Learning Statistical Texture for Semantic Segmentation

Lanyun Zhu[1]* Deyi Ji[2]* Shiping Zhu[1]† Weihao Gan[2]† Wei Wu[2] Junjie Yan[2]
Beihang University [1] SenseTime Research [2]
{zhulanyun, shiping.zhu}@buaa.edu.cn
{jideyi, ganweihao, wuwei, yanjunjie}@sensetime.com

Abstract

Existing semantic segmentation works mainly focus on learning the contextual information in high-level semantic features with CNNs. In order to maintain a precise boundary, low-level texture features are directly skip-connected into the deeper layers. Nevertheless, texture features are not only about local structure, but also include global statistical knowledge of the input image. In this paper, we fully take advantages of the low-level texture features and propose a novel Statistical Texture Learning Network (STL-Net) for semantic segmentation. For the first time, STL-Net analyzes the distribution of low level information and efficiently utilizes them for the task. Specifically, a novel Quantization and Counting Operator (QCO) is designed to describe the texture information in a statistical manner. Based on QCO, two modules are introduced: (1) Texture Enhance Module (TEM), to capture texture-related information and enhance the texture details; (2) Pyramid Texture Feature Extraction Module (PTFEM), to effectively extract the statistical texture features from multiple scales. Through extensive experiments, we show that the proposed STL-Net achieves state-of-the-art performance on three semantic segmentation benchmarks: Cityscapes, PASCAL Context and ADE20K.

1. Introduction

Semantic segmentation aims to predict a label for each pixel in an image. It is one of the most fundamental problems in computer vision, and is widely applied in a lot of areas such as automatic driving and human-machine interaction.

Recent semantic segmentation methods mainly focus on exploiting the contextual information in high-level features with deep fully convolution based networks [23]. However, only using high-level features from deep layers results in

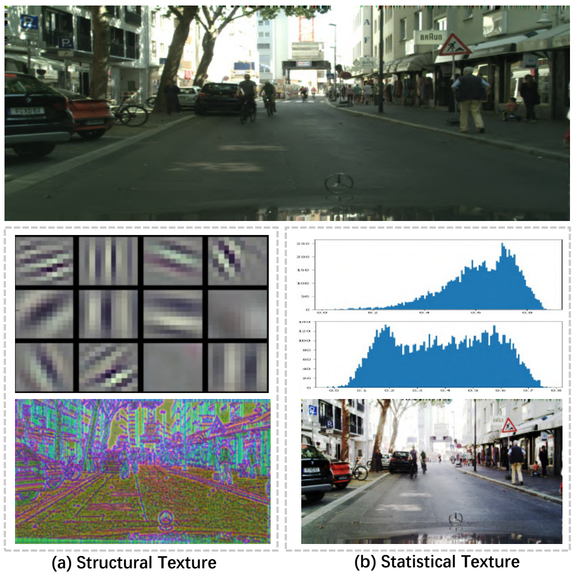

Figure 1. Examples of the structural texture and statistical texture of a image. (a) shows the convolution filters of shallow layers and the corresponding extracted structural texture in typical CNN pipeline. (b) shows the original histogram, equalized histogram and image after statistical texture enhancement, respectively.

coarse and inaccurate output, as they are extracted from a large receptive field and miss some crucial low-level details, such as edges. To alleviate this problem, people employ skip connection to fuse low- and high-level features. DeepLabv3+ [4] directly combines feature maps from shallow layers and deep layers before feeding them into the prediction head. Some FPN-like methods [25, 19, 30] employ encoder-decoder structure with lateral path to refine features in a top-down manner, where multi-scale boundaries are implicitly learned. SFNet [17] then applies a semantic flow to align detailed object boundaries from different levels. All of them show that low-level local features in shallow CNN layers provide the structural texture information, such as edge, which is essential for pixel-wise segmentation task.

From the perspective of digital image processing, image texture is not only about the local structural property,

*The first two authors contributed equally. Work done at SenseTime. Lanyun Zhu firstly proposed the paper idea.
†Corresponding Author.

but also about global statistical property [9, 20, 10]. The structural one usually refers to some local patterns, such as boundary, smoothness and coarseness. In a typical CNN pipeline, filters in the shallow layers are good at extracting these local texture features. When visualizing those filters, we can observe that different frequencies of the local signal are analyzed (Fig. 1(a)). Therefore, structural property is also referred to spectral domain analysis [9]. Another important property of texture is statistical one. Based on some low-level information (such as pixel value or local region property), it focuses on the distribution analysis of the image, such as histogram of intensity. For example in Fig. 1(b), an image captured in dark environment is usually in poor-visualization quality. After the histogram equalization enhancement step, it turns to be more detailed and better for segmentation. Some traditional methods try utilizing statistical texture in their tasks [1, 24, 10]. However, there is no mechanism in modern CNN architecture to explicitly extract and utilize the statistical texture information for semantic segmentation. Therefore, in this paper, we propose a Statistical Texture Learning Network (STLNet) to describe and exploit statistical texture information for this task.

Firstly, we propose a Quantization and Counting Operator (QCO) to effectively describe texture intensities in a statistical manner in deep neural networks. Specifically, statistical texture of the input image is usually of a wide variety and a continuous distribution in spectral domain, which is difficult to extract and optimize in deep neural networks. Thus in the QCO, we first quantize the input feature into multiple levels. Each level can represent a kind of texture statistics, by which the continuous texture can be well sampled for easier description. After the quantization, the intensity of each level is then counted for texture feature encoding.

Based on QCO, we further propose the Texture Enhancement Module (TEM) and Pyramid Texture Feature Extraction Module (PTFEM), to enhance the texture details of low-level features and exploit texture-related information from multiple scales, respectively. More comprehensively, low-level features are usually of low qualities and difficult for statistical features extraction. Inspired by histogram equalization, TEM is designed to build a graph to propagate information of all original quantization levels for texture enhancement. Furthermore, PTFEM exploits the texture information from multiple scales with a texture feature extraction unit and pyramid structure.

Overall, our contributions are summarized as follows:

- For the first time, we introduce the statistical texture information to semantic segmentation and propose a novel STLNet to take full advantage of texture information, where both low-level and high-level features are well learned in an end-to-end manner.
- For effective description of statistical texture in deep neural networks, a novel Quantization and Counting Operator (QCO) is designed to quantize the continuous texture into multiple level intensities.
- With the help of QCO, we propose the Texture Enhancement Module (TEM) and Pyramid Texture Feature Extraction Module (PTFEM) successively to enhance the statistical details and extract the texture features, respectively.
- The proposed method is practical and can be implemented in a plug-and-play fashion. Experiments show that our method achieves state-of-the-art results on Cityscapes, Pascal Context and ADE20K datasets.

2. Related Works

Semantic Segmentation. In recent years, benefited from the rapid development of deep learning, semantic segmentation has achieved great success. Most of the modern semantic segmentation models [23, 13, 25, 19, 5, 32, 8, 12] are based on fully convolutional networks (FCN) [23], which first replaces the fully connected layers in common classification networks by convolutional layers, getting pixel-level prediction results. After that, a lot of methods [37, 3, 40, 21, 28, 22, 39] are proposed to improve the basic FCN. Some methods employ pyramid structure to capture information from different sizes of receptive fields. PSPNet [37] feeds features into pyramid pooling layers with different pooling scales, and DeeplabV3 series [3, 4] proposes an atrous spatial pyramid pooling layer consisting of multiple dilated convolutions with different dilated rates. In our proposed methods, we also employ a pyramid structure to harvest texture information from different scales.

Low-level Features Extraction. Low-level features play a crucial role in boosting performance of semantic segmentation. A lot of methods are proposed to extract and utilize low-level features. Some methods [25, 19, 4, 30] employ skip connections to deliver low-level features to deeper layers. However, the simple multi-level features adding or concatenation operations may cause feature misalignment problems, weakening the effectiveness of low-level features. To address the issue, ANN [40] proposes a non-local block to fuse multi-level features. SFNet [17] aligns low-level and high-level features by learning and utilizing the semantic flow. Some works aim to exploit some more explicit low-level information such as boundary. [6] proposes a boundary-aware feature propagation module to propagate features guided by the boundary-related low-level information. [15] explicitly generates and supervise object edges. However, all these methods only consider structural features such as edge. In contrast to them, our proposed methods further focus on statistical texture features.

Feature Encoding. The proposed QCO in this paper can

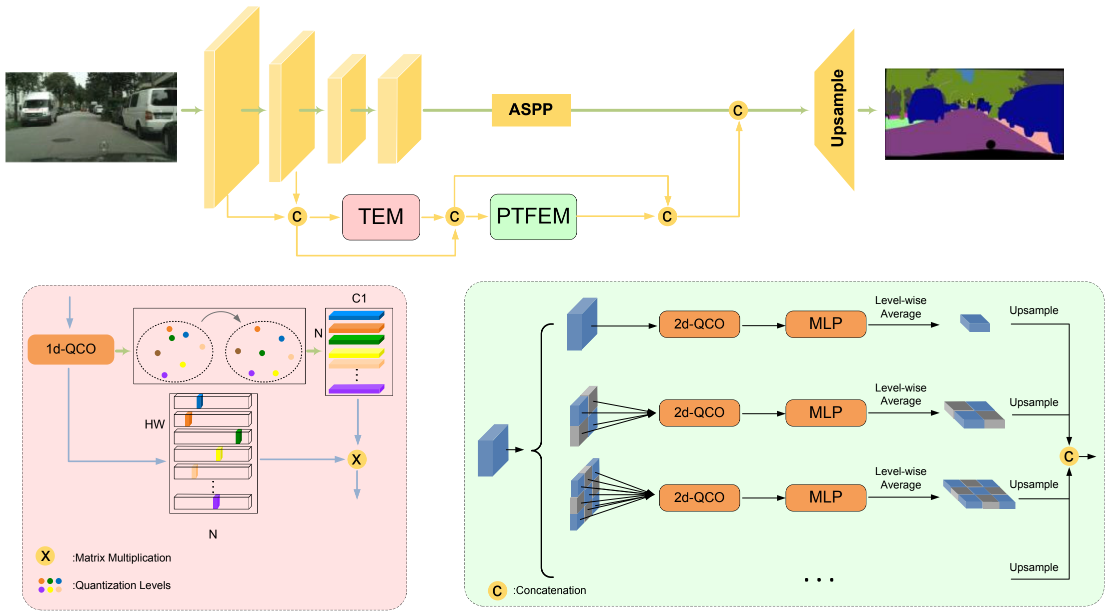

Figure 2. An overview of the proposed Statistical Texture Learning Network (STLNet). The base network is ResNet101 followed by an ASPP module. An input image is fed into backbone ResNet-101 to extract high-level feature and multiple low-level features. The high-level feature is then fed into the ASPP module to obtain high-level context feature. We concate the first two low-level features and send into the Texture Enhance Module (TEM) to enhance the texture details, followed by the Pyramid Texture Feature Extraction Module (PTFEM) to extract the multi-scale statistical texture information. Finally, context and texture features are combined and upsampled for semantic prediction.

be treated as a feature encoding method. Some previous methods [35, 33, 26] try encoding features to better utilize information containing in them. [35] encodes the residual vector with a learned codebook. [33] employs the similar method to harvest contextual information. In contrast to them, QCO is in a self-adaptive manner when quantization, which is more stable. The most similar work may be [26], which encodes the feature to a learnable histogram. The main difference lies in that it quantizes each channel separately, while our methods can quantize and count the high-dimensional representation by a well-designed manner.

3. Method

In this section, we introduce the proposed Statistical Texture Learning Network (STLNet) in detail. First, we describe the overall structure of STLNet in subsection 3.1. Then we expound the proposed Quantization and Counting Operator (QCO), Texture Enhancement Module (TEM) and Pyramid Texture Feature Extraction Module (PTFEM) in subsection 3.2, 3.3 and 3.4, respectively. Finally, we present the loss function used to supervise the proposed network in subsection 3.5.

3.1. Overall Structure

The overall structure of STLNet is shown in Fig. 2. The network can be divided into two parts: base network and texture extraction branch. For base network, we utilize the dilated ResNet101 followed by an Atrous Spatial Pyramid Pooling (ASPP) module. For texture extraction branch, we first extract features from layer 1 and layer 2 of backbone ResNet, downsample them to the same size as the output of base network, and then concatenate them. After that, a Texture Enhancement Module (TEM) and Pyramid Texture Feature Extraction Module (PTFEM) are followed serially to enhance texture details and exploit texture-related information respectively. Finally, we concatenate output features from base network, TEM, PTFEM and the extracted low-level features from backbone, and pass them through a convolutional layer to get the final prediction map.

3.2. Quantization and Counting Operator

Recently, most of deep networks for image processing are constructed with fully convolutional layers. Convolution operator is sensitive to the local variation, which can help to exploit some local features such as boundary. However, it is powerless for describing statistical texture. Therefore, we first propose a Quantization and Counting Operator (QCO) to describe texture in a statistical manner. Specifically, it aims to quantize the input feature into multiple levels, then count the number of features belonging to each level. QCO can be classified into two categories: One-dimensional (1-d) QCO and Two-dimensional (2-d) QCO, which are utilized in TEM and and PTFEM respectively. Both QCOs contain three parts: quantization, counting and average feature encoding. The structure of 1-d QCO is shown in Fig. 3.

3.2.1 1-d QCO

Quantization. The input feature map of QCO is denoted as $\mathbf{A} \in \mathbb{R}^{C \times H \times W}$ and first passed through a global average pooling to get the global averaged feature $g \in \mathbb{R}^{C \times 1 \times 1}$. For each spatial position $\mathbf{A}_{i,j}(i \in [1, W], j \in [1, H])$ on $\mathbf{A}$, we calculate the cosine similarity between g and $\mathbf{A}_{i,j}$, obtaining $\mathbf{S} \in \mathbb{R}^{1 \times H \times W}$. Specifically, each position $\mathbf{S}_{i,j}$ of $\mathbf{S}$ is denoted as:

$$\mathbf{S}_{i,j} = \frac{g \cdot \mathbf{A}_{i,j}}{\|g\|_2 \cdot \|\mathbf{A}_{i,j}\|_2}. \tag{1}$$

$\mathbf{S}$ is then reshaped to $\mathbb{R}^{HW}$. Next we quantize $\mathbf{S}$ into N levels $\mathbf{L} = [L_1, L_2, ..., L_N]$, and $\mathbf{L}$ is obtained by equally dividing N points between the minimum and maximum values of $\mathbf{S}$. Concretely, the nth level $\mathbf{L}_n$ is calculated by:

$$\mathbf{L}_n = \frac{max\left(\mathbf{S}\right) - min\left(\mathbf{S}\right)}{N} \cdot n + min(\mathbf{S}). \tag{2}$$

For each spatial pixel $\mathbf{S}_i \in \mathbb{R}(i \in [1, HW])$, we quantize it to a quantization encoding vector $\mathbf{E}_i \in \mathbb{R}^N(i \in [1, HW])$, thus $\mathbf{S}$ will be finally quantized to a quantization encoding matrix $\mathbf{E} \in \mathbb{R}^{N \times HW}$. More comprehensively, $\mathbf{S}_i$ is quantized with N functions $F = \{f_1, f_2, ..., f_N\}$, where each f_n produces $\mathbf{E}_{i,n}$ by:

$$\mathbf{E}_{i,n} = \begin{cases} 1 - |\mathbf{L}_n - \mathbf{S}_i| & if \quad -\frac{0.5}{N} \leq \mathbf{L}_n - \mathbf{S}_i < \frac{0.5}{N} \\ 0 & \qquad\qquad else \end{cases}. \tag{3}$$

Finally, the N results of F are concatenated to get $\mathbf{E}_i$. By this way, index of the maximum value of $\mathbf{E}_i$ can reflect the quantization level of $\mathbf{S}_i$.

Note that, in contrast to traditional argmax operation or one-hot encoding mechanism with binarization, we apply a smoother way for quantization encoding so that the gradient vanishing problem can be avoided during the process of back propagation.

Counting. Given the quantization encoding matrix $\mathbf{E}$, we further generate the quantization counting map $\mathbf{C} \in \mathbb{R}^{N \times 2}$, where the first dimension represents each quantization level and the second dimension represents corresponding normalized counting number. Specifically, $\mathbf{C}$ is written as:

$$\mathbf{C} = Cat\left(\mathbf{L}, \frac{\sum_{i=1}^{HW} \mathbf{E}_{i,n}}{\sum_{n=1}^{N}\sum_{i=1}^{HW} \mathbf{E}_{i,n}}\right), \tag{4}$$

where Cat denotes concatenation operation.

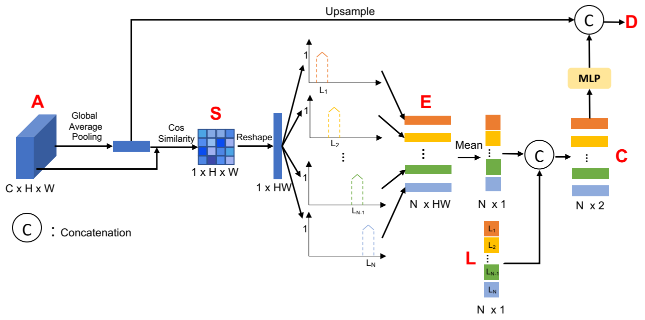

Figure 3. Structure of 1-d QCO. H and W represent feature map height and width respectively, and N represents the number of quantization levels.

Average Feature Encoding. The quantization counting map $\mathbf{C}$ reflects the relative statistics of input feature map, as the counted object is the distance from average feature. To further obtain absolute statistical information, we encode the global average feature g into $\mathbf{C}$ and produce $\mathbf{D}$. We first upsample g to $\mathbb{R}^{N \times C}$, and then obtain $\mathbf{D}$ by:

$$\mathbf{D} = Cat\left(MLP\left(\mathbf{C}\right), g\right), \tag{5}$$

where MLP aims to increase the dimension for $\mathbf{C}$. It contains two layers, in which the first one is followed by a Leaky ReLU to enhance nonlinearity.

Both $\mathbf{E}$ and $\mathbf{D}$ will be output, and denote the quantization encoding map and statistical feature respectively.

3.2.2 2-d QCO

The output of 1-d QCO reflects the distribution of features from all spatial positions. However, it carries no information regarding spatial relationships between pixels, which plays an important role in describing texture. To this end, we further propose a 2-d QCO to count the distribution of co-occurring pixel features.

Quantization. Quantization in 2-d QCO is intent to count the co-occurring spatial relationships among pixels in the input feature map, and can be extended by that in 1-d QCO. Specifically, the input $\mathbf{A} \in \mathbb{R}^{C \times H \times W}$ is first passed through the similar procedure as 1-d QCO to get quantization encoding map $\mathbf{E}$ and quantization levels $\mathbf{L}$. We then reshape $\mathbf{E}$ to $\mathbb{R}^{N \times 1 \times H \times W}$. For each pair of adjacent pixels $\mathbf{E}_{i,j} \in \mathbb{R}^{N \times 1}$ and $\mathbf{E}_{i,j+1} \in \mathbb{R}^{N \times 1}$, we calculate their product $\hat{\mathbf{E}}_{i,j}$:

$$\hat{\mathbf{E}}_{i,j} = \mathbf{E}_{i,j} \cdot \mathbf{E}_{i,j+1}^{T}, \tag{6}$$

where T refers to matrix transposition. Only when $\mathbf{A}_{i,j}$ is quantized to $\mathbf{L}_m$ and $\mathbf{A}_{i,j+1}$ is quantized to $\mathbf{L}_n$, the corresponding $\hat{\mathbf{E}}_{m,n,i,j}$ dose not equal to zero. Therefore, $\hat{\mathbf{E}} \in \mathbb{R}^{N \times N \times H \times W}$ can represent the quantization co-occurrence of every two adjacent pixels.

Counting. Given $\hat{\mathbf{E}}$, we generate a 3-dimensional map $\mathbf{C}$, where the first two dimensions represent each possible quantization co-occurrence and the third dimension represents the corresponding normalized counting number. $\mathbf{C}$ is denoted as:

$$\mathbf{C} = Cat\left(\hat{\mathbf{L}}, \frac{\sum_{i=1}^{H}\sum_{j=1}^{W}\hat{\mathbf{E}}_{m,n,i,j}}{\sum_{m=1}^{N}\sum_{n=1}^{N}\sum_{i=1}^{H}\sum_{j=1}^{W}\hat{\mathbf{E}}_{m,n,i,j}}\right), \\ \hat{\mathbf{L}} \in \mathbb{R}^{2\times N\times N}, \quad \hat{\mathbf{L}}_{m,n} = [\mathbf{L}_m, \mathbf{L}_n], \tag{7}$$

where $\hat{\mathbf{L}}$ denotes all possible quantization levels pairs of adjacent pixels.

Average Feature Encoding. Following 1-d QCO, we also encode the average feature to obtain absolute representation. Denote the average feature of processed region as $g \in \mathbb{R}^{C}$, we upsample it to $\mathbb{R}^{N\times N\times C}$. Then the final output $\mathbf{D}$ is obtained by:

$$\mathbf{D} = Cat\left(MLP(\mathbf{C}), g\right). \tag{8}$$

3.3. Texture Enhancement Module

Low-level features extracted from shallow layers of backbone network are often with low quality, especially with low contrast, causing the texture details to be ambiguous, which has negative impacts on the extraction and utilization of low-level information. Therefore, we propose a Texture Enhancement Module (TEM) to enhance the texture details of low-level features so that it is easier to capture texture-related information in the following steps.

The way we enhance texture is inspired by histogram quantification, a classical method for image quality enhancement. Specifically, this method first produces a histogram, whose horizontal axis and vertical axis represent each gray level and its counting value respectively. We denote these two axes as two feature vectors $\mathbf{G}$ and $\mathbf{F}$. Histogram quantification aims to reconstruct levels to $\mathbf{G}'$ using the statistical information containing in $\mathbf{F}$. Each level $\mathbf{G}_n$ is transformed to $\mathbf{G}'_n$ by:

$$\mathbf{G}'_n = \frac{(N-1)\sum_{i=0}^{n}\mathbf{F}_n}{\sum_{i=0}^{N}\mathbf{F}_n}, \tag{9}$$

where N is the total number of gray levels.

In the proposed TEM, we first pass the input feature map through a 1-d QCO to get the quantization encoding map $\mathbf{E} \in \mathbb{R}^{N\times HW}$ and statistics feature $\mathbf{D} \in \mathbb{R}^{C_1\times N}$, where $\mathbf{D}$ plays the role of histogram. After that, we get the new quantization levels $\mathbf{L}'$ using $\mathbf{D}$. Inspired by Equation. 9, each new level should be obtained by perceiving statistical information of all original levels, which can be treated as a graph. To this end, we build a graph to propagate information from all levels. The statistical feature of each quantization level is defined as a node. In the traditional histogram quantification algorithm, the adjacent matrix is a manually defined diagonal matrix, and we extend it to a learned one as follows:

$$\mathbf{X} = Softmax\left(\phi_1(\mathbf{D})^T \cdot \phi_2(\mathbf{D})\right), \tag{10}$$

where ϕ_1 and ϕ_2 refer to two different 1×1 convolutions, and Softmax performed on the first dimension works as a nonlinear normalization function. Then we update each node by fusing features from all other nodes, getting the reconstructed quantization levels $\mathbf{L}' \in \mathbb{R}^{C_2\times N}$:

$$\mathbf{L}' = \phi_3(\mathbf{D}) \cdot \mathbf{X}, \tag{11}$$

where ϕ_3 refers to another 1×1 convolution.

After that, we assign the reconstructed levels $\mathbf{L}'$ to each pixel to get the final output $\mathbf{R}$ using the quantization encoding map $\mathbf{E} \in \mathbb{R}^{N\times HW}$, since $\mathbf{E}$ can reflect the original quantization level of each pixel. $\mathbf{R}$ is obtained by:

$$\mathbf{R} = \mathbf{L}' \cdot \mathbf{E}. \tag{12}$$

Finally, $\mathbf{R}$ is reshaped to $\mathbf{R}^{C_2\times H\times W}$.

3.4. Pyramid Texture Feature Extraction Module

We further propose a Pyramid Texture Feature Extraction Module (PTFEM), which aims to exploit texture-related information from multiple scales using feature maps containing rich texture details. We first describe the unit to capture texture features from each processed region, then introduce the pyramid structure to build PTFEM.

Texture Feature Extraction Unit. Texture is highly correlated with the statistical information about spatial relationships between pixels. The way we extract texture information is inspired by gray level co-occurrence matrix (GLCM), a widely used method to capture texture representations. GLCM first produces a co-occurrence matrix $\mathbf{M} \in \mathbb{R}^{H\times W}$ where the value $\mathbf{M}_{h,w}$ of each position on $\mathbf{M}$ denotes the number of occasions that the gray values of two adjacent pixels form a 2-d vector $[h, w]$. Then some statistical descriptors such as contrast, uniformity and homogeneity are employed to represent the texture information of the region. In our proposed methods, for the processed region of feature map, we first feed it into the 2-d QCO to get the co-occurrence statistics feature $\mathbf{F} \in \mathbb{R}^{C\times N\times N}$, where C denotes the channel number and N denotes the number of quantization levels. Different from the handcrafted descriptors used in GLCM, benefited from the powerful feature extraction ability of deep learning, we force the network to learn useful statistical representations automatically. Specifically, an MLP followed by a level-wise average is employed to produce the texture feature $\mathbf{T}$ of processed region:

$$\mathbf{F}' = MLP(\mathbf{F}), \quad \mathbf{F}' \in \mathbb{R}^{C'\times N\times N}, \\ \mathbf{T} = \frac{\sum_{m=1}^{N}\sum_{n=1}^{N}\mathbf{F}'_{:,m,n}}{N\cdot N}. \tag{13}$$

Pyramid Structure. As discovered by some previous works [37, 3], multi-scale features help to boost the performance and robustness of semantic segmentation effectively. Such features can be captured by the pyramid structure such as Spatial Pyramid Pooling and Atrous Spatial Pyramid Pooling. Inspired by the success of these methods, we also employ a pyramid structure to carve texture features from multiple scales. Specifically, the pyramid structure passes input feature map through four parallel branches with different scales [1,2,3,6]. For each branch, the feature map is separated into different number of sub-regions, and each sub-region is passed through the texture feature extraction unit to exploit the region's corresponding texture representation. Then we upsample the obtained texture feature map of each branch to the original size as input map via nearest interpolation, and concatenate the output of four branches, getting the output of PTFEM.

3.5. Loss

Following [37], we apply deep supervision to make the deep network easier to train. Specifically, besides getting the prediction map from the final output, we also add an auxiliary layer after the backbone ResNet layer 3, getting an auxiliary prediction map. Both prediction maps are supervised. The overall loss can be denoted as:

$$\mathcal{L} = \mathcal{L}_f + \alpha . \mathcal{L}_a, \tag{14}$$

where $\alpha = 0.4$ is a hyperparameter to balance the weight between final prediction loss $\mathcal{L}_f$ and auxiliary prediction loss $\mathcal{L}_a$. For $\mathcal{L}_a$, we use the common cross entropy loss. For $\mathcal{L}_f$, we employ the online hard examples mining (OHEM) loss to handle the problem caused by class imbalance. Hard examples refer to pixels whose predicted probabilities of their corresponding correct classes are smaller than θ. To ensure a stable training process, we keep at least K pixels within each batch for training. In our experiments, θ and K are set to 0.7 and 100000, respectively.

4. Experiments

4.1. Datastes

We evaluate our methods on three popular datasets, including Cityscapes, PASCAL Context and ADE20K.
Cityscapes. Cityscapes is a large urban scene understanding dataset. It contains 5000 finely annotated images and 20000 coarsely annotated images. We only use the 5000 finely annotated images for experiments, which can be divided into 2975/500/1525 images for training, validation and testing, respectively.
PASCAL Context. PASCAl Context is an extended dataset for PASCAL 2010 by providing more detailed segmentation annotation. It contains 4998 images for training and 5105 images for validation.

Method	mIoU(%)
ResNet-101	76.4
ResNet-101 + SLF	77.0
ResNet-101 + SLF + TEM	80.3
ResNet-101 + SLF + PTFEM	80.0
ResNet-101 + SLF + TEM + PTFEM	80.9
ResNet-101 + ASPP	78.8
ResNet-101 + ASPP + SLF	79.3
ResNet-101 + ASPP + SLF + TEM	80.9
ResNet-101 + ASPP + SLF + PTFEM	80.5
ResNet-101 + ASPP + SLF + TEM + PTFEM	81.5

Table 1. Ablation studies of different components in STLNet. SLF refers to low-level features (SLF) fusion.

ADE20K. ADE20K is a large and challenging dataset for scene parsing. It is consisted of 30K, 10K and 10K images for training, validation and testing, respectively. 150 classes are utilized for evaluation.

4.2. Implementation Details

The ResNet101 pretrained on ImageNet is used as the backbone of the propose STLNet. Following [37], we replace the last two downsampling operations by dilated convolutional layers with dilation rates being 2 and 4 respectively, making the output stride to 8. All BN layers in the network are replaced by Sync-BN. Quantization level is 128 in 1-d QCO and 8 in 2-d QCO. We apply stochastic gradient descent(SGD) as the optimizer. For training, we use 'poly' policy to set the learning rate, where the learning rate for each iteration equals to initial rate multiplied by $(1 - \frac{iter}{max\_iter})^{0.9}$. We apply random scaling in the range of [0.5, 2], random cropping and random left-right flipping for data augmentation. For Cityscapes, we set the initial learning rate as 0.01, weight decay as 0.0005, crop size as 769×769, batch size as 8 and training iteration as 40K. For PASCAL Context. we set the initial learning rate as 0.001, weight decay as 0.0001, crop size as 513×513, batch size as 16 and training iterations as 30K. For ADE20K, we set the initial learning rate as 0.01, weight decay as 0.0005, crop size as 513×513, batch size as 16 and training iteration as 100K. All experiments are conducted using 8 $\times$ NVIDIA TITAN Xp GPUs.

4.3. Ablation Study

All the following ablation study experiments are conducted on Cityscapes.
Ablation of STLNet. We conduct experiments to verify the effectiveness of different components in STLNet. Experimental results are shown in Table 1. We first choose ResNet101 with dilated convolutions as the baseline, which achieves 76.4% mIoU. First, we simply concatenate fea-

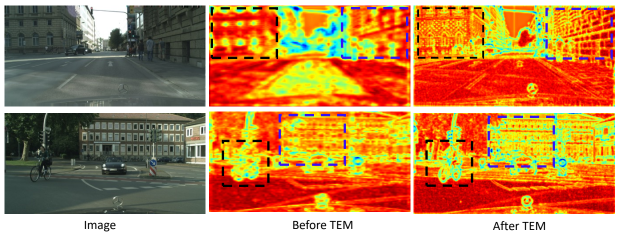

Figure 4. Visualization comparison on feature maps before and after TEM. In the indicated regions (marked by the black and blue boxes), the texture details of buildings and bicycle is clearly enhanced after TEM.

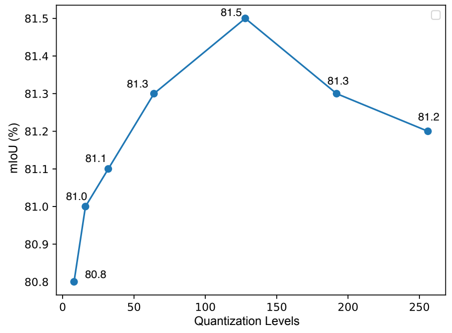

Figure 5. Ablation of quantization levels. The horizontal axis refers to number of quantization leves adopted by 1-d QCO, and vertical axis refers to the corresponding mIoU.

Method	mIoU(%)
ResNet-101 + ASPP + SLF	79.3
ResNet-101 + ASPP + SLF + TEM(w/o graph)	79.9
ResNet-101 + ASPP + SLF + TEM(w/ graph)	80.9

Table 2. Ablation studies of the graph in TEM.

1	2	3	6	mIoU(%)
✓				81.0
	✓			81.0
		✓		81.1
			✓	81.2
✓	✓	✓	✓	81.5

Table 3. Ablation of different pyramid scales adopted by PTFEM.

Method	Flip	MS	mIoU(%)
STLNet			81.5
STLNet	✓		81.7
STLNet		✓	81.9
STLNet	✓	✓	82.3

Table 4. Ablation studies of evaluation strategies.

tures from backbone ResNet layer1 and layer2, denoted as shallow layer features (SLF), and then connect them with the output of base network. This modification slightly improves the performance to 77.0%. Then we plug TEM and PTFEM after SLF individually, increasing mIoU to 80.3% and 80.0% respectively. Finally, by employing both TEM and PTFEM, the mIoU result is improved to 80.9%, which is 4.5% better than the baseline.

We further perform experiments based on a more powerful network ResNet101 + ASPP and repeat experiments mentioned above. TEM and PTFEM also improve the performance significantly in this case, and finally achieves 81.5% mIoU under ResNet101 + ASPP + SLF + TEM + PTFEM setting. It demonstrates that TEM and PTFEM can be generally plugged into any existing semantic segmentation network to improve performance.

Abaltion of Quantization Levels. During the quantization process, the number of quantization levels can be chosen flexibly, and we conduct experiments to explore the most suitable choice and results are shown in Fig. 5. Specifically, we use ResNet-101 + ASPP + SLE + TEM + PTFEM as the baseline, and report results of 6 different numbers (8, 16, 32, 64, 128, 256) of quantification levels used in 1-d QCO in TEM. The performance is unsatisfactory when the number of quantization levels is too small, and increasing the number to 128 can significantly boost mIoU to 81.5%. However, further increasing the number may cause the performance to decrease slightly. The reason may be that a too dense quantization will result in a sparse high-dimensional statistics feature, which may affect the training stability negatively and cause overfitting.

Ablation of Graph in TEM. The proposed TEM can be dived into three parts: 1-d QCO, graph construction and feature reassignment, and we evaluate the impact of the graph in TEM. We use ResNet-101 + ASPP + SLF with 79.3% mIoU as the baseline. As shown in Table 2, we first plug 1-d QCO on the baseline, and directly assign features obtained from it to each pixel, getting 79.9% mIoU. Further building the graph boosts mIoU to 80.9%. It demonstrates the importance of getting reconstructed quantization levels by perceiving all original levels.

Ablation of Pyramid Scales. The proposed PTFEM is constructed in a pyramid manner to harvest texture information from multiple scales. To verify the effectiveness of multi-scale features, we evaluate performance for models with different scale rates adopted in PTFEM. As shown in Table 3, extracting multi-scale texture features outperforms single-scale features by a large margin, which verifies the necessity of employing pyramid structure.

Ablation of Evaluation Strategies. Following [14, 8], we apply some strategies for evaluation, including left-right flipping and multi-scale input with scale sizes [0.75, 1, 1.25, 1.5, 1.75, 2]. We conduct experiments based on ResNet101 + ASPP + SLF + TE + PTFE. As shown in Table 4, Flip and MS improve performance by 0.2% and 0.4% respectively,

Method	FLOPS	Parameters
Baseline	275.55G	80.08M
TEM	15.76G	0.94M
PTFEM	1.72G	0.37M
TEM + PTFEM	17.48G	1.31M

Table 5. Flops and parameters.

Method	Backbone	mIoU(%)
PSANet [38]	ResNet-101	80.1
ANN [40]	ResNet-101	81.3
CPNet [29]	ResNet-101	81.3
CCNet [14]	ResNet-101	81.4
DANet [8]	ResNet-101	81.5
SpyGR [16]	ResNet-101	81.6
ACFNet [32]	ResNet-101	81.8
OCR [31]	ResNet-101	81.8
SFNet [17]	ResNet-101	81.8
Ours	ResNet-101	**82.3**

Table 6. Results comparison on Cityscapes test set.

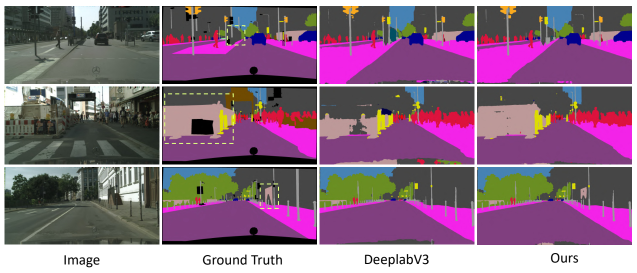

Figure 6. Visualization comparison on Cityscapes validation set.

and adopting both strategies boost mIoU to 82.3%.

FLOPs and Parameters. We report the FLOPs and parameters of proposed methods in Table 5, which are estimated for an input image with size of $1 \times 3 \times 769 \times 769$. It shows that TEM and PTFEM are lightweight and only bring very little extra cost.

Visualization of Features. TEM is proposed to enhance the texture details of low-level features. To verify the effectiveness intuitively, we visualize and compare feature maps before and after TEM, which is shown in Fig. 4. After passing through TEM, the feature maps are refined significantly and texture details become more clear.

4.4. Comparison with State-of-the-arts

We further report performance comparison with state-of-the-art methods on three datasets: Cityscapes, PASCAL Context and ADE20K. For Cityscapes, we train the proposed STLNet using both training set and validation set, and evaluate on the test set. For another two datasets, we train model on the training set and report results on the validation set. Results on three datasets are shown in Table 6, Table 7 and Table 8, respectively, and it can be observed that STLNet achieves state-of-the-art performance on all datasets. We also provide visual comparison results with DeeplabV3 on Cityscapes validation set in Fig. 6.

Method	Backbone	mIoU(%)
FCN-8s [23]		37.8
VeryDeep [27]		44.5
Deeplab-v2 [2]	ResNet-101	45.7
RefineNet [19]	ResNet-152	47.3
PSPNet [37]	ResNet-101	47.8
MSCI [18]	ResNet-152	50.3
EncNet [34]	ResNet-101	51.7
DANet [8]	ResNet-101	52.6
SpyGR [16]	ResNet-101	52.8
SVCNet [7]	ResNet-101	53.2
CPNet [29]	ResNet-101	53.9
CFNet [36]	ResNet-101	54.0
DMNet [11]	ResNet-101	54.4
RecoNet [5]	ResNet-101	54.8
CaC-Net [21]	ResNet-101	55.4
Ours	ResNet-101	**55.8**

Table 7. Results comparison on PASCAL Context validation set.

Method	Backbone	mIoU(%)
RefineNet [19]	ResNet-152	40.70
PSPNet [37]	ResNet-101	43.29
SAC [37]	ResNet-101	44.30
EncNet [34]	ResNet-101	44.65
CFNet [36]	ResNet-101	44.89
CCNet [14]	ResNet-101	45.22
RecoNet [5]	ResNet-101	45.54
CaC-Net [21]	ResNet-101	46.12
CPNet [29]	ResNet-101	46.27
Ours	ResNet-101	**46.48**

Table 8. Results comparison on ADE20K validation set.

5. Conclusion

In this paper, we propose a novel STLNet for semantic segmentation. The key innovation lies in learning texture features from the statistical perspective. Specifically, we propose a Quantization and Counting Operator to describe statistical texture representations, and then use Texture Enhancement Module and Pyramid Texture Feature Extraction Module to enhance details and exploit texture-related low-level information, respectively. Extensive ablation experiments demonstrate the effectiveness of proposed methods. And comparison with state-of-the-art methods shows that STLNet achieves outstanding performance on multiple datasets.

References

[1] S Arivazhagan and L Ganesan. Texture segmentation using wavelet transform. *Pattern Recognition Letters*, 24(16):3197–3203, 2003. 2

[2] 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 transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 8

[3] 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

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 1, 2

[5] Wanli Chen, Xinge Zhu, Ruoqi Sun, Junjun He, Ruiyu Li, Xiaoyong Shen, and Bei Yu. Tensor low-rank reconstruction for semantic segmentation. In *European Conference on Computer Vision*, pages 52–69. Springer, 2020. 2, 8

[6] Henghui Ding, Xudong Jiang, Ai Qun Liu, Nadia Magnenat Thalmann, and Gang Wang. Boundary-aware feature propagation for scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[7] Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Semantic correlation promoted shape-variant context for segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8885–8894, 2019. 8

[8] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 2, 7, 8

[9] Rafael C Gonzales and Richard E Woods. Digital image processing, 2002. 2

[10] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973. 2

[11] Junjun He, Zhongying Deng, and Yu Qiao. Dynamic multi-scale filters for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3562–3572, 2019. 8

[12] Hanzhe Hu, Deyi Ji, Weihao Gan, Shuai Bai, Wei Wu, and Junjie Yan. Class-wise dynamic graph convolution for semantic segmentation. *arXiv preprint arXiv:2007.09690*, 2020. 2

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2

[14] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 603–612, 2019. 7, 8

[15] Xiangtai Li, Xia Li, Li Zhang, Guangliang Cheng, Jianping Shi, Zhouchen Lin, Shaohua Tan, and Yunhai Tong. Improving semantic segmentation via decoupled body and edge supervision. *arXiv preprint arXiv:2007.10035*, 2020. 2

[16] Xia Li, Yibo Yang, Qijie Zhao, Tiancheng Shen, Zhouchen Lin, and Hong Liu. Spatial pyramid based graph reasoning for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8950–8959, 2020. 8

[17] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, Shaohua Tan, and Yunhai Tong. Semantic flow for fast and accurate scene parsing. In *European Conference on Computer Vision*, pages 775–793. Springer, 2020. 1, 2, 8

[18] Di Lin, Yuanfeng Ji, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Multi-scale context intertwining for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 603–619, 2018. 8

[19] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017. 1, 2, 8

[20] G Linda. Shapiro, george c. stockman. computer vision. 2001. 2

[21] Jianbo Liu, Junjun He, Yu Qiao, Jimmy S Ren, and Hongsheng Li. Learning to predict context-adaptive convolution for semantic segmentation. In *European Conference on Computer Vision*, pages 769–786. Springer, 2020. 2, 8

[22] Zhikang Liu and Lanyun Zhu. Label-guided attention distillation for lane segmentation. *Neurocomputing*, 2021. 2

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1, 2, 8

[24] Ayushman Ramola, Amit Kumar Shakya, and Dai Van Pham. Study of statistical methods for texture analysis and their modern evolutions. *Engineering Reports*, 2(4):e12149, 2020. 2

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1, 2

[26] Zhe Wang, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learnable histogram: Statistical context features for deep neural networks. In *European Conference on Computer Vision*, pages 246–262. Springer, 2016. 3

[27] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv:1605.06885*, 2016. 8

[28] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3684–3692, 2018. 2

[29] Changqian Yu, Jingbo Wang, Changxin Gao, Gang Yu, Chunhua Shen, and Nong Sang. Context prior for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12416–12425, 2020. 8

[30] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 325–341, 2018. 1, 2

[31] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. *arXiv preprint arXiv:1909.11065*, 2019. 8

[32] Fan Zhang, Yanqin Chen, Zhihang Li, Zhibin Hong, Jingtuo Liu, Feifei Ma, Junyu Han, and Errui Ding. Acfnet: Attentional class feature network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6798–6807, 2019. 2, 8

[33] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7151–7160, 2018. 3

[34] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7151–7160, 2018. 8

[35] Hang Zhang, Jia Xue, and Kristin Dana. Deep ten: Texture encoding network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 708–717, 2017. 3

[36] Hang Zhang, Han Zhang, Chenguang Wang, and Junyuan Xie. Co-occurrent features in semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 548–557, 2019. 8

[37] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 2, 6, 8

[38] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 267–283, 2018. 8

[39] Lanyun Zhu, Shiping Zhu, Xuanyi Liu, and Li Luo. Distance guided channel weighting for semantic segmentation. *arXiv preprint arXiv:2004.12679*, 2020. 2

[40] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 593–602, 2019. 2, 8