Learning Debiased and Disentangled Representations for Semantic Segmentation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks are highly susceptible to learn biased and entangled feature representations, which may lead to sub par performances on tasks such as semantic segmentation. This is especially true for under-represented classes, where a lack of variety in the data can further exacerbate this issue. To this end, we propose a model-agnostic and stochastic training scheme that facilitates the learning of debiased and disentangled representations. First, we extract class-specific information from highly entangled features. Then, information related to a randomly sampled class is suppressed by class-specific feature selection in the feature space. By randomly eliminating certain class information in each training iteration, we effectively reduce feature dependencies among classes, and the model is able to learn more debiased and disentangled feature representations. Models trained with our method demonstrate strong results on multiple semantic segmentation benchmarks, with especially notable performance gains on under-represented classes.

1 Introduction

Semantic segmentation is a pixelwise classification task that is applicable to a wide range of applications including autonomous driving [1], medical image diagnosis [2], scene understanding [3], and many others. Ever since deep neural networks have been adopted for semantic segmentation [4, 5, 6, 7, 8], its accuracy has increased gradually with the introduction of stronger network architectures, improved training schemes, and large-scale datasets.

Despite such phenomenal achievement, semantic segmentation approaches still suffer from the chronic bias issue caused by class imbalance and stereotyped scene context in datasets. For example, the "road" class in the Cityscapes dataset [1] accounts for over 36% of pixels of all training images while the "motorcycle" class occupies only 0.1% of pixels. Such class imbalance issues are prevalent in semantic segmentation because certain classes appear more frequently than others, and the size of class may vary immensely. More importantly, the objects in the "motorcycle" class always appear in the vicinity of the "road" class, and the spatial layout of the two classes are almost identical in all images. Consequently, the models often learn substantially correlated representations and fail to accurately delineate the boundaries between frequently co-located objects.

These issues cause the model to learn sub-optimal feature representations, especially for classes that are under-represented. These feature representations are often entangled with one another, causing confusion between classes, and are also susceptible to encode adverse biases as well, which may be detrimental to the model's overall performance. On the other hand, for classes with abundant labels, the model is able to leverage the large variations in the data to learn more robust feature representations that are debiased and disentangled from other classes. As such, the model exhibits strong performance on classes with sufficient labels and weaker performances on those that are under-represented.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

We propose DropClass, a simple yet innovative training scheme for semantic segmentation that aims to alleviate the issue of feature entanglement and bias vulnerability. Given an input data sample, we first disentangle the feature representations in a class-specific manner. Then, the features corresponding to a randomly sampled class is dropped, and the remaining features are aggregated to generate the predictions. This procedure is somewhat similar to Dropout [9], but is specifically designed with a different goal in mind; while Dropout aims to reduce co-adaptations among features, DropClass aims to disentangle the class representations and reduce the underlying biases that arise from inter-class relationships. We also propose a new loss function that appropriately accompanies our training scheme. Finally, DropClass requires no additional data or annotations, and more importantly, is model agnostic, *i.e.,* can be applied to any existing network architecture.

Our contributions can be summarized as follows:

- We propose DropClass, a model-agnostic training scheme that results in learning more debiased and disentangled feature representations for semantic segmentation.
- Our experimental results show that the model trained with DropClass exhibits stronger performance compared to the baseline across multiple datasets and network architectures.
- We observe significant performance gains especially on under-represented classes, and provide analysis that further validate the effectiveness of our training scheme.

2 Proposed Method

We first define the semantic segmentation task and its relevant notations. For a dataset with a set of classes, $\mathcal{C}$, we sample an input image $x \in \mathbb{R}^{h \times w \times 3}$ and its corresponding label $y \in \mathbb{R}^{h \times w \times |\mathcal{C}|}$. The feature extractor g takes x as input and produces an intermediate feature representation $A \in \mathbb{R}^{h_1 \times w_1 \times k}$, where k denotes an arbitrary channel size of the feature map that depends on the choice of network architecture. A is then fed into the classifier, h, to produce $\hat{y} \in \mathbb{R}^{h \times w \times |C|}$, which serves as the final output of the model before the softmax operation, *i.e.,* the logits. Altogether, we denote the entire model as a composition of the feature extractor and classifier: $\hat{y} = h(A) = h(g(x))$. Note that throughout the paper, we use subscripts to refer to specific spatial and channel locations on tensors.

2.1 Debiased Feature Representations

Learning debiased representations is a crucial step towards building robust models that generalize well to new samples. Consider the class relationship between the "car" and "road" classes. The two classes have a strong spatial correlation, in that the vast majority of "car" samples will be spatially located above pixels of the "road" class. Semantic segmentation models can easily learn this spatial correlation — or, bias — in the process of maximizing the pixel-level accuracy. However, this could have adverse effects when the model encounters an image where cars do not appear above a road. In this sense, learning debiased representations helps the model's ability to generalize.

The fundamental goal of our proposed method is to develop a training scheme that allows the model to learn significant representations for each class, such that the model can make robust predictions without depending on the feature representations of other classes. In essence, this is achieved by removing the class-specific features of a random class at each iteration — as described in Section 2.3. Our training process not only breaks up inter-class relationships in a stochastic manner, but also exposes the classifier to a diverse set of class-specific feature combinations given the same input.

2.2 Disentangled Feature Representations

The feature representations of multiple classes should not be entangled with one another in order to facilitate less confusion and stronger discriminability. To achieve this, we remove the feature dependencies among classes by selectively discarding the information related to a randomly chosen class at each training iteration.

The difficulty is that neural networks are black-box in nature, and produce highly entangled feature representations. To alleviate this issue, we leverage Grad-CAM [10], a well-known gradient-based method of highlighting important class-specific information from entangled feature representations. Grad-CAM uses the backpropagated gradients to produce a coarse localization map that highlights

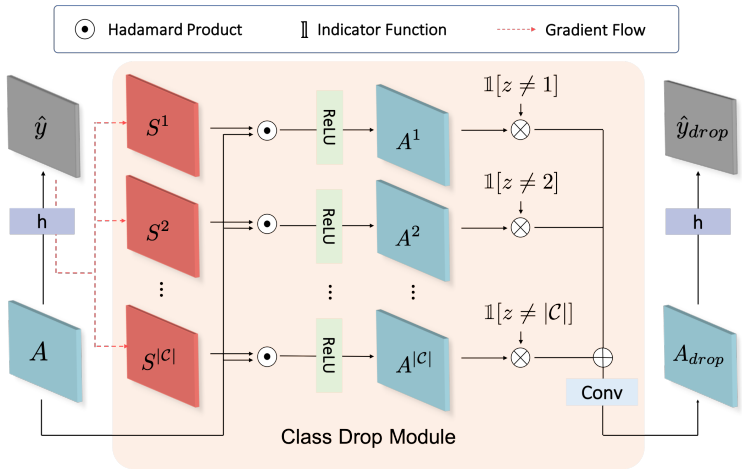

Figure 1: An illustration of the DropClass procedure. We generate importance weights $S^1, S^2, \ldots, S^{|\mathcal{C}|}$, which are used to extract class-specific feature maps $A^1, A^2, \ldots, A^{|\mathcal{C}|}$. We drop the class-specific feature map corresponding to a random class z, and average all other feature maps. Finally, A_{drop} is fed to the classifier to produce the output without information of class z.

important regions in the image, and is widely used to visualize class-discriminative features in deep neural networks. In similar fashion, we employ the gradients of feature $A_{i,j,k}$ with respect to the class logit $\hat{y}^c$ as the degree of importance that $A_{i,j,k}$ has on the prediction for class c. Since semantic segmentation is a pixel-level classification task, we average the gradients of $A_{i,j,k}$ with respect to every pixel location that corresponds to class c. This is used as the importance score of $A_{i,j,k}$ for class c, and is denoted by $S^c_{i,j,k}$:

$$S^c_{i,j,k} = \frac{1}{h \times w} \sum_{u=1}^{h} \sum_{v=1}^{w} \frac{\partial \hat{y}_{u,v,c}}{\partial A_{i,j,k}}. \tag{1}$$

The importance score for a certain class can be calculated for all locations i, j, k of the feature map A to obtain a importance score map, S^c with the same dimensions. Then, we perform element-wise multiplication of the feature and importance score maps to obtain class specific feature representations:

$$A^c = \text{ReLU}(A \odot S^c), \tag{2}$$

where $\odot$ denotes the Hadamard product, and ReLU denotes the ReLU activation function. The ReLU activation is used to preserve all features of A that have a positive influence on $\hat{y}^c$ and filter out all negative ones. A^c is the feature representation that highlights relevant information for class c, but the feature map specific to other classes may still contain the information about class c. Our approach suppresses the disentangled feature using the objective function as defined in Section 2.4.

2.3 DropClass for Debiasing and Disentangling

With the highlighted feature representation A^c, $\forall c \in \mathcal{C}$, we are now able to discard information related to a certain class from the output of the feature extractor. By doing so, the resulting feature is relieved from bias induced by the dropped class, and the model can focus on learning debiased representations.

In every iteration of the training process, a class z is randomly sampled as the class to drop:

$$Z \sim U(1, |\mathcal{C}|), \tag{3}$$

where $U(1, |\mathcal{C}|)$ denotes a discrete uniform distribution across elements of $\mathcal{C}$. Through the stochasticity of z, we obtain a different combination of disentangled feature representations at each iteration.

Then, we deactivate A^z and average all other class-specific features $A^c, \forall c \in \mathcal{C} \setminus \{z\}$ to obtain A_{drop}:

$$A_{\text{drop}} = \frac{1}{|\mathcal{C}|} \sum_{c=1}^{|\mathcal{C}|} \mathbb{1}[c \neq z] \cdot A^c, \tag{4}$$

where $\mathbb{1}$ represents an indicator function. After passing A_{drop} through a convolution layer to compensate missing information, we obtain a new feature representation $A' = \text{conv}(A_{\text{drop}})$ that contains information for all classes except class z. This feature is passed through the classifier to generate new logits, $\hat{y}$. We illustrate the DropClass procedure in Figure 1.

In DropClass, the disentangling and debiasing processes act like positive feedback loops. The highlighted feature, A^c, serves as a disentangled feature representation, and is directly used to help debias feature representations of other classes. In turn, more debiased feature representations lead to better disentanglement as well. Ultimately, the two processes complement each other.

2.4 Objective Function

We train the model with multiple loss functions, each with a specific goal in mind. We first define the cross-entropy loss for semantic segmentation:

$$\mathcal{L}_{\text{CE}}(\hat{y}, y) = -\frac{1}{h \times w} \sum_{u=1}^{h} \sum_{v=1}^{w} \sum_{c=1}^{|\mathcal{C}|} y_{u,v,c} \cdot \log(p_{u,v,c}), \tag{5}$$

where $p = \text{softmax}(\hat{y})$ applied over the channel dimension. To account for the dropped class z, we design a modified cross-entropy objective function that masks the pixel-wise loss at all locations where class z is the ground truth:

$$\mathcal{L}_{\text{CE_drop}}(\hat{y}, y) = -\frac{1}{h \times w} \sum_{u=1}^{h} \sum_{v=1}^{w} \sum_{c=1}^{|C|} y_{u,v,c} \cdot \log(p_{u,v,c}) \cdot \mathbb{1}[y_{u,v,z} \neq 1]. \tag{6}$$

We use $\mathcal{L}_{\text{CE}}$ with the model's original output and $\mathcal{L}_{\text{CE_drop}}$ for the DropClass output:

$$\mathcal{L}_{\text{seg}} = (1 - \lambda) \cdot \mathcal{L}_{\text{CE}}(\hat{y}, y) + \lambda \cdot \mathcal{L}_{\text{CE_drop}}(\hat{y}_{\text{drop}}, y), \tag{7}$$

where λ is a scaling constant that balances the two loss terms.

The scaling constant λ plays an important role in ensuring the stability and effectiveness of training. The process outlined in Section 2.2 and 2.3 is based on the assumption that the gradient of $A_{i,j,k}$ with respect to $\hat{y}^c$ accurately quantifies the importance of $A_{i,j,k}$ on the prediction of class c. In the initial stages of training, however, $\hat{y}$ is likely to be largely inaccurate, and thus, the gradient may not accurately reflect the relationship between the feature and class prediction. This may lead to an unstable training process. To alleviate this issue, we train the model primarily with $\mathcal{L}_{\text{CE}}$ at the early stages of training and linearly decrease its weight as training progresses. Conversely, we gradually increase the weight for $\mathcal{L}_{\text{CE_drop}}$, thereby facilitating learning of debiased and disentangled representations towards the end of training. This is achieved by linearly increasing the value of λ from 0 to 1 across the duration of training. To further stabilize training, we linearly increase the probability of dropping out any class from 0 to 1 as well.

Finally, we add an additional objective term $\mathcal{L}_{\text{sup}}$ that helps suppress the output probability of the dropped class z:

$$\mathcal{L}_{\text{sup}} = \frac{1}{h \times w} \sum_{u=1}^{h} \sum_{v=1}^{w} \hat{y}^z_{u,v}. \tag{8}$$

Even after removing information about class z, some information could still persist in the dropped feature A_{drop} which would allow the model to predict a score for class z, albeit with low confidence. Hence, by suppressing the output logit for class z directly, the model is encouraged learn debiased representations, *i.e.,* purge any redundant information related to class z from feature representations of all other classes $c \in \mathcal{C} \setminus z$. Note that this loss is only calculated when a class is dropped. Altogether, the objective function that facilitates learning of disentangled and debiased representations is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{seg}} + \alpha \mathcal{L}_{\text{sup}}, \tag{9}$$

where α is a hyper-parameter that weighs the relative importance of $\mathcal{L}_{\text{sup}}$. A summary of our proposed training scheme can be found in Algorithm 1.

Algorithm 1 Training scheme for DropClass

Require: Dataset $D = \{(x_i, y_i)\}$, set of classes $\mathcal{C}$, feature extractor $g(\cdot)$ and classifier $h(\cdot)$

for $i = 1, 2, \ldots$ **do**
  Compute $A = g(x_i)$, $\hat{y} = h(A)$
  for $c = 1, 2, \ldots$ **do**
    Compute $S^c_{i,j,k} = \frac{1}{h \times w} \sum_{u=1}^{h} \sum_{v=1}^{w} \frac{\partial \hat{y}_{u,v,c}}{\partial A_{i,j,k}}$. ▷ Eq. (1)
    Compute $A^c = \text{ReLU}(A \odot S^c)$ ▷ Eq. (2)
  end for
  Sample a class z to drop and deactivate A^z. ▷ Eq. (3)
  Gather all A^c for $\forall c \in \mathcal{C} \setminus \{z\}$ and feed them into the classifier. $h(\cdot)$. ▷ Eq. (4)
  Compute loss function and update parameters ▷ (9)
end for

2.5 Further Details

Test time behavior The gradient computation in Eq. (1) are calculated using the class logits $\hat{y}^c$, rather than the ground truth label y^c. Thus, computation of S^c and A^c are all possible at test time as well, where ground truth labels are unavailable. Furthermore, we do not sample a drop class z at test time; thus, A_{drop} becomes a simple average of class-specific features, and is fed into the convolution and classification layers.

Computational overhead Calculating gradients for each class also impose a heavy computational burden, which may lead to slow training. However, we take advantage of a simple trick which allows us to calculate the gradient in Eq. (1) without a large computational overhead. Since convolution layers are linear operations, $\partial \hat{y}^c_{u,v} / \partial A_{i,j,k}$ conveniently reduces to the weight of the convolution kernel. Thus, we employ a 1×1 convolution as the final classification layer, h, which is a common design choice in most semantic segmentation models. Using this trick reduces both the computational and memory overhead of the gradient calculation, and allows it to be processed in parallel as well.

3 Experiments

To evaluate the effectiveness of DropClass, we experiment on two well-known semantic segmentation datasets: Cityscapes [1] and Pascal VOC Segmentation[11]. We also test with two different segmentation networks — HRNet [4] and DeepLabV3 [5] — to highlight that our method is not dependent on any specific network architecture. We compare DropClass with a baseline, where the model is trained with an ordinary cross-entropy loss (Eq. (5)) only. In order to ensure fair comparison between the two experiments, we use the same set of hyperparameters for both experiments, unless noted otherwise.

Datasets We choose to evaluate on Cityscapes and Pascal VOC for a few reasons. First, both datasets are relatively small, with 2975 train images on Cityscapes and 1464 train images on Pascal VOC. Second, both datasets have large class imbalances, with the pixel frequency ranging from 0.1% to 36.9% on Cityscapes, and the instance frequency ranging from 1.7% to 14.5% on Pascal VOC. These two commonalities make it difficult for an ordinary model to learn robust, debiased representations on either datasets. In addition, there are a few key differences between these datasets that help illustrate our method's efficacy in diverse settings. First, Cityscapes is a urban road scene dataset where images are captured from the dashcam of a vehicle, while the images in Pascal VOC are more random images that are captured in every-day life. Second, the average number of classes that appear in each image vary greatly between the two datasets: while Cityscapes has an average of around 12 classes per image, Pascal VOC averages at 1.5 classes per image (without the "background" class). Although we omit the "background" class for Pascal VOC, we present experimental results including the background class in the supplementary materials. All in all, the two datasets provide a diverse and challenging environment to demonstrate the effectiveness of our proposed method.

Table 1: Categorical IoU scores for the Cityscapes dataset. Classes are sorted in order of increasing pixel frequency. mIoU† indicates mIoU for the 9 classes with lowest pixel frequency. DeepLabV3 uses MobileNetV3 architecture as the backbone.

		motorcycle†	rider†	t.light†	train†	bus†	truck†	bicycle†	t.sign†	wall†	fence	terrain	person	pole	sky	sidewalk	car	vegetation	building	road	mIoU	mIoU†
Model	Pixel %	0.1	0.1	0.2	0.2	0.2	0.3	0.4	0.6	0.7	0.9	1.2	1.2	1.2	4.0	6.1	7.0	15.9	22.8	36.9		
HRNet W18 v1	Baseline	47.9	53.5	63.3	47.3	71.8	48.6	72.8	72.8	44.7	52.0	61.5	77.0	56.7	93.5	81.6	92.9	91.5	90.5	97.5	69.3	58.1
	Ours	50.2	55.7	65.3	48.0	73.4	48.6	74.1	75.3	47.1	53.7	61.9	78.7	61.6	94.0	82.3	93.3	92.1	91.1	97.6	70.7	59.7
	Δ IoU	2.3	2.2	2.0	0.7	1.6	0.0	1.3	2.5	2.4	1.7	0.4	1.7	4.9	0.5	0.7	0.4	0.6	0.6	0.1	1.4	1.6
	Δ (%)	4.8	4.1	3.2	1.5	2.2	0.0	1.8	3.4	5.4	3.3	0.7	2.2	8.6	0.5	0.9	0.4	0.7	0.7	0.1	2.0	2.9
HRNet W18 v2	Baseline	58.3	61.5	69.3	62.7	80.6	68.2	76.3	76.3	50.3	59.5	65.0	81.0	62.6	94.1	84.6	94.4	92.2	92.0	98.0	75.1	67.1
	Ours	59.7	60.9	71.7	69.7	83.2	68.5	76.8	79.1	55.5	60.2	63.7	82.2	66.9	95.0	84.0	94.9	92.7	92.7	98.0	76.6	69.5
	Δ	1.4	-0.6	2.4	7.0	2.6	0.3	0.5	2.8	5.2	0.7	-1.3	1.2	4.3	0.9	-0.6	0.5	0.5	0.7	0.0	1.5	2.4
	Δ (%)	2.4	-1.0	3.5	11.2	3.2	0.4	0.7	3.7	10.3	1.2	-2.0	1.5	6.9	1.0	-0.7	0.5	0.5	0.8	0.0	2.0	3.8
DeepLab V3	Baseline	45.9	51.6	53.7	52.3	66.1	52.3	68.8	66.0	46.0	52.9	57.5	73.1	48.9	92.6	79.5	92.1	90.4	89.7	97.4	67.2	55.9
	Ours	49.7	53.1	53.7	55.0	67.4	51.9	68.4	66.8	47.1	52.4	58.0	73.5	48.5	92.8	79.1	92.3	90.3	89.9	97.3	67.7	57.0
	Δ	3.8	1.5	0.0	2.7	1.3	-0.4	-0.4	0.8	1.1	-0.5	0.5	0.4	-0.4	0.2	-0.4	0.2	-0.1	0.2	-0.1	0.5	1.1
	Δ (%)	8.3	2.9	0.0	5.2	2.0	-0.8	-0.6	1.2	2.4	-0.9	0.9	0.5	-0.8	0.2	-0.5	0.2	-0.1	0.2	-0.1	0.7	2.3

Table 2: Categorical IoU scores for the Pascal VOC segmentation dataset. Classes are sorted in order of increasing instance frequency. mIoU† indicates mIoU for the 10 classes with lowest instance frequency. DeepLabV3 uses MobileNetV3 architecture as the backbone.

		cow†	sheep†	bus†	pottedplant†	bicycle†	horse†	boat†	motorbike†	sofa†	train†	diningtable	tvmonitor	bottle	aeroplane	bird	car	cat	chair	dog	person	mIoU	mIoU†
	Instance %	1.7	1.9	2.4	2.7	2.8	2.8	2.8	3.0	3.0	3.1	3.2	3.2	3.5	3.6	4.2	5.6	5.8	5.9	6.6	14.5		
HRNet W18 v1	Baseline	83.9	60.6	64.2	68.4	60.9	80.7	79.9	77.7	31.6	50.5	59.2	65.0	55.1	74.1	82.1	71.4	62.4	52.4	80.5	74.7	66.8	65.8
	Ours	84.4	62.8	70.8	65.4	64.2	79.0	83.2	79.2	31.7	52.4	57.0	64.4	53.3	75.9	81.1	72.8	66.9	44.8	79.2	78.1	67.3	67.3
	Δ	0.5	2.2	6.6	-3.0	3.3	-1.7	3.3	1.5	0.1	1.9	-2.2	-0.6	-1.8	1.8	-1.0	1.4	4.5	-7.6	-1.3	3.4	0.5	1.5
	Δ (%)	0.6	3.6	10.3	-4.4	5.4	-2.1	4.1	1.9	0.3	3.8	-3.7	-0.9	-3.3	2.4	-1.2	2.0	7.2	-14.5	-1.6	4.6	0.8	2.2
HRNet W18 v2	Baseline	61.2	74.4	91.7	79.6	74.8	65.7	80.9	85.6	58.5	92.1	72.0	87.2	78.7	92.2	83.6	87.5	89.6	45.9	77.5	87.4	78.3	76.5
	Ours	70.5	77.2	91.4	84.1	74.0	70.5	87.5	87.7	58.4	90.4	70.8	86.4	78.1	92.5	85.5	89.7	90.2	46.0	78.6	87.6	79.9	79.2
	Δ	9.3	2.8	-0.3	4.5	-0.8	4.8	6.6	2.1	-0.1	-1.7	-1.2	-0.8	-0.6	0.3	1.9	2.2	0.6	0.1	1.1	0.2	1.6	2.7
	Δ (%)	15.2	3.8	-0.3	5.7	-1.1	7.3	8.2	2.5	-0.2	-1.8	-1.7	-0.9	-0.8	0.3	2.3	2.5	0.7	0.2	1.4	0.2	2.0	3.6
DeepLab V3	Baseline	77.4	87.4	96.9	81.1	80.5	76.3	87.6	84.7	48.7	97.2	68.3	85.3	83.1	94.5	89.9	88.9	90.3	42.2	82.2	85.3	81.4	81.8
	Ours	76.9	89.1	96.4	77.6	81.9	78.2	85.6	84.6	53.3	97.2	65.5	86.4	81.9	92.9	89.6	88.5	91.8	45.5	83.5	86.0	81.6	82.1
	Δ	-0.5	1.7	-0.5	-3.5	1.4	1.9	-2.0	-0.1	4.6	0.0	-2.8	1.1	-1.2	-1.6	-0.3	-0.4	1.5	3.3	1.3	0.7	0.2	0.3
	Δ (%)	-0.6	2.0	-0.6	-4.3	1.8	2.4	-2.3	-0.1	9.4	0.0	-4.2	1.3	-1.4	-1.7	-0.3	-0.5	1.7	7.9	1.6	0.8	0.3	0.3

3.1 Implementation Details

Network Architectures We employ two recent segmentation network architectures, both known for their strong performance in semantic segmentation: HRNet [4] and DeepLabV3 [5] with a MobileNetV3 [12] backbone. For HRNet, we use the code provided by the authors[1], and use two different versions of HRNet: HRNet W18 v1 and HRNet W18 v2. For DeepLabV3, we use the official PyTorch [13] code[2] and change the default backbone to MobileNetV3. All backbones are trained on the ImageNet [14] dataset.

Hyperparameters The value of the loss weighing term α in Eq. (9) is set to 10 for all experiments, which is based on the scale of the two loss terms. As mentioned in Section 2.4, the value of λ is initialized as 0 and scaled linearly up to 1 until the end of training. Since DropClass depends on GradCAM to generate disentangled features, it takes more iterations for the model to fully converge under the DropClass training scheme. Furthermore, we sample a new class z for the entire batch at each training iteration.

[1] `https://github.com/HRNet/HRNet-Semantic-Segmentation`

[2] `https://github.com/pytorch/vision/blob/master/torchvision/models/segmentation/deeplabv3.py`

The rest of our hyperparameters are organized as a table in the supplementary materials, where we detail the number of iterations, batch size, learning rate, learning rate decay, image size, and type of GPU used for each of our experiments.

3.2 Quantitative Results

Presentation format We present our categorical Intersection-over-Union (IoU) and mean-IoU (mIoU) scores for Cityscapes and Pascal VOC in Tables 1 and 2, respectively. For all models across both tables, we compare our method with the baseline and report the absolute change in IoU (Δ), as well as the relative change in IoU ($\Delta(\%)$).

For better interpretability of our results, the classes are sorted in order of increasing frequency, and we also calculate a separate mIoU score for the least frequent 50% of classes (denoted by mIoU†). However, it is important to note that we employ a different metric to define frequency on each dataset. On Cityscapes, we count the number of pixels that appear in the training set for each class, and normalize the counts to obtain the pixel frequency. We use this metric because each image in Cityscapes contains a large number of classes, and the average size of each class varies greatly, *e.g.,* one instance of "road" has far more pixels than one instance of "pole". On Pascal VOC, however, only a few classes appear per image (1.5), and the class sizes do not vary as much as they do in Cityscapes. Thus, we count the number of images that each class appears in, and calculate the normalized instance frequency.

Cityscapes On the Cityscapes dataset, our method consistently outperforms the baseline across all models, with 2.0%, 2.0%, and 0.7% gains on the mIoU for HRNet W18 v1, HRNet W18 v2, and DeepLabV3 models, respectively. More importantly, we observe even larger improvements on the mIoU for the 9 most under-represented classes, which account for merely 2.8% of all pixels: 2.9% on HRNet W18 v1, 3.8% on HRNet W18 v2, and 2.3% on DeepLabV3. These empirical results supports our claim that DropClass helps the model learn debiased feature representations that benefit the model's overall performance, especially for under-represented classes.

There are a few more interesting details we observe from Table 1. First, the IoU improves on most classes, albeit with smaller margins on well-represented classes. Also, we notice especially large improvements across all models on the "train" and "wall" categories, which also happen to be two of the least frequently appearing categories: "train" is the least frequent class, appearing in only 0.4% of all images, and "wall" is the fifth least frequent class, appearing in only 2.8% of all images.

Pascal VOC Segmentation We observe similar performance gains on the Pascal VOC Segmentation dataset as well. Our model outperforms the overall mIoU by 0.8%, 2.0%, and 0.3% on HRNet W18 v1, HRNet W18 v2, and DeepLabV3, respectively. Our model exhibits even stronger performance on under-represented classes as well, with a 2.2%, 3.6% and 0.3% increase in mIoU.

Compared to the results on Cityscapes, however, the performance gains are much less consistent across all classes. In other words, we observe categorical performance drops more frequently. We hypothesize that this behavior is caused by the low number of average classes per image in Pascal VOC (1.5 classes per image without the "background" class, compared to 12 classes per image on Cityscapes). Since fewer classes appear in each image, there is inherently less bias caused by inter-class relationship that our model can reduce. Thus, the IoU gains are less consistent on this dataset than they are on the Cityscapes dataset. Nevertheless, our model still outperforms the baseline by a significant amount, which demonstrates that our method is effective on a wide range of datasets, regardless of their characteristics.

Table 3: Ablation experiments on Cityscapes with HRNet W18 v1.

Method	mIoU	mIoU†
Baseline	69.3	58.1
Ours	70.7	59.7
Δ	1.4	1.6
Δ (%)	2.0	2.9
w/o Eq. (8)	70.2	58.6
Δ	0.9	0.5
Δ (%)	1.4	0.9
Ignore Label	69.2	58.0
Δ	-0.1	-0.1
Δ (%)	-0.1	-0.1

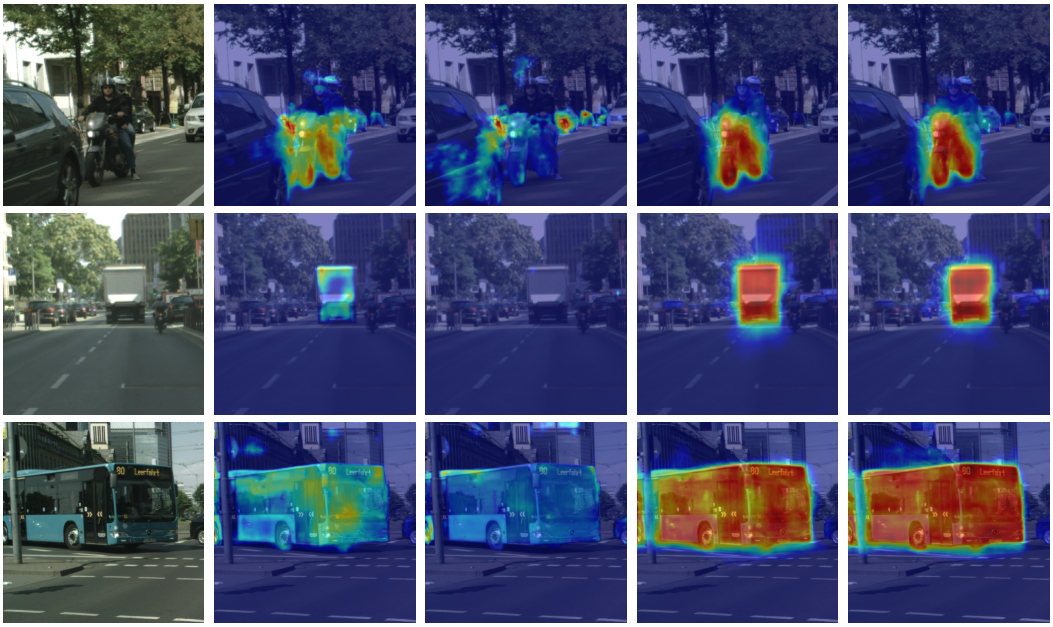

Figure 2: Grad-CAM visualizations of incorrect and correct classes. Incorrect class used, from top to bottom: "bicycle", "train", and "truck". For Grad-CAMs of the incorrect class, our model generates low activations (column 3) compared to Grad-CAMs of the baseline model (column 2). On the other hand, for Grad-CAMs of the correct class, both models generate strong activations (columns 4 and 5). This suggest that the DropClass model reduces biases from other classes while maintaining strong disciminative features for the correct class.

3.3 Analysis

Ablations To better understand our proposed method and its effectiveness, we conduct ablation experiments on the Cityscapes dataset with the HRNet W18 v1 model. First, we omit $\mathcal{L}_{\text{sup}}$ defined in Eq (8), which has the purpose of suppressing outputs for the dropped class. We observe that when we remove this loss term, the effectiveness of our method decreases slightly in terms of the total mIoU and the under-represented class mIoU (mIoU†), by 0.5%p and 1.1%p, respectively. However, it still outperforms the baseline by 0.9%p and 0.5%p on the two metrics. Next, we also test the effects of dropping the labels of a randomly selected class, without DropClass or any of its accompanying loss functions. In other words, a model is trained with the ordinary cross-entropy loss, Eq. (5), while random class labels are dropped at each iteration. Unsurprisingly, we observe that this model performs worse than the baseline model, by 0.1%p in both metrics. Thus, our ablation experiments highlight the importance of the suppression loss and validate the effectiveness of DropClass.

Grad-CAM Visualizations We further analyze our method by examining Grad-CAM visualizations of trained models. We first select three pairs of classes that are likely cause confusion for a segmentation model: Train-Truck, Truck-Bus, and Bicycle-Motorcycle. Then we generate Grad-CAM visualizations of the incorrect class with the baseline and DropClass models (HRNet W18 v1 on Cityscapes), and compare the results in Figure 2. Each row corresponds to a different class pair. The second and third columns correspond to the visualizations of the baseline and DropClass models, respectively, when visualizing for the incorrect class. Compared to the visualizations of the baseline model, the visualizations of our model consistently display lower activations for the incorrect class. In other words, the DropClass model is less vulnerable to confusion, which strengthens the argument our proposed DropClass training scheme leads to more debiased and disentangled feature representations. Furthermore, our model exhibits strong gradients for the correct class as well, which indicates that the model does not lose any discriminability in the process of debiasing.

4 Related Works

Dataset Bias Dataset bias is a critical issue in modern machine learning because trained models often achieve outstanding accuracy simply by capturing the correlated features instead of identifying the proper representations of target classes. To tackle the limitation, various debiasing techniques have been proposed under supervised [15, 16, 17, 18] and unsupervised [19, 20] settings, but most of existing approaches are for simple classification tasks. Technically, they are limited to sample reweighting [21, 22, 23] and loss adjustment [15, 19, 16, 17, 18]. The proposed algorithm is unique in the sense that we aim to learn a classifier for dense prediction.

Semantic Segmentation Semantic segmentation works have tried to improve performance with diverse operations that aim to widen the receptive field, while maintaining details in the features from different resolutions. To this end, DeconvNet [6] learns deconvolution layers to replace fixed upsampling operations and DeepLabV3 [5] employs atrous convolution to capture objects at multiple scales. Furthermore, PSPNet [24] proposes a pyramid pooling module to capture different sub-region representations and HRNet [4] aggregates four different resolution of feature maps to preserve details, especially for the deeper layers with lower resolutions.

Visual explanations Even with extensive results in broad domains, non-convexity and high-dimensionality of deep neural networks restrict human-level understandings of its internal process. To remedy this issue, [25, 10, 26, 27] attempt to make visual explanations of deep models. CAM [25] first proposes to find discriminative regions relevant to target classes, but has architectural restriction due to the use of the global pooling layer. Grad-CAM [10] visualizes the importance of each class by leveraging the gradient information. [26] extends Grad-CAM and applies it to semantic segmentation. CAM and Grad-CAM have often been employed in weakly supervised object detection [28, 29] and segmentation [30, 31, 32], where they highlight relevant objects or regions without full supervision for target tasks. We compute Grad-CAM for each class, but employ the estimated class-specific feature maps for learning debiased and disentangled representations.

5 Limitations & Negative Societal Impact

Limitations One limitation of the proposed DropClass training scheme is that it requires more iterations for the model to fully converge. This is because DropClass depends on the model's ability to generate disentangled feature representations. Hence, the model must be trained with ordinary supervision first, which is also the reason we ramp up λ from 0 to 1 during training.

Negative Societal Impact The negative societal impact of our work do not necessarily pertain to our work itself, but moreso to deep learning and semantic segmentation. Semantic segmentation, and deep learning in general, requires massive amounts labeled data to train the model. As more and more large-scale datasets are created and made public, one natural point of concern is privacy. For example, even the datasets that we use — Cityscapes and Pascal VOC — both have images containing people; people who most likely are completely unaware that pictures of them are being used by machine learning researchers all over the world. Although we do not know the extent of negative impact this would have on society, concerns over privacy is definitely something that needs to be considered when curating new datasets.

6 Conclusion

We presented a model agnostic training scheme that helps the model learn more debiased and disentangled feature representations for the semantic segmentation task. Our method is based on the simple and intuitive idea of randomly dropping disentangled class-specific features at each iteration of training. The new combination of features removes certain inter-class dependencies, which leads to learning debiased feature representations. Our experimental results on two distinct network architectures and datasets validate our method's effectiveness, and our analysis reflect that our method achieves the intended goals.

References

[1] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In CVPR. (2016)

[2] Ronneberger, O., P.Fischer, Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention (MICCAI). (2015)

[3] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In CVPR. (2017)

[4] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al.: Deep high-resolution representation learning for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2020)

[5] Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation (2017)

[6] Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In ICCV. (2015)

[7] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. CoRR (2016)

[8] Long, J., Shelhamer, E., Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. In CVPR. (2015)

[9] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1) (2014) 1929–1958

[10] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In ICCV. (2017)

[11] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision (IJCV) (2010)

[12] Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In ICCV. (2019)

[13] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In NeurIPS. (2019)

[14] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In CVPR. (2009)

[15] Sagawa, S., Koh, P.W., Hashimoto, T.B., Liang, P.: Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In ICLR. (2020)

[16] Wang, T., Zhao, J., Yatskar, M., Chang, K.W., Ordonez, V.: Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In ICCV. (2019)

[17] Zhang, B.H., Lemoine, B., Mitchell, M.: Mitigating unwanted biases with adversarial learning. In AAAI. (2018)

[18] Gong, S., Liu, X., Jain, A.K.: Jointly de-biasing face recognition and demographic attribute estimation. In ECCV. (2020)

[19] Sohoni, N., Dunnmon, J., Angus, G., Gu, A., Ré, C.: No subclass left behind: Fine-grained robustness in coarse-grained classification problems. In NeurIPS. (2020)

[20] Nam, J., Cha, H., Ahn, S., Lee, J., Shin, J.: Learning from failure: Training debiased classifier from biased classifier. In NeurIPS. (2020)

[21] Li, Y., Vasconcelos, N.: Repair: Removing representation bias by dataset resampling. In CVPR. (2019)

[22] Sagawa, S., Raghunathan, A., Koh, P.W., Liang, P.: An investigation of why overparameterization exacerbates spurious correlations. In ICML. (2020)

[23] Kamiran, F., Calders, T.: Data preprocessing techniques for classification without discrimination. Knowledge and Information Systems **33**(1) (2012) 1–33

[24] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In CVPR. (2017)

[25] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In CVPR. (2016)

[26] Vinogradova, K., Dibrov, A., Myers, G.: Towards interpretable semantic segmentation via gradient-weighted class activation mapping (student abstract). In AAAI. (2020)

[27] Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In WACV. (2018)

[28] Zhang, J., Yu, X., Li, A., Song, P., Liu, B., Dai, Y.: Weakly-supervised salient object detection via scribble annotations. In CVPR. (2020)

[29] Li, G., Xie, Y., Lin, L.: Weakly supervised salient object detection using image labels. In AAAI. (2018)

[30] Kwak, S., Hong, S., Han, B.: Weakly supervised semantic segmentation using superpixel pooling network. In AAAI. (2017)

[31] Huang, Z., Wang, X., Wang, J., Liu, W., Wang, J.: Weakly-supervised semantic segmentation network with deep seeded region growing. In CVPR. (2018)

[32] Lee, J., Kim, E., Lee, S., Lee, J., Yoon, S.: Ficklenet: Weakly and semi-supervised semantic image segmentation using stochastic inference. In CVPR. (2019) 5267–5276

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Sections 2 and 3.
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 3.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Segmentation experiments are very time-consuming. However, the random seed was fixed for all experiments
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3.1 and Supplementary materials.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Datasets used are all public and were cited.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]