Expectation-Maximization Attention Networks for Semantic Segmentation

Xia Li[1,2], Zhisheng Zhong[2], Jianlong Wu[2,3], Yibo Yang[2,4], Zhouchen Lin[2], Hong Liu[1,✉]

[1] Key Laboratory of Machine Perception, Shenzhen Graduate School, Peking University

[2] Key Laboratory of Machine Perception (MOE), School of EECS, Peking University

[3] School of Computer Science and Technology, Shandong University

[4] Academy for Advanced Interdisciplinary Studies, Peking University

{ethanlee,zszhong,jlwu1992,ibo,zlin,hongliu}@pku.edu.cn

Abstract

*Self-attention mechanism has been widely used for various tasks. It is designed to compute the representation of each position by a weighted sum of the features at all positions. Thus, it can capture long-range relations for computer vision tasks. However, it is computationally consuming. Since the attention maps are computed w.r.t all other positions. In this paper, we formulate the attention mechanism into an expectation-maximization manner and iteratively estimate a much more compact set of bases upon which the attention maps are computed. By a weighted summation upon these bases, the resulting representation is low-rank and deprecates noisy information from the input. The proposed **E**xpectation-**M**aximization **A**ttention (EMA) module is robust to the variance of input and is also friendly in memory and computation. Moreover, we set up the bases maintenance and normalization methods to stabilize its training procedure. We conduct extensive experiments on popular semantic segmentation benchmarks including PASCAL VOC, PASCAL Context and COCO Stuff, on which we set new records*[1].

1. Introduction

Semantic segmentation is a fundamental and challenging problem of computer vision, whose goal is to assign a semantic category to each pixel of the image. It is critical for various tasks such as autonomous driving, image editing and robot sensing. In order to accomplish the semantic segmentation task effectively, we need to distinguish some confusing categories and take the appearance of different objects into account. For example, 'grass' and 'ground' have similar color in some cases and 'person' may have various scales, figures and clothes in different locations of the image. Meanwhile, the label space of the output is quite compact and the amount of the categories for a specific dataset is limited. Therefore, this task can be treated as projecting data points in a high-dimensional noisy space into a compact sub-space. The essence lies in de-noising these variation and capturing the most important semantic concepts.

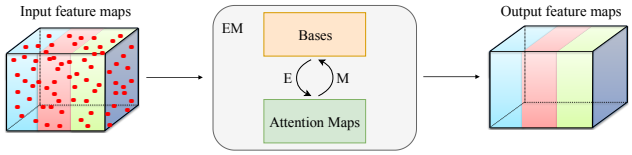

Figure 1: Pipeline of the proposed expectation-maximization attention method.

Recently, many state-of-the-art methods based on fully convolutional networks (FCNs) [22] have been proposed to address the above issues. Due to the fixed geometric structures, they are inherently limited by local receptive fields and short-range contextual information. To capture long-range dependencies, several works employ the multi-scale context fusion [17], such as astrous convolution [4], spatial pyramid [37], large kernel convolution [25] and so on. Moreover, to keep more detailed information, the encoder-decoder structures [34, 5] are proposed to fuse mid-level and high-level semantic features. To aggregate information from all spatial locations, attention mechanism [29, 38, 31] is used, which enables the feature of a single pixel to fuse information from all other positions. However, the original attention-based methods need to generate a large attention map, which has high computation complexity and occupies a huge number of GPU memory. The bottleneck lies in that both the generation of attention map and its usage are computed w.r.t all positions.

Towards the above issues, in this paper, we rethink the attention mechanism from the view of expectation-maximization (EM) algorithm [7] and propose a novel attention-based method, namely **E**xpectation-**M**aximization **A**ttention (EMA). Instead of treating all pixels themselves as the reconstruction bases [38, 31], we

[1]Project address: https://xialipku.github.io/EMANet

use the EM algorithm to find a more compact basis set, which can largely reduce the computational complexity. In detail, we regard the bases for construction as the parameters to learn in the EM algorithm and attention maps as latent variables. In this setting, the EM algorithm aims to find a maximum likelihood estimate of parameters (bases). Given the current parameters, the expectation (E) step works as estimating the expectation of attention map and maximization (M) step functions as updating the parameters (bases) by maximizing the complete data likelihood. The E step and the M step execute alternately. After convergence, the output can be computed as the weighted sum of bases, where the weights are the normalized final attention maps. The pipeline of EMA is shown in Fig. 1.

We further embed the proposed EMA method into a module for neural network, which is named EMA Unit. EMA Unit can be simply implemented by common operators. It is also light-weighted and can be easily embedded into existing neural networks. Moreover, to make full use of its capacity, we also propose two more methods to stabilize the training process of EMA Unit. We also evaluate its performance on three challenging datasets.

The main contributions of this paper are listed as follows:

- We reformulate the self-attention mechanism into an expectation-maximization iteration manner, which can learn a more compact basis set and largely reduce the computational complexity. To the best of our knowledge, this paper is the first to introduce EM iterations into attention mechanism.
- We build the proposed expectation-maximization attention as a light-weighted module for neural network and set up specific manners for bases' maintenance and normalization.
- Extensive experiments on three challenging semantic segmentation datasets, including PASCAL VOC, PASCAL Context and COCO Stuff, demonstrate the superiority of our approach over other state-of-the-art methods.

2. Related Works

Semantic segmentation. Fully convolutional network (FCN) [22] based methods have made great progress in image semantic segmentation by leveraging the powerful convolutional features of classification networks [14, 15, 33] pre-trained on large-scale data [28]. Several model variants are proposed to enhance the multi-scale contextual aggregation. For example, DeeplabV2 [4] makes use of the astrous spatial pyramid pooling (ASPP) to embed contextual information, which consists of parallel dilated convolutions with different dilated rates. DeeplabV3 [4] extends ASPP with image-level feature to further capture global contexts. Meanwhile, PSPNet [37] proposes a pyramid pooling module to collect contextual information of different scales. GCN [25] adopts decoupling of large kernel convolution to gain a large receptive field for the feature map and capture long-range information.

For the other type of variants, they mainly focus on predicting more detailed output. These methods are based on U-Net [27], which combines the advantages of high-level features with mid-level features. RefineNet [21] makes use of the Laplacian image pyramid to explicitly capture the information available along the down-sampling process and output predictions from coarse to fine. DeeplabV3+ [5] adds a decoder upon DeeplabV3 to refine the segmentation results especially along object boundaries. Exfuse [36] proposes a new framework to bridge the gap between low-level and high-level features and thus improves the segmentation quality.

Attention model. Attention is widely used for various tasks such as machine translation, visual question answering and video classification. The self-attention methods [2, 29] calculate the context coding at one position by a weighted summation of embeddings at all positions in sentences. Non-local [31] first adopts self-attention mechanism as a module for computer vision tasks, such as video classification, object detection and instance segmentation. PSANet [38] learns to aggregate contextual information for each position via a predicted attention map. A^2Net [6] proposes the double attention block to distribute and gather informative global features from the entire spatio-temporal space of the images. DANet [11] applies both spatial and channel attention to gather information around the feature maps, which costs even more computation and memory than the Non-local method.

Our approach is motivated by the success of attention in the above works. We rethink the attention mechanism from the view of the EM algorithm and compute the attention map in an iterative manner as the EM algorithm.

3. Preliminaries

Before introducing our proposed method, we first review three highly correlated methods, that is the EM algorithm, the Gaussian mixture model and the Non-local module.

3.1. Expectation-Maximization Algorithm

The expectation-maximization (EM) [7] algorithm aims to find the maximum likelihood solution for latent variables models. Denote $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ as the data set which consists of N observed samples and each data point $\mathbf{x}_i$ has its corresponding latent variable $\mathbf{z}_i$. We call $\{\mathbf{X}, \mathbf{Z}\}$ the complete data and its likelihood function takes the form $\ln p\left(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}\right)$, where $\boldsymbol{\theta}$ is the set of all parameters of the model. In practice, the only knowledge of latent variables in $\mathbf{Z}$ is given by the posterior distribution $p\left(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}\right)$.

The EM algorithm is designed to maximize the likelihood $\ln p\left(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}\right)$ by two steps, i.e., the E step and the M step.

In the E step, we use the current parameters $\boldsymbol{\theta}^{\text{old}}$ to find the posterior distribution of $\mathbf{Z}$ given by $p\left(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}\right)$. Then we use the posterior distribution to find the expectation of the complete data likelihood $\mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}\right)$, which is given by:

$$\mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}\right) = \sum_{\mathbf{z}} p\left(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}\right) \ln p\left(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}\right). \tag{1}$$

Then in the M step, the revised parameter $\boldsymbol{\theta}^{\text{new}}$ is determined by maximizing the function:

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}\right). \tag{2}$$

The EM algorithm executes the E step and the M step alternately until the convergence criterion is satisfied.

3.2. Gaussian Mixture Model

Gaussian mixture model (GMM) [26] is a special case of the EM algorithm. It takes the distribution of data $\mathbf{x}_n$ as a linear superposition of Gaussians:

$$p\left(\mathbf{x}_n\right) = \sum_{k=1}^{K} z_{nk} \mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right), \tag{3}$$

where the mean $\boldsymbol{\mu}_k$ and the covariance $\boldsymbol{\Sigma}_k$ are parameters for the k-th Gaussian basis. Here we leave out the prior π_k. The likelihood of the complete data is formulated as:

$$\ln p\left(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \sum_{n=1}^{N} \ln \left[\sum_{k=1}^{K} z_{nk} \mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right], \tag{4}$$

where $\sum_k z_{nk} = 1$. z_{nk} can be viewed as the responsibility that the k-th basis takes for the observation $\mathbf{x}_n$. For GMM, in the E step, the expected value of z_{nk} is given by:

$$z_{nk}^{\text{new}} = \frac{\mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_k^{\text{new}}, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_j^{\text{new}}, \boldsymbol{\Sigma}_j\right)}. \tag{5}$$

In the M step, the parameters are re-estimated as follows:

$$\begin{aligned} \boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} z_{nk}^{\text{new}} \mathbf{x}_n, \\ \boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} z_{nk}^{\text{new}} \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{old}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{old}}\right)^{\top}, \end{aligned} \tag{6}$$

where

$$N_k = \sum_{n=1}^{N} z_{nk}^{\text{new}}. \tag{7}$$

After the convergence of the GMM parameters, the re-estimated $\mathbf{x}_n^{\text{new}}$ can be formulated as:

$$\mathbf{x}_n^{\text{new}} = \sum_{k=1}^{K} z_{nk}^{\text{new}} \boldsymbol{\mu}_k^{\text{new}}. \tag{8}$$

In real applications, we can simply replace $\boldsymbol{\Sigma}_k$ as the identity matrix $\mathbf{I}$ and leave out the $\boldsymbol{\Sigma}_k$ in the above equations.

3.3. Non-local

The Non-local module [31] functions the same as the self-attention mechanism. It can be formulated as:

$$\mathbf{y}_i = \frac{1}{\mathcal{C}\left(\mathbf{x}_i\right)} \sum_{j} f\left(\mathbf{x}_i, \mathbf{x}_j\right) g\left(\mathbf{x}_j\right), \tag{9}$$

where $f\left(\cdot, \cdot\right)$ represents a general kernel function, $\mathcal{C}\left(\mathbf{x}\right)$ is a normalization factor and $\mathbf{x}_i$ denotes the feature vector for the location i. As this module is applied upon the feature map of convolutional neural networks (CNN).

Considering that $\mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$ in Eq. (5) is a specific kernel function between $\mathbf{x}_n$ and $\boldsymbol{\mu}_k$, Eq. (8) is just a specific design of Eq. (9). Then, from the viewpoint of GMM, the Non-local module is just a re-estimation of $\mathbf{X}$, without E steps and M steps. Specifically, $\boldsymbol{\mu}$ is just selected as the $\mathbf{X}$ in Non-local.

In GMM, the number of Gaussian bases is selected manually and usually satisfies $K \ll N$. But in the Non-local module, the bases are selected as the data themselves, so it has $K = N$. There are two obvious disadvantages of the Non-local module. First, the data are lying in a low dimensional manifold, so the bases are over-complete. Second, the computation overhead is heavy and the memory cost is also large.

4. Expectation-Maximization Attention

In view of the high computational complexity of the attention mechanism and limitations of the Non-local module, we first propose the expectation-maximization attention (EMA) method, which is an augmented version of self-attention. Unlike the Non-local module that selects all data points as bases, we use the EM iterations to find a compact basis set.

For simplicity, we consider an input feature map $\mathbf{X}$ of size $C \times H \times W$ from a single sample. $\mathbf{X}$ is the intermediate activations of a CNN. To simplify the symbols, we reshape $\mathbf{X}$ into $N \times C$, where $N = H \times W$, and $\mathbf{x}_i \in \mathbb{R}^C$ indexes the C dimensional feature vector at pixel i. Our proposed EMA consists of three operations, including **responsibility estimation** (A_{E}), **likelihood maximization** (A_{M}) and **data re-estimation** (A_{R}). Briefly, given the input $\mathbf{X} \in \mathbb{R}^{N \times C}$ and the initial bases $\boldsymbol{\mu} \in \mathbb{R}^{K \times C}$, A_{E} estimates the latent variables (or the 'responsibility') $\mathbf{Z} \in \mathbb{R}^{N \times K}$, so it functions as the E step in the EM algorithm. A_{M} uses the estimation to update the bases $\boldsymbol{\mu}$, which works as the M step. The A_{E} and A_{M} steps execute alternately for a pre-specified number of iterations. Then, with the converged $\boldsymbol{\mu}$ and $\mathbf{Z}$, A_{R} reconstructs the original $\mathbf{X}$ as $\mathbf{Y}$ and outputs it.

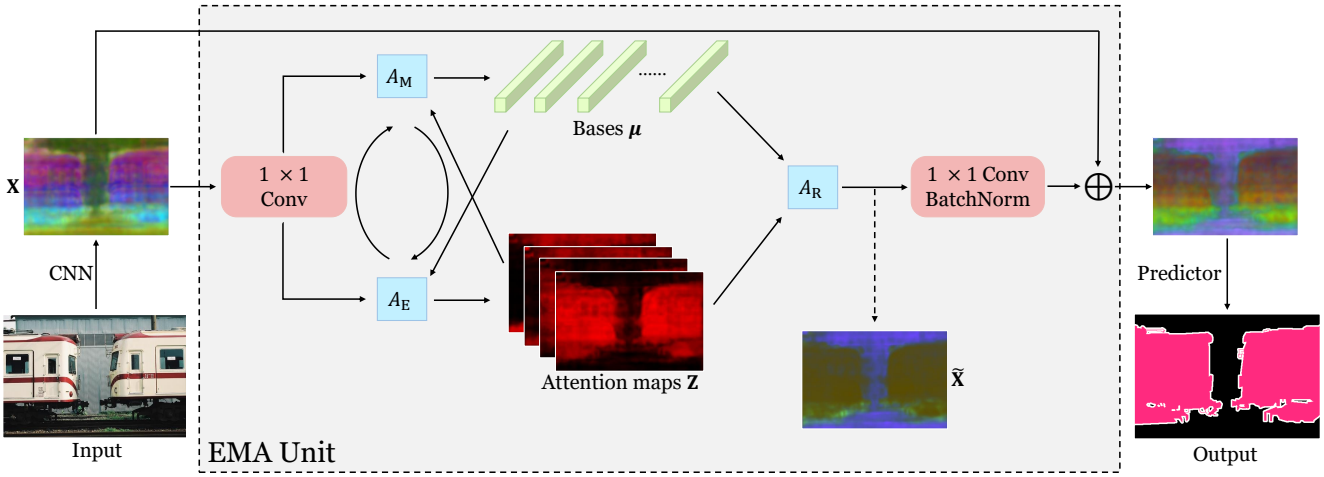

Figure 2: Overall structure of the proposed EMAU. The key component is the EMA operator, in which A_{E} and A_{M} execute alternately. In addition to the EMA operator, we add two 1×1 convolutions at the beginning and the end of EMA and sum the output with original input, to form a residual-like block. *Best viewed on screen.*

It has been proved that, with the iteration of EM steps, the complete data likelihood $\ln p(\mathbf{X}, \mathbf{Z})$ will increase monotonically. As $\ln p(\mathbf{X})$ can be estimated by marginalizing $\ln p(\mathbf{X}, \mathbf{Z})$ with $\mathbf{Z}$, maximizing $\ln p(\mathbf{X}, \mathbf{Z})$ is a proxy of maximizing $\ln p(\mathbf{X})$. Therefore, with the iterations of A_{E} and A_{M}, the updated $\mathbf{Z}$ and $\boldsymbol{\mu}$ have better ability to reconstruct the original data $\mathbf{X}$. The reconstructed $\tilde{\mathbf{X}}$ can capture important semantics from $\mathbf{X}$ as much as possible.

Moreover, compared with the Non-local module, EMA finds a compact set of bases for pixels of an input image. The compactness is non-trivial. Since $K \ll N$, $\tilde{\mathbf{X}}$ lies in a subspace of $\mathbf{X}$. This mechanism removes much unnecessary noise and makes the final classification of each pixel more tractable. Moreover, this operation reduces the complexity (both in space and time) from $O\left(N^2\right)$ to $O(NKT)$, where T is the number of iterations for A_{E} and A_{M}. The convergence of EM algorithm is also guaranteed. Notably, EMA takes only three iterations to get promising results in our experiments. So T can be treated as a small constant, which means that the complexity is only $O(NK)$.

4.1. Responsibility Estimation

Responsibility estimation (A_{E}) functions as the E step in the EM algorithm. This step computes the expected value of z_{nk}, which corresponds to the responsibility of the k-th basis $\boldsymbol{\mu}$ to $\mathbf{x}_n$, where $1 \leq k \leq K$ and $1 \leq n \leq N$. We formulate the posterior probability of $\mathbf{x}_n$ given $\boldsymbol{\mu}_k$ as follows:

$$p(\mathbf{x}_n | \boldsymbol{\mu}_k) = \mathcal{K}(\mathbf{x}_n, \boldsymbol{\mu}_k), \tag{10}$$

where $\mathcal{K}$ represents the general kernel function. And now, Eq. (5) can be reformulated into a more general form:

$$z_{nk} = \frac{\mathcal{K}(\mathbf{x}_n, \boldsymbol{\mu}_k)}{\sum_{j=1}^{K} \mathcal{K}(\mathbf{x}_n, \boldsymbol{\mu}_j)}. \tag{11}$$

There are several choices for $\mathcal{K}(\mathbf{a}, \mathbf{b})$, such as inner dot $\mathbf{a}^\top \mathbf{b}$, exponential inner dot $\exp\left(\mathbf{a}^\top \mathbf{b}\right)$, Euclidean distance $\|\mathbf{a} - \mathbf{b}\|_2^2$, RBF kernel $\exp\left(-\|\mathbf{a} - \mathbf{b}\|_2^2 / \sigma^2\right)$ and so on. As compared in the Non-local module, the choice of these functions makes trivial differences in the final results. So we simply take the exponential inner dot $\exp\left(\mathbf{a}^\top \mathbf{b}\right)$ in our paper. In experiments, Eq. (11) can be implemented as a matrix multiplication plus one softmax layer. In conclusion, the operation of A_{E} in the t-th iteration is formulated as:

$$\mathbf{Z}^{(t)} = \mathrm{softmax}\left(\lambda \mathbf{X} \left(\boldsymbol{\mu}^{(t-1)}\right)^\top\right), \tag{12}$$

where λ is a hyper-parameter to control the distribution of $\mathbf{Z}$.

4.2. Likelihood Maximization

Likelihood maximization (A_{M}) works as the EM algorithm's M step. With the estimated $\mathbf{Z}$, A_{M} updates $\boldsymbol{\mu}$ by maximizing the complete data likelihood. To keep the bases lying in the same embedding space as $\mathbf{X}$, we update the bases $\boldsymbol{\mu}$ using the weighted summation of $\mathbf{X}$. So $\boldsymbol{\mu}_k$ is updated as

$$\boldsymbol{\mu}_k^{(t)} = \frac{z_{nk}^{(t)} \mathbf{x}_n}{\sum_{m=1}^{N} z_{mk}^{(t)}} \tag{13}$$

in the t-th iteration of A_{M}.

It is noteworthy that if we set $\lambda \to \infty$ in Eq. (12), then $\{z_{n1}, z_{n2}, \cdots, z_{nK}\}$ will become a one-hot embedding. In this situation, each pixel is assigned to only one basis. And the basis is updated by the average of those pixels assigned to it. This is what the K-means clustering algorithm [10] does. So the iterations of A_{E} and A_{M} can also be viewed as a soft version of K-means clustering.

4.3. Data Re-estimation

EMA runs A_{E} and A_{M} alternately for T times. After that, the final $\boldsymbol{\mu}^{(T)}$ and $\mathbf{Z}^{(T)}$ are used to re-estimate the $\mathbf{X}$. We adopt Eq. (8) to construct the new $\mathbf{X}$, namely $\tilde{\mathbf{X}}$, which is formulated as:

$$\tilde{\mathbf{X}} = \mathbf{Z}^{(T)}\boldsymbol{\mu}^{(T)}. \tag{14}$$

As $\tilde{\mathbf{X}}$ is constructed from a compact basis set, it has the low-rank property compared with the input $\mathbf{X}$. We depict an example of $\tilde{\mathbf{X}}$ in Fig. 2. It's obvious that $\tilde{\mathbf{X}}$ outputed from A_{R} is very compact in the feature space and the feature variance inside the object is smaller than that of the input.

5. EMA Unit

In order to better incorporate the proposed EMA with deep neural networks, we further propose the Expectation-maximization Attention Unit (EMAU) and apply it to semantic segmentation task. In this section, we will describe EMAU in detail. We first introduce the overall structure of EMAU and then discuss bases' maintenance and normalization mechanisms.

5.1. Structure of EMA Unit

The overall structure of EMAU is shown in Fig. 2. EMAU looks like the bottleneck of ResNet at the first glance, except it replaces the heavy 3×3 convolution with the EMA operations. The first convolution without the ReLU activation is prepended to transform the value range of input from $(0, +\infty)$ to $(-\infty, +\infty)$. This transformation is very important, or the estimated $\boldsymbol{\mu}^{(T)}$ will also lie in $[0, +\infty)$, which halves the capacity compared with general convolution parameters. The last 1×1 convolution is inserted to transform the re-estimated $\tilde{\mathbf{X}}$ into the residual space of $\mathbf{X}$.

For each of A_{E}, A_{M} and A_{R} steps, the computation complexity is $O(NKC)$. As we set $K \ll C$, several iterations of A_{E} and A_{M} plus one A_{R} is just the same magnitude as a 1×1 convolution with input and output channel numbers all being C. Adding the extra computation from two 1×1 convolutions, the whole FLOPs of EMAU is around $1/3$ of a module running 3×3 convolutions with the same number of input and output channels. Moreover, the parameters maintained by EMA just counts to KC.

5.2. Bases Maintenance

Another issue for the EM algorithm is the initialization of the bases. The EM algorithm is guaranteed to converge, because the likelihood of complete data is limited, and at each iteration both E and M steps lift its current lower bound. However, converging to global maximum is not guaranteed. Thus, the initial values of bases before iterations are of great importance.

We only describe how EMA is used to process one image above. However, for a computer vision task, there are thousand of images in a dataset. As each image $\mathbf{X}$ has different pixel feature distributions from others, it is not suitable to use the $\boldsymbol{\mu}$ computed upon an image to reconstruct feature maps of other images. So we run EMA on each image.

For the first mini-batch, we initialize $\boldsymbol{\mu}^{(0)}$ using Kaiming's initialization [13], where we treat matrix multiplication as a 1×1 convolution. For the following mini-batches, one simple choice is to update $\boldsymbol{\mu}^{(0)}$ using standard back propagation. However, as iterations of A_E and A_M can be unrolled as a recurrent neural network (RNN), the gradients propagating though them will encounter the vanishing or explosion problem. Therefore, the updating of $\boldsymbol{\mu}^{(0)}$ is unstable, and the training procedure of EMA Unit may collapse.

In this paper, we use the moving averaging to update $\boldsymbol{\mu}^{(0)}$ in the training process. After iterating over an image, the generated $\boldsymbol{\mu}^{(T)}$ can be regarded as a biased update of $\boldsymbol{\mu}^{(0)}$, where the bias comes from the image sampling process. To make it less biased, we first average $\boldsymbol{\mu}^{(T)}$ over a mini-batch and get the $\bar{\boldsymbol{\mu}}^{(T)}$. Then we update $\boldsymbol{\mu}^{(0)}$ as:

$$\boldsymbol{\mu}^{(0)} \leftarrow \alpha\boldsymbol{\mu}^{(0)} + (1-\alpha)\,\bar{\boldsymbol{\mu}}^{(T)}, \tag{15}$$

where $\alpha \in [0, 1]$ is the momentum. For inference, the $\boldsymbol{\mu}^{(0)}$ keeps fixed. This moving averaging mechanism is also adopted in batch normalization (BN) [16].

5.3. Bases Normalization

In the above subsection, we accomplish the maintenance of $\boldsymbol{\mu}^{(0)}$ for each mini-batch. However, the stable update of $\boldsymbol{\mu}^{(t)}$ inside A_{E} and A_{M} iterations is still not guaranteed, due to the defect of RNN. The moving averaging mechanism described above requires $\bar{\boldsymbol{\mu}}^{(T)}$ not to differ significantly from $\boldsymbol{\mu}^{(0)}$, otherwise it will also collapse like back-propagation. This requirement also constrains the value range of $\boldsymbol{\mu}^{(t)}, 1 \le t \le T$.

To this end, we need to apply normalization upon $\boldsymbol{\mu}^{(t)}$. At the first glance, BN or layer normalization (LN) [1] sound to be good choices. However, these aforementioned normalization methods will change the direction of each basis $\boldsymbol{\mu}_k^{(t)}$, which changes their properties and semantic meanings. To keep the direction of each basis untouched, we choose Euclidean normalization (L2Norm), which divides each $\boldsymbol{\mu}_k^{(t)}$ by its length. By applying it, $\boldsymbol{\mu}^{(t)}$ then lies in a K-dimensional united hyper-sphere, and sequence of $\left\{\boldsymbol{\mu}_k^{(0)}, \boldsymbol{\mu}_k^{(1)}, \cdots, \boldsymbol{\mu}_k^{(T)}\right\}$ forms a trajectory on it.

5.4. Compare with the Double Attention Block

A^2 Net [6] proposes the double attention block (A^2 block), in which the output $\mathbf{Y}$ is computed as:

$$\mathbf{Y} = \left[\phi(\mathbf{X}, W_\phi)\,\mathrm{sfm}(\theta(\mathbf{X}, W_\theta))^\top\right]\mathrm{sfm}(\rho(\mathbf{X}, W_\rho)), \tag{16}$$

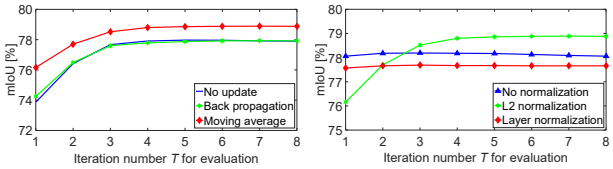

Figure 3: Ablation study on strategy of bases maintenance (left) and normalization (right) of EMAU. Experiments are carried out upon ResNet-50 with batch size 12 and training output stride 16 on the PASCAL VOC dataset. The iteration number T for training is set as 3. *Best viewed on screen.*

where sfm represents the $\mathrm{softmax}$ function. ϕ, θ and ρ represent three 1×1 convolutions with convolution kernels W_ϕ, W_θ and W_ρ, respectively.

If we share parameters between θ and ρ, then we can mark both W_θ and W_ρ as $\boldsymbol{\mu}$. We can see that $\mathrm{sfm}\left(\theta\left(\mathbf{X}, W_\theta\right)\right)$ just computes $\mathbf{Z}$ the same as Eq. (5) and those variables lying inside $[\cdot]$ update $\boldsymbol{\mu}$. The whole process of A^2 block equals to EMA with only one iteration. The W_θ in A^2 block is updated by the back-propagation, while our EMAU is updated by moving averaging. Above all, double attention block can be treated as a special form of EMAU.

6. Experiments

To evaluate the proposed EMAU, we conduct extensive experiments on the PASCAL VOC dataset [9], the PASCAL Context dataset [24], and the COCO Stuff dataset [3]. In this section, we first introduce implementation details. Then we perform ablation study to verify the superiority of proposed method on the PASCAL VOC dataset. Finally, we report our results on the PASCAL Context dataset and the COCO Stuff dataset.

6.1. Implementation Details

We use ResNet [14] (pretrained on ImageNet [28]) as our backbone. Following prior works [37, 4, 5], we employ a poly learning rate policy where the initial learning rate is multiplied by $(1-\mathrm{iter}/\mathrm{total\_iter})^{0.9}$ after each iteration. The initial learning rate is set to be 0.009 for all datasets. Momentum and weight decay coefficients are set to 0.9 and 0.0001, respectively. For data augmentation, we apply the common scale (0.5 to 2.0), cropping and flipping of the image to augment the training data. Input size for all datasets is set to 513×513. The synchronized batch normalization is adopted in all experiments, together with the multi-grid [4]. For evaluation, we adopt the commonly used Mean IoU metric.

The output stride of the backbone is set to 16 for training on PASCAL VOC and PASCAL Context, and 8 for training on COCO Stuff and evaluating on all datasets. To speed up the training procedure, we carry out all ablation studies on ResNet-50 [14], with batch size 12. For all models to be compared with state-of-the-art, we train them on ResNet-101, with batch size 16. We train 30K iterations on PASCAL VOC and COCO Stuff, and 15K on PASCAL Context. We use a 3×3 convolution to reduce the channel number from $2,048$ to 512, and then stack EMAU upon it. We call the whole network as **EMANet**. We set the basis number $K = 64$, $\lambda = 1$ and the number of iterations $T = 3$ for training as default.

Evaluation Iterations (mIoU %)

Training Iterations	1	2	3	4	5	6	7	8
1	77.34	77.52	77.60	77.59	77.59	77.59	77.59	77.59
2		77.75	78.04	78.15	78.15	78.12	78.12	78.17
3			78.52	78.80	78.86	78.88	78.89	78.88
4				78.14	78.25	78.27	78.28	78.27
5					77.70	77.76	77.82	77.86
6						77.85	77.91	77.92
7							77.11	77.14
8								77.24

Figure 4: Ablation study on the iteration number T. Experiments are conducted upon ResNet-50 with training output stride 16 and batch size 12 on the PASCAL VOC dataset.

6.2. Results on the PASCAL VOC Dataset

6.2.1 Bases Maintenance and Normalization

In this part, we first compare different strategies of maintaining $\boldsymbol{\mu}^{(0)}$. We set $T = 3$ in training, and $1 \leq T \leq 8$ in evaluation. As shown in the left part of Fig. 3, performance of all strategies increases with more iterations of A_{E} and A_{M}. When $T \geq 4$, the gain from more iterations becomes marginal. **Moving average** performs the best among them. It achieves the highest performances in all iterations and surpasses others by at least 0.9 in mIoU. Surprisingly, updating by the back propagation shows no merit compared with no updating and even performs worse when $T \geq 3$.

We then compare the performances with no normalization, LN and L2Norm as described above. From the right part of Fig. 3, it is clear to see that LN is even worse than no normalization. Since it can partially relieve the gradient chores of RNN-like structure. The performance of LN and no normalization has little correlation with the number of iteration T. By contrast, L2Norm's performance increases as the iterations become larger and it outperforms LN and no normalization when $T \geq 3$.

6.2.2 Ablation Study for Iteration Number

From Fig. 3, it is obvious that the performance of EMAU gain from more iterations during evaluation, and the gain becomes marginal when $T > 4$. In this subsection, we also study the influence of T in training. We plot the performance matrix upon T_{train} and T_{eval} as Fig. 4.

Table 1: Detailed comparisons on PASCAL VOC with DeeplabV3/V3+ and PSANet in mIoU (%). All results are achieved with the backbone ResNet-101 and output stride 8. The FLOPs and memory are computed with the input size 513×513. **SS**: Single scale input during test. **MS**: Multi-scale input. **Flip**: Adding left-right flipped input. EMANet (256) and EMANet (512) represent EMANet with the number of input channels as 256 and 512, respectively.

Method	SS	MS+Flip	FLOPs	Memory	Params
ResNet-101	-	-	190.6G	2.603G	42.6M
DeeplabV3 [4]	78.51	79.77	+63.4G	+66.0M	+15.5M
DeeplabV3+ [5]	79.35	80.57	+84.1G	+99.3M	+16.3M
PSANet [38]	78.51	79.77	+56.3G	+59.4M	+18.5M
EMANet (256)	79.73	80.94	**+21.1G**	**+12.3M**	**+4.87M**
EMANet (512)	**80.05**	**81.32**	+43.1G	+22.1M	+10.0M

Table 2: Comparisons on the PASCAL VOC test set.

Method	Backbone	mIoU (%)
Wide ResNet [32]	WideResNet-38	84.9
PSPNet [37]	ResNet-101	85.4
DeeplabV3 [4]	ResNet-101	85.7
PSANet [38]	ResNet-101	85.7
EncNet [35]	ResNet-101	85.9
DFN [34]	ResNet-101	86.2
Exfuse [36]	ResNet-101	86.2
IDW-CNN [30]	ResNet-101	86.3
SDN [12]	DenseNet-161	86.6
DIS [23]	ResNet-101	86.8
EMANet	ResNet-101	**87.7**
GCN [25]	ResNet-152	83.6
RefineNet [21]	ResNet-152	84.2
DeeplabV3+ [5]	Xception-71	87.8
Exfuse [36]	ResNeXt-131	87.9
MSCI [20]	ResNet-152	88.0
EMANet	ResNet-152	**88.2**

From Fig. 4, it is clear that mIoU increases monotonically with more iterations in evaluation, no matter what T_{train} is. They finally converge to a fixed value. However, this rule does not work in training. The mIoUs peak when $T_{\text{train}} = 3$ and decrease with more iterations. This phenomenon may be caused by the RNN-like behavior of EMAU. Though Moving Average and L2Norm can relieve to a certain degree, the problem persists.

We also carry out experiments on A^2 block [6], which can be regarded as a special form of EMAU as mentioned in Sec. 5.4. Similarly, the non-local module can also be viewed as a special form of EMAU without A_{M} step, which includes more bases and $T_{\text{train}} = 1$. With the same backbone and training scheduler, A^2 block achieves 77.41% and the non-local module achieves 77.78% in mIoU, respectively. As a comparison, EMANet achieves 77.34% when $T_{\text{train}} = 1$ and $T_{\text{eval}} = 1$. These three results have small differences, which is coincident with our analysis.

Table 3: Comparisons with state-of-the-art on the PASCAL Context test set. '+' means pretrained on COCO Stuff.

Method	Backbone	mIoU (%)
PSPNet [37]	ResNet-101	47.8
DANet [11]	ResNet-50	50.1
MSCI [20]	ResNet-152	50.3
EMANet	ResNet-50	50.5
SGR [18]	ResNet-101	50.8
CCL [8]	ResNet-101	51.6
EncNet [35]	ResNet-101	51.7
SGR+ [18]	ResNet-101	52.5
DANet [11]	ResNet-101	52.6
EMANet	ResNet-101	**53.1**

Table 4: Comparisons on the COCO Stuff test set.

Method	Backbone	mIoU (%)
RefineNet [21]	ResNet-101	33.6
CCL [8]	ResNet-101	35.7
DANet [11]	ResNet-50	37.2
DSSPN [19]	ResNet-101	37.3
EMANet	ResNet-50	37.6
SGR [18]	ResNet-101	39.1
DANet [11]	ResNet-101	39.7
EMANet	ResNet-101	**39.9**

6.2.3 Comparisons with State-of-the-arts

We first thoroughly compare EMANet with three baselines, namely DeeplabV3, DeeplabV3+ and PSANet on the validation set. We report mIoU, FLOPs, memory cost and parameter numbers in Tab. 1. We can see that EMANet outperforms these three baselines by a large margin. Moreover, EMANet is much lighter in computation and memory.

We further compare our method with existing methods on the PASCAL VOC test set. Following previous methods [4, 5], we train EMANet successively over COCO, the VOC *trainaug* and the VOC *trainval* set. We set the base learning rate as 0.009, 0.001 and 0.0001, respectively. We train 150K iterations on COCO, and 30K for the last two rounds. When inferring over the *test* set, we make use of multi-scale testing and left-right flipping.

As shown in Tab. 2, our EMANet sets the new record on PASCAL VOC, and improves DeeplabV3 [4] with the same backbone by **2.0%** in mIoU. Our EMANet achieves the best performance among networks with backbone ResNet-101, and outperforms the previous best one by **0.9%**, which is significant due to the fact that this benchmark is very competitive. Moreover, it achieves the performance that is comparable with methods based on some larger backbones.

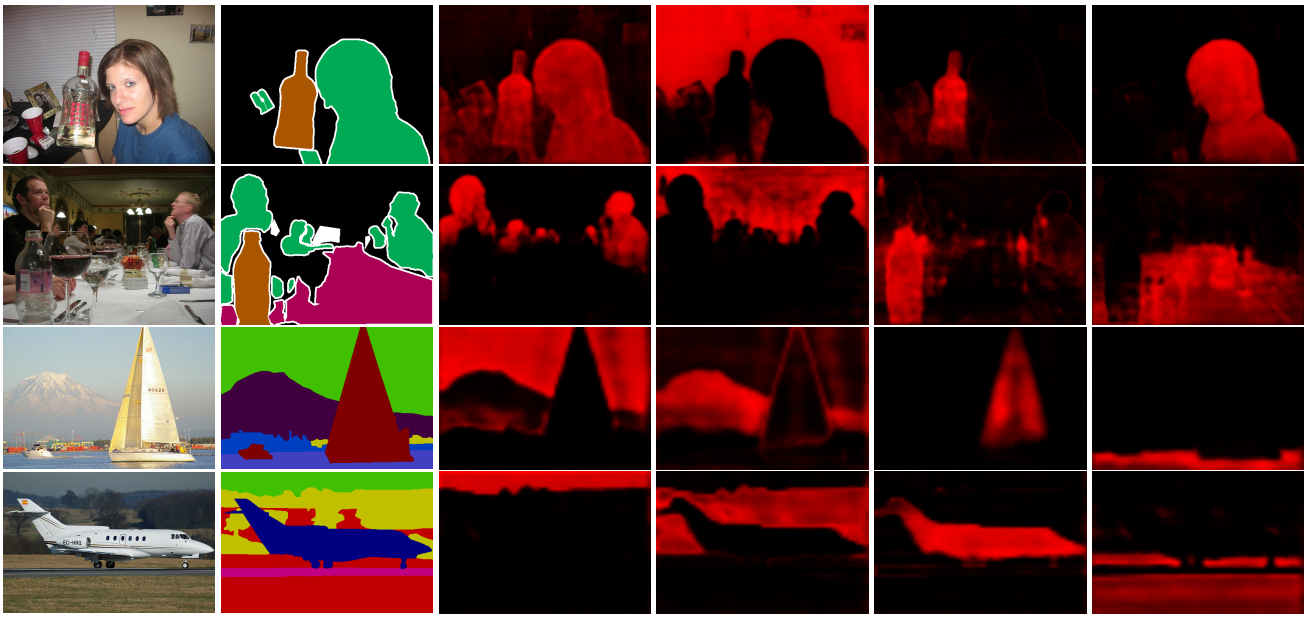

Figure 5: Visualization of responsibilities $\mathbf{Z}$ at the last iteration. The first two rows illustrate two examples from the PASCAL VOC validation set. The last two rows illustrate two examples from the PASCAL Context validation set. $z_{\cdot i}$ represents the responsibilities of the i-th basis to all pixels in the last iteration, i, j, k and l are four *randomly* selected indexes, where $1 \leq i, j, k, l \leq K$. *Best viewed on screen.*

6.3. Results on the PASCAL Context Dataset

To verify the generalization of our proposed EMANet, we conduct experiments on the PASCAL Context dataset. Quantitative results of PASCAL Context are shown in Tab. 3. To the best of our knowledge, EMANet based on ResNet-101 achieves the highest performance on the PASCAL Context dataset. Even pretrained on additional data (COCO Stuff), SGR+ is still inferior to EMANet.

6.4. Results on the COCO Stuff Dataset

To further evaluate the effectiveness of our method, we also carry out experiments on the COCO Stuff dataset. Comparisons with previous state-of-the-art methods are shown in Tab. 4. Remarkably, EMANet achieves 39.9% in mIoU and outperforms previous methods by a large margin.

6.5. Visualization of Bases Responsibilities

To get a deeper understanding of our proposed EMAU, we visualize the iterated responsibility map $\mathbf{Z}$ in Fig. 5. For each image, we randomly select four bases (i, j, k and l) and show their corresponding responsibilities of all pixels in the last iteration. Obviously, each basis corresponds to an abstract concept of the image. With the progress of iterations A_{E} and A_{M}, the abstract concept becomes more compact and clear. As we can see, these bases converge to some specific semantics and do not just focus on foreground and background. Concretely, the bases of the first two rows focus on specific semantics such as human, wine glass, cutlery and profile. The bases of the last two rows focus on sailboat, mountain, airplane and lane.

7. Conclusion

In this paper, we propose a new type of attention mechanism, namely the expectation-maximization attention (EMA), which computes a more compact basis set by iteratively executing as the EM algorithm. The reconstructed output of EMA is low-rank and robust to the variance of input. We well formulate the proposed method as a light-weighted module that can be easily inserted to existing CNNs with little overhead. Extensive experiments on a number of benchmark datasets demonstrate the effectiveness and efficiency of the proposed EMAU.

Acknowledgment

Zhouchen Lin is supported by National Basic Research Program of China (973 Program) (Grant no. 2015CB352502), National Natural Science Foundation (NSF) of China (Grant nos. 61625301 and 61731018), Qualcomm and Microsoft Research Asia. Hong Liu is supported by National Natural Science Foundation of China (Grant nos. U1613209 and 61673030) and funds from Shenzhen Key Laboratory for Intelligent Multimedia and Virtual Reality (ZDSYS201703031405467).

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *CVPR*, pages 1209–1218, 2018.

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

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[6] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. A2-nets: Double attention networks. In *NeurIPS*, pages 350–359, 2018.

[7] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[8] Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Context contrasted feature and gated multi-scale aggregation for scene segmentation. In *CVPR*, pages 2393–2402, 2018.

[9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[10] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21:768–769, 1965.

[11] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, pages 3146–3154, 2019.

[12] Jun Fu, Jing Liu, Yuhang Wang, Jin Zhou, Changyong Wang, and Hanqing Lu. Stacked deconvolutional network for semantic segmentation. *IEEE Transactions on Image Processing*, 2019.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[17] Xia Li, Jianlong Wu, Zhouchen Lin, Hong Liu, and Hongbin Zha. Recurrent squeeze-and-excitation context aggregation net for single image deraining. In *ECCV*, pages 254–269, 2018.

[18] Xiaodan Liang, Zhiting Hu, Hao Zhang, Liang Lin, and Eric P Xing. Symbolic graph reasoning meets convolutions. In *NeurIPS*, pages 1858–1868, 2018.

[19] Xiaodan Liang, Hongfei Zhou, and Eric Xing. Dynamic-structured semantic propagation network. In *CVPR*, pages 752–761, 2018.

[20] Di Lin, Yuanfeng Ji, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Multi-scale context intertwining for semantic segmentation. In *ECCV*, pages 603–619, 2018.

[21] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, pages 1925–1934, 2017.

[22] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[23] Ping Luo, Guangrun Wang, Liang Lin, and Xiaogang Wang. Deep dual learning for semantic image segmentation. In *ICCV*, pages 2718–2726, 2017.

[24] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, pages 891–898, 2014.

[25] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *CVPR*, pages 4353–4361, 2017.

[26] Sylvia Richardson and Peter J Green. On bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society: series B (statistical methodology)*, 59(4):731–792, 1997.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[30] Guangrun Wang, Ping Luo, Liang Lin, and Xiaogang Wang. Learning object interactions and descriptions for semantic image segmentation. In *CVPR*, pages 5859–5867, 2017.

[31] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018.

[32] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.

[33] Yibo Yang, Zhisheng Zhong, Tiancheng Shen, and Zhouchen Lin. Convolutional neural networks with alternately updated clique. In *CVPR*, pages 2413–2422, 2018.

[34] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Learning a discriminative feature network for semantic segmentation. In *CVPR*, pages 1857–1866, 2018.

[35] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, pages 7151–7160, 2018.

[36] Zhenli Zhang, Xiangyu Zhang, Chao Peng, Xiangyang Xue, and Jian Sun. Exfuse: Enhancing feature fusion for semantic segmentation. In *ECCV*, pages 269–284, 2018.

[37] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017.

[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 *ECCV*, pages 267–283, 2018.