Semi-Supervised Semantic Image Segmentation with Self-correcting Networks

Mostafa S. Ibrahim[1*], Arash Vahdat[2], Mani Ranjbar[3], William G. Macready[4]

[1]Simon Fraser University
[2]NVIDIA
[3]Sportlogiq
[4]Sanctuary AI
[*]Work done while interning at D-Wave Systems

CVPR SEATTLE
JUNE 14-19 2020 WASHINGTON

Content

- Fast Intro to Semantic Segmentation
- Our Semi-supervised Semantic Segmentation

Semantic Segmentation

Label **every** pixel!

Don't differentiate instances (cows)

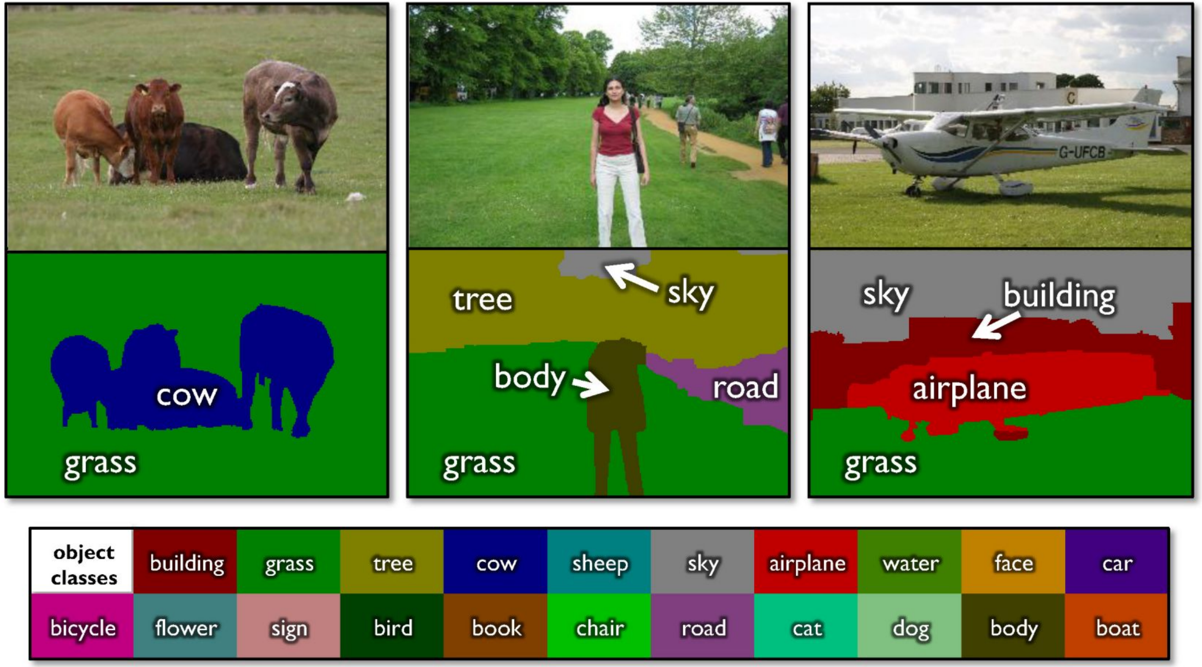


Figure credit: Shotton et al, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context", IJCV 2007

Semantic Segmentation: Apps

- Road Scene Understanding / Autonomous cars
- Editing images / Robots domain
- Medical purposes: e.g. segmenting tumours, dental

Image credit: [cityscapes dataset]
Apps credit: [Torr] Vision Group

Semantic Segmentation: Datasets

PASCAL VOC

- 20 classes
- 3.5k/1.5k trainval/test images
- Accurate Segmentation
- Evaluation Server
- 9k train **Aux**
  - missing segmented parts
  - under/over segmentations

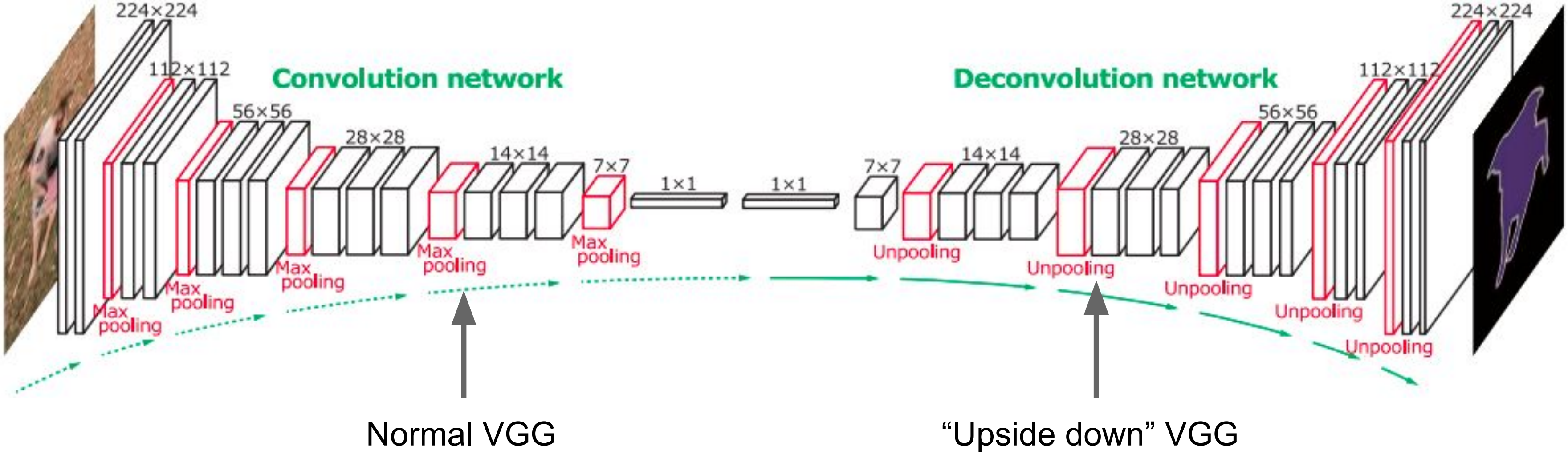
Image credit

Semantic Segmentation: Datasets

COCO

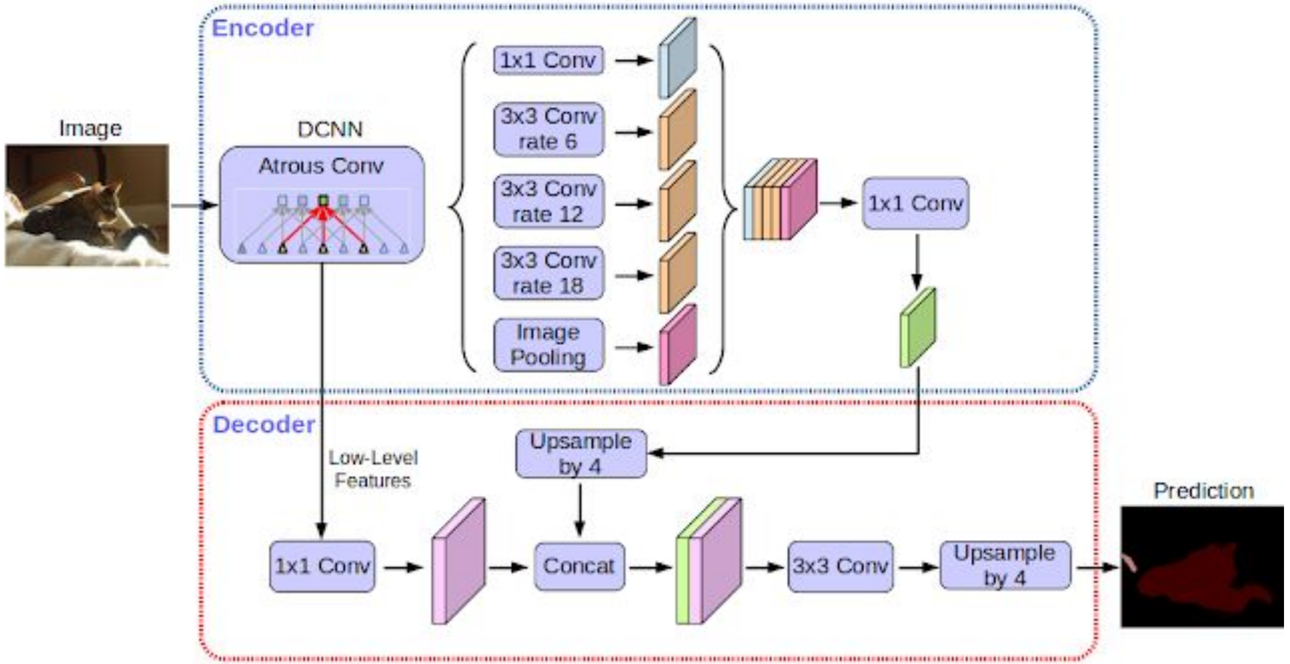
- 80 classes
- 300k images
- Not so accurate boundaries
- *Missing Segmented objects*

Semantic Segmentation: AutoEncoder Styles



Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic Segmentation: Deeplab v3+



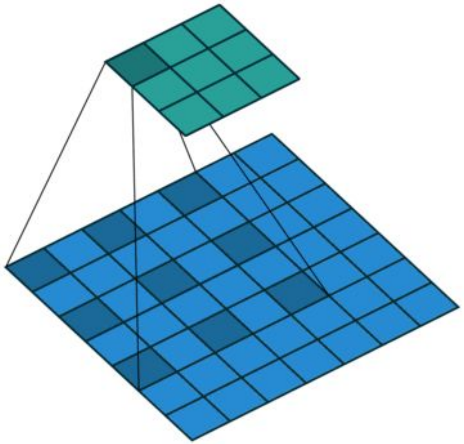


Image credit: [Google blog](#) [Vincent Dumoulin](#)

Semantic Segmentation: Loss Function

- Input is image x = (WxHx3)
- Output is map y = (WxH).
- Output per pixel is one of K classes
- On right: One hot encoding

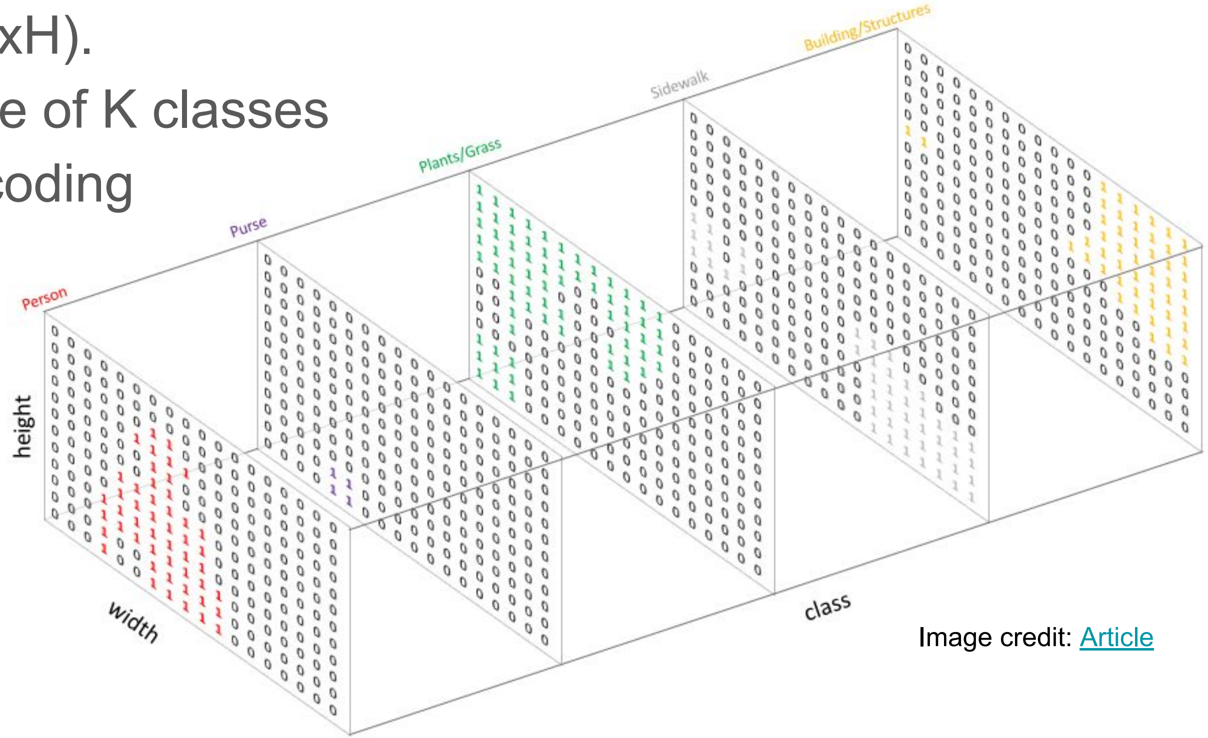


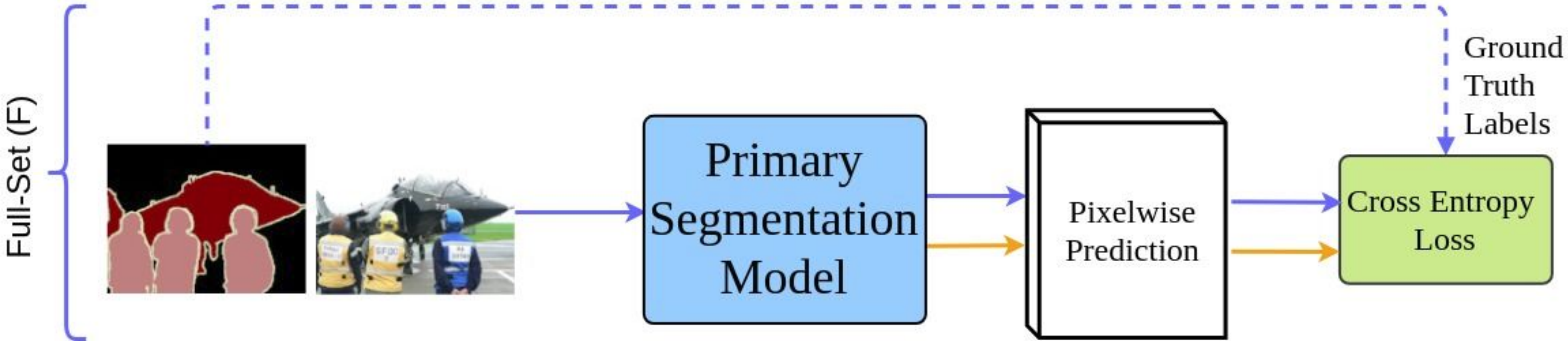
Image credit: [Article]

Semantic Segmentation: Loss Function

- Given x, our goal is to find map y that maximizes joint dist **$p(y|x;\theta)$**
- This will typically be interactable.
- To make it simpler, assume a **factorial distribution** over n pixels
  - $p(y \mid x) = \prod_{i=0}^{n} p(y_i \mid x)$

Semantic Segmentation: Loss Function

- Now, just compute cross entropy for every pixel **independently**
  - Typically Ground truth for CE is one-hot coding
  - *But it also can be any distribution over K classes*



Semantic Segmentation: Loss Function

- Network logits ⇒ Softmax activation ⇒ probability ⇒ CE(p1, p2)
  - tf.nn.softmax_cross_entropy_with_logits(labels, logits)

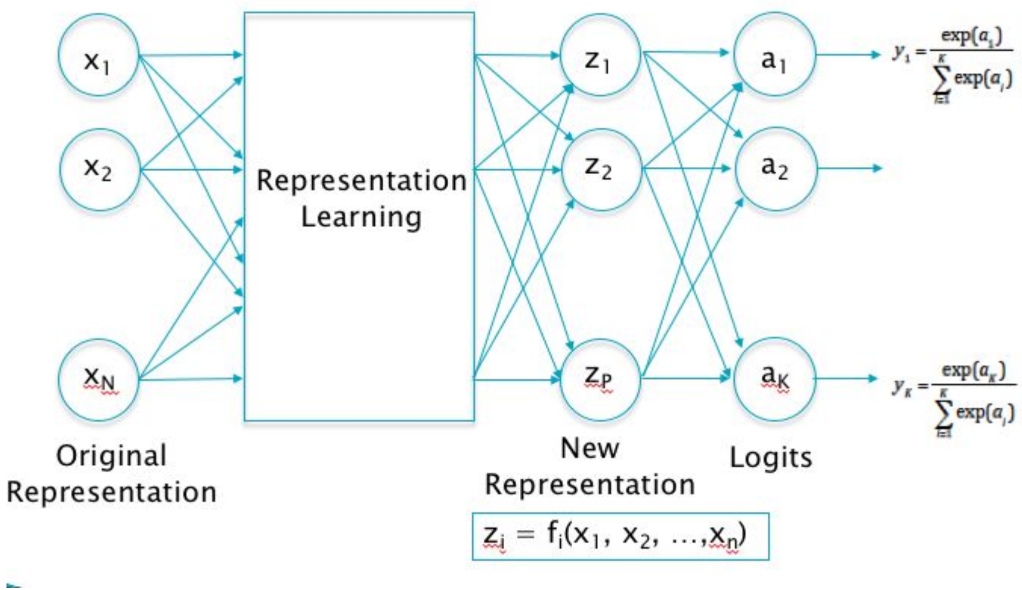
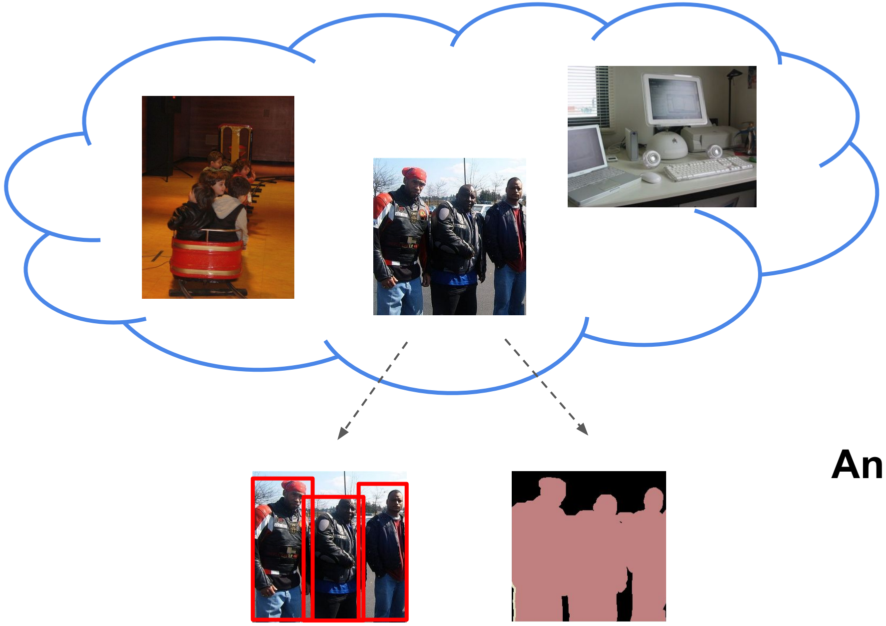


Image credit: Article

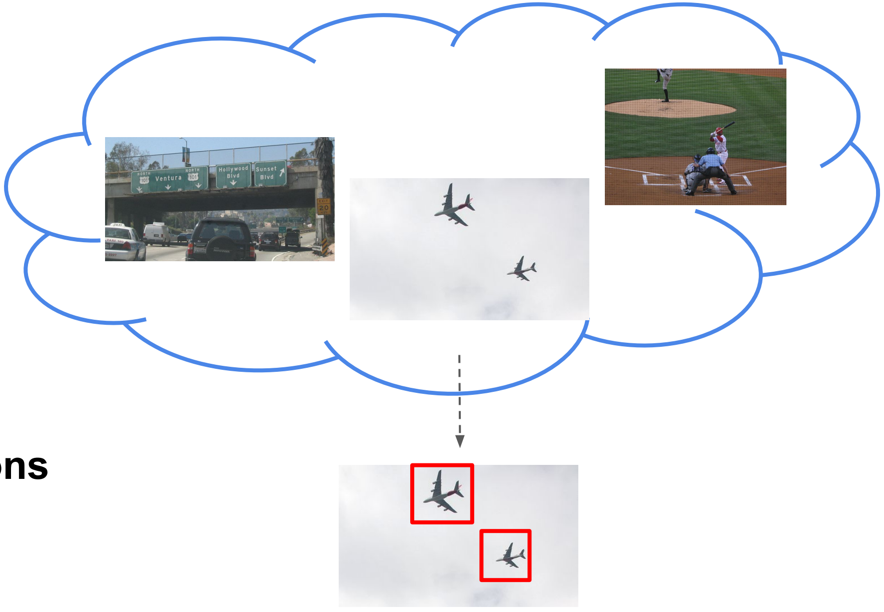
Our Semi-supervised setup

Fully labeled Set + **Weakly labeled Set**



Bounding Box   Segmentation

Annotations



Bounding Box

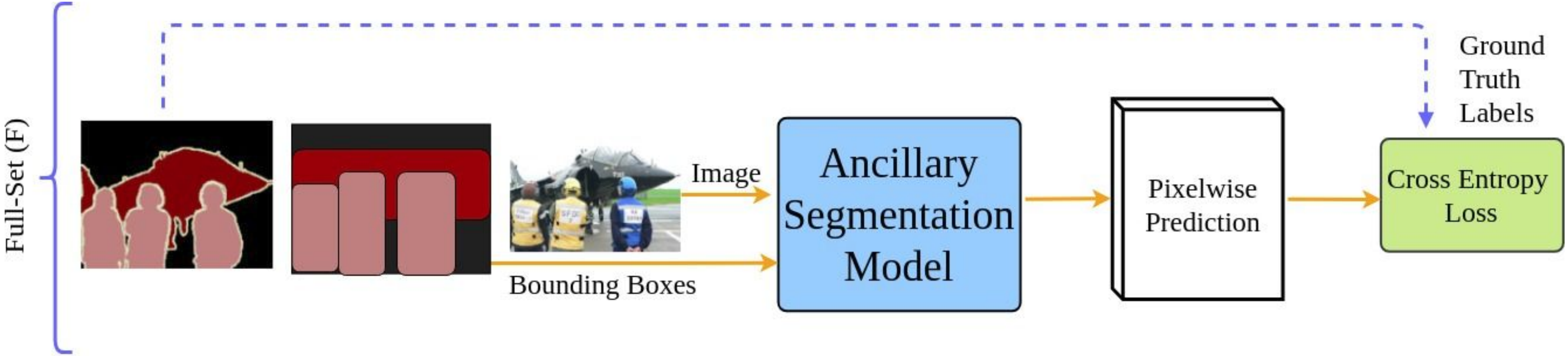
Framework Overview

- **Ancillary** segmentation network
  - Generate Initial segmentation (logits) for the weak-set
  - Network Input: **image & bounding boxes**
  - Network Output: segmentation labels (logits)
- **Primary** segmentation network
  - Standard Segmentation + Refine logits during training
  - **Self correction**: 2 approaches for refining logits

Ancillary Segmentation Model

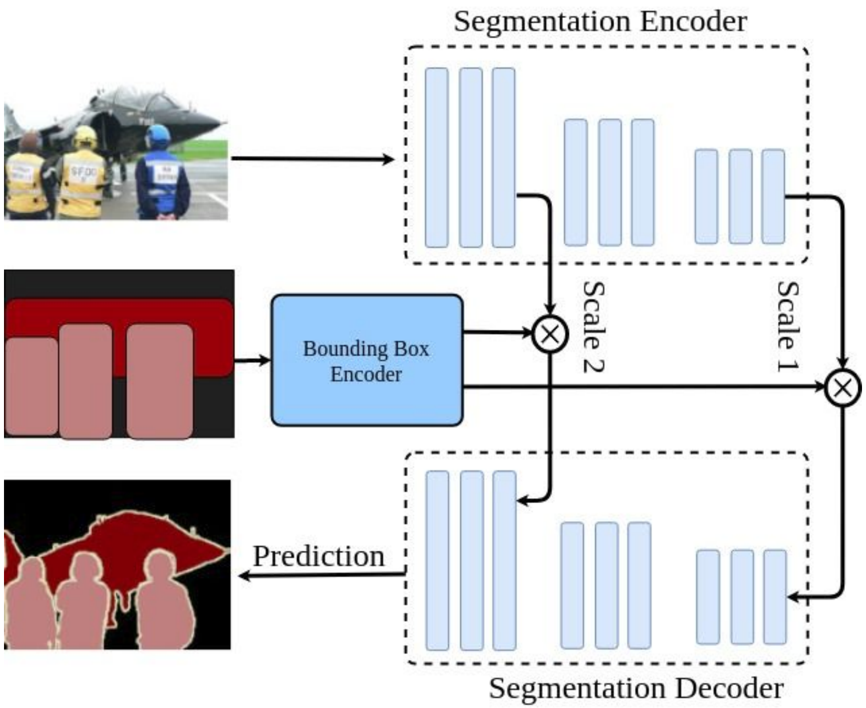
- Input: an image (x) and bounding boxes (b)
- Output: segmentation mask (y)
- Model: $\mathbf{p_{anc}(y|x, b)}$
  - Encoder-decoder-based segmentation network
    - Extra sub-network for encoding bounding boxes
  - Bbox representation is injected **after** the encoder
    - Inject on several scales
- This is very strong, as network knows more about ground truth

Ancillary segmentation model using the full-set



Ancillary segmentation model: Bbox Encoder

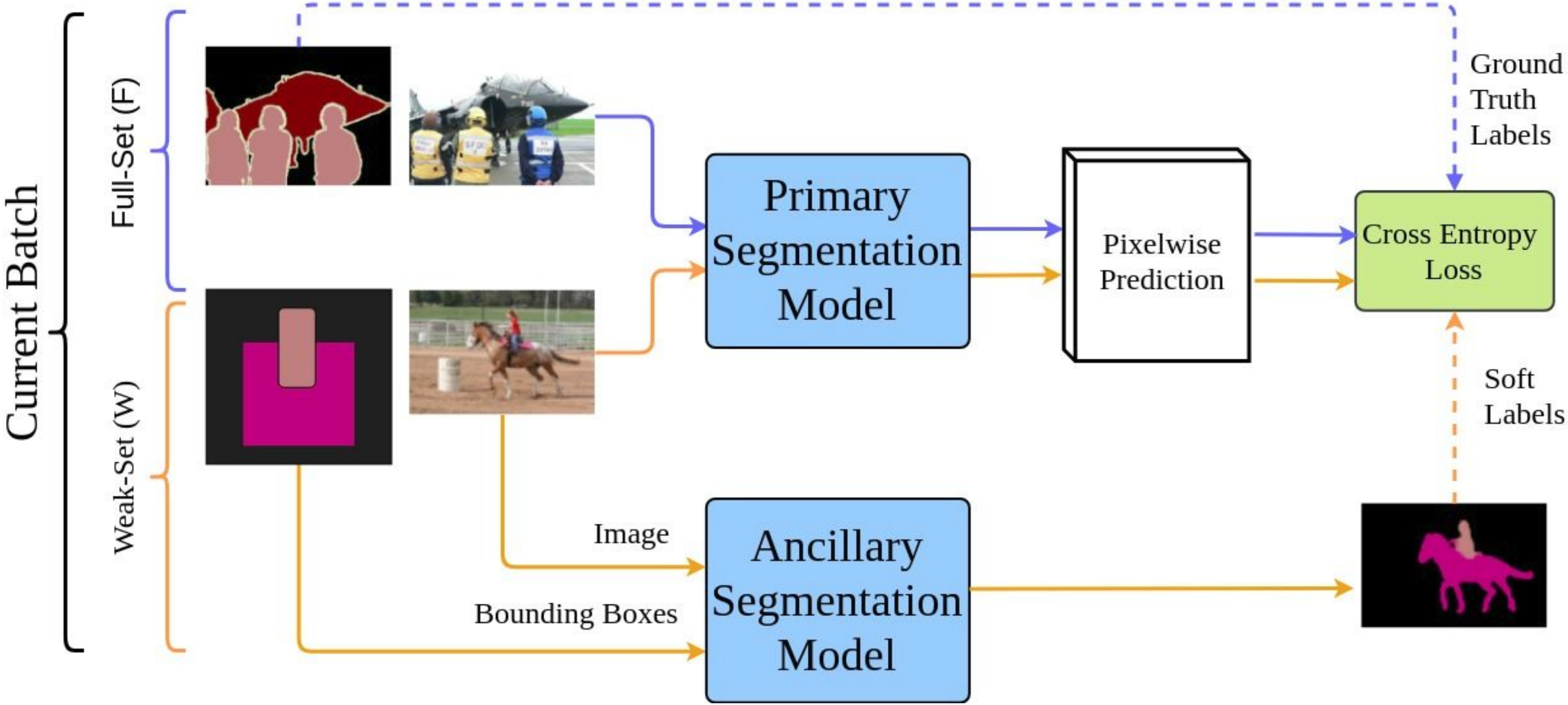
- Bbox Encoder
  - Input: 3D binary tensor for C+1 classes (bboxes marked)
  - Output: heatmap representation for the bboxes (for a scale)



No Self-Correction Approach

- Use the generated **logits** of from the ancillary model
  - **No refining** for the logits
- Train **Primary** segmentation network **$p(y|x)$**
  - Dataset: full-set (discrete labels) + weak-set(logits)

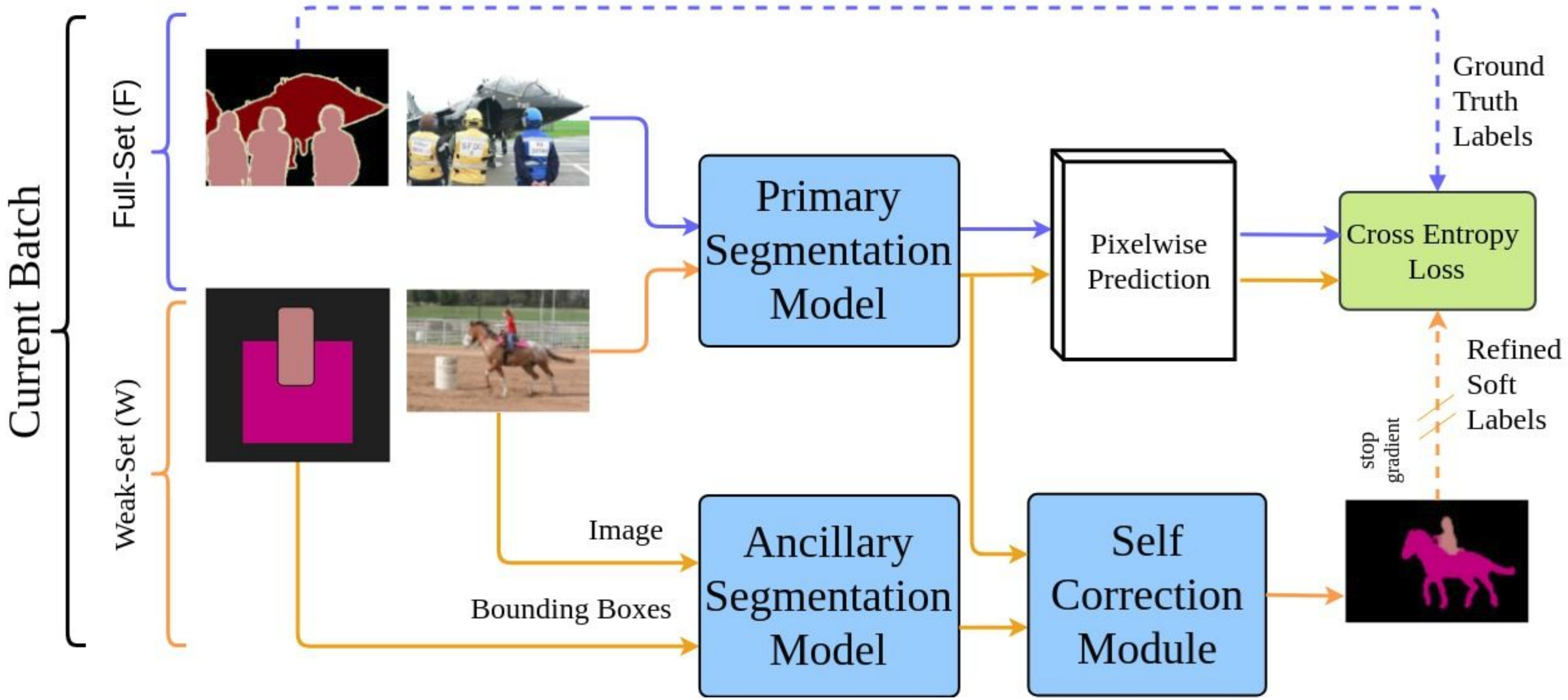
Full model: fixed weak-set logits



Self-Correction: 2 approaches

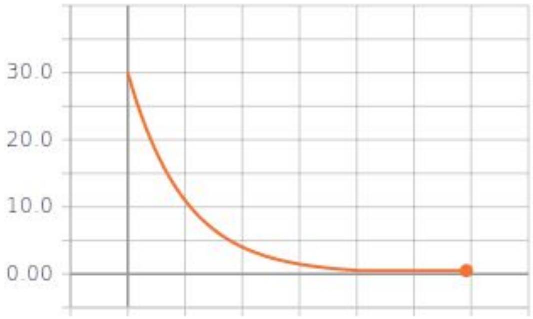
- Goal: Refine predictions made by the ancillary model **when training** the primary network
    - **Combine** the ancillary logits & with primary logits
    - $q(y|x, b) = \text{Combine}(p_{anc}(y|x, b), p(y|x))$
- Two self-correction approaches:
    - **Linear** Self-Correction
    - **Convolutional** Self-Correction

Full model: refining weak-set logits



Linear Self-Correction

- Find **$q(y|x, b)$** that is close to both **$p_{anc}(y|x, b)$ & $p(y|x)$**
  - But primary model is weak in early iterations
  - Use α for weighting the 2 terms
  - α blending (e.g. from 30 to 0.50)



Linear Self-Correction

- Find **factorial** distribution q_{min}:
  - KL(**q(y|x, b)** || **p(y|x)**) + α KL(**q(y|x, b)** || **p_{ans}(y|x, b)**)
- Expand and rearrange - Find q_{min}:
  - $1/(1+\alpha)$ KL(**q(y|x, b)** || [**p(y|x)** . **p_{ans}(y|x, b)**$^{\alpha}$]$^{1/(1+\alpha)}$)
  - The weighted geometric mean of the two distributions
- For a softmax activation, combined logits are: $(\boldsymbol{l}_m + \alpha\, \boldsymbol{l}_m^{anc})/(\alpha + 1)$
- So just anneal α during training and compute the above logits per pixel!

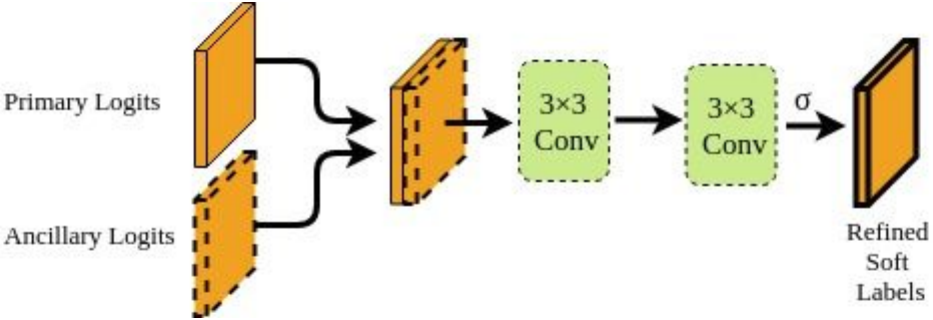
Linear Self-Correction

$$\max_{\boldsymbol{\phi}} \quad \sum_{\mathcal{F}} \log p(\boldsymbol{y}^{(f)}|\boldsymbol{x}^{(f)};\boldsymbol{\phi}) + \qquad (5)$$
$$\sum_{\mathcal{W}} \sum_{\boldsymbol{y}} q(\boldsymbol{y}|\boldsymbol{x}^{(w)},\boldsymbol{b}^{(w)}) \log p(\boldsymbol{y}|\boldsymbol{x}^{(w)};\boldsymbol{\phi}).$$

- Above is just 2 calls for tf.losses.softmax_cross_entropy
  - softmax_cross_entropy_with_logits(one-hot from the ground truth, p's logits)
  - softmax_cross_entropy_with_logits(softmax(q's logits), p's logits)
- Be careful: <u>Stop gradient</u> on q's logits before applying CE

Convolutional Self-Correction

- To avoid tuning the α hyperparameter, use a small network to learn the merging:
  - Input: logits of **$p_{anc}(y|x, b)$ & $p(y|x)$**
  - Output: factorial distribution: **$q_{conv}(y|x, b)$**
- Cons: careful 3-stage training procedure



Results on PASCAL VOC 2012

Data Split $\mathcal{F}$	Data Split $\mathcal{W}$	Method	Val	Test
1464	9118	No Self-Corr.	80.34	81.61
1464	9118	Lin. Self-Corr.	81.35	81.97
1464	9118	Conv. Self-Corr.	**82.33**	**82.72**
1464	9118	EM-fixed Ours [41]	79.25	-
10582	-	Vanilla DeepLabv3+ [9]	81.21	-
1464	9118	BoxSup-MCG [12]	63.5	-
1464	9118	EM-fixed [41]	65.1	-
1464	9118	M ∩ G+ [26]	65.8	-
1464	9118	FickleNet [30]	65.8	-
1464	9118	Song *et al*. [50]	67.5	-
10582	-	Vanilla DeepLabv1 [6]	69.8	-

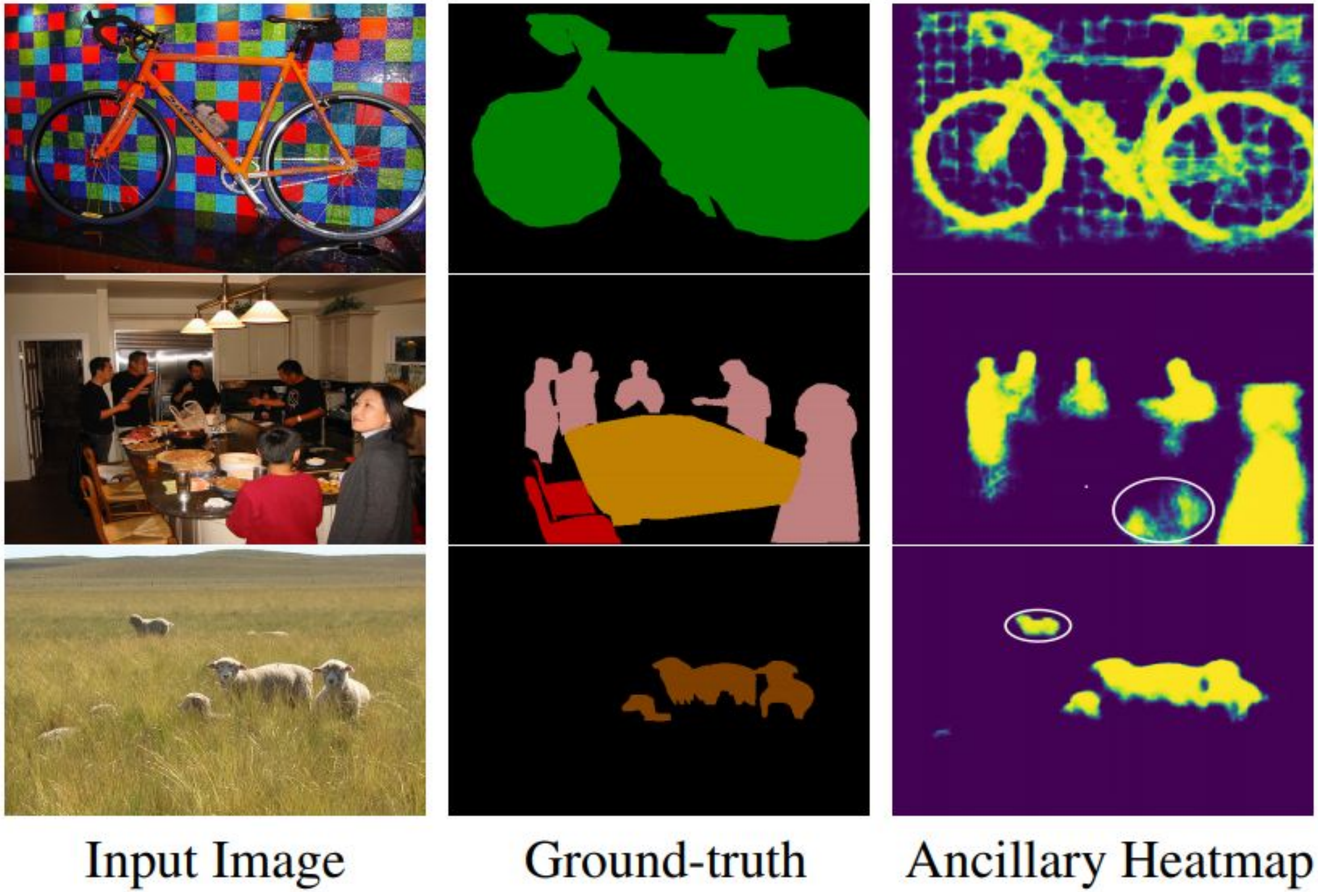
# images in $\mathcal{F}$	200	400	800	1464
Ancillary Model	81.57	83.56	85.36	86.71
No Self-correction	78.75	79.19	80.39	80.34
Lin. Self-correction	**79.43**	79.59	**80.69**	81.35
Conv. Self-correction	78.29	**79.63**	80.12	**82.33**

- Surprisingly, our semi-supervised models outperform the fully supervised model.
- See the paper for our analysis

Results on Cityscapes validation set

Data Split F	Data Split W	Method	mIOU
914	2061	No Self-Corr.	75.44
914	2061	Lin. Self-Correction	76.22
914	2061	Conv. Self-Correction	**79.46**
914	2061	EM-fixed [41]	74.97
2975	-	Vanilla DeepLabv3+$_{ours}$	77.49

# images in $\mathcal{F}$	200	450	914
Ancillary Model	79.4	81.19	81.89
No Self-correction	**73.69**	75.10	75.44
Lin. Self-correction	73.56	75.24	76.22
Conv. Self-correction	69.38	**77.16**	**79.46**



The ancillary model can successfully correct the labels for missing or over segmented objects in these images (marked by ellipses).

Recall: 9k noisy dataset in pascal

Input Image Ground-truth Ancillary Heatmap

Thoughts

- Ancillary model
  - Don’t use handcrafted rules if the network can do it :)
  - Careful injection for bbox encoder to use pretrained models
  - Its high initial performance made it hard for refinement modules to really perform stronger
- Self-correction is useful for noisy ground truth
- Cons: Model not suitable for classes that span whole image (e.g. sky)
- Competing with SOTA in this problem/**setup** is hard, as all approaches close to fully-supervised performance

Linear Self-Correction: Math

$$KL(q, p) + \alpha\, KL(q, c)$$

$$\sum q \log \frac{q}{p} + \sum q\, \alpha \log \frac{q}{c}$$

$$\sum q \left[\log \frac{q}{p} + \log \frac{q^\alpha}{c^\alpha}\right]$$

$$\sum q \left[\log q - \log p + \alpha \log q - \log c^\alpha\right]$$

$$\sum q \left[(1+\alpha) \log q - \log p\, c^\alpha\right]$$

$$\frac{1}{1+\alpha} \sum q \left[\log q - \frac{\log p\, c^\alpha}{1+\alpha}\right]$$

$$\frac{1}{1+\alpha} \sum q \log \frac{q}{(p\, c^\alpha)^t} \qquad t = 1/1+\alpha$$

$$\frac{1}{1+\alpha} KL\left(q \mid (p\, c^\alpha)^{1/1+\alpha}\right)$$

$$= \alpha\, KL(\ \ )$$

Based on standard factorization assu

Let's focus on a specific pixel

let a_i $i \in [1, c]$ logits of $P(y|X)$

that is $P_j = \frac{e^{a_j}}{\sum e^{a_i}} = \frac{e^{a_j}}{A}$

Same for $P_{anc}(y|X)$

$$P_{anc\text{-}j} = \frac{e^{b_j}}{\sum e^{b_i}} = \frac{e^{b_j}}{B}$$

$A, B \Rightarrow$ normalization const.

$$\left(P_j^{1}\, P_{anc\text{-}j}^{\alpha}\right)^{1/1+\alpha} = \left(\frac{e^{a_j}\, e^{\alpha b_j}}{A * B^\alpha}\right)^{1/1+\alpha}$$

$$= \frac{e^{(a_j + \alpha b_j)/(\alpha+1)}}{C}$$

C = some const = who care!