FAKD: Feature Augmented Knowledge Distillation for Semantic Segmentation

Jianlong Yuan[1,2*†] Minh Hieu Phan[3*] Liyang Liu[3] Yifan Liu[3]
[1]Damo Academy, Alibaba Group [2]Hupan Lab [3]University of Adelaide

Discussion on the trade-off between training time and model's efficiency. We measured training time and inference speed (in seconds) on a V100 GPU. We used PSPNet-R101 as the teacher and PSPNet-R18 as the student, with a batch size of 4. The training (trn.) time for each iteration and inference (inf.) speed are shown in Tab. 1. Compared to CIRKD, our FAKD has a faster training time. Compared to CWD, our FAKD needs to compute covariance matrix, incurring extra training computing overhead. This computational aspect represents a limitation of our approach. However, it is essential to highlight that our method introduces no additional model parameters. After training, the student network still has the same real-time inference speed (0.089 seconds), while achieving a remarkable 4.38% increase in mIoU. In critical tasks such as edge computing for autonomous driving, our model still ensures high accuracy with real-time inference, justifying the extended training time in precision-demanding applications.

Table 1. Training time (in seconds) and other metrics for PSPNet-R101 (teacher) and PSPNet-R18 (student) with a batch size of 4 on a Nvidia V100 GPU.

Methods	Trn. time(s)	Inf. time (s)	mIoU
SKDS	0.348	0.089	29.42
IFVD	0.461	0.089	32.15
CIRKD	0.537	0.089	32.25
CWD	0.348	0.089	33.82
Ours	0.505	0.089	35.30

Derivation of the surrogate loss for implicit feature augmentation. This section derives the upper-bound of the loss objectives for implicit infinite data augmentations, which is Proposition 1 in Section 3:

Proposition 1. *Suppose that $\tilde{S}_i \sim \mathcal{N}(s_i, \lambda_i \Sigma_i)$, we have*

$$L_{aug} \leq \frac{\tau^2}{C} \sum_{i=1}^{M} \sum_{c=1}^{C} O^T_{i,c} \log\Big\{ \sum_{k=1}^{M} \exp\big[\frac{w_c^\top (S_i - S_k)}{\tau} + \frac{w_c^\top (\lambda_c \Sigma_i + \lambda_k \Sigma_k) w_c}{2\tau}\big]\Big\}. \tag{1}$$

Remind that

$$L_{\text{aug}}^{\text{CWD}} = -\frac{\tau^2}{M} \sum_{i=1}^{M} \sum_{c=1}^{C} \mathbb{E}_{\tilde{S}_i}\big[O^T_{i,c} \log \sum_{k=1}^{M} \exp \frac{w_c^\top (\tilde{S}_i - \tilde{S}_k)}{\tau}\big]. \tag{2}$$

According to Jensen's Inequality, the loss in Eq. 2 has an upper bound be as follows

$$L_{aug} \leq \frac{\tau^2}{C} \sum_{i=1}^{M} \sum_{c=1}^{C} O^T_{i,c} \log \left[\mathbb{E}_{\tilde{S}_i} \left[\sum_{k=1}^{M} \exp \frac{w_c^\top (\tilde{S}_i - \tilde{S}_k)}{\tau}\right]\right]. \tag{3}$$

Following existing work [1, 2], we apply Gaussian distribution to approximate the distribution for deep features. Specifically, we assume that $\hat{S}_i \sim \mathcal{N}(S_i, \lambda_i \Sigma_i)$, where Σ_i are the covariance of the semantic distribution of the i-th example. The next section discusses how to estimate this covariance matrix Σ_i. With the assumption about distribution of $\hat{S}_i$, $\hat{S}_i - \hat{S}_k$ also follows the Gaussian distribution:

$$\frac{w_c^\top (\tilde{S}_i - \tilde{S}_k)}{\tau} \sim \mathcal{N}\left(\frac{w_c^\top (S_i - S_k)}{\tau}, \frac{w_c^\top (\lambda_i \Sigma_i + \lambda_k \Sigma_k) w_c}{\tau}\right). \tag{4}$$

For a variable x that follows Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, the moment-generating function shows that $\mathbb{E}[\exp(a^\top x)] = \exp(a^\top \mu + \frac{1}{2} a^\top \Sigma a)$. Therefore, by taking the statistics for each example, the upper-bound in Eq. 3

*Equal contribution
†Corresponding Author

can be simplified as

$$L_{\text{aug}}^{PD} \leq \frac{1}{M}\sum_{i=1}^{M}\sum_{c=1}^{C} O_{i,c}^{T}$$

$$\log\left\{\sum_{k=1}^{C}\exp\left[\Delta w_k S_i + b_k - b_c + \frac{\lambda}{2}\Delta w_k \Sigma_c \Delta w_k\right]\right\}, \tag{5}$$

where $\Delta w_k = w_k^\top - w_c^\top$. Eq. 5 is our final feature-augmentation loss for pixel-wise distillation.

Covariance estimation. Following [3], instead of random sampling, we approximate the human-annotated procedure by drawing random vectors from a zero-mean normal distribution with the covariance proportional to the intra-class covariance matrix of the pixel-wise sample to be augmented. The covariance matrix is a mode of the category conditional distribution that captures rich semantic knowledge as it encodes category-specific variation. We generate augmented students corresponding to s_i along the class modes. $y_i \in \{1, ..., C\}$ is the label of the i-th pixel sample x_i over C classes. First, we setup a zero-mean multi-variate normal distribution $N(0, \Sigma_{y_i})$, where Σ_{y_i} is the category conditional covariance matrix estimated from the deep features of all samples in y_i. We compute the matrices online by taking into account the statics of all mini-batches. Formally, the online estimation algorithm for the covariance matrices is given by:

$$\boldsymbol{\mu}_j^{(t)} = \frac{n_j^{(t-1)}\boldsymbol{\mu}_j^{(t-1)} + m_j^{(t)}\boldsymbol{\mu}'^{(t)}_j}{n_j^{(t-1)} + m_j^{(t)}} \tag{6}$$

$$\Sigma_j^{(t)} = \frac{n_j^{(t-1)}\Sigma_j^{(t-1)} + m_j^{(t)}\Sigma'^{(t)}_j}{n_j^{(t-1)} + m_j^{(t)}} + \frac{n_j^{(t-1)}m_j^{(t)}(\boldsymbol{\mu}_j^{(t-1)} - \boldsymbol{\mu}'^{(t)}_j)(\boldsymbol{\mu}_j^{(t-1)} - \boldsymbol{\mu}'^{(t)}_j)^T}{(n_j^{(t-1)} + m_j^{(t)})^2}, \tag{7}$$

$$n_j^{(t)} = n_j^{(t-1)} + m_j^{(t)} \tag{8}$$

where $\boldsymbol{\mu}_j^{(t)}$ and $\Sigma_j^{(t)}$ are the estimates of average values and covariance matrices of the features of j^{th} class at t^{th} step. $\boldsymbol{\mu}'^{(t)}_j$ and $\Sigma'^{(t)}_j$ are the average values and covariance matrices of the features of j^{th} class in t^{th} mini-batch. $n_j^{(t)}$ denotes the total number of training samples belonging to j^{th} class in all t mini-batches, and $m_j^{(t)}$ denotes the number of training samples belonging to j^{th} class only in t^{th} mini-batch.

Discussion. While ISDA [4] computes the covariance matrix from image-wise samples, which requires a large batch size to sufficiently capture the covariance matrix, our FAKD updates covariance matrix from pixel-wise samples. A single image contains sufficiently large number of samples (e.g., $512 \times 512 \approx 200$K pixel-level samples). Hence, a small batch size of 16 is sufficient to capture the meaningful covariance matrix.

Pesudo-Code for FAKD. Figure 5 shows the pytorch-based pseudo-code of FAKD. Figure 6 illustrates how to calculate Σ.

Ablation study of infinite teachers/students. Following ℓ_{aug}^{CWD}, we could replace infinite students with infinite teachers. We define a student as consistent with all of the teachers. As shown in Table 2, both methods could get improvement. Furthermore, infinite students have 0.07% improvement compared with infinite teachers.

Table 2. Experiment for infinite teachers/students. Based on ℓ_{aug}^{CWD}, an infinite number of teachers and students are introduced in parts.

Formula	mIoU	mAcc(%)
Infinite teachers	35.23	44.51
Infinite students	35.30	44.06

Table 3. Experiment with data augmentation. Compared with others, our method gets better performance.

equation	mIoU	mAcc(%)
CE	29.42	38.48
CE+CWD	33.82	42.41
CE+CWD+ISDA	34.67	43.46
CE+FAKD	35.30	44.06

Ablation study with data augmentation. As shown in Table 3, we also compared with ISDA [3,4] which is a method of data augmentation in supervised learning. We can see that FAKD could improve 0.63% compared with ISDA. So, introducing infinite samples in distillation has better performance.

References

[1] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015. 1

[2] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Proc. Adv. Neural Inform. Process. Syst.*, 30, 2017. 1

[3] Yulin Wang, Gao Huang, Shiji Song, Xuran Pan, Yitong Xia, and Cheng Wu. Regularizing deep networks with semantic data augmentation. *IEEE TPAMI*, 2021. 2

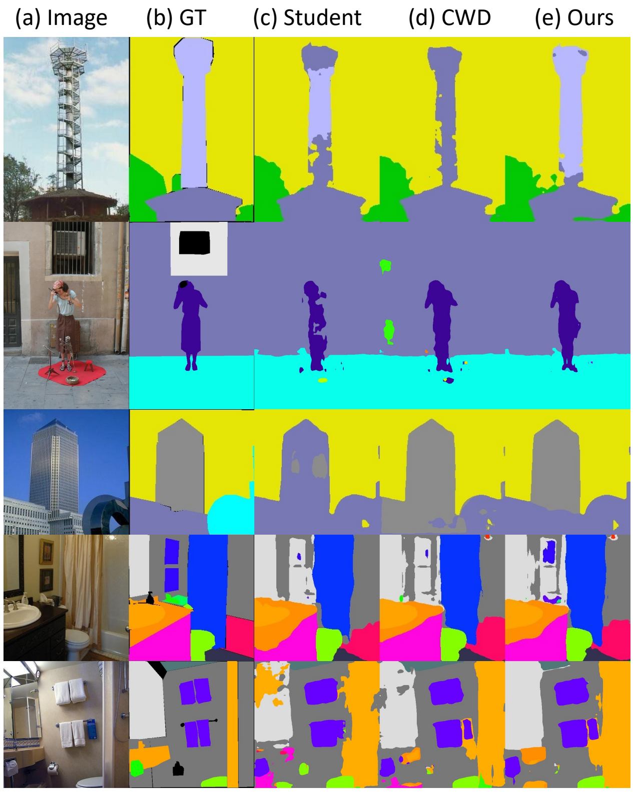

Figure 1. Qualitative segmentation results on the validation set of ADE20K using the PSPNet-ResNet18 network: (a) raw images, (b) ground truth, (c) student, (d) channel wise distillation, (e) our method FAKD.

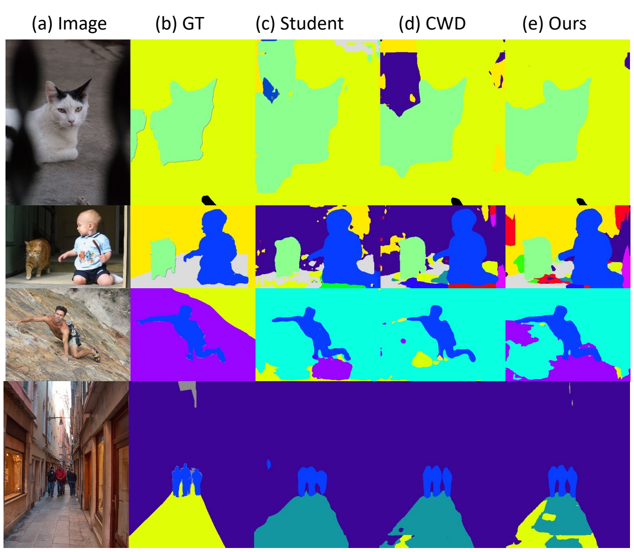

Figure 2. Qualitative segmentation results on the validation set of Pascal Context using the PSPNet-ResNet18 network: (a) raw images, (b) ground truth, (c) student, (d) channel wise distillation, (e) our method FAKD.

[4] Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. Implicit semantic data augmentation for deep networks. *Proc. Adv. Neural Inform. Process. Syst.*, 32:12635–12644, 2019. 2

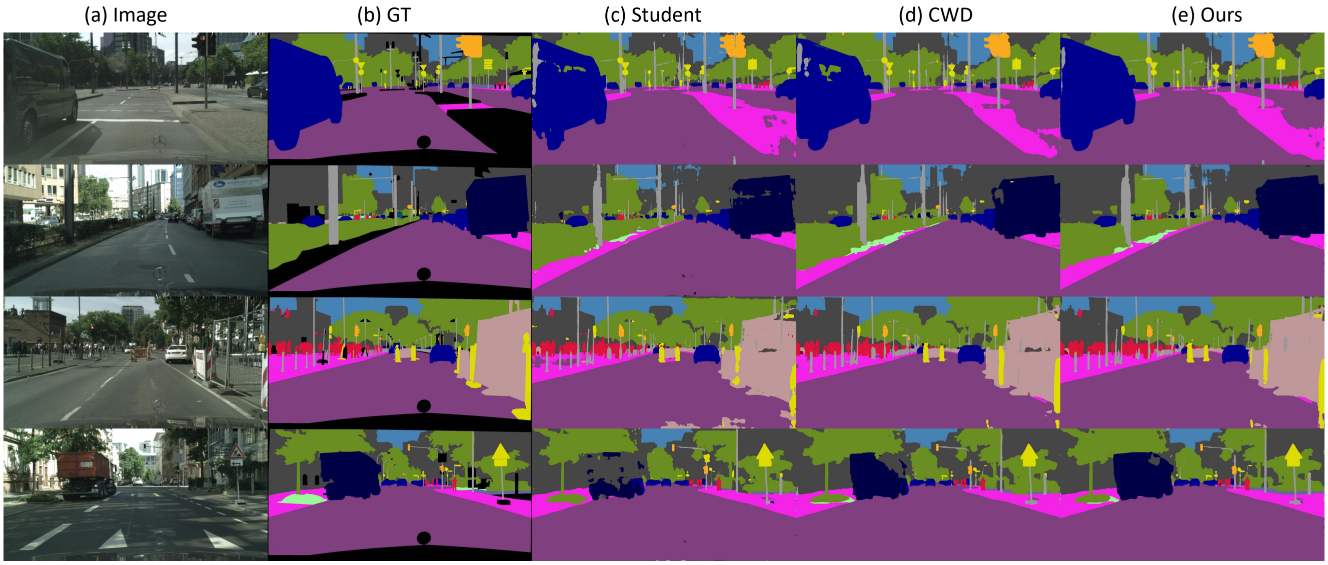

Figure 3. Qualitative segmentation results on the validation set of Cityscapes using the PSPNet-ResNet18 network: (a) raw images, (b) ground truth, (c) student, (d) channel wise distillation, (e) our method FAKD.

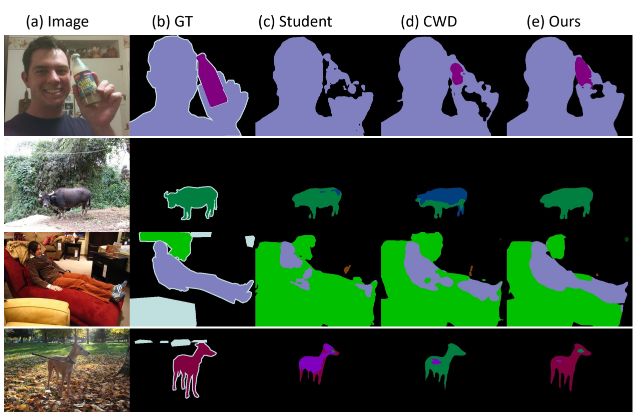

Figure 4. Qualitative segmentation results on the validation set of Pascal Voc using the PSPNet-ResNet18 network: (a) raw images, (b) ground truth, (c) student, (d) channel wise distillation, (e) our method FAKD.

```python
def FAKD(teacher_preds, student_preds, student_features, conv, gt, estimator, ratio):
    '''
        teacher_preds and student_preds are logits from the teacher and the student
        student_features is features from the studnet
        conv is classification layer
        tau is temprature
    '''
    N, C, W, H = student_preds.shape
    student_features = semantic_aug(student_preds, student_features, conv, gt, estimator, ratio)
    teacher_preds_softmax = F.softmax(teacher_preds.reshape(-1, W * H) / tau, dim=1)
    student_preds_logsoftmax = F.log_softmax(student_features.reshape(-1, W * H) / tau, dim=1)
    loss = torch.sum(- teacher_preds_softmax * student_preds_logsoftmax) * (tau ** 2)
    return loss_weight * loss / (C * N)

def semantic_aug(preds, features, conv, gt, estimator, ratio):
    '''
        N, A, H, W is the shape of features
        C is number classes
    '''
    gt = F.interpolate(gt, size=(H, W)).reshape(-1)
    features = features.permute(0, 2, 3, 1).reshape(-1, A)
    preds = preds.permute(0, 2, 3, 1).reshape(-1, C)
    with torch.no_grad():
        estimator(features, gt)
    sv = semantic_vector(conv, features, preds, gt, CoVariance, ratio).reshape(N, H, W, C).permute(0, 3, 1, 2)
    return sv

def semantic_vector(conv, features, preds, gt, CoVariance, ratio):
    gt_mask = gt == ignore_label
    labels = (1 - gt_mask).mul(gt)
    N, A, C = features.size(0), features.size(1), preds.shape[1]
    weight_m = list(conv.parameters())[0].squeeze()
    CV_temp = CoVariance[labels]
    sigma2 = ratio * weight_m.pow(2).mul(CV_temp.reshape(N, 1, A).expand(N, C, A)).sum(2)
    aug_result = preds + 0.5 * sigma2.mul((1 - gt_mask).reshape(N, 1).expand(N, C))
    return aug_result
```

Figure 5. Python code for FAKD based upon pytorch.

```python
def estimator(features, labels):
    '''
        C is class number
        CoVariance, Mean, Amount are the statistical values
    '''
    N, A = features.size()
    NxCxA_Features = features.view(N, 1, A).expand(N, C, A)
    onehot = torch.zeros(N, C)
    onehot.scatter_(1, labels.view(-1, 1), 1)
    NxCxA_onehot = onehot.view(N, C, 1).expand(N, C, A)
    features_by_sort = NxCxA_Features.mul(NxCxA_onehot)
    Amount_CxA = NxCxA_onehot.sum(0)
    Amount_CxA[Amount_CxA == 0] = 1
    mean_CxA = features_by_sort.sum(0) / Amount_CxA
    var_temp = features_by_sort - mean_CxA.expand(N, C, A).mul(NxCxA_onehot)
    var_temp = var_temp.pow(2).sum(0).div(Amount_CxA)
    sum_weight_CV = onehot.sum(0).view(C, 1).expand(C, A)
    weight_CV = sum_weight_CV.div(sum_weight_CV + self.Amount.view(C, 1).expand(C, A))
    weight_CV[weight_CV != weight_CV] = 0
    additional_CV = weight_CV.mul(1 - weight_CV).mul((Mean - mean_CxA).pow(2))
    CoVariance = (CoVariance.mul(1 - weight_CV) + var_temp.mul(weight_CV)) + additional_CV
    Mean = (Mean.mul(1 - weight_CV) + mean_CxA.mul(weight_CV))
    Amount = Amount + onehot.sum(0)
```

Figure 6. Python code for calculating Σ based upon pytorch.