ZERO-SHOT SEMANTIC SEGMENTATION

AUTHOR: ROHAN DOSHI
ADVISOR: OLGA RUSSAKOVSKY

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE IN ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE
PRINCETON UNIVERSITY

May 7th, 2018

This thesis represents my own work in
accordance with university regulations.

On the shoulders of giants: I cannot thank Professor Russakovsky, Professor Feamster, and Professor Fellbaum (right to left, excluding myself) enough for their patience over the years in teaching me the fundamentals of conducting research.

Acknowledgements

This thesis was a team effort. Without the constant support of my academic partners, close friends, and family, I would not have the knowledge or capacity to have completed this work.

I have to start by thanking Professor Olga Russakovsky. You inspired my love of computer vision and have been the most flexible, brilliant, and thoughtful advisor I could have ever asked for. And, it's been a privilege working for you in the Princeton Visual AI Lab, alongside William Hinthorn, Prem Nair, and Kenji Hata, whose constant ideas have been critical in reasoning about this work.

I also have to thank Professor Feamster and Professor Fellbaum for guiding my independent research projects from my junior year. You two taught me the fundamentals of research which I rely on constantly. I really cannot thank you enough you for your patience and wisdom.

Additionally, my thesis would not have been possible without the constant support of my friends - all of whom have seen me at my best and worst, and, for reasons I do not understand, choose to stick

by my side. For example, I'd have to thank David Mitchell, Evan Wood, Stefan Keselj, Avinash Nayak, Vishan Nigam, Soham Daga, Austin Addison, just to name a few in particular.

I have have to thank my two younger siblings, who are my closest friends in life. Shivani, you're smart, wise, and caring beyond your years - you never fail to amaze me each day. Rajat, at times I feel like we are the same person - I have zero doubt you'll achieve great things and make me proud.

Lastly, and most importantly, I have to thank my parents. Mom and Dad, I know you've sacrificed an ungodly amount when you came to this country to build a life so that Shivani, Rajat, and I can enjoy the best of what the world has to offer. I've seen you sacrifice your happiness time and time again for the sake of our family. No matter how exhausted or stressed you were, you always found time to drag us to tennis practices, piano classes, and art museums, just with the faint hope of igniting our love for learning in the process. Without you, I'm nothing.

Contents

Abstract

Visual recognition systems struggle to identify uncommon classes not captured in real-world image datasets because annotation labels are limited to finite sets of common objects. This motivates the need to develop vision systems that can infer the identity of previously unseen objects from an open vocabulary. To that end, we propose a new task, zero-shot semantic segmentation, in which each pixel is assigned a label from a either a set of seen classes (observed in the training dataset) or unseen classes (not observed in the training dataset). To attempt this task, we propose the Seenmask Zero-shot Network (SZN), a deep learning model that aligns visual and semantic embedding encodings for pixel-label pairs in a joint-embedding space. By projecting pixels from unseen object classes into this joint-embedding space and selecting nearby semantic embeddings, we can infer new labels. We also propose a seenmask mechanism that allows the model to determine whether a given pixel's label belongs among the seen or unseen labels for a more intelligent inference approach. Our results demonstrates the importance of having high class density for properly aligning visual and semantic embeddings in a joint-embedding space. Our analysis also reveals the theoretical upper bound on accuracy if the seenmask mechanism were perfect, revealing the need for more work on this approach.

1. Introduction

The world is populated with many objects which we may perceive as ambiguous or unidentifiable. Visual recognition systems struggle to identify these previously unobserved objects. Often, they train on existing real-world image datasets with labels from fixed sets of common classes. These image label sets are limited because it is expensive and impractical to label images from an open vocabulary. Thus, while common objects are captured in real-world image datasets, most types of objects, which lie in the long tail of the object frequency distribution, often are poorly (if at all) captured and represented in computer vision datasets. This motivates the need to develop vision systems with the ability to infer the identity of previously unseen objects.

While visual recognition systems struggle with unseen objects, the human brain is able to predict the identity of unseen objects by transferring its understanding of similar objects and concepts from other domains. Many state-of-the-art systems have attempted to emulate this approach for the zero-shot learning task. This task is defined as the prediction of image labels for both *seen* object classes from training data and *unseen* object classes not yet observed. In the literature, we find that vision systems have managed to generate image-level predictions for zero-shot learning with some limited success by transferring learning from the language domain.

Yet, little to no work has been focused on extending zero-shot learning to the pixel-level scale, a novel task which we define as *zero-shot semantic segmentation*. Pixelwise labels allow for the identification and disambiguation of specific objects or parts of objects with high locational granularity, something not captured with image-level labels.

Past zero-shot learning work has proposed aligning images and labels in a joint-embedding space. With this approach, visual and semantic models are trained to map image-label pairs to the same point in the joint-embedding space. The intuition is that these models will learn a generalizable mapping which can be used to project unseen images to new labels in the joint-embedding space. One of this work's insights is to extend the visual models used in the joint-embedding approach from images to pixels; this enables pixel-level inference of unseen classes. To that end, our work

also raises the question of how to develop a visual model that outputs pixel-level embeddings with sufficient local and global image information.

Another open question is to what degree will the visual model's mapping of seen pixel classes to labels in the joint-embedding space generalize to unseen pixel classes. The weaker this mapping, the more likely it is that a visual model will simply map all pixel classes to joint-embedding space regions belonging to seen classes. One of ideas to prevent this is to have the model first decide whether a given pixel class has been seen before in the training data. If so, the model will only infer among seen labels, and otherwise only among unseen labels. By segmenting the label-space, we can enable a smarter inference approach that is more likely to infer unseen labels. To this end, our second key insight is to generate a pixelwise "seenmask."

We leverage these two critical insight to formulate the Seenmask Zero-shot Network (SZN). We reformulate the zero-shot joint-embedding space approach by using the visual model to map pixels, not images, to labels. And, we propose a seenmask mechanism to produce pixelwise predictions of whether a pixel belongs to a previously seen class. This will enable a more intelligent inference approach, one that facilitate the mapping of unseen classes to unseen labels in the joint-embedding space.

To be explicit, our two key novel contributions to the literature are as follows:

1. We define a novel task: zero-shot semantic segmentation.
2. We propose the Seenmask Zero-shot Network (SZN) as a first attempt at this task.

We hope the introduction of a our new tasks will drive the computer vision search community to push the capabilities and accuracy of visual recognition systems. We also hope this task will spur the development of new datasets and new networks to attempt the proposed task, in which case, the SZN will serve as a baseline for comparison.

2. Background and Related Work

2.1. Semantic Segmentation

One of the most popular tasks in computer vision is semantic segmentation, the labeling of each image-pixel with one class. Fully Convolutional Networks (FCNs) have been particular successful at this task. For example, [6] adapted a 1000-way deep convolutional network trained on the ImageNet Large Scale Visual Recognition Competition (ILSVRC) dataset, replaced the final fully connected classification layers with fully convolutional layers, and fine-tuned on PASCAL VOC to achieve state-of-the-art semantic segmentation results. The key insight with this work is to combine down-sampled and up-sampled layers in the network via skip layers. This approach helps both locate and identify each pixel correctly by combining local and global information. One of our insights on this work is that by upsampling intermediate dense feature maps via transpose convolutions, a FCN can be used to generate pixelwise visual embeddings.

Another key advantage of FCNs over regular convolutional networks is faster inference. If a convolutional network has any fully connected layers, input images must be split into overlapping regions matching the input size of the network and processed individually, requiring multiple forward passes. In contrast, with a fully connected network, the entire input image can be processed in a single forward pass, speeding up inference.

Many real-world image datasets such as PASCAL VOC [2], PASCAL-Context [7], and Microsoft COCO [5] contain semantic segmentation annotations for subsets of their images. However, these datasets all have a fixed-size set of label classes (21, 33, and 91 for Pascal VOC, Pascal-Context, and Microsoft COCO, respectively). While these semantic segmentation annotations have provided a value source of training data for visual recognition networks, scaling these networks to include labels from an open vocabulary for all possible varieties of objects remains unsolved. Thus, existing classification-based semantic segmentation systems fail when evaluating objects with previously unseen labels due to fixed-vocabulary constraints. This motivates the need for applying zero-shot learning approaches to semantic segmentation.

2.2. Zero-Shot Learning

The leading approach to the zero-shot task is to create a joint-embedding space that aligns visual and semantic embeddings for image-label pairs. At a very high level, the intuition is to learn a mapping of visual image embeddings to semantic embeddings that generalizes from observed to unobserved object classes. For example, the DeViSE paper proposes mapping images directly into the semantic embedding space for zero-shot learning [3]:

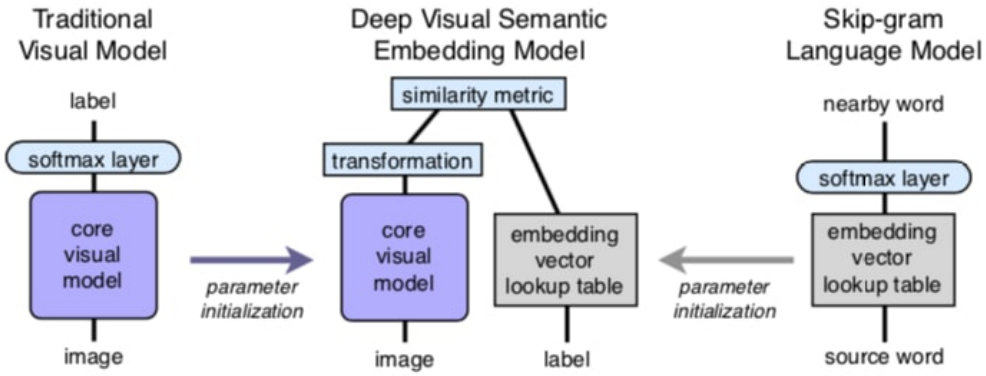


Figure 1: DeViSE Architecture [3]

As depicted in Figure 1, a visual model ("core visual model") and semantic model ("embedding vector lookup table") are pretrained to output vectors (also referred to as embeddings). The visual model can use a convolutional network architecture such as VGG [9] or ResNet [4] as an image-level feature extractor. And, the semantic model can be implemented with a embedding lookup model based on Word2Vec or GloVe. The assumption behind word embeddings is that that words that co-occur with high frequency (e.g. cow and cheese) in text corpses share semantic meaning, so they should be closer together in the semantic embedding space.

During inference, a linear transformation is applied to map a visual embedding into the joint-embedding space. Finally, the nearest semantic embedding, which is already in the joint-embedding space, is chosen as the predicted label. The choice of loss function is critical in this work. It is a

combination of a cosine similarity and a hinge-rank loss:

$$L(\hat{y},\bar{y}) = \sum_{j \neq label} max[0, margin - \bar{y}_{label} W\hat{y} + \bar{y}_{image} W\hat{y}] \quad (1)$$

where $\hat{y}$ is the output of the core visual model, W are the weights of the linear transformation layer, and $\bar{y}$ is the semantic embedding vector.

Many of the same authors from the DeViSE paper collaborated in a follow-on paper, a work often referred to as the CONSE paper [8]. The key insight is to represent images as the convex combination of all the candidate labels' semantic embeddings. This approach yields improvement with little effort since no additional training is required. Additionally, this work further validates the joint-embedding approach to zero-shot learning.

Building upon the DeViSE paper, we propose extending the zero-shot joint-embedding approach from image-level to pixel-level inference by having the visual model output pixel-level visual embeddings as opposed to image-level visual embeddings. To do so, we propose modifying the FCN architectures for this task by modifying the upsampling of intermediate dense feature maps.

Still, there are many challenges in going from image-level to pixel-level zero-shot inference. The first is learning a generalizable mapping between pixels and labels that applies to both seen and unseen objects. If the visual model overfits on seen objects in the training data, it is likely the model will only learn to map pixels to regions in the joint-embedding space corresponding to seen labels. Thus, prediction accuracy for unseen objects will suffer. Our insight is to have the model first learn to predict whether a pixel's label is from a seen or unseen class. Then, inference should only be among the relevant subset of semantic embeddings (seen or unseen). Thus, our model must learn to predict a "seenmask" to enable this new inference methodology.

3. Task Formalization: Zero-shot Semantic Segmentation

We consider an image I, a set of seen labels S (for objects observed in the training data), and a non-intersecting set of unseen labels U (for objects unobserved in the training data). During

inference, we assign each pixel in the image to either a seen on unseen label:

$$I_{i,j} \in S \cup U \tag{2}$$

While U could extend to an open vocabulary (all possible labels) excluding the seen labels, in practice, we find that U is often limited to a smaller subset of relevant labels to remove noise in inference predictions.

4. Seenmask Zero-shot Network (SZN) Proposal

4.1. Inference Overview

At a high level, the network projects pixels into a joint-embedding space and selects nearby seen and unseen labels. A seenmask is used to choose between between predicted seen and unseen labels in a pixelwise manner. More concretely, the network starts by taking a RGB image I as well as a set of seen and unseen labels S and U as input. From here, we break the inference algorithm into three steps: the visual model, the semantic model, and seenmask embedding inference.

4.2. Visual Model

The goal is to produce visual embeddings and a seenmask from I using the two output heads of the FCN (the visual embedding head and the seenmask head):

1. **FCN** We extract a dense feature map by passing I through the FCN and extracting the 32x down-sampled fc7 output $(4096x\lceil\frac{h}{32}\rceil x\lceil\frac{w}{32}\rceil)$.
2. **Visual Embedding Head** We generate visual embeddings by applying 1x1 convolutions to the dense feature map with j output channels, where j is the dimensionality of the joint-embedding space. This is upsampled 32x via transpose convolutions initialized with bilinear interpolation weights to a *jxhxw* dimension output, giving pixelwise visual embeddings in the joint-embedding space.
3. **Seenmask Head** We generate a binary seenmask, where a 1 indicates that a pixel belongs to a

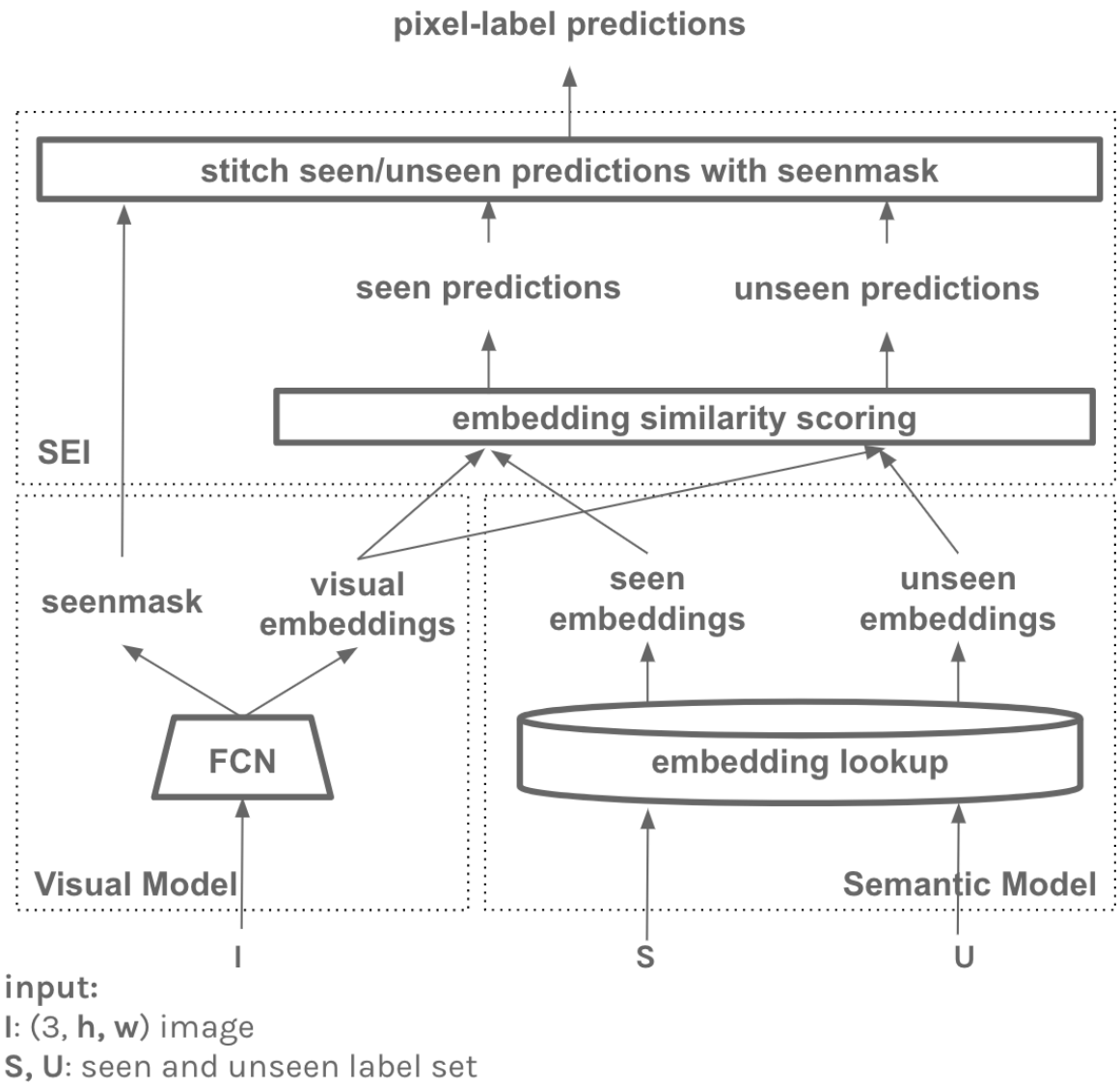


Figure 2: Proposed Seenmask Zero-shot Network (SZN)

seen class and a 0 indicates that a pixel belongs to an unseen class. This is done by applying 1x1 convolutions to the fc7 dense feature map and outputting 2 channels. This is upsampled 32x via transpose convolutions initialized with bilinear interpolation weights to a $2xhxw$ dimension output, giving pixelwise seenmask confidence scores. Finally, a pixelwise argmax is applied for binary predictions of either a 0 or 1.

4.3. Semantic Model

A semantic model maps labels in S and U to semantic embeddings that are in the joint-embedding space. This is done by using an semantic embedding lookup generated during training, which we'll

describe in Section 5.3.

4.4. Seenmask Embedding Inference (SEI)

The goal of this inference methodology is to infer labels at each pixel from visual embeddings and seen/unseen semantic embedding using the seenmask. We use the following 3-step procedure:

1. For each pixel, calculate the cosine similarity between the visual embedding and each seen semantic embedding. Choose the label with the highest similarity score for the seen prediction.
2. For each pixel, calculate the cosine similarity between the visual embedding and each unseen semantic embedding. Choose the label with the highest similarity score for the unseen prediction.
3. We assign each pixel to either the seen prediction if the seenmask is 1 or the unseen prediction if the seenmask is 0. Stitching the seen and unseen predictions with the seenmask produces the final semantic segmentation output of $1xhxw$ dimensions.

5. Implementation

5.1. Training Overview

As depicted in Figure 3, we propose the following two step procedure for training the two-headed SZN network:

1. Train the FCN and visual embedding head
2. Train the seenmask head

5.1.1. Preprocessing: Training Class Splits We preprocess the training dataset and class splits to facilitate the seenmask head. We start by discarding training images with any pixel labels from the unseen label set U. The seenmask head needs examples of unseen classes for binary classification. But, by the zero-shot formulation, our model cannot observe unseen classes from U during training. So, we split the seen label set S into non-overlapping $\text{train}_{\text{seen}}$ and $\text{train}_{\text{unseen}}$ sets.

The FCN and the visual embedding head only train on images in which all image-pixels have $\text{train}_{\text{seen}}$ labels. Thus, the FCN will output dense feature maps unbiased by $\text{train}_{\text{unseen}}$ examples. But, the seenmask head trains on all images (all pixels belong either to $\text{train}_{\text{seen}}$ or $\text{train}_{\text{unseen}}$ labels).

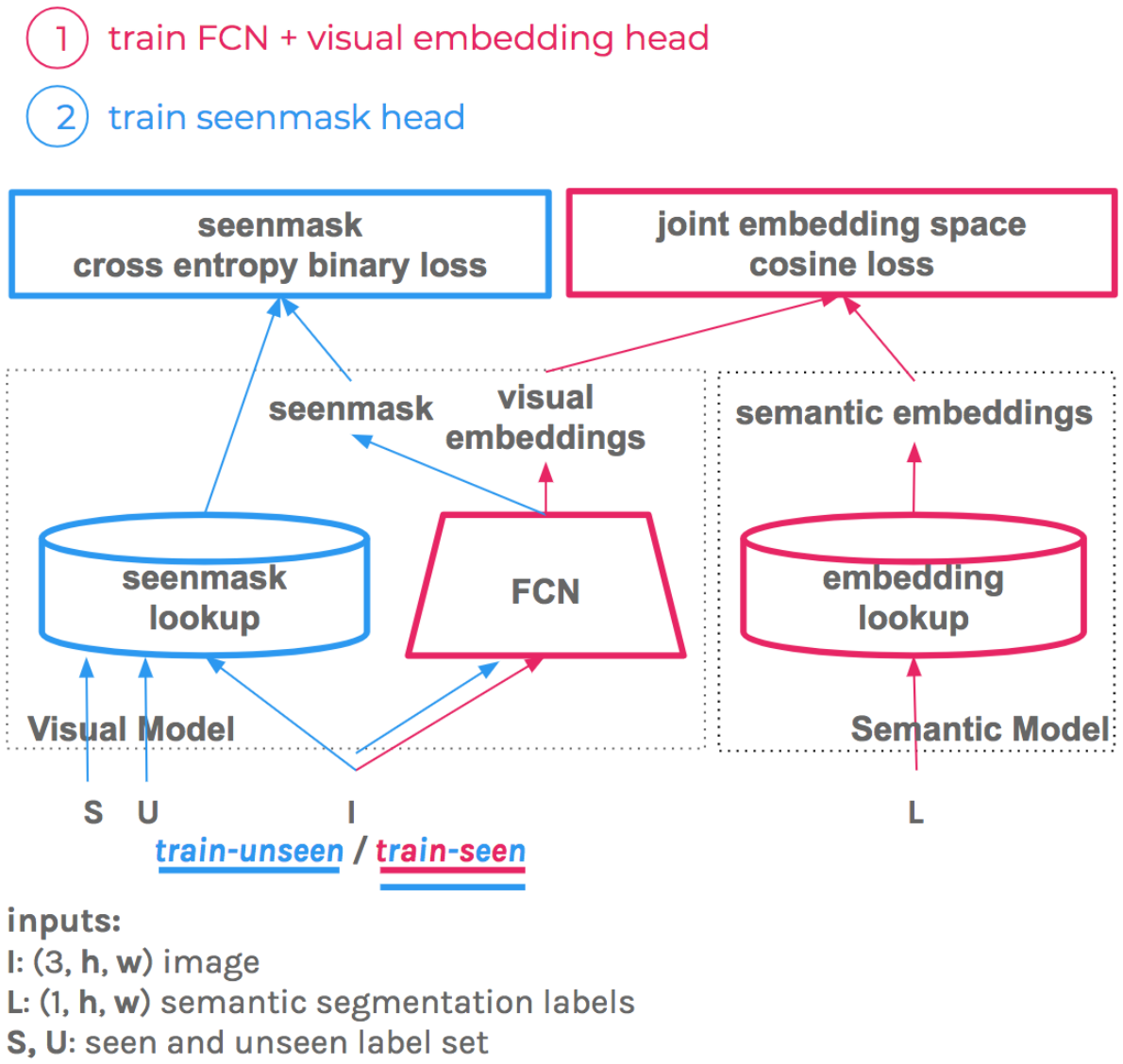


Figure 3: SZN 2-Step Training Methodology

The randomly selected $\text{train}_{\text{unseen}}$ classes are meant to approximate a random sampling of real-world unseen classes U. This formulation allows the seenmask to discern observed classes (e.g. $\text{train}_{\text{seen}}$) from unobserved classes (e.g. $\text{train}_{\text{unseen}}$ and U).

One tricky implementation note is that during test-time inference, the unseen label set U should be appended with $\text{train}_{\text{unseen}}$. Thus, if the seenmask predicts a 0 for a given pixel (a.k.a. unseen), the predicted label should be among the labels in $\text{train}_{\text{unseen}} \cap U$.

5.1.2. Train FCN + Visual Embedding Head We train the FCN and visual embedding head of the SZN network (pink components of the model in Figure 3) on images with all pixel-labels only

in $\text{train}_{\text{seen}}$. We align the visual and semantic embeddings for corresponding pixel-label pairs in the joint-embedding space with a pixelwise cosine loss (see Section 5.4). And, we update the model weights using the Adam optimizer for 30 epochs with a batch size of 1 image and learning rate of 1e-5 for a joint-embedding space dimensionality of $j = 20$.

We later detail the analysis that led to our training methodology choices for the joint-embedding loss function (Section 7.1), optimizer (Section 7.2), and joint-embedding space dimensionality (Section 7.3).

5.1.3. Train Seenmask Head We now freeze the FCN weights and treat the FCN as a dense feature map extractor to be fed into the seenmask head.

Then, we train the seenmask head weights a via pixelwise binary cross-entropy loss between the predicted seenmask $\hat{M}$ and true seenmask $\bar{M}$, where $\hat{M}_{i,j}, \bar{M}_{i,j} \in \{0,1\}$:

$$L(\hat{M}, \bar{M}) = \sum_{i,j \in M} -(\bar{M}_{i,j} log(\hat{M}_{i,j}) + (1 - \bar{M}_{i,j}) log(1 - \hat{M}_{i,j})) \tag{3}$$

We obtain $\bar{M}$ by mapping the semantic segmentation labels in a pixelwise manner to 1 if the label is in $\text{train}_{\text{seen}}$ or 0 if the label is in $\text{train}_{\text{unseen}}$.

5.2. Visual Model

Our FCN architecture is based on VGG-16 (see Figure 4), a deep convolutional classification model [9]. As described in [6], we replace VGG's fully connected layers with convolutional layers to make the network fully convolutional. The downsampling layers (including fc7) are initialized to pretrained weights from a VGG-16 model trained on the ILSVRC 2014 task. The fc7 layer outputs a dense feature map of $4096x\lceil h/32 \rceil x \lceil w/32 \rceil$ dimensions, reflecting a 32x down-sampling of the input volume after 5 layers of convolution and pooling.

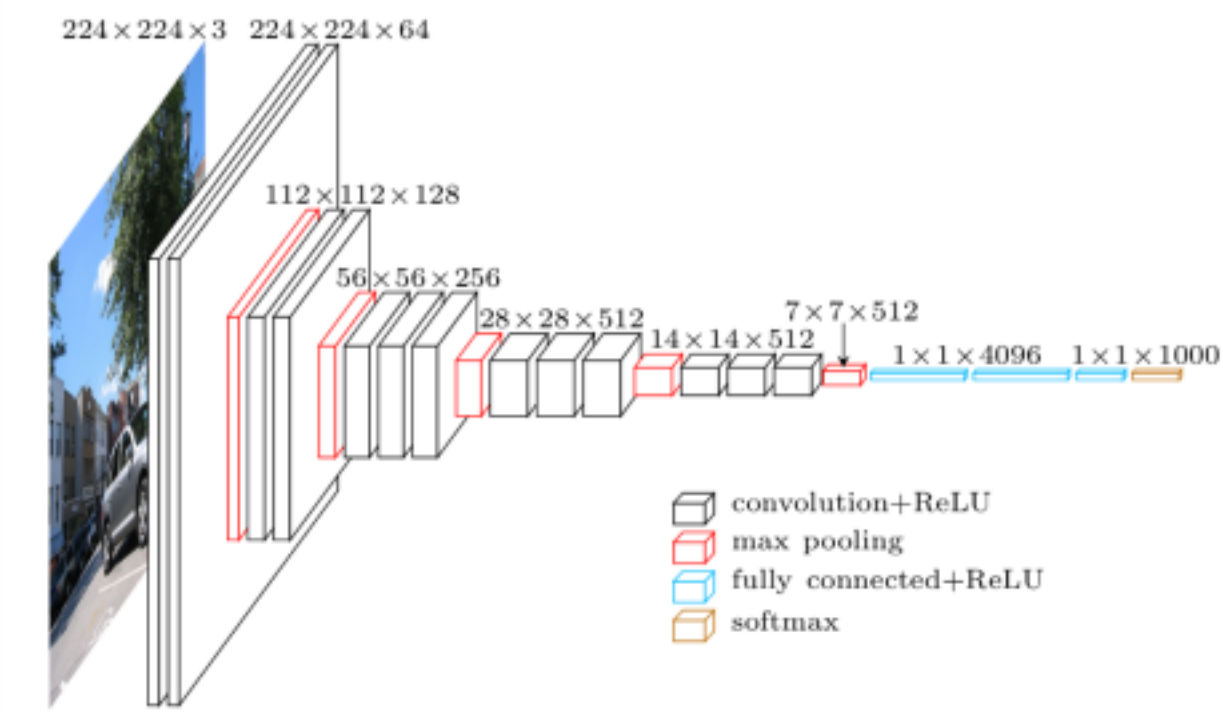


Figure 4: VGG-16 Architecture [9]

Our FCN model upsamples the same dense feature map for two different output heads: 1) the visual embedding head and the 2) seenmask head. The visual embedding and seenmask head learn a linear mapping by applying 1x1 convolutions with j and 2 output channels, respectively. Finally, a transpose convolution upsamples the image 32x back to the height and width of the input medium, with j and 2 output channels respectively. The transpose convolution weights are initialized using bilinear interpolation values. Since we use the 32s variant of the FCN architecture described in [6], we do no use skip layers during upsampling.

5.3. Semantic Model

The semantic model leverages a Word2Vec model pretrained on the 3 billion word Google News corpus. We start with a 3000000x300 matrix, reflecting the 300-dimension Word2Vec embeddings for the 3 million most commonly occurring tokens. We reduce the dimensionality of this matrix via principle component analysis from 300 to j dimensions to project the labels into the joint-embedding space. In this work, we set $j = 20$, which allows us to capture about 29% of the variance of the

original matrix while minimizing computational load (a.k.a. running time for our deep learning models to converge).

5.4. Joint-embedding Loss

We experimented with two potential joint-embedding loss functions for training the FCN and visual embedding head of the network: a mean square error loss and a cosine loss.

The mean square error loss calculates a pixelwise loss for a given image I using two arguments, the visual-embedding v and the semantic embedding s and is defined as:

$$L(v,s) = \frac{1}{\|I\|} \sum_{i,j \in I} \left\| v_{i,j} - s_{i,j} \right\|^2 \tag{4}$$

where $\left\| s_{i,j} \right\| \leq 1$. We normalize all the semantic embeddings by the max norm among all the embeddings.

The second loss is based on the idea of the cosine similarity between two vectors. Cosine similarity is often used in natural language processing to quantify word similarity. The cosine similarity score $sim(x,y)$ between two vectors x and y is calculated as:

$$sim(x,y) = \frac{x \cdot y}{\|x\| \, \|y\|} \tag{5}$$

where $sim(x,y) \in [-1,1]$. We formulate a joint-embedding loss function around cosine similarity as follows:

$$L(v,s) = \frac{1}{\|I\|} \sum_{i,j \in I} 1 - sim(v_{i,j}, s_{i,j}) \tag{6}$$

5.5. Code Details

The code is implemented using PyTorch, a python deep learning library optimized for GPU utilization. The code for this project is publicly available on GitHub and will remain open-sourced [1]. It adapts an existing PyTorch FCN implementation for the visual model [10].

6. Evaluation Methodology

6.1. Datasets

We are working with two different datasets, each of which tests different assumptions and behaviors of the model.

6.1.1. PASCAL VOC Our first dataset is PASCAL VOC 2011, a real-world image dataset for object class recognition. It has 21 classes, including 6 vehicle classes (e.g. aeroplane and boat), 7 animal classes (e.g. bird and cat), 6 household object classes (e.g. sofa and bottle), and 1 background class. The default PASCAL VOC dataset contains semantic segmentation annotations for a 2223 image subset (1112 train and 1111 val), which is insufficient in size for our needs. So, we leverage the Semantic Boundary Dataset (SBD), which offers further annotation of PASCAL VOC 2011, with 11318 images divided into 8498 train and 2820 val images (~7x increase in train data).

One issue is that the train/val splits for SBD and PASCAL VOC 2011 do not match and often intersect, requiring extra attention. To reconcile these two sets of annotations, we train on 8498 images from the SBD's train split. And, we test on a 735 image subset of PASCAL VOC 2011's val split, which we get after ignoring intersections with SBD's train and val splits.

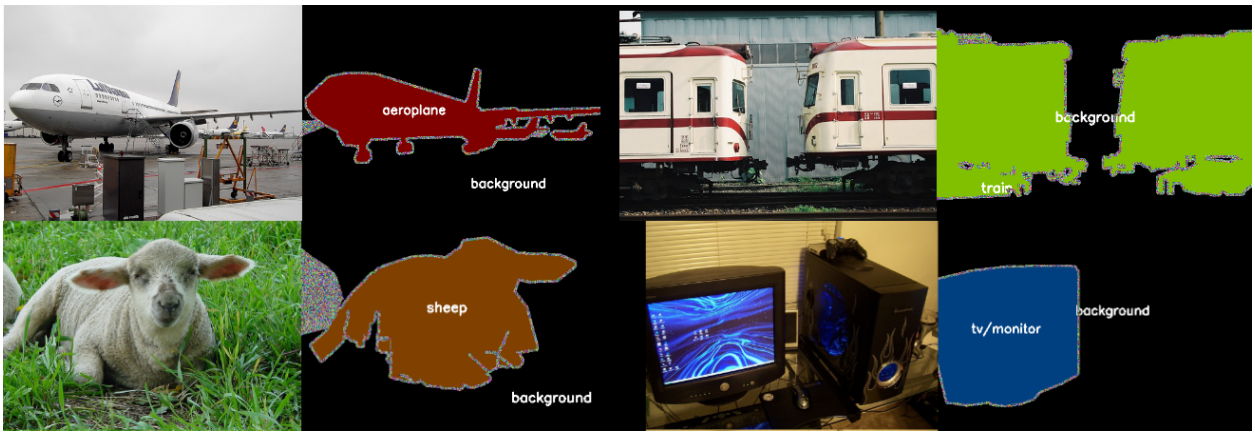


Figure 5: Sampled images from PASCAL VOC with semantic segmentation labels

6.1.2. PASCAL-Context Most of the images in PASCAL VOC typically contain one or two object classes that are prominently featured, with a majority of other types of objects simply ignored and encapsulated in the background class. This motivates the need for a richer set of semantic

segmentation annotations with more types of object classes (a higher class density) and annotations for the entire scene (a.k.a. no background class).

PASCAL-Context is a further annotation of PASCAL VOC 2010 which provides annotations for the whole scene. It contains 33 classes, including the 20 object classes from PASCAL VOC, along with an additional 13 classes to cover background scenery (e.g. grass and sky) which would have been relegated to being part of the background class in PASCAL VOC. Pascal-Context is a more challenging dataset than PASCAL VOC for semantic segmentation. As seen in Figure 6, it has a higher class density, with more object instances per image region, often in more complex orientations. And, there are more classes for visual models to choose between, resulting in errors at ambiguous class boundaries in embedding spaces. We train on 4998 images from the official training split, and we test on 5105 images from the official val split. Notice that there are fewer training images in PASCAL-Context (4998) compared to PASCAL VOC with SBD annotations (8498).

Figure 6: Sampled images from PASCAL-Context with semantic segmentation labels

6.2. Evaluation Metrics

We will be using these 4 popular semantic segmentation evaluation metrics that are variations of pixel accuracy and intersection over union (IU), as defined in [6]. First, we define these three values used in the metric definitions:

- $n_{i,j}$ is the number of pixels of class i predicted to belong to class j
- n_{cl} is the number of classes
- $t_i = \sum_j n_{i,j}$ is the total number of pixels of class i

The four evaluation metrics are:

$$\text{pixel accuracy (PA)} = \frac{\sum_i n_{ii}}{\sum_i t_i} \tag{7}$$

$$\text{class accuracy (CA)} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i} \tag{8}$$

$$\text{mean IU (mIoU)} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{9}$$

$$\text{frequency weighted mean IU (wIoU)} = \frac{1}{n_{cl}} \frac{1}{\sum_k t_k} \sum_i \frac{t_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{10}$$

6.3. Overview of Models

In this section, we define the shorthand used to describe the common types of models we will be comparing.

- **FCN+Softmax (FCN-S)** This includes a FCN with n-way softmax classifier for outputting a class probabilities vector. We apply an argmax after applying the softmax function to the class probability vector to get a label. Since FCN-S does not use a joint-embedding approach, it is not able to make zero-shot inferences; rather, it can only classes it has observed prior during training. This also implies no seenmask is generated or used.
- **FCN w/ Joint Embeddings (FCN-JE)** This includes a joint-embedding model in which a FCN projects pixels into the same joint-embedding space as a set of candidate labels. However, a seenmask is not generated, so only basic nearest embedding inference is used (similar to DeViSE). Thus, pixelwise inference is among all $\text{train}_{\text{seen}}$ classes; it is not broken down among seen classes and unseen classes since there is no seenmask to patch together the seen and unseen predictions in a pixelwise manner.
- **Seenmask Zero-shot Network (SZN)** This is the full Seenmask Zero-shot Network described in Section 4. Unlike FCN-JE, this model incorporates the seenmask head of the FCN and SEI

(seenmask embedding inference) to leverage the seenmask.

- **SZN with Perfect Seenmask (SZN-PS)** This is a modification of SZN, except the seenmask head of the FCN automatically outputs perfect seenmask during test-time inference based on the true labels. This model helps demonstrates the theoretical best performance of the SZN if the seenmask were to work perfectly.

Also, we will we labeling each model with a tuple to indicate the number of classes assigned to each of the three class splits:

$$(||\text{train}_{\text{seen}}||, ||\text{train}_{\text{unseen}}||, ||U||) \tag{11}$$

To give a concrete example, a SZN (9, 2, 10) model working with the 21 class PASCAL VOC dataset:

- discards images with any pixel-labels from among the 10 classes in U during training.
- trains its FCN and visual embedding head on images with pixels from only among a set of 9 $\text{train}_{\text{seen}}$ classes.
- trains its seenmask head using images whose every pixel belongs to either the 9 classes of the $\text{train}_{\text{seen}}$ set (label = 1) or the 2 classes of the $\text{train}_{\text{unseen}}$ set (label = 0).

6.4. Reading Semantic Segmentation Figures

Our model outputs semantic segmentations in a two-row format, with the top row featuring the true labels and the bottom row featuring the predicted labels. Using Figure 7 as a reference, let us break things down further by column:

- **First Column: Input Image** The first column contains the original image, which is the same across both rows.
- **Second Column: Semantic Segmentation** The top row contains the true semantic segmentation labels. The bottom row contains the predicted semantic segmentation labels.
- **Third Column: Seenmask** For zero-shot models, we add a third column to indicate whether pixel-labels from the second column belong to classes from the training dataset ($\text{train}_{\text{seen}}$ or

Figure 7: Sample semantic segmentation output (with zero-shot seenmask in third column)

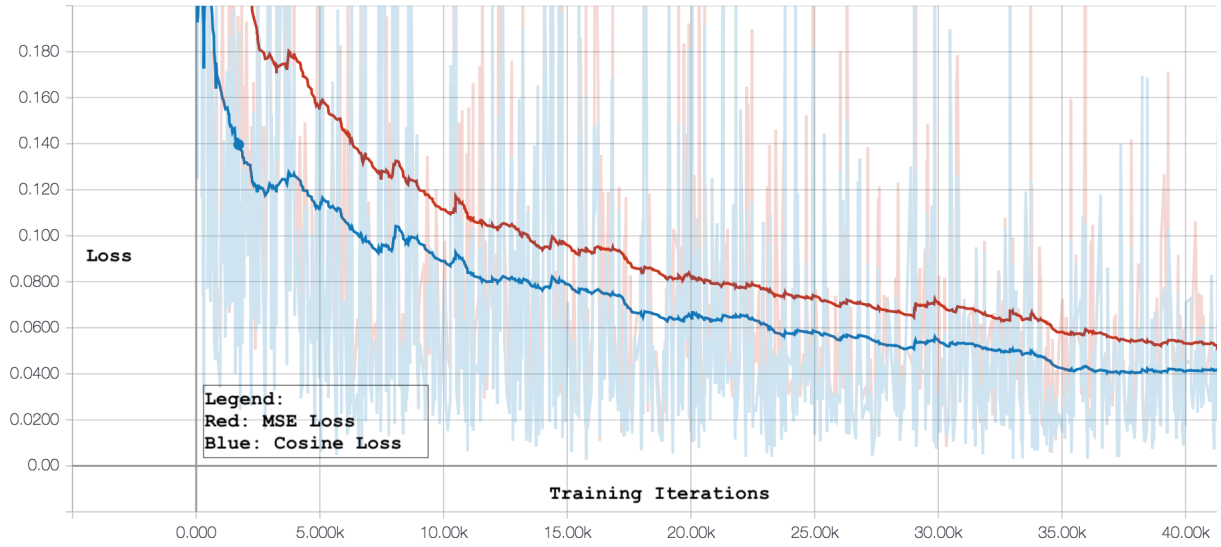
$\text{train}_{\text{unseen}}$), which are colored white, or to unseen classes from the test dataset (U), which are colored black. The bottom row of the third column really helps us understand the correctness of the seenmask for SZN networks since black pixels indicate that the model is able to predict unseen classes not seen by any part of the training network.

7. Results: Training Methodology

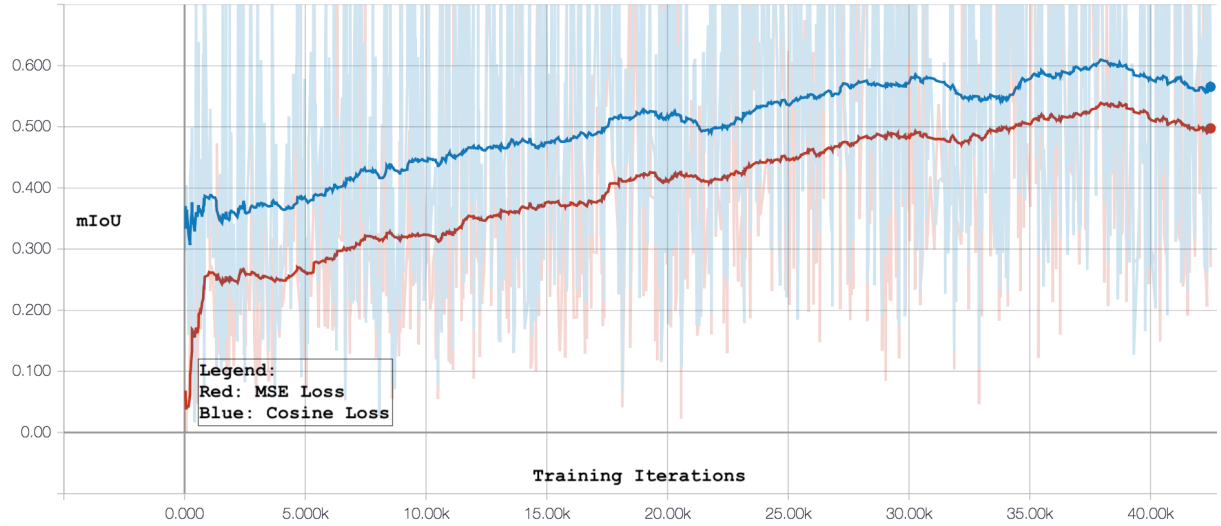
The goal of this section is to explain the choice of training parameters for our SZN model using experimental results.

7.1. Joint-embedding Loss: MSE vs Cosine

Our joint-embedding model with a cosine loss converges faster and achieves higher accuracy than one with a mean square error (MSE) loss. We trained FCN-JE VOC (21, 0, 0) using both loss functions and examined the training loss and training mIoU curves after 10 epochs. Figure 8a reveals that the cosine training loss converges faster than the MSE loss. And, Figure 8b shows the cosine loss converging to higher accuracy (roughly .1 mIoU higher) relative to the MSE loss. This can be attributed to how a cosine loss during training better emulates the nearest embedding inference approach used during testing, which relies on the cosine similarity function.



(a) Training Loss Curves



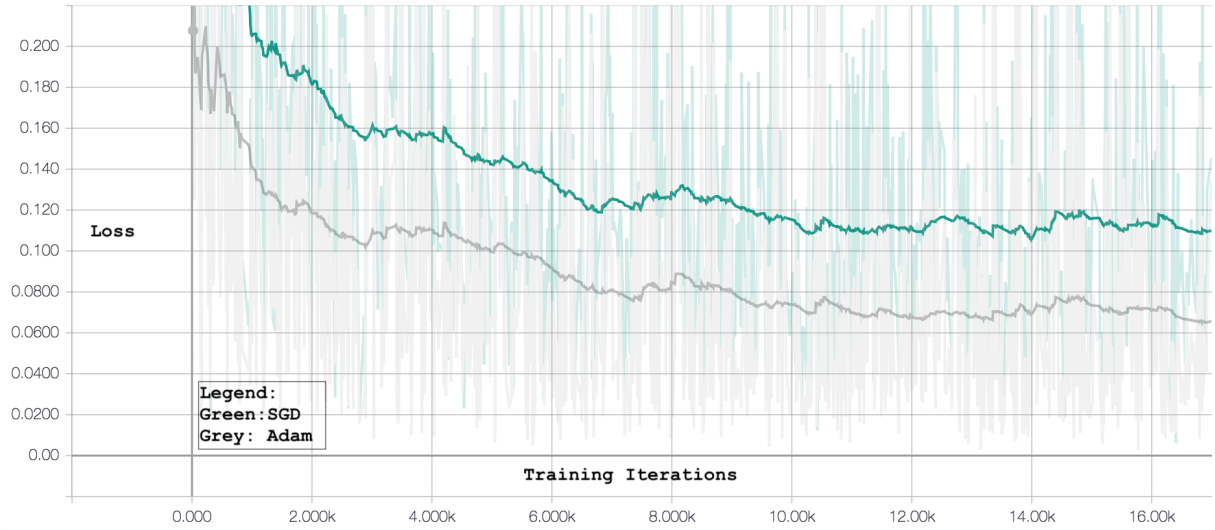
(b) Training mIoU

Figure 8: MSE vs Cosine Loss

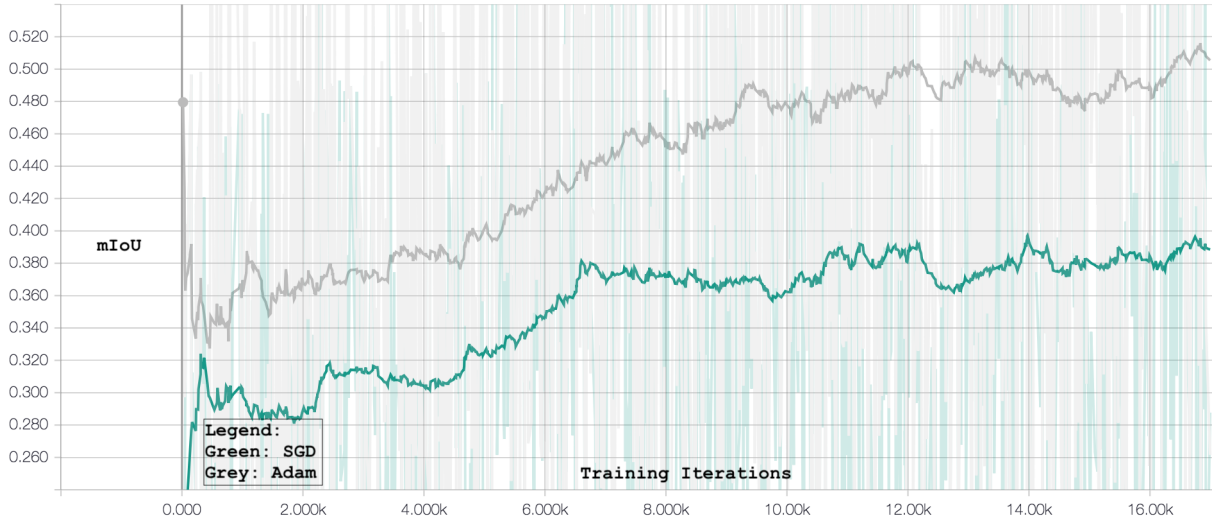
7.2. Optimizer: SGD vs Adam

Joint-embedding models coverage faster and reach higher accuracy using the Adam optimizer relative to a stochastic gradient descent (SGD) optimizer. We trained FCN-JE VOC (21, 0, 0) using both optimizers and examined the training loss and training mIoU curves after 3 epochs. Figure 9a confirms the loss converges faster with the Adam optimizer. And, Figure 9b verifies the Adam optimizer converges to a much higher training accuracy. This can perhaps be attributed to how an Adam optimizer uses an adaptive learning rate in response to the evolving magnitudes of the

gradients, while the SGD optimizer uses a fixed learning rate.



(a) Training Loss Curves



(b) Training mIoU

Figure 9: SGD vs Adam Optimizer

7.3. Joint-embedding Space Dimensionality

We considered a variety of dimensionalities j for the joint-embedding space: 1,2,5,20, 50, 100, 200, and 300. Larger joint-embedding dimensionality will allow for higher-dimension semantic embeddings that capture more signal. We reduce the dimensionality of our original Word2Vec

embeddings to j after applying principal component analysis (PCA). We quantify the explained variance of each candidate j after applying PCA in Table 1:

Dimensions	Variance
2	.07
5	.13
10	.20
20	.29
50	.46
100	.62
200	.84
300	1.00

Table 1: Explained variance of original 300-dim Word2Vec embeddings after applying PCA

Explained variance is calculated from the largest-to-smallest ordered list of eigenvalues $e_1, ..., e_k$ of the matrix, where $k = 300$ in this case. The explained variance after reducing the matrix to $j < k$ dimension is calculated as:

$$\frac{\sum_{a=1}^{j} e_a}{\sum_{b=1}^{k} e_b} \tag{12}$$

Higher j allows for more expressive semantic embeddings and, theoretically, a richer continuity of concepts in the intermediate spaces between labels in the joint-embedding space. To illustrate the idea behind conceptual continuity for semantic embeddings, the embedding for Dalmatian should be in the space between dog and zebra. However, larger j also requires a greater number of computations and will increase training time. Furthermore, a higher dimensionality will also require a higher class density of seen classes to enable the FCN learn how to map pixels into various regions of the joint-embedding space; otherwise, without enough classes, it will be difficult to map to intermediate spaces between known concepts in the joint-embedding space for zero-shot learning.

We find that setting $j = 20$ hits the right balance between capturing sufficient expressiveness from the original 300-dim Word2Vec embeddings (with 29% of the explained variance) and keeping running times for our joint-embedding models within a reasonable time-frame for rapid iteration

(~12 hours). And, it works with the limited class density of our dataset (21 classes in PASCAL VOC and 33 classes in PASCAL-Context).

8. Results: Model Accuracy

8.1. Semantic Segmentation Baselines

We baseline the semantic segmentation accuracy of our FCN with two different inference schemes. FCN-S is a fully connected network that outputs a vector of class probabilities normalized by the softmax function. By going from FCN-S to FCN-JE, we adapt our FCN to be part of a joint-embedding model. Using a joint-embedding model enables zero-shot learning via nearest neighbor inference. We examine the accuracy impact across two semantic segmentation datasets (PASCAL VOC and PASCAL-Context).

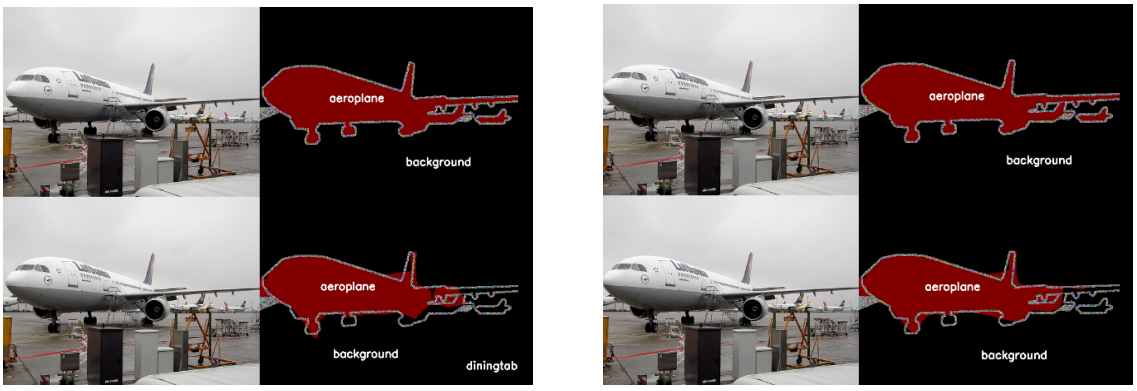
Model	Overall (%)			
	PA	CA	mIoU	wIoU
VOC FCN-S (21, 0, 0)	90.7	74.7	63.2	83.6
VOC FCN-JE (21, 0, 0)	90.6	71.7	61.5	83.6
Context FCN-S (21, 0, 0)	58.3	45.1	32.5	41.9
Context FCN-JE (21, 0, 0)	58.0	44.9	31.7	42.0

Table 2: Semantic Segmentation Baseline Results

Table 2 shows that there is a minimal sacrifice in accuracy going from softmax inference to nearest embedding inference via a joint embedding model for both the PASCAL VOC and PASCAL-Context dataset. The pixel accuracy and mIoU roughly stay the same. Notice how all the accuracy metrics for PASCAL-Context are significantly lower than PASCAL VOC using the same models. This reinforces our conclusion from the dataset discussion, which predicted that Pascal-Context dataset is a more difficult dataset due to its higher class density, complex object orientations, and smaller dataset size.

Generally, the semantic segmentation predictions from FCN-S and FCN-JE are indistinguishable,

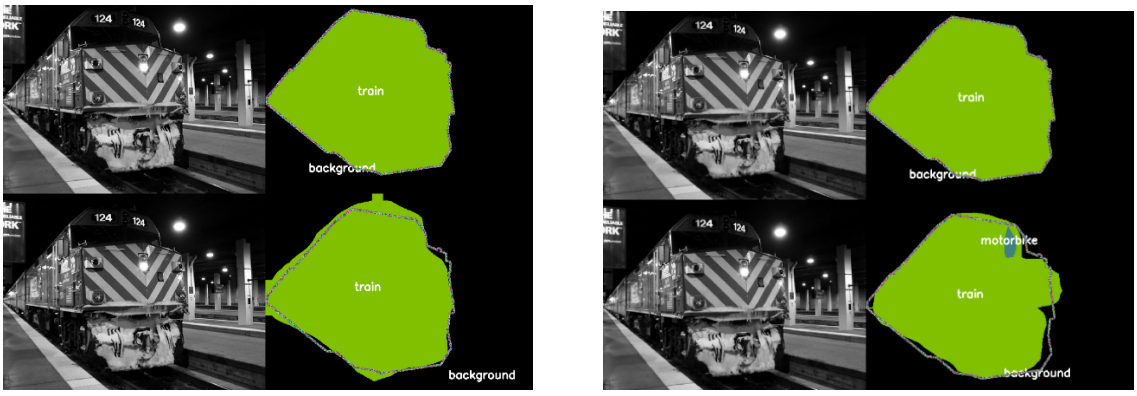
with only minor variances in narrower structures, such as in the wings of an an aeroplane, as seen in Figure 10.



(a) FCN-S (b) FCN-JE

Figure 10: FCN-S vs FCN-JE Error Analysis for the "Aeroplane" Object

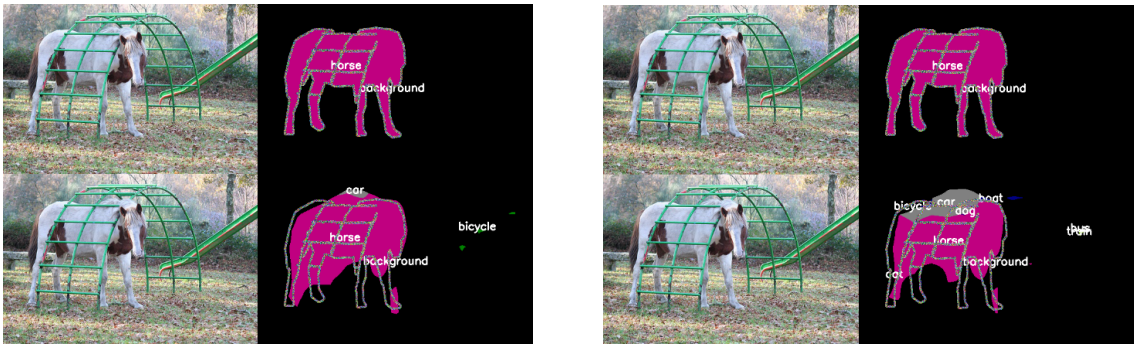
By examining where the joint-embedding model breaks down, we find evidence that the visual model is learning to interpolate a continuity of concepts in the joint-embedding space. For example in Figure 11b, there is small patch of pixels of the object "train" (green) that are misclassified as "motorbike" (blue), a similar concept in the vehicle category that is nearby in semantic embedding space. Thus, the predicted visual embeddings in the joint-embedding space for the misclassified patch of pixels was generally in the correct region but must have been at an ambiguous boundary between the "train" and "motorbike" concepts.



(a) FCN-S (b) FCN-JE

Figure 11: FCN-S vs FCN-JE Error Analysis for the "Train" Object

There are also many examples of where FCN-JE struggles with object cohesiveness and smoothness in a way that the FCN-S does not, suggesting a potential room for improvement in the joint-embedding model. For example, in Figure 12a, we see that the FCN-S is able to correctly identify most of the horse as one continuous object, with a few misclassified pixels on the top of the horse. But, as seen in Figure 12b, FCN-JE's error on the top of the horse covers a much larger region broken down into multiple objects, suggesting that these visual embeddings appear to be at an ambiguous joint-embedding space boundary between multiple different classes. Most of the pixels are misclassified as belonging to the "car" class (gray pixels). Given how the other misclassifications range from boat to bicycle, the concept continuity in the joint-embedding space is not perfect, which may be a function of limitations in the concept continuity of the source Word2Vec embeddings.



(a) FCN-S (b) FCN-JE

Figure 12: FCN-S vs FCN-JE Error Analysis for the "Horse" Object

8.2. Zero-shot: Analyzing Class Density

In this section, we investigate the importance of class density (the number of $\text{train}_{\text{seen}}$ classes in the training image dataset) for zero-shot inference. We examine three zero-shot models trained on different class densities:

- **Reduced Data Model** VOC (9, 2, 10)
- **Standard Data Model** VOC (17, 2, 2)
- **Extended Data Model** Context (31, 2, 2)

To help with the inter-comparability of the data-splits, all three models include these two PASCAL object classes among $\text{test}_{\text{unseen}}$: "sheep" (the animal) and "train" (the vehicle).

8.2.1. Reduced Data Model We train the Reduced Data Models on 9 seen classes from VOC using the (9, 2, 10) split. In Table 3, from the FCN-JE row, we can see that the model overfits on seen classes and is unable to predict any unseen classes, which is why all the unseen metrics are at zero.

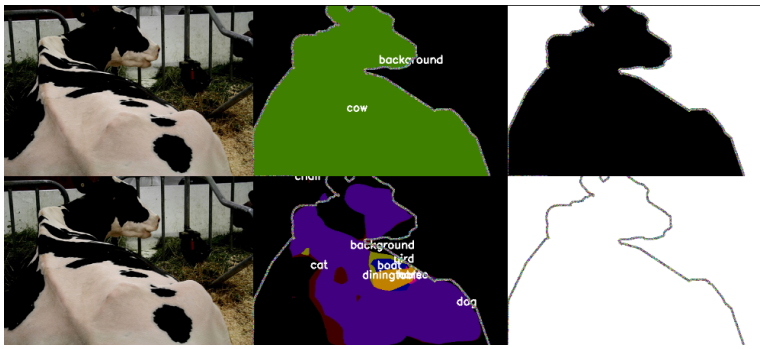
Model	Overall (%)				Seen (%)				Unseen (%)			
(9, 2, 10))	PA	CA	mIoU	wIoU	PA	CA	mIoU	wIoU	PA	CA	mIoU	wIoU
FCN-JE	77.5	31.3	17.2	65.1	92.3	59.8	31.0	87.2	00.0	00.0	00.0	00.0
SZN	77.9	33.6	21.3	66.7	91.5	59.5	28.9	87.3	6.5	5.0	1.5	4.9
SZN-PS	82.9	41.9	28.6	76.5	92.9	65.8	33.3	88.7	30.5	15.5	6.2	15.4

Table 3: Reduced Data Model Results

We can visually confirm this by looking at FCN-JE predictions for unseen classes such "cow", as seen in Figure 13a. The bottom row of the third column only contains white pixels, indicating all inferred pixels are among U.

The SZN shows accuracy improvements over the FCN-JE model, with pixel accuracy rising from 77.5% rising from 77.9% and mIoU rising from 17.2% to 21.3%. More importantly, the accuracy among unseen classes rose significantly. For example, wIoU rose from 0% to 4.9%. By comparing Figure 13b to Figure 13a, we see how the SZN, unlike the FCN-JE, is able to predict that some of the cow pixels as belonging to seen classes. In fact, in Figure 13b, a large number of the unseen predictions (black pixels in the bottom row of the third column) actually are labeled as cow (green pixels in the bottom row of the second column). This is significant because it shows the model is able to correctly identify objects which it correctly infers it has not seen before.

We now look at SZN-PS to understand the theoretical upper limit on model performance if the seenmask were perfect. As seen in Table 3, going from SZN to SZN-PS results in a pixel accuracy boost from 6.5% to 30.5% and a wIoU boost from 4.9% to 15.4%, demonstrating the value of focusing future efforts on improving the seenmask. Also, if we compare the bottom row of the third column of Figure 13c to that of Figure 13b, we see how more of the cow is colored black, correctly



(a) FCN-JE

(b) SZN

(c) SZN-PS

Figure 13: Reduced data model: comparing FCN-JE, SZN, and SZN-PS models

labeling the cow region as belonging to classes in U. The other white regions on the cow in Figure 13c indicate predicted labels are among the $\text{train}_{\text{unseen}}$ set of labels. Thus, even with the perfect seenmask, unseen inference is often predicting the incorrect label, even among the smaller set of candidate unseen labels ($\text{train}_{\text{unseen}} \cap U$). This demonstrates that the FCN is not able to properly map pixels to the correct semantic embedding in the joint-embedding space. This could be attributed to the low class density of the Reduced Data Model scenario. After all, the FCN only has observed 9 classes and is expected to infer the mappings of another 12 classes with a completely different set

of statistical properties. This draws our attention to the standard data model, with a higher class density.

Before that, we'd also like to understand the source of errors for unseen classes in the SZN model, especially at a per-class level, which is why we analyze the confusion matrix for pixelwise predictions. Each row is normalized to sum to 100. Thus, each square's value can be interpreted as the probability that a given true label (row label) has been inferred as another predicted label (column label). Among the unseen classes, we'll focus on the sheep class and train class, since these are also unseen classes in the other scenarios we examine later.

- **Sheep class error analysis** Less than 1% of true sheep pixels are predicted as belonging to the sheep class. Instead, 37% of sheep pixels are predicted as dog (from U), 21% as cow (from $\text{train}_{\text{seen}}$), and 16% as background (from $\text{train}_{\text{seen}}$). Thus, misclassification are often from conceptually similar classes since dog and cow are also both animals. And, the model is no longer overfitting to seen classes, as evidenced by the dog misclassifications.
- **Train class error analysis** Only 2.2% of train class pixels are inferred to actually belong to the train class. Instead, 36% are inferred as belonging to the background (from $\text{train}_{\text{seen}}$), 27% to the car class (from U), and 16% to the boat class (from $\text{train}_{\text{seen}}$). Thus, misclassification are often from conceptually similar classes since car and boat are also both vehicles. And, the model is no longer overfitting to seen classes, as evidenced by the car misclassifications.

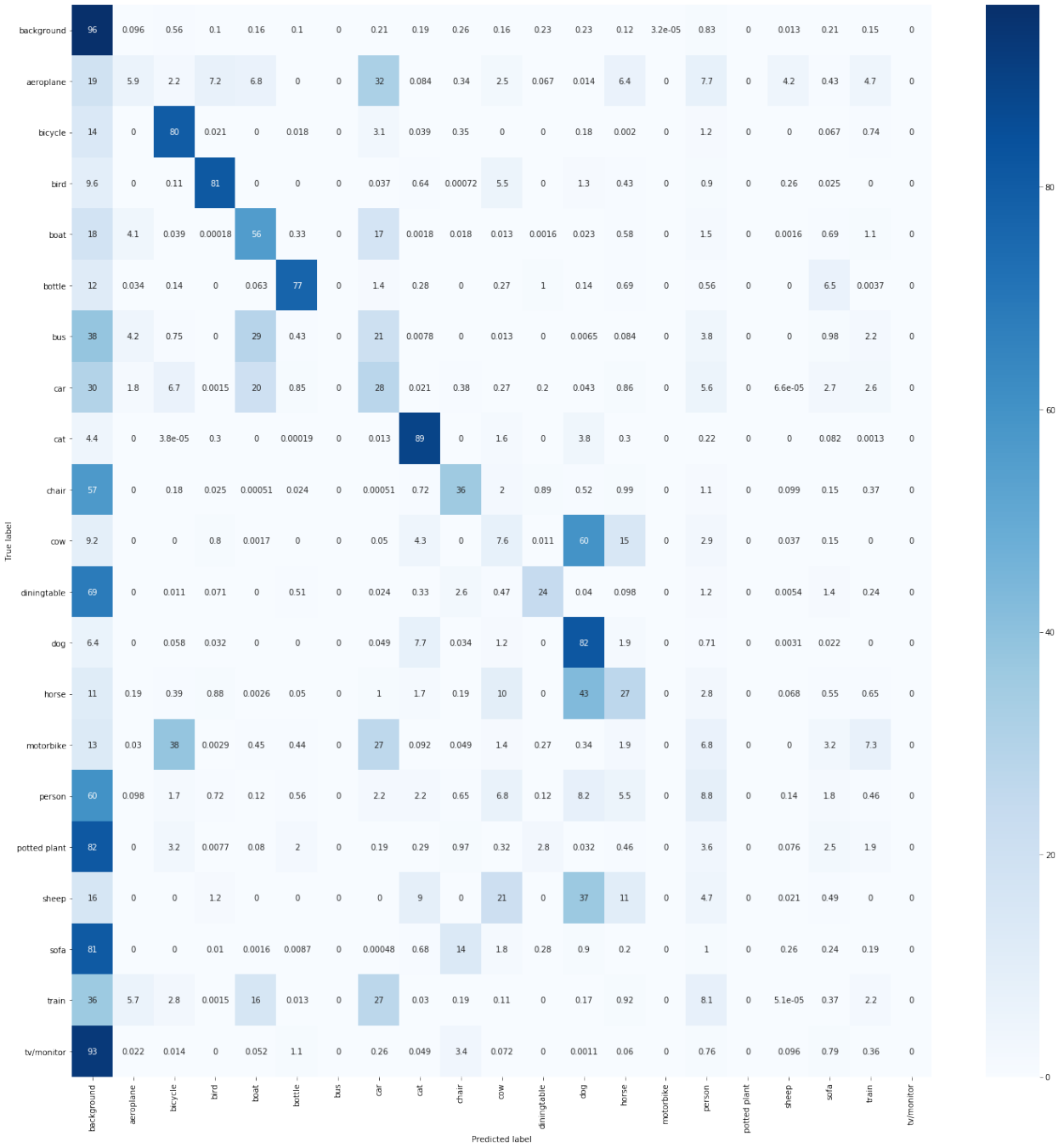


Figure 14: Reduced Data Model: SZN Confusion Matrix

In both cases, although the SZN model is mislabeling these two unseen classes almost in all cases, the misclassifications often belong to other conceptually related seen and unseen classes.

8.2.2. Standard Data Model To see how greater class density affects accuracy, we train the Standard Data Models on 17 seen classes from VOC using the (17, 2, 2) split. Here are our results:

Model	Overall (%)				Seen (%)				Unseen (%)			
(17, 2, 2)	PA	CA	mIoU	wIoU	PA	CA	mIoU	wIoU	PA	CA	mIoU	wIoU
FCN-JE	87.5	58.6	43.8	78.8	89.8	64.7	46.7	82.3	0.2	0.1	0	.2
SZN	87.8	61.3	48.8	80.0	89.6	65.8	49.9	82.9	18.6	18.6	2.1	18.6
SZN-PS	89.6	67.8	55.1	82.7	90.5	69.4	52.6	83.9	57.0	52.9	26.4	56.9

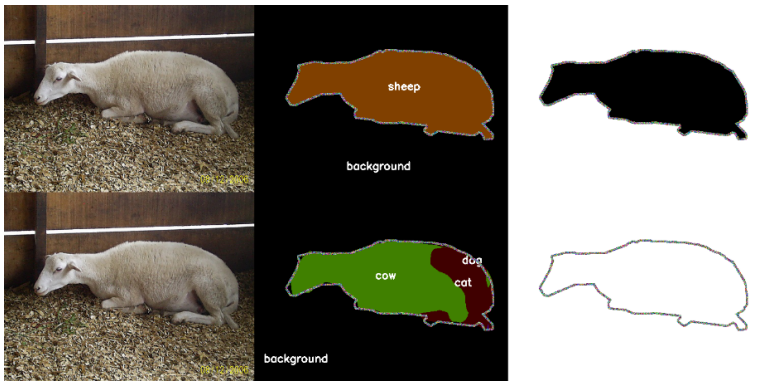
Table 4: Standard Data Model Results

As seen in Figure 15, many of the same trends from the Reduced Data Model still hold. Figure 15a shows the FCN-JE still overfits on seen classes and struggles to infer unseen classes, with pixel accuracy among unseen classes at just .2%. Still, we see a slight overall accuracy boost as we go from the FNC-JE model, with a mIoU of 43.8, to the FCN-JE model, with a mIoU of 48.8. This is driven by the large increase in unseen accuracy. Pixel accuracy and wIoU both rose from roughly 0 to 18.6, indicating two things. First, unseen pixels are being assigned unseen labels, so the seenmask is working. Second, the FCN is correctly able to map these unseen labels into the correct region of the joint-embedding space, allowing for correct inference, even among the unseen classes. This is visually confirmed with our example of a sheep, an unseen object, in 15b. In the third column, the black pixels which the seenmask predicts as unseen are correctly assigned the label of sheep. This is significant because it shows the model as able to correctly infer the identity of objects it correctly infers it has not previous observed.

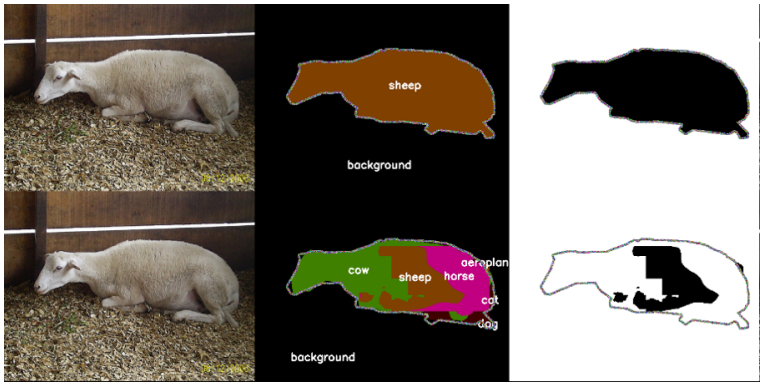
Now, if we compare SZN-PS to SZN, we see how with a perfect seen mask, wIoU among unseen classes jumps from 18.6% to 56.9%, indicating to degree to which a better seenmask can boost accuracy. The benefit of a perfect seenmask can be seen in Figure 15c, in which most of the sheep object is correctly assigned the sheep label.

The confusion matrix in Figure 16 helps us understand the nature of the misclassifications for the unseen classes (the "sheep" class and "train" class):

- **Sheep class error analysis** 19% of the sheep pixels are correctly assigned the sheep class label, a significant increase compared to the Reduced Data Model. The sheep pixels are often mislabeled as horse and cow, which happens for 28% and 16% of true sheep pixels, respectively. Given how incorrect labels belong to animals, which are conceptually similar objects, it appears misclassified



(a) FCN-JE



(b) SZN

(c) SZN-PS

Figure 15: Standard data model: comparing FCN-JE, SZN, and SZN-PS models

pixels are along some ambiguous concept boundary in joint-embedding space.

- **Train class error analysis** 19% of the train pixels are correctly assigned train labels, a significant increase compared to the Reduced Data Model. Still, 33% of the train object pixels are labeled as background and another 27% as bus. This suggests that the bus-train class border in the joint-embedding space must be ambiguous.

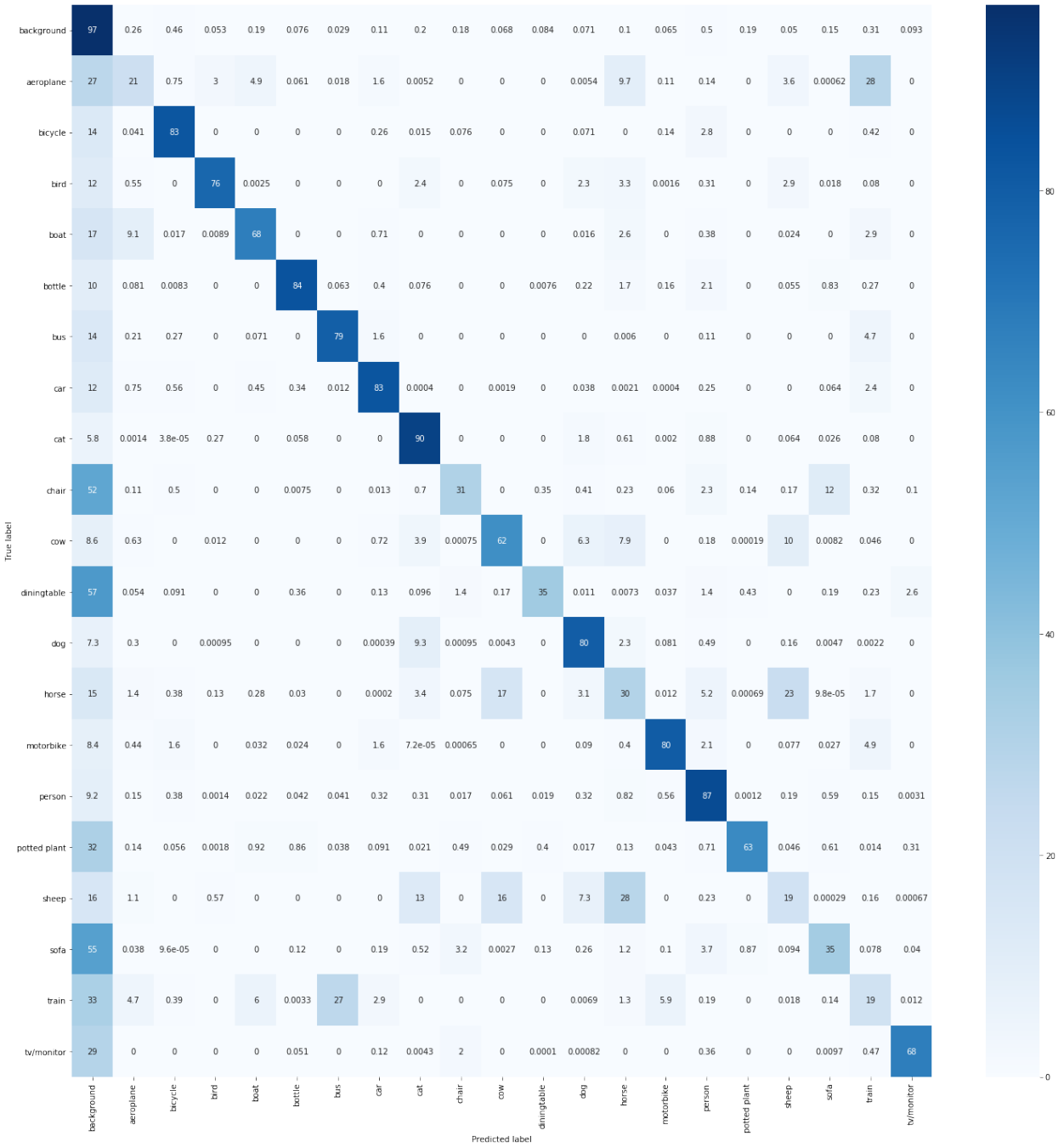
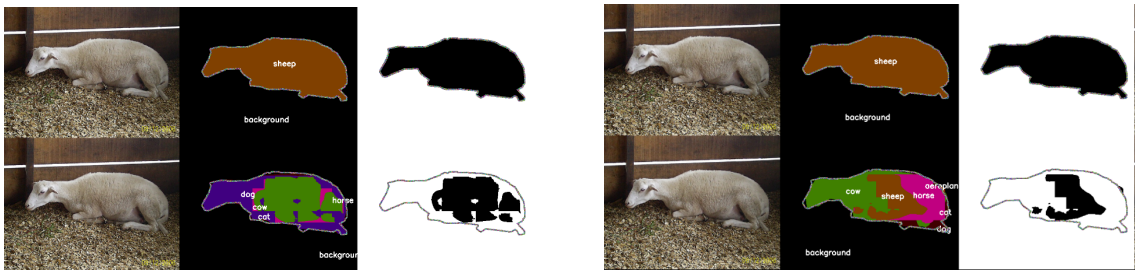


Figure 16: Standard Data Model: SZN Confusion Matrix

And, comparing the results from SZN (9, 2, 10) and SZN (17, 2, 2) visually in Figure 17, we can clearly see how much the higher class density helps the network learn how to map unseen objects into the joint-embedding space. In Figure 17a, we actually see that the seenmask for (9, 2, 10) actually manages to predict more of the sheep pixels as unseen. However, all of the unseen

predictions are for the cow class (green), a misclassification. But, in Figure 17b, even though the seenmask's unseen pixel predictions over the sheep object covered a smaller area, all of the unseen pixels were correctly labeled as belong to the sheep class. We can see how a higher class density allows the FCN learn how to map a wider variety of unseen pixels-label pairs into correct regions of the joint-embedding space. Yet, as we increase the class density, the seenmask accuracy will suffer unless we provide more examples of $\text{train}_{\text{unseen}}$ classes.



(a) SZN (b) SZN-PE

Figure 17: Reduced vs Standard data model: the importance of class density

8.2.3. Extended Data Model We train the Extended Data Models on 31 seen classes from Pascal-Context using the (31, 2, 2) split. Here, we we see if the class density observations extrapolate to another, more challenging dataset. Here are our results:

Model	Overall (%)				Seen (%)				Unseen (%)			
(31, 2, 2)	PA	CA	mIoU	wIoU	PA	CA	mIoU	wIoU	PA	CA	mIoU	wIoU
FCN-JE	56.0	38.5	25.7	39.5	57.2	41.0	26.2	40.8	.1	.1	0	.1
SZN	56.1	38.7	26.2	39.7	57.2	41.0	26.5	41.0	2.7	3.0	.2	2.7
SZN-PS	58.1	43.7	29.9	41.8	57.9	42.8	28.3	42.0	68.4	58.4	21.5	63.0

Table 5: Context (31, 2, 2) Results

Many of the same trends for the Extended Data Model match those of the Reduced and Standard Data Models. Interestingly, despite the high class density of the Extended Data Model (31 seen classes), the increase in accuracy for the SZN model relative to the FCN-JE model is relatively small. Given that unseen inference is only among 4 classes (2 $\text{train}_{\text{unseen}}$ and 2 $\text{test}_{\text{unseen}}$ classes), even random guessing on unseen pixels would give 25% accuracy. This suggest that the seenmask

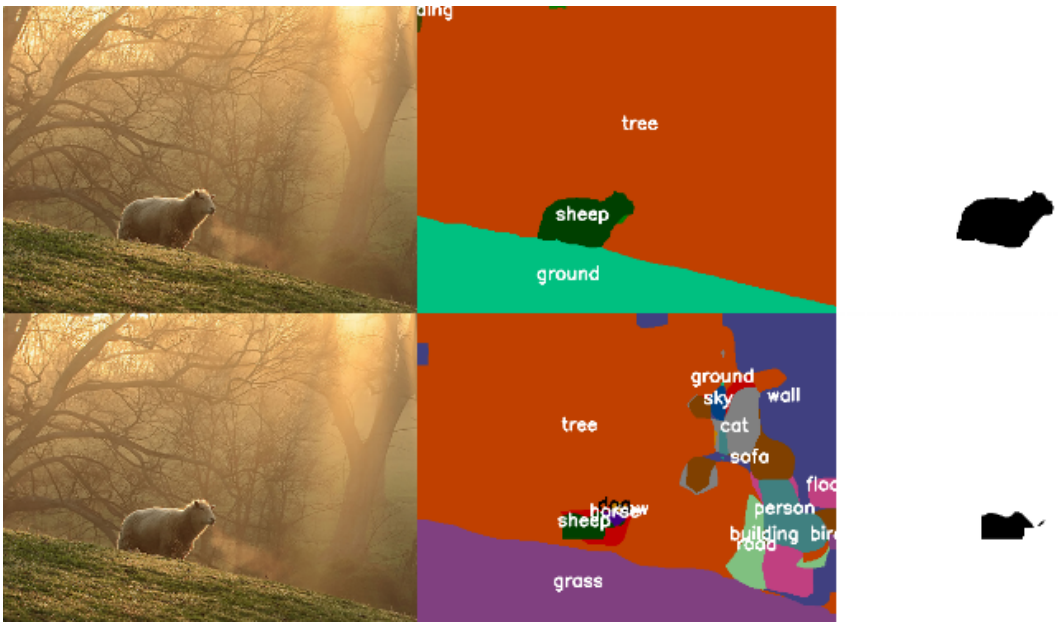


Figure 18: Extended Data Model: SZN example of sheep object (an unseen class)

is not correctly labeling the unseen pixels as unseen in the first place. For instance, look at the bottom row of the third column of the following sheep examples in Figure 18. Only the center of mass for the sheep is labeled correctly, with the edges of the sheep heavily misclassified as other seen objects.

It appears that as the number of $\text{train}_{\text{seen}}$ classes increases, the number of $\text{train}_{\text{unseen}}$ classes also needs to increase so that the binary seenmask classifier has a good balance of seen and unseen examples to learn from. Right now, the visual examples indicate that the seenmask head of the visual network rarely predicts a pixel as unseen because there is not enough distributional information from unseen objects to differentiate it from the seen classes.

Note that when we run SZN-PS with a perfect seenmask, pixel-accuracy increases dramatically from 2.7% to 68.4%. Thus, the joint-embedding alignment is really strong with a higher class density. The limitation in the SZN model seems to be the seenmask.

8.2.4. Comparing Reduced, Standard, and Extended Data Models With Perfect Seenmask To isolate the importance of class density, we will be comparing the Reduced, Standard, and Extended Data Models using a perfect seenmask as opposed to the predicted seenmask for a common set of two unseen classes (the "sheep" class and the "train" class). This allows us to understand how well

each model is learning a generalizable mapping of pixel-label pairs into the joint-embedding space. To this end, we modify the Reduced Data Model so that the unseen metrics only apply to the two unseen classes of sheep and train.

Model	Overall (%)				Seen (%)				Unseen (%)			
SZN-PS	PA	CA	mIoU	wIoU	PA	CA	mIoU	wIoU	PA	CA	mIoU	wIoU
(9, 2, 2)	82.9	41.9	28.6	76.5	92.9	65.8	33.3	88.7	8.5	7.2	1.0	8.5
(17, 2, 2)	89.6	67.8	55.1	82.7	90.5	69.4	52.6	83.9	57.0	52.9	26.4	56.9
(31, 2, 2)	58.1	43.7	29.9	41.8	57.9	42.8	28.3	42.0	68.4	58.4	21.5	63.0

Table 6: Comparing Reduced, Standard, and Extended Data Models With Perfect Seenmask

As seen in table 6, as we increase the number of seen classes from 9 to 17 to 31, the unseen pixel accuracy increases dramatically from 8.5% to 57.0% to 68.4%. The same holds for wIoU, which rises from 8.5% to 56.9% to 63.0%. Clearly, a higher class density allows for better alignment among visual embeddings and semantic embeddings in joint-embedding space.

9. Conclusions and Future Work

In this work, we have defined a new task, zero-shot semantic segmentation, to drive visual recognition models towards pixel-level inference of unseen objects from an open vocabulary. To attempt this new task, we propose Seenmask Zero-shot Network, a novel type of deep learning network. Our results validate our approach, which aligns pixel-label pairs in a joint-embedding space and uses seenmask neighboring inference (SEI) to intelligently infer previous observed and unobserved objects.

An analysis of our results suggests there are three key limitations on SZN accuracy. First, datasets with higher class density are needed for better aligning visual and semantic embeddings in joint-embedding space. More data will allow visual networks learn a more generalizable mapping of pixels into a wider variety of regions in the joint-embedding space. Second, the seenmask head of the visual network needs more data to reach its full potential; particularly, more types of object classes to simulate unseen classes are needed as more seen classes are introduced into training datasets. The third key challenge is object continuity, especially since the class boundaries between

related concepts in joint-embedding space are often ambiguous, resulting in larger coherent objects breaking down into smaller objects from related concepts. In summary, we have laid out some future potential directions for researchers to build upon this work. Thus, we hope more work in these directions will help enable visual AI systems to reason in a visually confusing world.

References

[1] R. Doshi, "Zero-shot semantic segmentation," Mar. 2018. [Online]. Available: github.com/RohanDoshi2018/ZeroshotSemanticSegmentation

[2] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[3] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, "Devise: A deep visual-semantic embedding model," 2013.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[5] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015.

[7] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[8] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean, "Zero-shot learning by convex combination of semantic embeddings," 2014.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[10] wkentaro, "pytorch-fcn," 2018. [Online]. Available: https://github.com/wkentaro/pytorch-fcn