Regularizing Deep Networks by Modeling and Predicting Label Structure

Mohammadreza Mostajabi Michael Maire Gregory Shakhnarovich
Toyota Technological Institute at Chicago
{mostajabi,mmaire,greg}@ttic.edu

Abstract

We construct custom regularization functions for use in supervised training of deep neural networks. Our technique is applicable when the ground-truth labels themselves exhibit internal structure; we derive a regularizer by learning an autoencoder over the set of annotations. Training thereby becomes a two-phase procedure. The first phase models labels with an autoencoder. The second phase trains the actual network of interest by attaching an auxiliary branch that must predict output via a hidden layer of the autoencoder. After training, we discard this auxiliary branch.

We experiment in the context of semantic segmentation, demonstrating this regularization strategy leads to consistent accuracy boosts over baselines, both when training from scratch, or in combination with ImageNet pretraining. Gains are also consistent over different choices of convolutional network architecture. As our regularizer is discarded after training, our method has zero cost at test time; the performance improvements are essentially free. We are simply able to learn better network weights by building an abstract model of the label space, and then training the network to understand this abstraction alongside the original task.

1. Introduction

The recent successes of supervised deep learning rely on the availability of large-scale datasets with associated annotations for training. In computer vision, annotation is a sufficiently precious resource that it is commonplace to pretrain systems on millions of labeled ImageNet [8] examples. These systems absorb a useful generic visual representation ability during pretraining, before being fine-tuned to perform more specific tasks using fewer labeled examples.

Current state-of-the-art semantic segmentation methods [25, 7, 46] follow such a strategy. Its necessity is driven by the high relative cost of annotating ground-truth for spatially detailed segmentations [12, 24], and the accuracy gains achievable by combining different data sources and label modalities during training. A collection of many images, coarsely annotated with a single label per image (*e.g.* ImageNet [8]), is still quite informative in comparison to a smaller collection with detailed per-pixel label maps for each image (*e.g.* PASCAL [12] or COCO [24]).

We show that detailed ground-truth annotation of this latter form contains additional information that existing schemes for training deep convolutional neural networks (CNNs) fail to exploit. By designing a new training procedure, we are able to capture some of this information, and as a result increase accuracy at test time.

Our method is orthogonal to recent efforts, discussed in Section 2, on learning from images in an unsupervised or self-supervised manner [34, 31, 43, 22, 23, 44, 11]. It is not dependent upon the ability to utilize an external pool of data. Rather, our focus on more efficiently utilizing provided labels makes our contribution complementary to these other learning techniques. Experiments show gains both when training from scratch, and in combination with pretraining on an external dataset.

Our innovation takes the form of a regularization function that is itself learned from the training set labels. This yields two distinct training phases. The first phase models the structure of the labels themselves by learning an autoencoder. The second phase follows the standard network training regime, but includes an auxiliary task of predicting the output via the decoder learned in the first phase. We view this auxiliary branch as a regularizer; it is only present during training. Figure 1 illustrates this scheme.

Section 3 further details our approach and the intuition behind it. Our regularizer can be viewed as a requirement that the system understand context, or equivalently, as a method for synthesizing context-derived labels at coarser spatial resolution. The auxiliary branch must predict this more abstract, context-sensitive representation in order to successfully interface with the decoder.

Experiments, covered in Section 4, focus on the PASCAL semantic segmentation task. We take baseline CNN architectures, the established VGG [36] network and the state-of-the-art DenseNet [16], and report performance gains of enhancing them with our custom regularizer during training. Section 4 also provides ablation studies, explores an alternative regularizer implementation, and visu-

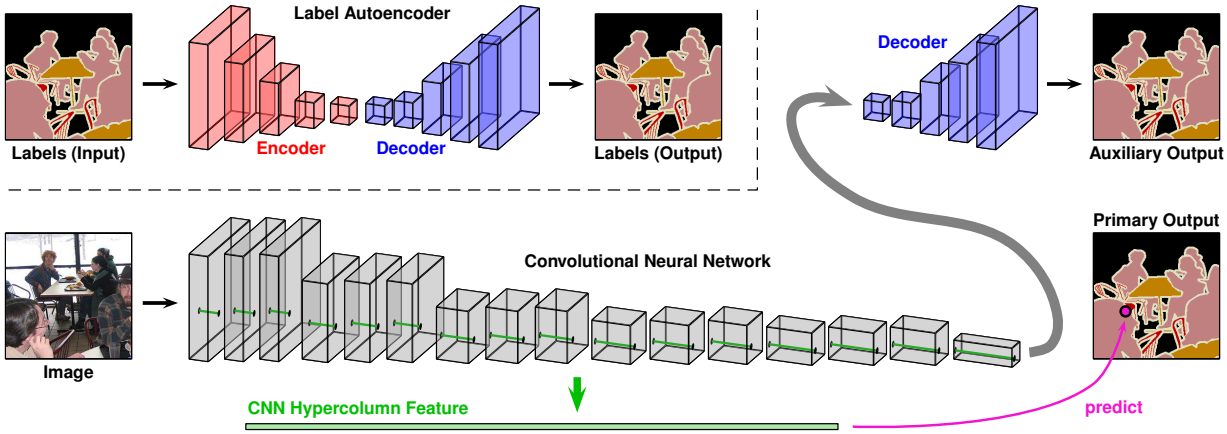

Figure 1. **Exploiting label structure when training semantic segmentation.** ***Top:*** An initial phase looks only at the ground-truth annotation of training examples, ignoring the actual images. We learn an autoencoder that approximates an identity function over segmentation label maps. It is constrained to compress and reconstitute labels by passing them through a bottleneck connecting an encoder (red) and decoder (blue). ***Bottom:*** The second phase trains a standard convolutional neural network (CNN) for semantic segmentation using hypercolumn [14, 29] features for per-pixel output. However, we attach an auxiliary branch (and loss) that also predicts segmentation by passing through the decoder learned in the first phase. After training, we discard this decoder branch, making the architecture appear standard.

alizes representations learned by the label autoencoder.

Results demonstrate performance gains under all settings in which we applied our regularization scheme: VGG or DenseNet, with or without data augmentation, and with or without ImageNet pretraining. Performance of a very deep DenseNet, with data augmentation and ImageNet pretraining, is still further improved with use of our regularizer during training. Together, these results indicate that we have discovered a new and generally applicable method for regularizing supervised training of deep networks. Moreover, our method has no cost at test time; it produces networks architecturally identical to baseline designs.

Section 5 discusses implications of our demonstration that it is possible to squeeze more benefit from detailed label maps when training deep networks. Our results open up a new area of inquiry on how best to build datasets and design training procedures to efficiently utilize annotation.

2. Related Work

The abundance of data, but more limited availability of ground-truth supervision, has sparked a flurry of recent interest in developing self-supervised methods for training deep neural networks. Here, the idea is to utilize a large reserve of unlabeled data in order to prime a deep network to encode generally useful visual representations. Subsequently, that network can be fine-tuned on a novel target task, using actual ground-truth supervision on a smaller dataset. Pretraining on ImageNet [8] currently yields such portable representations [10], but lacks the ability to scale without requiring additional human annotation effort.

Recent research explores a diverse array of data sources and tasks for self-supervised learning. In the domain of images, proposed tasks include inpainting using context [34], solving jigsaw puzzles [31], colorization [43, 22, 23], cross-channel prediction [44], and learning a bidirectional variant [11] of generative adversarial networks (GANs) [13]. In the video domain, recent works harness temporal coherence [28, 18], co-occurrence [17], and ordering [27], as well as tracking [40], sequence modeling [37], and motion grouping [33]. Owens *et al*. [32] explore cross-modality self-supervision, connecting vision and sound. Agrawal *et al*. [3] and Nair *et al*. [30] examine settings in which a robot learns to predict the visual effects of its own actions.

Training a network to perform ImageNet classification or a self-supervised task, in addition to the task of interest, can be viewed as a kind of implicit regularization constraint. Zhang *et al*. [45] explore explicit auxiliary reconstruction tasks to regularize training. However, they focus on encoding and decoding image feature representations. Our approach differs entirely in the source of regularization.

Specifically, by autoencoding the structure of the target task labels, we utilize a different reserve of information than all of the above methods. We design a new task, but whereas self-supervision formulates the new task on external data, we derive the new task from the annotation. This separation of focus allows for possible synergistic combination of our method with pretraining of either the self-supervised or supervised (ImageNet) variety. Section 4 tests the latter.

Another important distinction from recent self-supervised work is that, as detailed in Section 3, we use a generic mechanism, based on an autoencoder, for deriving our auxiliary task. In contrast, the vast majority of effort in self-supervision has relied on using domain-specific knowledge to formulate appropriate tasks. Inpainting [34], jigsaw puzzles [31], and colorization [43, 22, 23] exemplify

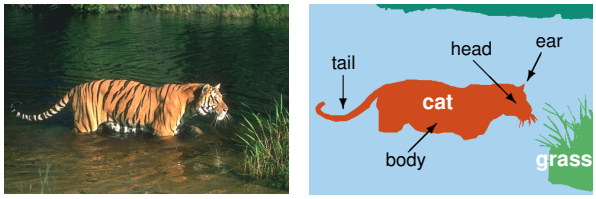

Figure 2. **Informative structure in annotation.** The shape of labeled semantic regions hints at unlabeled parts (black arrows). Object co-occurrence provides a prior on scene composition.

this mindset; BiGANs [11] are perhaps an exception, but to date their results compare less favorably [23].

The work of Xie *et al*. [41] shares similarities to our approach along the aspect of modeling label space. However, they focus on learning a shallow corrective model that essentially denoises a predicted label map using center-surround filtering. In contrast, we build a deep model of label space. Also, unlike [41], our approach has no test-time cost, as we impose it only as a regularizer during training, rather than as an ever-present denoising layer.

Inspiration for our method traces back to the era of vision prior to the pervasive use of deep learning. It was once common to consider context as important [39], reason about object parts, co-occurrence, and interactions [9], and design graphical models to capture such relationships [38]. We refer to only a few sample papers as fully accounting for a decade of computer vision research is not possible here. In the following section, we open a pathway to pull such thinking about compositional scene priors into the modern era: simply learn, and employ, a deep model of label space.

3. Method

Figure 2 is a useful aid in explaining the intuition behind the regularization scheme outlined in Figure 1. Suppose we want to train a CNN to recognize and segment cats, but our limited training set consists only of tigers. It is conceivable that the CNN will learn an equivalence between black and orange striped texture and the cat category, as such association suffices to classify every pixel on a tiger. It thus overfits to the tiger subclass and fails when tested on images of house cats. This behavior could arise even if trained with detailed supervision of the form shown in Figure 2.

Yet, the semantic segmentation ground-truth suggests to any human that texture should not be the primary criteria. There are no stripes in the annotation. Over the entire training set, regions labeled as cat share a distinctive shape that deforms in a manner suggestive of unlabeled parts (*e.g.* head, body, tail, ear). The presence or absence of other objects in the scene may also provide contextual cues as to the chance of finding a cat. How can we force the CNN to notice this wealth of information during training?

We could consider treating the ground-truth label map as an image, and clustering local patches. The patch containing the skinny tail would fall in a different cluster than that containing the pointy ear. Adding the cluster identities as another semantic label, and requiring the CNN to predict them, would force the CNN to differentiate between the tail and ear by developing a representation of shape. This clustering approach is reminiscent of Poselets [6, 5].

Layers	DenseNet-67	DenseNet-121
Convolution	$\begin{bmatrix} 3 \times 3 \text{ conv, stride } 2 \\ 3 \times 3 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 1$	7×7 conv, stride 2
Pooling	3×3 max pool, stride 2	
Dense Block (1)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	1×1 conv 2×2 average pool, stride 2	
Dense Block (2)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 8$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	1×1 conv 2×2 average pool, stride 2	
Dense Block (3)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 8$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Transition Layer (3)	1×1 conv 2×2 average pool, stride 2	
Dense Block (4)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 8$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$

Figure 3. **DenseNet architectural specifications.**

Following this strategy, we would need to hand-craft another scheme for capturing object co-occurrence relations, perhaps by clustering descriptors spanning a larger spatial extent. We would prefer a general means of capturing features of the ground-truth annotations, and one not limited to a few hand-selected characteristics. Fortunately, deep networks are a suitable general tool for building the kind of abstract feature hierarchy we desire.

3.1. Modeling Labels

Specifically, as shown in Figure 1, we train an autoencoder on the ground-truth label maps. This autoencoder consumes a semantic segmentation label map as input and attempts to replicate it as output. By virtue of being required to pass through a small bottleneck representation, the job of the autoencoder is nontrivial. It must compress the label map into the bottleneck representation. This compression constraint will (ideally) force the autoencoder to discover and implicitly encode parts and contextual relationships.

Ground-truth semantic segmentation label maps are simpler than real images, so this autoencoder need not have as high of a capacity as a network operating on natural images. We use a relatively simply autoencoder architecture, consisting of a mirrored encoder and decoder, with no skip connections. The encoder is a sequence of five 3×3 convolutional layers, with 2×2 max-pooling between them. The decoder uses 2x upsampling followed by 3×3 convolution. As a default, we set each layer to have 32-channels. We also experiment with some higher-capacity variants:

- conv1: 32-channels; conv2-5: 128 channels each
- conv1: 32; conv2-4: 128; conv5: 256 channels

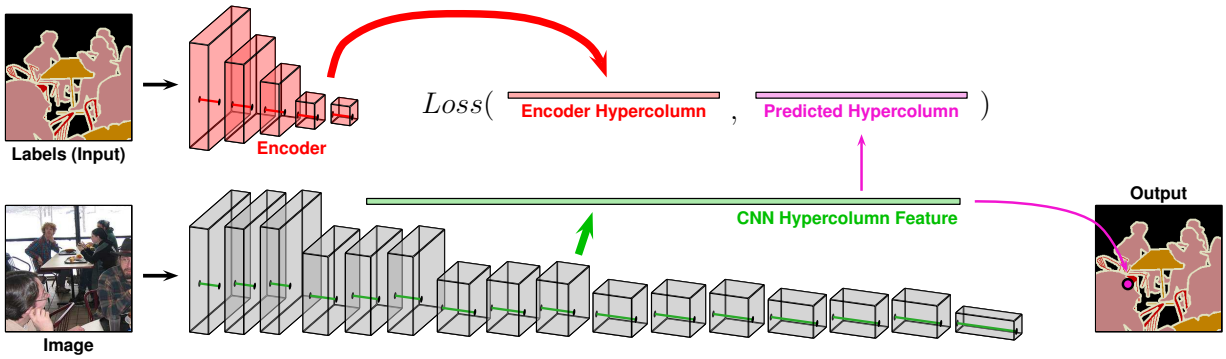

Figure 4. **Alternative regularization scheme.** Instead of predicting a representation to pass through the decoder, as in Figure 1, we can train with an auxiliary regression problem. We place a loss on directly predicting activations produced by the hidden layers of the encoder.

These channel progressions are for the encoder; the decoder uses the same in reverse order. We refer to these three autoencoder variants by the number of channels in their respective bottleneck layers (32, 128, or 256).

3.2. Baseline CNN Architectures

Convolutional neural networks for image classification gradually reduce spatial resolution with depth through a series of pooling layers [21, 36, 15, 16]. As the semantic segmentation task requires output at fine spatial resolution, some method of preserving or recovering spatial resolution must be introduced into the architecture. One option is to gradually re-expand spatial resolution via upsampling [35, 4]. Other approaches utilize some form of skip-connection to forward spatially resolved features from lower layers of the network to the final layer [26, 14, 29]. Dilated [42] or atrous convolutions [7] can also be mixed in. Alternatively, the basic CNN architecture can be reformulated in a multigrid setting [19].

Our goal is to examine the effects of a regularization scheme in isolation from major architectural design changes. Hence, we choose hypercolumn [14, 29] CNN architectures as a primary basis for experimentation, as they are are minimally separated from the established classification networks in design space. They also offer the added advantage of having readily available ImageNet pretrained models, easing experimentation in this setting.

We consider hypercolumn variants of VGG-16 [36] and DenseNet [16]. These variants simply upsample and concatenate features from intermediate network layers for use in predicting semantic segmentation. As shown in Figure 1, this can equivalently be viewed as associating with each spatial location a feature formed by concatenating a local slice of every CNN layer. The label of the corresponding pixel in the output is predicted from that feature.

VGG-16 is widely used, while DenseNet [16] represents the latest high-performance evolution of ResNet [15]-like designs. We use 67-layer and 121-layer DenseNets with the architectural details specified in Figure 3. The 67-layer net uses a channel growth rate of 48, while the 121-layer network, the same as in [16], uses a growth rate of 32. We work with 256×256 input and output spatial resolutions in both CNNs and our label autoencoder.

3.3. Regularization via Label Model

As shown by the large gray arrow in Figure 1, we impose our regularizer by connecting a CNN (*e.g.* VGG or DenseNet) to the decoder portion of our learned label autoencoder. Importantly, the *decoder parameters are frozen during this training phase*. The CNN now has two tasks, each with an associated loss, to perform during training. As usual, it must predict semantic segmentation using hypercolumns. It must also predict the same semantic segmentation via an auxiliary path through the decoder. Backpropagation from losses along both paths influences CNN parameter updates. Though they participate in one of these paths, parameters internal to the decoder are never updated.

We connect VGG-16 or DenseNet to the decoder by predicting input for the decoder from the output of the penultimate CNN layer prior to global pooling. This is the second-to-last convolutional layer, and is selected because its spatial resolution matches that of the expected decoder input. The prediction itself is made via a new 1×1 convolutional layer, dedicated for that purpose.

If the label autoencoder learns useful abstractions, requiring the CNN to work through the decoder ensures that it learns to work with those abstractions. The hypercolumn pathway allows the CNN to make direct predictions, while the decoder pathway ensures that the CNN has "good reasons" or a high-level abstract justification for its predictions.

Assuming autoencoder layers gradually build-up good abstractions, there exist alternative methods of connecting it as a regularizer. Figure 4 diagrams one such alternative. Here, we ask the CNN to directly predict the feature representation built by the label encoder. Encoder parameters are, of course, frozen here. An auxiliary layer attempts to predict the encoder hypercolumn from the CNN hypercolumn at the corresponding spatial location. The CNN must

also still solve the original semantic segmentation task.

As Section 4 shows, this alternative scheme works well, but not quite as well as using the decoder pathway. Using the decoder is also appealing for more reasons than performance alone. Defining an auxiliary loss in terms of decoder semantic segmentation output is more interpretable than defining it in terms of mean square error (MSE) between two hypercolumn features. Moreover, the decoder output is visually interpretable; we can see the semantic segmentation predicted by the CNN via the decoder.

4. Experiments

The PASCAL dataset [12] serves as our experimental testbed. We follow standard procedure for semantic segmentation, using the official PASCAL 2012 training set, and reporting performance in terms of mean intersection over union (mIoU) on the validation set (as validation ground-truth is publicly available). We explore both our decoder- and encoder-based regularization schemes in combination with multiple choices of base network, data augmentation, and pretraining. When applying the encoder as a regularizer, we task the CNN with predicting the concatenation of the encoder's activations in its conv1 and conv3 layers.

4.1. Setup

All experiments are done in PyTorch [1], using the Adam [20] update rule when training networks. Models trained from scratch use a batch size of 12 and learning rate of $1e^{-4}$ which after 80 epochs decreased to $1e^{-5}$ for an additional 20 epochs. For the case of ImageNet pretrained models, we normalize hypercolumn features such that they have zero-mean and unit-variance. We keep the deep network weights frozen and train the classifier for 10 epochs with learning rate of $1e^{-4}$. Then we decrease the learning rate to $1e^{-5}$ and train end-to-end for additional 40 epochs.

Data augmentation, when used, includes: a crop of random size in the (0.08 to 1.0) of the original size and a random aspect ratio of 3/4 to 4/3 of the original aspect ratio, which is finally resized to create a 256×256 image. Plus random horizontal flip. Pretrained models are based on the PyTorch torchvision library [2].

We use cross-entropy loss on auxiliary regularization branches, except where indicated by a superscript † in results tables. For these experiments, we use MSE loss.

4.2. Semantic Segmentation Results

Tables 1, 3, and 4 summarize the performance benefits of training with our regularizer. In the absence of pretraining or data augmentation, we boost performance of both VGG-16 and DenseNet-67 by 5.1 and 4.7 mIoU, respectively, which is more than a 10% relative boost. Regularization with our decoder still improves mIoU (from 58.8 to 60.6) of DenseNet-67 trained with data augmentation. To further show the robustness of our regularization scheme to the choice of architecture, we also experiment with an FCN [26] version of VGG-16, as included in Table 1.

Table 2 demonstrates the necessity of our two-phase training procedure. If we unfreeze the decoder and update its parameters in the second training phase, test performance of the primary output deteriorates. Likewise, if we skip the first phase, and train from scratch with an unfrozen, randomly initialized decoder, the accuracy gain disappears. Thus, the regularization effect is due to a transfer of information from the learned label model, rather than stemming from an architectural design of dual output pathways.

Architecture	Data-Aug?	Auxiliary Regularizer	mIoU
VGG-16 -hypercolumn	no	**none**	**37.3**
	no	Encoder (conv1 & conv3)†	41.1
	no	**Decoder (32 channel)†**	**42.4**
	yes	**none**	**55.2**
	yes	**Decoder (128 channel)**	**57.1**
VGG-16-FCN8s	yes	**none**	**51.5**
	yes	**Decoder (128 channel)**	**54.1**
DenseNet-67 -hypercolumn	no	**none**	**40.5**
	no	Encoder (conv1 & conv3)†	44.0
	no	**Decoder (32 channel)†**	**45.2**
	no	Decoder (128 channel)	42.5
	yes	**none**	**58.8**
	yes	Decoder (32 channel)	59.4
	yes	**Decoder (128 channel)**	**60.6**
	yes	Decoder (256 channel)	59.8

Table 1. **PASCAL mIoU without ImageNet pretraining.** In each experimental setting (choice of architecture, and presence or absence of data augmentation), training with any of our regularizers improves performance over the baseline (shown in gray).

Architecture	Data-Aug	Auxiliary Regularizer	mIoU
DenseNet-67 -hypercolumn	yes	**none**	**58.8**
	yes	**Decoder (128 channel)**	**60.6**
	yes	Unfrozen Decoder	60.2
	yes	Random Init. Decoder	58.8

Table 2. **Ablation study.** PASCAL mIoU deteriorates if the decoder parameters are not held fixed while training the main CNN.

Architecture	Data-Aug?	Auxiliary Regularizer	mIoU
VGG-16 -hypercolumn	no	**none**	**67.1**
	no	**Decoder (32 channel)**	**68.8**
DenseNet-121 -hypercolumn	yes	**none**	**71.6**
	yes	**Decoder (128 channel)**	**71.9**
ResNet-101 -PSPNet	yes	**none**	**75.4**
	yes	**Decoder (128 channel)**	**75.9**

Table 3. **PASCAL mIoU with ImageNet pretraining.**

Architecture	Data-Aug	Auxiliary Regularizer	mIoU
DenseNet-67 -hypercolumn	yes	**none**	**72.3**
	yes	**Decoder (128 channel)**	**73.6**

Table 4. **PASCAL mIoU with COCO pretraining.**

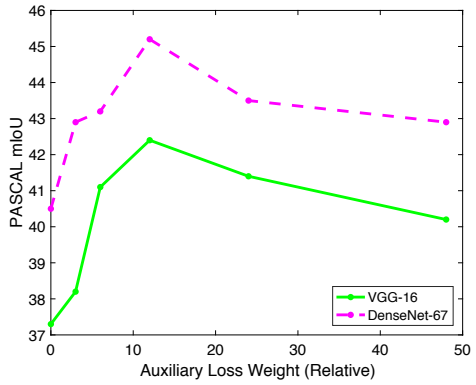

Figure 5. **Auxiliary loss weighting.** We plot test performance as a function of the relative weight of the losses on the auxiliary vs primary output branches when training with the setup in Figure 1. Weighting is important, but the optimal balance appears consistent when changing architecture from VGG-16 (green) to DenseNet-67 (magenta). Performance is mIoU on PASCAL, without ImageNet pretraining or data augmentation. Note that any nonzero weight on the auxiliary loss (any regularization) improves over the baseline.

Table 3 shows that our regularization scheme synergizes with ImageNet pretraining. It improves VGG-16 performance, and even provides some benefit to a very deep 121-layer DenseNet pretrained on ImageNet, while using data augmentation. A baseline 71.6 mIoU for DenseNet appears near state-of-the-art for networks that do not employ additional tricks (*e.g.* custom pooling layers [46], use of multi-scale, or post-processing with CRFs [7]). Our improvement to 71.9 mIoU may be nontrivial. Expanding trials in combination with pretraining, our regularizer improves results when pretraining on COCO, as shown in Table 4.

We also combine our regularizer with the latest network design for semantic segmentation: dilated ResNet augmented with the pyramid pooling module of PSPNet [46]. We used the output of the pyramid pooling layer to predict input for the decoder and semantic segmentation. Table 3 shows gain over the corresponding PSPNet baseline.

Beyond autoencoder architecture choice, application of our regularizer involves one free parameter: the relative weight of the auxiliary branch loss with respect to the primary loss. Figure 5 shows how performance of the trained network varies with this parameter, when using our 32-channel bottleneck layer decoder with MSE loss on the auxiliary branch.

We have also run similar experiments with cross-entropy loss on the auxiliary branch with the weight parameter in $[0, 6]$. Here, the weight parameter range is changed due to the difference in the dynamic range of values between MSE loss and cross-entropy loss. Behaving similarly to Figure 5, relative weighting of 0.5 achieves the highest accuracy. We use this weight value across all of the experiments using our decoder with 128-channel bottleneck layer. While the regularizer always provides a benefit, placing a proper relative weight on the auxiliary loss is important.

Figure 6 visualizes the impact of training with our learned label decoder as a regularizer. Most notably, the network trained with regularization appears to correct some global or large-scale semantic errors in comparison to the baseline. Contrast such behavior to CRF-based post-processing, which typically achieves impact through fixing local mistakes. Also notable is that our auxiliary output itself is quite reasonable. This suggests that the autoencoder training phase is successful in creating encoders and decoders that model label structure.

4.3. Label Model Introspection

To further investigate what the autoencoder learns, we consider using the bottleneck representation produced by the encoder as defining features by which we can perform queries in label space. Specifically, we pick a region of a training image label and represent that region with features extracted from bottleneck layer. As the bottleneck layer is low resolution, we are selecting features at coarse, but corresponding spatial location.

Next, we perform nearest neighbor search over all regions in the validation set and find the two closest regions to the query region. Figure 7 shows the results of this experiment. Returned regions not only have the same object class types as the query regions, but also share similar shapes to that of the query. This reveals that our label autoencoder has learned to capture object shape characteristics.

We also repeat this experiment, except with queries starting from images. Here the bottleneck representation is produced by a CNN, which was trained with both hypercolumn and decoder prediction pathways; the latter yields the required features. As shown in the top-right of Figure 7, returned regions have similar context and shape to the query.

5. Conclusion

Our novel regularization method, when applied to training deep networks for semantic segmentation, consistently improves their generalization performance. The intuition behind our work, that additional supervisory signal can be squeezed from highly detailed annotation, is supported by the types of errors this regularizer corrects, as well as our efforts at introspection into our learned label model.

Our results also indicate that one should now reevaluate the relative utility of different forms of annotation; our method makes detailed labeling more useful than previously believed. This observation may be especially important for applications of computer vision, such as self-driving cars, that demand detailed scene understanding, and for which large-scale dataset construction is essential.

Acknowledgements. This work was in part supported by the DARPA Lifelong Learning Machines program.

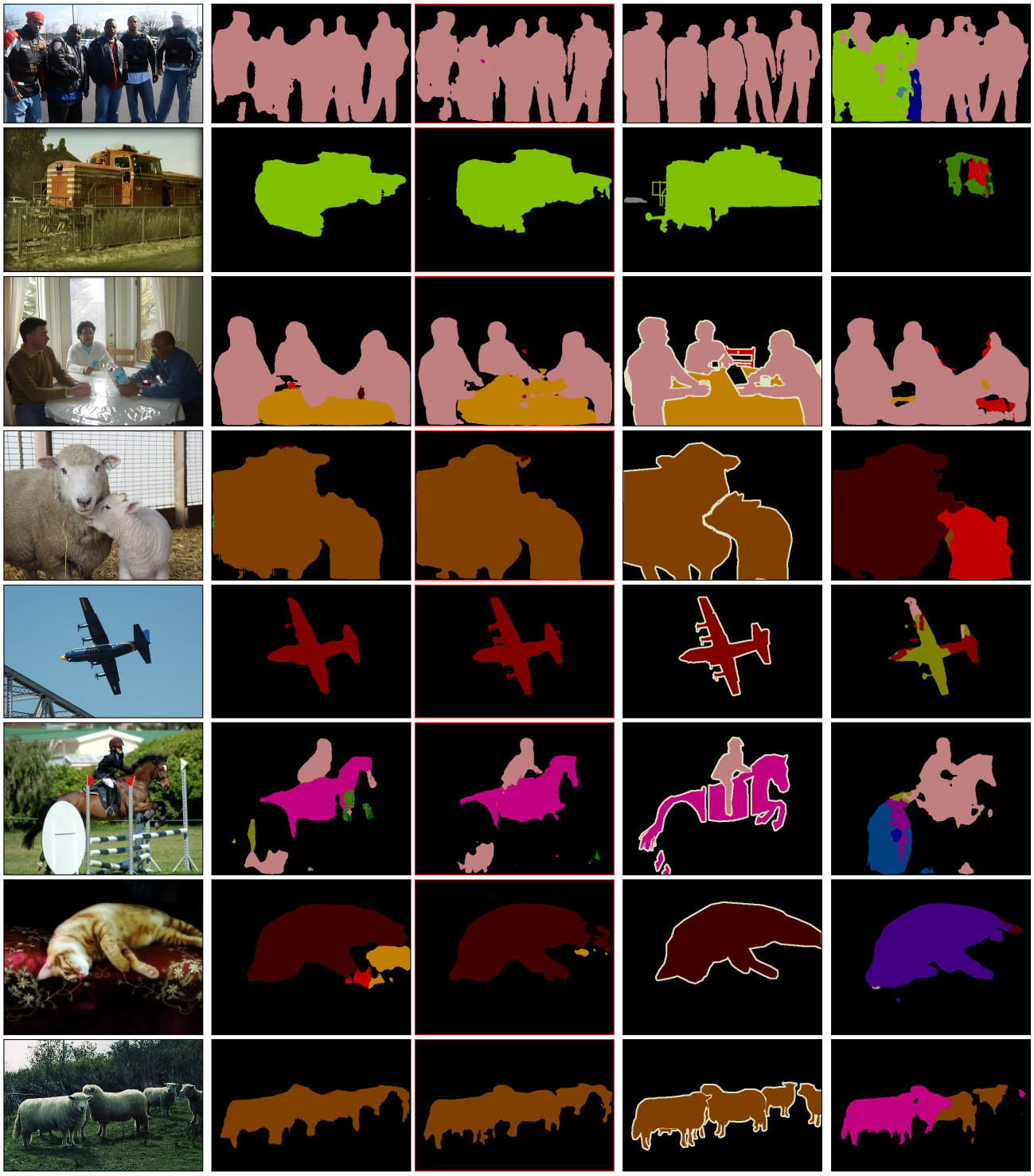

Figure 6. **Semantic segmentation results on PASCAL.** We show the output of a baseline 67-layer hypercolumn DenseNet (rightmost column) compared to that of the same architecture trained with our auxiliary decoder branch as a regularizer (middle columns). All examples are from the validation set. While we can discard the auxiliary branch after training, we include its output here to display the decoder's operation. Our network provides high-level signals to the decoder which, in turn, produces reasonable segmentations. To best illustrate the effect of regularization, all results shown are for networks trained from scratch, without ImageNet pretraining or data augmentation. This corresponds to the 40.5 to 45.2 jump in mIoU reported in Table 1, between the baseline and our primary output.

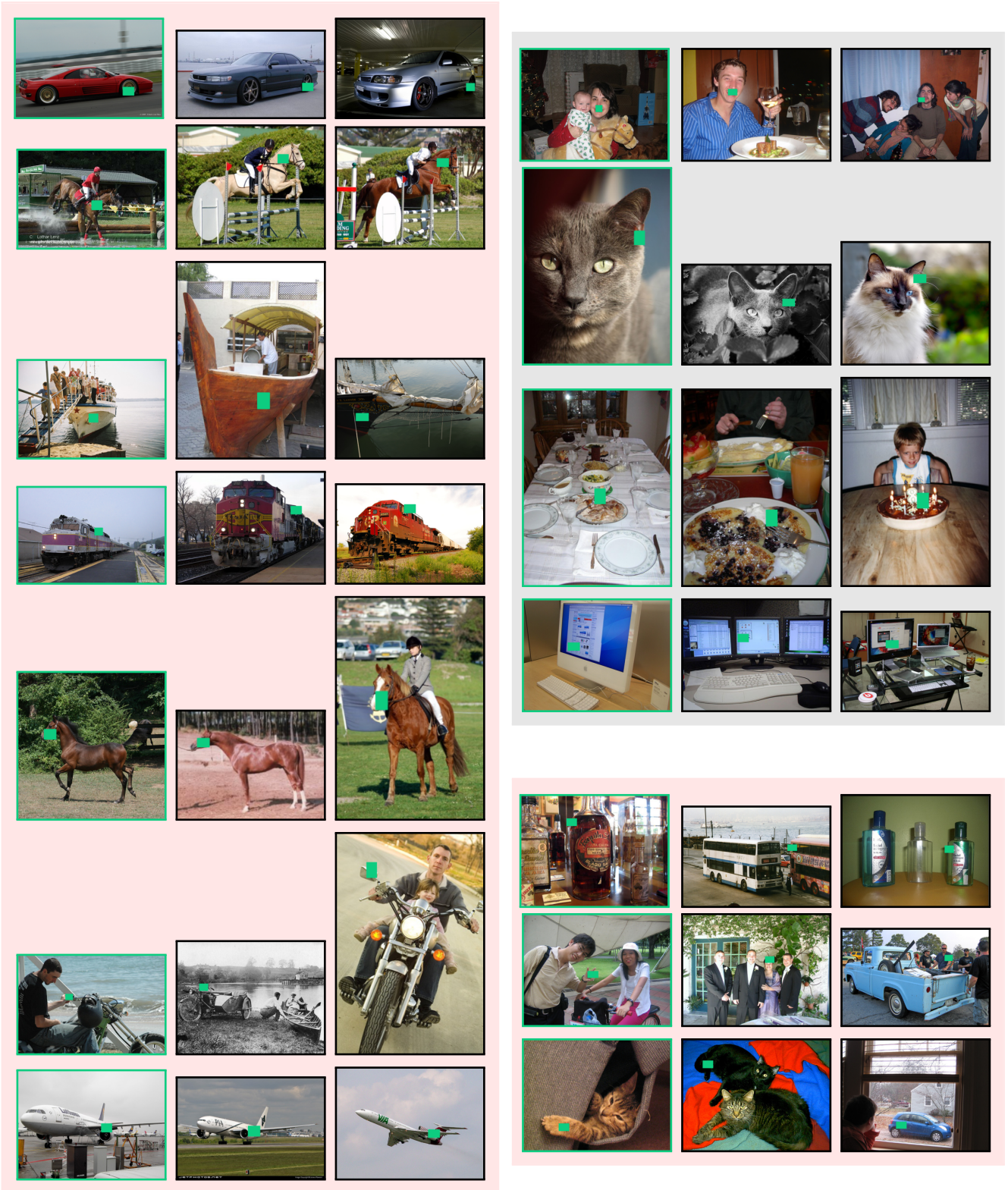

Figure 7. **Finding regions with similar representations.** For each **query** image (green border) and region (green dot), the next two images to the right are those in the validation set containing the nearest regions to the query region. All query images are from the training set. For examples on red background, search is conducted not by looking at images, but via matching features produced by the **encoder** run on ground-truth label maps. The bottom-right shows failure cases, such as matching a cat's arm to the car rear door. For examples on gray background, our **DenseNet-67-hypercolumn CNN** is used to predict the label space search representations from images.

References

[1] PyTorch. https://github.com/pytorch/pytorch.

[2] PyTorch torchvision. https://github.com/pytorch/vision.

[3] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *NIPS*, 2016.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *PAMI*, 2017.

[5] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. *ECCV*, 2010.

[6] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. *ICCV*, 2009.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *arXiv:1606.00915*, 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. *CVPR*, 2009.

[9] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. *IJCV*, 2011.

[10] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. *ICML*, 2014.

[11] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *ICLR*, 2017.

[12] M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *IJCV*, 2010.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *NIPS*, 2014.

[14] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *CVPR*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[16] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *CVPR*, 2017.

[17] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Learning visual groups from co-occurrences in space and time. *ICLR, workshop paper*, 2016.

[18] D. Jayaraman and K. Grauman. Slow and steady feature analysis: Higher order temporal coherence in video. *CVPR*, 2016.

[19] T.-W. Ke, M. Maire, and S. X. Yu. Multigrid neural architectures. *CVPR*, 2017.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *NIPS*, 2012.

[22] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. *ECCV*, 2016.

[23] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. *CVPR*, 2017.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. *ECCV*, 2014.

[25] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. *ICCV*, 2015.

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[27] I. Misra, C. L. Zitnick, and M. Hebert. Unsupervised learning using sequential verification for action recognition. *ECCV*, 2016.

[28] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. *ICML*, 2009.

[29] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. *CVPR*, 2015.

[30] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *ICRA*, 2017.

[31] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *ECCV*, 2016.

[32] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. *ECCV*, 2016.

[33] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. *CVPR*, 2017.

[34] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. *CVPR*, 2016.

[35] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[37] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. *ICML*, 2015.

[38] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects, and parts. *ICCV*, 2005.

[39] A. Torralba. Contextual priming for object detection. *IJCV*, 2003.

[40] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. *ICCV*, 2015.

[41] S. Xie, X. Huang, and Z. Tu. Top-down learning for structured labeling with convolutional pseudoprior. *ECCV*, 2016.

[42] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016.

[43] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *ECCV*, 2016.

[44] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *CVPR*, 2017.

[45] Y. Zhang, K. Lee, and H. Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. *ICML*, 2016.

[46] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CVPR*, 2017.