

# Language-based Semantic Segmentation

Samarth Bhatia

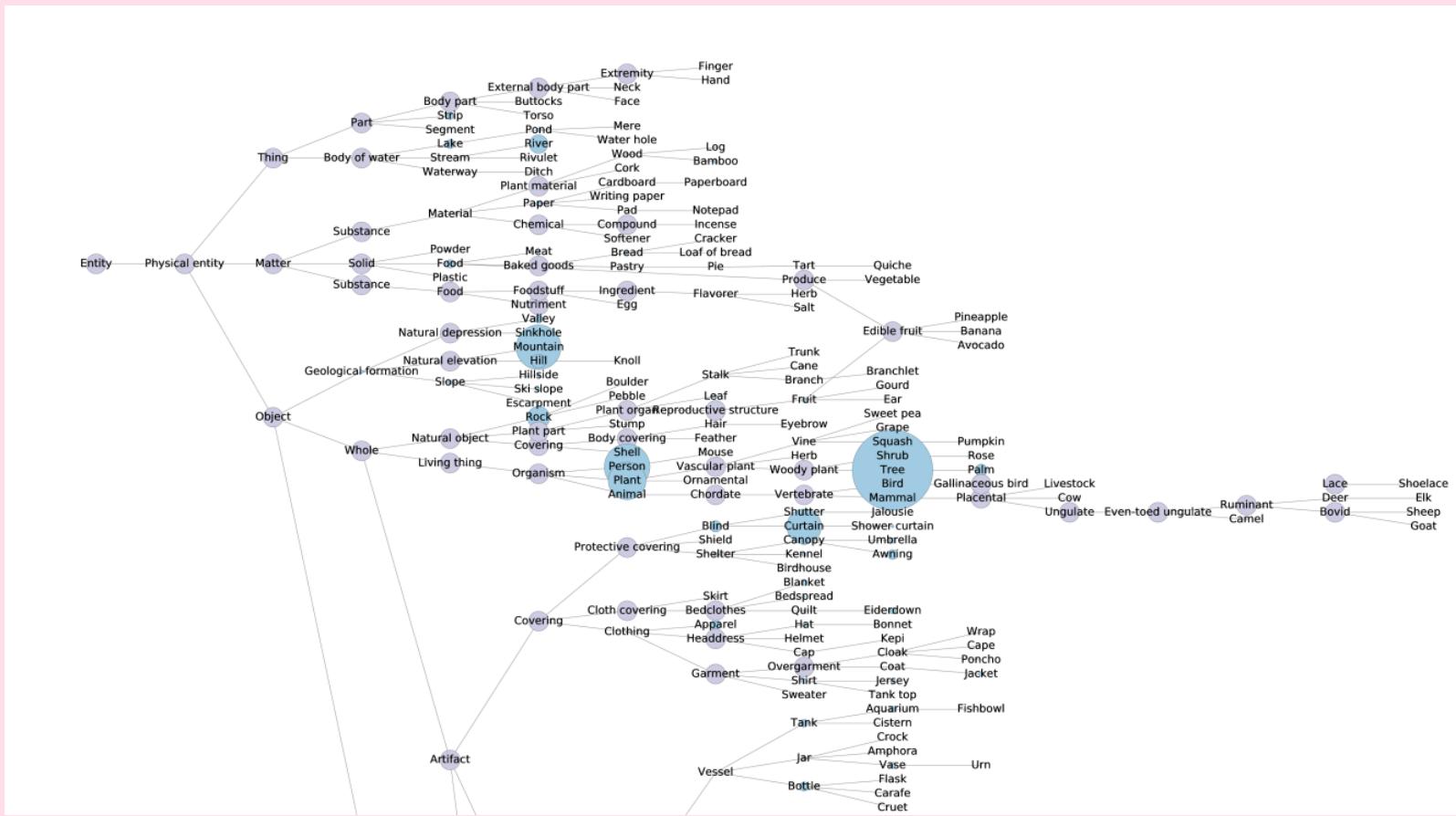
# Open Vocabulary Scene Parsing

What is it?

Model predictions not being limited to a fixed set of categories (**coco**: 80 classes),  
and instead being part of a large open dictionary (**WordNet**: 100,000 synsets).

Example : If the model has never seen **tricycle**, it still should give a plausible prediction as **vehicle**.

They take each class in **ADE20K** dataset and relate it with a synset(synonym set) from **WordNet**, end up with 2019 unique synsets forming a DAG with **entity** being the common root.



Part of the concept map created (The leaves are the specific objects and inner nodes are general concepts). The root is entity, since everything is an entity.

## Problem Settings

1. **Supervised:** Testing on the 150 training classes, pixel embedding is compared with all 150 concept embeddings and highest rank
2. **Zero-shot:** Tested on unseen validation classes, taken classes above a threshold to be predictions. (this threshold is determined before testing from 100 validation images)

# Framework overview

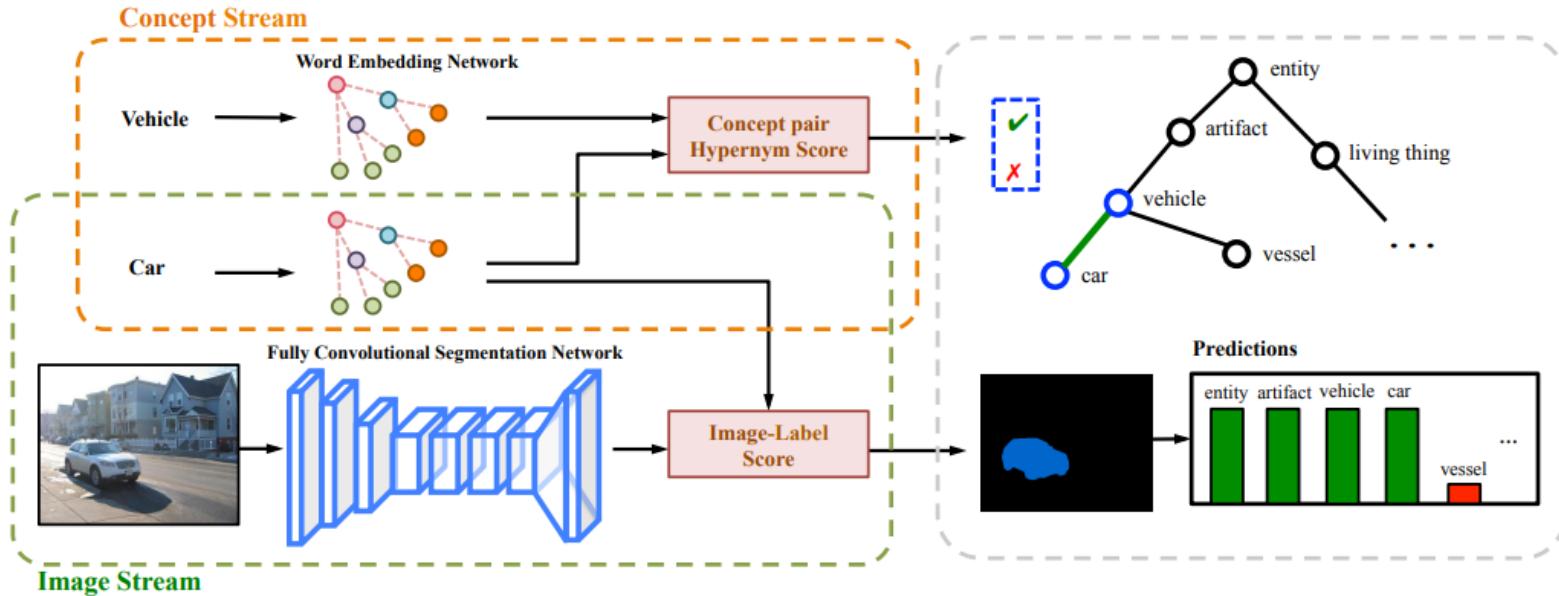


Figure 3. The open vocabulary parsing network. The concept stream encodes word concept hierarchy based on dictionaries like WordNet. The image stream parses images based on the learned hierarchy.

A **max-margin** loss is used to learn the embedding function  $f(\cdot)$  for mapping the concept space to joint embedding space.

They argue that since label retrieval is a ranking problem, negative labels should be introduced to push scores of positive labels to be larger than those of negative.

Initially, they use a **max-margin** loss for learning the mapping  $g(\cdot)$  from pixel feature space to the joint embedding space, but find that using **softmax** in the form of a triplet loss performs better.

$$\mathcal{L}_{image}(x_{i,j}) = -\log \left( \frac{e^{S_{image}(f(y_{i,j}), g(x_{i,j}))}}{e^{S_{image}(f(y_{i,j}), g(x_{i,j}))} + \sum_{y'_{i,j}} e^{S_{image}(f(y'_{i,j}), g(x_{i,j}))}} \right)$$

where,  $x_{i,j}$  = pixel features of the  $(i, j)^{th}$  pixel

$y_{i,j}$  = label of the  $(i, j)^{th}$  pixel

$y'_{i,j}$  = negative labels for the  $(i, j)^{th}$  pixel

Their '**Image Stream**' uses an adapted version of VGG-16 (to make the embeddings have a dimension equal to the word concept embeddings).

Also, (in the latent space), they fix the norms of the image embedding pixels to 30 to improve numerical stability (since pixel embeddings are the most specific concept in the joint embedding space).

The '**Concept Stream**' is trained first and the trained word embeddings are used as initializations for training loop.

## Metrics

They use standard metrics (per-pixel accuracy, mean accuracy, mIOU, weighted IOU),  
alongwith

1. open vocab metrics like **hierarchical precision, recall and F-score** which depends on the **depth of the word concept** in the whole concept map.
2. Information content ratio: defined as  $-\log(\text{probability})$ , (probability is taken as the **frequency of that concept and its hyponyms** in the whole dataset)

## Results/Conclusion

**Supervised:** They were not able to beat the baseline score of multi class classification using the same CNN (**Softmax**).

Interestingly, another baseline (**Conditional Softmax**) which was specifically designed for hierarchical classification was also less than **Softmax**.

Only standard metrics (accuracy, mean accuracy, mIOU, wIOU) were used to compare models.

**Zero-shot:** Here, however, they were able to consistently perform better than the baselines.

They also find that using the **asymmetric** scoring function gives a significant improvement.

Only hierarchical metrics and information content ratio were used for comparisons here.

Qualitatively, they show that in places where the model is **unsure** of the specific object, it correctly predicts a more **general concept**.

*For example*, in a rocking chair, the top part looks like a **chair** so it classifies that correctly, but the bottom part is **not like a normal chair**, and since it hasn't seen that particularly, it classifies it as '**furniture**', which is plausible and human-like.

They also do a ‘concept search’ in the embedding space to show that though baseline models can learn specific objects equally well, when **more abstract terms are searched** for in the joint embedding space, their **model is still able** to detect them in images whereas **baseline models aren’t**.

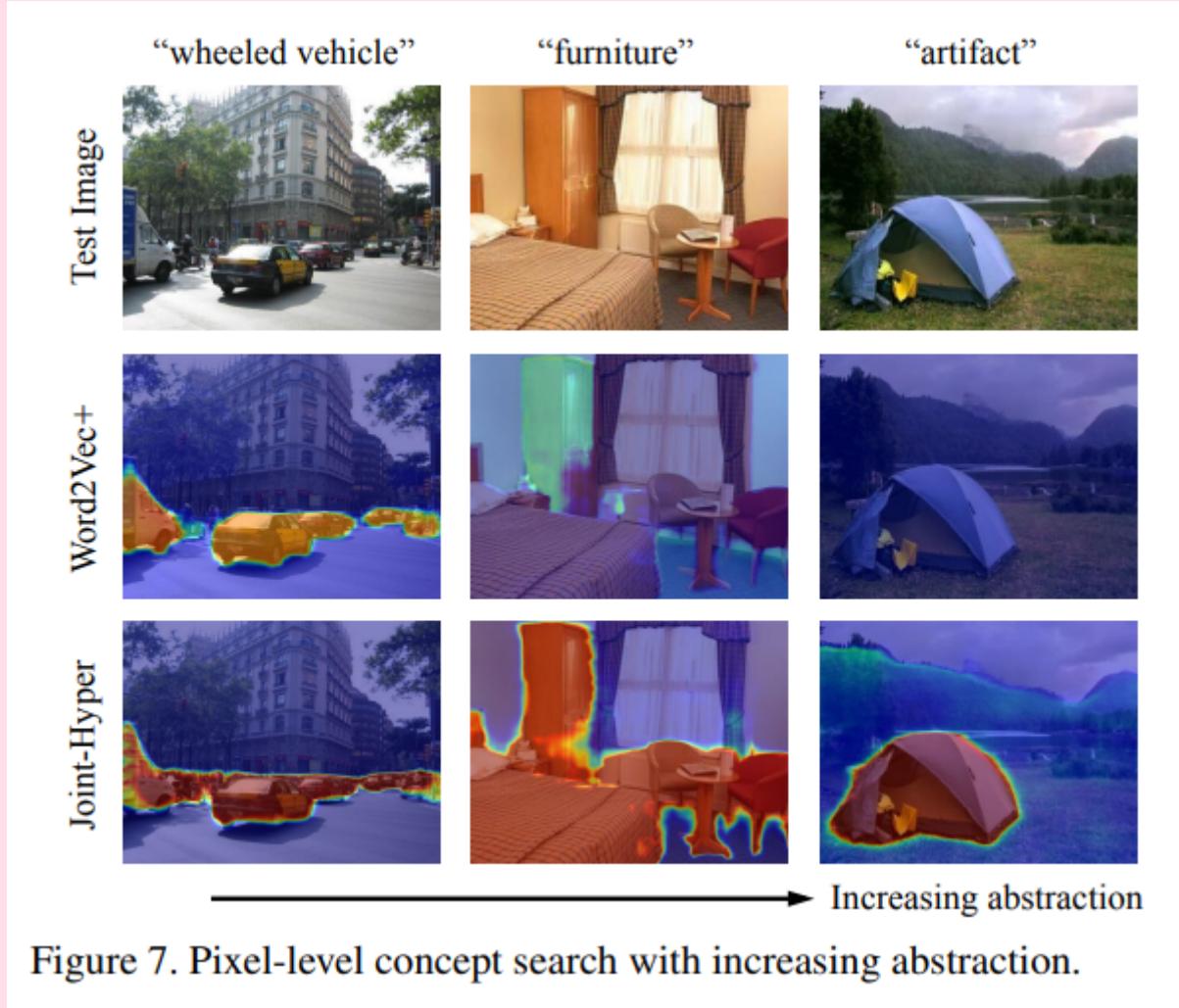


Figure 7. Pixel-level concept search with increasing abstraction.

They also show that because objects like **chair** and **bench** are close in the joint embedding space, so by looking in the vicinity of **chair**, they hypothesize that they will find **sittable** objects.

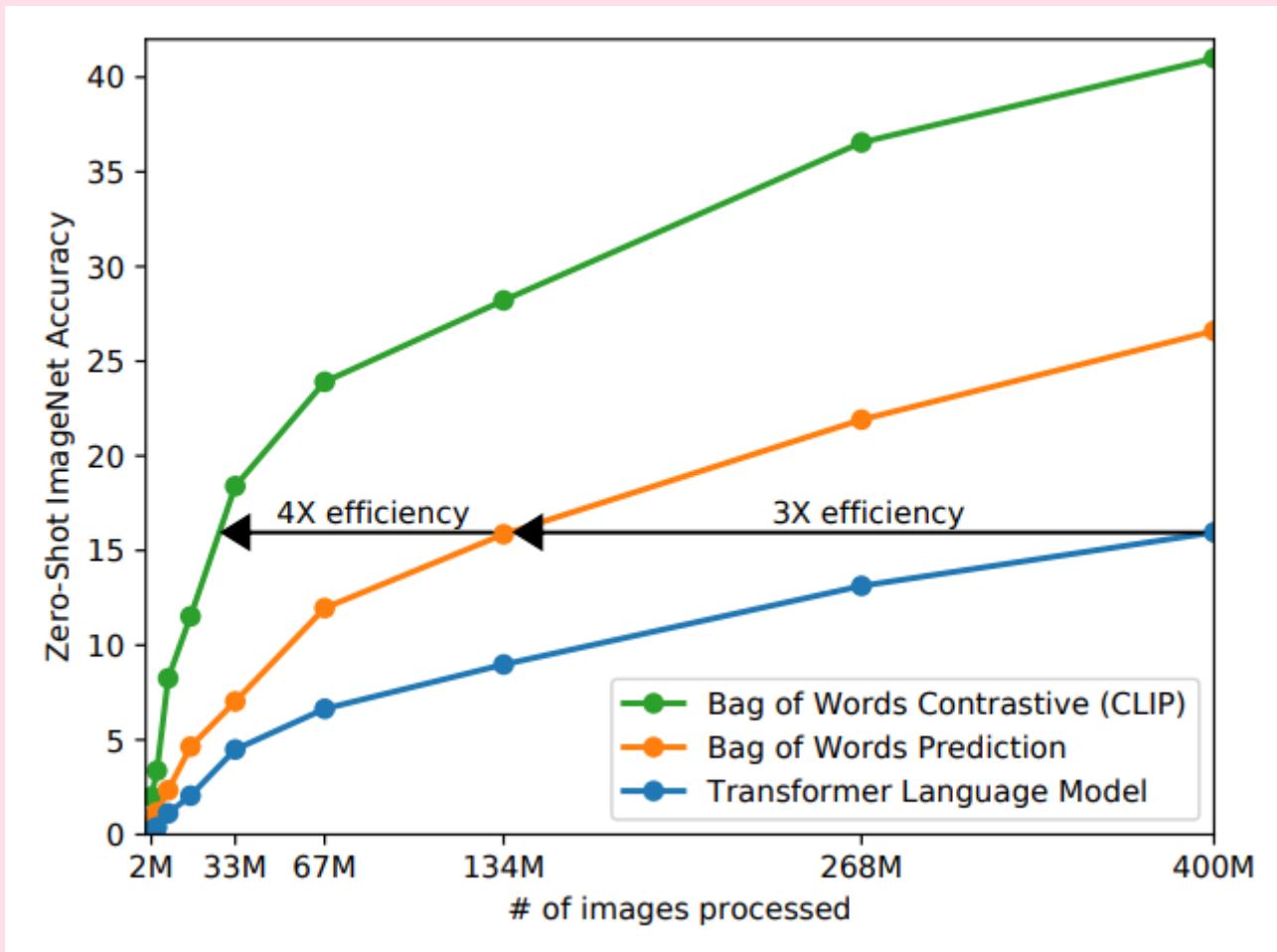
End.

# CLIP (Contrastive Language-Image Pretraining)

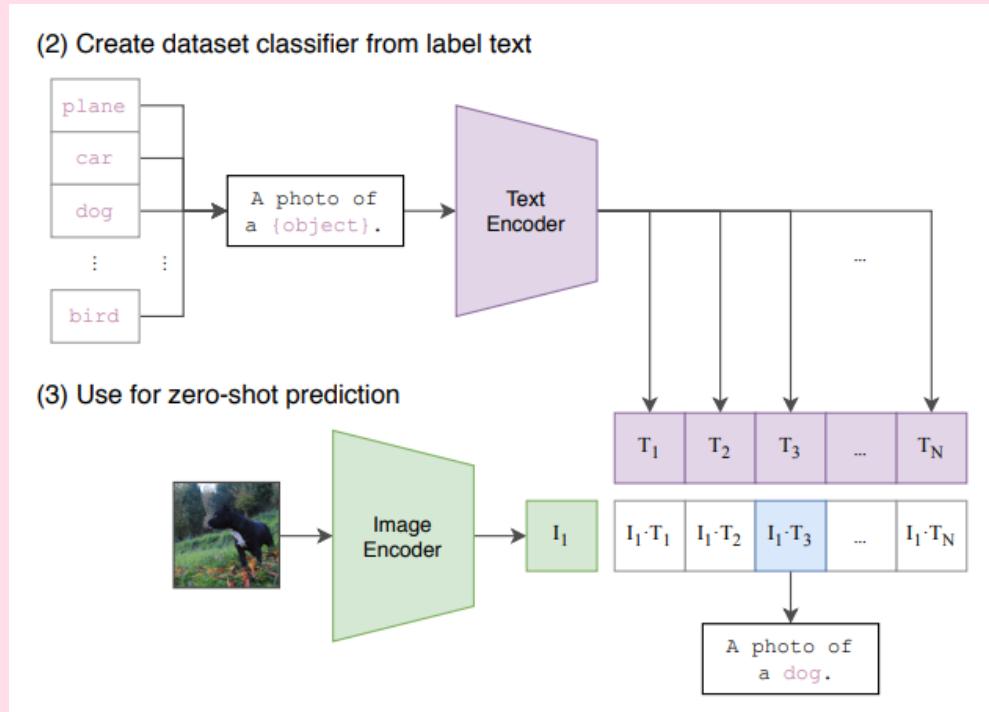
## What is it

They show that transformers are not good at zero-shot learning. So, they improve it by employing a bag-of-words objective and employ a contrastive objective, showing improvements over simply predictive objective.

They pretrain a large scale model that can perform multiple tasks.



# Contrastive Objective

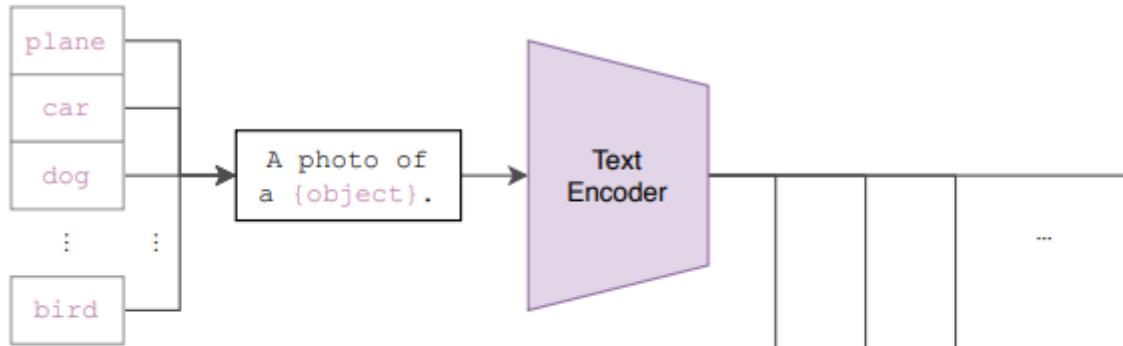


In a batch of  $N$  (image, text) pairs, they take all possible pairings of images and text ( $N^2$ ) and train CLIP to predict which out of those possible pairings actually occurred.

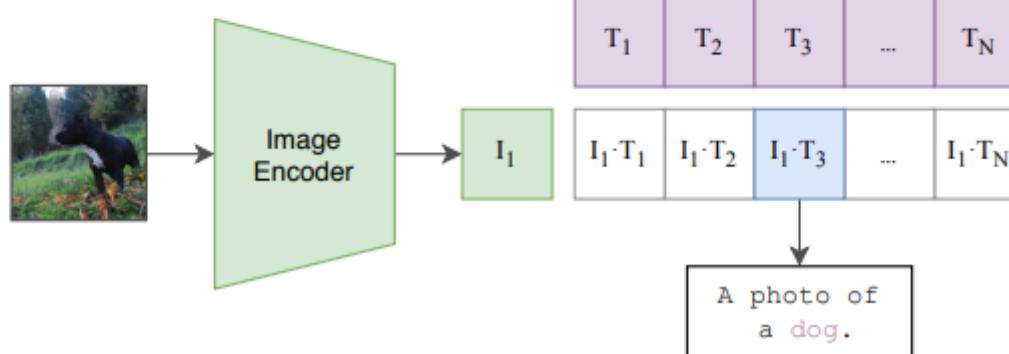
They do this by maximizing the agreement (via cosine similarity) of the  $N$  correct pairs, and, pushing away/reducing agreement between the  $N^2 - N$  negative pairs.

# Framework Overview

(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



They train **CLIP** from scratch on their **WebImageText** dataset containing ~400 million images.

**Image Encoder:** Because of the wide variety of architectures and designs available, they ended up choosing two architectures.

- One is based on **ResNet50**, with modifications to the layers and **replacing global average pooling with 'attention pooling'**. They mention  
*'transformer-style' multi head QKV attention, query is conditioned on the global average-pooled representation of the image*
- The other one is based on the recent **Vision Transformer (ViT)**. They make only minor changes to this architecture.

- They argue that for the ResNet based encoders, increasing one dimension alone (either depth, width or resolution) is less beneficial than increasing all dimensions together (keeping the computing resources same).

**Text Encoder:** The text encoder is taken as a transformer with some previously published modifications.

They only scale the width of this encoder as they find that CLIP is less sensitive to the text encoder.

## Training Details

**ResNets:** They train 5 models (**ResNet50, ResNet101, "Efficient-Net" style RN50x4, RN50x16, RN50x64**)

**ViT:** They train 3 models (**ViT-B/32, ViT-B/16, ViT-L/14**)

*The largest ResNet model, **RN50x64**, took 18 days to train on 592 V100 GPUs while the largest Vision Transformer took 12 days on 256 V100 GPUs.*

## Zero-shot performance

They use this pre-trained (on WebImageText dataset) CLIP model and test the zero-shot transfer ability on other CV datasets like ImageNet, aYahoo and SUN, showing a significant improvement above Visual N-Grams.

Also, they test it against a fully supervised logistic regression trained on the features of ResNet50 and beat it on 16 out of 27 datasets. They note that CLIP performs worse in more specialized datasets like satellite images, lymph node tumors, traffic sign recognition, etc.

Further, they also compare their zero shot results with few-shot linear probes and show that they outperform them.



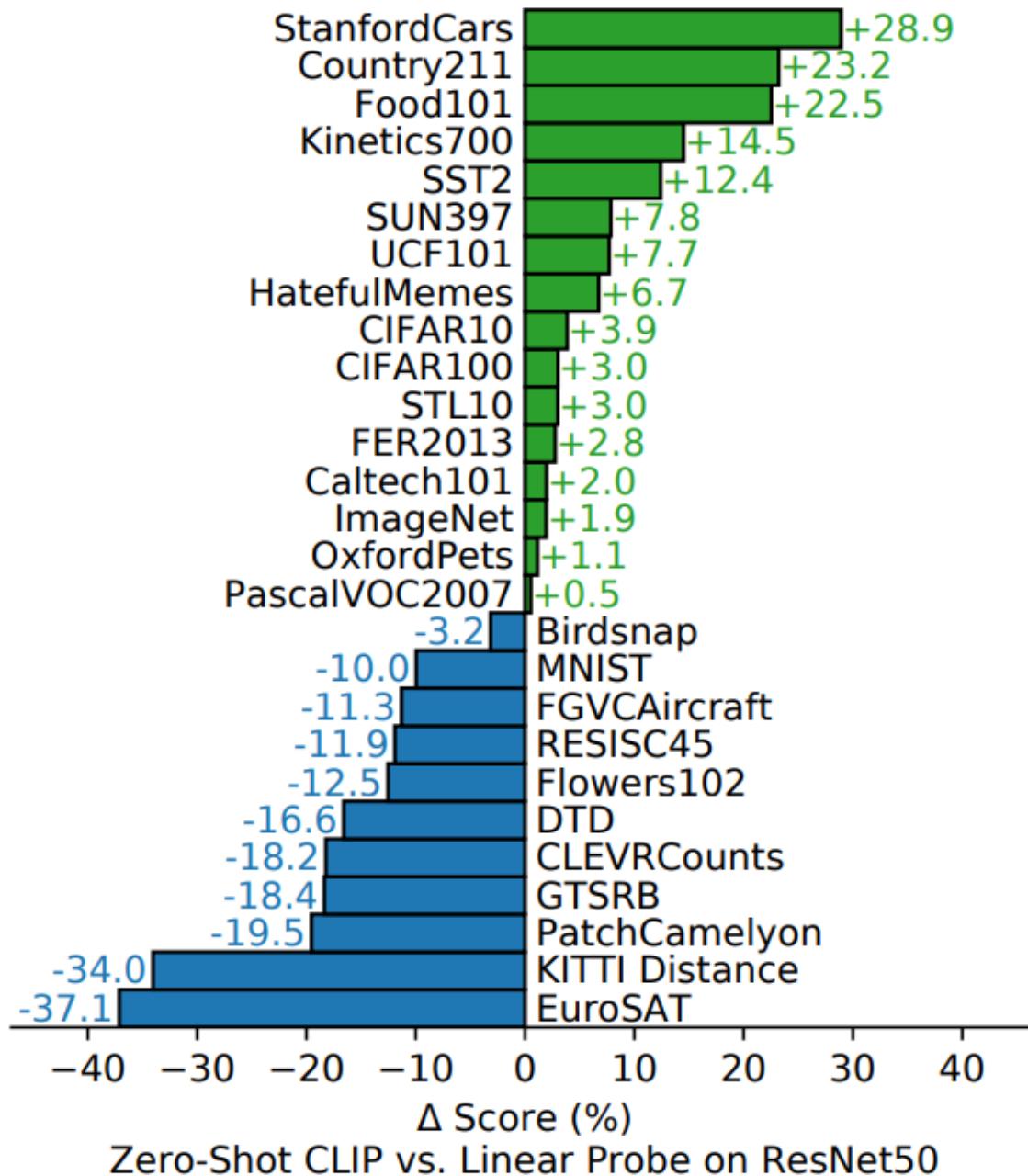


Figure 4. Zero-shot CLIP is competitive with a fully super-

## Discussion

They talk about natural distribution shift and deep models exceeding accuracy on ImageNet, while in reality, more robust/better metrics show that that is not the case.

They also use **Effective robustness** and **Relative robustness**, which are made to measure improvements in accuracy under distribution shift, and out-of-distribution accuracy respectively. They also argue that because a zero-shot model cannot exploit the patterns of a specific dataset/ distribution, they empirically have more **effective robustness** than few shot models.

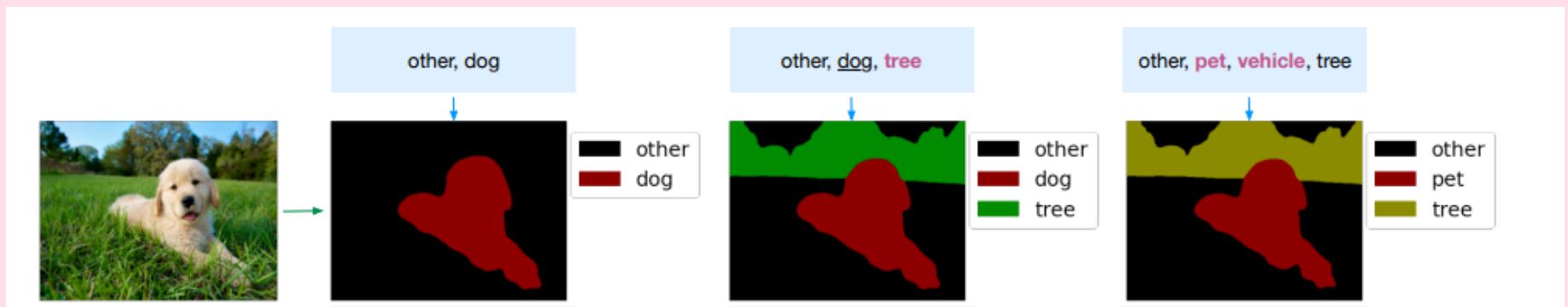
They showed that the overlap in the datasets was also very low (average 3.2%), and the maximum improvement in accuracy is only 0.6%, which is in line with other large scale pre-trained models.

Other than this, they briefly talk about the societal impact and privacy/risk implications because of **CLIP** etc.

# LSeg : Language-Driven Semantic Segmentation

Problem setting: Zero-shot segmentation

One-line approach: Use the text encoder of models like CLIP, train a separate visual encoder to produce pixel embeddings close to the label embeddings in a joint embedding space.



Advantage: flexibility, i.e. being able to segment different classes within the same image given a different label set. (It can also segment with a label that is close to another label in the embedding space, i.e. given **pet** as a label, it classifies the **dog** as **pet**)

# Framework

They use only the text part of CLIP, discarding the image encoder and training their own image encoder architecture based on **Dense Prediction Transformers** (DPT).

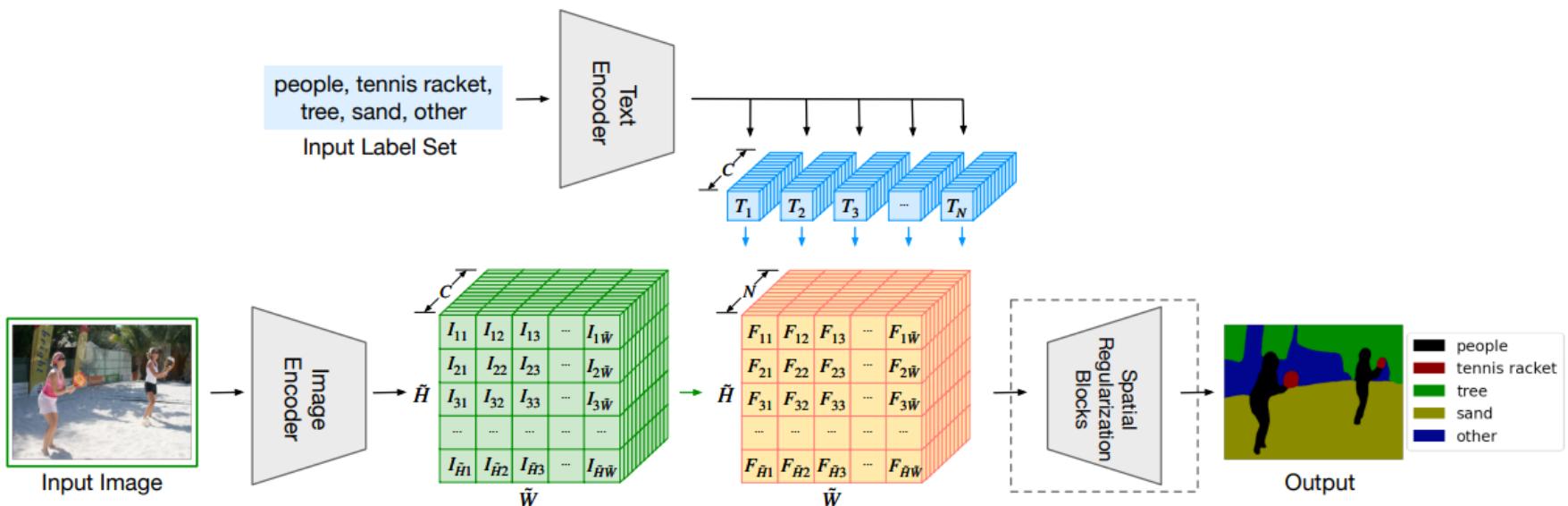


Figure 2: Overview. A text encoder embeds labels into a vector space. An image encoder extracts per-pixel embeddings from the image and correlates the feature of each pixel to all label embeddings. The image encoder is trained to maximize the correlation between the text embedding and the image pixel embedding of the ground-truth class of the pixel. A final spatial regularization block spatially regularizes and cleans up the predictions.

$F$  is calculated as the dot product of the image embeddings  $I$  and label embeddings  $T$ .

$$F_{i,j,k} = I_{i,j} \cdot T_k$$

*dimensions :  $I_{i,j} \in \mathbb{R}^C$ ,  $\{i, j\}$  represent pixels*

$$T_k \in \mathbb{R}^C, k \in \{1..N\}$$

$$F_{i,j} \in \mathbb{R}^N$$

So, they want to maximize the dot product  $F_{i,j,k}$  for those pixels  $\{i, j\}$  where  $y_{i,j} = k$  (GT label). They do this by applying softmax over  $k$  on  $F_{i,j,k}$  and taking a **CrossEntropy** loss.

For the final step, The softmaxed feature block  $F$  (equivalent to predictions) is then ‘spatially regularized’ using a **DepthwiseBlock**(Depthwise Conv) or a **BottleneckBlock**(Depthwise Conv augmented with max-pooling), and is upsampled to the input image’s resolution using bilinear interpolation.

**Training Details:** They use pretrained weights on **ImageNet** for **ResNet** and **ViT** image encoders, and take random initialization for **DPT**. They freeze the text encoder(the **ViT-B/32** from **CLIP**) while training.

They show results that are comparable with 1-shot state-of-the-art(**HSNet**) results, and significantly higher than previous zero-shot models on **PASCAL-5i** and **COCO-20i**. They outperform **HSNet** on **FSS-1000**.

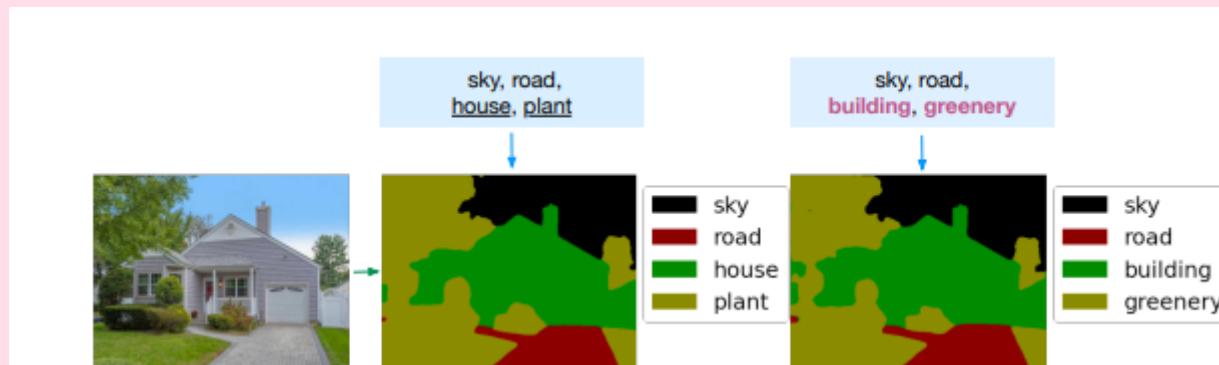
They use different text encoders from CLIP and compare them. (The text encoder is always a simple Transformer, the difference is the image encoder it is co-trained with in the CLIP pretraining step).

Method	Backbone	Text Encoder (fixed)	embedding dimension	pixAcc [%]	mIoU [%]
LSeg	ViT-B/32	ViT-B/32	512	79.70	37.83
LSeg	ViT-B/32	ViT-B/16	512	79.77	38.69
LSeg	ViT-B/32	RN50 × 4	640	79.85	38.93
LSeg	ViT-B/32	RN50 × 16	768	<b>80.26</b>	<b>40.36</b>

# Qualitative Analysis

## Related but unseen labels

They show that **LSeg** is able to predict objects belonging to unseen classes close to the points in embedding space.

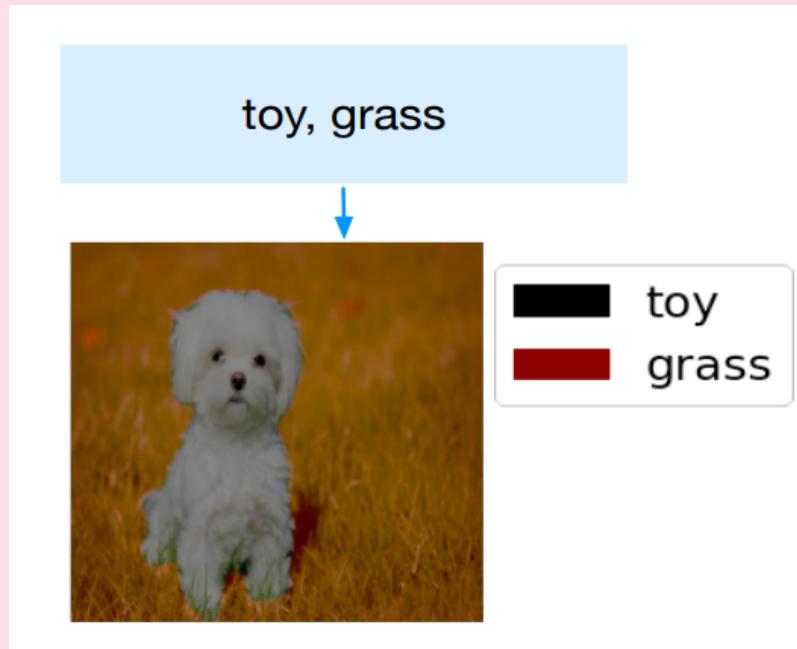


They show the same behavior with hierarchical unseen labels (i.e. being able to predict correctly when a parent category is present in label set instead of the specific object).

## Failure Cases

They mention that since LSeg is trained only on positive examples of classes (unlike CLIP which had a contrastive objective), it can give wrong predictions sometimes.

For example,



In this image, it predicts the dog as toy (when only toy and grass are provided) because a dog is probably closer to a toy than grass visually and semantically.

# RegionCLIP

Extract regions and their text descriptions from images and use language-image training similar to CLIP on these (contrastively).

## Need?

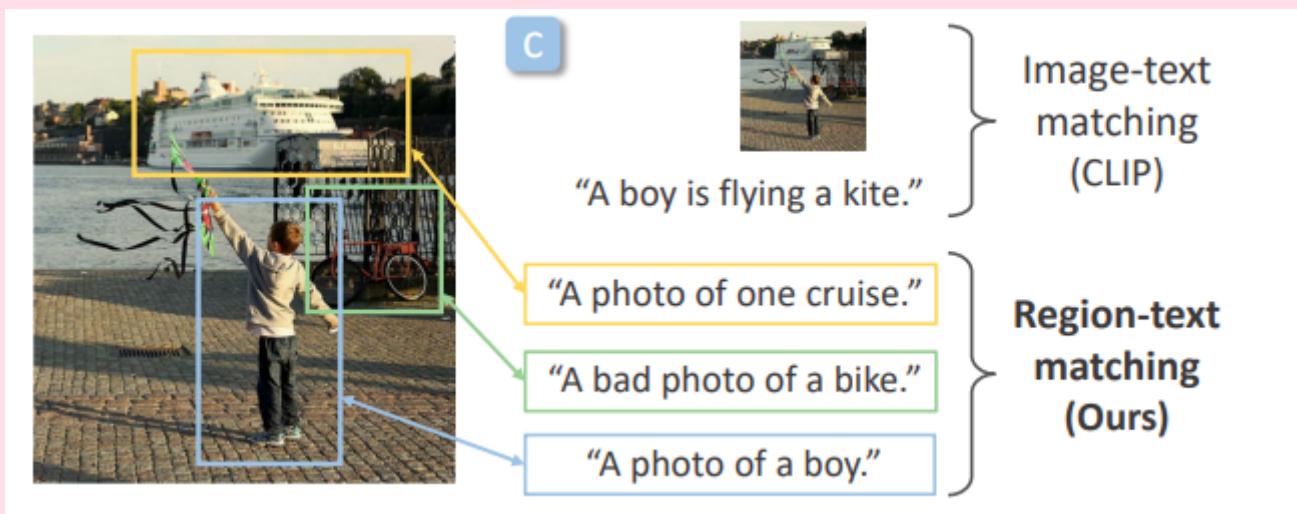
Acc. to authors, we cannot directly apply CLIP to regions and have it work well because there is a major domain shift (?) and thus has unsatisfactory performance. This is because CLIP is trained to match an image with its image-level description, and does not know about the alignment between local image regions and text descriptions of those regions.

## Problems:

1. Fine grained alignment between image regions and text is not usually available, expensive to annotate.
2. Image-level descriptions might leave out the description of some objects in the image.

## Solution:

Bootstrap from a pretrained language-vision model (CLIP) and fill in the missing region descriptions and then align them with proposed regions based on a metric.



# Framework

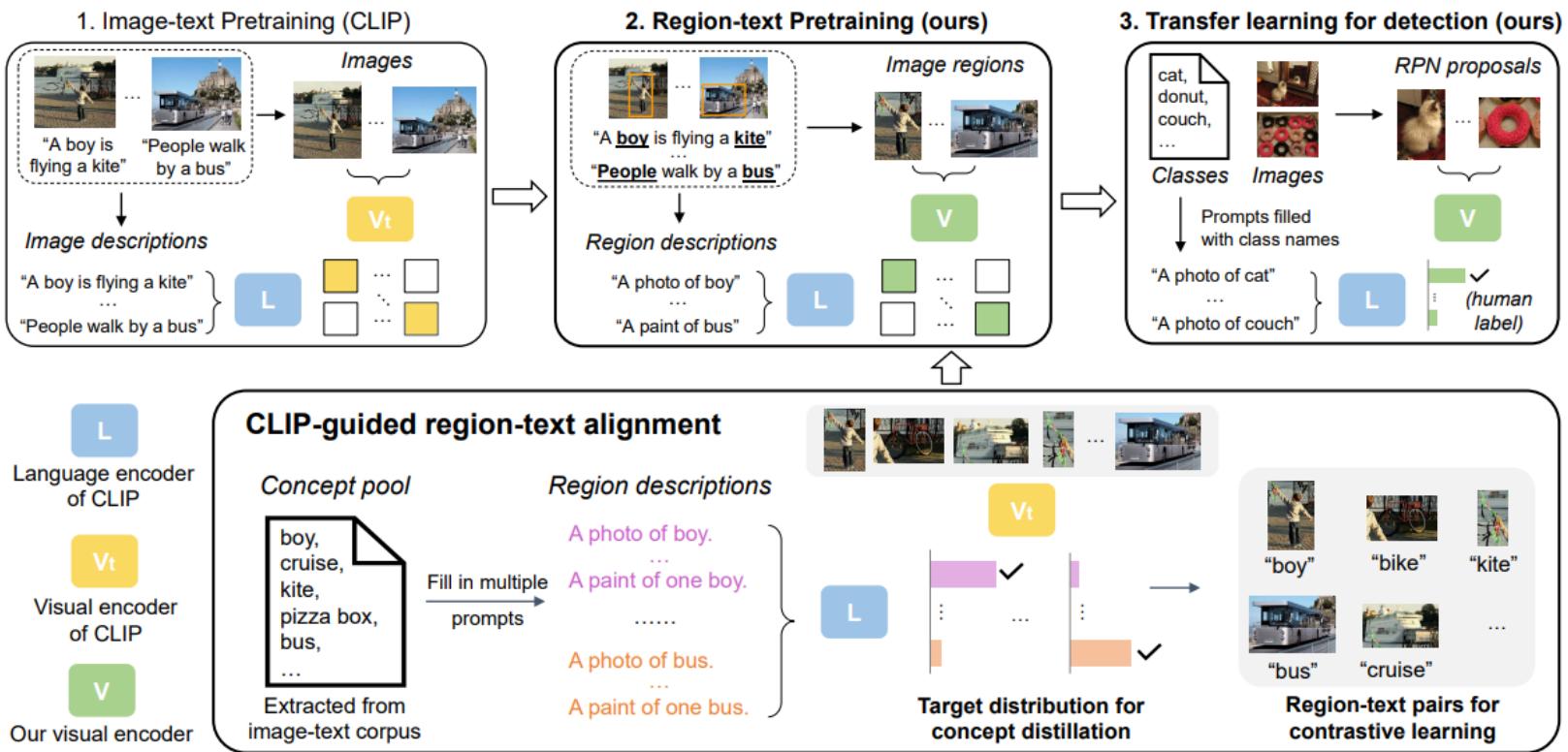


Figure 2. Method overview. We propose to learn visual representation for image regions via vision-language pretraining. Panel 1: With contrastive learning, CLIP is able to match images and their descriptions. Panel 2: Initialized by pretrained CLIP, our visual encoder learns visual region representation from the created region-text pairs. Specifically, as shown in the bottom row, we first create texts by filling the prompts with object concepts which are parsed from image descriptions, then use pretrained CLIP to align these texts and image regions proposed by RPN. Panel 3: When human annotation for image regions is available, we transfer our visual encoder for object detection.

They make region descriptions by filling ‘object concepts’(from concept pool) into prompts and then, using a **teacher** model  $\mathcal{V}_t$  (from **CLIP**), and sees which region (proposed by the Region Proposal Network, **RPN**, pretrained) aligns with the region description the most, and assigns it to that.

Once these region-text pairs are generated, the new encoder can be contrastively trained on these, similar to **CLIP**’s contrastive language-image pretraining.

They use **RoIAlign** to extract the region’s visual features from the encoder  $\mathcal{V}$ , which pools regional features from the image’s feature map using interpolation.

$\mathcal{V}$  takes initial weights from  $\mathcal{V}_t$  for a good start in the visual-semantic space.

## Details

The CC3M (contextual captions dataset) was used for training.

The region descriptions are made by filling the concepts from concept pool into prompts, i.e. kite is filled into the prompt a photo of a .... to make the description a photo of a kite. These are then passed through the pretrained language encoder (CLIP) to get the semantic text embedding.

Cosine Similarity is used as the metric of how much the region proposed aligns with some region description for the contrastive loss between region-text pairs

$$L_{cntrst}.$$

## Losses

They use a distillation loss  $L_{dist}$  in addition to a contrastive loss, which is defined as:

$$L_{dist} = \frac{1}{N} \sum_i L_{KL}(q_i^t, q_i)$$

where,  $q_i^t$  is a 'soft target' =  $\text{softmax}_j(\text{distance}(v_i^t, l_j))$ ,  
 $v_i^t$  is region's visual features from teacher  $\mathcal{V}_t$   
and,  $v_i$  is region's visual features from  $\mathcal{V}$

They also added the contrastive loss at image level  $L_{cntrst-img}$  (with negative samples being labels of different images), like CLIP to their final loss. So, final loss is

$$L = L_{cntrst} + L_{dist} + L_{cntrst-img}$$

# Extensions to object detection and open-vocabulary object detection

They extend this framework to object detection by simply using the **RPN** to generate regions and find which one matches the target object class the most, and simple output that as the localization/bounding box for the object. However, no work is done in segmenting the target object.

For openvoc object detection, they evaluate the model on 48 base and 17 novel categories for **COCO** and 866 base and 337 novel categories from **LVIS** (general classes are termed as base and specific object classes are termed as novel).

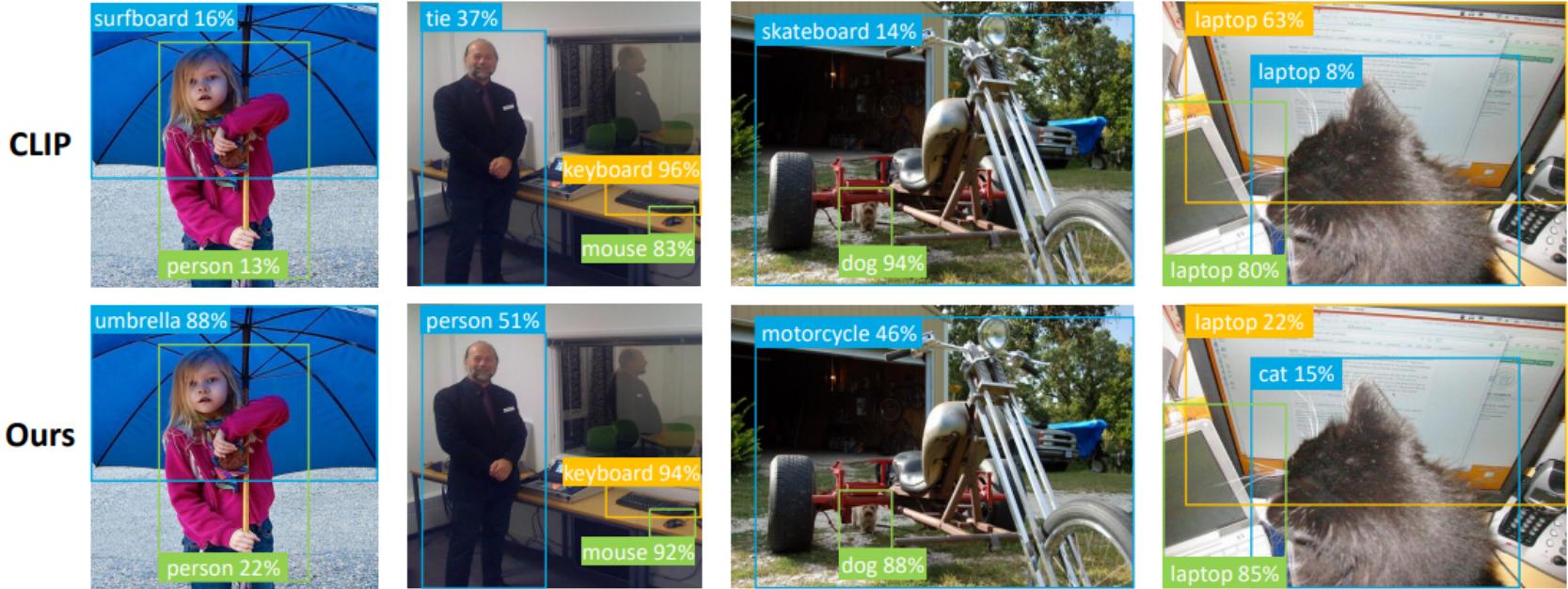
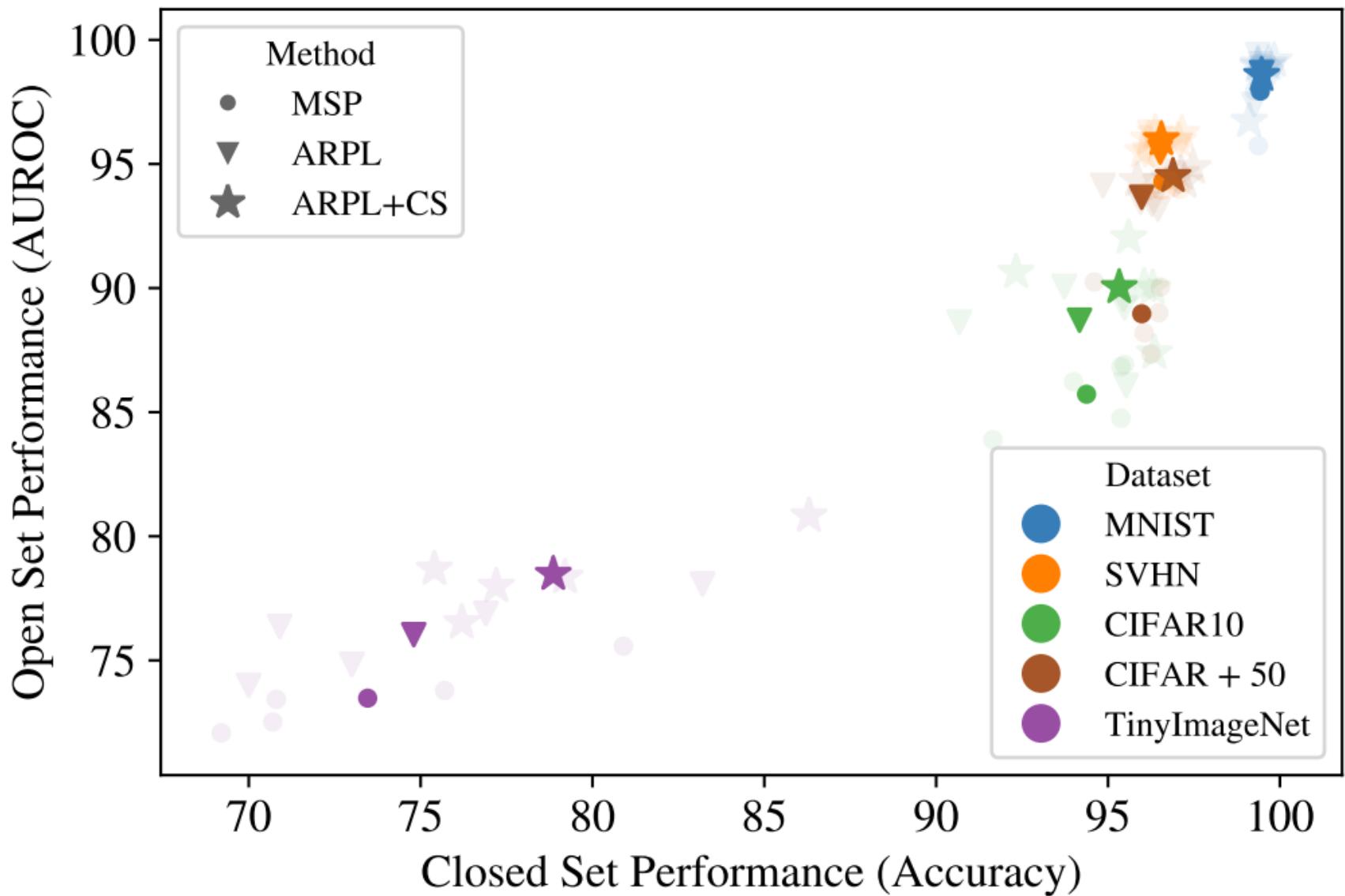


Figure 3. Visualization of zero-shot inference on COCO dataset with *ground-truth boxes*. Without finetuning, the pretrained models (top: CLIP, bottom: Ours) are directly used to recognize image regions into the categories in COCO. (Image IDs: 9448, 9483, 7386, 4795)

# OPEN-SET RECOGNITION: A GOOD CLOSED-SET CLASSIFIER IS ALL YOU NEED?

ICLR '22

They show that the closed set accuracy is *highly correlated* to the open set performance.



- Performed multiple experiments using a variety of models: **ViT**, **ResNet**, **EfficientNet**, **VGG**.
- **ViT** doesn't overfit its representation to the training classes and outperforms other methods.

Good closed-set performance => Better **OSR**

To enhance the closed set performance, they leverage **existing techniques from image recognition**:

- label smoothing
- longer training times
- better augmentations
- better LR schedules

They also try changing the open set scoring rule to Maximum Logit Score (MLS).

Using MLS gives better performance in OSR but softmax normalization is better in combined (OSCR) (because softmax normalization cancels the effect of the feature norm)

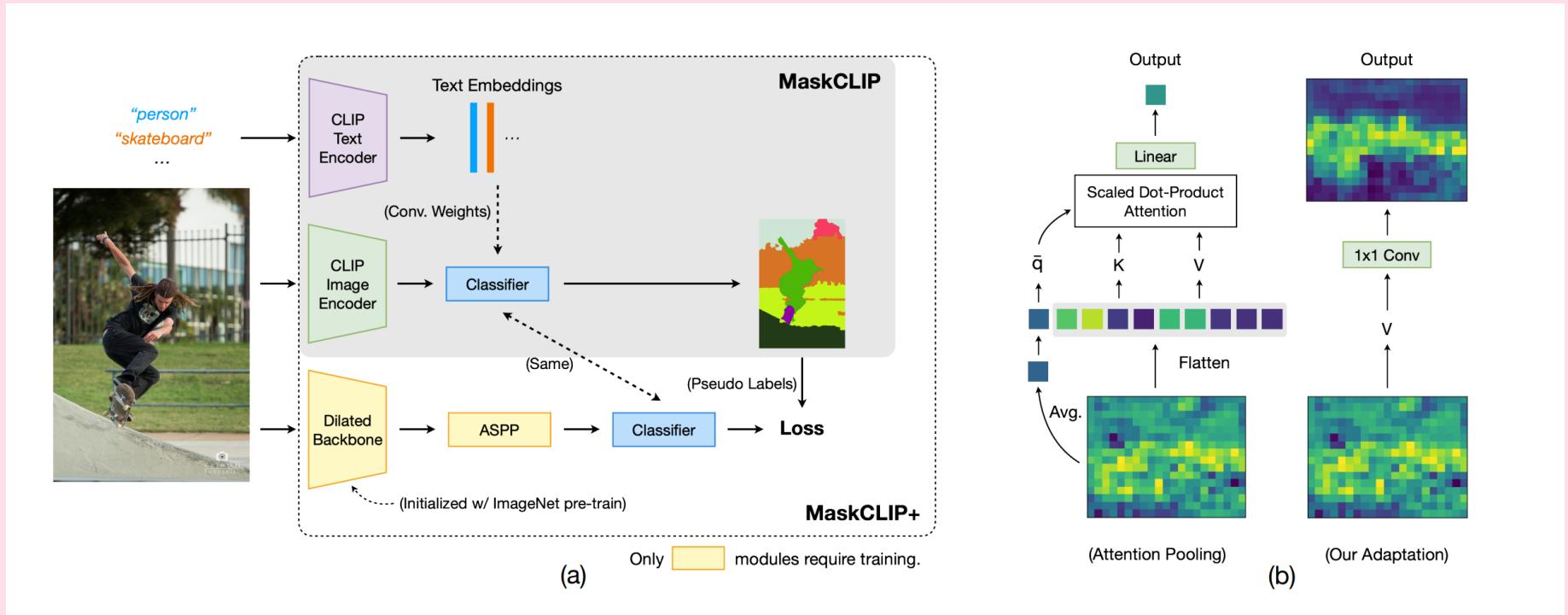
# Extract Free Dense Labels from CLIP

ECCV '22

# Using CLIP features for dense prediction

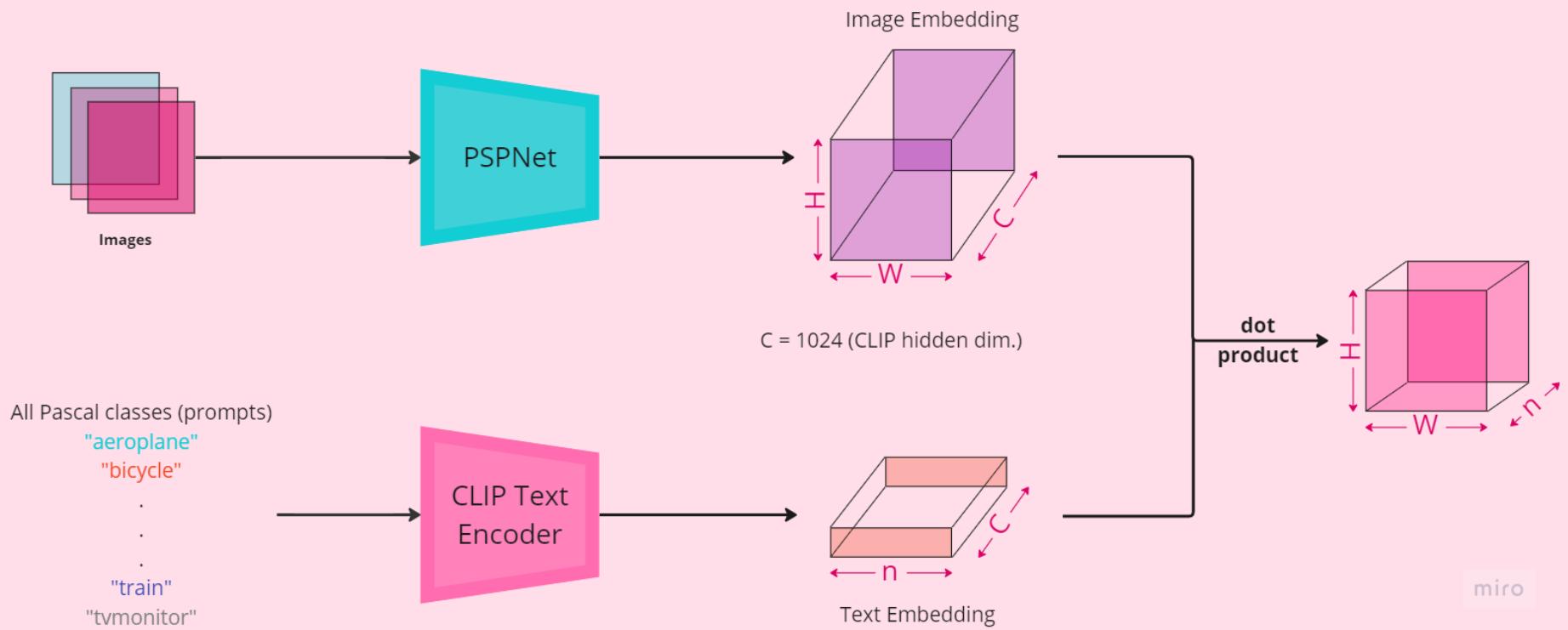
- **Failure:** Fine-tuning the image encoder of CLIP for segmentation tasks.
- Performance is good on seen classes but modified DeepLabv2 in conjunction with CLIP's text fails to segment novel classes.
- Reasons:
  - The visual-language association of CLIP features should remain intact for best performance.
  - Loss of generality => Additional mapper trained on seen classes.

# MaskCLIP

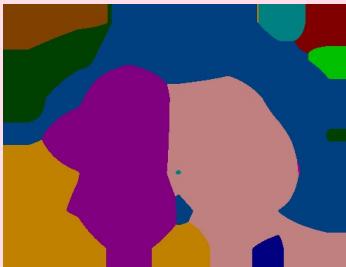


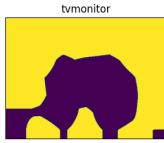
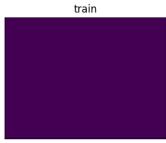
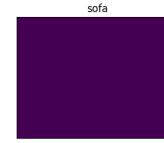
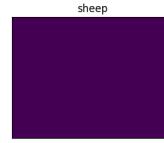
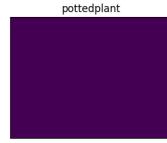
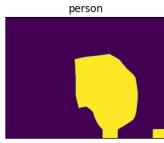
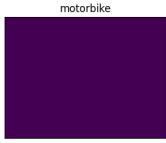
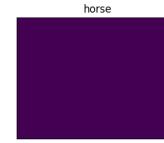
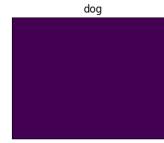
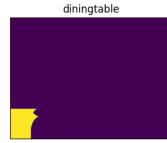
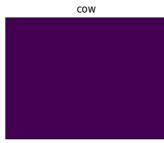
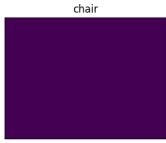
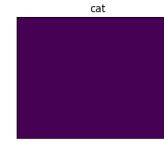
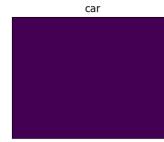
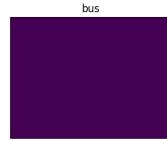
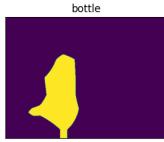
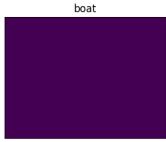
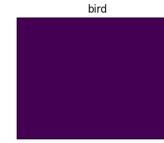
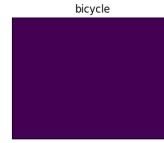
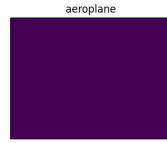
Doesn't modify the CLIP feature space

# Comparative Analysis between MaskCLIP and Our Results

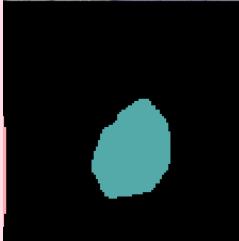
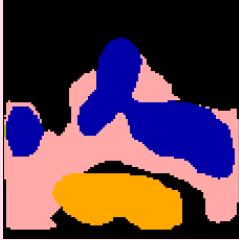
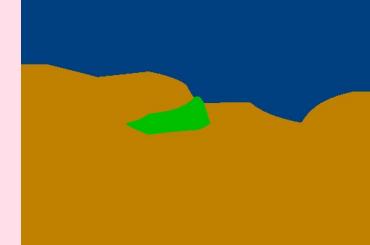
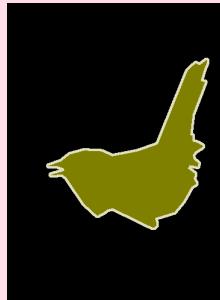
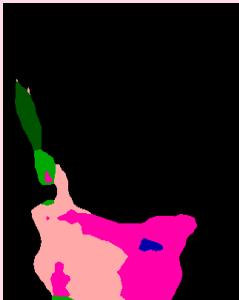
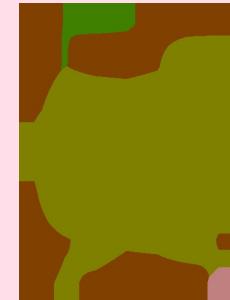


# Base Class Performance

Image	Ground Truth	Ours (PSPNet)	MaskCLIP (w/o PD and KS)	MaskCLIP (w/ PD and KS)
				
				



# Novel Class Performance

Image	Ground Truth	Ours (PSPNet)	MaskCLIP (w/o PD and KS)	MaskCLIP (w/ PD and KS)
				
				
				

<b>Class</b>	<b>IoU</b>	<b>Acc</b>	<b>Prec</b>
aeroplane	90.65	99.87	90.75
bicycle	55.04	94.25	56.95
bird	92.39	94.18	97.98
boat	52.58	94.06	54.38
bottle	56.82	83.66	63.92
bus	90.02	95.26	94.24
car	83.61	93.85	88.46
cat	84.9	87.19	97.0
chair	17.4	18.73	71.15
cow	53.38	64.41	75.72
diningtable	57.32	86.57	62.91

Class	IoU	Acc	Prec
dog	79.62	86.45	90.97
horse	59.05	96.59	60.31
motorbike	71.93	86.76	80.8
person	40.78	43.7	85.93
pottedplant	59.96	78.03	72.13
sheep	66.82	84.0	76.56
sofa	50.45	92.7	52.54
train	82.8	94.33	87.13
tvmonitor	64.51	91.8	68.45

Summary:

aAcc	mIoU	mAcc	mPrec
77.78	65.5	83.32	76.42

