Unsupervised Semantic Segmentation through Cross-Instance Representation Similarity

by

Griffin Bishop

A Thesis

Submitted to the faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

May 2020

APPROVAL:

Professor Jacob R. Whitehill, Master Thesis Advisor

Professor Kyumin Lee, Thesis Reader

Professor Craig E. Wills, Head of Department

Unsupervised Semantic Segmentation through Cross-Instance Representation Similarity

Griffin Bishop [1]

Abstract

Semantic segmentation methods using deep neural networks typically require huge volumes of annotated data to train properly. Due to the expense of collecting these pixel-level dataset annotations, the problem of semantic segmentation without ground-truth labels has been recently proposed. Many current approaches to unsupervised semantic segmentation frame the problem as a pixel clustering task, and in particular focus heavily on color differences between image regions. In this paper, we explore a weakness to this approach: By focusing on color, these approaches do not adequately capture relationships between similar objects across images. We present a new approach to the problem, and propose a novel architecture that captures the characteristic similarities of objects between images directly. We design a synthetic dataset to illustrate this flaw in an existing model. Experiments on this synthetic dataset show that our method can succeed where the pixel color clustering approach fails. Further, we show that plain autoencoder models can implicitly capture these cross-instance object relationships. This suggests that some generative model architectures may be viable candidates for unsupervised semantic segmentation even with no additional loss terms.

[1]Worcester Polytechnic Institute. Correspondence to: Griffin Bishop <grbishop@wpi.edu>.

Contents

List of Figures

List of Tables

1. Introduction

In recent years, improvements to perception tasks such as object detection and classification in images ([31], [19], [28], [13]) have enabled many real-world computer vision applications. However, in other instances, the information afforded by these solutions alone is not sufficient; many applications require a fine-grained understanding of the spatial layout of images. Notable examples of these domains include perception in autonomous vehicles and medical imaging, where segmentation information is necessary. In semantic segmentation, the objective is to create a machine that consumes an image and for each pixel, predicts the object class to which that pixel belongs.

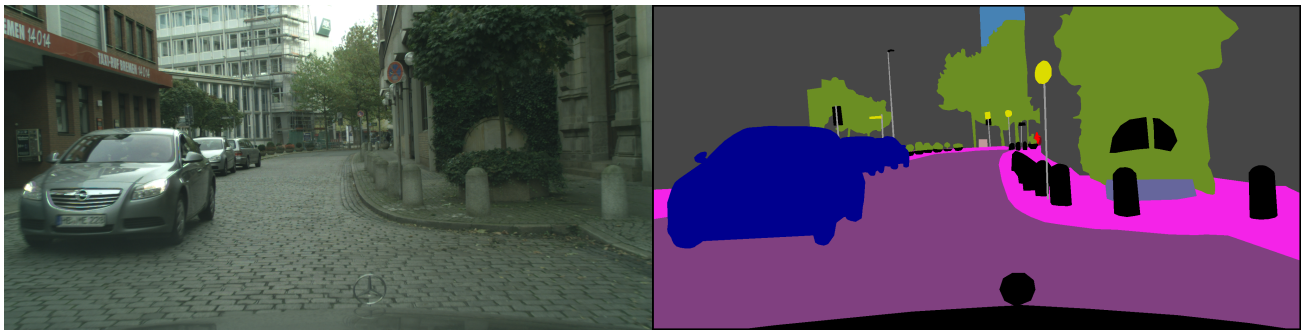
Although a great deal of excitement about the problem has developed in recent years with the prevalence of deep learning methods, early research on semantic segmentation is still relevant and inspires current methods. Researchers before the deep learning era characterized the problem as discovering image regions that are ”uniform and homogeneous with respect to some characteristic” [12]. Broadly, research on semantic segmentation focused on techniques such as histogram thresholding [6], region growing [1], edge detection ([3], [35]), and graph-based segmentation [10]. With the advent of deep learning, many have taken inspiration from these techniques.

Deep learning approaches to the problem can be broadly categorized as either supervised, wherein the model is supplied with images and pixel-level labels during training; weakly-supervised, wherein the model is supplied with images and image-level labels during training; and unsupervised, where the model is supplied with only images and no ground-truth annotations during training. Relevant prior research on each of these methods is described in detail in section 2.

While recent progress in segmentation models has resulted in useful, high-fidelity outputs in some domains ([20], [30], [29], [14], [33]), these deep neural network models rely on large-scale datasets with pixel-level annotations. These types of annotations are expensive to collect, because a human must attend to many different parts of the image and consider small, complicated regions. Figure 1 shows an example image paired with its ground-truth segmentation in the domain of perception in autonomous vehicles. Notice that in the portion of the image that is farther away from the camera, the scene becomes is complicated; very small regions can be composed of many different objects. Although there are tools which can speed up the annotation process, this inherent complexity makes collection segmentation annotations like the one in Figure 1 time consuming and expensive to collect.

The problem of perception in autonomous vehicles demonstrates this in particular because it has an exceptionally long-tailed event distribution. This represents a problem for scaling these models to work in real-world domains, because exceptionally large datasets are required to capture a workable portion of the event distribution. With autonomous vehicles, large volumes of raw images are available, yet resources allow for only a small portion of this to be labeled and used.

The expense of label collection has led many to consider approaches to the segmentation problem that require fewer annotations, or ones that are easier to collect. This is the goal of weakly supervised segmentation: to reduce this expense by requiring only image-level labels relating to the presence or absence of a given object. During training time, weakly-



(a) (b)

Figure 1: Image of a road scene (a) and its corresponding segmentation map (b) from the Cityscapes dataset [7].

supervised models are given access only to images and image-level object labels. For example, the left image in Figure 1 may be given, but instead of the segmentation map being given, the binary presence of each object would be given. In this example, the following objects would be given as present for the label: car, street, sidewalk, sign, pole, building, tree, sky, but the object "person" would not be given. The advantage of weakly-supervised segmentation is the ease of label collection, because during inference, the model still produces pixel-level labels. One of the drawbacks of the weakly-supervised paradigm is that it becomes much harder if a large portion of the same objects are present in most of the dataset images.

The paradigm of deep unsupervised segmentation has also recently been proposed [36] to reduce the expense of annotations and increase scalability. In the unsupervised paradigm, models are trained only on raw images, not requiring ground-truth annotations. Note that during evaluation, annotations are used to produce a performance metric. A viable unsupervised model would make segmentation models much more scalable in real-world environments. However, in the context of deep learning, this task has only recently been researched, and there is possibly a large amount of improvement even in the foundational assumptions of this version of the problem.

In terms of large, real-world datasets, the current state of the art method for unsupervised segmentation [16] reports a 27.7% pixel accuracy on COCO-stuff. Not surprisingly, supervised segmentation models generally outperform weakly-supervised models, which in turn outperform unsupervised ones. With this in mind, rather than aiming to immediately improve upon existing methods on real world datasets, we took the perspective of analysing the current research for possible weaknesses in direction. In this paper, our goal is to approach the problem from a *first principles* perspective, and challenge some of the assumptions of existing methods. We discuss these foundational assumptions at a high level in section 3, and show the theoretical implications of these assumptions with a low-dimensional example in section 4.

In section 5, we expand the objective formulated in section 4 to apply to high-dimensional data with occlusion. We call this the Cross-Instance Representation Similarity(CIRS) objective. With this reformulated objective, we propose a novel neural network architecture in section 6. In section 7, we describe a phenomenon present in some unsupervised deep neural network models and propose a training procedure to reduce these unwanted effects. We evaluate the CIRS model

trained on this procedure. In section 8, we compare the performance of our proposed model to an existing model reflecting the color contiguity focused approach. In section 9, we investigate the ability for plain autoencoder models to implicitly capture the cross-instance object relationships specified by the CIRS loss. Using the results from the previous section, we then formulate an alternative model design based on the bottleneck autoencoder architecture: the Argmax Individual Reconstruction (AIR) model. In section 11, we conclude with a discussion of our results and implications for future work.

2. Prior Work

There exist several main supervision paradigms for semantic segmentation: supervised, weakly-supervised, and unsupervised. In the supervised problem, the model has access to both images and their pixel-level annotations for training. In the weakly supervised problem, the model has access to images and object labels on the image-level. No labels are given on a per-pixel basis. In the unsupervised version of the problem, no ground-truth labels are given during training at all; only image data is provided.

2.1. Supervised

Broadly, supervised methods for semantic segmentation can be seen as being based on the following different methods. The first is fully convolutional networks (as in [20] or [30]) where an image is fed into a convolutional neural network (CNN), and the pixel-wise predictions are directly computed in the CNN. The model is trained with a loss function that minimizes the difference between the predicted segmentation maps and the ground-truth ones.

One such model, "U-Net" features an architecture with a contracting path followed by an expanding path [30]. On the contracting path, spatial resolution is down-scaled, while the number of filters is increased. The authors claim that this gradually increases the "what" information, while decreasing the "where" information. This is similar to the concept of the "bottleneck" of an autoencoder, which forces the model to learn important instance details while discarding common features. Next, on the expanding path, this trend is reversed, and the number of filters are decreased while the spatial resolution is increased until the result is produced with the same number of channels as classes. Skip-connections are used; the output of each module on the contracting path is copied to the input of each module on the expanding path. Additionally, U-Net takes in a larger region than the segmentation it produces, so that predictions for the inner region can be informed by the outer one. This overlapping tile method allows U-Net to process arbitrarily large images.

Supervised models are usually end-to-end, meaning that the segmentation is directly computed by the model, and no post-processing needs to be applied to the output in order to produce workable segmentations. In contrast, much of the work in unsupervised semantic segmentation requires post-processing steps (e.g. CRF smoothing) to achieve segmentations that match expectations of connectedness and smoothness, as in [34] and [16]. The method we propose is end-to-end and would not require hand-tailored post-processing methods.

2.2. Weakly Supervised

Weakly supervised methods also attempt to circumvent the high cost of obtaining segmentation labels. Instead of having to tediously attend to each pixel, weakly supervised methods allow one work with image-level labels. Given just the image-level labels on the training set, the goal is to train a machine that can estimate the class of every pixel in a test image.

The main category of weakly supervised methods can be seen as correlating features of image regions to the given image-level labels, as done in "From Image-level to Pixel-level Labeling with Convolutional Networks" ([26] and [17]). This process can be viewed as a refinement of the class activation maps [38] of networks pre-trained on an image classification

dataset. By augmenting a standard multi-class image classification model, it is possible to visualize how much each image pixel is used to calculate the final prediction. These activation maps usually do not overlay the object in fine detail, so several methods have been put forward to refine the map, including "adversarial erasing" of image regions [33], and post-processing smoothing techniques [17].

2.3. Unsupervised

Prior to the recent adoption of deep learning techniques, there were many attempts at segmentation without labels involving non-deep methods such as histogram thresholding, pixel-color based region growing, graph-cuts, and superpixels ([6],[2],[8],[9]). Deep learning methods have made great progress in this task, and many prevalent model concepts and post-processing methods are derived from this earlier work.

2.3.1. Clustering

One class of deep methods for unsupervised semantic segmentation views the task as an extension of clustering. These methods work by dividing the image into small, square samples that are individually fed into the model to generate a cluster label. This cluster label can be applied to all pixels within the patch, and this prediction can be refined by information from overlapping patch-level predictions. Some unsupervised clustering methods include autoencoders, principal component analysis, k-means [24], DeepCluster [4], and "Adversarial Autoencoders" [22]. At a high level, the goal is to find a representation that preserves salient information while discarding non instance specific details such that data of the same label share important characteristics.

One recent method in this category is "Invariant Information Clustering" (IIC) [16], which currently holds the state of the art for unsupervised image clustering and segmentation. IIC learns these important details shared between images of the same object by applying a label-preserving (cluster-preserving) transformation to the original image. The approach uses a loss based on mutual information between the predicted one-hot cluster labels which encourages the model to classify both the original image and its transformation into the same category. For the segmentation task specifically, the assumption is made that neighboring patches likely belong to the same object.

While the method works well for segmenting regions of uniform colors and textures, it relies on these lower level details; it has trouble interpreting complicated regions of different colors. This can be seen in IIC's performance difference between different variations of the COCO dataset. On the COCO-stuff-3 dataset, which considers only sky, ground, and vegetation regions, IIC achieves a 72.3% per-pixel accuracy. In contrast, on the more complicated COCO-stuff dataset, featuring complex regions of high color variance, the model achieves a 27.7% accuracy.

2.3.2. Autoencoders

The autoencoder architecture is another basis for unsupervised learning. Many approaches in the literature for unsupervised semantic segmentation build off of an encoder-decoder base model. One approach, called W-Net [34] took an

architecture which worked well for supervised learning, U-Net [30], and formed it into an encoder-decoder model, where the hidden representation between the encoder and decoder is the intended segmentation map. W-Net has an added loss term to encourage color contiguity in segmented regions. Post-processing techniques are used to transform the resulting hidden representation into a workable segmentation map. The cardiac image segmentation model [18], along with the CycleGAN model [39] can also be considered autoencoder architectures for self-supervised learning. The cardiac model is unique in that it uses adversarial learning to enforce that the distribution of predicted segmentations is indistinguishable from the distribution of true segmentations, similar to the CycleGAN technique.

2.3.3. Generative Adversarial Networks

Adversarial learning [11] is an unsupervised technique that has been found to discover interesting feature representations in an unsupervised manner. The Deep Convolutional GAN paper ([27]) shows how adversarial models have a capacity for unsupervised representation learning: in the process of learning to generate novel samples from the dataset distribution, the generator captures hierarchical representations of objects that can be used for downstream tasks.

Adversarial learning can also be used to directly improve segmentations in the supervised paradigm; the authors of [21] claim that their added adversarial loss can detect and correct higher order inconsistencies in the outputs. In the technique, a discriminator is trained simultaneously with a conventional supervised segmentation model. The segmentation model is trained using loss on the pixel labels for each image. Additionally, discriminator loss is computed to update the segmentation model. The discriminator ensures that the distribution of predicted segmentations is indistinguishable from the distribution of ground-truth segmentations. A variation of this method is applied in [18] and [39] as mentioned previously.

Figure 2: Image with tree, sky, and moon object classes from the Berkeley Segmentation Dataset [23].

3. Object Priors in Current Methods

Unsupervised semantic segmentation is a task where, given an image (without labels), the objective is to group the pixels into discrete, semantically meaningful sets, often corresponding to natural objects. This is quite different from the supervised version of the problem, where a label is given for each pixel. In the unsupervised case, not only does the model have to classify pixels, but it must also simultaneously discover the classes into which it will classify them. There are several different existing approaches to the unsupervised semantic segmentation task. Each method either explicitly or implicitly relies on a prior belief about the nature of objects.

One such method is W-Net. W-Net is an autoencoder model where dataset images are processed by one U-Net model (the encoder) to predict a segmentation, then this prediction is fed into a second U-Net model (the decoder). As mentioned in section 2.1, U-Net is a commonly used architecture for supervised object segmentation. A per-pixel reconstruction loss is used between the output of the decoder and the original image. Additionally, W-Net introduces a soft-normalized cut loss objective which encourages segmented regions to be uniform in color, while encouraging spatial contiguity. Figure 2 shows an image from the BSDS300 [23] dataset for which it makes sense to segment in this fashion.

The image will be adequately segmented into intuitive object regions by the color clustering method. In this image, one would recognize the sky, moon, and trees as the objects present. The high uniformity of color within these objects, and sharp contrast in color at object borders makes it a great candidate for color-based segmentation.

This objective illustrates a particular prior belief about the nature of objects: intuitively, one *does* expect objects to be spatially contiguous and relatively uniform in color. However, despite many existing approaches explicitly or implicitly privileging these priors, they are not necessarily sufficient. The purpose of these experiments is to show that these color-based priors are sub-optimal, and in fact unworkable in some important cases.

3.1. The Nature of Objects

As discussed above, current methods can be seen as implying a view about the nature of objects. The spatial contiguity and segment color uniformity objectives imply another more subtle view. They work on the image-level, meaning that

(a) (b)

Figure 3: Humans with colored clothing (a). Two flowers with distinctly colored regions (b). Images shown are from the Berkeley Segmentation Dataset [23].

patterns across the dataset are not explicitly taken into account. Intuitively, this implies that an object's identity is defined by its color. Consider a scenario in which an object always has two regions of distinct color across the dataset, e.g. a dataset of humans wearing red shirts. The spatial contiguity and color uniformity priors would encourage the segmenting of the face into a different region than the shirt, despite the shirt pattern only ever appearing coincident with the face pattern. This example is also a significant problem in supervised semantic segmentation. Two example images illustrating this from the BSDS300 dataset are shown in Figure 3a.

As one can see in the images, there is a sharp contrast in color between the borders of the humans and their surroundings, which supports the color prior. However, there are also sharp differences in color within regions of the humans: the shirt region is quite different from the face region. One could argue that the shirt should be segmented into a separate region, but this causes another problem. In the image on the right, having the shirt as a separate object would disconnect the face from the hands, which is a conflict with the spatial contiguity prior. This illustrates a problem with segmentation methods based on color contiguity: they cannot capture these higher-order, non-color based relationships.

An alternative prior can be described as segmenting by cross-instance object similarity. Rather than privileging color differences in individual images, we seek to discover segments that feature the most frequent object patterns present across the entire dataset.

Consider an example of a dataset of flower images similar to the one in Figure 3b. In this example, one would discover a single frequent pattern in the dataset: the flower as a whole, because the outer petals always appear coincident with the inner petals. The fact that the two major regions are very different in color is irrelevant. Instead of implying that an object's color distribution is paramount to its identity, we imply that an object's identity is defined in relation to the common characteristics of that object across all appearances.

In the next section, we present a concrete example in one dimension that illustrates the differences in these two paradigms and their implications.

4. One-Dimensional Proof

4.1. Color Clustering Approach

Here we imagine an example in the one-dimensional case. Our pixels are integers, and we work with images of $L = 6$ cells. In order to allow for objects to be separable in one-dimension, we allow for a background segment which is already known, and may be occluded by objects. Take "0" as this background symbol. Our example will feature k non-occluded objects, with every object present in each image exactly once.

Let $O = \{11, 22\}$ be a set of $k = 2$ objects from which we can generate the following dataset of $n = 5$ possible images (an exhaustive list is not necessary):

$$
\begin{aligned}
D = [& \\
& [(0,1),(1,1),(2,0),(3,0),(4,2),(5,2)], \\
& [(0,0),(1,1),(2,1),(3,2),(4,2),(5,0)], \\
& [(0,0),(1,0),(2,2),(3,2),(4,1),(5,1)], \\
& [(0,0),(1,2),(2,2),(3,1),(4,1),(5,0)], \\
& [(0,2),(1,2),(2,0),(3,0),(4,1),(5,1)] \\
&]
\end{aligned}
$$

We represent an image as a list of pixels, where each pixel is a (location, value) tuple. For simplicity of notation, the above can be written with the location index of each pixel below the value, as follows:

$$
\begin{aligned}
& D = \\
& [110022, 011220, 002211, 022110, 220011] \\
& \ 012345, 012345, 012345, 012345, 012345
\end{aligned}
$$

We represent a dataset segmentation as a list of image segmentations, where an image segmentation is a list of segments. A segment is a list of pixels sorted by location. The index of each segment within an image can be thought of as a unique identifier that is assigned to each object and persists across images. Segments with the same index correspond to the same object. The following segmentation example shows this correspondence:

$$
\begin{aligned}
& S(D) = [\\
& \quad [[11],[00],[22]], \\
& \quad\ \ 01 \quad 23 \quad 45 \\
& \\
& \quad [[11],[00],[22]], \\
& \quad\ \ 12 \quad 05 \quad 34 \\
& \\
& \quad [[11],[00],[22]], \\
& \quad\ \ 45 \quad 01 \quad 34 \\
& \\
& \quad [[11],[00],[22]], \\
& \quad\ \ 34 \quad 05 \quad 12 \\
& \\
& \quad [[11],[00],[22]] \\
& \quad\ \ 45 \quad 23 \quad 01 \\
&]
\end{aligned}
$$

We seek to find the segmentation $S(D)$ which minimizes the following quantity:

$$min_{S(D)} \quad \left[\alpha_1 \sum_i \sum_{k=0}^{K_i-1} \text{var}(S(D)_k^i) - \alpha_2 \sum_i \sum_{k=0}^{K_i-1} \frac{J_i(k)}{K_i}\right]$$

Where:

- S is a function that maps a dataset D to a segmentation $S(D)$
- $S(D)_k^i(j)$ denotes the value of the j^{th} pixel in the k^{th} segment of the i^{th} image
- $\overline{S(D)_k^i} = \frac{\sum_{j=0}^{J_i(k)-1} S(D)_k^i(j)}{J_i(k)}$ is the mean pixel value of the k^{th} segment of the i^{th} image
- $\text{var}(S(D)_k^i) = \sum_{j=0}^{J_i(k)-1}(S(D)_k^i(j) - \overline{S(D)_k^i})^2$, the variance of colors in the k^{th} segment of the i^{th} image
- α_1 and α_2 are hyperparameters weighing the importance of the variance objective against the size of segments discovered. $\alpha_1, \alpha_2 = 1$ is satisfactory for the purposes of the example shown.

Note that the sizes of segments may not be uniform (there may be a size 4 segment and a size 2 segment in each image). We use the following to denote these differences:

- $J_i(k)$ is the number of pixels in the k^{th} segment of the i^{th} image
- K_i is the number of segments in the i^{th} image

In the color clustering paradigm, we seek to discover the largest segments in which (non-background) pixels are spatially connected, while minimizing the variance of color within segments. For example, according to the objective, the optimal segmentation for the image 110022 is $[[11], [00], [22]]$, because the variance within each segment is zero, and no larger segment has zero variance. For simplicity in these examples, we enforce the hard constraint that the color variance is exactly zero, but this could be extended to weigh color variance vs segment size.

Another candidate segmentation is $[[1], [1002], [2]]$, but the variance within the segment 1002 is nonzero, so it is not optimal.

Another candidate segmentation is each pixel individually: $[[1], [1], [0], [0], [2], [2]]$. The variance within each segment is also exactly zero, but the average segment size is less than 2 (as in the segmentation above), so it is not optimal.

4.1.1. EXAMPLE: COLOR CLUSTERING FAILURE

Given the object "1122", one can generate the following dataset of images of 6 cells:

$$D = [[112200], [011220], [001122]]$$

As this is the color clustering paradigm, we will discover the largest segments in which pixels are spatially connected, while minimizing the variance of color values within those segments. The optimal segmentation based on this objective groups $[00]$, $[11]$, $[22]$ together, and is then $[[11], [22], [00]]$ for the first image.

The actual solution is $[[1122], [00]]$ because we generated images containing the exact object $[1122]$. However, this is suboptimal by the color objective because the variance within the $[1122]$ segment makes the average variance greater than the $[[11], [22], [00]]$ solution, which has zero average variance. Despite the pattern "1122" only ever appearing as one connected object, the best way to segment by color divides the object. This is analogous to the human/shirt situation; "11" could be the head/face in this case, and "22" could represent the red shirt.

4.2. Cross-Instance Representation Similarity Approach

To correctly segment the "1122" object example, we need a method that is based on the frequency of the component patterns in the dataset. Whereas the above objective can be calculated locally for each image, we propose the cross-instance representation similarity objective, which takes into account variance across the dataset. From all possible segmentations with spatially contiguous object segments, we find the partition which optimizes the following quantity:

$$min_{S(D)} \left[\alpha_1 \sum_{k=0}^{K_i-1} \sum_{j=0}^{J_i(k)-1} \text{var}_k[S(D)(j)] - \alpha_2 \sum_i \sum_{k=0}^{K_i-1} \frac{J_i(k)}{K_i} \right]$$

Where

- $\text{var}_k[S(D)(j)] = \sum_i (S(D)_k^i(j) - \overline{S(D)_k(j)})2$ is the sum of variances of the pixel in the j^{th} location of the k^{th} segment for the matched segments across the whole dataset
- $\overline{S(D)_k(j)} = \frac{\sum_i S(D)_k^i(j)}{I}$ is the mean value of the j^{th} pixel in the k^{th} segment across all images
- α_1 and α_2 are hyperparameters weighing the importance of the variance objective against the size of segments discovered. $\alpha_1, \alpha_2 = 1$ is satisfactory for the purposes of the example shown.

4.2.1. EXAMPLE: CROSS-INSTANCE REPRESENTATION SIMILARITY SUCCESS

Using the example the color clustering objective failed on, we start with the object "1122" and generate a dataset of images of 6 cells:

$$D = \underset{012345,}{[[112200]}, \underset{012345,}{[011220]}, \underset{012345}{[001122]]}$$

Intuitively, the objective is to find the largest segments in which the variance of aligned pixels is zero. Following this, one can see that the optimal segmentation would be

$$D* = [\ \underset{0123}{[[1122]}, \underset{45}{[00]}],\ \underset{1234}{[[1122]}, \underset{05}{[00]}],\ \underset{2345}{[[1122]}, \underset{01}{[00]}\ ]$$

because the variances of aligned pixels in corresponding segments is exactly 0. Other segmentations are possible with zero variance, such as the following:

$$D* = \underset{01}{[[11]}, \underset{23}{[22]}, \underset{45}{[00]}], \underset{12}{[[11]}, \underset{34}{[22]}, \underset{05}{[00]}], \underset{23}{[[11]}, \underset{45}{[22]}, \underset{01}{[00]}]$$

However, the average segment size is smaller (2) than for the above (3). Note that this solution is exactly the failed solution from the color clustering objective. An explicit procedure for discovering segments with this method is given in appendix section 12.1.

5. Extension to Natural Images

In the one-dimensional example, one was able to compare pixel locations directly across images because no occlusion was allowed; the pixel positions were fixed relative to the location of the object. In natural images, however, objects occlude not only other objects, but themselves as well, due to rotation in 3D space. In this scenario, no such pixel alignment is possible, because the same or similar objects may appear in a variety of different orientations across images. Moreover, a single object may appear in a variable number of pixels in different images, depending on orientation and occlusion. Instead of discovering patterns exactly equal across images, one must discover probabilistically similar patterns throughout the dataset.

To be able to compare patterns across images in this case, we require fixed-length feature representations of objects. We find a function g which transforms some subset of image pixels (containing an object) I' into a fixed-length feature representation vector $g(I')$. In our experiments, we learn function g as part of a larger deep neural network, trained end-to-end with mini-batch stochastic gradient descent.

Another consideration in the extension beyond the one-dimensional, non-occluding case is the number of objects K one intends to discover. In the 1-d example, this number was discovered as part of the objective. However, in our model, the number of segments implies the dimensions of convolutional layers. It must be known at run-time, so we manually specify K as a hyperparameter, instead of including it as part of the optimization problem. After considering the stated differences, one can write the new objective, extended for use on object feature representations in natural images, as follows:

$$min_{S(D)} \quad \left(\frac{\sum_{k=0}^{K-1} \text{var}_k[S(D)]}{K} \right)$$

Where

- $\text{var}_k(S(D)) = \sum_{i=0}^{I-1}(g[S(D)_k^i] - \overline{g[S(D)_k]})^2$ is the variance of feature representations in segment k across all images in the dataset
- $\overline{g[S(D)_k]} = \frac{\sum_{i=0}^{I-1} g\left(S(D)_k^i\right)}{I}$ is the mean feature representation of the k^{th} segment across all images in the dataset
- g is a function that maps sets of pixels(segments) to feature representations. Note that each set of pixels passed to g belongs to exactly 1 predicted object.
- $S(D)_k^i$ is the set of pixels in the k^{th} segment of the i^{th} image produced by segmentation function S

For the previously described one-dimensional example, this objective was sufficient. However, when dealing with high dimensional data with occlusion, this objective by itself is not sufficient. To deal with the issues arising from comparison

objects in natural images, we instituted a representation function g. We want to use g to be useful to minimize the loss, but also to be useful as an object representation. In addition to the extension below, we discuss further implications of this design in section 7. If the objective is for g to minimize the variance of object representations in each object class separately, g could trivially map every object to zero regardless of object class. This would perfectly minimize the loss, but would completely discard the function's usefulness as an object representation. This was not a problem in the 1-d example because the object representations were the exact candidate objects themselves, and every possible partition of pixels was evaluated.

As this is not computationally feasible in the high dimensional case, we prevent this degenerate minimization of total variance by adding that the variance of mean object representations should be maximized. The final cross-instance representation similarity objective is shown below:

$$min_{S(D)} \left[\alpha_1 \left(\sum_{k=0}^{K-1} \text{var}_k(S(D)) \right) \frac{1}{K} - \alpha_2 \sum_{k=0}^{K-1} \left(\overline{g(S(D)_k)} - \left[\sum_{k=0}^{K-1} \overline{g(S(D)_k)} \right] \frac{1}{K} \right)^2 \right]$$

Where

- α_1 is a hyperparameter weighing the term to *minimize the mean variance of segment feature representations of the same class*
- α_2 is a hyperparameter weighing the term to *maximize the variance between mean feature representations of different classes*

5.1. Experimental Design

The paradigm of unsupervised semantic segmentation is unique in its requirements for evaluation, and is an extremely challenging task on large, real-world image datasets (one state-of-the-art model reports 27.7% per-pixel segmentation accuracy on the COCO-stuff dataset [16]). With the current state of performance on real world data, it would be difficult to clearly illustrate the difference in consequence of the two approaches. Instead, we opted to design a synthetic dataset to concretely show a comparison of the theoretical differences in approach.

5.1.1. Synthetic Dataset

We design a simple synthetic dataset to concretely illustrate the differences in approach. We have 4 different objects (sky, lemon, apple, blueberry).

Figure 4: Example generated images from the synthetic dataset.

This dataset is intended to show a failing of the color-clustering based approach in a simple case, much like the one-dimensional example. The sky object has high internal color variance, so despite it appearing cohesively in every image, one would expect color clustering methods to produce an oversegmentation of the sky into multiple segments, each with low internal color variance. For a model built on the cross-instance object representation similarity approach, one would expect a segmentation into the classes as given, because the appearances of these objects have minimal variance over the dataset, despite occlusion and augmentation.

We generate 64x64 pixel images with the object images placed at a random position and a random rotation on top of the background image. The fruit objects occlude the sky object. We generate 10000 training images, 1500 validation images, and 1500 test images. For the validation and test images, we also compute the ground-truth segmentations during image generation. Some example images generated for the dataset are shown in Figure 4.

5.1.2. Evaluation

With supervised semantic segmentation, performance is evaluated either using per-pixel accuracy, or by computing the mean intersection over union (mIoU) of predicted segment pixels.

Given an image, the segmentation model produces a segment mask for each class in the image. The segment mask is 1 for every pixel part of the object in question, and 0 otherwise. The intersection is computed by taking the area (sum) of the element-wise product of the predicted segment mask and the ground-truth segment mask. The union is computed by taking the sum of both masks, and subtracting the intersection. Therefore, the quotient of these values will be 1 if the segmentation masks are exactly equal, and zero if no pixels are shared. A formula for the mIoU metric on binary segment masks for a single image is given below:

$$\mathrm{mIoU}\big(S(D)\big) =$$

$$\left(\sum_{k=0}^{K-1}\left[\frac{\sum_{j=0}^{J-1} S(D)_j^k * \widehat{S(D)_j^k}}{\sum_{j=0}^{J-1}\left(\left[S(D)_j^k + \widehat{S(D)_j^k}\right] - \left[S(D)_j^k * \widehat{S(D)_j^k}\right]\right)}\right]\right)\frac{1}{K}$$

Where

- $S(D)_k^j = \{1$ if pixel j of segment k is predicted to belong to object k, 0 otherwise$\}$
- $\widehat{S(D)}_k^j = \{1$ if pixel j of segment k belongs to ground-truth object k, 0 otherwise$\}$
- K is the number of ground-truth segments
- J is the number of pixels in the image

The advantage of the mIoU metric is if one of the masks completely covers the other, but contains additional pixels, it will be penalized. In contrast, per-pixel accuracy allows for degenerate solutions where one segment covers everything. In the dataset presented above, a solution like this could be imagined; if all pixels were predicted as part of the sky object, this would result in a very high per pixel accuracy because the sky takes up the majority of the pixels, despite incorrectly labeling 100% of the other three object's pixels.

The unsupervised case presents an additional challenge to evaluation. The model is given no labels, so it outputs segments with no predetermined order. The challenge therefore arises because one does not know which predicted segment corresponds to which ground-truth segment in each image. To account for this, it is possible to simply compute mIoU values for all potential bipartite mappings, and take the maximum. For larger values of K, there are other methods which are computationally feasible [5]. Though labels are used, the consequence of the procedure is solely to provide invariance to the order of the labels, so it does not comprise supervision.

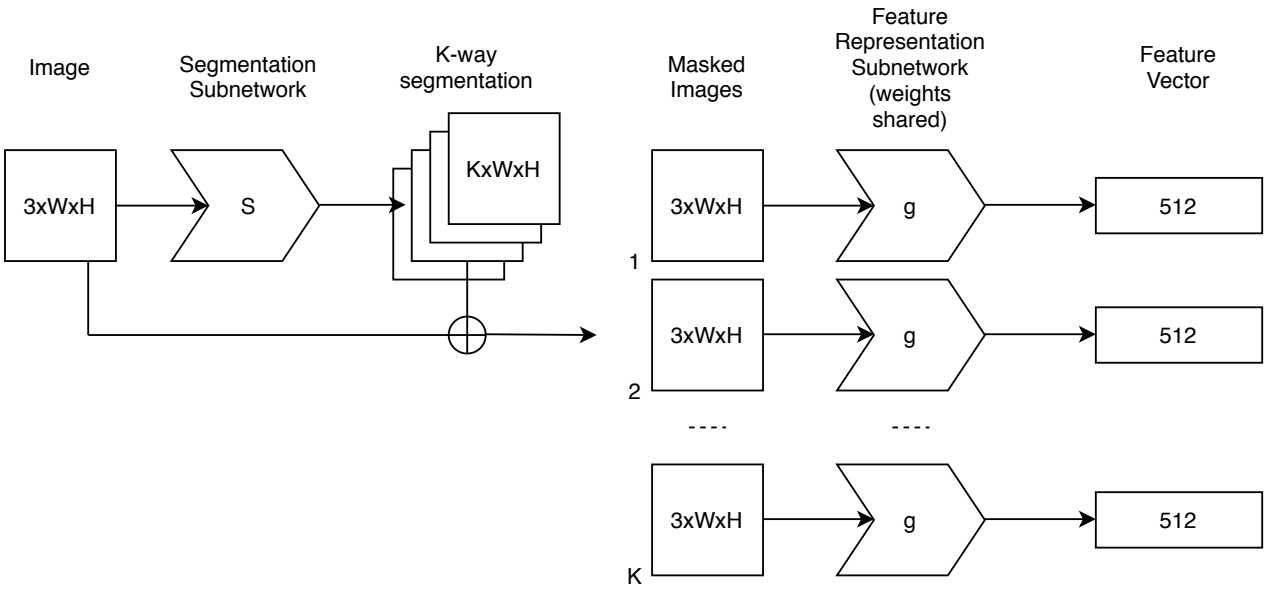


Figure 5: Diagram of the CIRS loss model

6. Cross-Instance Representation Similarity loss model (CIRS)

$$D_i \ \rightarrow_S \ S(D_i) \ \rightarrow_g \ g[(S(D_i)]$$

The goal of the training procedure is to learn a function S which consumes an image D_i and produces a segmentation $S(D_i)$ that is as close as possible to the ground-truth segmentation. However, we do not have access to the ground-truth segmentation labels, so we instead learn the segmentation function $S(D_i)$ as part of a larger neural network with an alternative objective. We use the cross-instance representation similarity loss by simultaneously training the required segment feature representation network g with the segmentation network S.

Each image is passed forward through the segmentation subnetwork S, which produces an output of size (K, W, H). We use the U-Net architecture, described previously in section 2.1. A softmax function is applied along the first dimension of this output. This output can then be interpreted intuitively as a vector of length K probabilities that each pixel in the image belongs to each of the K segments. We then apply the argmax-in-place function: for each pixel, we set the largest value to 1, and set the other $K-1$ values in the dimension to 0. This results in K segmentation masks (one for each object) of size (W, H). Next, we compute the element-wise product of every mask with the original image. Each of these K masked object images is concatenated with its segment number k and passed forward through the fully connected feature representation subnetwork g. This produces the feature vectors upon which the cross-instance representation similarity loss can be calculated. One could use a convolutional g subnetwork, but we found that a fully connected network was more

```python
# Differentiable argmax-in-place gradient approximation
class ArgmaxInPlace(torch.autograd.Function):
    @staticmethod
    def forward(ctx, input):
        with torch.enable_grad():
            softmax = torch.nn.functional.softmax(input, dim=1)
        ctx.save_for_backward(softmax)

        argmax = softmax.max(1)[1]
        segmentation = torch.nn.functional.one_hot(argmax, k).float()
        return segmentation

    @staticmethod
    def backward(ctx, grad_output):
        softmax, = ctx.saved_tensors
        return softmax.grad_fn(grad_output)
```

Figure 6: Pytorch autograd code for the argmax gradient approximation

workable for our experiments on small images.

6.1. Argmax-in-place Gradient Approximation

The whole model is trained end-to-end with backpropagation. Note that the argmax operation that takes place between the S and g subnetworks is crucial to the training procedure so that discrete separation of predicted segments can take place, but introduces a discontinuity in the neural network. To make it differentiable, we approximate the gradient on the backward pass by directly copying the gradients over the discontinuity. This is workable because of the softmax function directly before the discontinuity, which approximates the argmax-in-place function in a useful way: softmax converts the range of values to $(0, 1)$ with a sum of 1. The largest argument value becomes the closest to 1. In the argmax-in-place function, the result is analogous, but the largest argument value becomes exactly 1, and all others exactly 0. In this fashion, during backpropagation, the hard function is differentiated as if it were a softmax function.

To our knowledge, the argmax-in-place gradient approximation is novel to unsupervised methods, and believe that the approximation could be useful to future semantic segmentation models, especially in those which are unsupervised or where it is useful to perform a segmentation in the middle of the network as opposed to the end. There are times in both the supervised and the unsupervised case where one would like to use a loss function that is not trivially made differentiable. The approximation could be used in these cases. The approximation is simple to implement in the Pytorch framework by defining a new autograd function, as shown in Figure 6.

6.2. CIRS Model Experiment

We run the model on the synthetic dataset (section 5.1.1) for 170 epochs, calculating the mIoU metric after every epoch on the validation set. Training hyperparameter details are given in appendix section 12.2. We save the model with the best performance over all epochs, and finally compute the mIoU metric for this model on the test set.

6.2.1. RESULTS

The best performance achieved was a mean intersection over union (mIoU) of 45.81%. Four example images and their segmentations predicted by the model are shown in Figure 7.

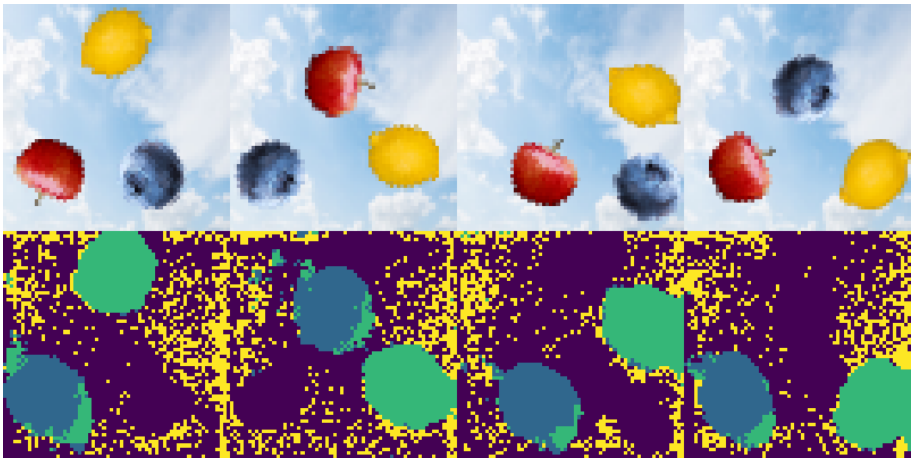


Figure 7: Output of the cross-instance representation similarity loss model

Table 1: Performance of initial CIRS model.

Cross-Instance Representation Similarity (CIRS) Loss Model
45.81%

7. The Representation Overfitting Problem

From this experiment, we discover a problem in unsupervised models which to our knowledge has not been previously described in the literature. We call this problem "representation overfitting," because it is analogous to overfitting in supervised learning. However, it is important to realize that conventional overfitting can occur in the unsupervised setting as well, and can occur alongside with representation distribution overfitting. For the purposes of this section, we will call conventional overfitting "label overfitting," and the presented phenomenon as "representation overfitting."

7.1. Supervised Paradigm Overfitting

In deep neural network models, one or more optimization objectives are specified on the output representations of one or more layers. In other words, one identifies some goal for the characteristics of the layer distributions. In these terms, it is useful to imagine a neural network as computing two discrete functions in series; first, it computes a feature representation on the input, then it predicts a label given this feature representation. In actuality, these functions are not neatly separable. Multiple layers work in concert to both learn intermediate feature representations and to transform them into label predictions. However, a difference exists conceptually, and we identify the two separate conceptual objectives in deep neural network models below:

1. **Feature Representation Objective**: Encourage the output distribution (and implicitly, the feature representation distribution) to match a prior distribution

2. **Label Correspondence Objective**: Encourage the input of the model to correspond with its output label

In supervised learning, the label correspondence objective is explicit, and the representation objective is implicit. Humans have a general prior distribution in mind for the feature representation when they annotate the dataset. For example, consider a task to train a model to determine whether a cat is in the given image or not. For ideal generality, the feature representation would learn the characteristics of a cat relative to any other object typically encountered by humans. However, given a dataset of finite size, the model objective is instead only to learn a feature representation sufficiently adequate to distinguish features present in the cat images from all other instances of objects present in the dataset. Because of this, it may be sufficient for the model to only attend to a small portion of the cat; perhaps the ears are all that is required to distinguish the cat object from any other object in the dataset. Or maybe, every cat image, by chance, has some other feature which no non-cat image has present. A model that uses these features instead of what one would see intuitively as "cat" features is described as "overfitting," because the model will not generalize to images where the presence of the cat object is not correlated with this feature.

In this way, the term "overfitting" as conventionally used in the supervised setting describes a scenario in which the model learns an exact correspondence between the input labels and output labels, but in doing so fails to learn a model with a feature representation matching the intuitive prior distribution. Of the objectives shown above, it fully achieves the

label correspondence objective, yet ignores the representation objective completely.

7.2. Unsupervised Paradigm

In the supervised paradigm above, the label objective is explicit, while feature distribution objective is often implicit, and ignored in the case of label overfitting. In unsupervised deep learning, however, no labels are available, so a common approach is to explicitly describe the desired representation distribution. Note that nothing prevents one from explicitly defining a desired distribution in supervised learning; in fact, some recent research on domain generalization has focused on doing this ([32], [15]).

If we only provide an objective that describes a goal representation distribution (as we do in the CIRS model above), the model can learn the distribution, but would be able to minimize the loss even in a case where inputs were mapped to outputs arbitrarily. The goal is to design the architecture and training procedure such that this correspondence has a high likelihood to match human intuition (ground-truth) without using a signal from annotations.

This pattern is present in one previously mentioned model [18], where the goal is to predict segmentations of cardiac images using unpaired data. To clarify, in this formulation of the problem, the model is given access to a set of unlabeled cardiac images, and a set of cardiac segmentations that do not correspond to the cardiac images. The model can be thought of as unsupervised in the sense that it is not given labels for each pixel during training, but it still requires a collection of example segmentations for training. It accomplishes the task with a neural network architecture with several different modules; a segmentor module, a discriminator module, and a reconstructor module. The segmentor module maps the input image to a predicted segmentation. The distribution of outputs from this module is encouraged to be indistinguishable from the distribution of real segmentations by the discriminator module, which is trained adversarially. This technique is described in [22]. The reconstructor module maps the predicted segmentation back to match the input image through L2 loss.

The discriminator module specifies the goal representation distribution. If this was the only objective, then representation overfitting would occur – the discriminator objective ensures that the output of the first module will look like a segmentation, but does not directly enforce that the segmentation should match the input image. Here, an arbitrary mapping may be just as good as the ground-truth in terms of loss.

This is addressed with an auxiliary reconstruction objective which has the effect of encouraging the intermediate distribution (the segmentation) to maintain an intuitive correspondence to the output. In this case, an arbitrary mapping would not be as likely as a mapping to the ground-truth, since reconstruction would be more difficult with an arbitrary mapping. In other words, the original model without reconstruction produces a mapping between images and segmentations that is under-constrained. By adding the inverse mapping back to the input image (the reconstruction objective), we may sufficiently constrain the mapping. This idea is discussed in detail with respect to adversarially learned distributions in the CycleGAN paper [39].

However, it is important to note that the auxiliary reconstruction object does not completely preclude the issue. The representation can still be overfit to the additional reconstruction objective if the model learns an arbitrary invertible mapping. We hypothesize that conventional techniques used to combat label overfitting in supervised models, such as dropout and regularization, could also be used to reduce representation overfitting when the problem is sufficiently constrained.

With the added auxiliary objective, the model not only has the goal distribution specified, but also increases the probability of an intuitive correspondence between inputs and outputs. The model's results on the multi-modal whole heart segmentation challenge dataset are very compelling, but since it requires unpaired segmentation data, it is not useful for cases where the goal is to eliminate the need to collect a large set of segmentation annotations [18].

In conclusion, oftentimes in unsupervised learning, one is able to explicitly describe the desired representation distribution or the output distribution, but not the exact correspondence between inputs and output labels. The training and generalization problems arising from this are analogous to those arising from overfitting in the supervised case, and may be mitigated through use of an auxiliary self-supervised loss.

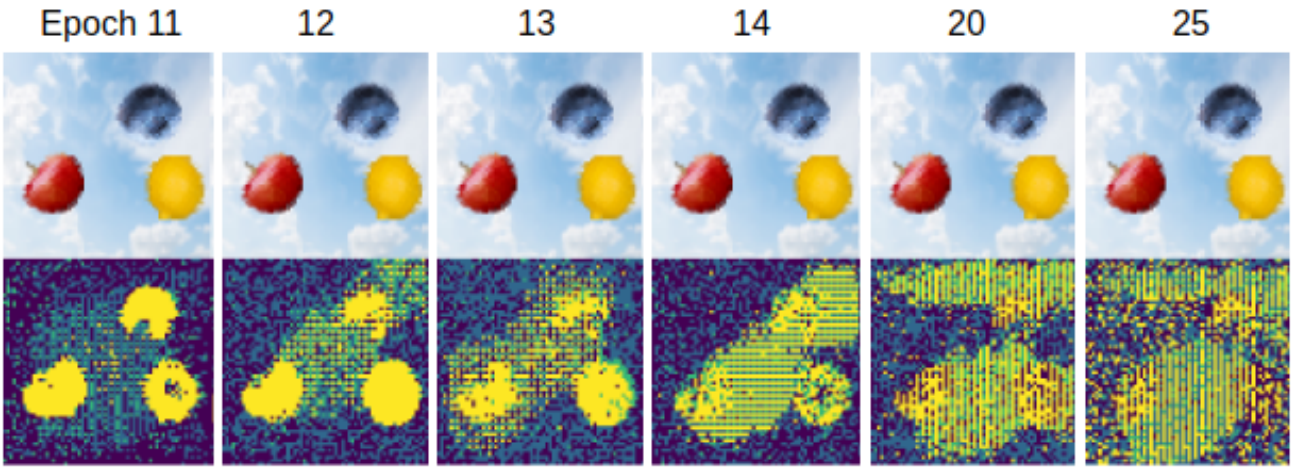


Figure 8: Segmentation output of CIRS model as representation overfitting progresses.

7.3. Reducing representation distribution overfitting in CIRS model

As one can see from the results of the previous section in Figure 7, it is possible to achieve a better than chance result without considering representation overfitting or instituting an auxiliary (reconstruction) objective. We explore the interaction between a reconstruction objective and the CIRS model in section 9. Below, we present another technique that can reduce representation overfitting which we call imbalanced mini-batch training.

In the CIRS neural network model, there are two subnetworks, S and g. The purpose of S is to compute a segmentation, but it relies on g for a loss signal which evaluates how good the segmentation is. The purpose of g is to evaluate the segmentation, but it relies on the output distribution of S to to perform this evaluation. If one already had a feature representation function g which had the property of producing feature vectors of high variance between different ground-truth objects, then the training would be trivial. However, because the problem is unsupervised, we must simultaneously learn the feature representation g and use it to improve the segmentation.

The consequence is that training trajectories are biased by initial conditions and small changes: if g does not initially cluster the representations well, then S will adjust the segmentation to improve it according to a slightly bad representation. This process may cycle and the segmentation will then reflect an amplification of an initially small perturbation. This phenomenon is shown more explicitly in the training progression in Figure 8, where we train S for 100 mini-batches for every 5 batches that we train g (so that any small change in g is immediately reflected in S).

As one can see in the figure above, epoch 11 shows a segmentation where most foreground object pixels belong to the same segment in yellow. However, there is a bit of noise; there are yellow segment pixels in the middle of the image (not part of any foreground object). Since we train S on 100 mini-batches in a row for every 3 g mini-batches, one can think of this as overfitting S to the current state of g. For instance, it is possible that the segmentation shown above in epoch 11 has only slightly better loss than a segmentation with no yellow pixels not in the center. When g is next trained, it will

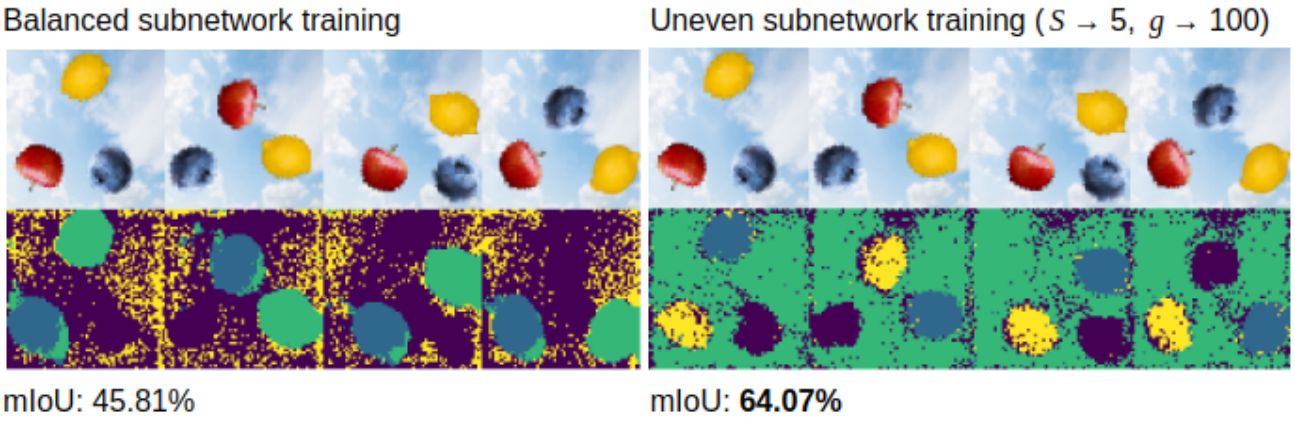


Figure 9: Segmentation result from original CIRS model (left). Segmentation result from CIRS model with imbalanced mini-batch turn-based training procedure (right).

update the loss to accommodate these noise pixels. As the epochs progress, one can see that this initial imperfection in g is amplified.

7.4. Imbalanced Training Procedure Experiment

In an attempt to preserve the g subnetwork's usefulness as a feature representation and prevent subnetwork drift, we apply the reverse training procedure of the above example. We train the S and g subnetworks in turns, but train S for 5 mini-batches for every 100 mini-batches of g.

7.4.1. Results

This procedure achieves a mIoU of 64.07%. A comparison of the CIRS model with uneven training and without are shown in Figure 9.

Table 2: Comparison of CIRS training procedures.

CIRS Model	CIRS Model with imbalanced training
45.81%	**64.07%**

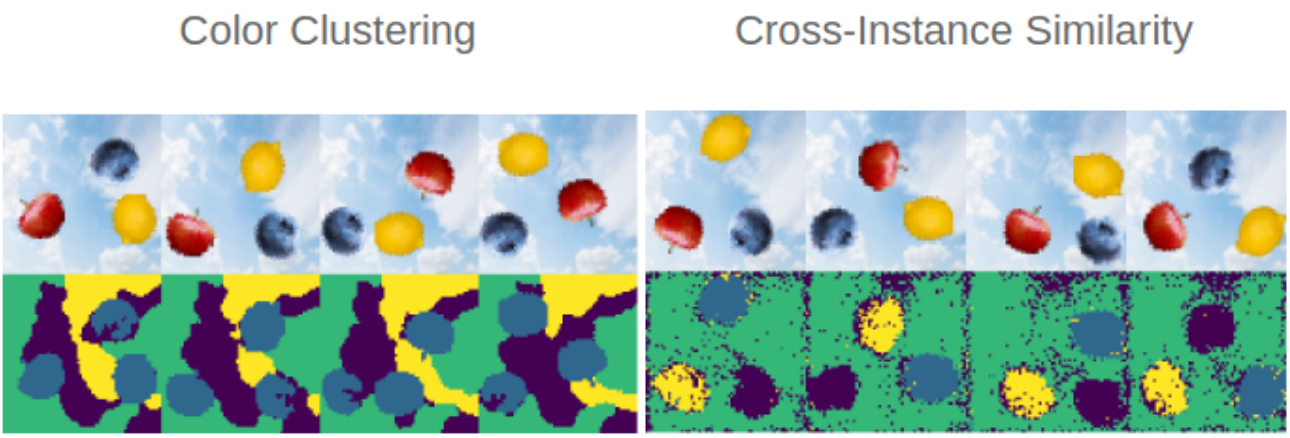


Figure 10: Left: segmentation result from the W-Net architecture (without post-processing). Right: segmentation result from the CIRS model with imbalanced mini-batch turn-based training.

8. Color Clustering versus Cross-Instance Representation Similarity Comparison Experiment

The color clustering approach privileges the contiguity of color within segments of an image. Given a large object with multiple different colors, one would expect this method to produce multiple segments, one for each color range. In contrast, the cross-instance representation similarity approach focuses on finding similar features between images. Given this same object over multiple instances, one would expect this method to produce a single segment for this object. To illustrate the differences of the two approaches, we train a model representing each and compare the results.

Much like the one-dimensional example where the "1122" object was divided, we show that with the color clustering approach, the sky object is divided into 2 segments due to having high internal color variance, despite the object appearing in every image in the same position and orientation. In Figure 10, we show several example segmentation predictions from the validation set for each model.

8.1. Results

We train both the W-Net model (color clustering) and the cross-instance representation similarity model for 170 epochs. We calculate the mean intersection over union (mIoU) on the validation set after every epoch, and save the model if the mIoU is maximal. We set the number of segmentation channels to be $K = 4$ (one for each object as described above).

Table 3: mIoU comparison of W-Net versus proposed model

Color Clustering	Cross-Instance Similarity
31.64%	**64.07%**

The significant difference in mIoU illustrates the advantage of the cross-instance representation similarity approach. One can see from the examples shown that this method learns that the background is a cohesive object because its features are highly correlated across the dataset images: the white and the blue always appear together, so they are a single object. In contrast, the color clustering approach views the sky as multiple objects because of the high interval color variance.

From this experiment, we notice that despite acting on an intermediate representation, W-Net's soft normalized cut objective does not suffer from the problem of representation overfitting. We theorize that this may be due to the auxiliary reconstruction objective. In the next section, we explore how a model that uses reconstruction loss could effectively perform the CIRS loss across images while avoiding representation overfitting.

9. Cross-Instance Properties of the Bottleneck

Instead of explicitly modeling the cross-instance representation similarity objective in the form of a loss, we theorized that one could design an autoencoder model to implicitly encourage the CIRS objective as a consequence of the reconstruction loss. Despite the reconstruction loss being computed on a per-image basis, the presence of a bottleneck layer causes the model to learn cross instance representations with the desired CIRS property. Since the model must represent each image with fewer bits than are input, the bottleneck effectively creates a compression objective. Features specific to individual images must be represented in the encoder output. Since the bottleneck representation capacity is limited, features common across multiple images must be omitted, and instead represented in the weights of the decoder. To achieve a low reconstruction loss on the entire dataset, the bottleneck must therefore learn to discard features common across all instances while preserving those features specific to each instance. In this way, the reconstruction loss of a bottleneck autoencoder effectively operates across images despite explicitly acting on each image individually. The following thought experiment is useful to understand this.

In this scenario, we view a trained autoencoder model as a dictionary: given an image, the encoder "looks up" the code for the image. The decoder takes the code and looks up the corresponding image. Now, imagine one trains it on a dataset of only 2 distinct images: a and b. The encoder need only output a single bit for the decoder to determine which image to produce. In this fashion, one can think of the decoder as "memorizing" the images in its weights, conditioned on the code supplied.

If the bottleneck is exactly 1 bit, then the above is in fact the only way to minimize the loss. The decoder has therefore discovered all variation in the dataset. To put this in terms used for the CIRS loss, we imagine that the a images produce representation 0 and the b images produce representation 1. The variance in representation within each class has been minimized (it is in fact exactly zero) and the variance in representation between classes is a maximum given the 1 bit representation space.

However, if the bottleneck is made larger, then this decoder "memorization" of common features does not have to occur to the same degree; the model can encode some non-instance specific information in the bottleneck representation. This can result in solutions that do not operate across images or optimize the CIRS objective. For instance, if the bottleneck is made to be equal in dimensionality to the input image, then both the encoder and decoder can simply form an identity mapping.

This is why the W-Net model does not operate across images: although W-Net is an autoencoder, it does not have a bottleneck. If the designated number of segments K is greater than 3; the input is of size (3xWxH), while the segmentation (middle) layer is of size (KxWxH). Therefore the "bottleneck" representation will have more bits than the input or output of the model.

9.1. Object Clustering Experiment

The bottleneck architecture causes the model to learn useful representations. Despite having no explicit loss function to do so, an autoencoder model with a sufficient bottleneck implicitly minimizes the object representation similarity objective. That is, in order to achieve low reconstruction loss, autoencoder models effectively maximize the variance of object representations between dissimilar inputs across the dataset. We show that this is the case with a simple clustering experiment. Instead of viewing segmentation as an extension of clustering, we view clustering as a special case of segmentation, where each image contains exactly 1 segment.

We train an autoencoder with 8 fully connected layers (4 encoder layers, 4 decoder layers) on the CIFAR10 dataset. The CIFAR10 dataset consists of 50,000 training images of size 32 by 32 pixels, in color. It contains 10 object classes like "airplane", "automobile", "bird", "cat", etc, with each image containing a single object.

We use dropout with probability $p = 0.2$ and train for 50 epochs. After each epoch, we calculate the variance of the mean feature representation between image classes on the 10,000 image test set. Note that the image labels are solely used for calculation of the metric, and not for training the model. We set the bottleneck layer to have b hidden units, meaning each (3x32x32) 3072 dimensional image is encoded to b dimensions. We examine the relationship between the number of hidden units and the lowest value of the α_2 term of the CIRS loss. The α_1 term is not included because it has no correlation between different training runs (because the global variance across all representations is different across runs).

9.1.1. RESULTS

Despite the bottleneck model being only trained with a reconstructive objective, one can see that it implicitly maximizes the variance between the mean feature representations each class.

Table 4: Effect of number of bottleneck units on CIRS metric in autoencoder model.

Number of bottleneck units b	192	96	48	12
α_2 term of CIRS	6.5e-4	6.6e-3	0.21	1.39

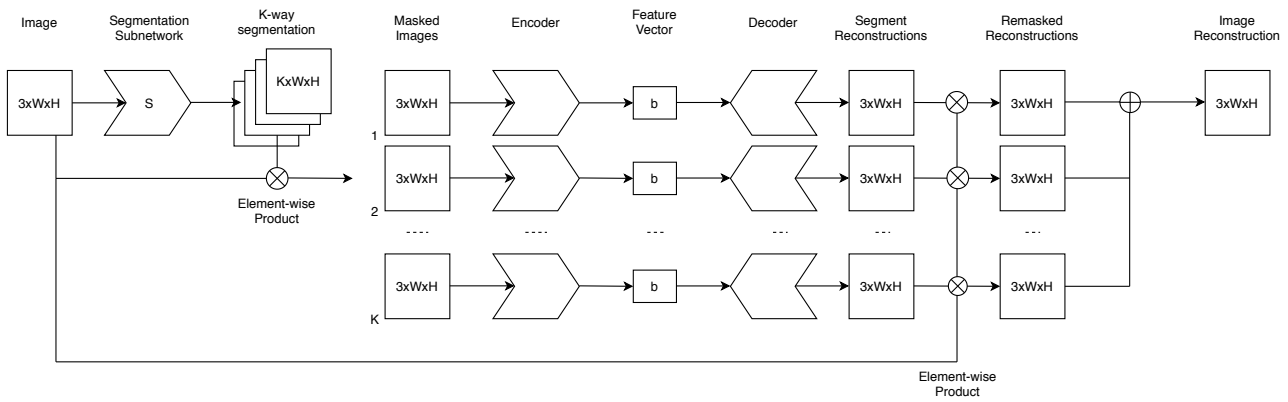


Figure 11: Diagram of the Argmax Individual Reconstruction model.

10. Argmax Individual Reconstruction (AIR) Model

In the previous section, we argued that the presence of a bottleneck causes an autoencoder to search for a representation that discards features common across instances while preserving instance specific details. To extend this to the segmentation case, we develop a new architecture similar to the CIRS model, but take inspiration from split-brain autoencoders [37]. We employ the same segmentation subnetwork S from the CIRS model with the argmax-in-place gradient approximation, and we mask the original image with each segment map. In contrast, after this, the masked object images are fed individually forward through a bottleneck autoencoder specific to that object. This means that if we have $K = 4$ segments allowed, we will have 4 separate autoencoders. The individual reconstructions are then combined and L2 reconstruction loss is computed between the combined reconstruction and the origin input image.

By having a single autoencoder module for each segment, we tie the number of model parameters to the number of segments K, which is somewhat uncommon. To keep the number of total model parameters the same in models with varying K, one can simply decrease the number of parameters/ size of the bottleneck representation in each segment autoencoder as the number of segments K increases. For example, if one had a model with $K = 10$ and with each segment autoencoder having bottleneck dimensionality 20, in the comparable model with $K = 20$, each segment autoencoder would be given bottleneck dimensionality 10. The same must be done with the number of weights in each layer of the encoder and decoder.

The presence of a separate autoencoder individual to each object is key. The subnetwork S must learn to segment such that each individual autoencoder g_k can reconstruct as well as possible. Since each g_k has a bottleneck, this means that S must learn to partition pixels such that instances of each segment have as many common features as possible (α_1 term of CIRS loss). In addition to the individual segment autoencoders, the whole model functions as an autoencoder as well, where the segmentation layer is itself the bottleneck. This is an effective bottleneck because of the argmax-in-place operation. Since there is a limited total bottleneck capacity, S must produce segments that are also as different from the

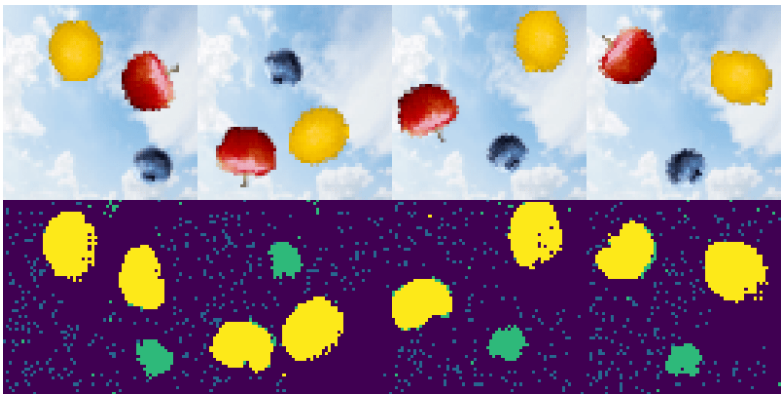


Figure 12: Segmentation output of the Argmax Individual Reconstruction Model.

other segments as possible (α_2 term of CIRS loss).

The model as a whole implicitly optimizes the CIRS objective. Because the representation learned by each individual segment autoencoder must be useful for reconstruction, the idea is that representation overfitting could be diminished relative to the model with explicit CIRS loss. A diagram of the model is shown in Figure 11.

The Argmax Individual Reconstruction (AIR) model extends the latter portion of the CIRS model into an autoencoder architecture, with an added reconstruction loss objective. Apart from this, both models are the same. Subnetwork S predicts K softmax probabilities for each pixel, of which the argument max (argmax) is taken. This is used with the original image to compute a masked image for every predicted segment, containing only those pixels belonging to the segment. The masked object images are then passed forward through subnetwork g and feature representations are output.

In AIR, the subnetwork g functions as the first half of an autoencoder. The output of g can be interpreted as the "bottleneck" of the autoencoder. We showed in the previous section that bottleneck autoencoders implicitly optimize the α_2 part of the CIRS loss depending on the severity of the bottleneck.

Instead of explicitly computing the CIRS loss objective here, we pass the output features for each predicted segment to the decoders to reconstruct the segments individually. The output segment reconstructions are then remasked by the corresponding predicted segmentation mask. Finally, the sum is taken across the K separate segment reconstructions from the same image, and reconstruction loss is computed. The remasking step is necessary to ensure that the latent feature representation of each segment is only affected by that segment. Note that we do not backpropagate gradients from this multiplication back to segmentation mask.

10.1. Results

We train the model for 170 epochs, and as with the previous procedure, we save the model with the best performance and evaluate on the test set. We show the resulting segmentation output in Figure 12. The model achieves a mIoU of 51.55%. While this model did not beat the performance of the previously described CIRS model, we felt it important to mention. We speculate that, as a generative model, the AIR model may be more difficult to train in practice, but may be less prone to representation overfitting.

Table 5: AIR Model Performance.

Argmax Individual Reconstruction (AIR) Model
51.55%

11. Conclusion and Future Work

In this paper, we investigated a possible weakness in the philosophy of current approaches to the problem of unsupervised semantic segmentation. We compared two different philosophies about the nature of objects and how they influence segmentation methods. In the first, one views the contiguity of color to be paramount to object identity. We identified methods privileging this view "color clustering" based methods. We showed that this approach fails in cases where objects have high internal color variance.

In the second view, an object is defined in relation to how similar or different it is from all other objects. We called methods inspired by this viewpoint "cross-instance representation similarity" based methods. We showed with a simple one-dimensional example how this approach succeeds where the other approach fails in clustering objects with high internal color variance.

We created a synthetic dataset designed to illustrate the effects of different model approaches. We adapted the one-dimensional objective to work in more complex environments, calling it the "Cross-Instance Representation Similarity" loss objective. We proposed a novel neural network model architecture based on this CIRS loss, and showed that it successfully segments objects in the synthetic dataset despite high internal color variance.

There are several key ideas in this project that we think will be useful for designing unsupervised semantic segmentation models in the future. The first is the principle that loss objectives for segmentation models should effectively operate across images. We illustrated this with the CIRS loss model, which is computed across all images in the mini-batch. However, the designation of "effectively" is important here, because some objectives that can be computed on a single image can still learn a principled representation across images. We illustrated this in our second proposed model, the argmax-in-place individual reconstruction (AIR) model. Although the L2 reconstruction loss can be computed on a single input, we showed that the bottlenecks cause the model to learn useful object representations across the whole dataset.

We identified a phenomenon present in unsupervised models which, to our knowledge, has not been previously described in the literature, and present mitigation strategies. We called this problem "representation overfitting," because it is analogous to conventional "label overfitting" in the supervised case. In general, in deep neural network models, one specifies one or more optimization objectives on the output representations of one or more layers. In other words, we identify some goal for the characteristics of the output distribution. In the supervised setting, we know what the output distribution should look like, and we know the exact correspondence between each input and its output representation, because it is given explicitly by the labels.

In unsupervised deep learning, however, we know what the output distribution should look like, but we do not know the correspondence between the inputs and the outputs. The goal is to design the architecture and training procedure such that this correspondence has a high likelihood to match human intuitive (ground-truth) without using a signal from annotations. We showed that a correspondence between inputs and outputs could be encouraged through the use of an

auxiliary self-supervised loss objective such as reconstruction loss.

We showed that this problem could also be mitigated in cases where one would like to design neural network modules with discrete purposes. For the CIRS model, we wanted to have 2 modules: a segmentation subnetwork, and a feature representation subnetwork. However, training the network end-to-end caused the distinction between these two purposes to be lost; the intuitive functions of the networks become intertwined as training progresses. We showed that using a turn-based, imbalanced training procedure can reduce the representation distribution overfitting effect and result in a more intuitive outcome.

We believe that the argmax-in-place approximation, as presented in the CIRS model could be useful to future semantic segmentation research. Conventionally, the softmax function is applied to the output of the segmentation model to produce a vector of class probabilities for each output pixel. To compute a loss on this output, one must find a soft approximation for the function in mind. In many supervised models, either the pixel-wise cross entropy loss is used between the ground truth and the predicted segmentation, or a soft approximation of the mIoU is done. The argmax-in-place gradient approximation could be used in these cases and others where soft approximations are difficult.

We foresee several ways to build off of this project. One possible extension of the AIR model would be to add the ability to generate novel images or segments. This could be done by converting the autoencoder model to a variational autoencoder model by learning the parameters of the bottleneck representation in the individual reconstruction subnetwork. In this fashion, one could use an existing image segmentation, but sample from the distribution for each segment to generate an image with similar semantics but different style. This is done in a supervised setting in Nvidia's SPADE model [25].

Another possible extension of this project would be to find a suitable substitute for the feature representation subnetwork g. The training procedure for the CIRS model would be much improved if one did not have to simultaneously train both subnetworks. By first training g on some self-supervised task and freezing its weights, one eliminates the possibility of representation overfitting entirely. However, in our initial attempts with self-supervised training procedures, the representations learned only enabled the distinction between larger regions of the image.

Another possible extension is to scale up the reconstruction architecture of the AIR model. It is possible that improving the generational capabilities of the AIR model could result in a better segmentation. We found that the AIR model was more difficult to train, and theorize that this may be because when the segmentation subnetwork updates, the input distributions to the individual reconstruction subnetworks change sharply. It is possible that improvements to the architecture of the reconstruction subnetwork could result in better performance.

Unsupervised semantic segmentation is a very difficult problem, with current state of the art approaches reaching under 30% per-pixel accuracy on complex, real world datasets like COCO [16]. A viable algorithm for this problem on complex, real world images would allow for the usage of huge amounts of unlabeled data at scale, and would represent a milestone in machine understanding of images.

References

[1] Adams, Rolf and Bischof, Leanne. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994.

[2] Alexe, Bogdan, Deselaers, Thomas, and Ferrari, Vittorio. Classcut for unsupervised class segmentation. In *European conference on computer vision*, pp. 380–393. Springer, 2010.

[3] Canny, John. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[4] Caron, Mathilde, Bojanowski, Piotr, Joulin, Armand, and Douze, Matthijs. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018. URL http://arxiv.org/abs/1807.05520.

[5] Chen, Feiyang, Chen, Nan, Mao, Hanyang, and Hu, Hanlin. The application of bipartite matching in assignment problem. *CoRR*, abs/1902.00256, 2019. URL http://arxiv.org/abs/1902.00256.

[6] Chow, CK and Kaneko, T. Automatic boundary detection of the left ventricle from cineangiograms. *Computers and biomedical research*, 5(4):388–410, 1972.

[7] Cordts, Marius, Omran, Mohamed, Ramos, Sebastian, Rehfeld, Timo, Enzweiler, Markus, Benenson, Rodrigo, Franke, Uwe, Roth, Stefan, and Schiele, Bernt. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] Deselaers, Thomas, Alexe, Bogdan, and Ferrari, Vittorio. Localizing objects while learning their appearance. In *European conference on computer vision*, pp. 452–466. Springer, 2010.

[9] Felzenszwalb, Pedro F and Huttenlocher, Daniel P. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[10] Felzenszwalb, Pedro F and Huttenlocher, Daniel P. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[11] Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[12] Haralick, Robert M and Shapiro, Linda G. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[13] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[14] He, Kaiming, Gkioxari, Georgia, Dollár, Piotr, and Girshick, Ross. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[15] Hoffman, Judy, Tzeng, Eric, Park, Taesung, Zhu, Jun-Yan, Isola, Phillip, Saenko, Kate, Efros, Alexei A, and Darrell, Trevor. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[16] Ji, Xu, Henriques, João F, and Vedaldi, Andrea. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9865–9874, 2019.

[17] Jing, Longlong, Chen, Yucheng, and Tian, Yingli. Coarse-to-fine semantic segmentation from image-level labels. *IEEE Transactions on Image Processing*, 29:225–236, 2019.

[18] Joyce, Thomas, Chartsias, Agisilaos, and Tsaftaris, Sotirios A. Deep multi-class segmentation without ground-truth labels. 2018.

[19] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL http://doi.acm.org/10.1145/3065386.

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

[21] Luc, Pauline, Couprie, Camille, Chintala, Soumith, and Verbeek, Jakob. Semantic segmentation using adversarial networks. *ArXiv*, abs/1611.08408, 2016.

[22] Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, Goodfellow, Ian, and Frey, Brendan. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[23] Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pp. 416–423, July 2001.

[24] Pappas, Thrasyvoulos N and Jayant, Nikil S. An adaptive clustering algorithm for image segmentation. In *International Conference on Acoustics, Speech, and Signal Processing,*, pp. 1667–1670. IEEE, 1989.

[25] Park, Taesung, Liu, Ming-Yu, Wang, Ting-Chun, and Zhu, Jun-Yan. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2337–2346, 2019.

[26] Pinheiro, Pedro O and Collobert, Ronan. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1713–1721, 2015.

[27] Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[28] Redmon, Joseph, Divvala, Santosh, Girshick, Ross, and Farhadi, Ali. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[29] Ren, Shaoqing, He, Kaiming, Girshick, Ross, and Sun, Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

[30] Ronneberger, Olaf, Philipp Fischer and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation, 2015.

[31] Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition, 2014.

[32] Tzeng, Eric, Hoffman, Judy, Saenko, Kate, and Darrell, Trevor. Adversarial discriminative domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[33] Wei, Yunchao, Feng, Jiashi, Liang, Xiaodan, Cheng, Ming-Ming, Zhao, Yao, and Yan, Shuicheng. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1568–1576, 2017.

[34] Xia, Xide and Kulis, Brian. W-net: A deep model for fully unsupervised image segmentation. *arXiv preprint arXiv:1711.08506*, 2017.

[35] Xiaohan, Yu, Yla-Jaaski, Juha, Huttunen, Olli, Vehkomaki, Tuorno, Sipila, Outi, and Katila, Toivo. Image segmentation combining region growing and edge detection. In *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis,*, pp. 481–484. IEEE, 1992.

[36] Zhan, Xiaohang, Liu, Ziwei, Luo, Ping, Tang, Xiaoou, and Loy, Chen Change. Mix-and-match tuning for self-supervised semantic segmentation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[37] Zhang, Richard, Isola, Phillip, and Efros, Alexei A. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *CoRR*, abs/1611.09842, 2016. URL http://arxiv.org/abs/1611.09842.

[38] Zhou, Bolei, Khosla, Aditya, Lapedriza, Agata, Oliva, Aude, and Torralba, Antonio. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

[39] Zhu, Jun-Yan, Park, Taesung, Isola, Phillip, and Efros, Alexei A. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. URL http://arxiv.org/abs/1703.10593.

12. Appendices

12.1. One-dimensional Cross-Instance Representation Similarity Segmentation Procedure

1. Generate a set of all possible sizes of connected/adjacent cells, starting from size 1, and going up to the maximum:

   Size 1: “1”, “2”, “0”

   Size 2: “11”, “12”, “22”, “20”, “00”, “01”

   Size 3: “112”, “122”, “220”, “200”, “011”

   Size 4: “1122”, “1220”, “2200”, “0112”

   Size 5: “11220”, “12200”, “01122”

2. Count the frequency of the patterns above. If a pattern appears at least once in an image, we add 1 to the count.

   Size 1: “1”:3, “2”:3, “0”:3

   Size 2: “11”:3, “12”:3, “22”:3, “20”:2, “00”:2, “01”:2

   Size 3: “112”:3, “122”:3, “220”:2, “200”:1, “011”:2

   Size 4: “1122”:3, “1220”:2, “2200”:1, “0112”:1

   Size 5: “11220”:2, “12200”:2, “01122”:2

3. Select the most frequent pattern with the largest size. In the example, many patterns are tied for the most frequency (3) but only one is the largest: “1122.”

4. Match the selected pattern once in every image, replacing matched characters with the background symbol

5. (necessary in datasets with more than 1 object) Repeat steps 1-4 on the new replaced dataset (created in step 4)

12.2. Training/ Hyperparameter Details for CIRS, AIR models

Hyperparameter	Value
Epochs	170
Batch Size	40
Dropout %	0.4
Optimizer	Adam
Batch Normalization	True
Learning Rate	1e-3