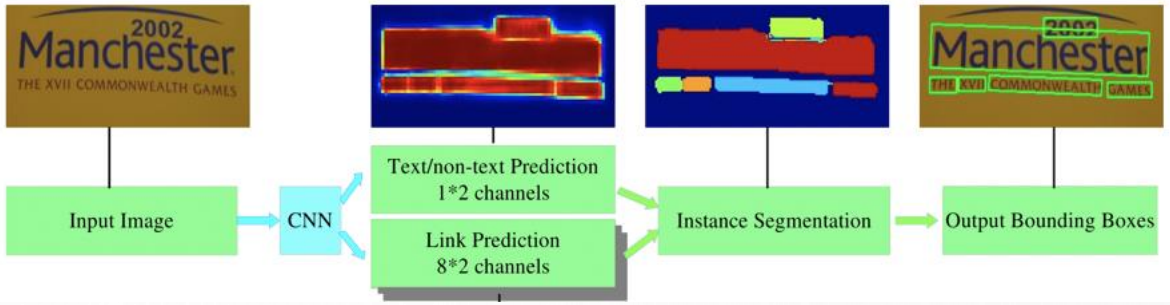
PIXEL LINK



Loss on Pixels Sizes of text instances might vary a lot. For example, in Fig. 1, the area of 'Manchester' is greater than the sum of all the other words. When calculating loss, if we put the same weight on all positive pixels, it's unfair to instances with small areas, and may hurt the performance. To deal with this problem, a novel weighted loss for segmentation, *Instance-Balanced Cross-Entropy Loss*, is proposed. In detail, for a given image with N text instances, all instances are treated equally by giving a same weight to everyone of them, denoted as B_i in Equ. 2. For the i-th instance with $area = S_i$, every positive pixels within it have a weight of $w_i = \frac{B_i}{S_i}$.

$$B_i = \frac{S}{N}, S = \sum_{i}^{N} S_i, \forall i \in \{1, \ldots, N\} \qquad (2)$$

PIXEL LINK



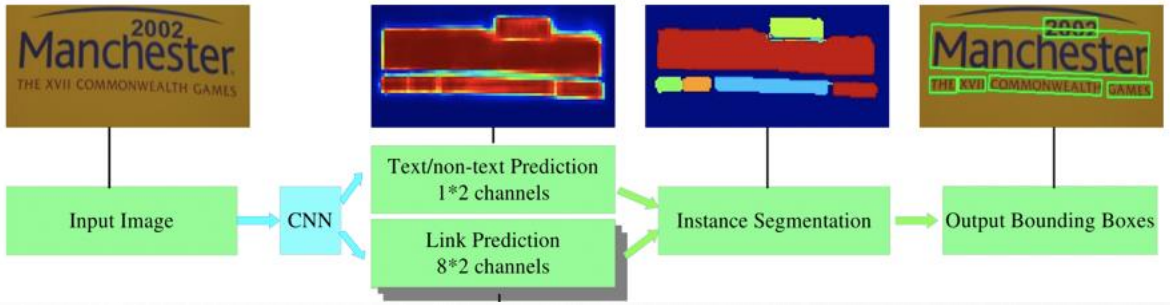
Online Hard Example Mining (OHEM) (Shrivastava, Gupta, and Girshick 2016) is applied to select negative pixels. More specifically, $r * S$ negative pixels with the highest losses are selected, by setting their weights to ones. r is the negative-positive ratio and is set to 3 as a common practice.

The above two mechanisms result in a weight matrix, denoted by W, for all positive pixels and selected negative ones. The loss on pixel classification task is:

$$L_{pixel} = \frac{1}{(1+r)S} W L_{pixel\_CE}, \tag{3}$$

where L_{pixel_CE} is the matrix of Cross-Entropy loss on text/non-text prediction.

PIXEL LINK



Loss on Links Losses for positive and negative links are calculated separately and on positive pixels only:

$$L_{link\_pos} = W_{pos\_link} L_{link\_CE},$$

$$L_{link\_neg} = W_{neg\_link} L_{link\_CE},$$

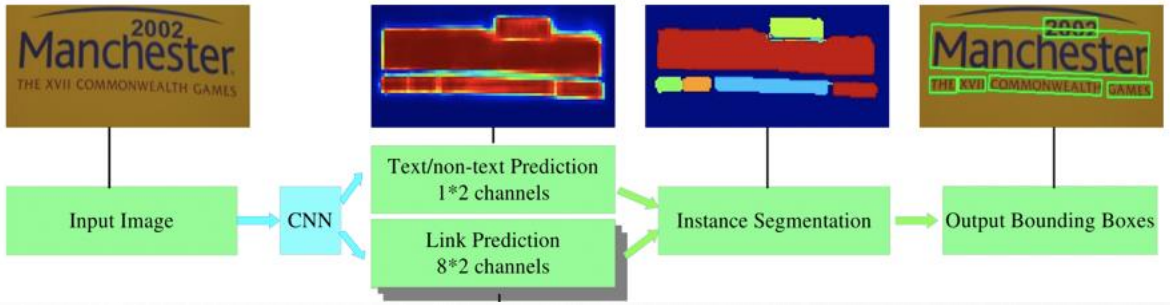
where L_{link_CE} is the Cross-Entropy loss matrix on link prediction. W_{pos_link} and W_{neg_link} are the weights of positive and negative links respectively. They are calculated from the W in Equ. 3. In detail, for the k-th neighbor of pixel (i, j):

$$W_{pos\_link}(i, j, k) = W(i, j) * (Y_{link}(i, j, k) == 1),$$

$$W_{neg\_link}(i, j, k) = W(i, j) * (Y_{link}(i, j, k) == 0),$$

where Y_{link} is the label matrix of links.

PIXEL LINK



Loss on Links Losses for positive and negative links are calculated separately and on positive pixels only:

$$L_{link\_pos} = W_{pos\_link} L_{link\_CE},$$

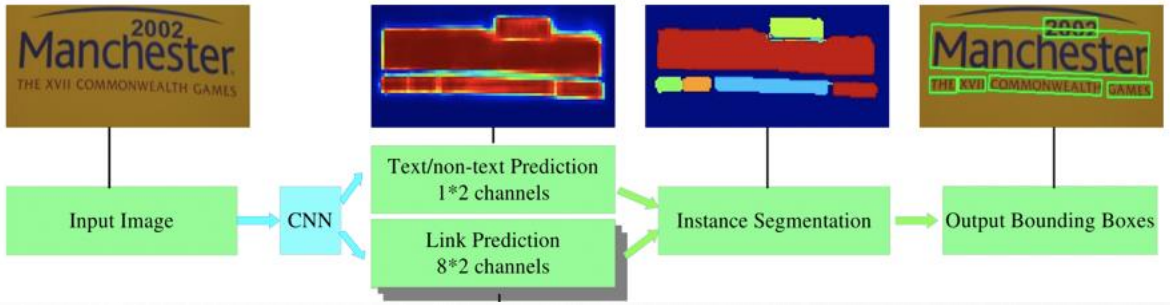
$$L_{link\_neg} = W_{neg\_link} L_{link\_CE},$$

The loss on link prediction is a kind of class-balanced cross-entropy loss:

$$L_{link} = \frac{L_{link\_pos}}{rsum(W_{pos\_link})} + \frac{L_{link\_neg}}{rsum(W_{neg\_link})}, \quad (4)$$

where $rsum$ denotes *reduce sum*, which sums all elements of a tensor into scalar.

PIXEL LINK



4.3 Data Augmentation

Data augmentation is done in a similar way to SSD with an additional random rotation step. Input images are firstly rotated at a probability of 0.2, by a random angle of $0, \pi/2, \pi$, or $3\pi/2$, the same with (He et al. 2017b). Then randomly crop them with areas ranging from 0.1 to 1, and aspect ratios ranging from 0.5 to 2. At last, resize them uniformly to 512×512. After augmentation, text instances with a shorter side less than 10 pixels are ignored. Text instances remaining less than 20% are also ignored. Weights for ignored instances are set to zero during loss calculation.

PIXEL LINK

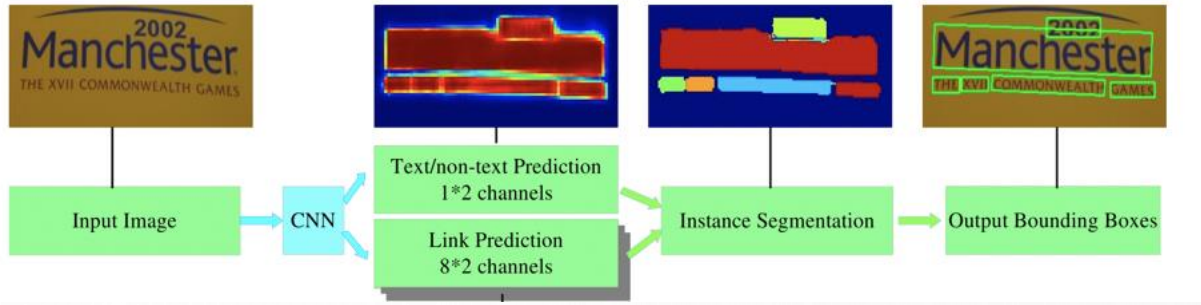


Figure 4: Examples of detection results. From left to right in columns: IC15, IC13, and MSRA-TD500.

Model	R	P	F	FPS
PixelLink+VGG16 2s	**82.0**	**85.5**	**83.7**	3.0
PixelLink+VGG16 4s	81.7	82.9	82.3	7.3
EAST+PVANET2x MS	78.3	83.3	81.0	—
EAST+PVANET2x	73.5	83.6	78.2	**13.2**
EAST+VGG16	72.8	80.5	76.4	6.5
SegLink+VGG16	76.8	73.1	75.0	—
CTPN+VGG16	51.6	74.2	60.9	7.1

PSE NET

Shape Robust Text Detection with Progressive Scale Expansion Network

Xiang Li[1]*, Wenhai Wang[21]*, Wenbo Hou[2], Ruo-Ze Liu[2], Tong Lu[2], Jian Yang[1]
[1]DeepInsight@PCALab, Nanjing University of Science and Technology
[2]National Key Lab for Novel Software Technology, Nanjing University

Abstract

The challenges of shape robust text detection lie in two aspects: 1) most existing quadrangular bounding box based detectors are difficult to locate texts with arbitrary shapes, which are hard to be enclosed perfectly in a rectangle; 2) most pixel-wise segmentation-based detectors may not separate the text instances that are very close to each other. To address these problems, we propose a novel Progressive Scale Expansion Network (PSENet), designed as a segmentation-based

- PAPER: HTTPS://ARXIV.ORG/PDF/1806.02559.PDF
- CODE: HTTPS://GITHUB.COM/LIUHENG92/TENSORFLOW_PSENET

PSE NET

(a) (b) (c) (d)

Figure 1: The results of different methods, best viewed in color. (a) is the original image. (b) refers to the result of bounding box regression-based method, which displays disappointing detections as the red box covers nearly more than half of the context in the green box. (c) is the result of semantic segmentation, which mistakes the 3 text instances for 1 instance since their boundary pixels are partially connected. (d) is the result of our proposed PSENet, which successfully distinguishs and detects the 4 unique text instances.

PSE NET

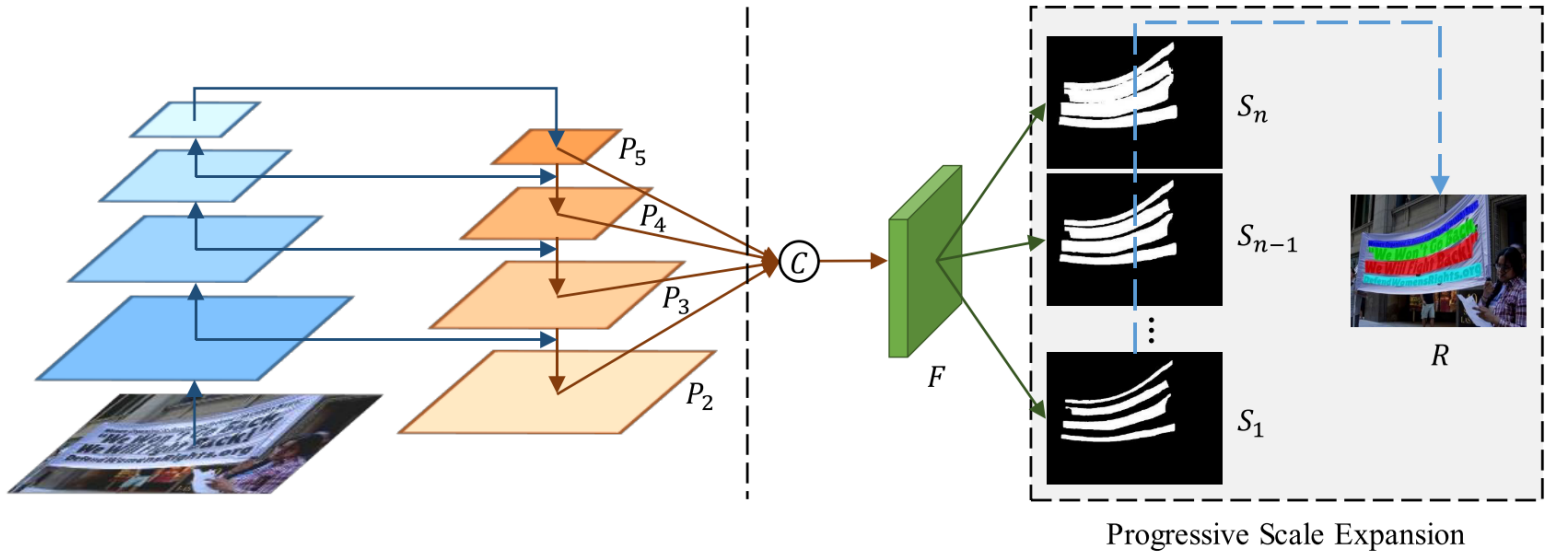


Figure 2: Illustration of our overall pipeline. The left part is implemented from FPN [16]. The right part denotes the feature fusion and the progressive scale expansion algorithm.

Firstly, as a segmentation-based method, PSENet is able to locate texts with arbitrary shapes. Secondly, we put forward a progressive scale expansion algorithm, with which the closely adjacent text instances can be identified successfully (see Fig. 1 (d)). Specifically, we assign each text instance with multiple predicted segmentation areas. For convenience, we denote these segmentation areas as "kernels" in this paper and for one text instance, there are several corresponding kernels. Each of the kernels shares the similar shape with the original entire text instance, and they all locate at the same central point but differ in scales. To obtain the final detections, we adopt the progressive scale expansion algorithm. It is based on Breadth-First-Search (BFS) and is composed of 3 steps: 1) starting from the kernels with minimal scales (instances can be distinguished in this step); 2) expanding their areas by involving more pixels in larger kernels gradually; 3) finishing until the largest kernels are explored.

PSE NET

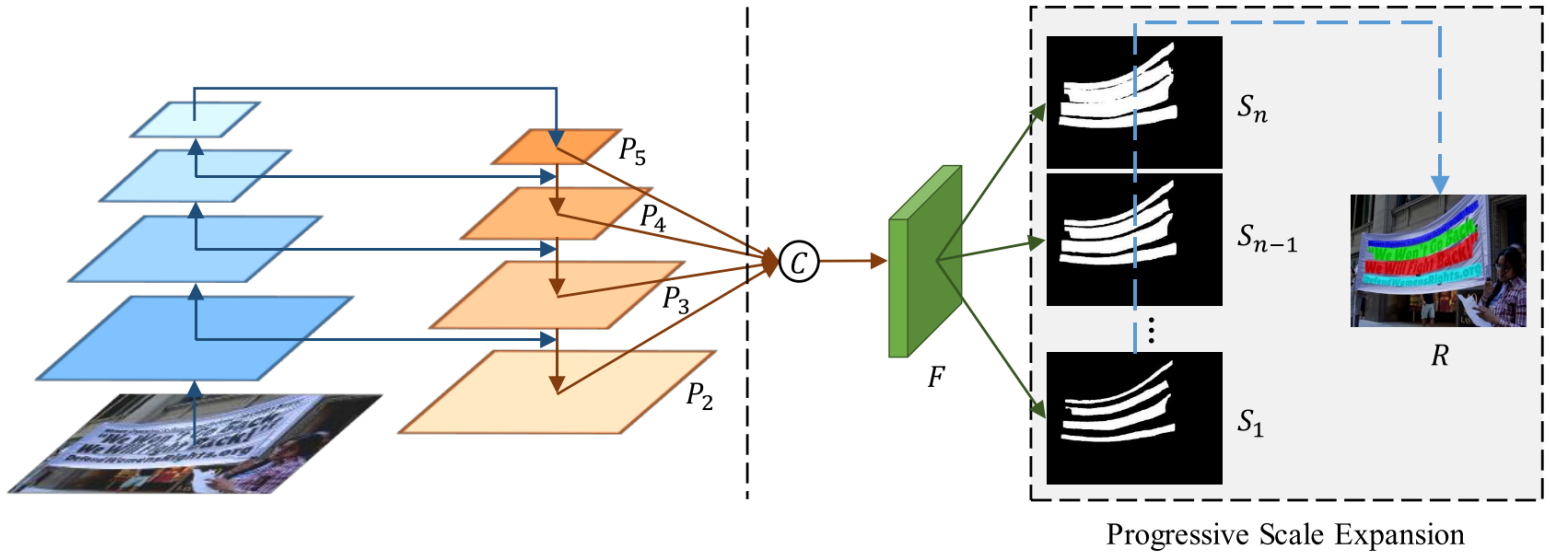


Figure 2: Illustration of our overall pipeline. The left part is implemented from FPN [16]. The right part denotes the feature fusion and the progressive scale expansion algorithm.

3.1 Overall Pipeline

The overall pipeline of the proposed PSENet is illustrated in Fig. 2. Inspired by FPN [16], we concatenate low-level feature maps with high-level feature maps and thus have four concatenated feature maps. These maps are further fused in F to encode informations with various receptive views. Intuitively, such fusion is very likely to facilitate the generations of the kernels with various scales. Then the feature map F is projected into n branches to produce multiple segmentation results $S_1, S_2, ..., S_n$. Each S_i would be one segmentation mask for all the text instances at a certain scale. The scales of different segmentation mask are decided by the hyper-parameters which will be discussed in Sec. 3.3. Among these masks, S_1 gives the segmentation result for the text instances with smallest scales (i.e., the minimal kernels) and S_n denotes for the original segmentation mask (i.e., the maximal kernels). After obtaining these segmentation masks, we use progressive scale expansion algorithm to gradually expand all the instances' kernels in S_1, to their complete shapes in S_n, and obtain the final detection results as R.

PSE NET

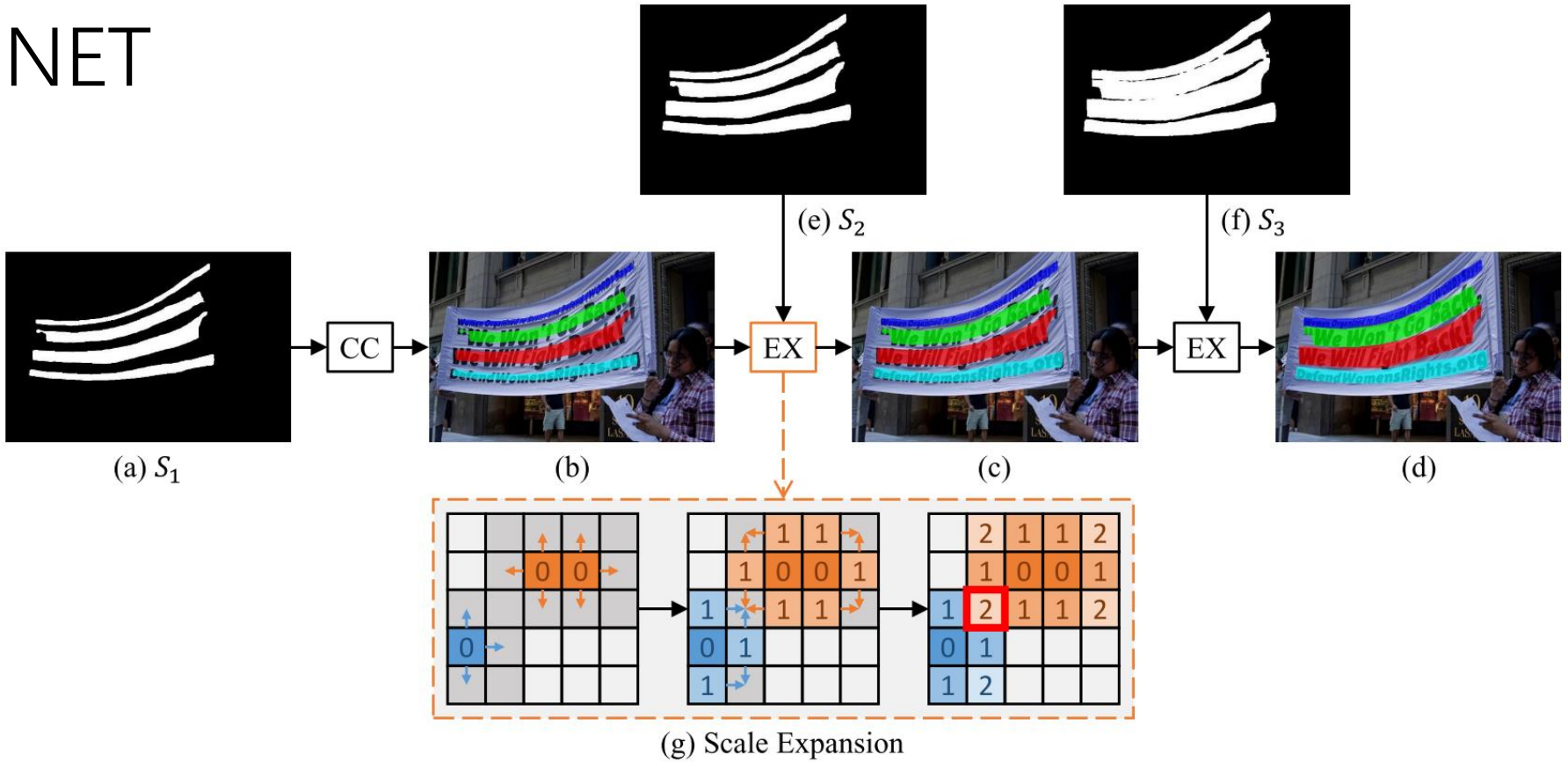


Figure 3: The procedure of progressive scale expansion algorithm. CC refers to the function of finding connected components. EX represents the scale expansion algorithm. (a), (e) and (f) refer to S_1, S_2 and S_3, respectively. (b) is the initial connected components. (c) and (d) is the results of expansion. (g) shows the illustration of expansion. The red box in (g) refers to the conflicted pixel.

PSE NET

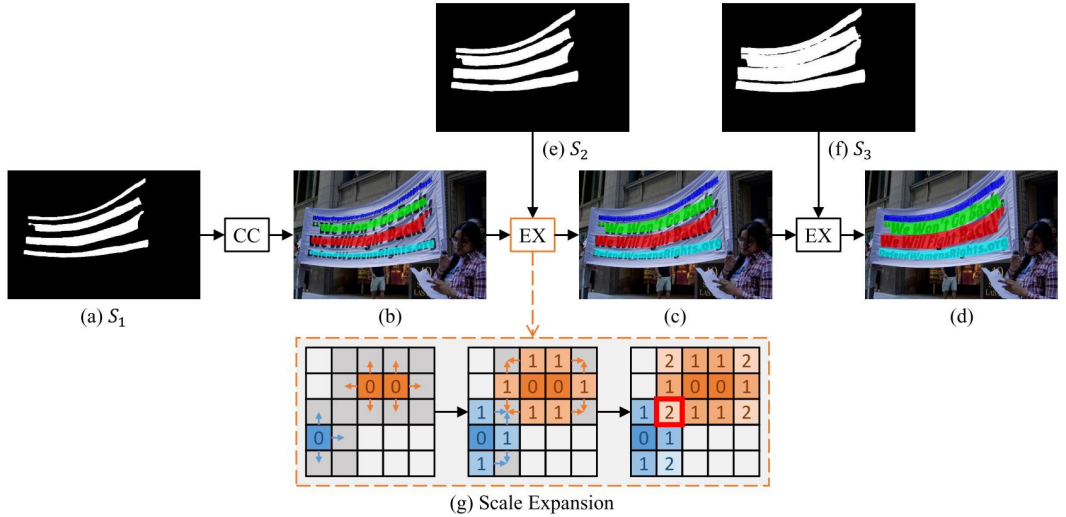


Figure 3: The procedure of progressive scale expansion algorithm. CC refers to the function of finding connected components. EX represents the scale expansion algorithm. (a), (e) and (f) refer to S_1, S_2 and S_3, respectively. (b) is the initial connected components. (c) and (d) is the results of expansion. (g) shows the illustration of expansion. The red box in (g) refers to the conflicted pixel.

Algorithm 1 Scale Expansion Algorithm

Require: Kernels: C, Segmentation Result: S_i
Ensure: Scale Expanded Kernels: E

function EXPANSION(C, S_i)
  $T \leftarrow \emptyset$; $P \leftarrow \emptyset$; $Q \leftarrow \emptyset$
  for each $c_i \in C$ **do**
    $T \leftarrow T \cup \{(p, label) \mid (p, label) \in c_i\}$; $P \leftarrow P \cup \{p \mid (p, label) \in c_i\}$
    Enqueue(Q, c_i) // push all the elements in c_i into Q
  end for
  while $Q \neq \emptyset$ **do**
    $(p, label) \leftarrow$ **Dequeue**(Q) // pop the first element of Q
    if $\exists q \in$ **Neighbor**(p) and $q \notin P$ and $S_i[q] =$ True **then**
      $T \leftarrow T \cup \{(q, label)\}$; $P \leftarrow P \cup \{q\}$
      Enqueue$(Q, (q, label))$ // push the element $(q, label)$ into Q
    end if
  end while
  $E =$ **GroupByLabel**(T)
  return E
end function

PSE NET

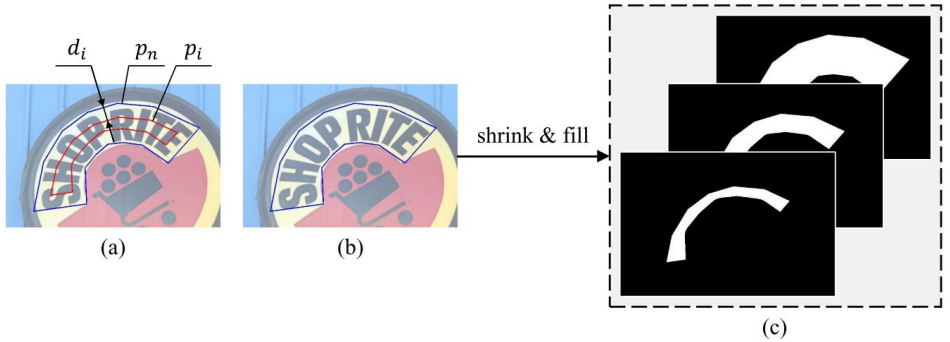


Figure 4: The illustration of label generation. (a) contains the annotations for d, p_i and p_n. (b) shows the original text instances. (c) shows the segmentation masks with different kernel scales.

3.3 Label Generation

As illustrated in Fig. 2, PSENet produces segmentation results (e.g. $S_1, S_2, ..., S_n$) with different kernel scales. Therefore, it requires the corresponding ground truths with different kernel scales as well during training. In our practice, these ground truth labels can be conducted simply and effectively by shrinking the original text instance. The polygon with blue border in Fig. 4 (b) denotes the original text instance and it corresponds to the largest segmentation label mask (see the rightmost map in Fig. 4 (c)). To obtain the shrunk masks sequentially in Fig. 4 (c), we utilize the Vatti clipping algorithm [28] to shrink the original polygon p_n by d_i pixels and get shrunk polygon p_i (see Fig. 4 (a)). Subsequently, each shrunk polygon p_i is transferred into a 0/1 binary mask for segmentation label ground truth. We denote these ground truth maps as $G_1, G_2, ..., G_n$ respectively. Mathematically, if we consider the scale ratio as r_i, the margin d_i between p_n and p_i can be calculated as:

$$d_i = \frac{\text{Area}(p_n) \times (1 - r_i^2)}{\text{Perimeter}(p_n)}, \tag{1}$$

where Area$(\cdot)$ is the function of computing the polygon area, Perimeter$(\cdot)$ is the function of computing the polygon perimeter. Further, we define the scale ratio r_i for ground truth map G_i as:

$$r_i = 1 - \frac{(1 - m) \times (n - i)}{n - 1}, \tag{2}$$

where m is the minimal scale ratio, which is a value in $(0, 1]$. Based on the definition in Eqn. (2), the values of scale ratios (i.e., $r_1, r_2, ..., r_n$) are decided by two hyper-parameters n and m, and they increase linearly from m to 1.

PSE NET

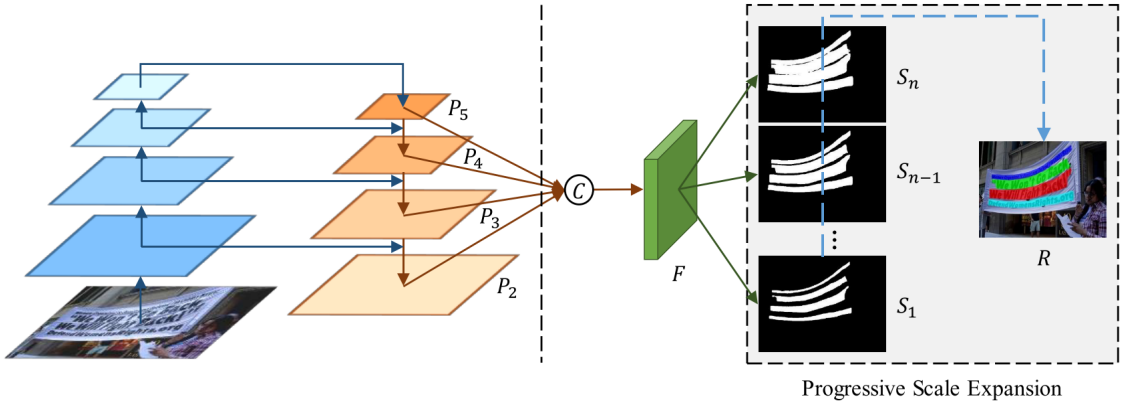


Figure 2: Illustration of our overall pipeline. The left part is implemented from FPN [16]. The right part denotes the feature fusion and the progressive scale expansion algorithm.

3.4 Loss Function

For learning PSENet, the loss function can be formulated as:

$$L = \lambda L_c + (1 - \lambda) L_s, \tag{3}$$

where L_c and L_s represent the losses for the complete text instances and the shrunk ones respectively, and λ balances the importance between L_c and L_s.

PSE NET

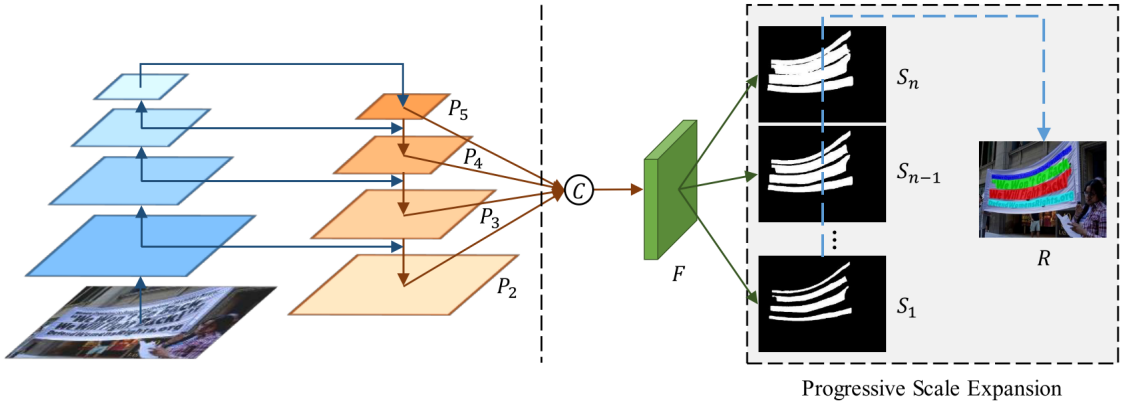


Figure 2: Illustration of our overall pipeline. The left part is implemented from FPN [16]. The right part denotes the feature fusion and the progressive scale expansion algorithm.

It is common that the text instances usually occupy only an extremely small region in natural images, which makes the predictions of network bias to the non-text region, when binary cross entropy [2] is used. Inspired by [20], we adopt dice coefficient in our experiment. The dice coefficient $D(S_i, G_i)$ is formulated as in Eqn. (4):

$$D(S_i, G_i) = \frac{2\sum_{x,y}(S_{i,x,y} * G_{i,x,y})}{\sum_{x,y} S_{i,x,y}^2 + \sum_{x,y} G_{i,x,y}^2}, \tag{4}$$

where $S_{i,x,y}$ and $G_{i,x,y}$ refer to the value of pixel (x, y) in segmentation result S_i and ground truth G_i, respectively.

Furthermore, there are many patterns similar to text strokes, such as fences, lattices, etc. Therefore, we adopt Online Hard Example Mining (OHEM) [24] to L_c during training to better distinguish these patterns.

L_c focuses on segmenting the text and non-text region. Let us consider the training mask given by OHEM as M, and thus L_c can be written as:

$$L_c = 1 - D(S_n \cdot M, G_n \cdot M), \tag{5}$$

PSE NET

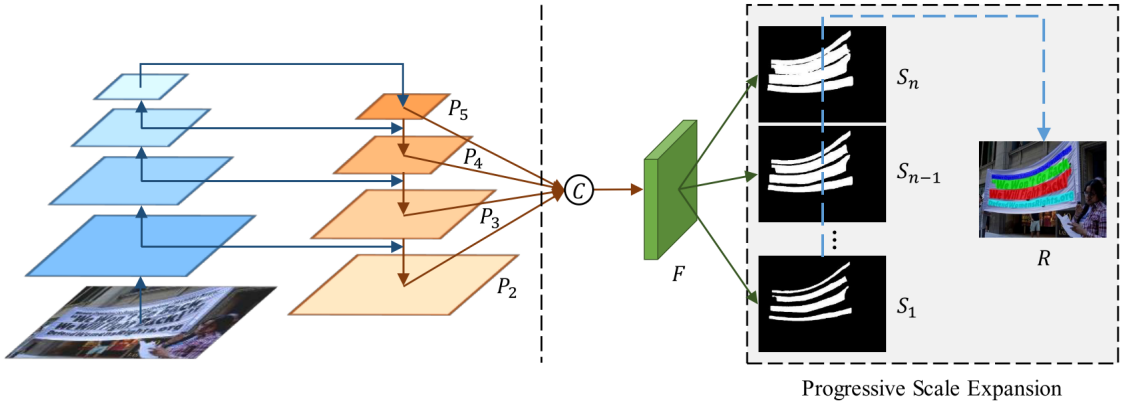


Figure 2: Illustration of our overall pipeline. The left part is implemented from FPN [16]. The right part denotes the feature fusion and the progressive scale expansion algorithm.

L_s is the loss for shrunk text instances. Since they are encircled by the original areas of the complete text instances, we ignore the pixels of non-text region in the segmentation result S_n to avoid a certain redundancy. Therefore, L_s can be formulated as follows:

$$L_s = 1 - \frac{\sum_{i=1}^{n-1} D(S_i \cdot W, G_i \cdot W)}{n-1}, \quad W_{x,y} = \begin{cases} 1, & if\ S_{n,x,y} \geq 0.5; \\ 0, & otherwise. \end{cases} \tag{6}$$

Here, W is a mask which ignores the pixels of non-text region in S_n, and $S_{n,x,y}$ refers to the value of pixel (x, y) in S_n.

PSE NET

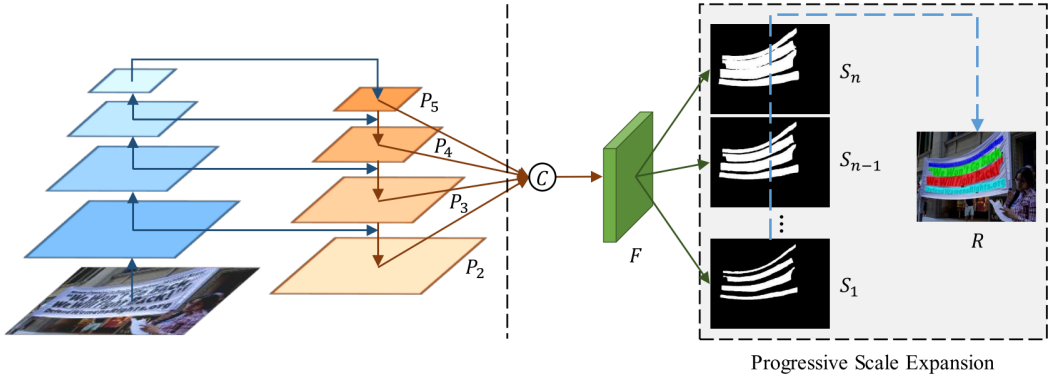


Figure 2: Illustration of our overall pipeline. The left part is implemented from FPN [16]. The right part denotes the feature fusion and the progressive scale expansion algorithm.

3.5 Implementation Details

The backbone of PSENet is implemented from FPN [16]. We firstly get four 256 channels feature maps (i.e. P_2, P_3, P_4, P_5) from the backbone. To further combine the semantic features from low to high levels, we fuse the four feature maps to get feature map F with 1024 channels via the function $C(\cdot)$ as: $F = C(P_2, P_3, P_4, P_5) = P_2 \parallel \text{Up}_{\times 2}(P_3) \parallel \text{Up}_{\times 4}(P_4) \parallel \text{Up}_{\times 8}(P_5)$, where "$\parallel$" refers to the concatenation and $\text{Up}_{\times 2}(\cdot)$, $\text{Up}_{\times 4}(\cdot)$, $\text{Up}_{\times 8}(\cdot)$ refer to 2, 4, 8 times upsampling, respectively. Subsequently, F is fed into Conv(3, 3)-BN-ReLU layers and is reduced to 256 channels. Next, it passes through multiple Conv(1, 1)-Up-Sigmoid layers and produces n segmentation results $S_1, S_2, ..., S_n$. Here, Conv, BN, ReLU and Up refer to convolution [14], batch normalization [11], rectified linear units [4] and upsampling.

We set n to 6 and m to 0.5 for label generation and get the scales $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. During training, we ignore the blurred text regions labeled as DO NOT CARE in all datasets. The λ of loss balance is set to 0.7. The negative-positive ratio of OHEM is set to 3. The data augmentation for training data is listed as follows: 1) the images are rescaled with ratio $\{0.5, 1.0, 2.0, 3.0\}$ randomly; 2) the images are horizontally fliped and rotated in range $[-10°, 10°]$ randomly; 3) 640×640 random samples are cropped from the transformed images; 4) the images are normalized using the channel means and standard deviations. For quadrangular text dataset, we calculate the minimal area rectangle to extract the bounding boxes as final predictions. For curve text dataset, the Ramer-Douglas-Peucker algorithm [21] is applied to generate the bounding boxes with arbitrary shapes.

PSE NET

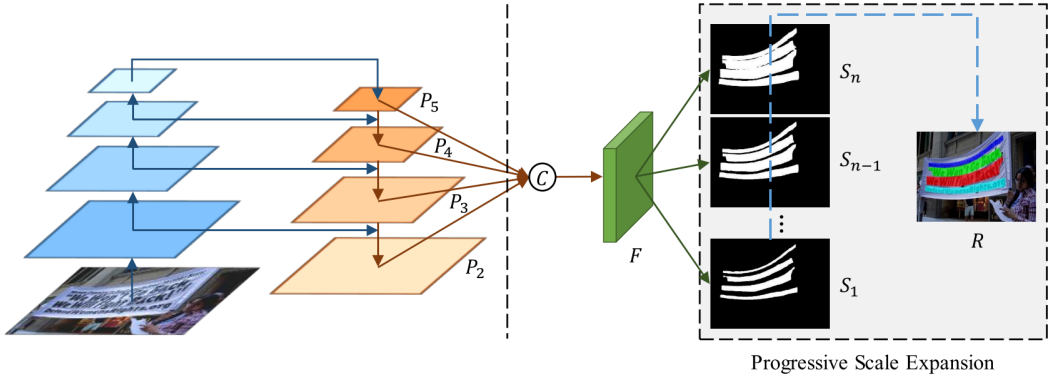
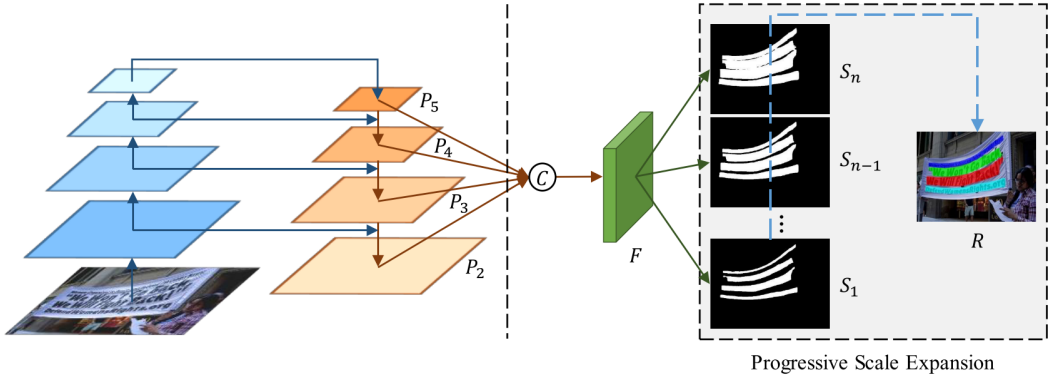


Figure 2: Illustration of our overall pipeline. The left part is implemented from FPN [16]. The right part denotes the feature fusion and the progressive scale expansion algorithm.

Table 1: Comparison to the traditional semantic segmentation baseline with the same backbone on ICDAR 2015. “P”, “R”, “F” refer to Precision, Recall, F-measure respectively.

Method	P (%)	R (%)	F (%)
PSENet-1s ($n = 1, m = 1.0$, semantic segmentation baseline)	77.41	61.53	68.56
PSENet-1s ($n = 6, m = 0.5$)	88.71	85.51	**87.08**

PSE NET

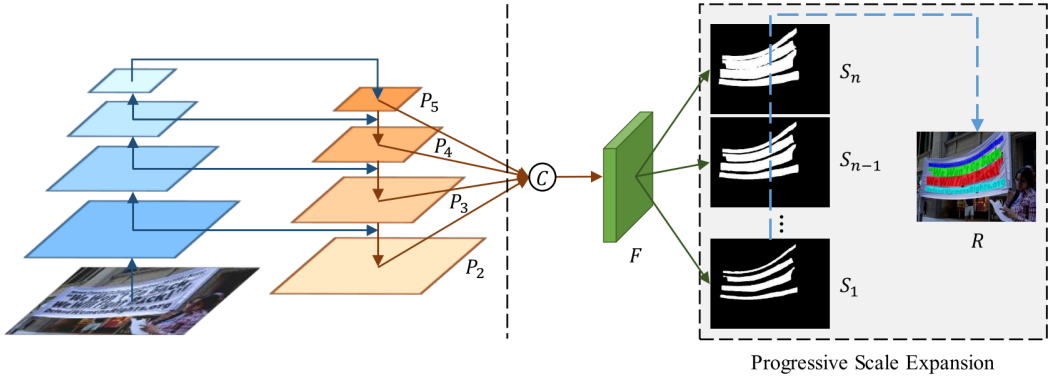


rall pipeline. The left part is implemented from FPN [16]. The right
nd the progressive scale expansion algorithm.

Table 2: The single-scale results on ICDAR 2015, ICDAR 2017 MLT and SCUT-CTW1500. "P", "R" and "F" refer to precision, recall and F-measure respectively. * indicates the results from [18]. "1s", "2s" and "4s" means the width and height of the output map are $1/1$, $1/2$ and $1/4$ of the input test image. The best, second-best F-measure are highlighted in red and blue, respectively.

Method	IC15				IC17-MLT			SCUT-CTW1500		
	P (%)	R (%)	F (%)	FPS	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
CTPN [26]	74.22	51.56	60.85	7.1	-	-	-	60.4*	53.8*	56.9*
SegLink [23]	74.74	76.50	75.61	-	-	-	-	42.3*	40.0*	40.8*
SSTD [8]	80.23	73.86	76.91	7.7	-	-	-	-	-	-
WordSup [9]	79.33	77.03	78.16	-	-	-	-	-	-	-
EAST [30]	83.27	78.33	80.72	6.52	-	-	-	78.7*	49.1*	60.4*
R^2CNN [12]	85.62	79.68	82.54	-	-	-	-	-	-	-
FTSN [1]	88.65	80.07	84.14	-	-	-	-	-	-	-
SLPR [31]	85.5	83.6	84.5	-	-	-	-	80.1	70.1	74.8
linkage-ER-Flow [27]	-	-	-	-	44.48	25.59	32.49	-	-	-
TH-DL [27]	-	-	-	-	67.75	34.78	45.97	-	-	-
TDN SJTU2017 [27]	-	-	-	-	64.27	47.13	54.38	-	-	-
SARI FDU RRPN v1 [27]	-	-	-	-	71.17	55.50	62.37	-	-	-
SCUT DLVClab1 [27]	-	-	-	-	80.28	54.54	64.96	-	-	-
Lyu et al. [19]	89.5	79.7	84.3	3.6	83.8	55.6	66.8	-	-	-
FOTS [17]	91.00	85.17	87.99	7.5	80.95	57.51	67.25	-	-	-
CTD+TLOC [18]	-	-	-	-	-	-	-	77.4	69.8	73.4
PSENet-4s (ours)	87.98	83.87	85.88	12.38	75.98	67.56	71.52	80.49	78.13	79.29
PSENet-2s (ours)	89.30	85.22	87.21	7.88	76.97	68.35	72.40	81.95	79.30	80.60
PSENet-1s (ours)	88.71	85.51	87.08	2.33	77.01	68.40	72.45	82.50	79.89	81.17

PSE NET



pipeline. The left part is implemented from FPN [16]. The right the progressive scale expansion algorithm.

Figure 7: Comparisons on SCUT-CTW1500. The proposed PSENet produces several detections that are even missed by the groundtruth labels.

THANKS