Consistent-Teacher: Towards Reducing Inconsistent Pseudo-targets in Semi-supervised Object Detection

Presented by: Youlkyeong Lee {yklee@islab.ulsan.ac.kr}

Submitted to CVPR2023

Augest 26 2023

Wang, Xinjiang et al. “Consistent-Teacher: Towards Reducing Inconsistent Pseudo-targets in Semi-supervised Object Detection.” (2022).

Problem for Semi-Supervised Object Detection

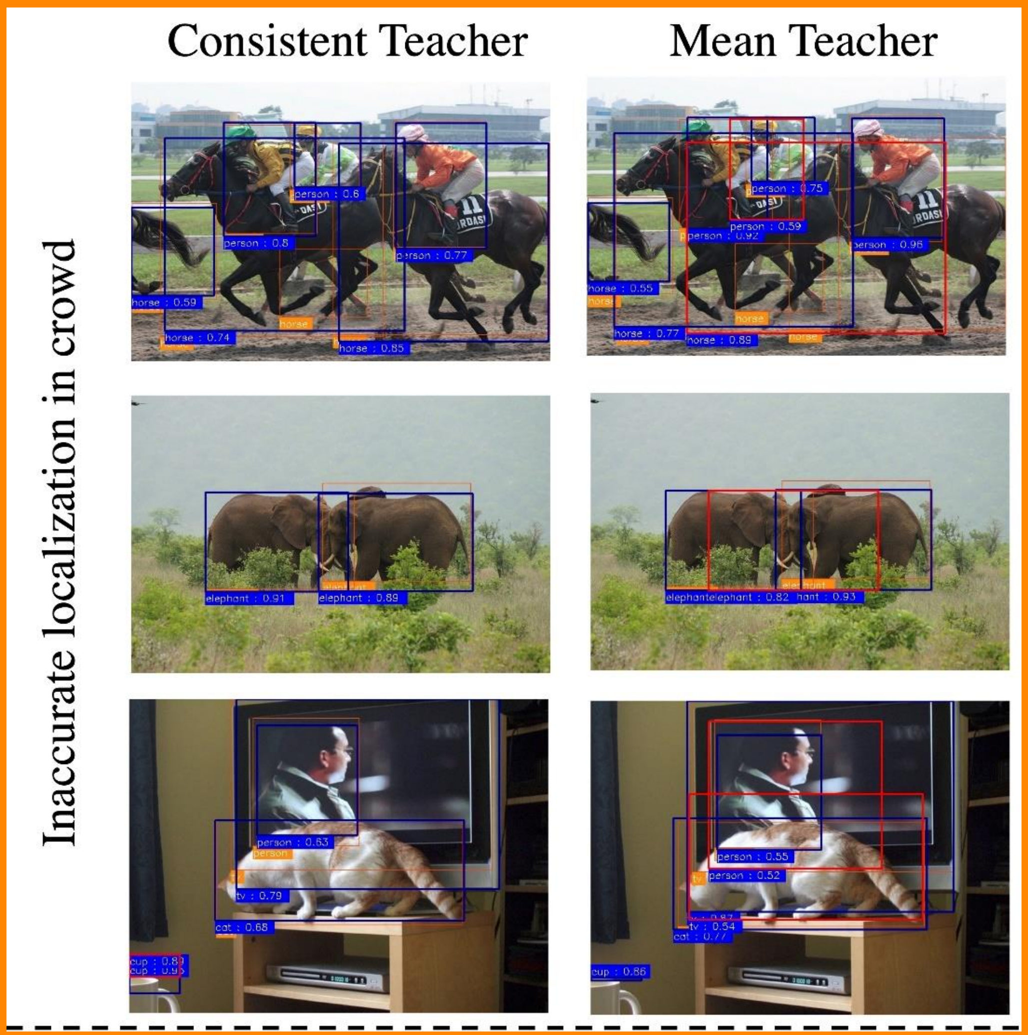
Main Problem

Inconsistency

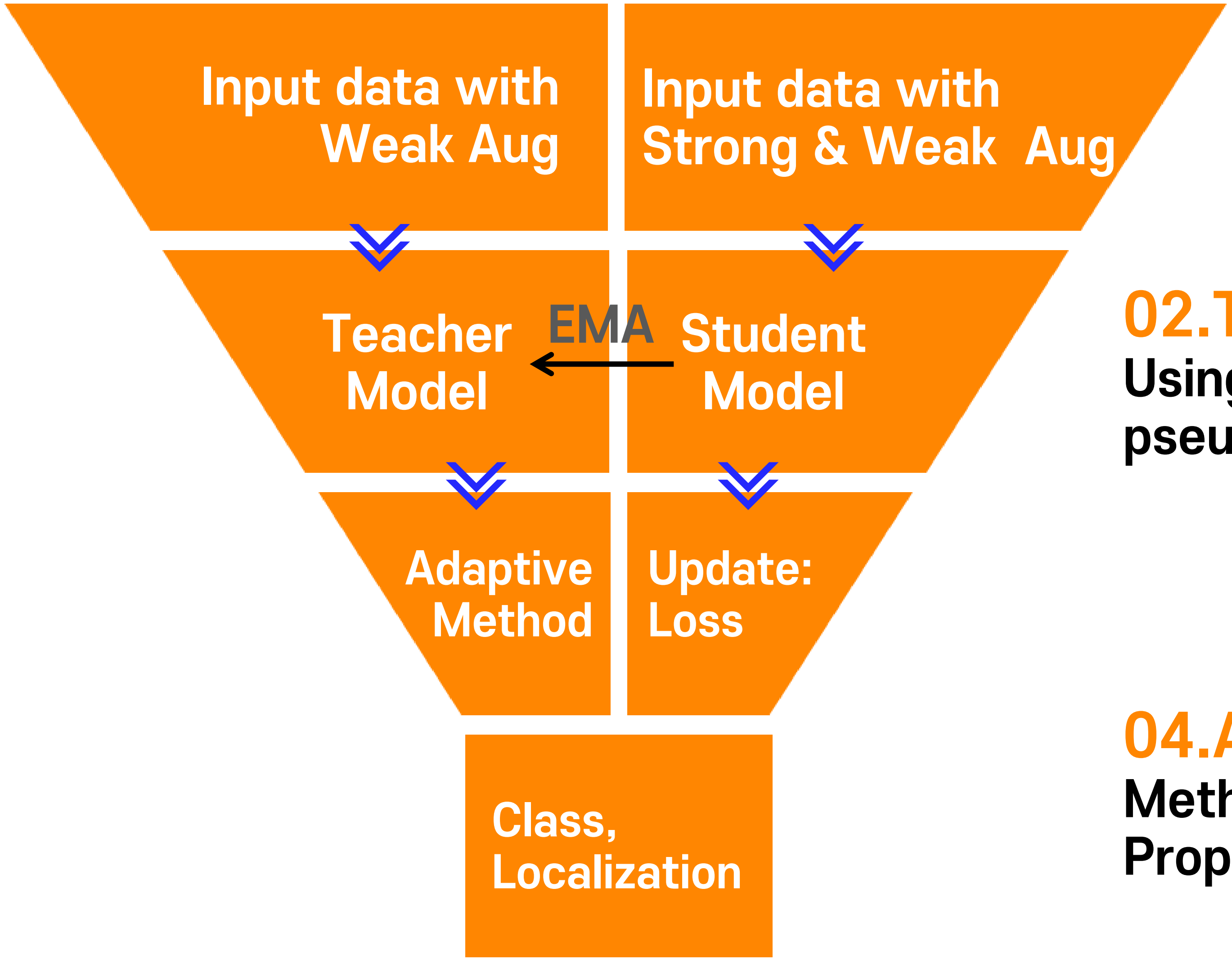
for Pseudo-labels

highly
Inaccurate

sensitively
to Nosie



Process for General Semi-Supervised Learning



01.Data Preparation

Weak: Random flipping
Strong: Randomly change the color, sharpness, contrast Gaussian noise, etc.

02.Teacher Model

Using unlabeled image, generating pseudo-label by teacher model

03.Student Model

Training the model with labeled image

04.Adaptive Method

Method for updated quality of pseudo-label
Proposed loss function

Technical Term

Label dataset

$$\mathcal{D}_L\{\mathbf{x}_i^l, \mathbf{y}_i^l\}^N$$

N: Number of sample
i: number of bbox
l: label data

Unlabel dataset

$$\mathcal{D}_U\{\mathbf{x}_j^u\}^M$$

M: Number of sample
j: number of bbox
u: unlabel data

Model

$$f_t(\cdot;\Theta_t)$$

$$f_s(\cdot;\Theta_s)$$

Augmentation

$$T, T'$$

T: Weak Aug
T': Strong Aug

Ground Truth

$$\mathbf{y} = \{y_l = (c_l, bbox_l)\}_{l=1}^L$$

L: Number of bbox
Cl: Classificition label

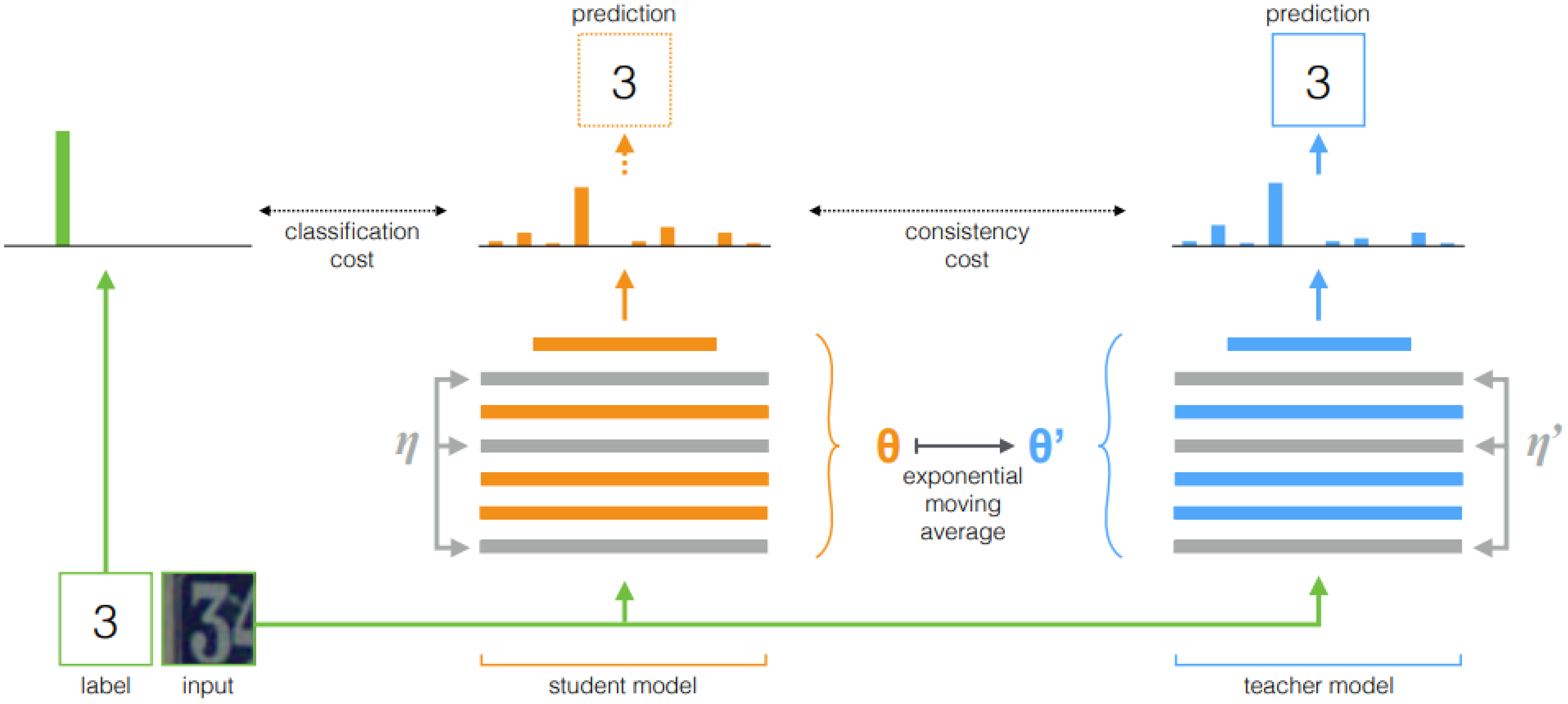
Pseudo label

$$\hat{\mathbf{y}} = f_t(T(\mathbf{x};\Theta_t)$$

ft: Teacher model

Baseline Semi-Supervised Detector

- Baseline: Mean-Teacher with RetinaNet detector
- Main Idea for Mean-Teacher: proposed how to update between teacher and student -> Exponential moving average



1.Baseline Semi-Supervised Detector

- Baseline: Mean-Teacher with RetinaNet detector
- Focal loss for classification / GIoU loss for regression

$$\mathcal{L} = \frac{1}{N} \sum_{i} \left[\mathcal{L}_{cls}\left(f_s(T(\mathbf{x}_i^l)), \mathbf{y}_i^l\right) + \mathcal{L}_{reg}\left(f_s(T(\mathbf{x}_i^l)), \mathbf{y}_i^l\right) \right]$$
$$+ \lambda_u \frac{1}{M} \sum_{j} \left[\mathcal{L}_{cls}\left(f_s(T'(\mathbf{x}_j^u)), \hat{\mathbf{y}}_j^u\right) + \mathcal{L}_{reg}\left(f_s(T'(\mathbf{x}_j^u)), \hat{\mathbf{y}}_j^u\right) \right],$$

Equation of Loss function for SSOD

Architecture of Consistent-Teacher

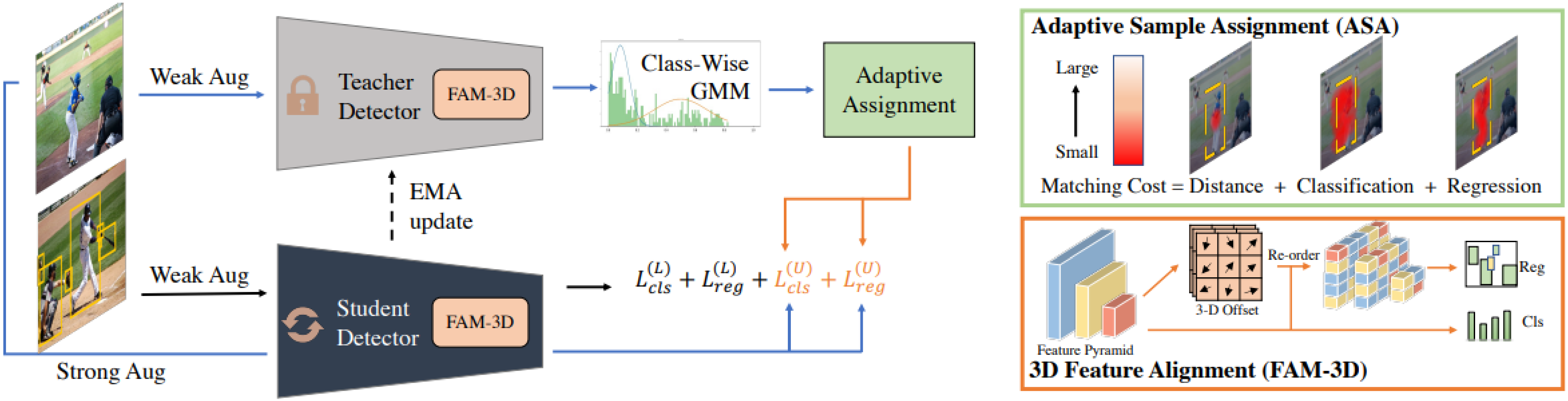


Figure 2. The pipeline of `Consistent-Teacher`. We design three modules to address the inconsistency in SSOD, where GMM dynamically determines the threshold; 3D feature alignment calibrates regression quality; Adaptive assignment assigns anchor based on matching cost.

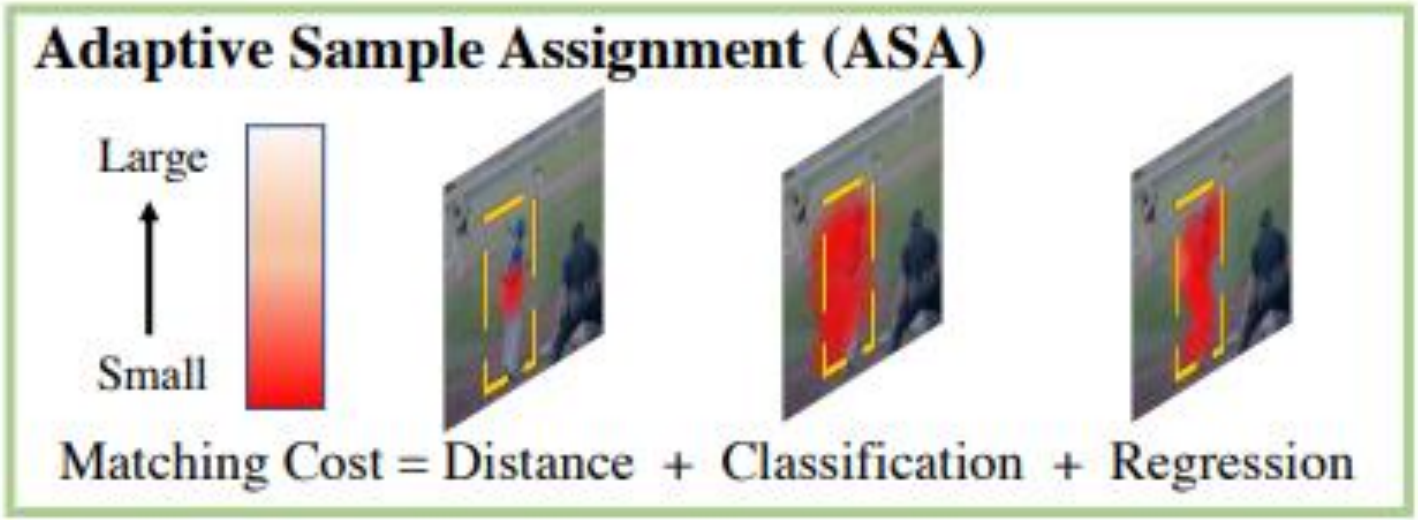
2.Consistent Adaptive Sample Assignment

- Assigning each anchor, if IoU with GT is larger than threshold -> positive

$$\hat{c} = \underset{c}{\operatorname{argmin}} \mathcal{L}(f_t(\mathbf{x}^u), c),$$

- However, the problem of drifting phenomenon about pseudo label in Fig. 1
- Suppose calculating the distance between center of anchor and pseudo-bbox

$$C_{nl} = \mathcal{L}_{cls}(p_n, y_l) + \lambda_{reg}\mathcal{L}_{reg}(p_n, y_l) + \lambda_{dist}C_{dist}$$



$\lambda_{dist} = 0.001$

Problem for Semi-Supervised Object Detection

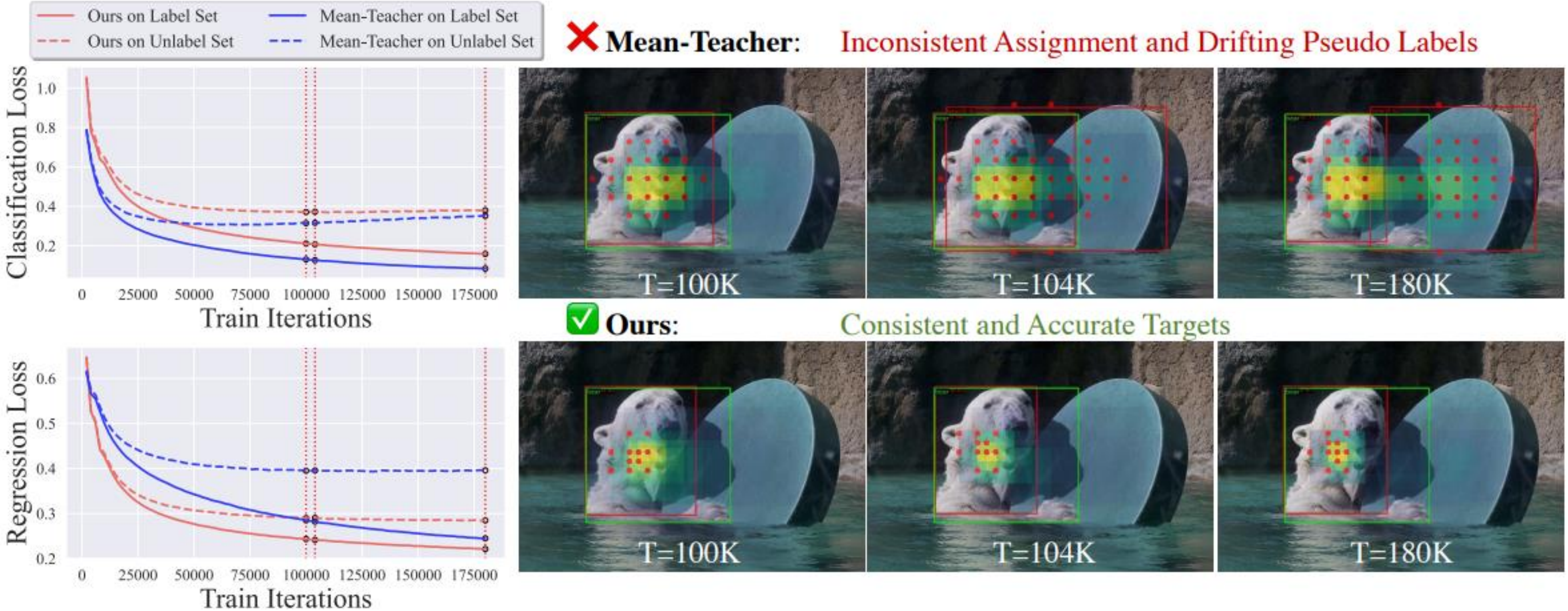


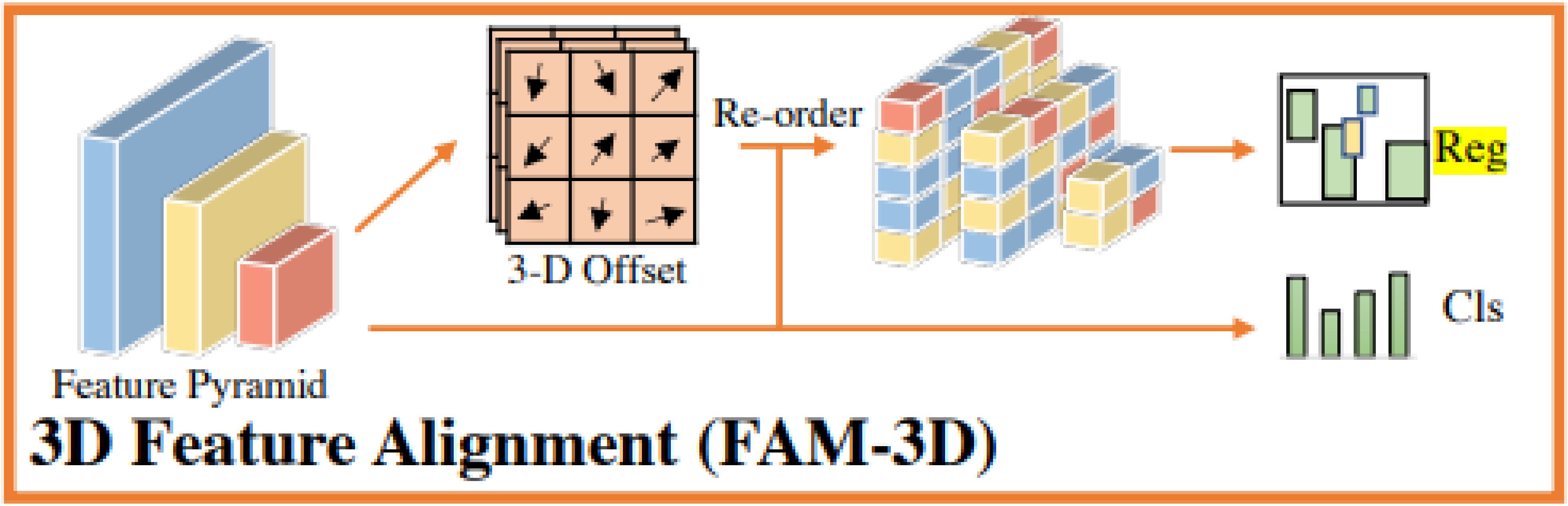
Figure 1. Illustration of inconsistency problem in SSOD on COCO 10 % evaluation. (Left) We compare the training losses between the Mean-Teacher and our `Consistent-Teacher`. In Mean-Teacher, inconsistent pseudo targets lead to overfitting on the classification branch, while regression losses become difficult to converge. In contrast, our approach sets consistent optimization objectives for the students, effectively balancing the two tasks and preventing overfitting. (Right) Snapshots for the dynamics of pseudo labels and assignment. The Green and Red bboxes refer to the ground-truth and pseudo bbox, respectively, for the polar bear. Red dots are the assigned anchor boxes for the pseudo label. The heatmap indicates the dense confidence score predicted by the teacher (brighter the larger). A nearby board is finally misclassified as a polar bear in the baseline while our adaptive assignment prevents overfitting.

3.BBox Consistency via 3-D Feature Alignment

- a 3-D Feature Alignment Module (FAM-3D) to calibrate the bbox localization with classification confidence, inspired by TOOD
- CONV3×3(RELU(CONV1×1)) layer at different FPN levels

$$\mathbf{d} = (d_0, d_1, d_2) \in \mathbb{R}^3$$

$$P'(i,j,l) \leftarrow P(i+d_0, j+d_1, l)$$
$$P'(i,j,l) \leftarrow P'(i',j',l+d_2),$$



4. Thresholding with Gaussian Mixture Model

- Static hyperparameter τ for pseudo-bboxes filtering
- To find a way **to automatically distinguish the positive** from negative pseudo-bboxes
- Gaussian mixture (GMM) distribution

$$\mathcal{P}(s^c) = w_n^c \mathcal{N}(s^c | \mu_n^c, (\sigma_n^c)^2) + w_p^c \mathcal{N}(s^c | \mu_p^c, (\sigma_p^c)^2),$$

- Adaptive score threshold for training student

$$\tau^c = \underset{s^c}{\operatorname{argmax}} \mathcal{P}(pos | s^c, \mu_p^c, (\sigma_p^c)^2)$$

Experiments

- Datasets
  - MS-COCO
- COCO-standard(train2017)
  - 118k labeled images
  - 850k instances from 80 classes
  - 123k unlabeled images
  - Randomly sample 1, 5, and 10% of labeled training data as **a labeled set**
  - Rest of labeled data as an unlabeled set
  - 1%: 1.2k images

Sohn, Kihyuk, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee and Tomas Pfister. "**A Simple Semi-Supervised Learning Framework for Object Detection**." *ArXiv* abs/2005.04757 (2020): n. pag.

Experiments

- PASCAL-VOC
  - VOC07 trainval: 5,011 training images from 20 classes as **a labeled set**
  - VOC12 trainval: 11,540 training images as **an unlabeled set**
  - Validation sets: COCO val2017 and VOC07 test set, respectively
- COCO-additional
  - Train2017-unlabeled data: 123k
- Details
  - 8 GPUs
  - Randomly sample 5 images from 1 labeled and 4 unlabeled data per GPU
  - EMA: 0.9995
- Network
  - ResNet-50-FPN backbone in RetinaNet
  - Initial weight: pre-trained on ImageNet

Inconsistency Leading to Noisy Labels

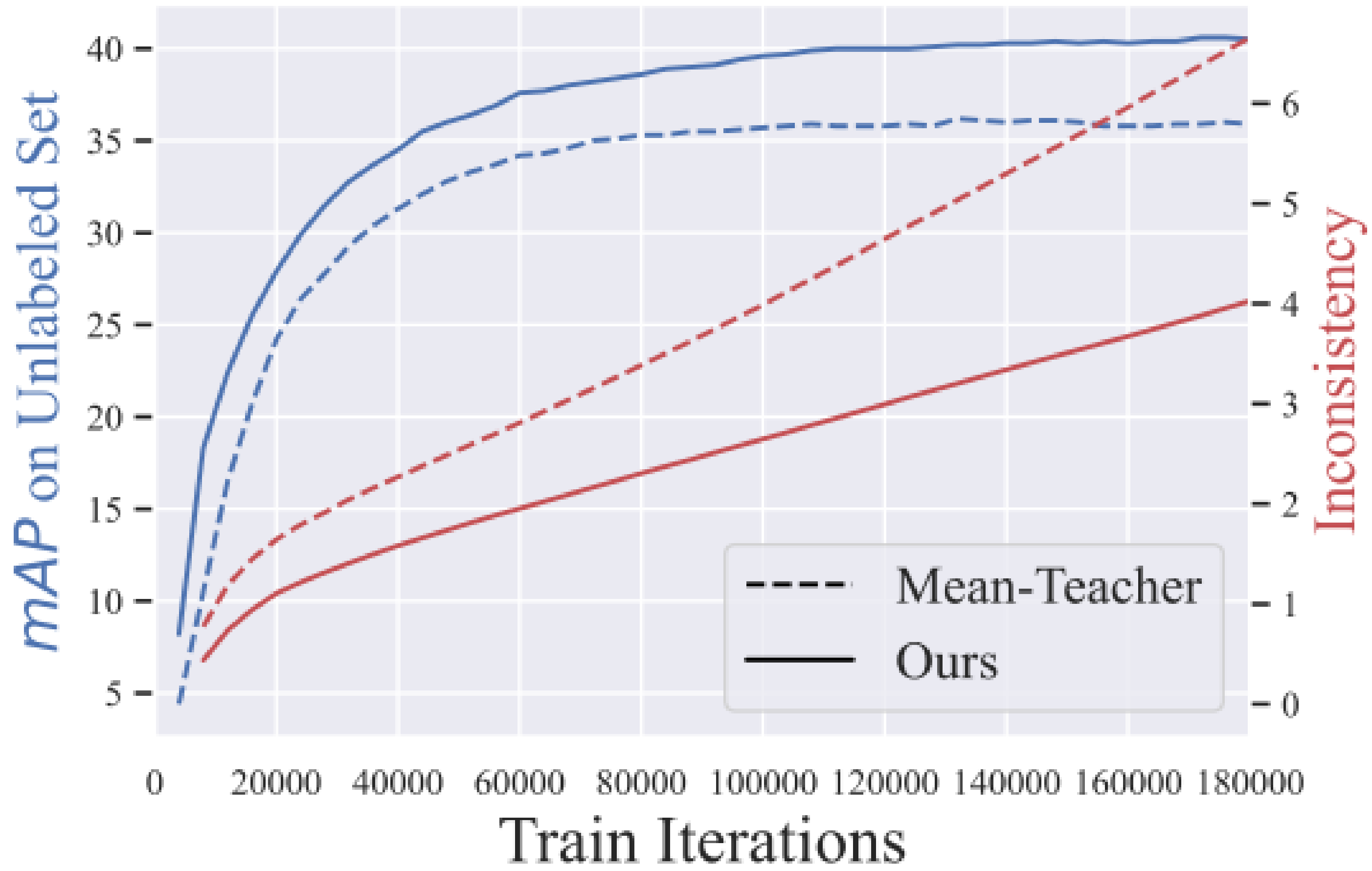


Figure 3. `Consistent-Teacher` improves the training consistency in SSOD. (Left axis) mAP on the unlabeled set at different times. (Right axis) The inconsistency of pseudo labels.

Inconsistency Caused by Classification-Regression Misalignment

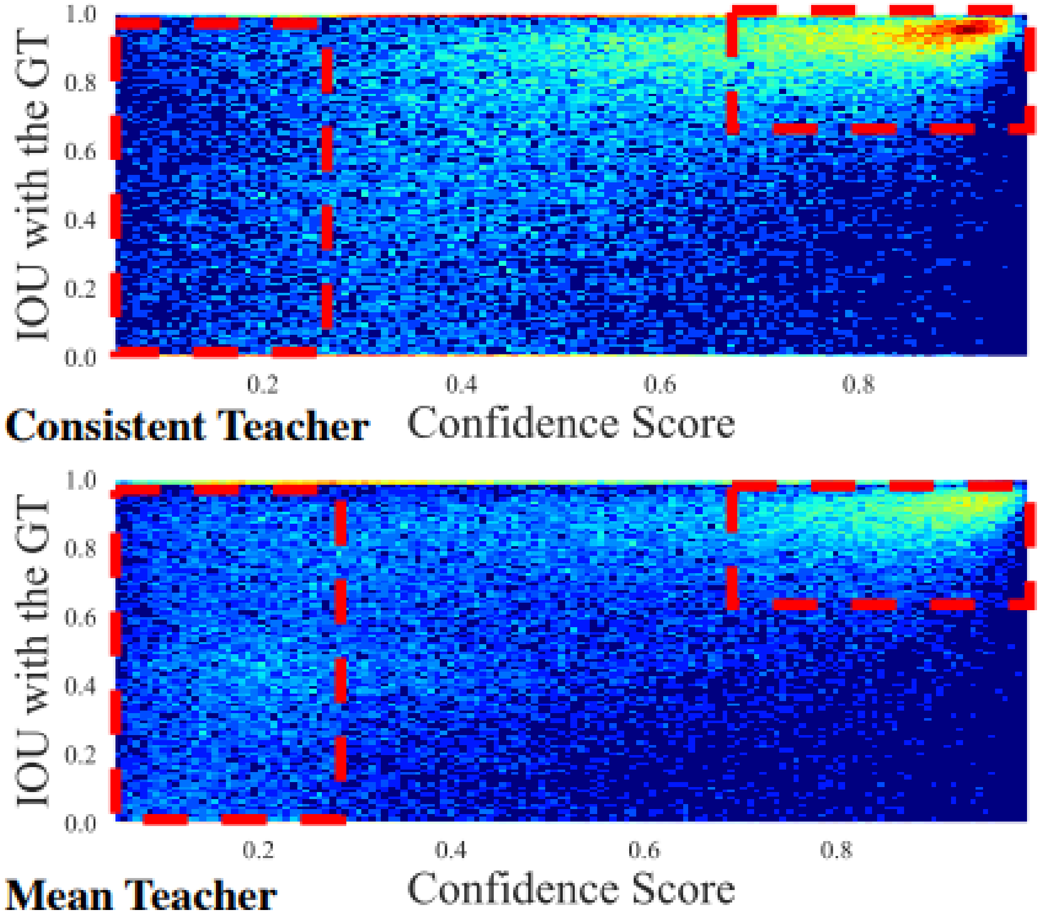


Figure 4. Heatmap of predicted bboxes confidence and its IoU score with GTs.

Inconsistency Caused by Hard Score Threshold

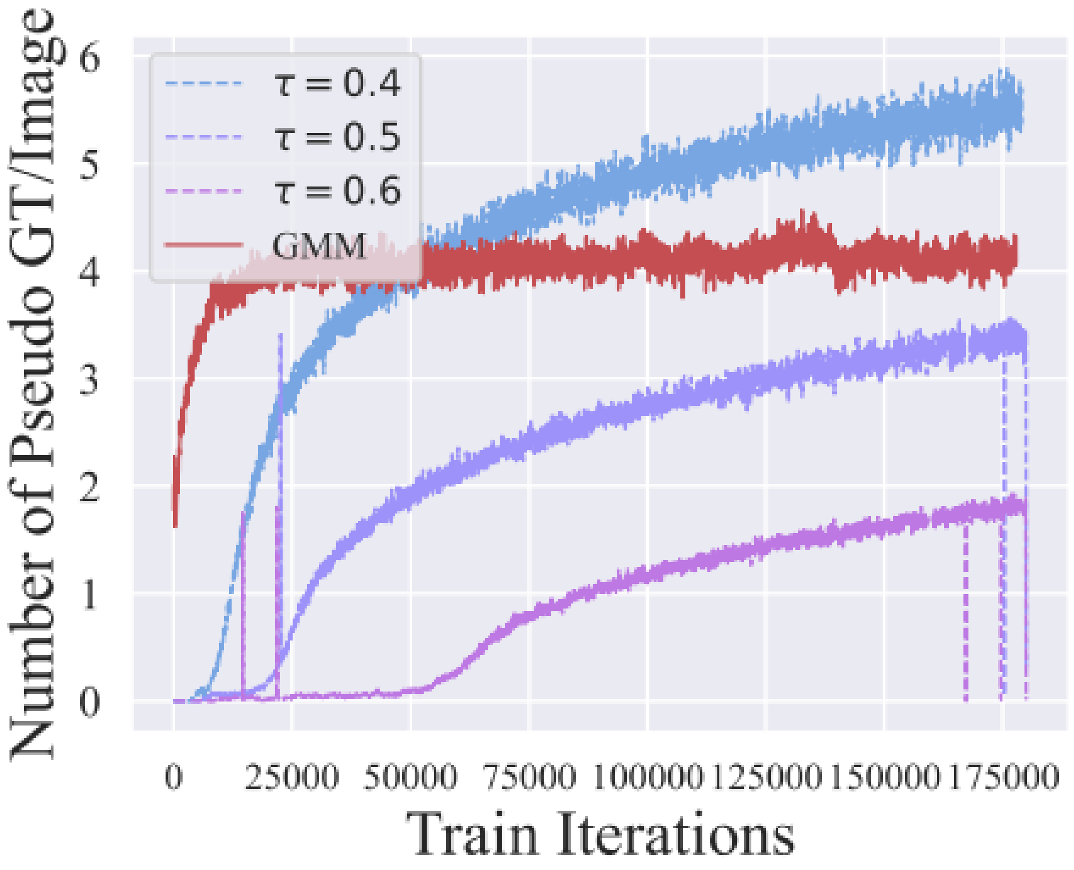


Figure 5. Number of pseudo labels/image with threshold schedules on COCO 10%.

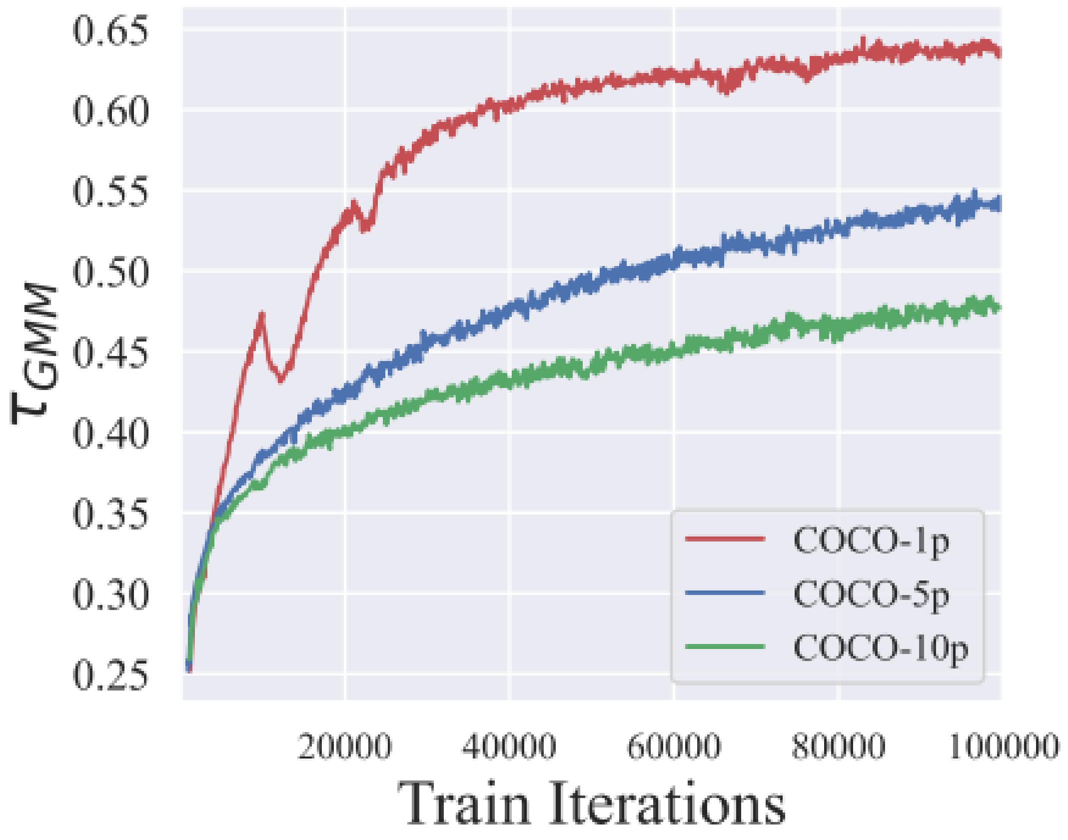


Figure 6. Average GMM thresholds across different classes along with the training.

Experiment Result

Table 1. COCO-PARTIAL comparison with other semi-supervised detector on val2017. The results for two-stage (upper half) and single-stage (lower half) detectors are listed separately. We also report the Faster-RCNN and RetinaNet performance trained on labeled data only. All models adopt ResNet50 with FPN as the backbone. We highlight the previous best record with <u>underline</u>.

	Method	1% COCO	2% COCO	5% COCO	10% COCO
Faster-RCNN	Labeled Only	9.05	12.70	18.47	23.86
	CSD	10.51	13.93	18.63	22.46
	STAC	13.97	18.25	24.38	28.64
	Instant Teaching	18.05	22.45	26.75	30.40
	Humble teacher	16.96	21.72	27.70	31.61
	Unbiased Teacher	20.75	24.30	28.27	31.50
	Soft Teacher	20.46	-	30.74	34.04
	ACRST	<u>26.07</u>	<u>28.69</u>	31.35	34.92
	PseCo	22.43	27.77	32.50	36.06
RetinaNet	Labeled Only	10.22	13.80	19.40	24.10
	Unbiased Teacher v2	22.71	26.03	30.08	32.61
	DSL	22.03	25.19	30.87	36.22
	Dense Teacher	22.38	27.20	<u>33.01</u>	<u>37.13</u>
	S4OD	20.10	-	30.00	32.90
	Mean-Teacher	20.40	26.00	30.40	35.50
	Consistent-Teacher	**25.30**	**30.40**	**36.10**	**40.00**

Experiment Result

Table 2. COCO-ADDITION experimental results on `val2017` with `unlabel2017` as unlabeled set. Note that 1× represents 90K training iterations, and N× represents N×90K iterations.

Method	$AP_{50:95}$
CSD(3×)	$40.20 \xrightarrow{-1.38} 38.82$
STAC(6×)	$39.48 \xrightarrow{-0.27} 39.21$
Unbiased Teacher(3×)	$40.20 \xrightarrow{+1.10} 41.30$
ACRST(3×)	$40.20 \xrightarrow{+2.59} 42.79$
Soft Teacher(16×)	$40.90 \xrightarrow{+3.70} 44.50$
DSL(2×)	$40.20 \xrightarrow{+3.60} 43.80$
PseCo(8×)	$41.00 \xrightarrow{+5.10} 46.10$
Dense Teacher(8×)	$41.24 \xrightarrow{+4.88} 46.12$
`Consistent-Teacher` (8×)	$40.50 \xrightarrow{\mathbf{+7.20}} \mathbf{47.70}$

- **STAC**: SSL for object detection(**Self-Training** and the **Augmentation** driven **Consistency** regularization), 2020
- **CSD**: **C**onsistency-based **s**emi-supervised learning for object **d**etection, 2019
- **Soft Teacher**: End-to-End Semi-Supervised Object Detection with **Soft Teacher,** 2021
- **Dense Teacher**: Dense Pseudo-Labels for Semi-supervised Object Detection, 2022
- **PseCo**: **Pse**udo Labeling and **Co**nsistency Training for Semi-Supervised Object Detection, 2022

Experiment Result

Table 3. VOC-PARTIAL experimental results comparison with other semi-supervised detector on VOC07 labeled and VOC12 unlabeled set.

Method	AP_{50}	$AP_{50:95}$
Labeled Only	72.63	42.13
CSD	74.70	-
STAC	77.45	44.64
ACRST	78.16	50.12
Instant Teaching	79.20	50.00
Humble Teacher	80.94	53.04
Unbiased Teacher	77.37	48.69
Unbiased Teacher v2	81.29	56.87
Mean-Teacher	77.02	53.61
Consistent-Teacher	**81.00**	**59.00**

- **ACRST**: Semi-Supervised Object Detection with **A**daptive **C**lass-**R**ebalancing **S**elf-**T**raining, 2021
- **Instant Teaching**: **Instant-Teaching**: An End-to-End Semi-Supervised Object Detection Framework, 2021
- **Humble Teacher**: Humble Teachers Teach Better Students for Semi-Supervised Object Detection, 2021
- **Unbiased Teacher**: **Unbiased Teacher** for Semi-Supervised Object Detection, 2021
- **Unbiased Teacher v2**: **Unbiased Teacher v2**: Semi-supervised Object Detection for Anchor-free and Anchor-based Detectors, 2022

Table 4. Comparisons between IoU-based and our adaptive anchor assignment on COCO.

Assignment	$AP^{1\times}_{50:95}$	$AP^{10\%}_{50:95}$
IoU-based	38.4	35.50
our ASA	40.1 (+1.7)	38.50 (+3.0)

Table 5. Ablation Study on detection head structure. We compare the performance, model size, and FLOPs on different head structures on COCO 10% and standard $1\times$ evaluation. FLOPs are measured on the input image size of 1280×800.

Method	FLOPs (G)	$AP^{1\times}_{50:95}$	$AP^{10\%}_{50:95}$
Ours w/o FAM	205.21	40.1	38.5
Ours w FAM-2D	205.70	40.4 (+0.3)	39.1 (+0.6)
Ours w FAM-3D	208.49	40.7 (+0.6)	39.5 (+1.0)

Experiment Result

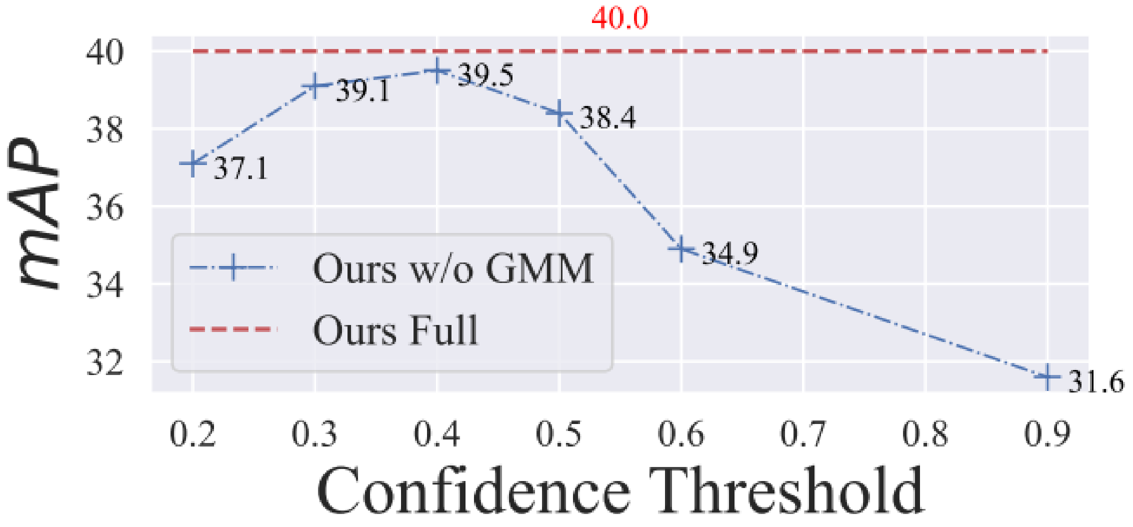


Figure 7. Ablative study of GMM-based pseudo-label filtering. Each value represents the mAP score on COCO 10% data.

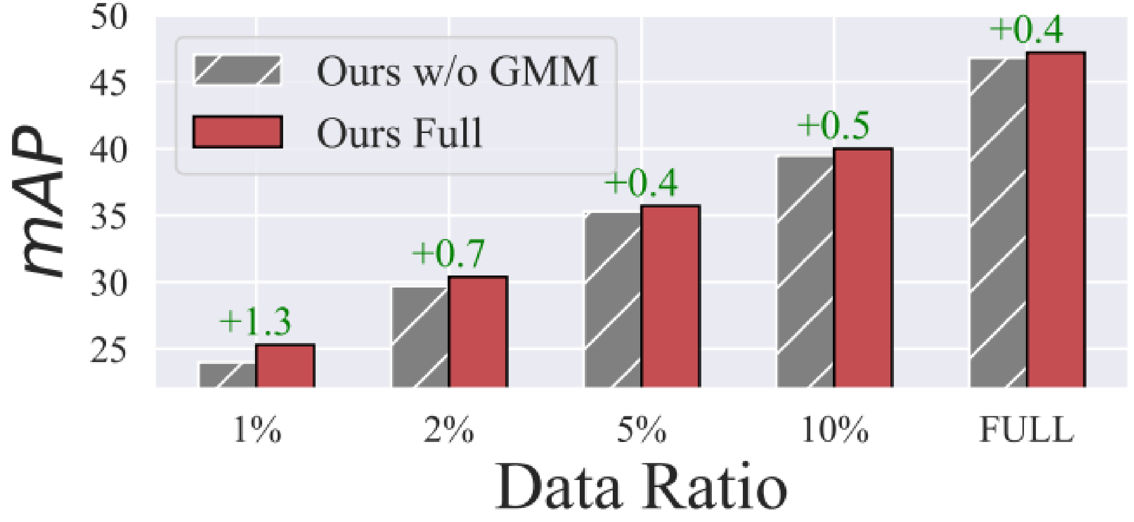


Figure 8. Ablation of GMM at different data ratio on COCO. Models are compared to baselines with a hard threshold 0.4.

Conclusion

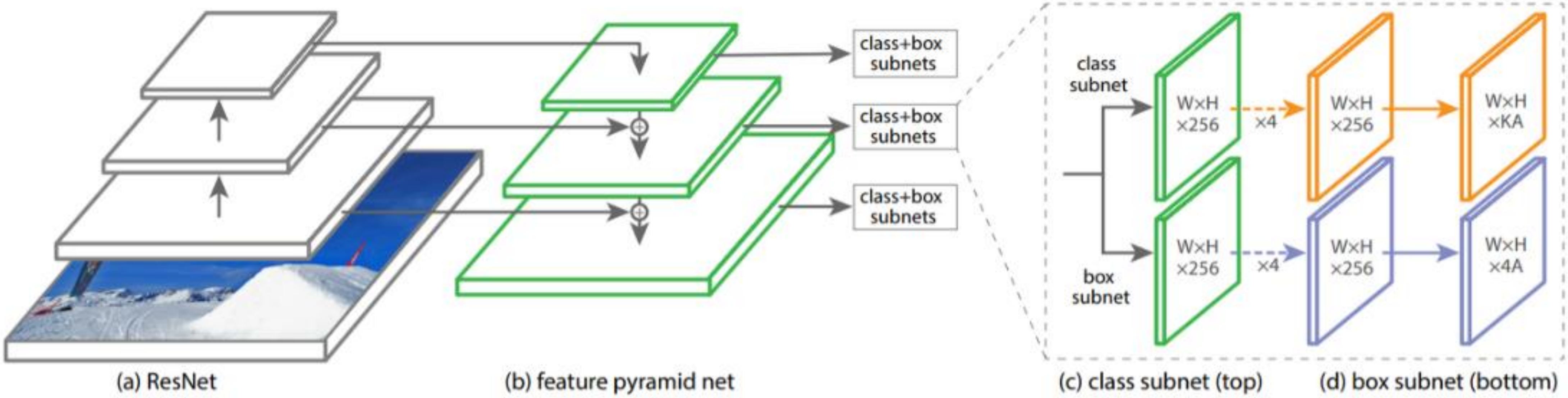
- RetinaNet based Semi-Supervised Object Detection
- Adaptive pseudo label assignment
  - **Consistent Adaptive Sample Assignment**
  - **BBox Consistency via 3-D Feature Alignment**
  - **Thresholding with Gaussian Mixture Model**

Thank you very much
for your attention!

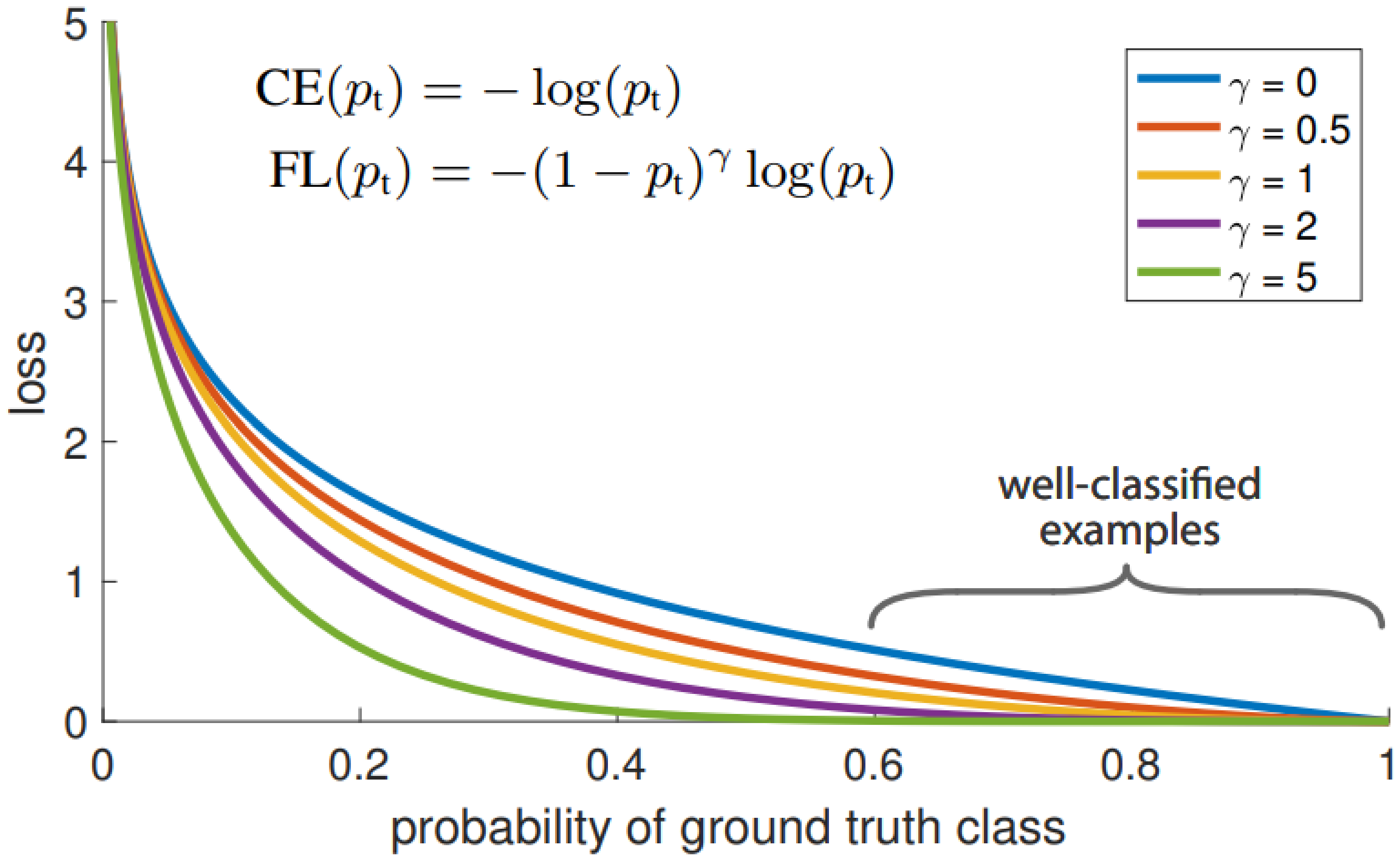
Appendix

Architecture of RetinaNet

- Main Idea for RetinaNet: Focal Loss
- Model: ResNet + FPN(feature pyramid network)



Cross Entropy Loss vs Focal Loss



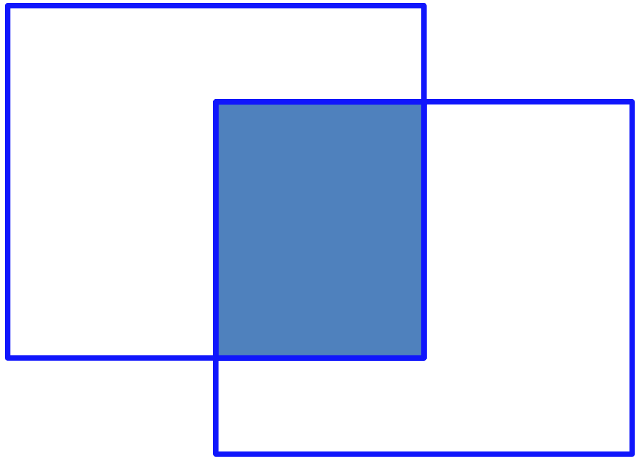
$$\text{Focal Loss} = -(1-p_t)^\gamma \log(p_t)$$

Lin, Tsung-Yi et al. "Focal Loss for Dense Object Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2020): 318-327.

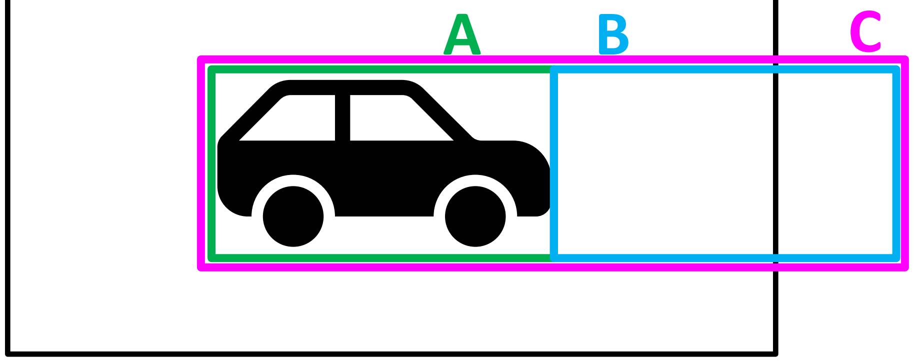
Intersection over Union

IoU =

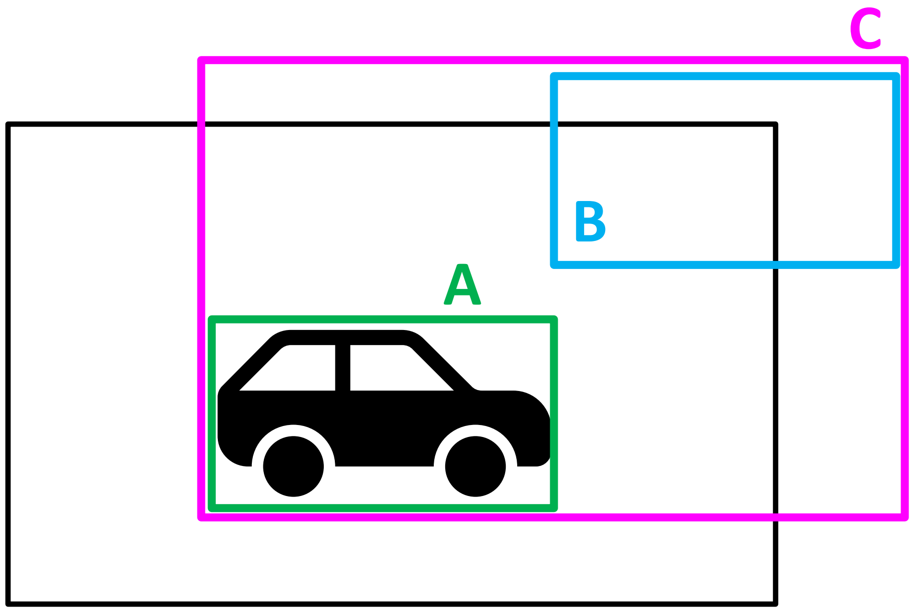
Area of Overlap



Area of Union



IoU=0
GIoU=0

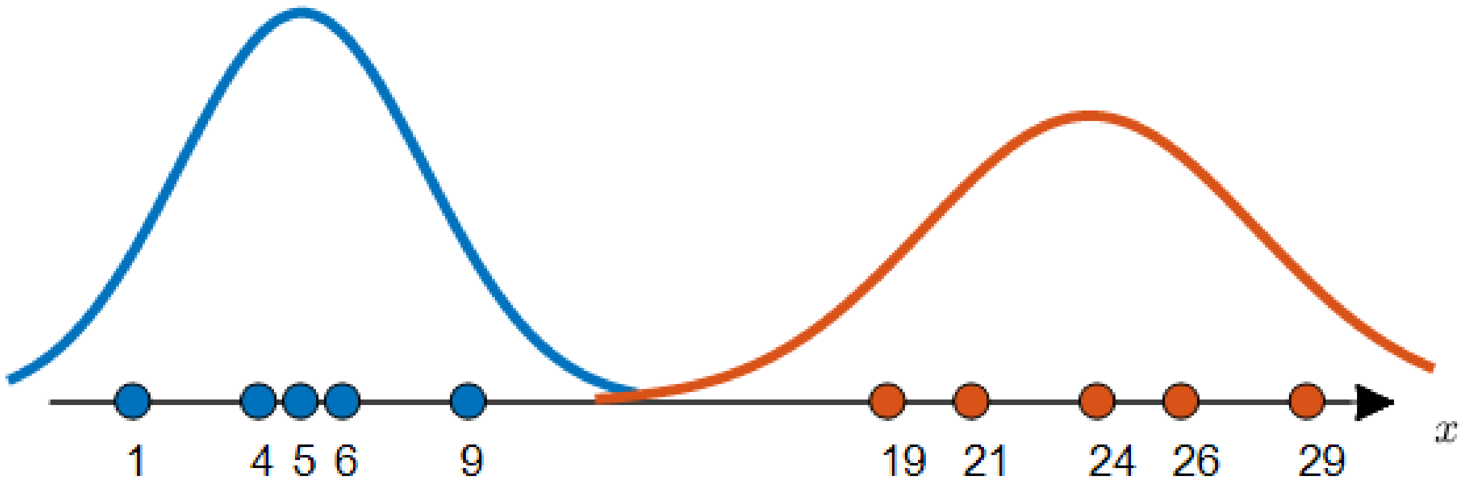


IoU=0
GIoU=-0.7

$$GIoU = IoU - \frac{|C-(A \cup B)|}{|C|}$$

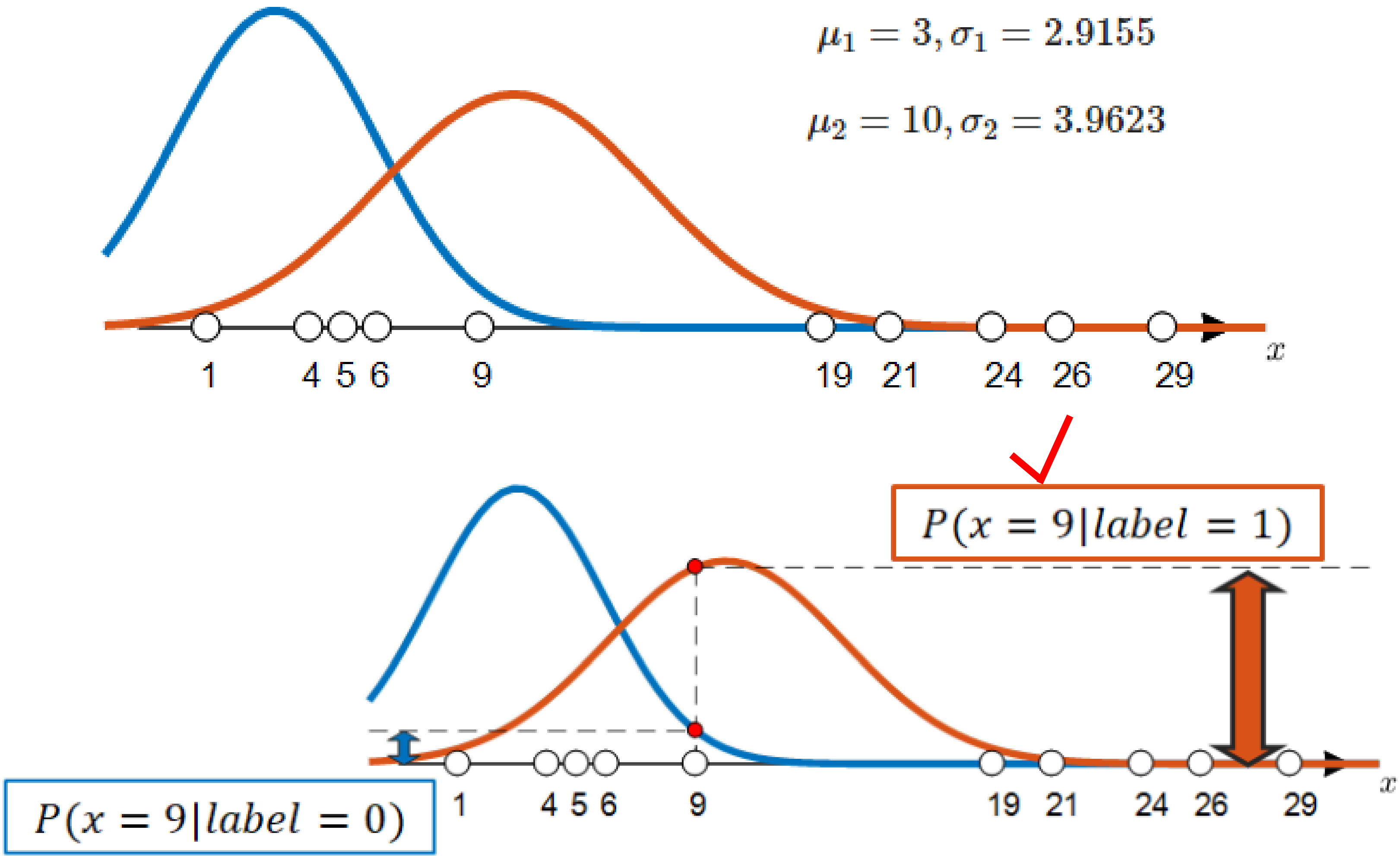
$$Loss = 1 - GIoU, 0 \leq Loss \leq 2$$

- In group of data, if there are distribution, it can expect to decide the label

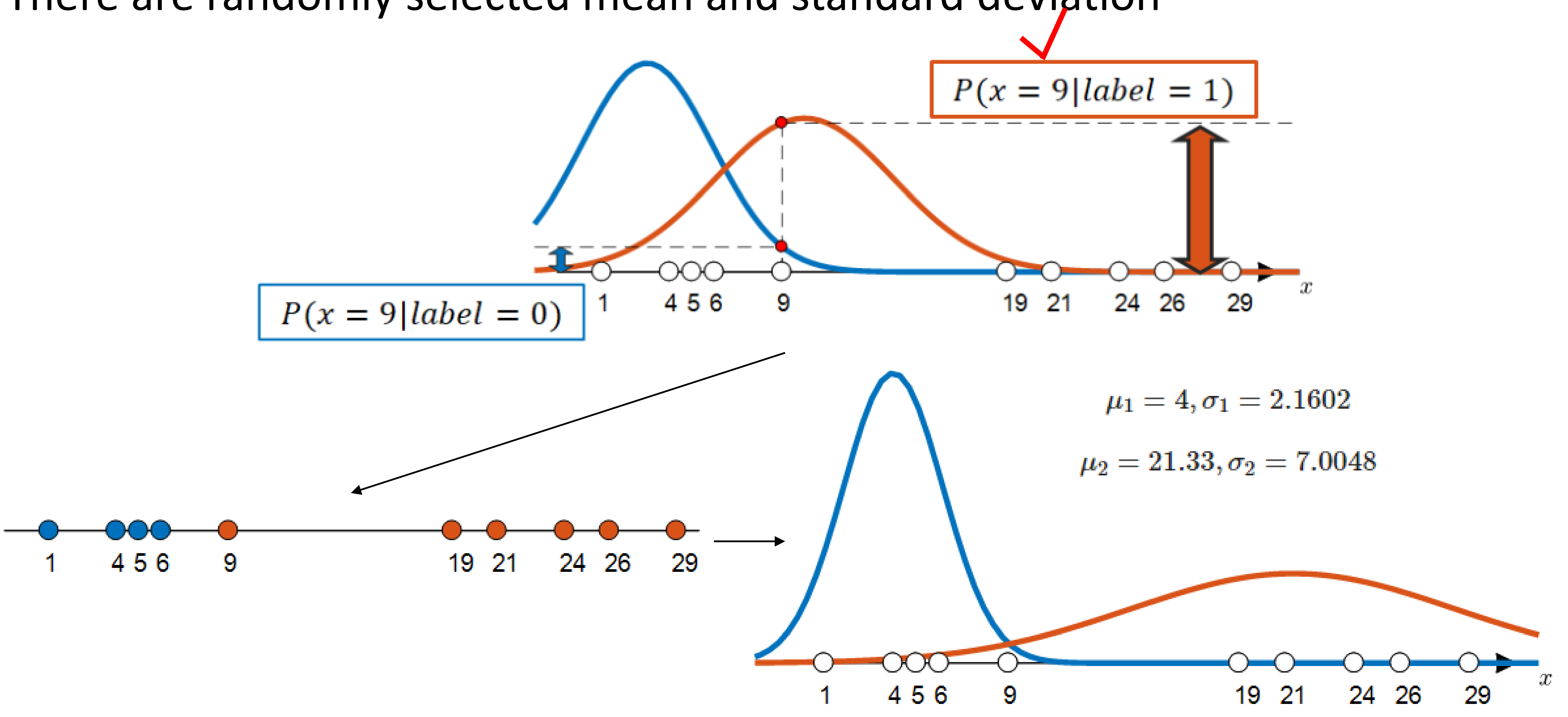


- How to decide the label?

- There are randomly selected mean and standard deviation



- There are randomly selected mean and standard deviation



Overview Framework

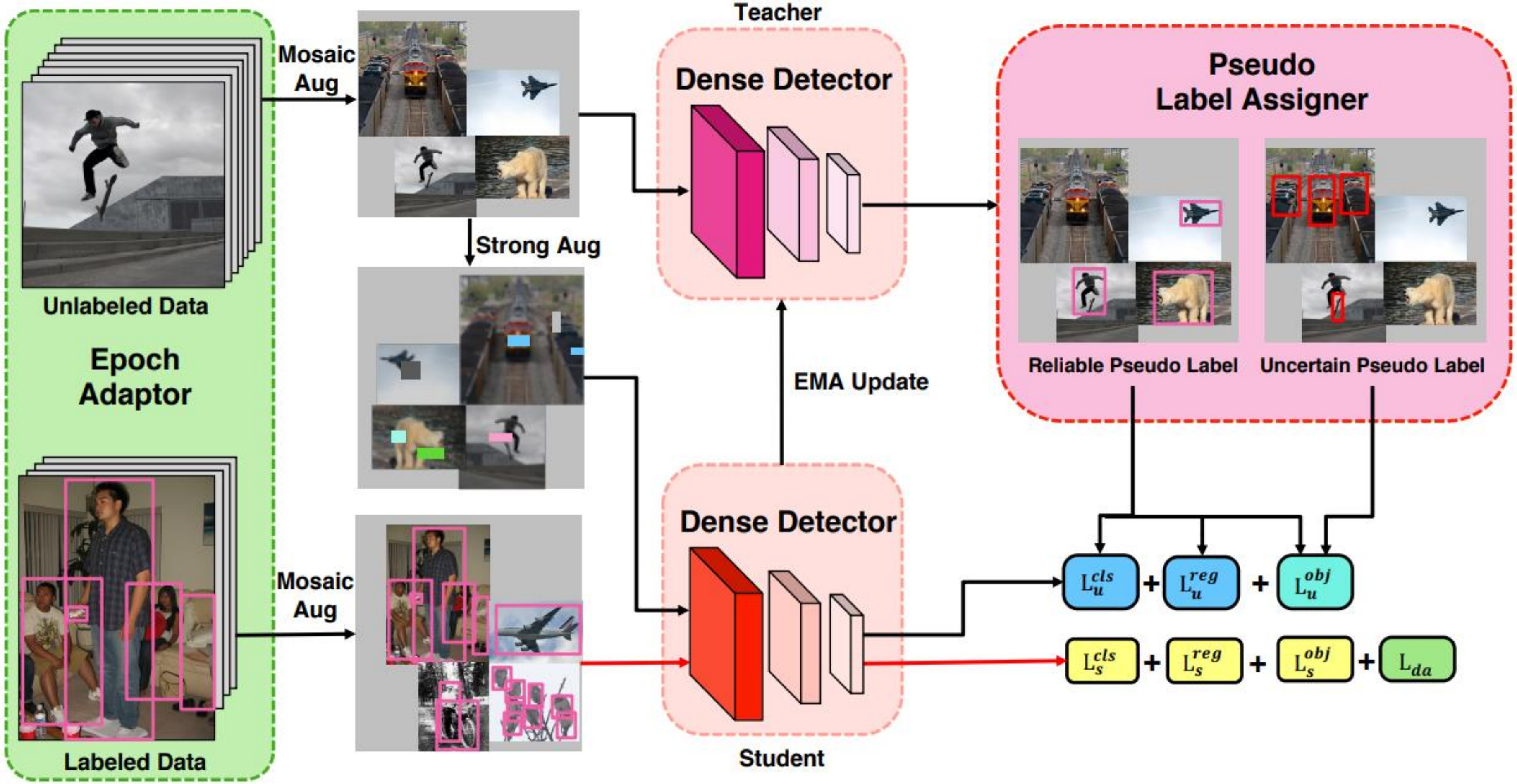
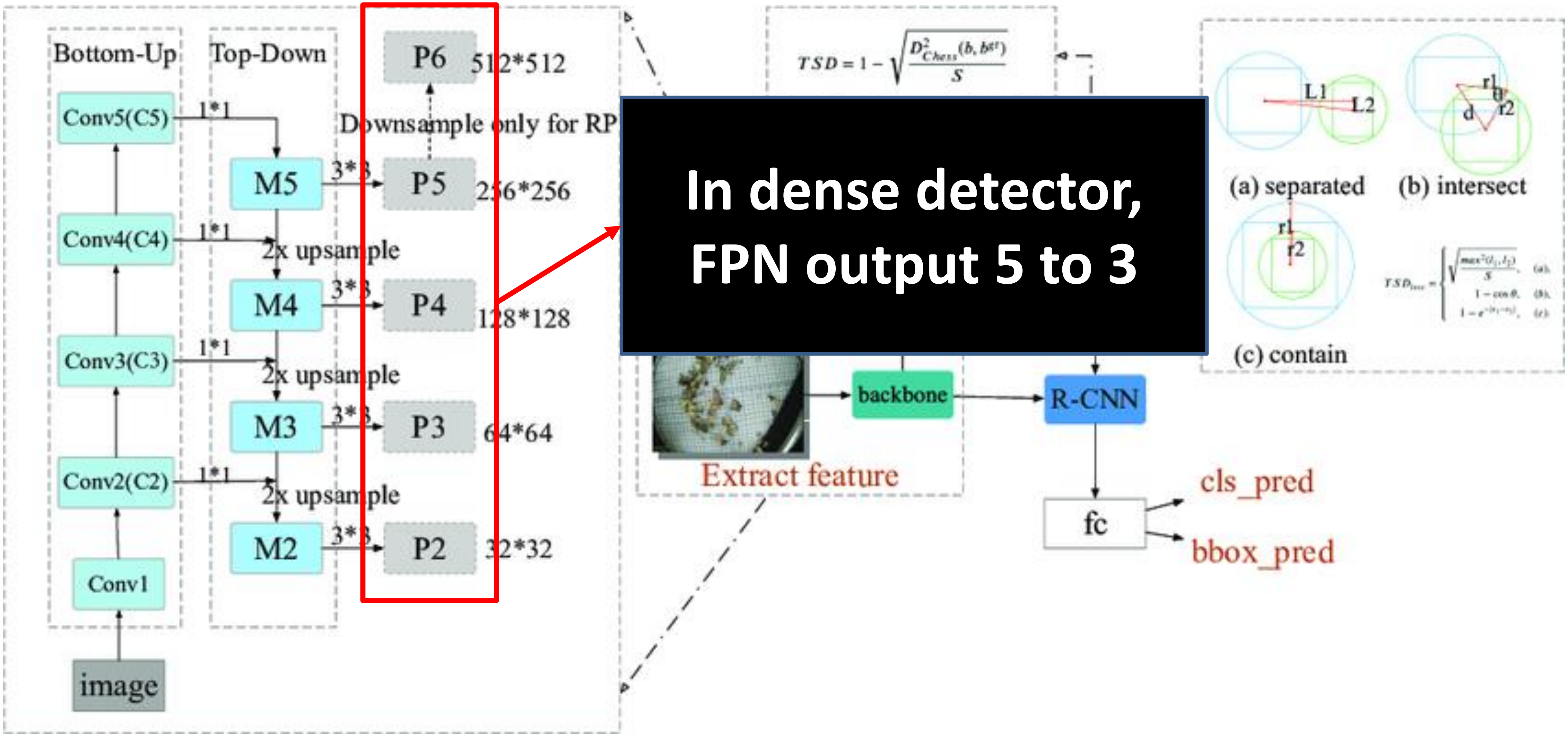


Figure 1. An overview of Efficient Teacher framework. Efficient Teacher proposes three modules to implement a scalable and effective SSOD framework, where Dense Detector improves the quality of pseudo labels with dense input while has better inference efficiency; Pseudo Label Assigner divides pseudo labels into two types to alleviate pseudo labels inconsistency problem; Epoch Adaptor reduces training time and the inconsistency of features.

Dense Detector

- One-stage anchor-based detector baseline
  - YOLO + RetinaNet
- Modified from RetinaNet with ResNet-50-FPN backbone
  - Changing the number of FPN output from 5 to 3
  - Eliminating the weight sharing between detection headers and reducing the input resolution from 1333 to 640
  - Dense Detector output:
    - classification score, bounding-box offset and objectness score



In dense detector, FPN output 5 to 3

Dense Detector

- Dense Detector output:
  - objectness score: Complete Intersection over Union(CIoU)
  - DIoU: Distance-IoU

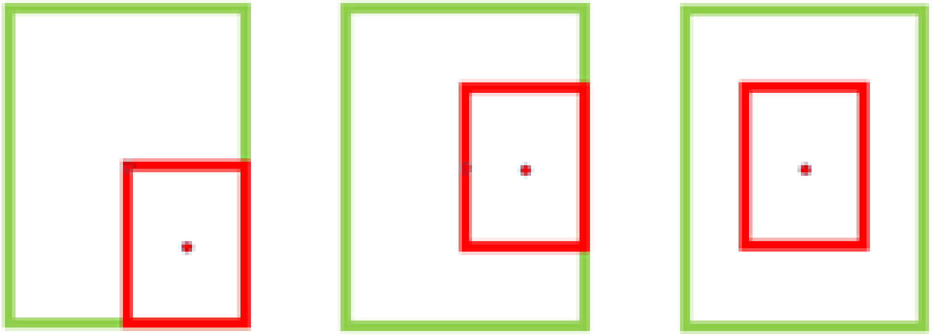


Figure 2: GIoU loss degrades to IoU loss for these cases, while our DIoU loss is still distinguishable. Green and red denote target box and predicted box respectively.

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b},\mathbf{b}^{gt})}{c^2}$$

$$\rho = \text{Euclidean distance}$$

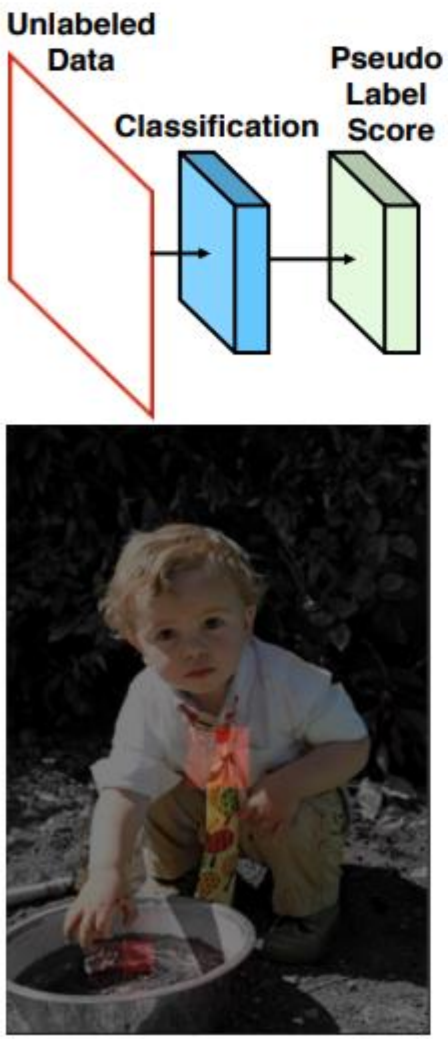
$$\mathbf{b}, \mathbf{b}^{gt} = \text{Bbox center points}$$

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b},\mathbf{b}^{gt})}{c^2} + \alpha v$$

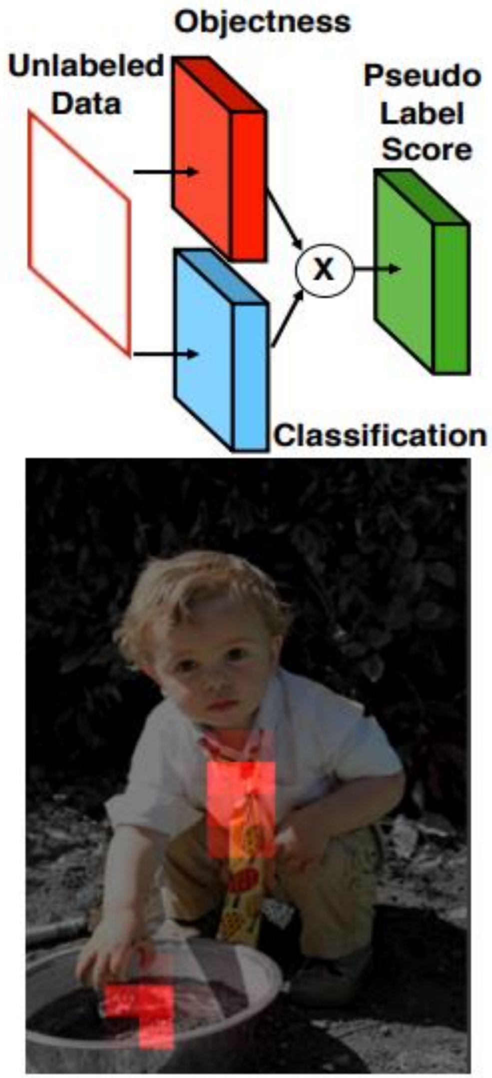
$$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}})^2$$

$$\alpha = \frac{v}{(1-IoU)+v}$$

Dense Detector



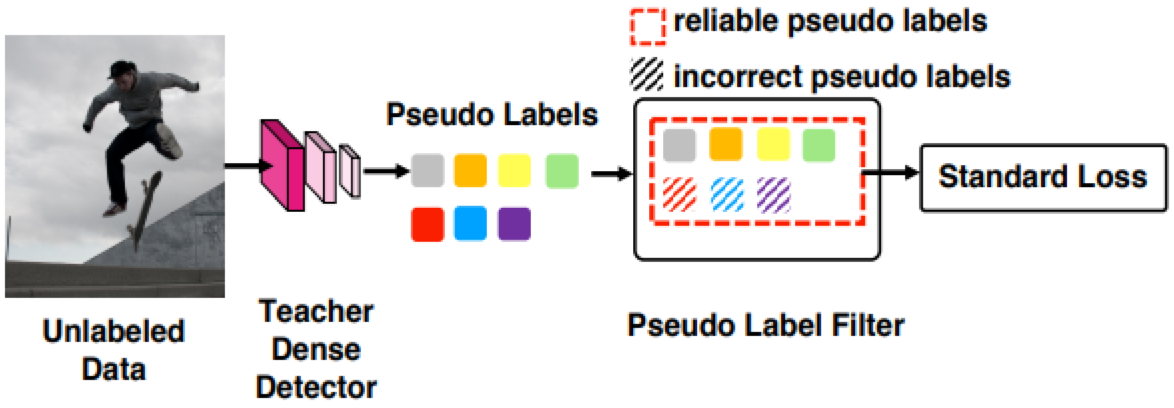
(a) RetinaNet

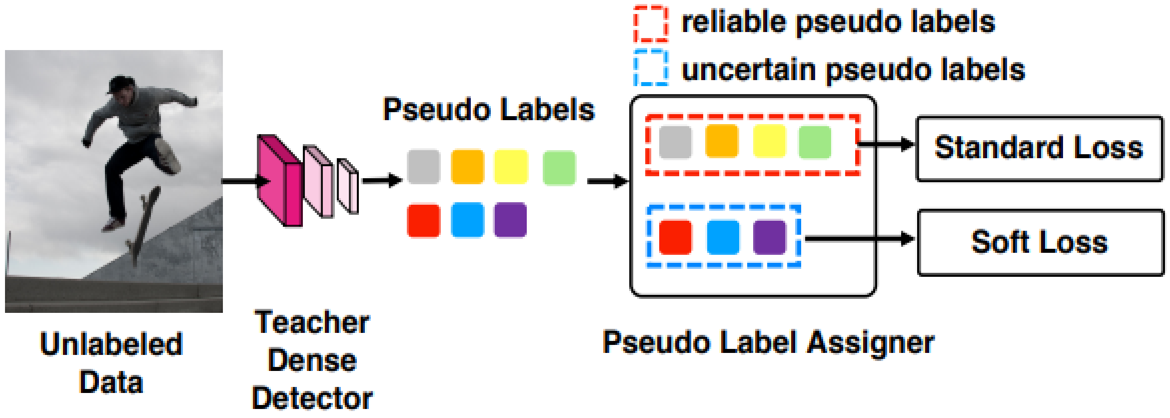


(b) Dense Detector

Pseudo Label Assigner

- Core problem in SSOD
  - Pseudo label filter with setting threshold
  - Pseudo label background: score < threshold
  - Reliable pseudo label: score > threshold





(a) General pseudo label
-fast method
-scores of pseudo labels continue to increase
-treating incorrect pseudo label
-fail to converge in SSOD training

(b) Proposed pseudo label(Pseudo Label Assigner)
-more refined assignment of the pseudo labels
-apply non-maximum suppression
-two categories: reliable and uncertain pseudo label score
- τ_1, τ_2 : high and low thresholds

Loss Function for Pseudo Label Assigner

- SSOD loss function

$$L = L_s + \lambda L_u$$

- Lambda = 3.0
- Supervised loss

$$L_s = \sum_{h,w}(CE(X^{cls}_{(h,w)}, Y^{cls}_{(h,w)}) + CIoU(X^{reg}_{(h,w)}, Y^{reg}_{(h,w)})$$
$$+ CE(X^{obj}_{(h,w)}, Y^{obj}_{h,w}))$$

- Output of student model: $X_{(h,w)}$
- Label assigner: $Y_{(h,w)}$

- Unlabeled loss function

$$L_u = L_u^{cls} + L_u^{reg} + L_u^{obj}$$

$$L_u^{cls} = \sum_{h,w} (\mathbb{1}_{\{p_{(h,w)} >= \tau_2\}} CE(X_{(h,w)}^{cls}, \hat{Y}_{(h,w)}^{cls}))$$

$$L_u^{reg} = \sum_{h,w} (\mathbb{1}_{\{p_{(h,w)} >= \tau_2 \text{ or } \hat{obj}_{(h,w)} > 0.99\}} CIoU(X_{(h,w)}^{reg}, \hat{Y}_{(h,w)}^{reg}))$$

$$L_u^{obj} = \sum_{h,w} (\mathbb{1}_{\{p_{(h,w)} <= \tau_1\}} CE(X_{(h,w)}^{obj}, \mathbf{0})$$

$$+ \mathbb{1}_{\{p_{(h,w)} >= \tau_2\}} CE(X_{(h,w)}^{obj}, \hat{Y}_{(h,w)}^{obj}))$$

$$+ \mathbb{1}_{\{\tau_1 < p_{(h,w)} < \tau_2\}} CE(X_{(h,w)}^{obj}, \hat{obj}_{(h,w)}))$$

Loss Function for Pseudo Label Assigner

- Classification score, regression, objectness score of sampled results from PLA

$$\hat{Y}^{cls}_{(h,w)}, \hat{Y}^{reg}_{(h,w)}, \hat{Y}^{obj}_{(h,w)}$$

Epoch Adaptor

- After pseudo label assigner, still facing challenge in pseudo label inconsistency
  - Lack: stability, high efficiency
  - Lambda: 0.1
  - Domain adaptation

$$L_{da} = -\sum_{h,w}\left[D\log p_{(h,w)} + (1-D)\log(1-p_{(h,w)})\right]$$

$$L_s = \sum_{h,w}(CE(X_{(h,w)}^{cls}, Y_{(h,w)}^{cls}) + CIoU(X_{(h,w)}^{reg}, Y_{(h,w)}^{reg})$$

$$+ CE(X_{(h,w)}^{obj} + Y_{h,w}^{obj})) + \boxed{\lambda L_{da}}$$

Epoch Adaptor

- Disrupting the label distribution ratio from mosaic data augmentation
- To tackle this problem, implement a distribution adaptation method
- From LabelMatch
- Alpha=60, N: number of labeled and unlabeled data
- P=pseudo label score at c-th class at the k-th epoch
- n: number of c-th class ground truth annotations

$$\tau_1^k = P_c^k[n_c^k \cdot \frac{N_u}{N_l}]$$

$$\tau_2^k = P_c^k[\alpha\% \cdot n_c^k \cdot \frac{N_u}{N_l}],$$

Binbin Chen, Weijie Chen, Shicai Yang, Yunyi Xuan, Jie Song, Di Xie, Shiliang Pu, Mingli Song, and Yueting Zhuang. Label matching semi-supervised object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14381–14390, 2022.

Experiments

- Datasets
  - MS-COCO and PASCAL-VOC
- COCO-standard(train2017)
  - 118k labeled images
  - 850k instances from 80 classes
  - 123k unlabeled images
  - Randomly sample 1, 5, and 10% of labeled training data as **a labeled set**
  - Rest of labeled data as an unlabeled set
  - 1%: 1.2k images
  - Dataset split strategy for semi-supervised learning from STAC(Self-Training and the Augmentation driven Consistency regularization)

Sohn, Kihyuk, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee and Tomas Pfister. "**A Simple Semi-Supervised Learning Framework for Object Detection**." *ArXiv* abs/2005.04757 (2020): n. pag.

Experiments

- PASCAL-VOC
  - VOC07 trainval: 5,011 training images from 20 classes as **a labeled set**
  - VOC12 trainval: 11,540 training images as **an unlabeled set**
  - Validation sets: COCO val2017 and VOC07 test set, respectively
- COCO-additional
  - Train2017-unlabeled data: 123k
- Details
  - NVIDIA-V100 * 8EA
  - Randomly sample 32 images from labeled and unlabeled data respectively
  - 300 epoch, 0.999 EMA

Experiments

- PASCAL-VOC
  - VOC07 trainval: 5,011 training images from 20 classes as **a labeled set**
  - VOC12 trainval: 11,540 training images as **an unlabeled set**
  - Validation sets: COCO val2017 and VOC07 test set, respectively
- Network
  - ResNet-50-FPN backbone in Dense Detector
  - Original backbone with CSPNet and Neck with PAN
  - Initial weight: pre-trained on ImageNet
- Details
  - NVIDIA-V100 * 8EA
  - Randomly sample 32 images from labeled and unlabeled data respectively
  - 300 epoch, 0.999 EMA

Experiments

Method		%1	%2	%5	%10	FLOPs
Two-stage anchor-based	Supervised	9.05	12.70	18.47	23.86	202.31G
	STAC [27]	13.97 ± 0.35(+4.92)	18.25 ± 0.25 (+5.91)	24.38 ± 0.12 (+5.91)	28.64 ± 0.21 (+4.78)	202.31G
	Instant Teaching [40]	18.05 ± 0.15 (+9.00)	22.45 ± 0.15 (+9.75)	26.75 ± 0.05 (+8.28)	30.40 ± 0.05 (+6.54)	202.31G
	Humber teacher [29]	16.96 ± 0.38 (+7.91)	21.72 ± 0.24 (+9.02)	27.70 ± 0.15 (+9.23)	31.61 ± 0.28 (+7.75)	202.31G
	Unbiased Teacher [21]	20.75 ± 0.12 (+11.70)	24.30 ± 0.07 (+9.80)	28.27 ± 0.11 (+9.80)	31.50 ± 0.10 (+7.64)	204.13G
	Soft Teacher [35]	20.46 ± 0.39 (+11.41)	-	30.74 ± 0.08 (+12.27)	34.04 ± 0.14 (+10.18)	202.31G
	LabelMatch [4]	**25.81** ± 0.28 (+16.76)	-	32.70 ± 0.18 (+14.23)	35.49 ± 0.17 (+11.63)	202.31G
	PseCo [17]	22.43 ± 0.36 (+13.38)	27.77 ± 0.18 (+15.07)	32.50 ± 0.08 (+14.03)	36.06 ± 0.24 (+12.20)	202.31G
One-stage anchor-free	Supervised	9.53	11.71	18.74	23.70	200.59G
	Unbiased Teacher v2 [22]	22.71 ± 0.42 (+13.18)	26.03 ± 0.12 (+14.32)	30.08 ± 0.04 (+11.34)	32.61 ± 0.03 (+8.91)	200.59G
	DSL [5]	22.03 ± 0.28 (+12.50)	25.19 ± 0.37 (+13.48)	30.87 ± 0.24 (+12.13)	36.22 ± 0.18 (+12.52)	200.59G
	Dense Teacher [39]	22.38 ± 0.31 (+12.85)	27.20 ± 0.20 (+15.49)	33.01 ± 0.21 (+14.27)	37.13 ± 0.12 (+13.43)	200.59G
One-stage anchor-based	Supervised	10.29	13.12	19.28	24.04	169.61G
	Unbiased Teacher∗ [21]	18.81 ± 0.28 (+9.07)	22.72 ± 0.21 (+9.60)	28.35 ± 0.12 (+8.15)	30.34 ± 0.09 (+6.30)	169.61G
	Ours	21.51 ± 0.21 (+11.22)	27.15 ± 0.13 (+14.03)	31.1 ± 0.08 (+11.82)	34.09 ± 0.11 (+10.05)	169.61G
	Ours †	23.76 ± 0.13 (+12.47)	**28.70** ± 0.14 (+15.58)	**34.11** ± 0.09 (+14.83)	**37.90** ± 0.04 (+13.86)	109.59G

Table 2. Experimental results on COCO-standard ($AP_{50:95}$), ∗ means re-implemented results on Dense Detector, † means Efficient Teacher with YOLOv5l [14]. All the results are the average of 5 folds.

Experiments

Method	$AP_{50:95}$
Supervised †	49.0
Ours †	**50.45(+1.45)**

Table 3. Experimental results on COCO-additional.

Method	$AP_{50:95}$	AP_{50}	FLOPs
STAC [27]	44.64	77.45	202.31G
Instant Teacher [40]	50.00	79.20	202.31G
Unbiased Teacher [21]	48.69	77.37	204.13G
Dense Teacher [39]	55.87	79.89	200.59G
DSL [5]	56.80	80.70	200.59G
Unbiased Teacher v2 [22]	56.87	81.29	200.59G
LabelMatch [4]	55.11	85.48	202.31G
Ours †	58.30	81.60	109.59G
Ours ‡	**60.56**	**86.54**	109.59G

Table 4. Experimental results on PASCAL-VOC. The ‡ indicates using a ImageNet pre-trained backbone to initialize the Efficient Teacher

Experiments

- Threshold: 0.3 for pseudo label

Method	$AP_{50:95}$	AP_{50}
Supervised	30.45	44.65
Unbiased Teacher [21]	32.10 (+1.65)	47.30 (+2.65)
Ignore uncertain pseudo label [5]	35.20 (+4.75)	52.00 (+7.35)
Pseudo Label Assigner	**37.90 (+7.45)**	**54.19 (+9.54)**

Table 5. Ablation study about different pseudo label assignment methods.

τ_2	$AP_{50:95}$	AP_{50}
0.4	37.20	54.08
0.5	37.20	54.10
0.6	36.90	53.77
0.7	35.10	51.60
EA	**37.90**	**54.80**

Table 6. Ablation studies on threshold value τ_2 , EA indicates τ_2 is calculated by Epoch Adaptor.

Method	$AP_{50:95}$	AP_{50}
w/o domain adaptation	37.25	54.16
domain adaptation	**37.90**	**54.80**

Table 7. Ablation studies on domain adaptation in EA.

- Epoch adaptor for efficient and effective approach

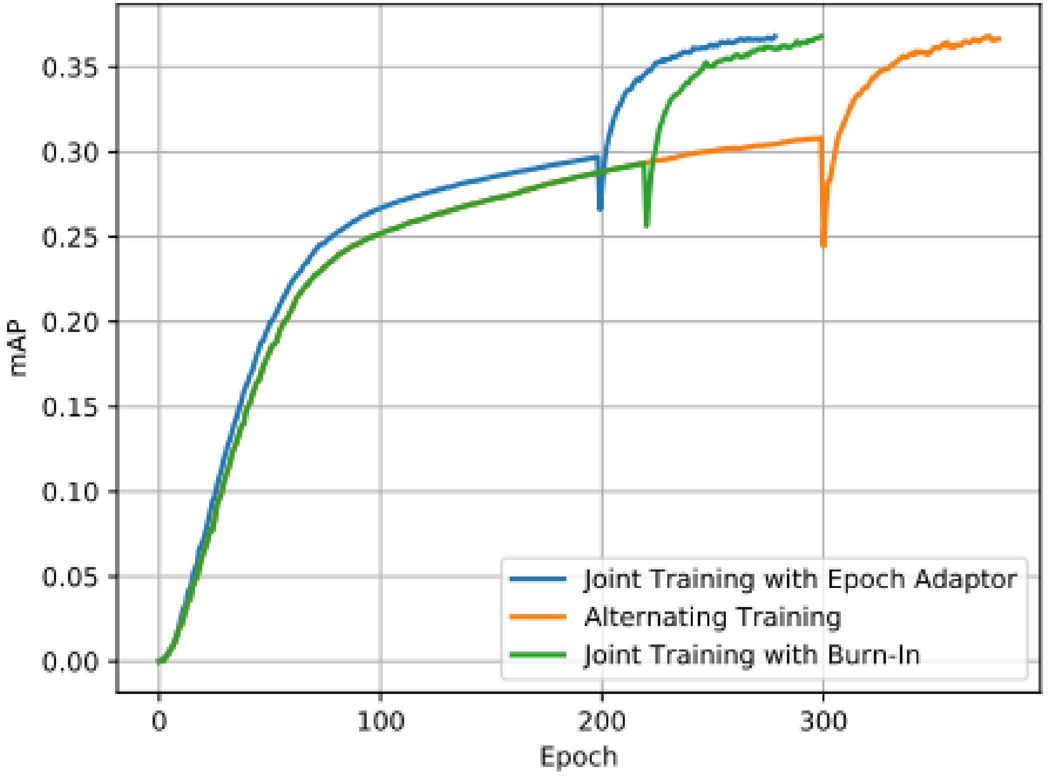


Figure 5. Performance ($AP_{50:95}$) comparisons of Epoch Adaptor, Alternating Training and Joint Training with Burn-In methods on COCO standard 10%.

Conclusion

- Proposed efficient teacher to bridge the gap between SSOD and one-stage anchor based detectors
- Dense detector for efficient and quality of pseudo label
- Pseudo label assigner for inconsistency of pseudo label
- Epoch adaptor for domain and distribution adaptation

Experiments

- **STAC**: SSL for object detection(**Self-Training** and the **Augmentation** driven **Consistency** regularization)
- **CSD**: **C**onsistency-based **s**emi-supervised learning for object **d**etection
- **Instant-Teaching**: An end-to-end semi-supervised object detection framework

Methods	Backbone	1% COCO	2% COCO	5% COCO	10% COCO	100% COCO
Supervised	R50-FPN	9.05±0.16	12.70±0.15	18.47±0.22	23.86±0.81	37.63
CSD† [22]	R50-FPN	10.20±0.15 (+1.15)	13.60±0.10 (+0.90)	18.90±0.10 (+0.43)	24.50±0.15 (+0.64)	38.87 (+1.24)
STAC[45]	R50-FPN	13.97±0.35 (+4.92)	18.25±0.25 (+5.55)	24.38±0.12 (+5.91)	28.64±0.21 (+4.78)	39.21 (+1.58)
Instant-Teaching (ours)	R50-FPN	**16.00±0.20** (+6.95)	**20.70±0.30** (+8.00)	**25.50±0.05** (+7.03)	**29.45±0.15** (+5.59)	**39.60** (+1.97)
Instant-Teaching* (ours)	R50-FPN	**18.05±0.15** (+9.00)	**22.45±0.15** (+9.75)	**26.75±0.05** (+8.28)	**30.40±0.05** (+6.54)	**40.20** (+2.57)

Table 1. Comparison of mAP for different semi-supervised methods on MS-COCO. CSD† is our implementation of the CSD method based on the Faster-RCNN detector. Instant-Teaching* represents our Instant-Teaching framework with co-rectify scheme. The value in brackets represents the mAP improvement compared to the supervised model.

Experiments

- **STAC**: SSL for object detection(**Self-Training** and the **Augmentation** driven **Consistency** regularization)
- **CSD**: **C**onsistency-based **s**emi-supervised learning for object **d**etection
- **Instant-Teaching**: An end-to-end semi-supervised object detection framework

Methods	Backbone	Unlabeled	$AP^{0.5:0.95}$	$AP^{0.5}$	$AP^{0.75}$
Supervised (Ours)	R50-FPN		43.60	76.70	44.50
CSD [22]	R101-R-FCN	VOC12	-	74.70	-
STAC [45]	R50-FPN		44.64 (+1.04)	77.45	-
Instant-Teaching	R50-FPN		**48.70** (+5.10)	**78.30**	**52.00** (+7.50)
Instant-Teaching*	R50-FPN		**50.00** (+6.40)	**79.20**	**54.00** (+9.50)
CSD [22]	R101-R-FCN	VOC12 & COCO	-	75.10	-
STAC [45]	R50-FPN		46.01 (+2.41)	79.08	-
Instant-Teaching	R50-FPN		**49.70** (+6.10)	79.00	**54.10** (+9.60)
Instant-Teaching*	R50-FPN		**50.80** (+7.20)	**79.90**	**55.70** (+11.20)

Table 2. Comparison of mAP for different semi-supervised methods on VOC07. We report the mAP at IoU=0.50:0.95 ($AP^{0.5:0.95}$), IoU=0.5 ($AP^{0.5}$) and IoU=0.75 ($AP^{0.75}$), which are the standard metrics for object detection [31, 7].

Methods	Strong data augmentations				mAP
	Color+Cutout	Geometric	Mixup	Mosaic	
STAC[45]	√	√			23.14
Instant-Teaching	√*				21.60 (-1.54)
	√				24.70 (+1.56)
	√		√		25.40 (+2.26)
	√			√	25.00 (+1.86)
	√		√	√	**25.60** (+2.46)

Table 3. Comparison of mAP of Instant-Teaching trained with various data augmentation methods at the protocol of 5% MS-COCO and 8× unlabeled data. √* denotes that we also apply strong augmentations "Color+Cutout" to unlabeled data in the first step during instant pseudo labeling.

Methods	Labeled Size	Unlabeled Size				
		1×	2×	4×	8×	Full
STAC[45]	5% COCO	19.81	20.79	22.09	23.14	24.38±0.12
Instant-Teaching	5% COCO	**23.60**	**24.30**	**25.30**	**25.60**	**25.60±0.14**
STAC[45]	10% COCO	25.38	26.52	27.33	27.95	28.64±0.21
Instant-Teaching	10% COCO	**28.80**	**29.00**	**29.20**	**29.50**	**29.53±0.17**

Table 4. Comparison of mAP of Instant-Teaching trained with various scales of unlabeled data on MS-COCO. $[n]\times$ denotes the scale of unlabeled data is $[n]$ times larger than that of labeled data.

Experiments

τ	0.3	0.5	0.7	0.9
mAP (%)	26.30	27.70	28.70	29.80

Table 5. Comparison of mAP with various values of confidence threshold τ.

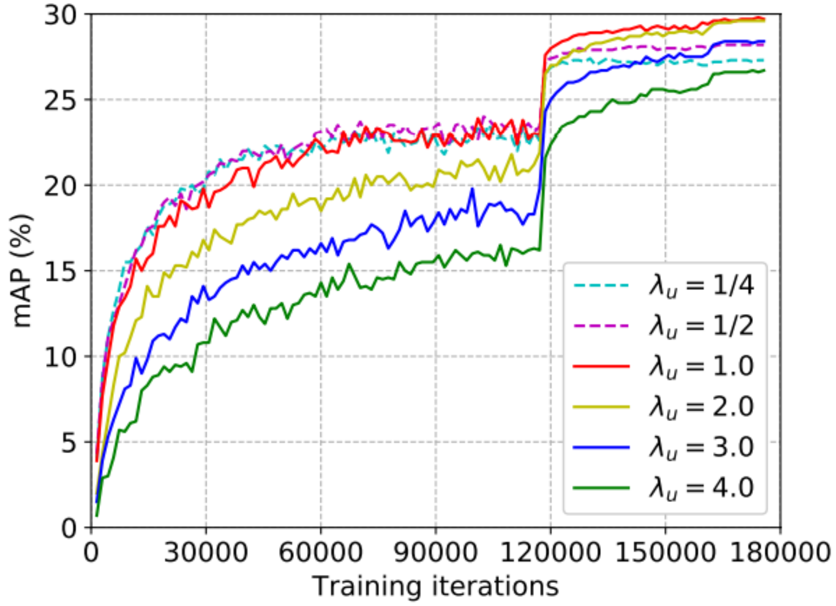


Figure 5. Comparison of mAP with various values of λ_u along training iterations.

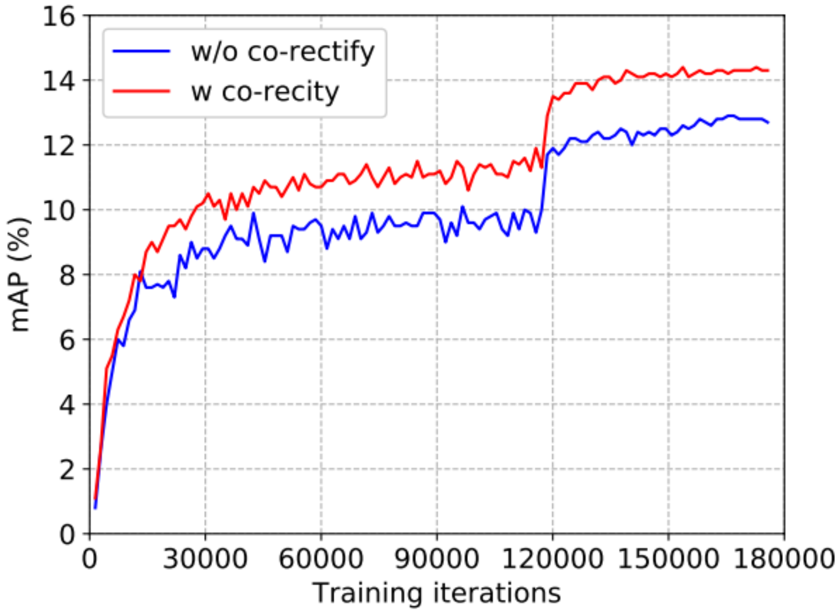


Figure 6. Comparison of mAP of generated pseudo annotations with different training iterations. The model is trained based on Instant-Teaching with and without co-rectify respectively.

- To encourage the model to learn useful information from pseudo label

1. Augmentation: MixUp

- Soft class label[(example)]

$$\begin{cases} \lambda_m & \sim & Beta(\alpha_m, \alpha_m), \\ \mathbf{x}_u & = & \lambda_m \mathbf{x}_u + (1-\lambda_m)\mathbf{x}_l, \\ c_u & = & \lambda_m c_u \cup (1-\lambda_m) c_l, \\ b_u & = & b_u \cup b_l. \end{cases}$$

mixing coefficient: λ_m

unlabeled image: $\mathbf{x}_u$

ground truth image: $\mathbf{x}_l$

bbox of pseudo box: b_u

bbox of ground truth: b_l

class of pseudo box: c_u

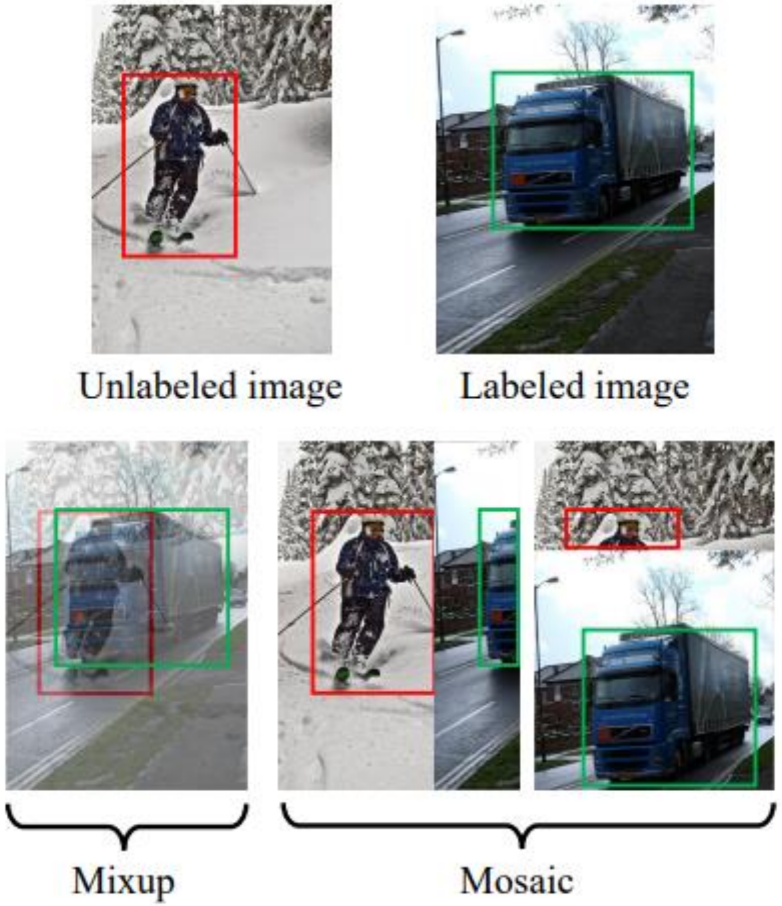
class of ground truth box: c_l

```
import numpy as np

lamb = np.random.beta(1,1)
print(lamb)
```

```
문제    출력    터미널

PS C:\Users\dbfru>
0.6575339009847244
PS C:\Users\dbfru>
0.182241651825588
PS C:\Users\dbfru>
0.184061643768410
PS C:\Users\dbfru>
0.88325354364798
PS C:\Users\dbfru>
0.9777441574350441
PS C:\Users\dbfru>
```



MixUp Augmentation

- MixUp: improving data generalization


Original


MixUp


Cutout


CutMix

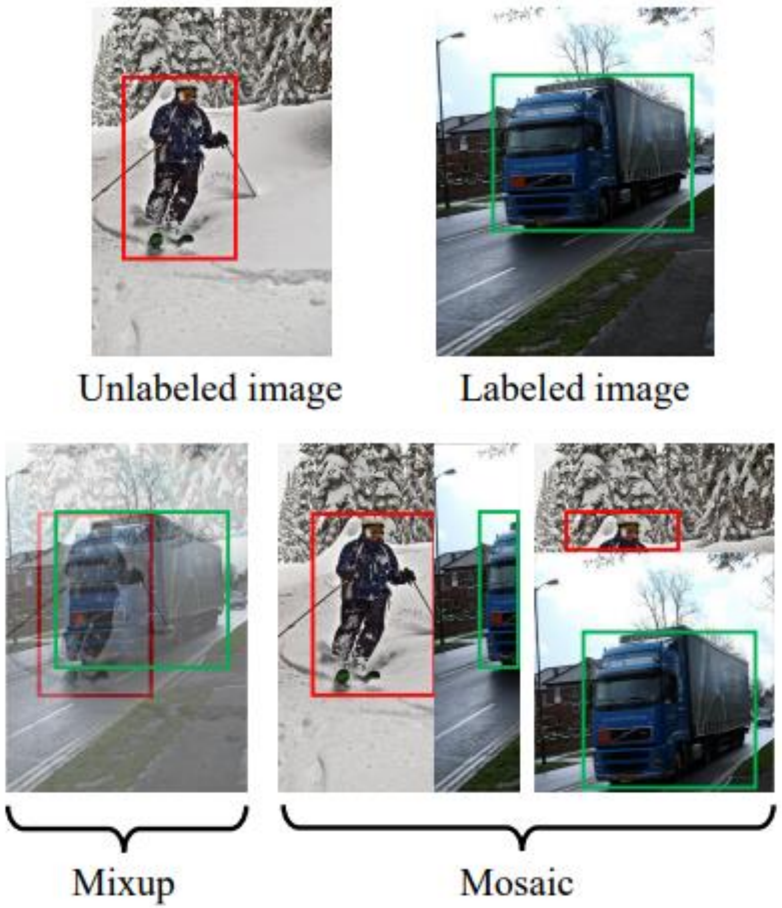
Zhang, Hongyi et al. "mixup: Beyond Empirical Risk Minimization." *ArXiv* abs/1710.09412 (2017): n. pag.

Weak-strong data augmentations(2/3)

- To encourage the model to learn useful information from pseudo label

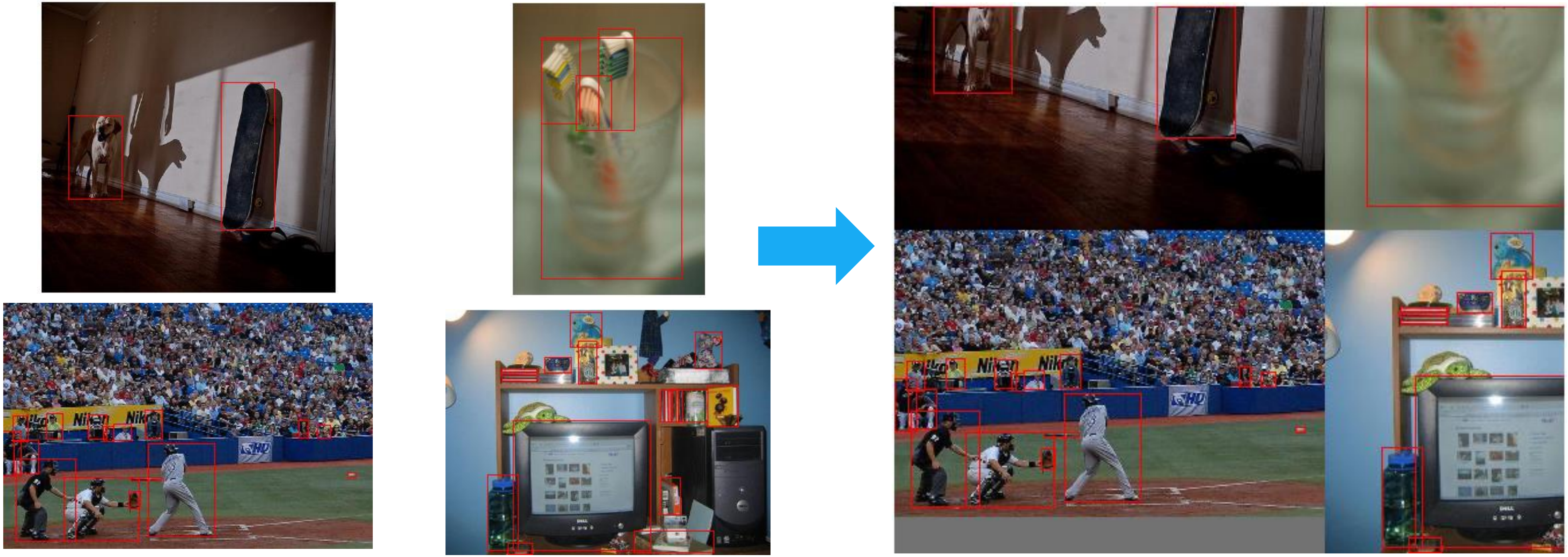
2. Augmentation: Mosaic
   - Randomly mixing styles: horizontal and vertical mixing

- Using both data augmentations, robusting the model for overfitting problem



Mosaic Augmentation

- Mosaic data augmentation: supposed by YOLOv4
- Mixed with 4 training images and 4 different contexts
- Efficient for number of batch size: 4 images -> 1 image



Bochkovskiy, Alexey et al. “YOLOv4: Optimal Speed and Accuracy of Object Detection.” *ArXiv* abs/2004.10934 (2020): n. pag.

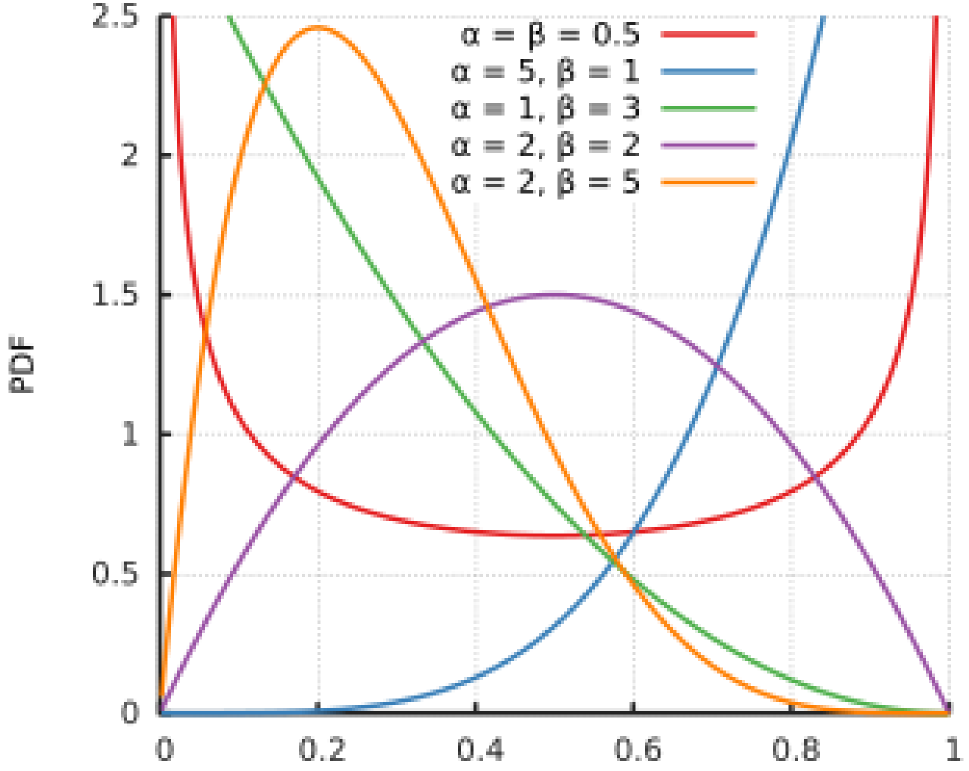
Weak-strong data augmentations, Code for Mixup and Mosaic(3/3)

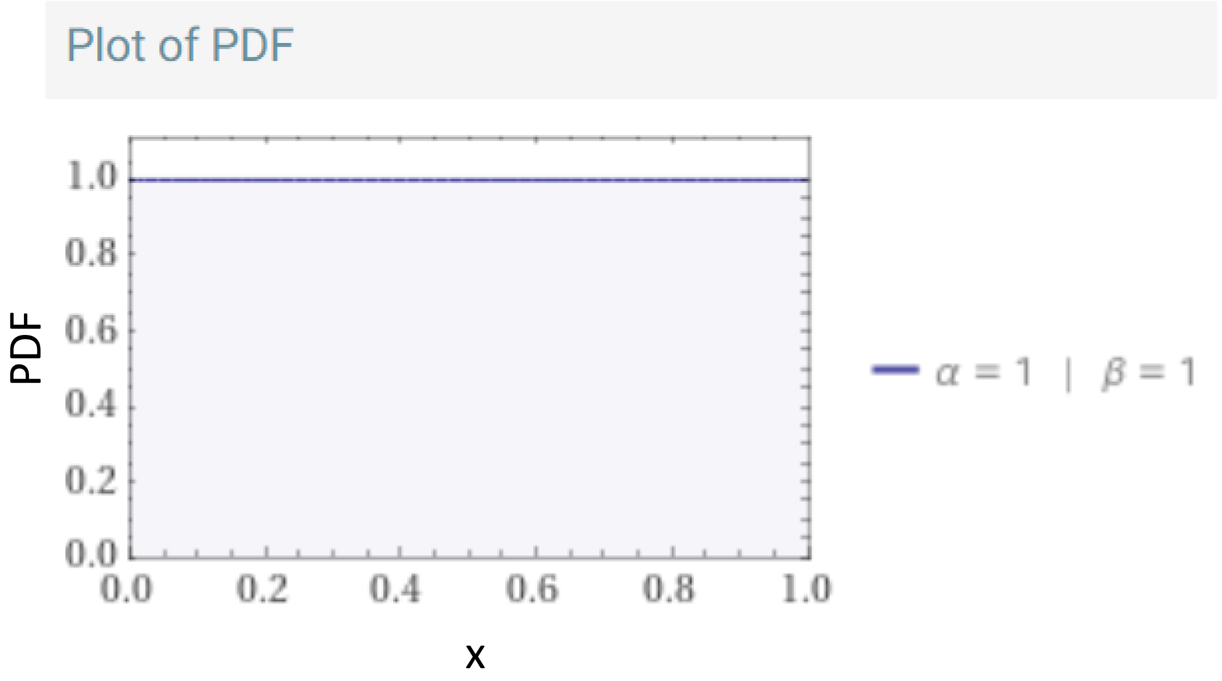
```python
if aug_type == 'ssl_with_mixup':
    lamb = np.random.beta(alpha, alpha)
    _img = lamb * img_s[[i_unlabel], ...] + (1 - lamb) * img_s[[j_label], ...]
    _gt_labels = torch.cat((gt_labels_s[i_unlabel] * lamb, gt_labels_s[j_label] * (1 - lamb)))
    _gt_bboxes = torch.cat((gt_bboxes_s[i_unlabel], gt_bboxes_s[j_label]))
elif aug_type == 'ssl_with_mosaic':
    _, _, _h, _w = img_s.shape
    _img = img_s[[i_unlabel], ...]
    if np.random.randint(0, 2) == 1:    ## split top-down
        cy = np.random.randint(_h // 4, _h // 4 * 3)
        _img[0, :, cy:, :] = img_s[[j_label], :, cy:, :]
        gt_bboxes_i, gt_labels_i = self._clip_gt(gt_bboxes_s[i_unlabel], gt_labels_s[i_unlabel], 0, 0, _w, cy)
        gt_bboxes_j, gt_labels_j = self._clip_gt(gt_bboxes_s[j_label], gt_labels_s[j_label], 0, cy, _w, _h)
        _gt_bboxes = torch.cat((gt_bboxes_i, gt_bboxes_j))
        _gt_labels = torch.cat((gt_labels_i, gt_labels_j))
    else:                               ## split left-right
        #cx = int(gt_bboxes[i][np.random.choice(list(range(len(gt_labels[i]))))][2])
        cx = np.random.randint(_w // 4, _w // 4 * 3)
        _img[0, :, :, cx:] = img_s[[j_label], :, :, cx:]
        gt_bboxes_i, gt_labels_i = self._clip_gt(gt_bboxes_s[i_unlabel], gt_labels_s[i_unlabel], 0, 0, cx, _h)
        gt_bboxes_j, gt_labels_j = self._clip_gt(gt_bboxes_s[j_label], gt_labels_s[j_label], cx, 0, _w, _h)
        _gt_bboxes = torch.cat((gt_bboxes_i, gt_bboxes_j))
        _gt_labels = torch.cat((gt_labels_i, gt_labels_j))
return _img, _gt_bboxes, _gt_labels
```

https://github.com/txdet/Instant-Teaching/blob/d07910c4c811d875b03200ffb1822c32556ccf9a/projects/InstantTeaching/models/detectors/instant_teaching.py#L36

- PDF : $f(x;\alpha,\beta) = constant \cdot x^{\alpha-1}(1-x)^{\beta-1}$

$$1 = \int_0^1 \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}dx \qquad B(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx$$





Co-rectify

- In semi-supervised learning, common problem → Confirmation bias
- Affecting the performance of model how to choose the unlabeled data

1. Same structure of models but different initialization(Model-a: $f_a(\cdot)$, Model-b: $f_b(\cdot)$)
2. Sharing same data in each batch but different data aug and pseudo annotations
3. Follows below:

$$\begin{cases} (c_i, \mathbf{t}_i) & = & f_a(\mathbf{x}_u), \\ (c_i^r, \mathbf{t}_i^r) & = & f_b(\mathbf{x}_u; \mathbf{t}_i), \\ c_i^* & = & \frac{1}{2}(c_i + c_i^r), \\ \mathbf{t}_i^* & = & \frac{1}{c_i + c_i^r}(\mathbf{t}_i c_i + \mathbf{t}_i^r c_i^r). \end{cases}$$

unlabeled image: $\mathbf{x}_u$
bounding box coordinates: $\mathbf{t}_i$
refined bbox coordinates: $\mathbf{t}_i^r$
rectify bbox coordinates: $\mathbf{t}_i^*$
class probabilities: $\mathbf{c}_i$
refine class probabilities: $\mathbf{c}_i^r$
rectify class probabilities: $\mathbf{c}_i^*$

Confirmation Bias

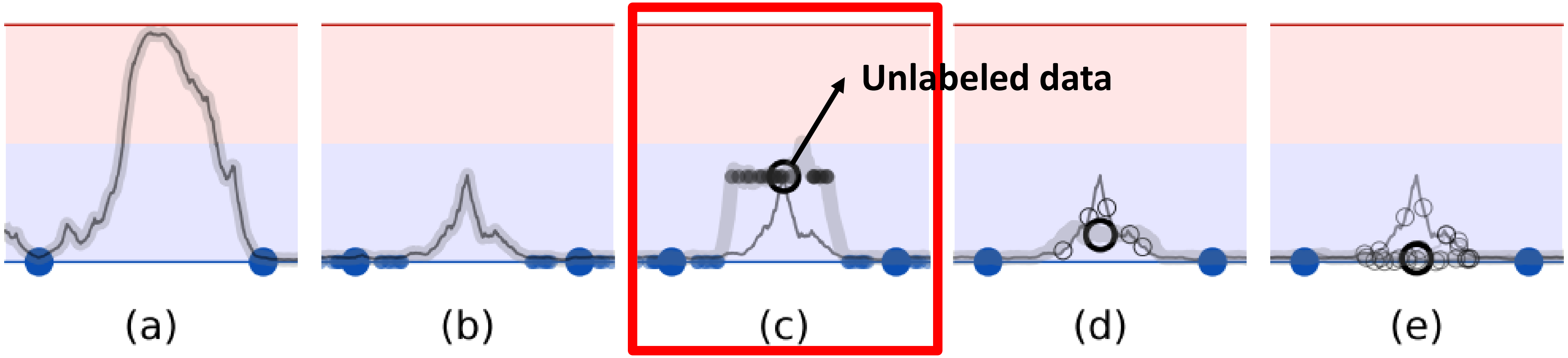


Figure 1: A sketch of a binary classification task with two labeled examples (large blue dots) and one unlabeled example, demonstrating how the choice of the unlabeled target (black circle) affects the fitted function (gray curve). **(a)** A model with no regularization is free to fit any function that predicts the labeled training examples well. **(b)** A model trained with noisy labeled data (small dots) learns to give consistent predictions around labeled data points. **(c)** Consistency to noise around unlabeled examples provides additional smoothing. For the clarity of illustration, the teacher model (gray curve) is first fitted to the labeled examples, and then left unchanged during the training of the student model. Also for clarity, we will omit the small dots in figures d and e. **(d)** Noise on the teacher model reduces the bias of the targets without additional training. The expected direction of stochastic gradient descent is towards the mean (large blue circle) of individual noisy targets (small blue circles). **(e)** An ensemble of models gives an even better expected target. Both Temporal Ensembling and the Mean Teacher method use this approach.

Tarvainen, Antti and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results." ***NIPS*** **(2017).**

Instant-Teaching: An End-to-End Semi-Supervised Object Detection Framework

Presented by: Youlkyeong Lee {yklee@islab.ulsan.ac.kr}

Published by CVPR2021

Author

- Alibaba Group
  - Industry: E-commerce, cloud, computing, etc.
  - Foundation date: 28 June 1999
  - Founder: Jack Ma
  - Owner: SoftBank Group(23.9%)
  - Location: China, Hangzhou

Alibaba Group
阿里巴巴集团

Motivation

- Founded the problem of STAC
  - **STAC**: SSL for object detection(**Self-Training** and the **Augmentation** driven **Consistency** regularization, 2020)
  1. Training procedure: Complicate and inefficient
     - Needs: Teacher model, pseudo label
  2. No longer updating the pseudo annotations
     - Limited performance with the constant label
- Proposed a novel end-to-end SSOD framework
  - Generating instant pseudo label with data augmentations(Mosaic and MixUp)
  - Single model: Model-a

Overview Framework

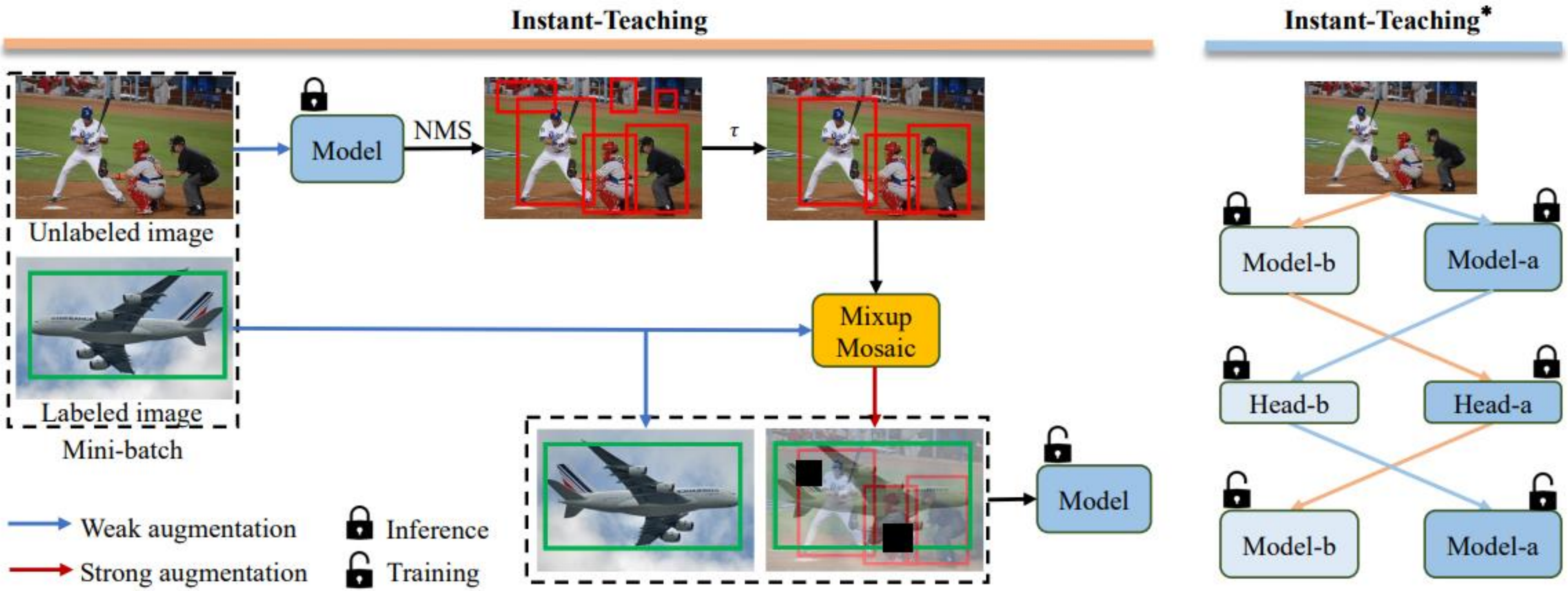


Figure 1. The proposed semi-supervised object detection framework. Instant-Teaching includes instant pseudo labeling with extended weak-strong data augmentations. Instant-Teaching* represents Instant-Teaching combined with our co-rectify scheme.

Total Loss

- Jointly minimizing the supervised loss and unsupervised loss

$$\ell = \ell_s + \lambda_u \ell_u$$

- Supervised loss

$$\ell_s = \sum_l [\frac{1}{N_{cls}} \sum_i L_{cls}(p(c_i \mid \alpha(\mathbf{x}_l)), c_i^*) + \frac{\lambda}{N_{reg}} \sum_i c_i^* L_{reg}(p(\mathbf{t}_i \mid \alpha(\mathbf{x}_l)), \mathbf{t}_i^*)].$$

weak aug: $\alpha(\cdot)$

Ground truth

- Unsupervised loss

hard label, $\hat{c}_i^u = \mathrm{argmax}(c^u)$

strong aug: $A(\cdot)$

$$\ell_u = \sum_u [\frac{1}{N_{cls}} \sum_i L_{cls}(p(c_i \mid A(\mathbf{x}_u)), \hat{c}_i^u) + \frac{\lambda}{N_{reg}} \sum_i (\max(c_i^u) \geq \tau) L_{reg}(p(\mathbf{t}_i \mid A(\mathbf{x}_u)), \mathbf{t}_i^u)],$$

confidence: $\tau > 0.9$

- To encourage the model to learn useful information from pseudo label

1. Augmentation: MixUp

- Soft class label(example)

```python
import numpy as np

lamb = np.random.beta(1,1)
print(lamb)
```

문제　출력　터미널

```
PS C:\Users\dbfru>
0.6575339009847244
PS C:\Users\dbfru>
0.18224165182558857
PS C:\Users\dbfru>
0.18406164376841033
PS C:\Users\dbfru>
0.88325354364798З
PS C:\Users\dbfru>
0.9777441574350441
PS C:\Users\dbfru>
```

$$
\begin{cases}
\lambda_m & \sim & Beta(\alpha_m, \alpha_m), \\
\mathbf{x}_u & = & \lambda_m \mathbf{x}_u + (1-\lambda_m)\mathbf{x}_l, \\
c_u & = & \lambda_m c_u \cup (1-\lambda_m) c_l, \\
b_u & = & b_u \cup b_l.
\end{cases}
$$

mixing coefficient: λ_m

unlabeled image: $\mathbf{x}_u$

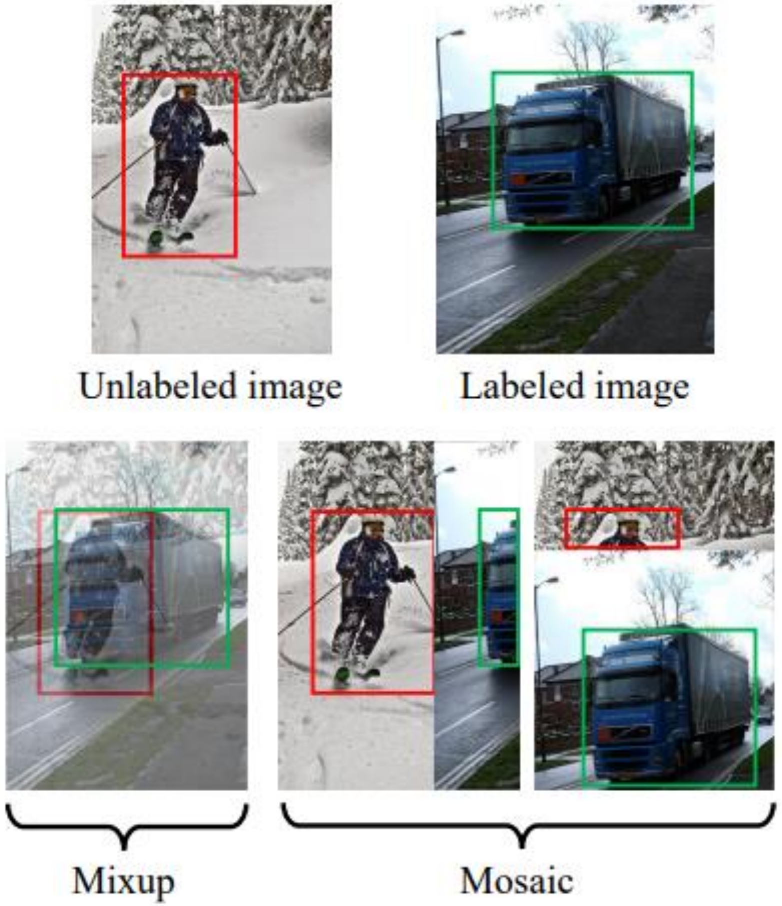
ground truth image: $\mathbf{x}_l$

bbox of pseudo box: b_u

bbox of ground truth: b_l

class of pseudo box: c_u

class of ground truth box: c_l



- MixUp: improving data generalization

Original

MixUp

Cutout

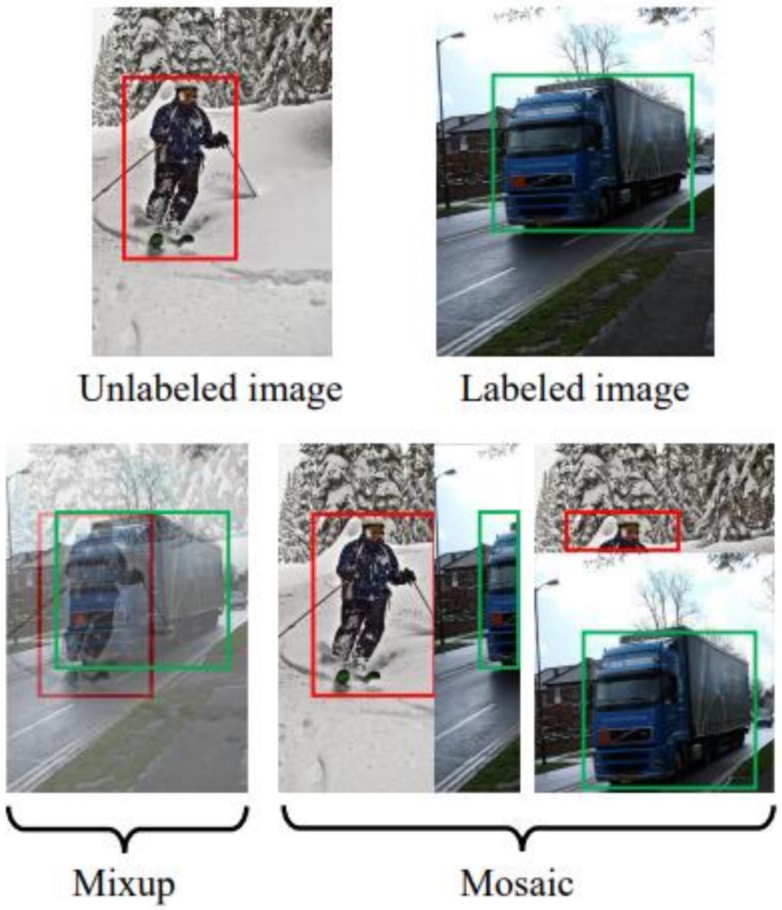
CutMix

Zhang, Hongyi et al. "mixup: Beyond Empirical Risk Minimization." *ArXiv* abs/1710.09412 (2017): n. pag.

- To encourage the model to learn useful information from pseudo label

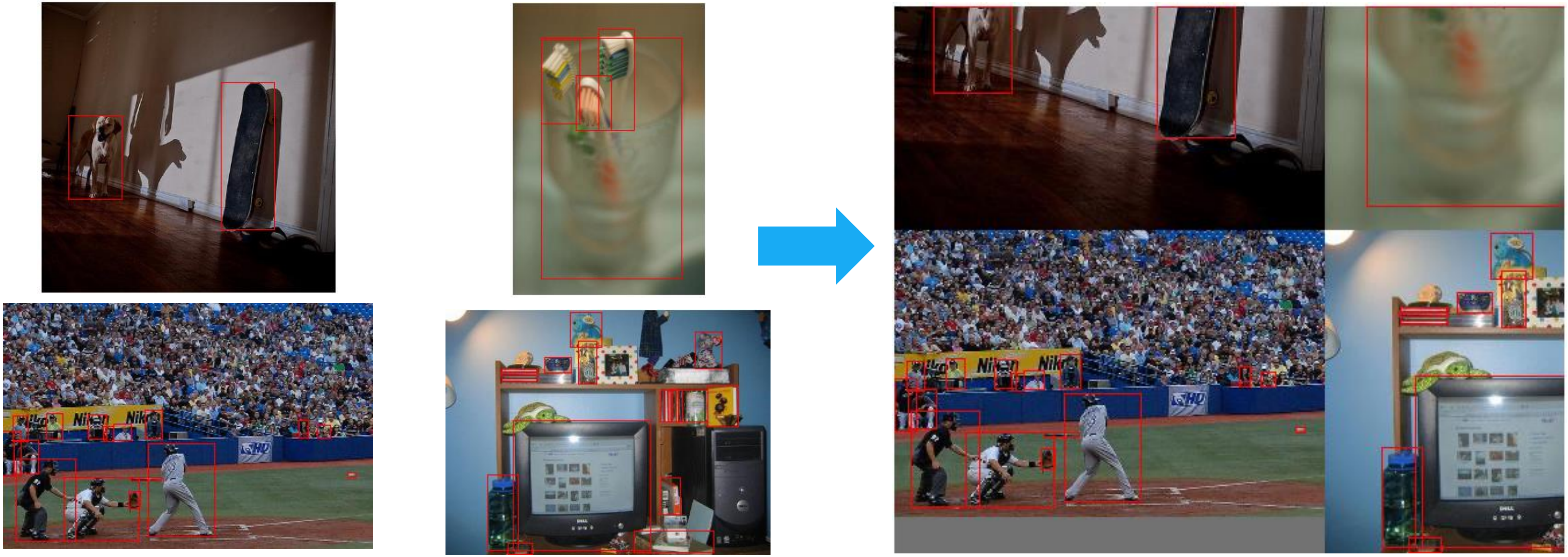
2. Augmentation: Mosaic
   - Randomly mixing styles: horizontal and vertical mixing

- Using both data augmentations, robusting the model for overfitting problem



Mosaic Augmentation

- Mosaic data augmentation: supposed by YOLOv4
- Mixed with 4 training images and 4 different contexts
- Efficient for number of batch size: 4 images -> 1 image



Bochkovskiy, Alexey et al. "YOLOv4: Optimal Speed and Accuracy of Object Detection." *ArXiv* abs/2004.10934 (2020): n. pag.

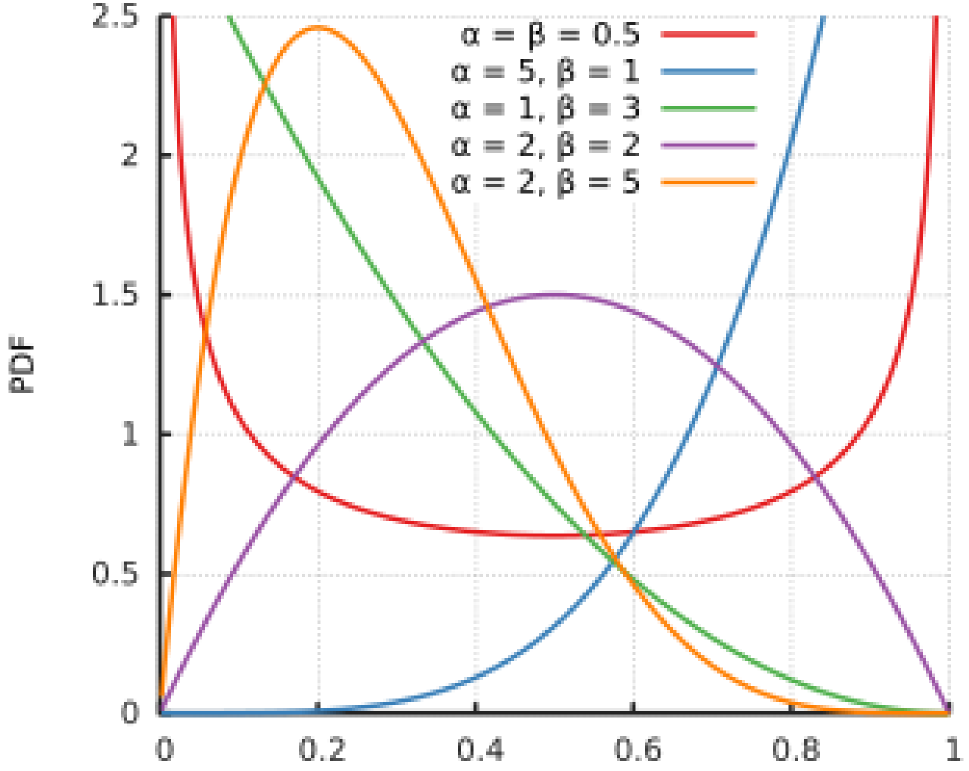
Weak-strong data augmentations, Code for Mixup and Mosaic(3/3)

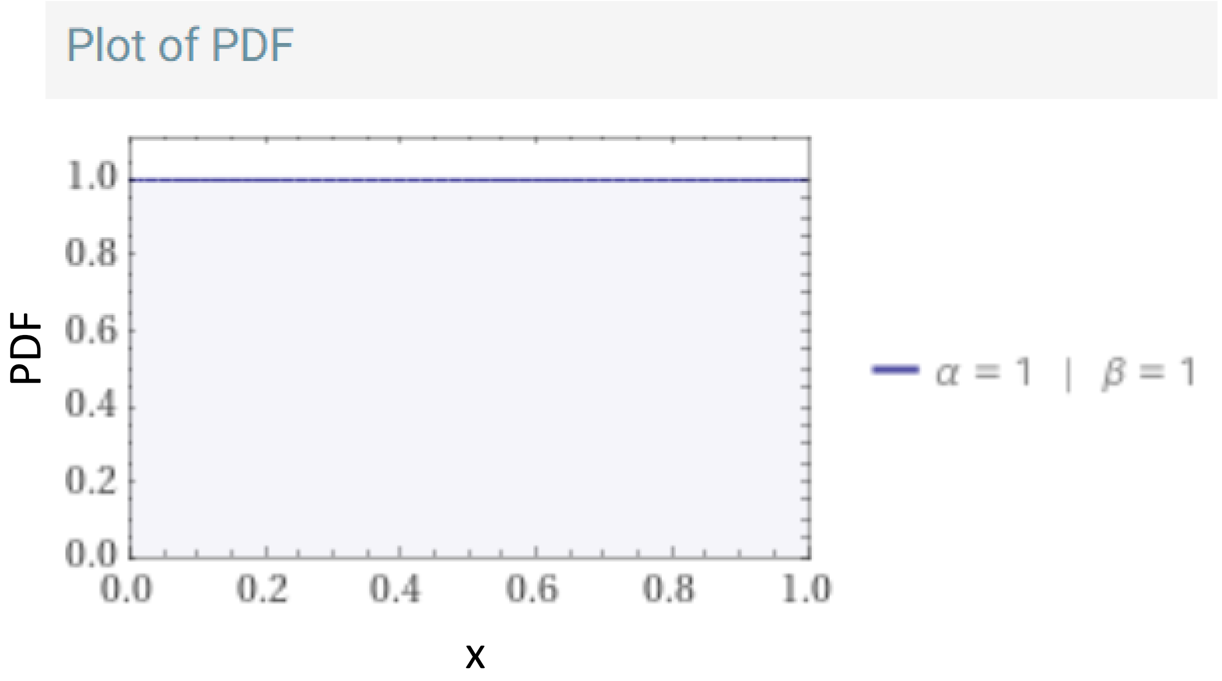
```
if aug_type == 'ssl_with_mixup':
    lamb = np.random.beta(alpha, alpha)
    _img = lamb * img_s[[i_unlabel], ...] + (1 - lamb) * img_s[[j_label], ...]
    _gt_labels = torch.cat((gt_labels_s[i_unlabel] * lamb, gt_labels_s[j_label] * (1 - lamb)))
    _gt_bboxes = torch.cat((gt_bboxes_s[i_unlabel], gt_bboxes_s[j_label]))
elif aug_type == 'ssl_with_mosaic':
    _, _, _h, _w = img_s.shape
    _img = img_s[[i_unlabel], ...]
    if np.random.randint(0, 2) == 1:    ## split top-down
        cy = np.random.randint(_h // 4, _h // 4 * 3)
        _img[0, :, cy:, :] = img_s[[j_label], :, cy:, :]
        gt_bboxes_i, gt_labels_i = self._clip_gt(gt_bboxes_s[i_unlabel], gt_labels_s[i_unlabel], 0, 0, _w, cy)
        gt_bboxes_j, gt_labels_j = self._clip_gt(gt_bboxes_s[j_label], gt_labels_s[j_label], 0, cy, _w, _h)
        _gt_bboxes = torch.cat((gt_bboxes_i, gt_bboxes_j))
        _gt_labels = torch.cat((gt_labels_i, gt_labels_j))
    else:                               ## split left-right
        #cx = int(gt_bboxes[i][np.random.choice(list(range(len(gt_labels[i]))))][2])
        cx = np.random.randint(_w // 4, _w // 4 * 3)
        _img[0, :, :, cx:] = img_s[[j_label], :, :, cx:]
        gt_bboxes_i, gt_labels_i = self._clip_gt(gt_bboxes_s[i_unlabel], gt_labels_s[i_unlabel], 0, 0, cx, _h)
        gt_bboxes_j, gt_labels_j = self._clip_gt(gt_bboxes_s[j_label], gt_labels_s[j_label], cx, 0, _w, _h)
        _gt_bboxes = torch.cat((gt_bboxes_i, gt_bboxes_j))
        _gt_labels = torch.cat((gt_labels_i, gt_labels_j))
return _img, _gt_bboxes, _gt_labels
```

https://github.com/txdet/Instant-Teaching/blob/d07910c4c811d875b03200ffb1822c32556ccf9a/projects/InstantTeaching/models/detectors/instant_teaching.py#L36

- PDF : $f(x;\alpha,\beta) = constant \cdot x^{\alpha-1}(1-x)^{\beta-1}$

$$1 = \int_0^1 \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}dx \qquad B(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx$$





- In semi-supervised learning, common problem → Confirmation bias
- Affecting the performance of model how to choose the unlabeled data

1. Same structure of models but different initialization(Model-a: $f_a(\cdot)$, Model-b: $f_b(\cdot)$)
2. Sharing same data in each batch but different data aug and pseudo annotations
3. Follows below:

$$\begin{cases} (c_i, \mathbf{t}_i) & = & f_a(\mathbf{x}_u), \\ (c_i^r, \mathbf{t}_i^r) & = & f_b(\mathbf{x}_u; \mathbf{t}_i), \\ c_i^* & = & \frac{1}{2}(c_i + c_i^r), \\ \mathbf{t}_i^* & = & \frac{1}{c_i + c_i^r}(\mathbf{t}_i c_i + \mathbf{t}_i^r c_i^r). \end{cases}$$

unlabeled image: $\mathbf{x}_u$
bounding box coordinates: $\mathbf{t}_i$
refined bbox coordinates: $\mathbf{t}_i^r$
rectify bbox coordinates: $\mathbf{t}_i^*$
class probabilities: $\mathbf{c}_i$
refine class probabilities: $\mathbf{c}_i^r$
rectify class probabilities: $\mathbf{c}_i^*$

Confirmation Bias

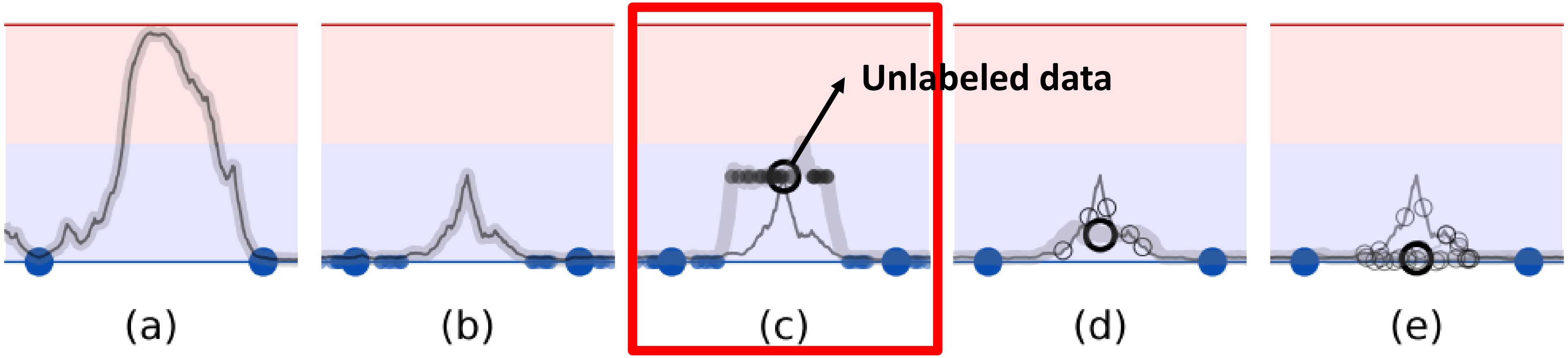


Figure 1: A sketch of a binary classification task with two labeled examples (large blue dots) and one unlabeled example, demonstrating how the choice of the unlabeled target (black circle) affects the fitted function (gray curve). **(a)** A model with no regularization is free to fit any function that predicts the labeled training examples well. **(b)** A model trained with noisy labeled data (small dots) learns to give consistent predictions around labeled data points. **(c)** Consistency to noise around unlabeled examples provides additional smoothing. For the clarity of illustration, the teacher model (gray curve) is first fitted to the labeled examples, and then left unchanged during the training of the student model. Also for clarity, we will omit the small dots in figures d and e. **(d)** Noise on the teacher model reduces the bias of the targets without additional training. The expected direction of stochastic gradient descent is towards the mean (large blue circle) of individual noisy targets (small blue circles). **(e)** An ensemble of models gives an even better expected target. Both Temporal Ensembling and the Mean Teacher method use this approach.

Tarvainen, Antti and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results." *NIPS* (2017).

Experiments

- Datasets
  - MS-COCO and PASCAL-VOC
- COCO-standard(train2017)
  - 118k labeled images
  - 850k instances from 80 classes
  - 123k unlabeled images
  - Randomly sample 1, 5, and 10% of labeled training data as **a labeled set**
  - Rest of labeled data as an unlabeled set
  - 1%: 1.2k images
  - Dataset split strategy for semi-supervised learning from STAC(Self-Training and the Augmentation driven Consistency regularization)

Sohn, Kihyuk, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee and Tomas Pfister. "**A Simple Semi-Supervised Learning Framework for Object Detection**." *ArXiv* abs/2005.04757 (2020): n. pag.

Experiments

- PASCAL-VOC
  - VOC07 trainval: 5,011 training images from 20 classes as **a labeled set**
  - VOC12 trainval: 11,540 training images as **an unlabeled set**
  - Validation sets: COCO val2017 and VOC07 test set, respectively
- Network
  - Faster-RCNN with FPN and ResNet-50 backbone
  - MMDetection: Open MMLab Detection Tool box and Benchmark
  - Initial weight: pre-trained on ImageNet

Experiments

- **STAC**: SSL for object detection(**Self-Training** and the **Augmentation** driven **Consistency** regularization)
- **CSD**: **C**onsistency-based **s**emi-supervised learning for object **d**etection
- **Instant-Teaching**: An end-to-end semi-supervised object detection framework

Methods	Backbone	1% COCO	2% COCO	5% COCO	10% COCO	100% COCO
Supervised	R50-FPN	9.05±0.16	12.70±0.15	18.47±0.22	23.86±0.81	37.63
CSD† [22]	R50-FPN	10.20±0.15 (+1.15)	13.60±0.10 (+0.90)	18.90±0.10 (+0.43)	24.50±0.15 (+0.64)	38.87 (+1.24)
STAC[45]	R50-FPN	13.97±0.35 (+4.92)	18.25±0.25 (+5.55)	24.38±0.12 (+5.91)	28.64±0.21 (+4.78)	39.21 (+1.58)
Instant-Teaching (ours)	R50-FPN	**16.00±0.20** (+6.95)	**20.70±0.30** (+8.00)	**25.50±0.05** (+7.03)	**29.45±0.15** (+5.59)	**39.60** (+1.97)
Instant-Teaching* (ours)	R50-FPN	**18.05±0.15** (+9.00)	**22.45±0.15** (+9.75)	**26.75±0.05** (+8.28)	**30.40±0.05** (+6.54)	**40.20** (+2.57)

Table 1. Comparison of mAP for different semi-supervised methods on MS-COCO. CSD† is our implementation of the CSD method based on the Faster-RCNN detector. Instant-Teaching* represents our Instant-Teaching framework with co-rectify scheme. The value in brackets represents the mAP improvement compared to the supervised model.

Experiments

- **STAC**: SSL for object detection(**Self-Training** and the **Augmentation** driven **Consistency** regularization)
- **CSD**: **C**onsistency-based **s**emi-supervised learning for object **d**etection
- **Instant-Teaching**: An end-to-end semi-supervised object detection framework

Methods	Backbone	Unlabeled	$AP^{0.5:0.95}$	$AP^{0.5}$	$AP^{0.75}$
Supervised (Ours)	R50-FPN		43.60	76.70	44.50
CSD [22]	R101-R-FCN	VOC12	-	74.70	-
STAC [45]	R50-FPN	VOC12	44.64 (+1.04)	77.45	-
Instant-Teaching	R50-FPN	VOC12	**48.70** (+5.10)	**78.30**	**52.00** (+7.50)
Instant-Teaching*	R50-FPN	VOC12	**50.00** (+6.40)	**79.20**	**54.00** (+9.50)
CSD [22]	R101-R-FCN	VOC12 & COCO	-	75.10	-
STAC [45]	R50-FPN	VOC12 & COCO	46.01 (+2.41)	79.08	-
Instant-Teaching	R50-FPN	VOC12 & COCO	**49.70** (+6.10)	79.00	**54.10** (+9.60)
Instant-Teaching*	R50-FPN	VOC12 & COCO	**50.80** (+7.20)	**79.90**	**55.70** (+11.20)

Table 2. Comparison of mAP for different semi-supervised methods on VOC07. We report the mAP at IoU=0.50:0.95 ($AP^{0.5:0.95}$), IoU=0.5 ($AP^{0.5}$) and IoU=0.75 ($AP^{0.75}$), which are the standard metrics for object detection [31, 7].

- N1: human-annotated instances
- N2: human-annotated instances and model generated
- Getting increase the number of pseudo labels according to high quality pseudo labels

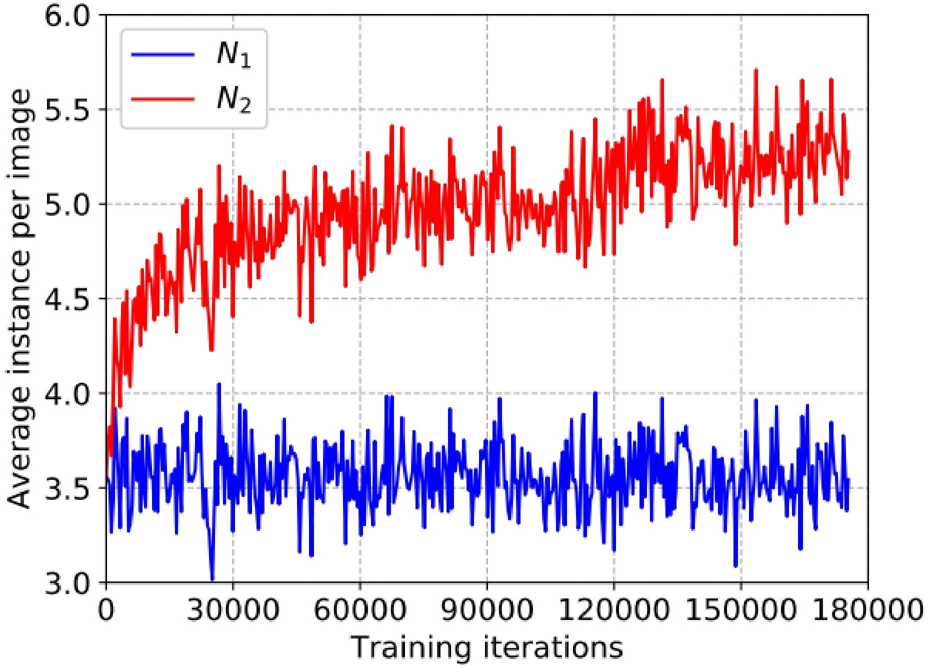


Figure 3. Changes in the number of annotations per image during training. N_1 refers human-annotated instances and N_2 refers total instances including human-annotated and model generated.

Experiments

Methods	Strong data augmentations				mAP
	Color+Cutout	Geometric	Mixup	Mosaic	
STAC[45]	√	√			23.14
Instant-Teaching	√*				21.60 (-1.54)
	√				24.70 (+1.56)
	√		√		25.40 (+2.26)
	√			√	25.00 (+1.86)
	√		√	√	**25.60** (+2.46)

Table 3. Comparison of mAP of Instant-Teaching trained with various data augmentation methods at the protocol of 5% MS-COCO and 8× unlabeled data. √* denotes that we also apply strong augmentations "Color+Cutout" to unlabeled data in the first step during instant pseudo labeling.

Methods	Labeled Size	Unlabeled Size				
		1×	2×	4×	8×	Full
STAC[45]	5% COCO	19.81	20.79	22.09	23.14	24.38±0.12
Instant-Teaching	5% COCO	**23.60**	**24.30**	**25.30**	**25.60**	**25.60±0.14**
STAC[45]	10% COCO	25.38	26.52	27.33	27.95	28.64±0.21
Instant-Teaching	10% COCO	**28.80**	**29.00**	**29.20**	**29.50**	**29.53±0.17**

Table 4. Comparison of mAP of Instant-Teaching trained with various scales of unlabeled data on MS-COCO. $[n]\times$ denotes the scale of unlabeled data is $[n]$ times larger than that of labeled data.

Experiments

τ	0.3	0.5	0.7	0.9
mAP (%)	26.30	27.70	28.70	29.80

Table 5. Comparison of mAP with various values of confidence threshold τ.

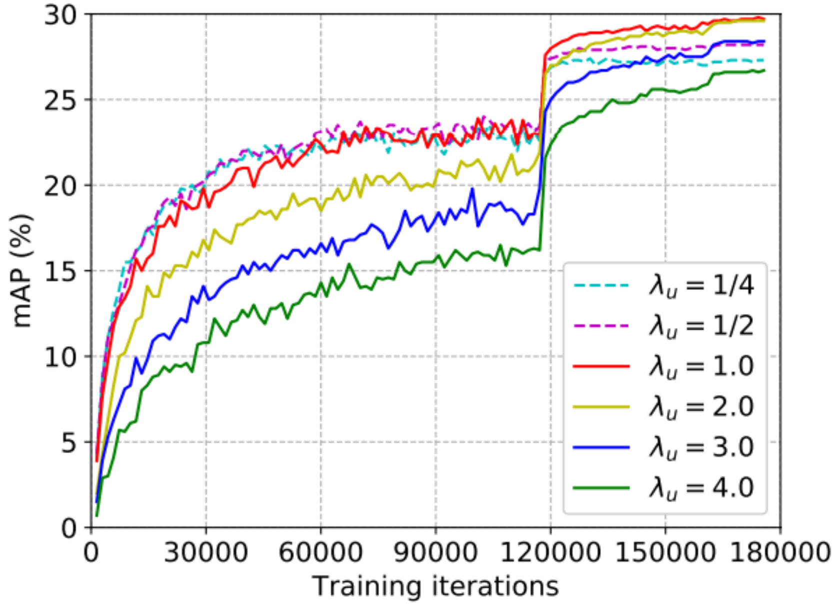


Figure 5. Comparison of mAP with various values of λ_u along training iterations.

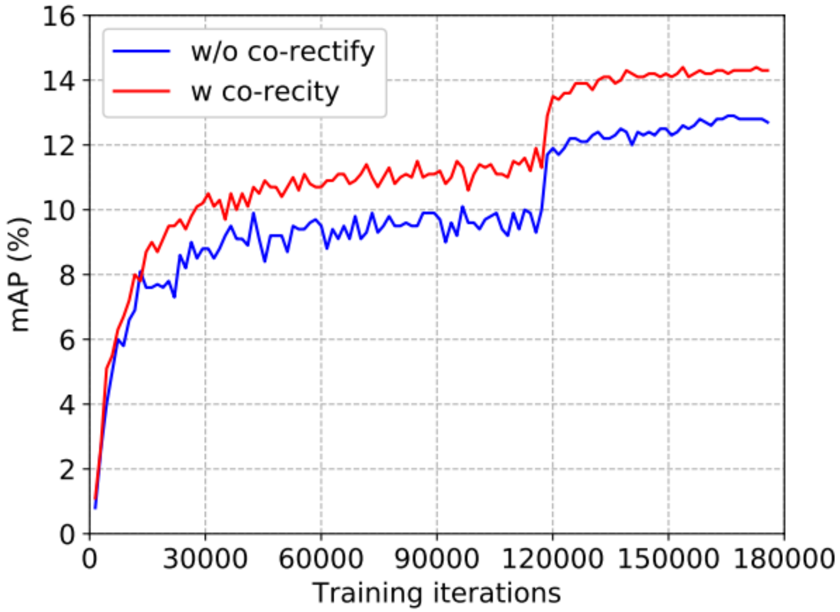


Figure 6. Comparison of mAP of generated pseudo annotations with different training iterations. The model is trained based on Instant-Teaching with and without co-rectify respectively.

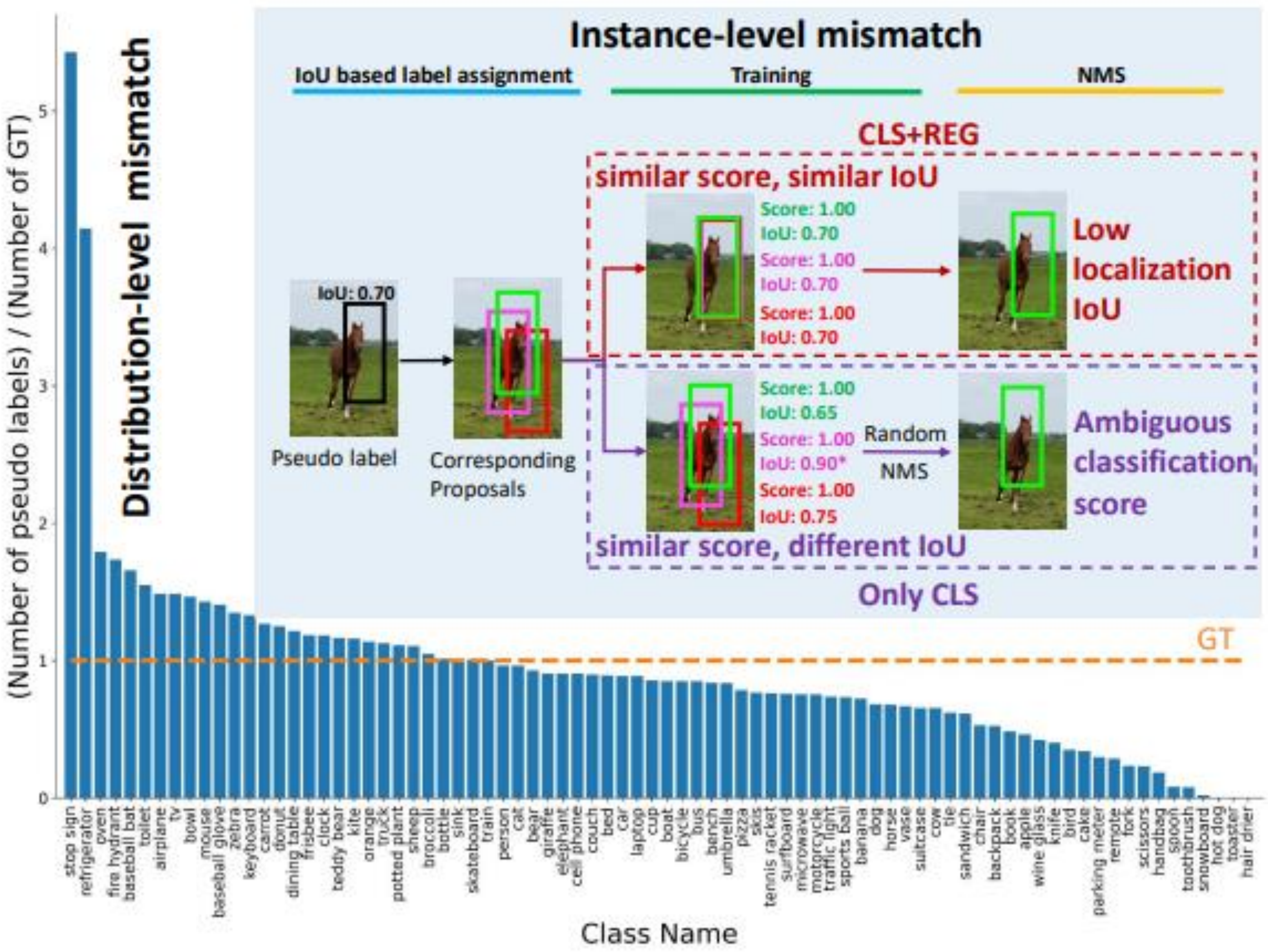
Experiments

Without co-rectify

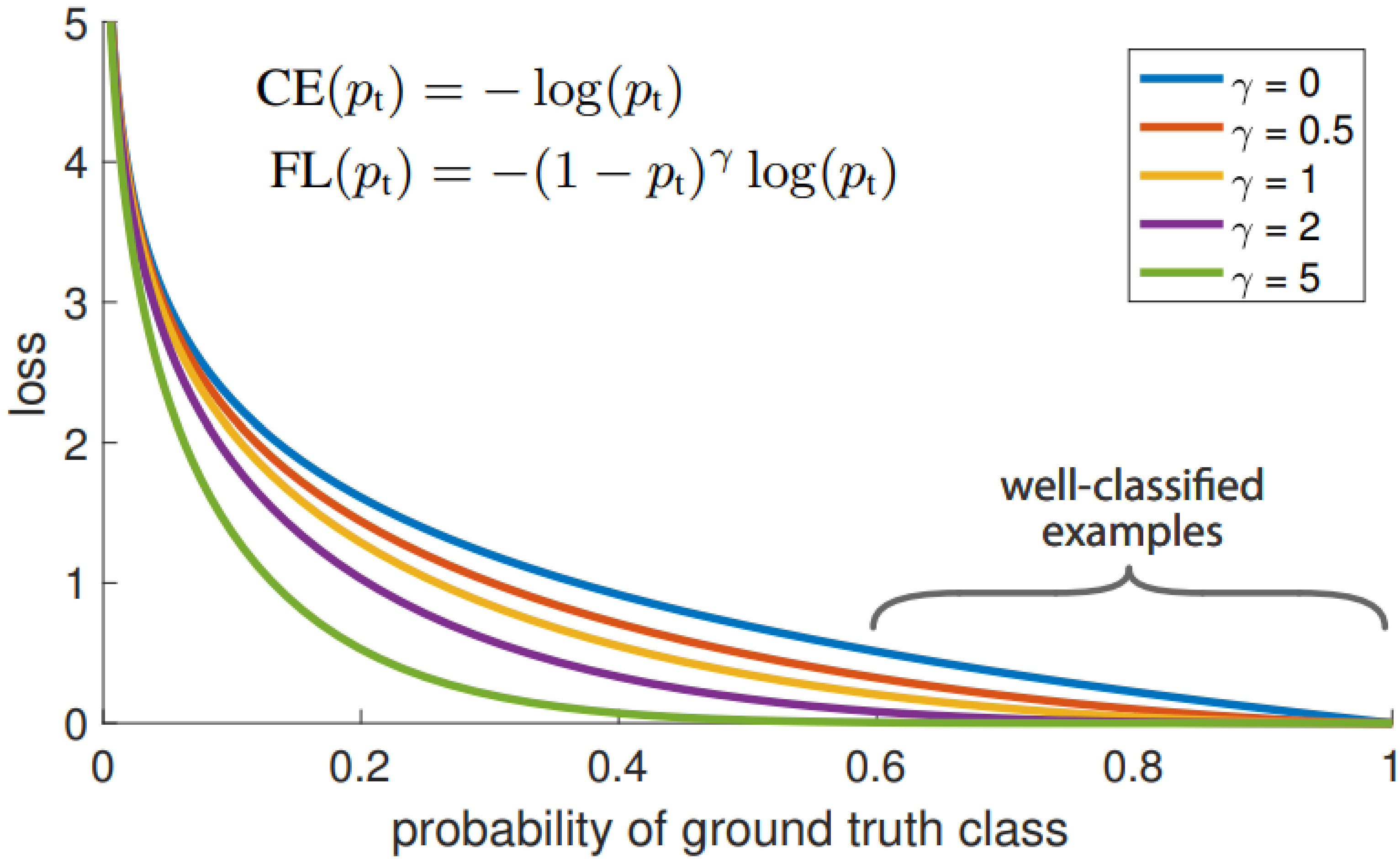
With co-rectify

Label Mismatch Problems on the MS-COCO dataset

- Distribution-level mismatch
  - Pseudo labels produced by the single confidence threshold and ground truth labels



Cross Entropy Loss vs Focal Loss



$$\text{Focal Loss} = -(1 - p_t)^\gamma \log(p_t)$$

Lin, Tsung-Yi et al. "Focal Loss for Dense Object Detection." ***IEEE Transactions on Pattern Analysis and Machine Intelligence*** **42 (2020): 318-327.**

- Cross Entropy Loss

$$\text{Cross Entropy}(p_t) = -\log(p_t)$$

$$CE(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise} \end{cases} \qquad p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise} \end{cases}$$

- Foreground, y=1, p=0.95

$$\text{CE}(Foreground) = -\log(0.95) = 0.05$$

- Background, y=0, p=0.05

$$\text{CE}(Background) = -\log(1-0.05) = 0.05$$

- FG Loss = BG Loss
  - Number of cases of Background >> Number of cases of Foreground
  - Updating the loss based on Background data & less training for the Foreground

- Balanced Cross Entropy Loss

$$\text{Balanced Cross Entropy}(p_t) = -\alpha_t \log(p_t) \quad 0 \leq \alpha_t \leq 1$$

- Foreground, y=1, p=0.95

$$\text{CE}(FG) = -0.75\log(0.95) = 0.038$$

- Background, y=0, p=0.05

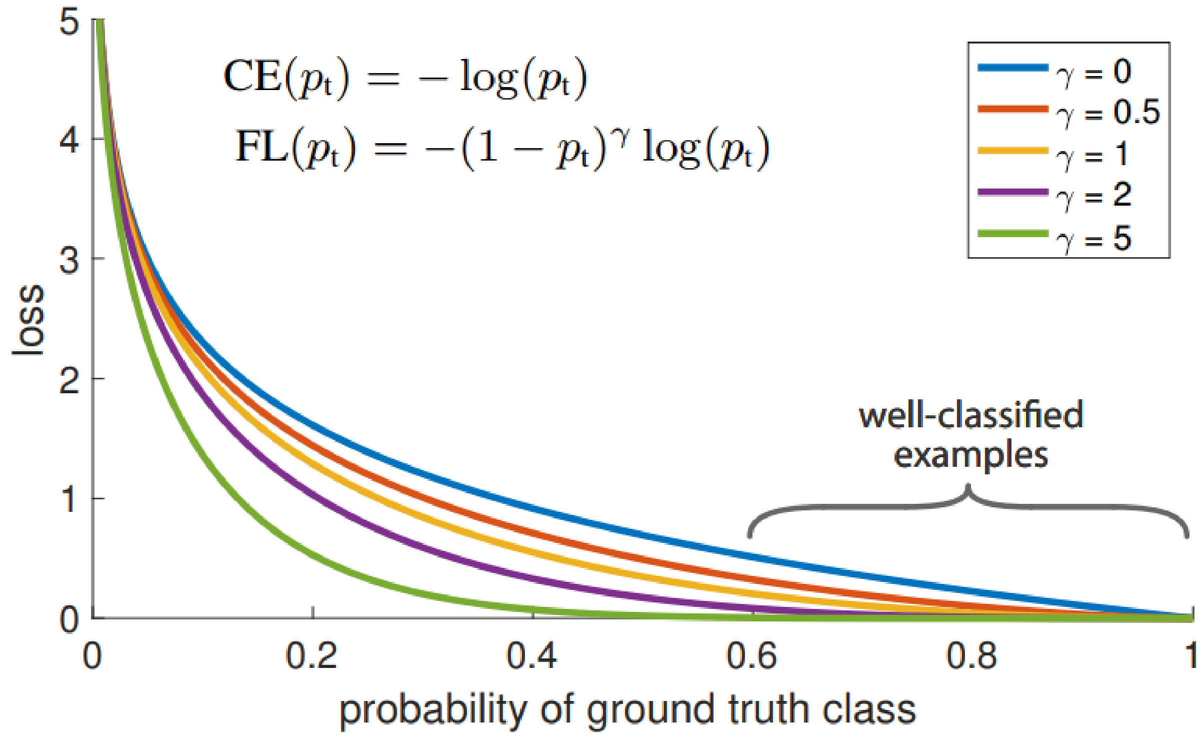
$$\text{CE}(BG) = -(1 - 0.75)\log(1 - 0.05) = 0.0128$$

- BCE for class imbalance(BG >> FG)
- Problem
  - Not distinguished Easy/Hard Example

- Focal Loss

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \qquad 0 \le \alpha_t \le 1 \qquad \gamma \ge 0$$

- In the paper, $\alpha = 0.25(\mathrm{FG}), \gamma = 2$



Lin, Tsung-Yi et al. "Focal Loss for Dense Object Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2020): 318-327.

- Focal Loss for hard/easy examples

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \qquad 0 \le \alpha_t \le 1 \qquad \gamma \ge 0$$

Hard

$$\mathrm{CE}(0.1) = -\log(0.1) = 2.30259\ldots \approx 2.3$$

$$\mathrm{FL}(0.1) = -(1-0.1)\log(0.1) = 2.07233\ldots \approx 2.1$$

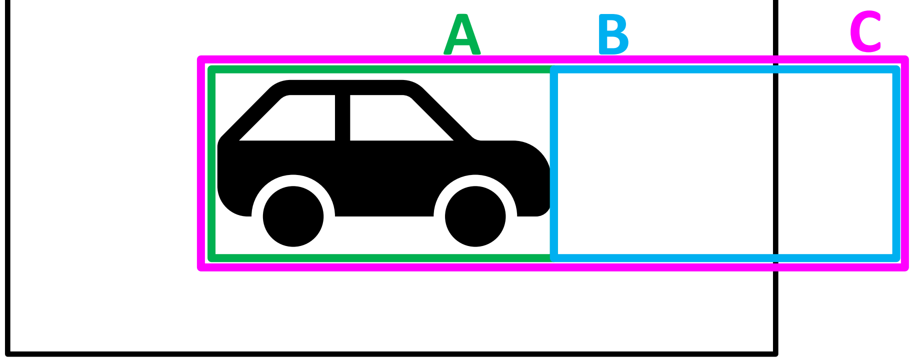
Easy

$$\mathrm{CE}(0.9) = -\log(0.9) = 0.105361\ldots \approx 0.1$$

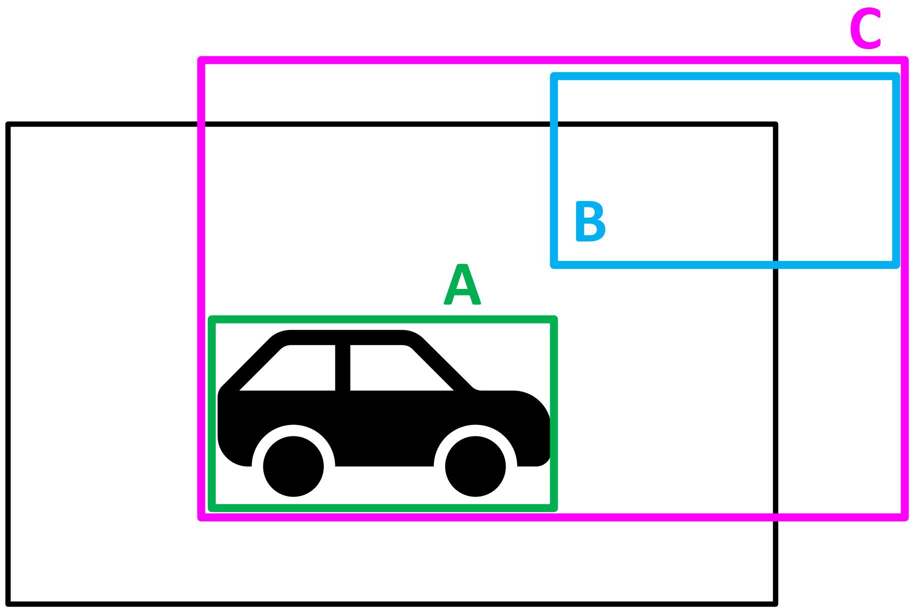
$$\mathrm{FL}(0.9) = -(1-0.9)\log(0.8) = 0.0105361\ldots \approx 0.01$$

Lin, Tsung-Yi et al. "Focal Loss for Dense Object Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2020): 318-327.

GIOU(Generalized Intersection over Union)



IoU=0
GIoU=0



IoU=0
GIoU=-0.7

$$GIoU = IoU - \frac{|C-(A \cup B)|}{|C|}$$

$$Loss = 1 - GIoU, 0 \leq Loss \leq 2$$

- Anchor-based Object Detection
  - Various hyperparameters: number of anchors, size, aspect ratio, etc.
    - Bad for small object and various object type
    - Fixed box scale and aspect ratio for anchor
  - Sample imbalance
    - Density of anchor box for high recall rate, imbalanced negative-positive sample
    - Appear a lot of negative anchor box
  - Expensive computation
    - Compute for IOU with predicted box and GT box

- Anchor-free Object Detection
- Without anchor, detecting the object
    1. Keypoint-based method for detecting the object position
    2. Center-based method for predicting the object boundary

Adaptive Filtering Strategy

- Ignoring the gradients computation and propagation for Ignorable regions <- Different point

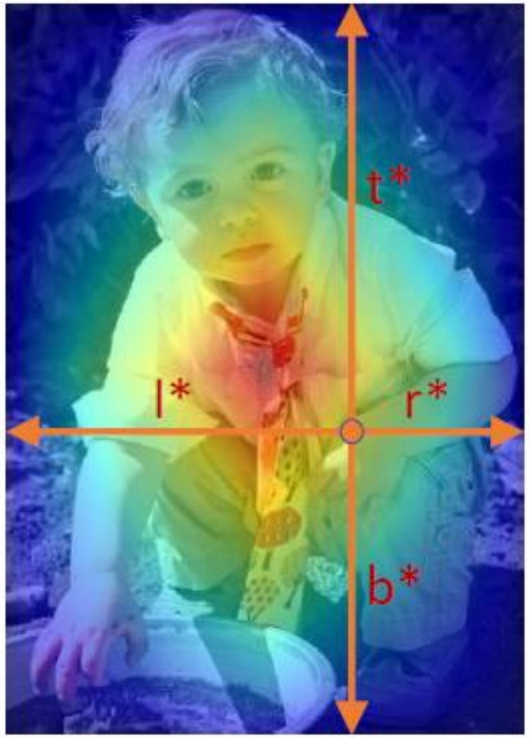
$$L_u = \frac{1}{N_{pos}} \sum_i \sum_{h,w} (\mathbf{1}_{\{\bar{p}^*_{h,w} \geq 0\}} L_{cls}(U_{i,h,w}) + \mathbf{1}_{\{\bar{p}^*_{h,w} \in [0, C-1]\}} L_{reg}(U_{i,h,w}) + \mathbf{1}_{\{\bar{p}^*_{h,w} \in [0, C-1]\}} L_{center}(U_{i,h,w})$$

Baseline

- Anchor-free detector from FCOS
  - ResNet50 backbone
  - FPN neck and a dense head

Tian, Z., Shen, C., Chen, H., & He, T. (2019). ***FCOS: Fully Convolutional One-Stage Object Detection****. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 9626-9635.*

- Prediction for the distances from the location to the four sides of bounding-box



$$\text{real vector, } \mathbf{t}^* = (l^*, t^*, r^*, b^*)$$

$$l^* = x - x_0^{(i)} \qquad t^* = y - y_0^{(i)}$$

$$r^* = x_1^{(i)} - x \qquad b^* = y_1^{(i)} - y$$

$$\text{center} = (x, y)$$

$$\text{left top} = (x_0^{(i)}, y_0^{(i)})$$

$$\text{right bottom} = (x_1^{(i)}, y_1^{(i)})$$

- Centerness
  - Close to center(x,y), centerness → 1
  - Far from center(x,y), centerness → 0

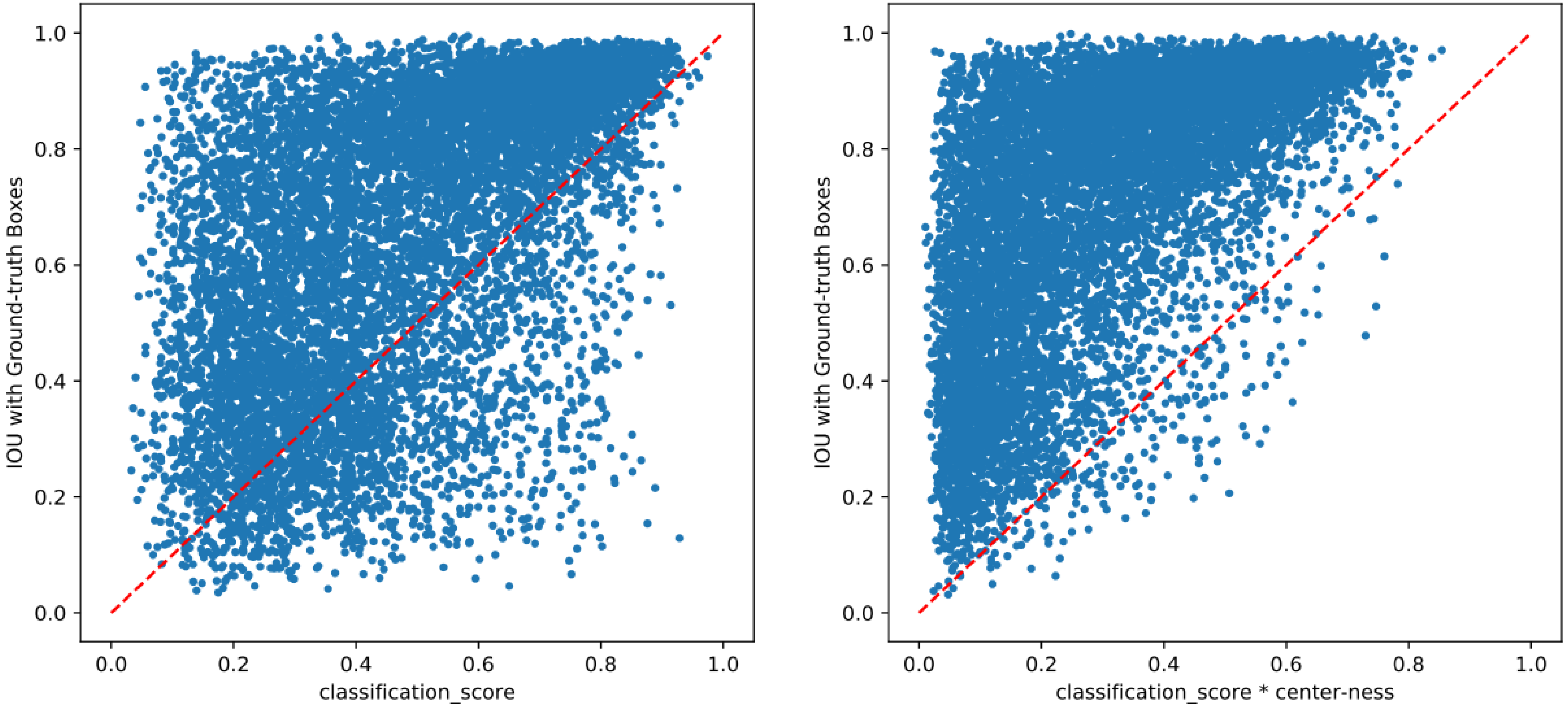
$$\text{centerness} = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

	l	r		l	r		l	r
	0.8	0.2		0.5	0.5		0.6	0.4
	t	b		t	b		t	b
	0.6	0.4		0.5	0.5		0.5	0.6
min	0.2	0.4	min	0.5	0.5	min	0.4	0.5
max	0.8	0.6	max	0.5	0.5	max	0.6	0.6
centerness	0.40824829		centerness	1		centerness	0.745355992	

- IOU with Ground-truth Bboxes

$$\text{classification score} * \text{center-ness}$$

- Easy to split predicted result from nms



FCOS Loss Function

- L_{cls} :,focal loss L_{reg} : IOU Loss

$$L(\{\mathbf{p}_{x,y}\}, \{\mathbf{t}_{x,y}\}) = \frac{1}{N_{pos}} \sum_{x,y} L_{cls}(\mathbf{p}_{x,y}, c^*_{x,y}) + \frac{\lambda}{N_{pos}} \sum_{x,y} \mathbf{1}_{\{c^*_{x,y}>0\}} L_{reg}(\mathbf{t}_{x,y}, \mathbf{t}^*_{x,y})$$

Focal Loss: Lin, T., Goyal, P., Girshick, R.B., He, K., & Dollár, P. (2020). Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*, 318-327.

IOU Loss: Yu, J., Jiang, Y., Wang, Z., Cao, Z., & Huang, T.S. (2016). UnitBox: An Advanced Object Detection Network. *Proceedings of the 24th ACM international conference on Multimedia*.

Recurrence Layer Aggregation

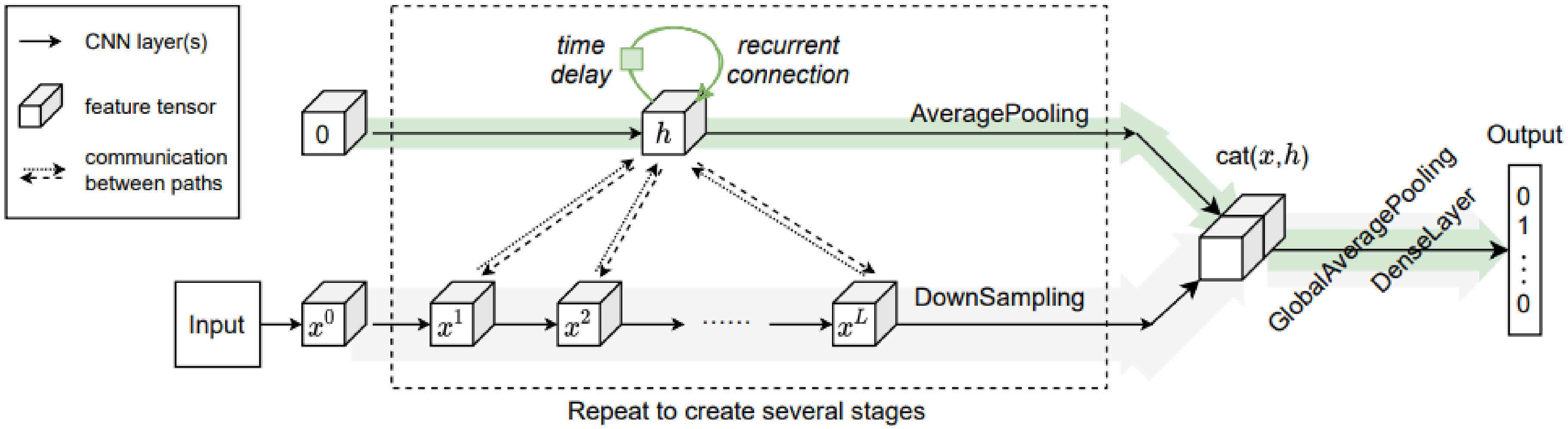
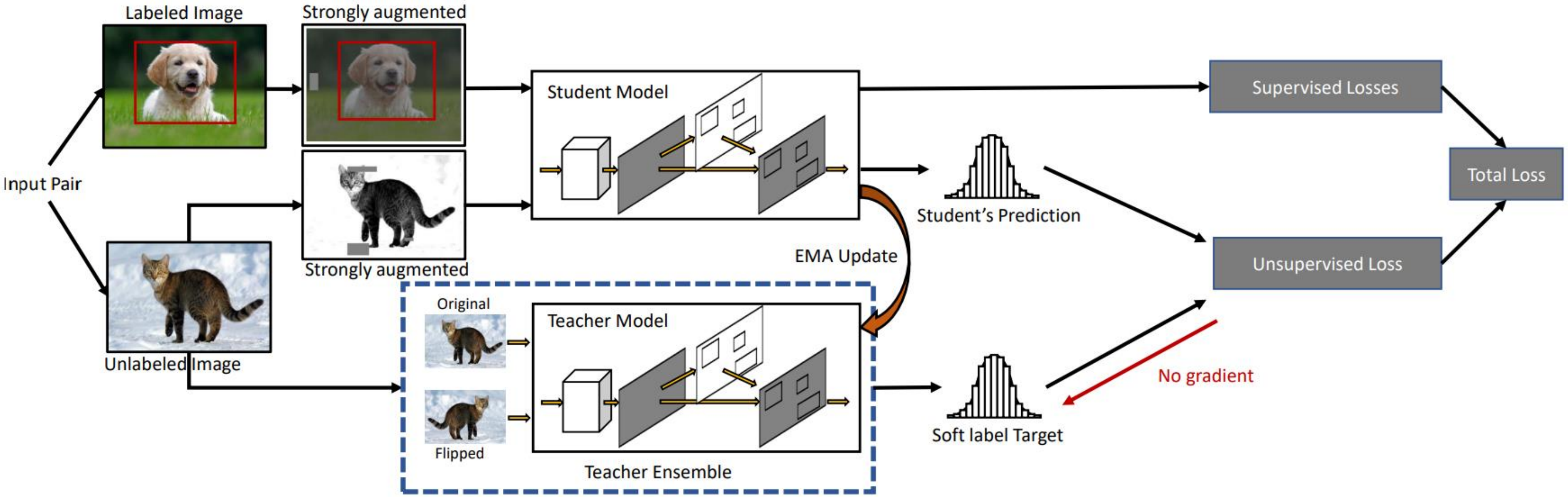


Figure 1: Schematic diagram of a CNN with recurrent layer aggregation for image classification.

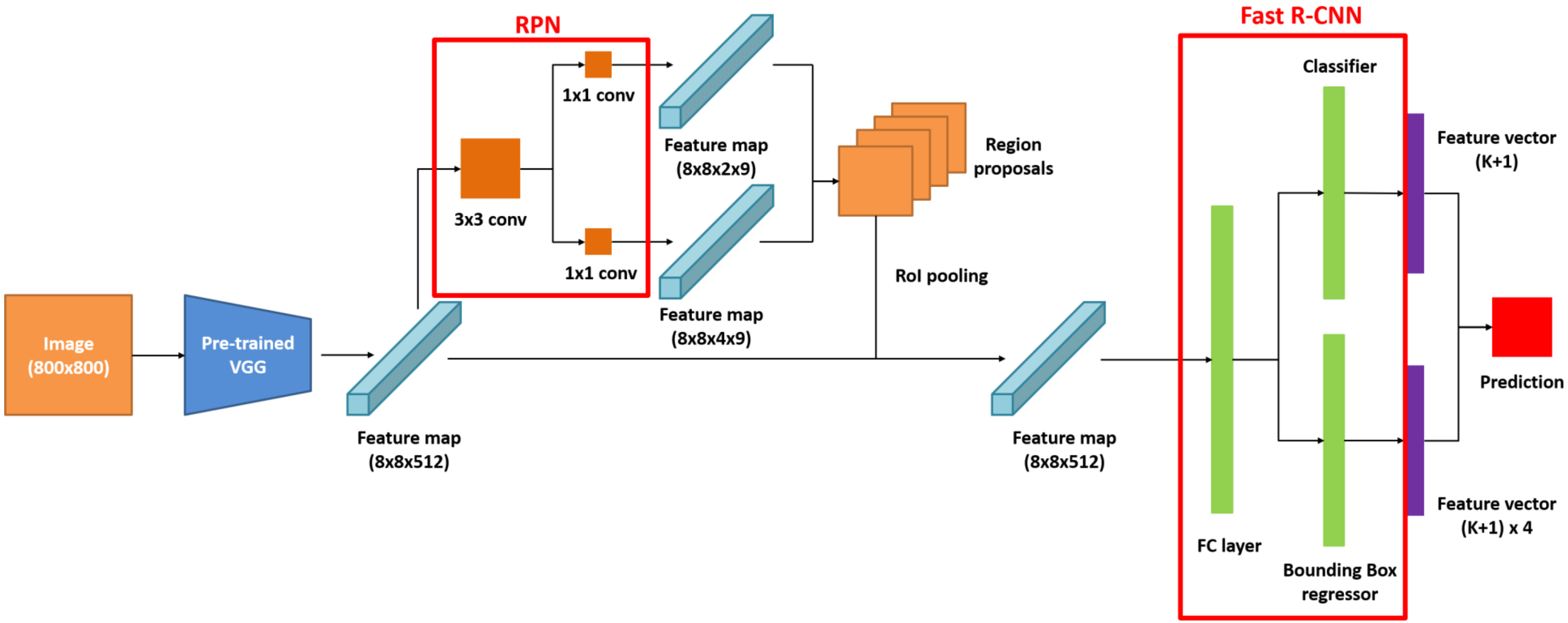
Zhao, J., Fang, Y., & Li, G. (2021). Recurrence along Depth: Deep Convolutional Neural Networks with Recurrent Layer Aggregation. *ArXiv, abs/2110.11852*.

Overview of Humble Teacher Approach

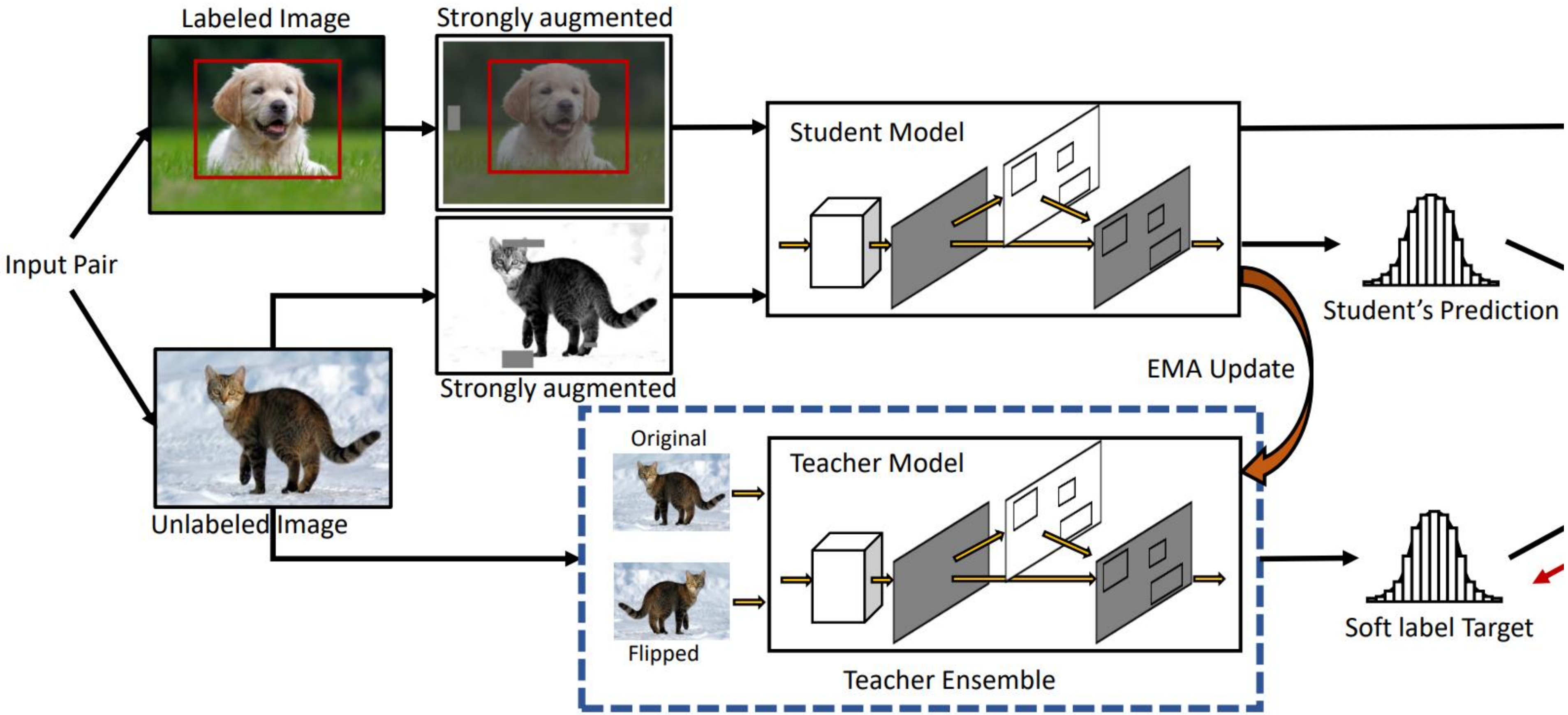


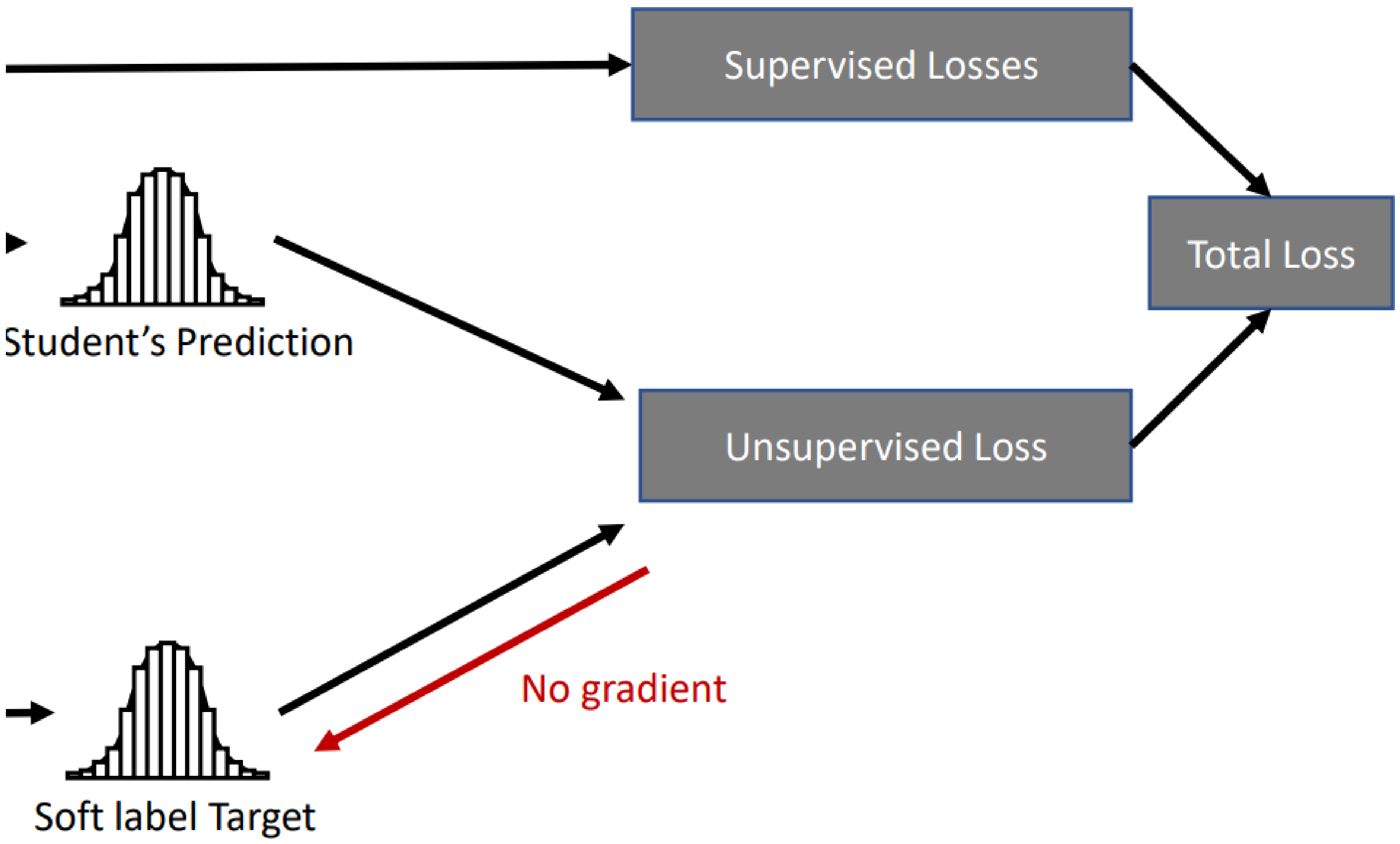
List of augmentation

Architecture of Faster R-CNN



Overview of Humble Teacher Approach




Supervised Losses
Total Loss
Student's Prediction
Unsupervised Loss
No gradient
Soft label Target

- By updating loss function, to improve the performance
- Total loss
  - Number of unlabeled images: n_U
  - Number of unlabeled images: n_S

$$\beta = 0.5 \qquad L = L_S + \frac{n_U}{n_S}\beta L_U$$

- Regular detection loss $\rightarrow$ Faster R-CNN
  - RPN classification loss: L_{cls}^{rpn}
  - RPN localization loss: L_{loc}^{rpn}
  - ROI head's classification loss: L_{cls}^{roi}
  - ROI head's localization loss: L_{loc}^{roi}

$$L_S = L_{cls}^{rpn} + L_{loc}^{rpn} + L_{cls}^{roi} + L_{loc}^{roi}$$

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, pages 91–99, 2015

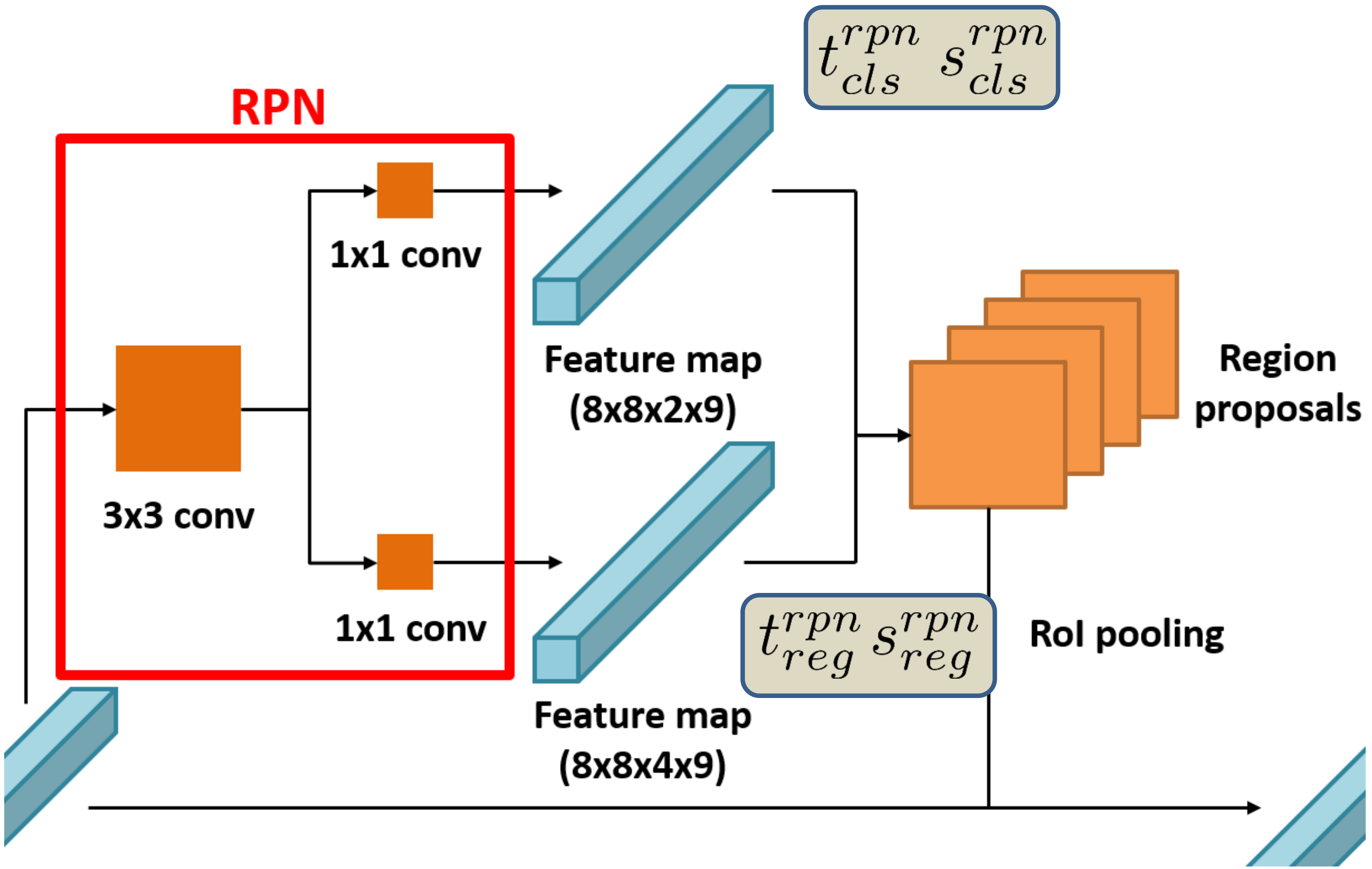
- Unsupervised loss for RPN

$$L_U^{rpn} = \sum_{i \in S_A} D_{KL}(\mathbf{t}_{cls}^{rpn,i} \parallel \mathbf{s}_{cls}^{rpn,i}) + \|\mathbf{t}_{reg}^{rpn,i} - \mathbf{s}_{reg}^{rpn,i}\|_2$$

- All anchors: S_A , KL divergence: D_{KL}
- Teacher and Student RPN for the i-th proposal:

	Classification Probability	**Bounding box regression output**
Teacher	t_{cls}^{rpn}	t_{reg}^{rpn}
Student	s_{cls}^{rpn}	s_{reg}^{rpn}

Part of Region Proposal Network(RPN)



- Unsupervised loss: $L_U = L_U^{rpn} + L_U^{roi}$

$$L_U^{rpn} = \sum_{i \in S_A} D_{KL}(\mathbf{t}_{cls}^{rpn,i} \parallel \mathbf{s}_{cls}^{rpn,i}) + \|\mathbf{t}_{reg}^{rpn,i} - \mathbf{s}_{reg}^{rpn,i}\|_2$$

$\text{A. } -\sum_x p(x)\text{log}q(x) \Rightarrow \text{CE}(p,q)$

$\mathbf{t}_{cls}^{rpn,i}$: probability of classification for teacher RPN

$\mathbf{s}_{cls}^{rpn,i}$: probability of classification for student RPN

$$-\sum \mathbf{t}_{cls}^{rpn,i}\text{log}\mathbf{s}_{cls}^{rpn,i}$$

$\text{B. } -\sum_x p(x)\text{log}p(x) \Rightarrow \text{E}(p)$

$\mathbf{t}_{cls}^{rpn,i}$: probability of classification for teacher RPN

$$-\sum \mathbf{t}_{cls}^{rpn,i}\text{log}\mathbf{t}_{cls}^{rpn,i}$$

$$\sum_{i \in S_A} D_{KL}(\mathbf{t}_{cls}^{rpn,i} \parallel \mathbf{s}_{cls}^{rpn,i}) = -\sum_{i \in S_A} \mathbf{t}_{cls}^{rpn,i}\text{log}\mathbf{s}_{cls}^{rpn,i} - \mathbf{t}_{cls}^{rpn,i}\text{log}\mathbf{t}_{cls}^{rpn,i} = \sum_{i \in S_A} \mathbf{t}_{cls}^{rpn,i}\text{log}\frac{\mathbf{t}_{cls}^{rpn,i}}{\mathbf{s}_{cls}^{rpn,i}}$$

- RPN NMS: 300 proposal regions

- RoI sampling: N=640(according to class score)
- Positive:Negative=1:1
- Positive: IoU >= 0.7
- Negative: IoU <= 0.3
- Between: ignore

- Unsupervised loss: $$L_U = L_U^{rpn} + L_U^{roi}$$

$$L_U^{roi} = \sum_{i \in S_A} D_{KL}(\mathbf{t}_{cls}^{roi,i} \parallel \mathbf{s}_{cls}^{roi,i}) + \|\mathbf{t}_{reg}^{roi,i} - \mathbf{s}_{reg}^{roi,i}\|_2$$

$\mathrm{A.} - \sum_x p(x)\mathrm{log}q(x) \Rightarrow \mathrm{CE}(p,q)$

$\mathbf{t}_{cls}^{roi,i}$: probability of classification for teacher RoI

$\mathbf{s}_{cls}^{roi,i}$: probability of classification for student RoI

$$\mathrm{CE}(p,q) = -\sum \mathbf{t}_{cls}^{roi,i} \mathrm{log} \mathbf{s}_{cls}^{roi,i}$$

$\mathrm{B.} - \sum_x p(x)\mathrm{log}p(x) \Rightarrow \mathrm{E}(p)$

$\mathbf{t}_{cls}^{roi,i}$: probability of classification for teacher RoI

$$\mathrm{E}(p) = -\sum \mathbf{t}_{cls}^{roi,i} \mathrm{log} \mathbf{t}_{cls}^{roi,i}$$

$$\sum_{i \in S_A} D_{KL}(\mathbf{t}_{cls}^{roi,i} \parallel \mathbf{s}_{cls}^{roi,i}) = -\sum_{i \in S_A} \mathbf{t}_{cls}^{roi,i} \mathrm{log} \mathbf{s}_{cls}^{roi,i} - \mathbf{t}_{cls}^{roi,i} \mathrm{log} \mathbf{t}_{cls}^{roi,i} = \sum_{i \in S_A} \mathbf{t}_{cls}^{roi,i} \mathrm{log} \frac{\mathbf{t}_{cls}^{roi,i}}{\mathbf{s}_{cls}^{roi,i}}$$

Exponential Moving Average for the Teacher Model Update

- Updating teacher weights from student weights

$$W_{teacher} = \alpha W_{teacher} + (1 - \alpha) W_{student}$$
$$\alpha = 0.999$$

- Slightly updated teacher
  - Even though training with a wrong label prediction, its influence on the teacher model

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: **Weight-averaged consistency targets improve semi-supervised deep learning results**. In Advances in Neural Information Processing Systems, pages 1195–1204, 2017

Teacher Ensemble with Horizontal Flipping(1/2)

- Teacher model by taking as input both the image and horizontally flipped image
- Average prediction: both > only original image
- Backbone feature with original image: f_B
- Backbone feature with flipped image: $\hat{f}_B$
- Proposals detected by RPN with original image: P
- Horizontally flipped proposal coordinates from P : $\hat{P}$

$$f = \mathrm{ROIAlign}(f_B, P)$$

$$\hat{f} = \mathrm{ROIAlign}(\hat{f}_B, \hat{P})$$

roialign

examples

Teacher Ensemble with Horizontal Flipping(2/2)

- Classification head including softmax: C
- Regression head: R
- Transformation-flip by x axis of all bounding boxes: T

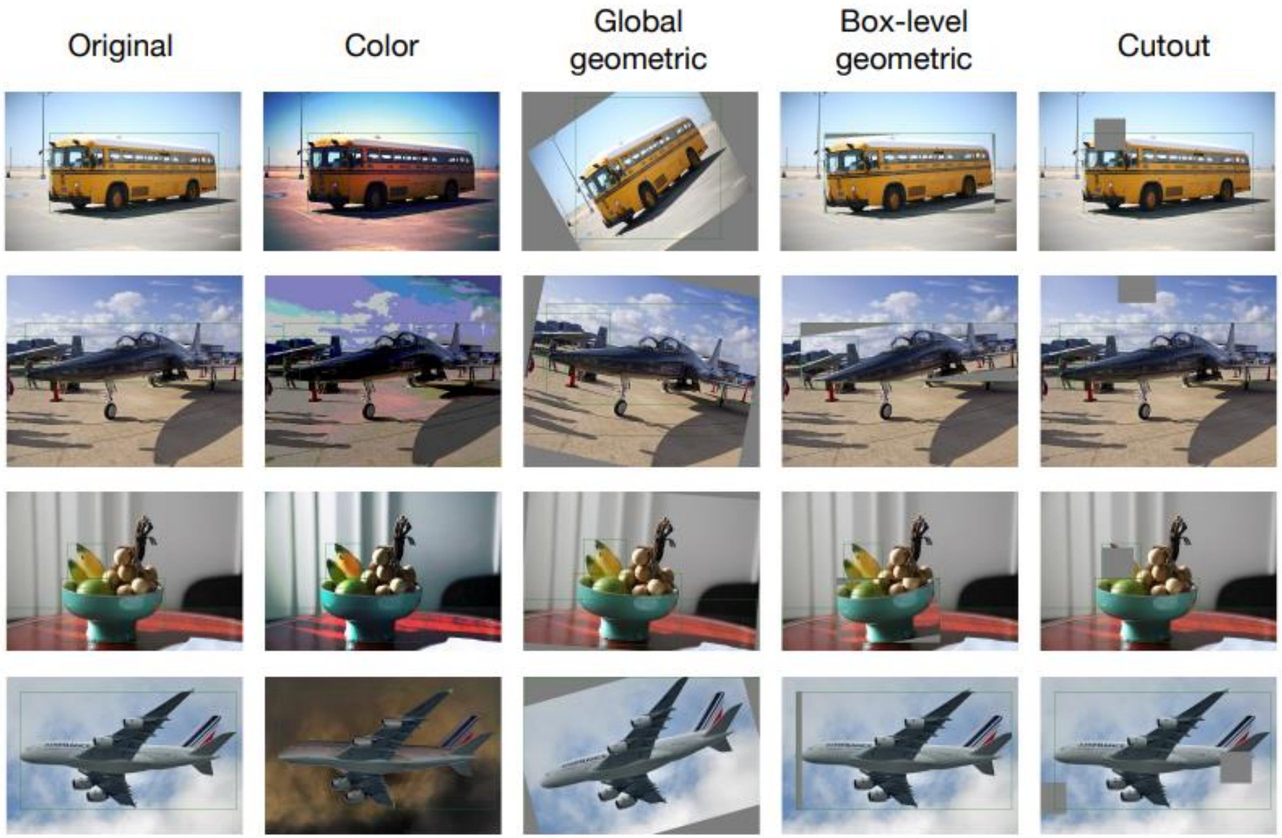
$$P_{cls} = 0.5(C(f) + C(\hat{f}))$$

$$\sigma_{reg} = 0.5(R(f) + T(R(\hat{f})))$$

- Inference stage
  - In teacher model, to produce object detection result
  - No data augmentation to the input image
- Augmentation
  - Weak augmentation
    - Random flipping and the image for teacher model
  - Strong augmentation
    - Randomly change the color, sharpness, contrast
    - Gaussian noise
    - Cutouts
- Model
  - Base model: Faster R-CNN with ResNet-50

Example of Strong Augmentation from STAC

- Strong augmentation
  - Color transformation / Cutout
  - Global geometric / Box-level geometric transformation



Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. A simple semi-supervised learning framework for object detection. arXiv preprint arXiv:2005.04757, 2020

Datasets

- MS COCO: Microsoft COCO: Common Objects in Context
  - Object segmentation & detection
  - 330K images & 1.5 million object instances
  - 80 object categories & 91 stuff categories
  - 250,000 people with keypoints

Example of Object Detection

- 200,000 images and 250,000 person instances labeled with keypoints

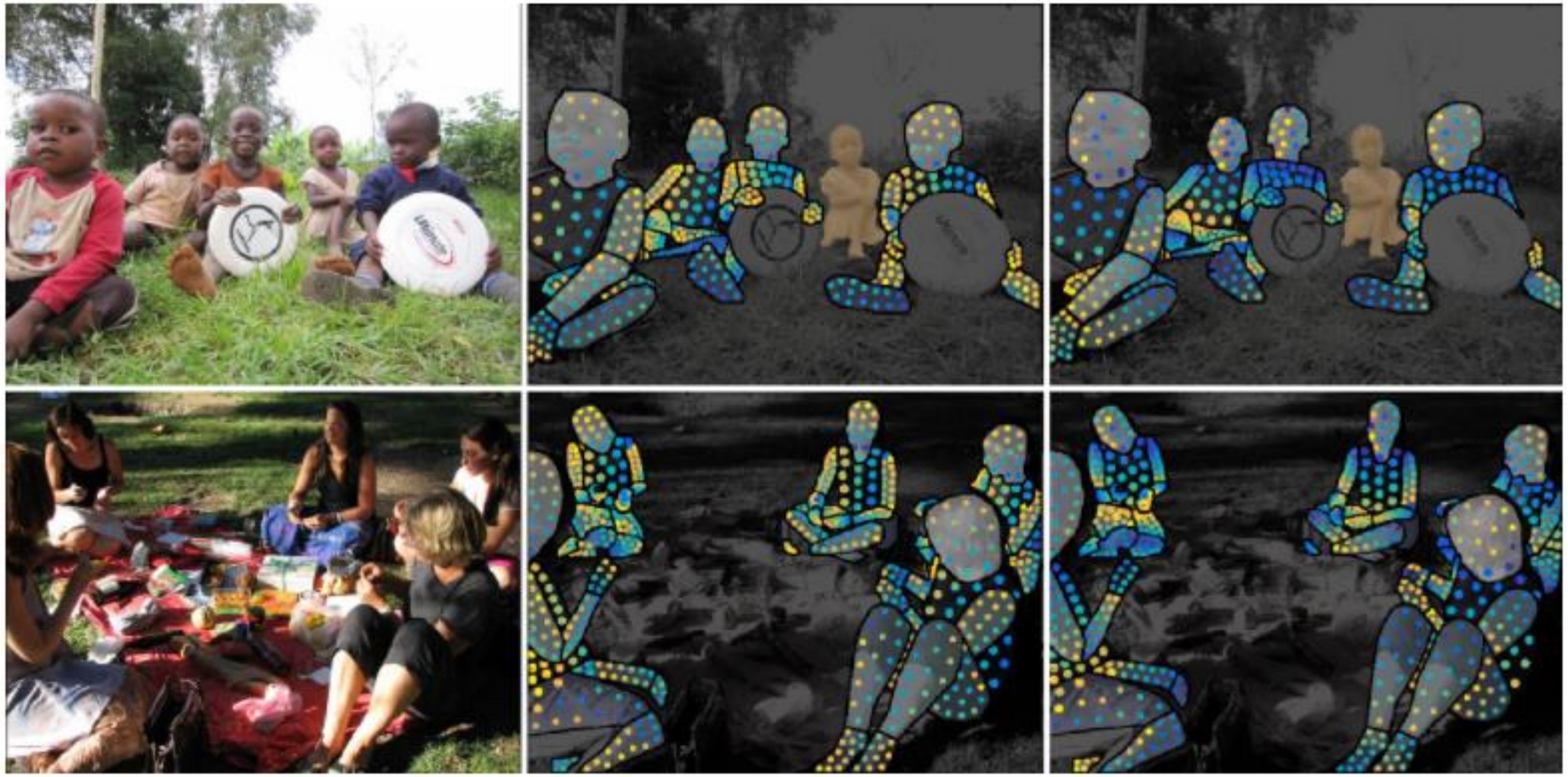
COCO 2020 Panoptic Segmentation Task

- 164K images from COCO 2017
- train 118K, val 5K, test-dev 20K, test-challenge 20K
- 172 classes: 80 thing classes, 91 stuff classes and 1 class 'unlabeled'

https://github.com/nightrome/cocostuff#label-hierarchy

COCO 2020 DensePose Task

- 39,000 images and 56,000 person instances labeled with DensePose annotations



PASCAL VOC

- PASCAL VOC project
  - Pattern Analysis, Statistical Modelling and Computational Learning(PASCAL),
  - VOC(Visual Object Classes)
  - Provides standardized image data sets for object class recognition
  - Provides a common set of tools for accessing the data sets and annotations
  - Enables evaluation and comparison of different methods
  - Ran challenges evaluating performance on object class recognition (from 2005-2012, now finished)
- Organizers
  - Mark Everingham (University of Leeds)
  - Luc van Gool (ETHZ, Zurich)
  - Chris Williams (University of Edinburgh)
  - John Winn (Microsoft Research Cambridge)
  - Andrew Zisserman (University of Oxford)

PASCAL VOC 2007/2012

- Classes: 20
  - Person(1): person
  - Animal(6): bird, cat, cow, dog, horse, sheep
  - Vehicle(7): aeroplane, bicycle, boat, bus, car, motorbike, train
  - Indoor(8): bottle, chair, dining, table, potted, plant, sofa, tv/monitor
- Train/validation/test
  - 9,963image containing 24,640 annotated objects

- Classes: 20
- Train/validation/test
  - 11,530 images containing 27,450 RoI annotated objects and 6,929 segmentations

PASCAL VOC 2007/2012

- PASCAL VOC 2007 labeled data
  - 5,011images
- PASCAL VOC 2012 unlabeled data
  - 11,530 images
- 1:2 = labeled : unlabeled
- PASCAL VOC 2012 and MS-COCO20(not included in VOC classes 20)
  - 4,993 : 124,834 = labeled : unlabeled = 1 : 26
  - In experiments, using MS-COCO2017 val dataset

Experiments

Model	Labeled Dataset	Unlabeled Dataset	AP50	AP
Supervised model	VOC07	N/A	76.3	42.60
Supervised model	VOC07 + VOC12	N/A	82.17	54.29
CSD‡	VOC07	VOC12	76.76	42.71
STAC [40]	VOC07	VOC12	77.45	44.64
Humble teacher (ours)	VOC07	VOC12	**80.94**	**53.04**
CSD‡	VOC07	VOC12 + MS-COCO20 (2017)	77.10	43.62
STAC [40]	VOC07	VOC12 + MS-COCO20 (2017)	79.08	46.01
Humble teacher (ours)	VOC07	VOC12 + MS-COCO20 (2017)	**81.29**	**54.41**

Table 1: Results on Pascal VOC, evaluated on the *VOC07 test* set. Our model consistently outperforms others in all experiment setups. CSD‡ is our ResNet-50-based re-implementation, which achieves better performance than the original CSD [19].

Percentage labeled	1%	2%	5%	10%
Supervised model	9.05±0.16	12.70±0.15	18.47±0.22	23.86±0.81
CSD‡	11.12±0.15 (+2.07)	14.15±0.13 (+1.45)	18.79±0.13 (+0.32)	22.76±0.09 (−1.10)
STAC [40]	13.97±0.35 (+4.92)	18.25±0.25 (+5.55)	24.38±0.12 (+5.91)	28.64±0.21 (+4.78)
Humble teacher (ours)	**16.96±0.38** (+7.91)	**21.72±0.24** (+9.02)	**27.70±0.15** (+9.23)	**31.61±0.28** (+7.74)

Table 2: The mAP (50:95) results on *MS-COCO val 2017* by models trained on different percentage of labeled *MS-COCO train 2017*. All models are with the ResNet-50 backbone. CSD‡ is our re-implementation with better performance. Our method consistently outperforms others.

Experiments

Model (Faster R-CNN with Resnet-50)	AP
Base supervised model	37.63
MOCOv2 + MS-COCO Unlabeled [7]	35.29
MOCOv2 + ImageNet-1M [7]	40.80
MOCOv2 + Instagram-1B [7]	41.10
Proposal learning [42]	38.4
CSD‡	38.52(+0.89)
STAC [40]	39.21(+1.58)
Humble teacher (ours)	**42.37(+4.74)**
Model (Cascade R-CNN with ResNet-152)	AP
Base supervised model	50.23
Humble teacher (ours)	**53.38 (+3.15)**

Table 3: The mAP (50:95) results on ***MS-COCO val 2017*** by models trained on *MS-COCO train 2017* + *MS-COCO unlabeled*. CSD‡ is with a ResNet-50 backbone.

Model (Cascade R-CNN with ResNet-152)	AP
Base supervised model	50.7
Humble teacher (ours)	**53.8 (+3.1)**

Table 4: The mAP (50:95) results on ***MS-COCO test-dev 2017*** by models trained on *MS-COCO train 2017* + *MS-COCO unlabeled*.

Model	AP
No update	27.26±0.21
Copy weights from student to teacher every 10K iters	28.61±0.18
EMA update at every iter	**31.61±0.28**

Table 5: Comparison between different update rules on *MS-COCO train 2017* with 10% data labeled. The mean and standard deviation over five data splits are reported (the same five splits of *MS-COCO train 2017* as in Sec. 4.1).

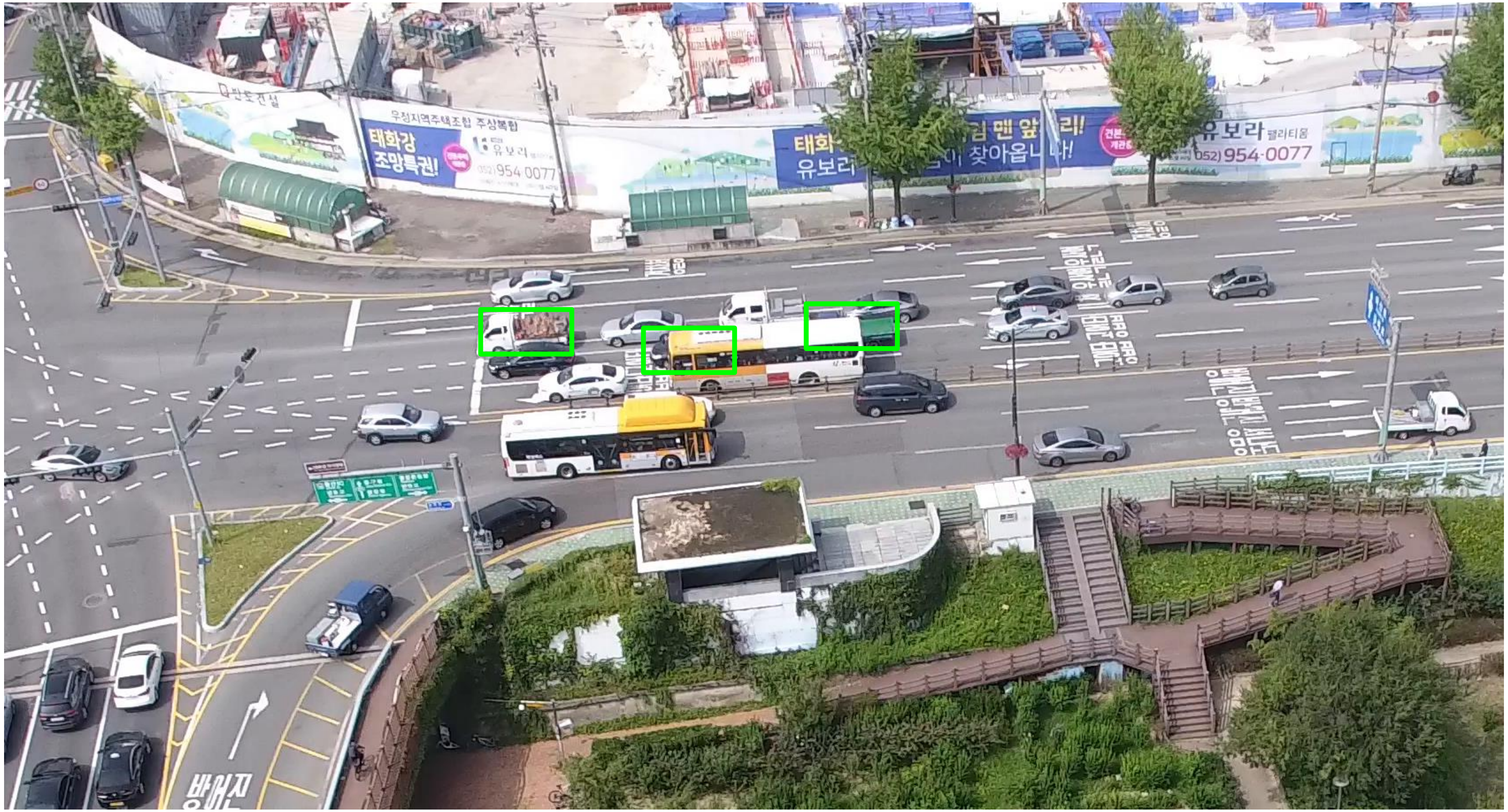
Model	AP
With hard label	27.97±0.13
With soft label	**30.97±0.16**

Table 6: Comparison between training on soft label and hard label when 10% labeled *MS-COCO train 2017* is provided. The mean and standard deviation over five data splits are reported (the same five splits of *MS-COCO train 2017* described in Sec. 4.1).

Conclusion

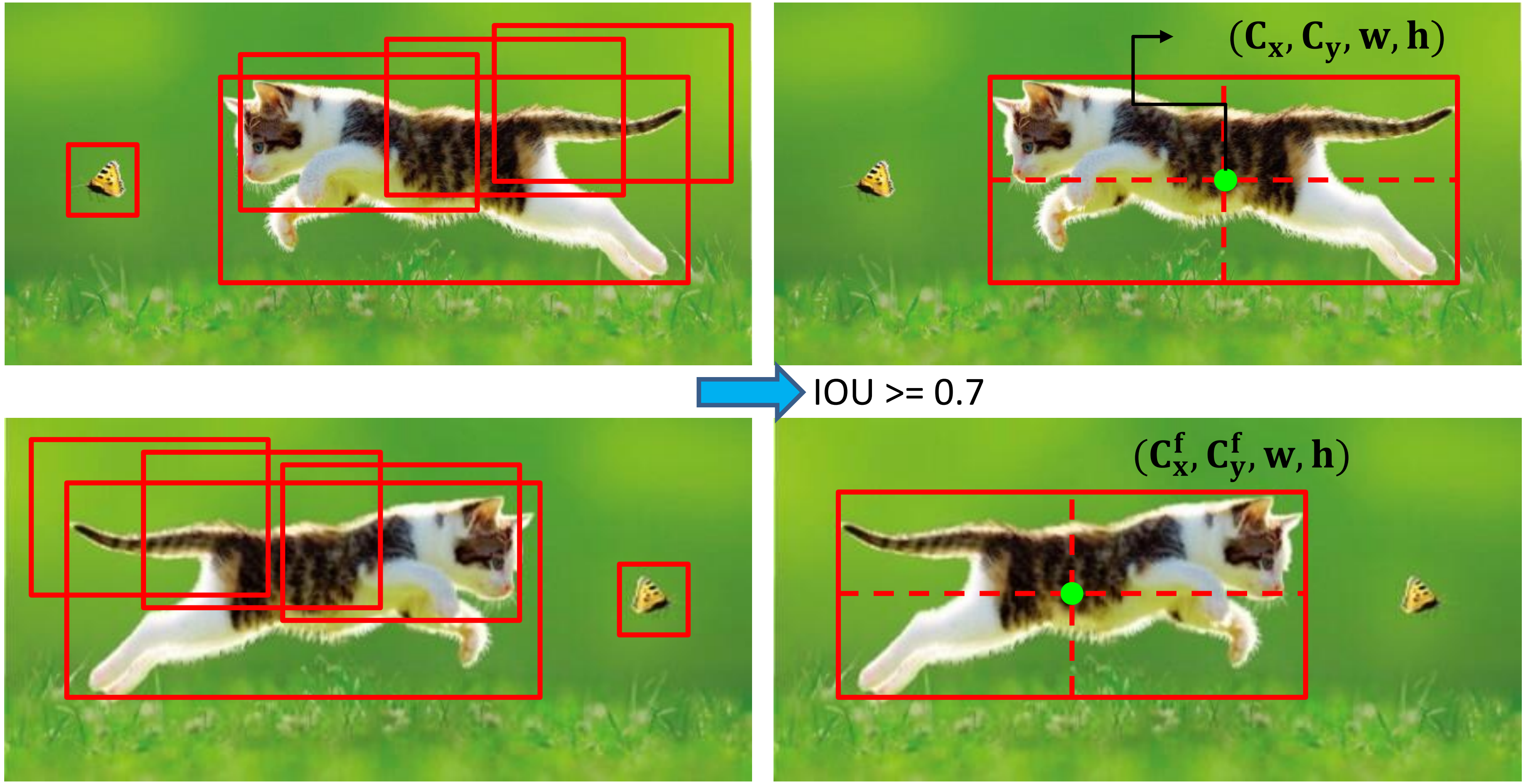
- "Humble Teacher" that obtained state-of-the-art performance on multiple benchmarks
- Demonstrated the effectiveness of our teacher-student model design
- Showed the importance of iteration-wise EMA teacher update

Conclusion



EXAMPLE

RPN Original Image vs RPN flipped Image



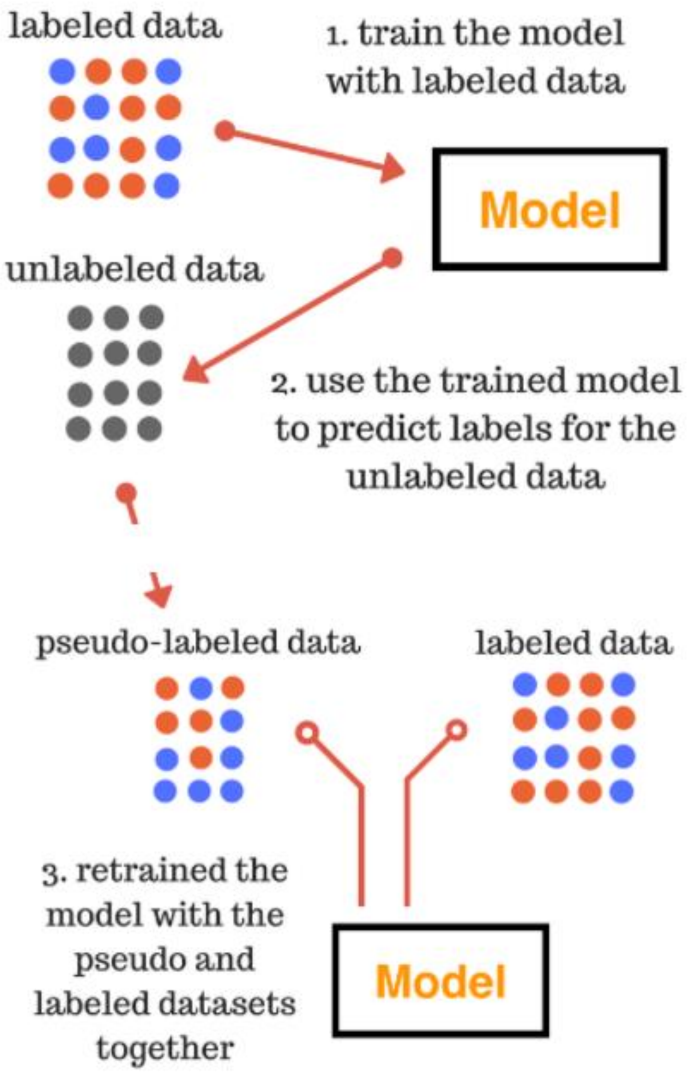
Horizontal flip

$$(C_x, C_y, w, h) \rightarrow (C'_x, C'_y, w, h) = (C_x^f, C_y^f, w, h)$$

Come back

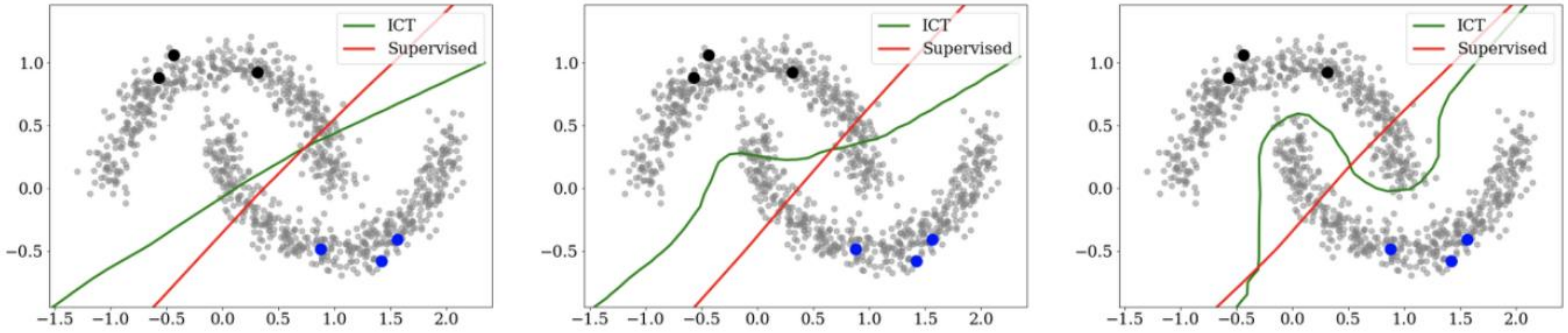
CONCEPT

Pseudo-label

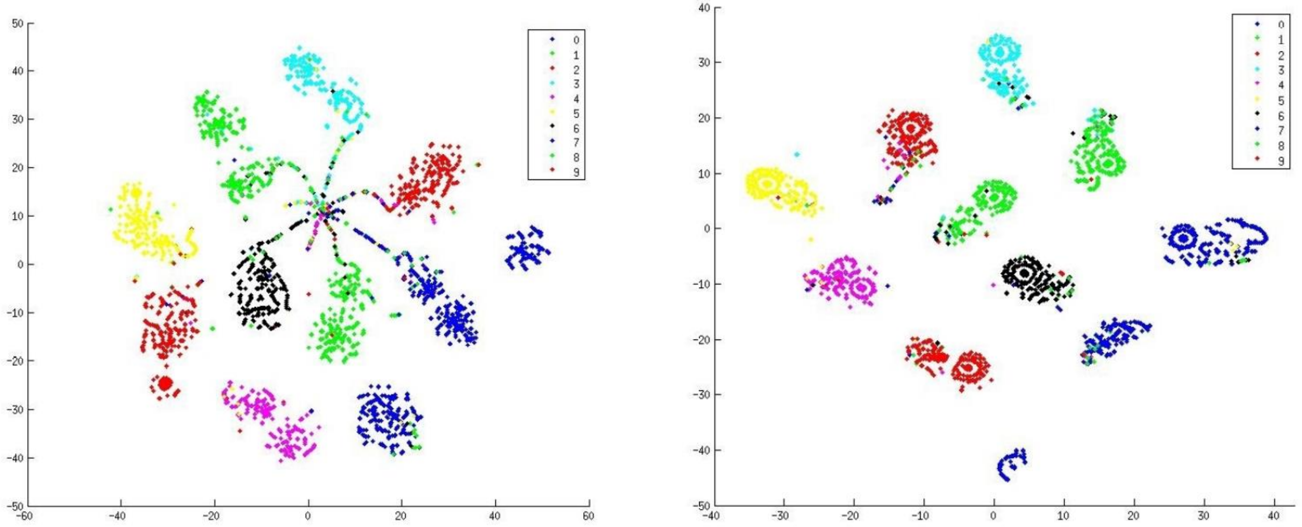


Lee, Dong-Hyun. (2013). Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. ICML 2013 Workshop : Challenges in Representation Learning (WREPL).

- **Low-Density Separation between Classes**



- **Entropy Regularization**



(a) without unlabeled data (dropNN)

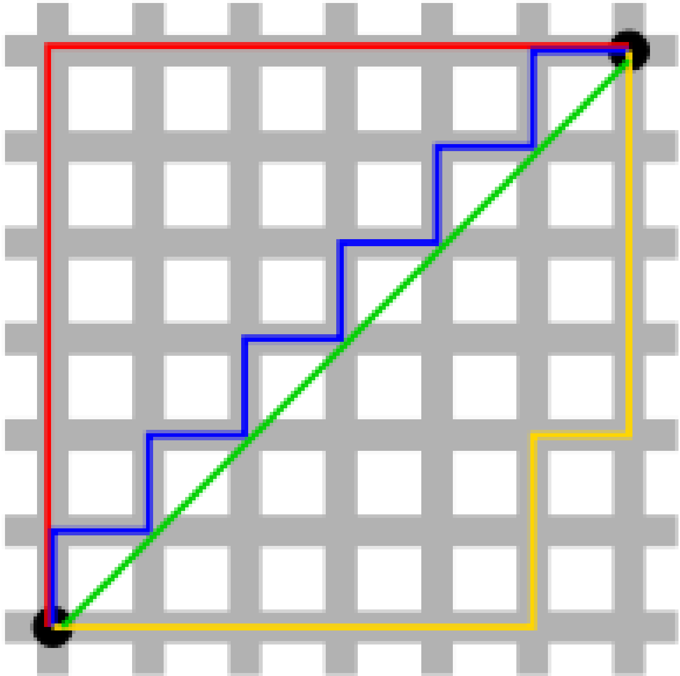
(b) with unlabeled data and Pseudo-Label (+PL)

Norm(L1 & L2)

- Manhattan distance
  - distance, L_1 absolute distance

$$\mathbf{p} = (p_0, p_1, \ldots, p_i) \quad \mathbf{q} = (q_0, q_1, \ldots, q_i)$$

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

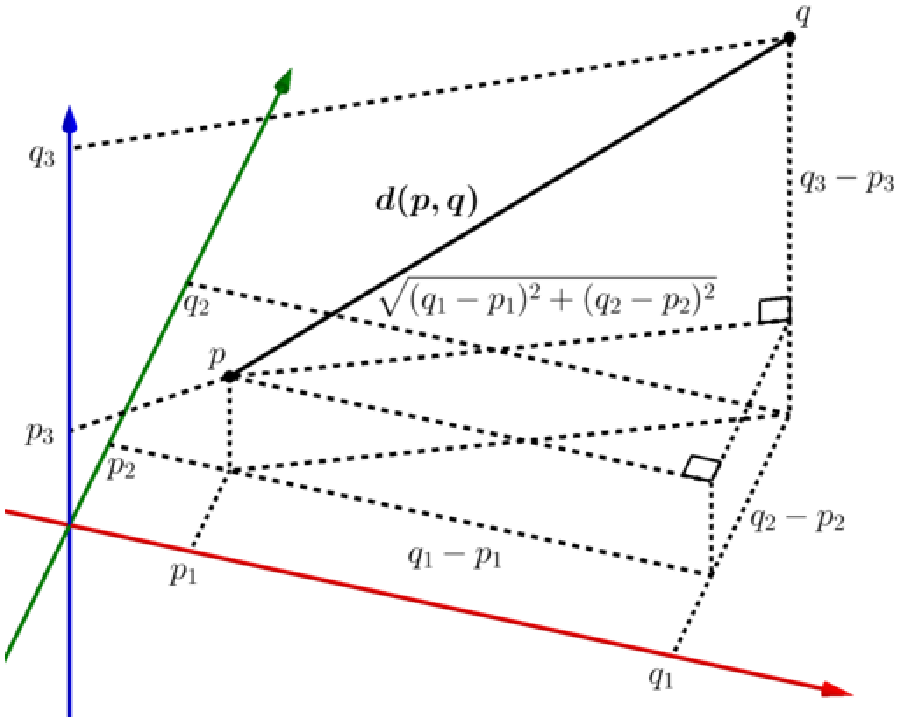


Manhattan street

- Euclidean distance
  - distance, L_2 distance between p and q

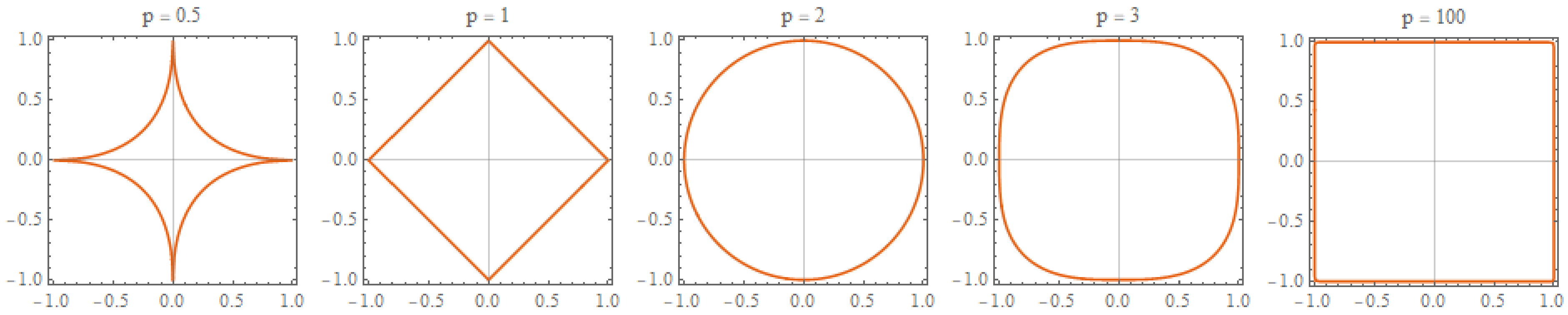
$$\mathbf{p} = (p_0, p_1, \ldots, p_i) \quad \mathbf{q} = (q_0, q_1, \ldots, q_i)$$

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2}$$



- Equation for $L_p - \mathrm{norm}$

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$



- Quantity of information:정보량
- Mass, height, velocity, and etc -> unit amount
- Providing little informational value when high probability event
- Providing high informational value when low probability event

$$\text{Informational value} \propto \frac{1}{P(x)}$$

$$45C6 = 8,145,060 \rightarrow 0.000000122773804\%$$

$$\frac{1}{0.000000122773804} \approx 8,145,060$$

$$\frac{1}{99.9999998772} = 0.01000000001$$

$\text{1000 people to death among 6B people per year}$

번개 맞을 확률 $\frac{1}{600{,}000} \approx 0.000001667\%$

번개 안맞을 정보값 $\frac{1}{99.999998333} \approx 0.01000000016$

Fig. Korea Lottery

- Entropy: Expectation of every event

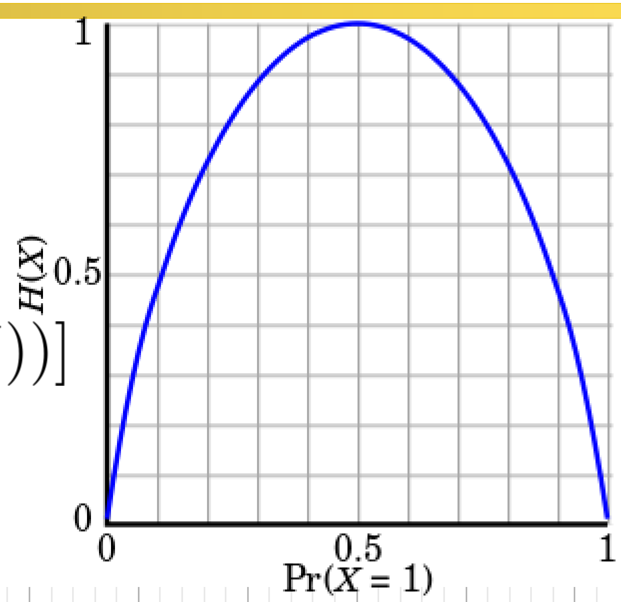
$$\text{Information} = \mathrm{I}(x) = \log\left(\frac{1}{P(x)}\right)$$

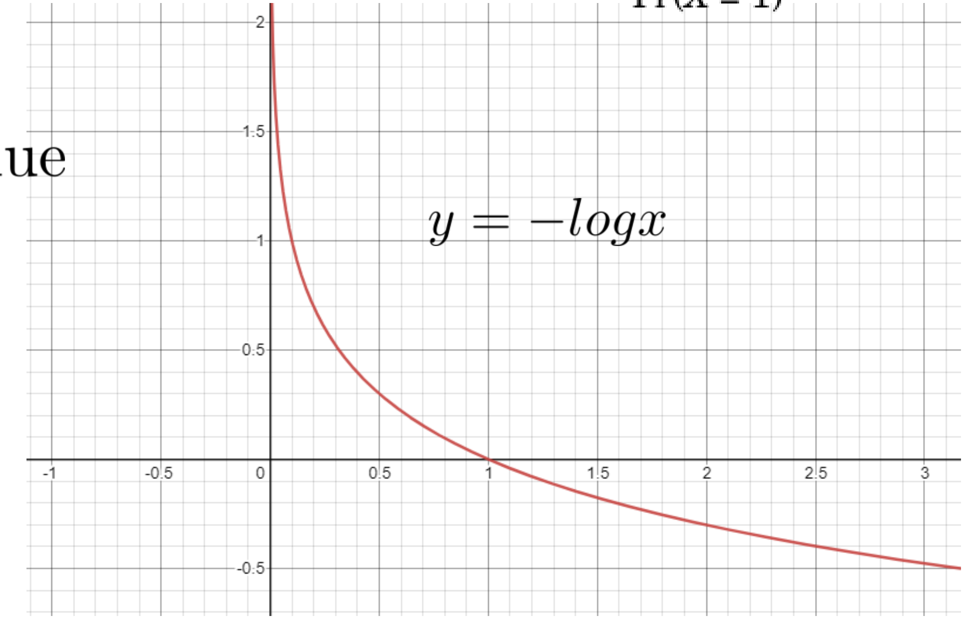
$$\text{Entropy} = \mathrm{H}(X) = \mathbb{E}[\mathrm{I}(X)] = \mathbb{E}[-\log(\mathrm{P}(X))]$$

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i)\log_b \mathrm{P}(x_i)$$

$P(x_i)$: probability of event

-$\log_b P(x_i)$: Informational value





- Between the probability distributions p and q

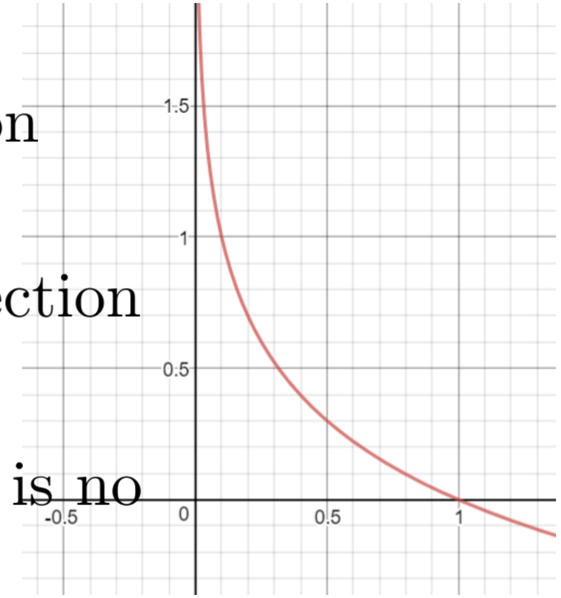
$$-\sum_i L_i log(S_i) = \sum_i L_i * (-log(S_i)) \quad S_i\text{: predicted value}$$

Example

$$L = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ : Label, A is no, B is detection}$$

$$S_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ : System 1, A is no, B is detection}$$

$$S_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ : System 2, A is detection, B is no}$$



-log(x) graph

Source: https://www.desmos.com/calculator/auubsajefh

Cross Entropy

$$-\sum_i L_i log(S_i) = \sum_i L_i * (-log(S_i)) \quad S_i: \text{ predicted value}$$

$$L = \begin{bmatrix} 0 \\ 1 \end{bmatrix} : \text{ real data}$$

$$S_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot -log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0 + 0 = 0$$

$$S_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot -log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} = 0 + \infty = \infty$$

- Small cost is the optimal point to the system
- Prediction score is denoted by cross entropy
  wrong prediction -> high cost, right prediction -> lower cost
- Goal of loss function that computes minimum cost for finding the point

- Kullback-Leibler divergence → relative entropy
  - 두 확률분포의 차이

$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log_b \left(\frac{P(x)}{Q(x)} \right)$$

$$= - \sum_{x \in \mathcal{X}} P(x) \log_b \left(\frac{Q(x)}{P(x)} \right)$$

$$= - \sum_{x \in \mathcal{X}} P(x) \log_b Q(x) - \sum_{x \in \mathcal{X}} P(x) \log_b \frac{1}{P(x)}$$

Example of KL-Divergence

- Real four sided shapes dices

$$p = (1/4, 1/4, 1/4, 1/4)$$

- YK expected dices distribution

$$q = (1/2, 1/4, 1/8, 1/8)$$

$$\text{A.} -\sum_x p(x) log_2 q(x)$$

$$= -\frac{1}{4} * log_2(0.5) - \frac{1}{4} * log_2(0.25) - \frac{1}{4} * log_2(0.125) - \frac{1}{4} * log_2(0.125) = 2.25$$

$$\text{B.} -\sum_x p(x) log_2 p(x)$$

$$= -\frac{1}{4} * log_2(0.25) - \frac{1}{4} * log_2(0.25) - \frac{1}{4} * log_2(0.25) - \frac{1}{4} * log_2(0.25) = 2$$

$$\mathrm{D}_{KL}(p \parallel q) = (-\sum_x p(x) log_2 q(x)) - (-\sum_x p(x) log_2 p(x)) = \left(-\sum_x p(x) log_2 \frac{q(x)}{p(x)}\right)$$

$$-\sum_x p(x) log_2 \frac{q(x)}{p(x)} = -\sum_x p(x)(log_2 q(x) - log_2 p(x)) = 2.25 - 2 = 0.25$$

KL-Divergence

$$KL(p \parallel q) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} \mathrm{d}x$$

$$p(x) = N(\mu_1, \sigma_1^2) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right)$$

$$q(x) = N(\mu_2, \sigma_2^2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right)$$

(i) p(x), p(x)

$$KL(p \parallel p) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{p(x)} \mathrm{d}x = \int_{-\infty}^{\infty} p(x) \ln(1) \mathrm{d}x = 0$$

(ii) p(x), q(x)

$$KL(p \parallel q) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} \mathrm{d}x$$

$$= \ln\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

- 확률변수의 기대값
  - 개별 가중치에 대해 곱해 구해지는 평균
  - 합리적인 평균 계산법
  - 극단적인 값에 영향이 적음

$$\sum_{i=1}^{n} x_i p(x_i) = E[X]$$

  - 예시
    - 게임에서 이길 확률은 0.99 입니다. 만약 이기면 100원을 받고, 지면 100,000원을 잃습니다. 확률변수 X는 게임에서 얻는 돈의 양으로 정의하겠습니다. X의 기대값 E[X]는 무엇일까?
    - E[X] = 0.99*100-0.01*100,000 = -901
    - 한 게임당 얻는 돈의 기대값은 -901원이 되어, 게임을 무수히 많이 진행하게 되면 결과적으로 돈을 잃게 됩니다.

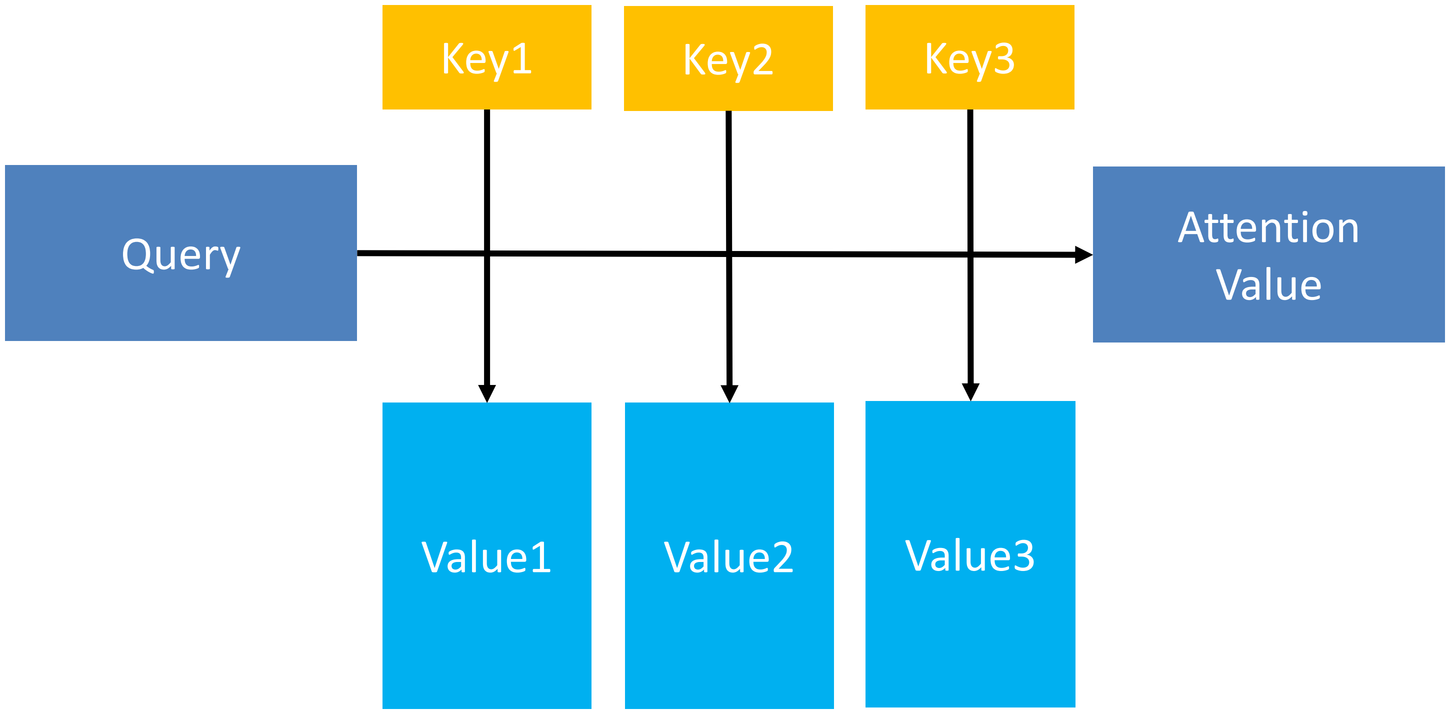
- 지도학습(Supervised Learning)
  - 정답이 있는 데이터를 학습
- 비지도학습(Unsupervised Learning)
  - 정답이 없는 데이터를 학습
- 준지도학습(Semi-Supervised Learning)
  - 정답 레이블이 적은 데이터셋으로 1차 (지도)학습 후 정답 레이블이 없는 많은 데이터셋으로 2차 학습

TRANSFORMER

Attention Mechanism

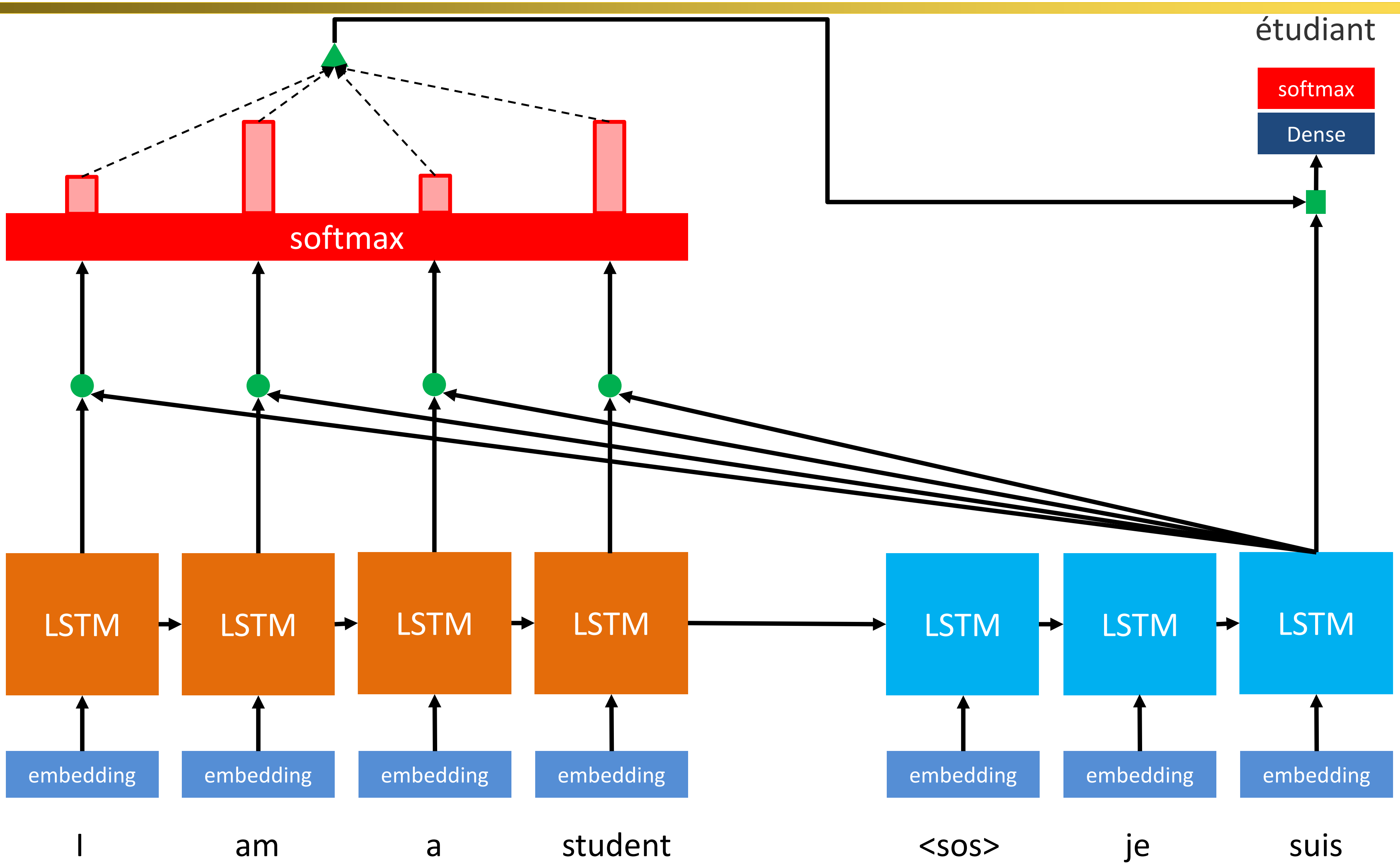
- The result that is to predict every time step in decoder refers the information from encoder
- Different weights for
- Machine Translation: Encoder-Decoder structure
- Attention(Q,K,V) = Attention Value
- Q, Query: decoder hidden status at time t
- K, Keys: encoder hidden status at every time
- V, Values: encoder hidden status at every time

- Attention(Q,K,V) = Attention Value
- Q, Query: decoder hidden status at time t
- K, Keys: encoder hidden status at every time
- V, Values: encoder hidden status at every time



https://wikidocs.net/22893

Dot-Product Attention

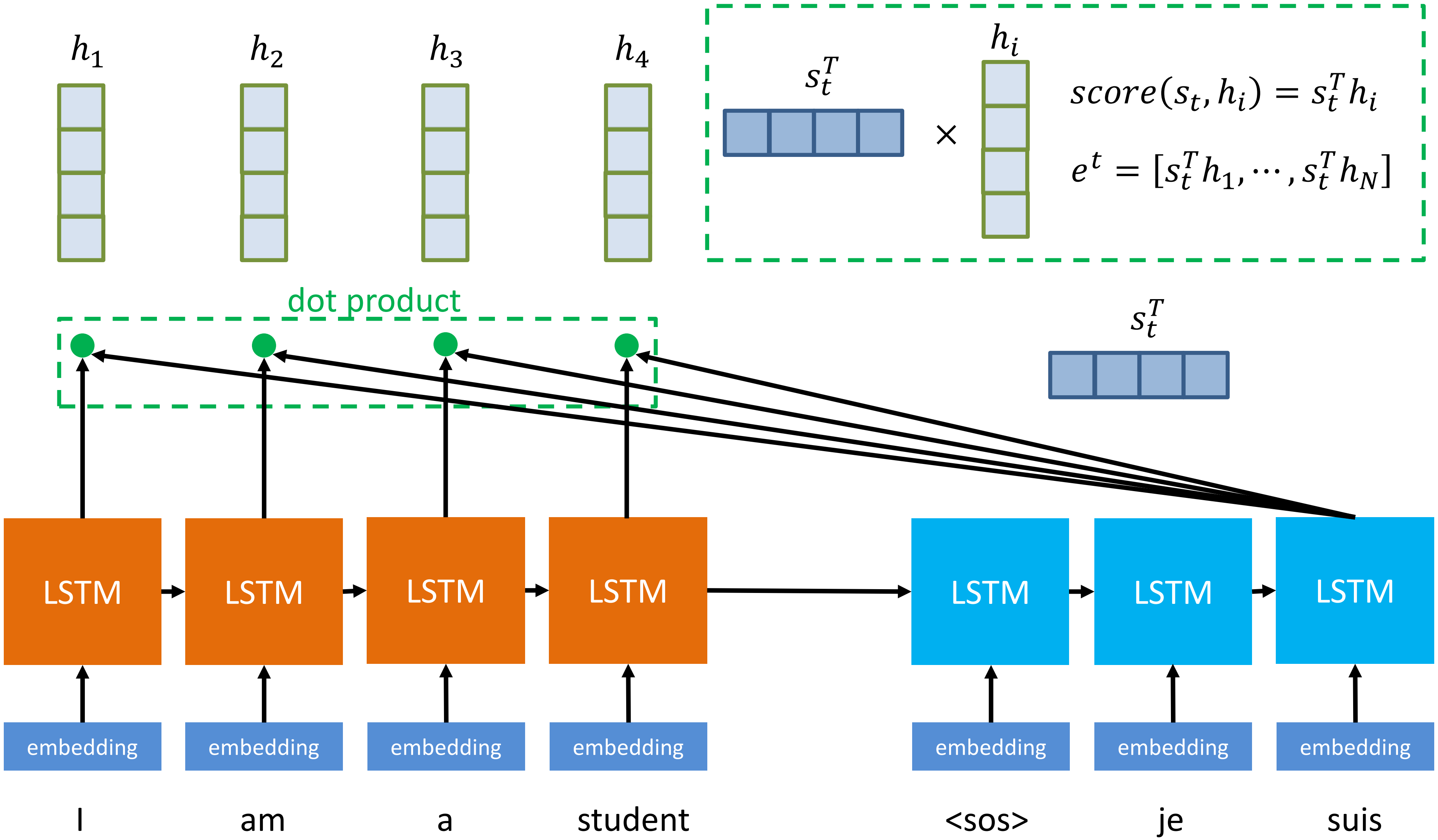


https://wikidocs.net/22893

1. Compute attention score
2. Create attention distribution from softmax
3. Compute attention value with attention weight and hidden status
4. Concatenate attention value and decoder of hidden status at time t
5. Compute output value, $\tilde{s}_t$
6. Generate softmax($\tilde{s}_t$)

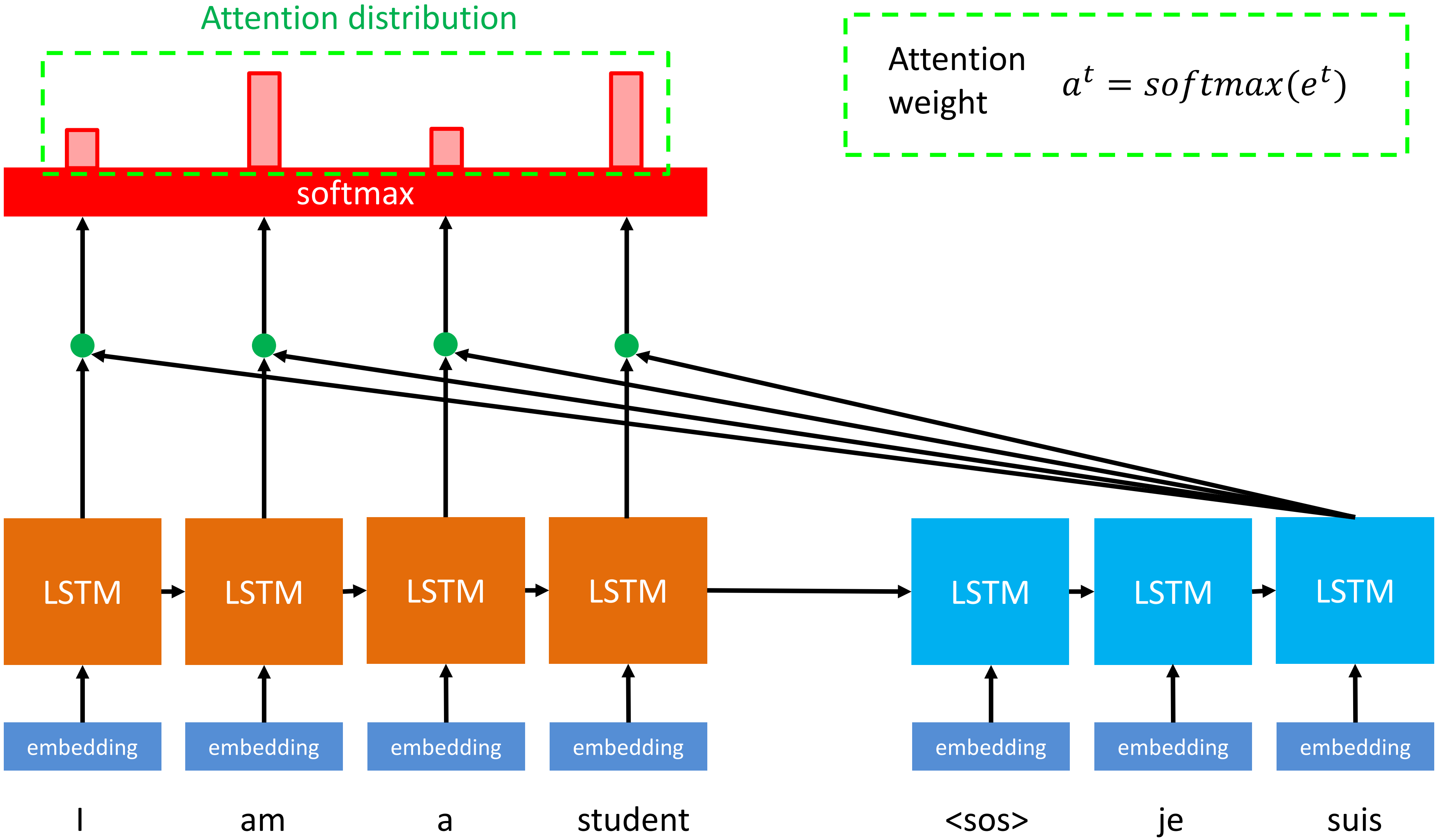
https://wikidocs.net/22893

Compute attention score



https://wikidocs.net/22893

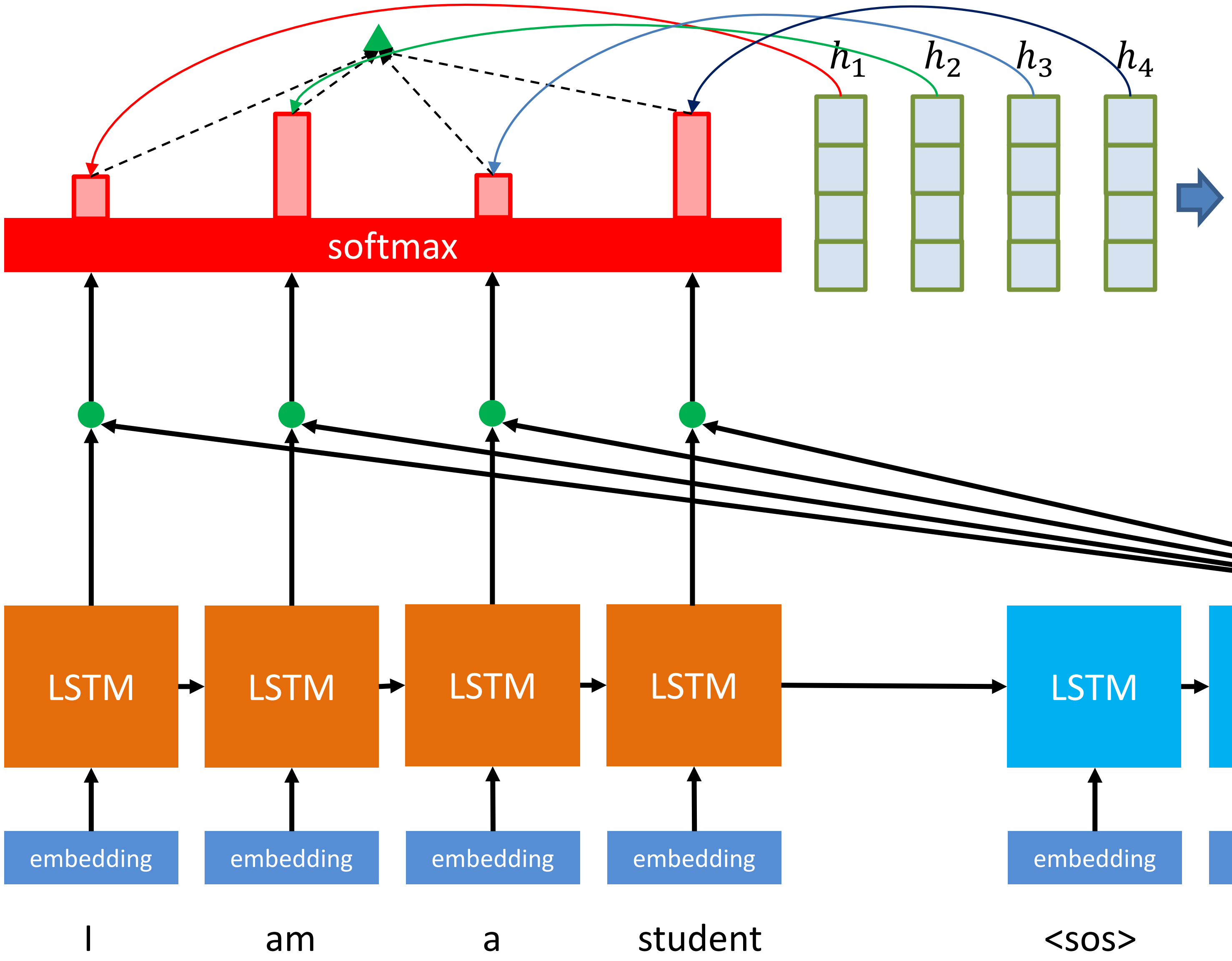
Create attention distribution from softmax



Attention weight $a^t = softmax(e^t)$

https://wikidocs.net/22893

Compute attention value with attention weight and hidden status

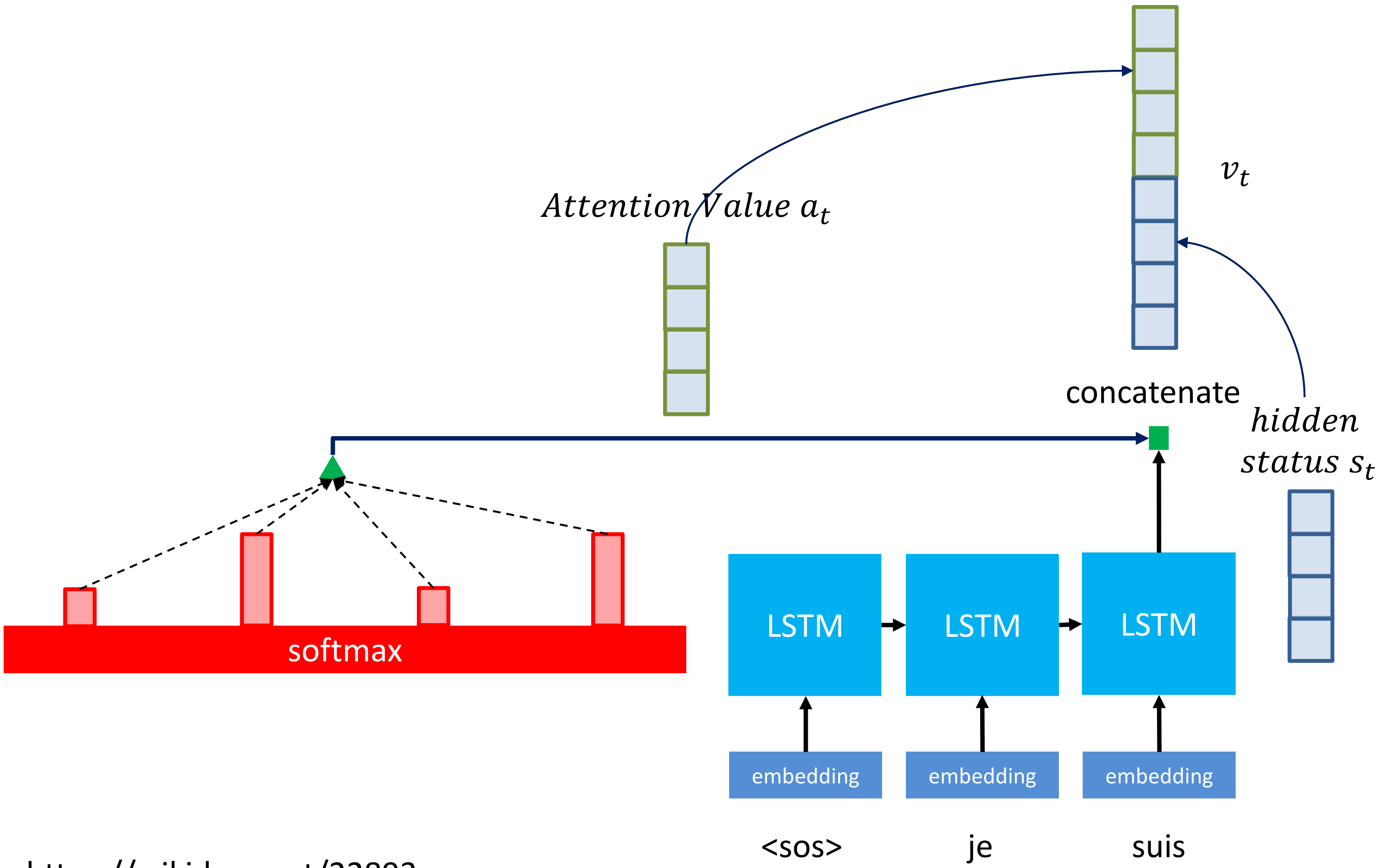


$context\ vector \rightarrow$

$$a^t = \sum_{i=1}^{N} a_i^t h_i$$

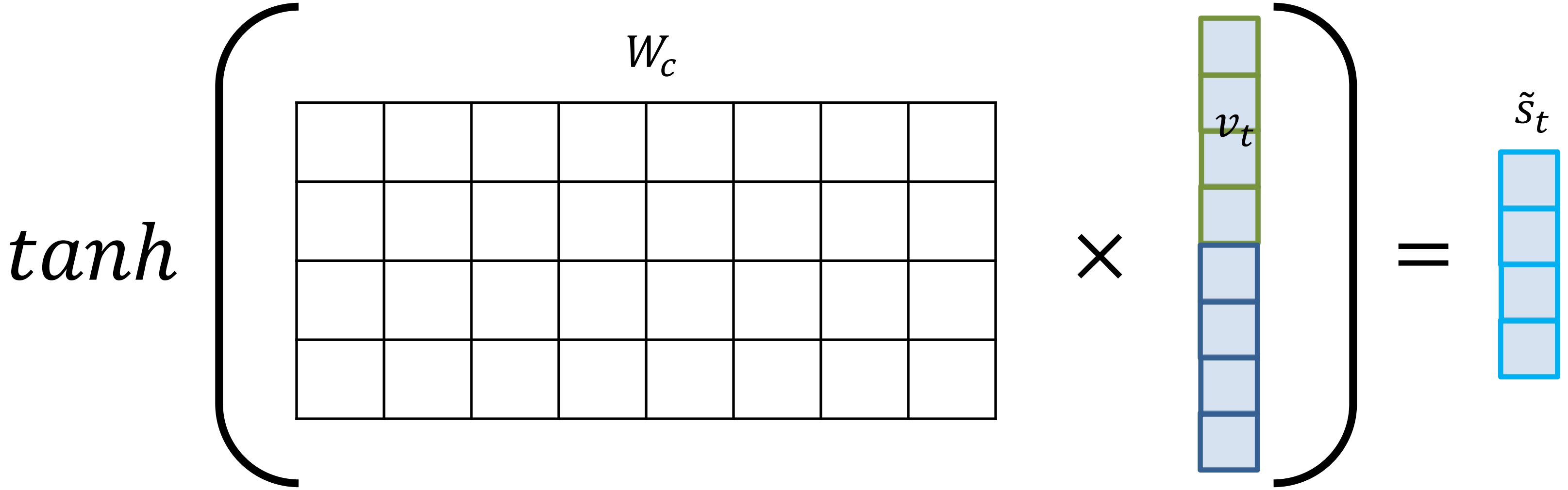
https://wikidocs.net/22893

Concatenate attention value and decoder of hidden status at time t



https://wikidocs.net/22893

- Before coming out from output, apply to compute neural network

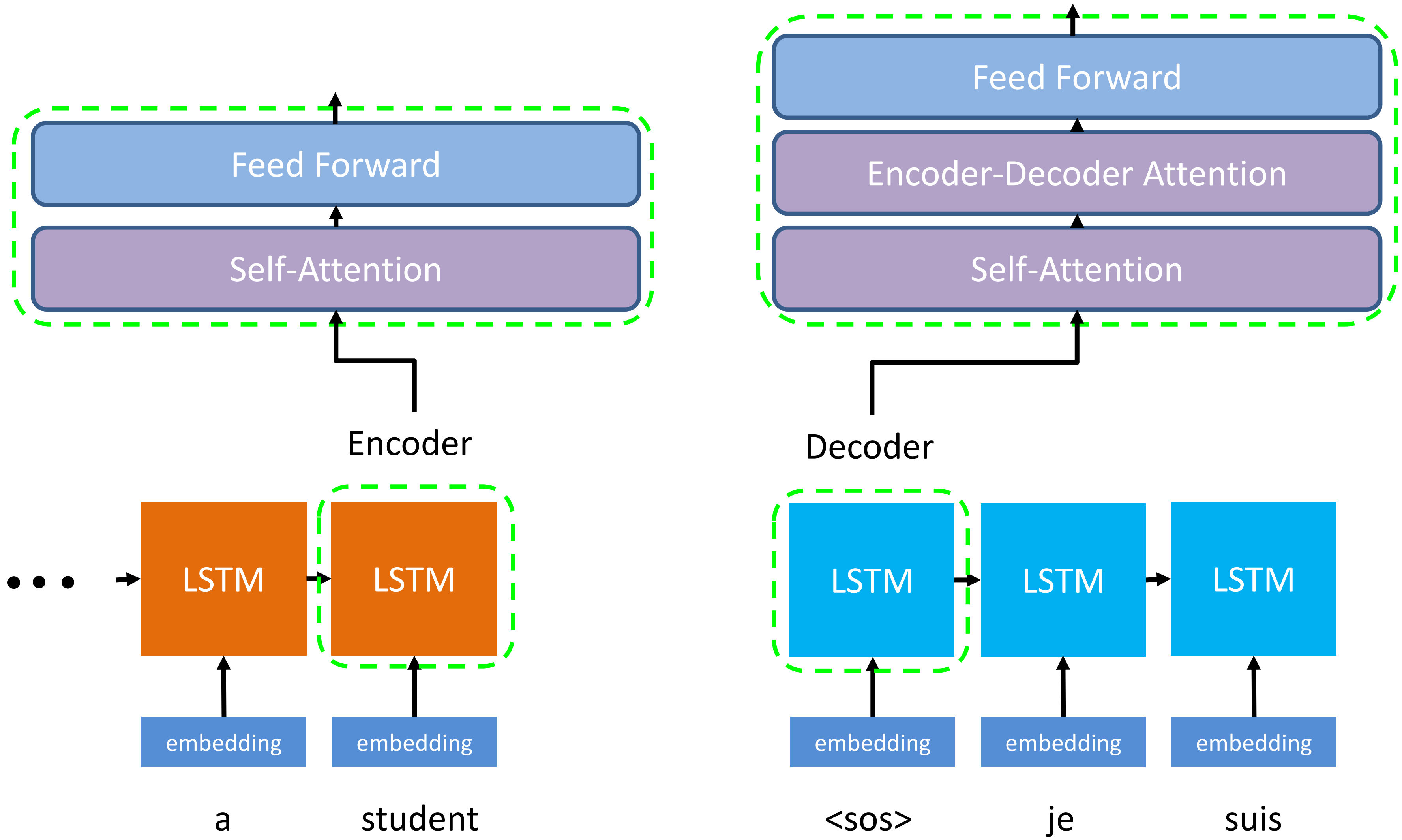


$$\tilde{s}_t = \tanh(W_c[a_t \cdot s_t] + b_c)$$

$$\hat{y}_t = Softmax(W_y\tilde{s}_t + b_y)$$

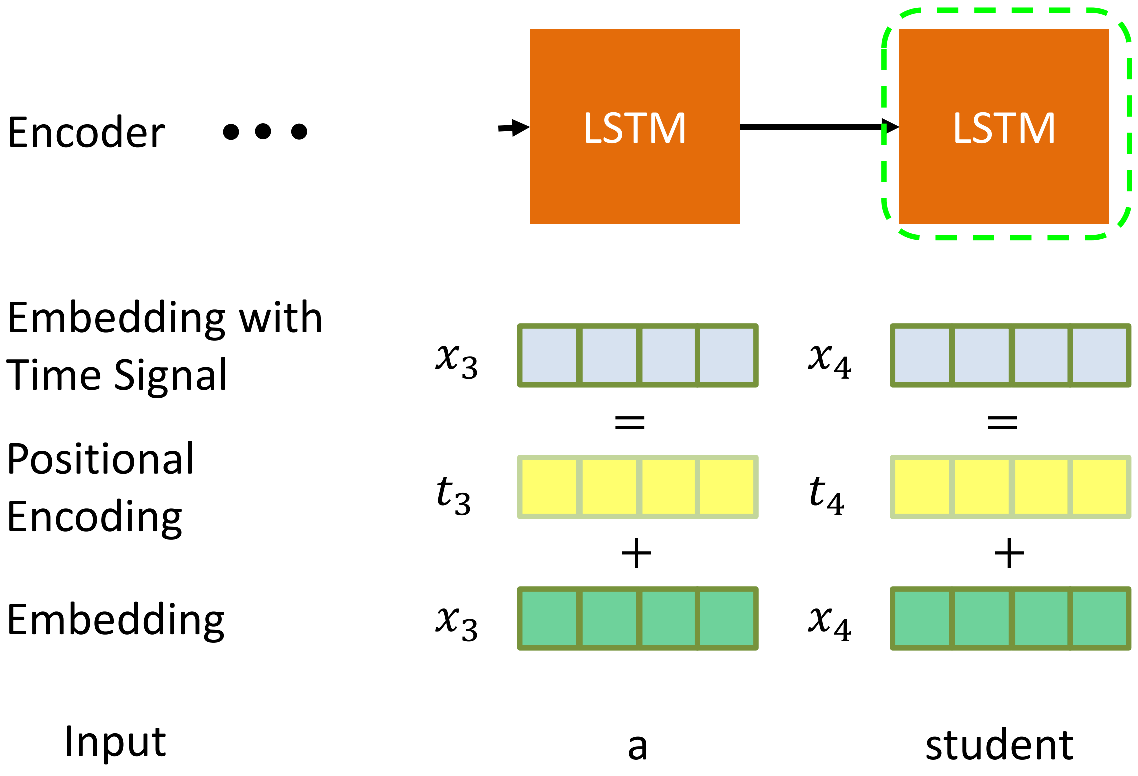
https://wikidocs.net/22893

Encoder-Decoder



Refer: https://wdprogrammer.tistory.com/72

- Original embedding: no consider of position
- Positional embedding in Transformer



Refer: https://wdprogrammer.tistory.com/72

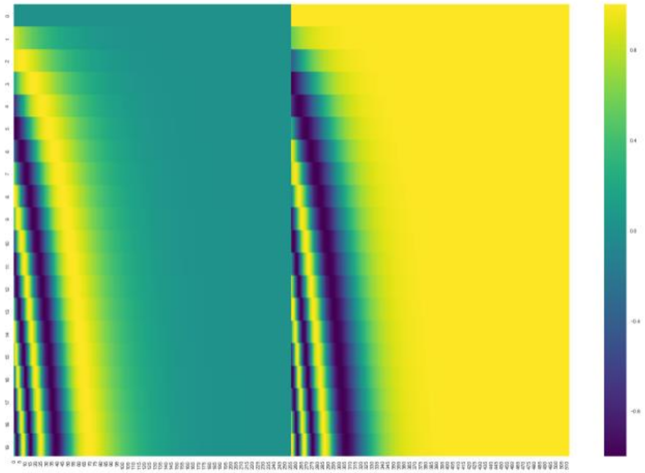
Input Embedding

Positional Encoding	t_3	0.84	0.001	0.54	1	t_4	0.91	0.002	-0.42	1

+ +

Embedding x_3 x_4

Input a student

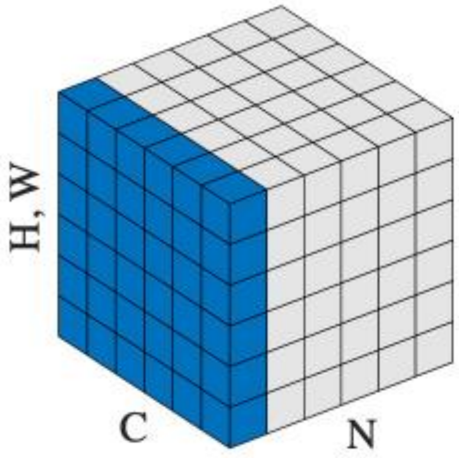


Visualize position encoding

Refer: https://wdprogrammer.tistory.com/72

- $y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta$
- Channel based normalization

```
>>> # NLP Example
>>> batch, sentence_length, embedding_dim = 20, 5, 10
>>> embedding = torch.randn(batch, sentence_length, embedding_dim)
>>> layer_norm = nn.LayerNorm(embedding_dim)
>>> # Activate module
>>> layer_norm(embedding)
>>>
>>> # Image Example
>>> N, C, H, W = 20, 5, 10, 10
>>> input = torch.randn(N, C, H, W)
>>> # Normalize over the last three dimensions (i.e. the channel and spatial dimensions)
>>> # as shown in the image below
>>> layer_norm = nn.LayerNorm([C, H, W])
>>> output = layer_norm(input)
```



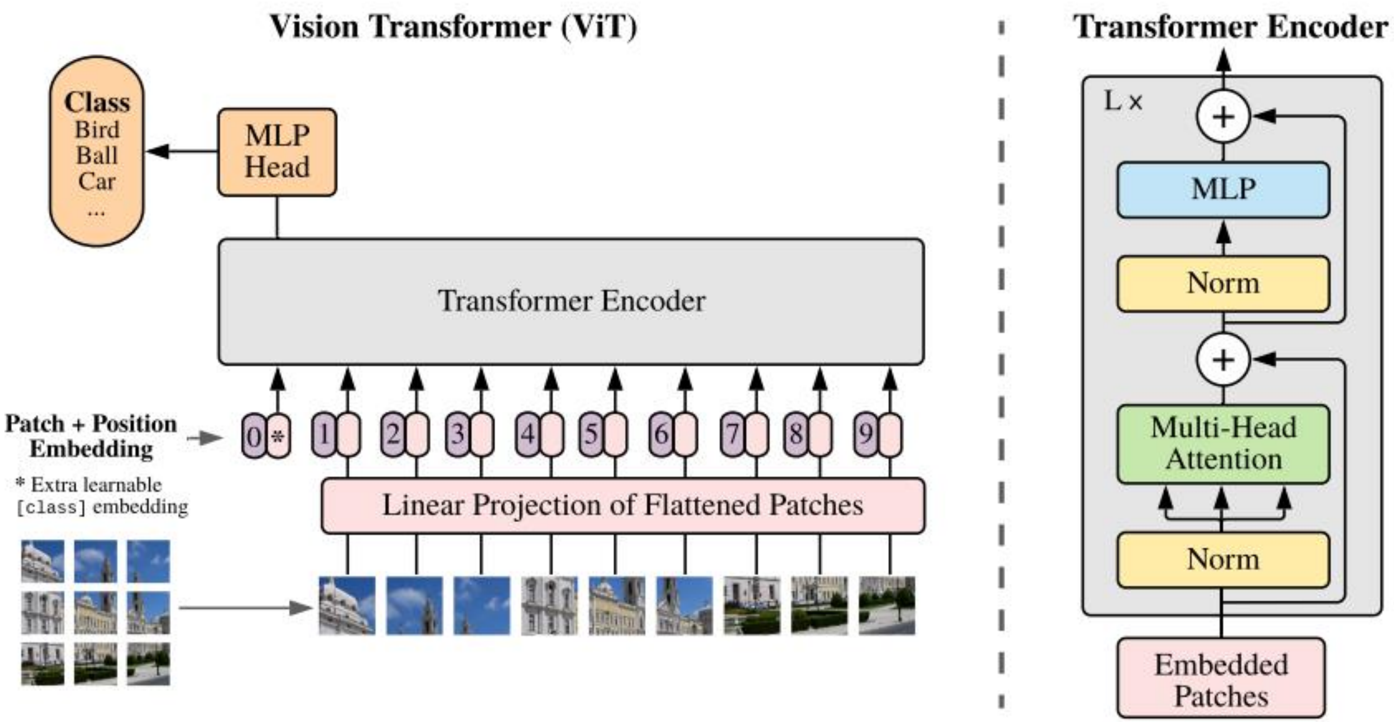


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

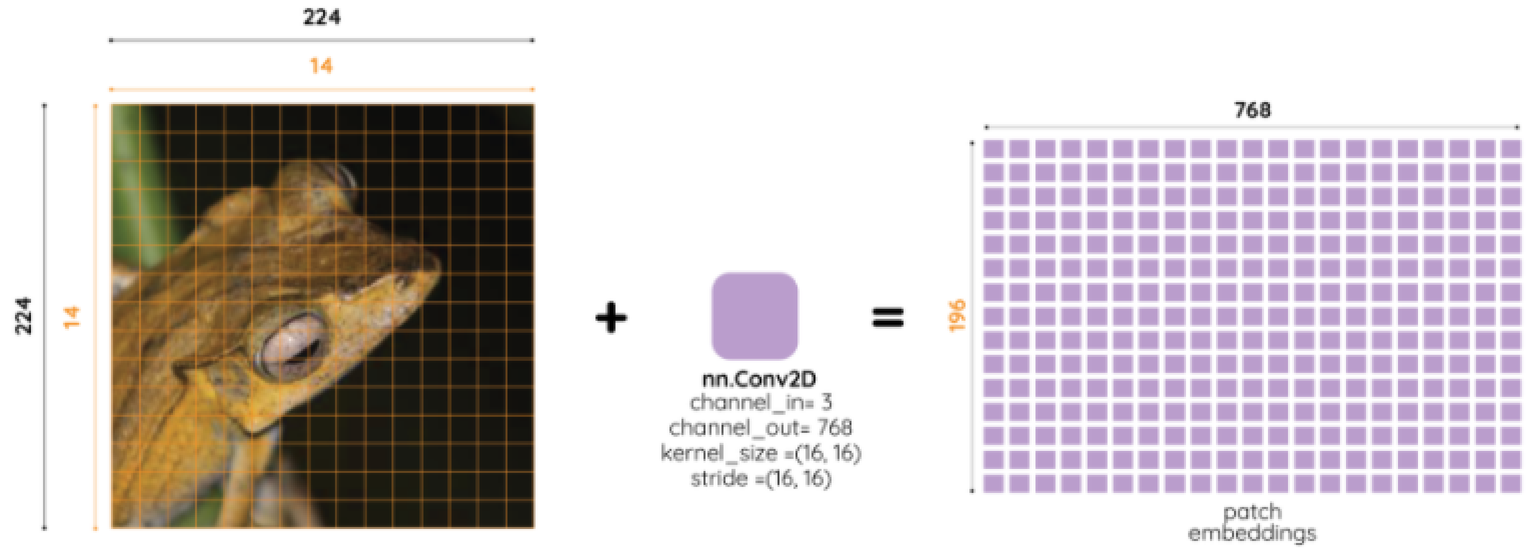
A. Dosovitskiy, L. Beyer, A. Kolesnikov, "An Image is Worth 16 x 16 Words: Transformers for Image Recognition at Scale", 2021, ICLR

Vision Transformer

- Input size: 3 x 224 x 224
- Patch size: 14 x 14

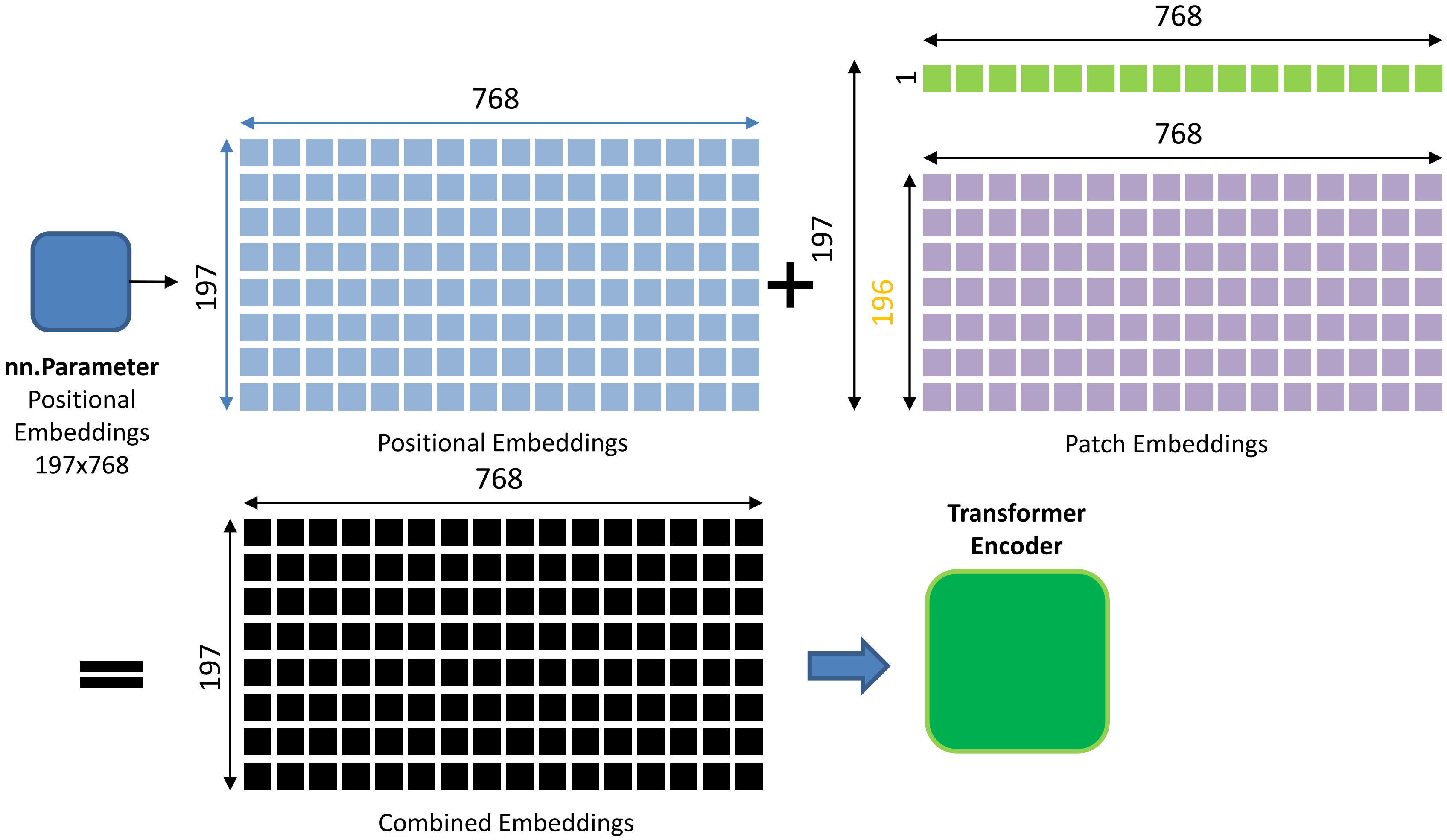
```
# Input image [B, C, H, W]
x = torch.randn(1, 3, 224, 224)

# 2D conv
conv = nn.Conv2d(3, 768, 16, 16)
x = conv(x) # [B, 768, 14, 14]
x = x.reshape(B, -1, 196).transpose(1, 2) # [B, 196, 768]
```



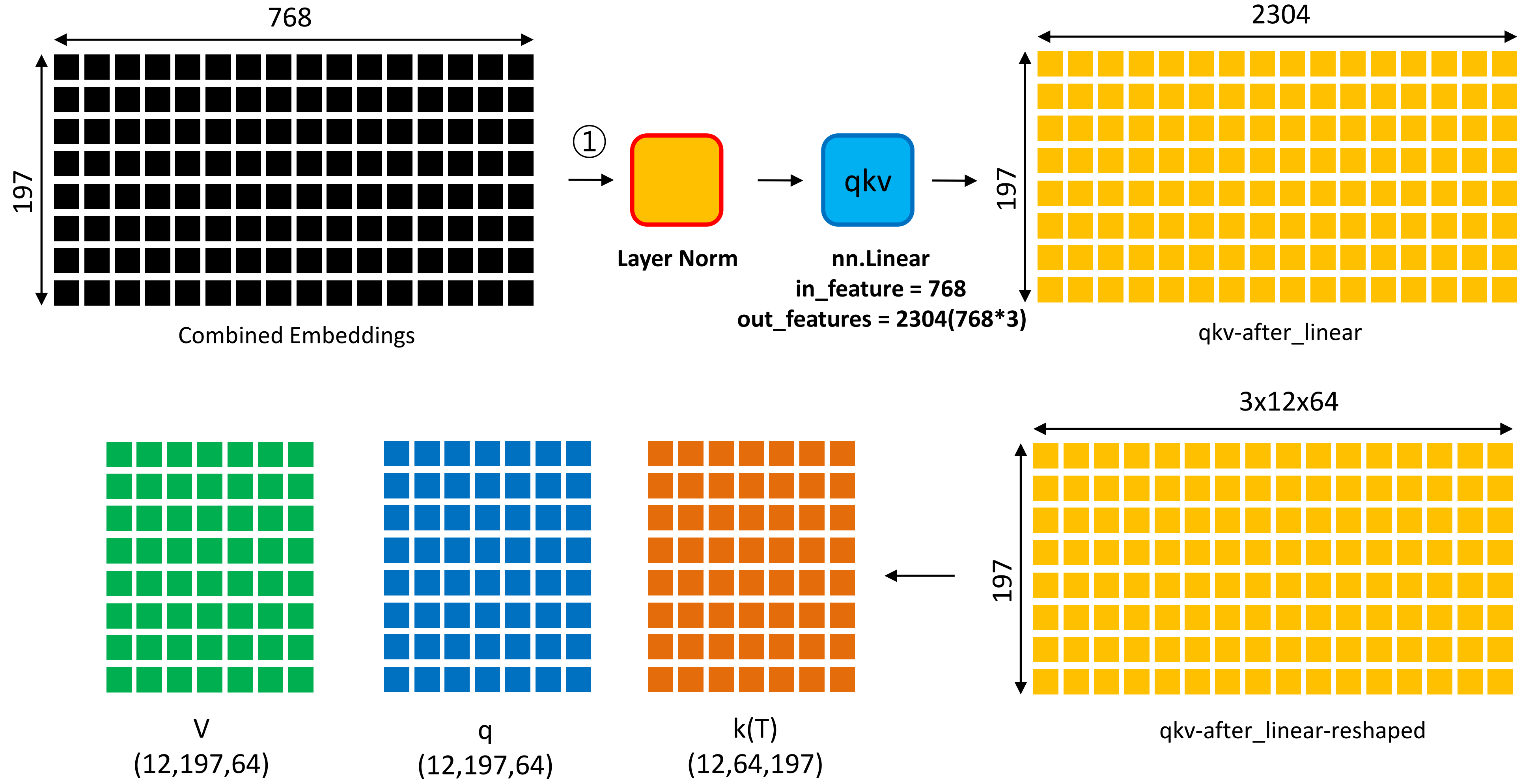
A. Dosovitskiy, L. Beyer, A. Kolesnikov, "An Image is Worth 16 x 16 Words: Transformers for Image Recognition at Scale", 2021, ICLR

- Classification 결과값은 MLP(Multi layer perceptron)를 통해 1 x 768 tensor를 생성함
- 최종 텐서의 크기(196 + 1) x 768

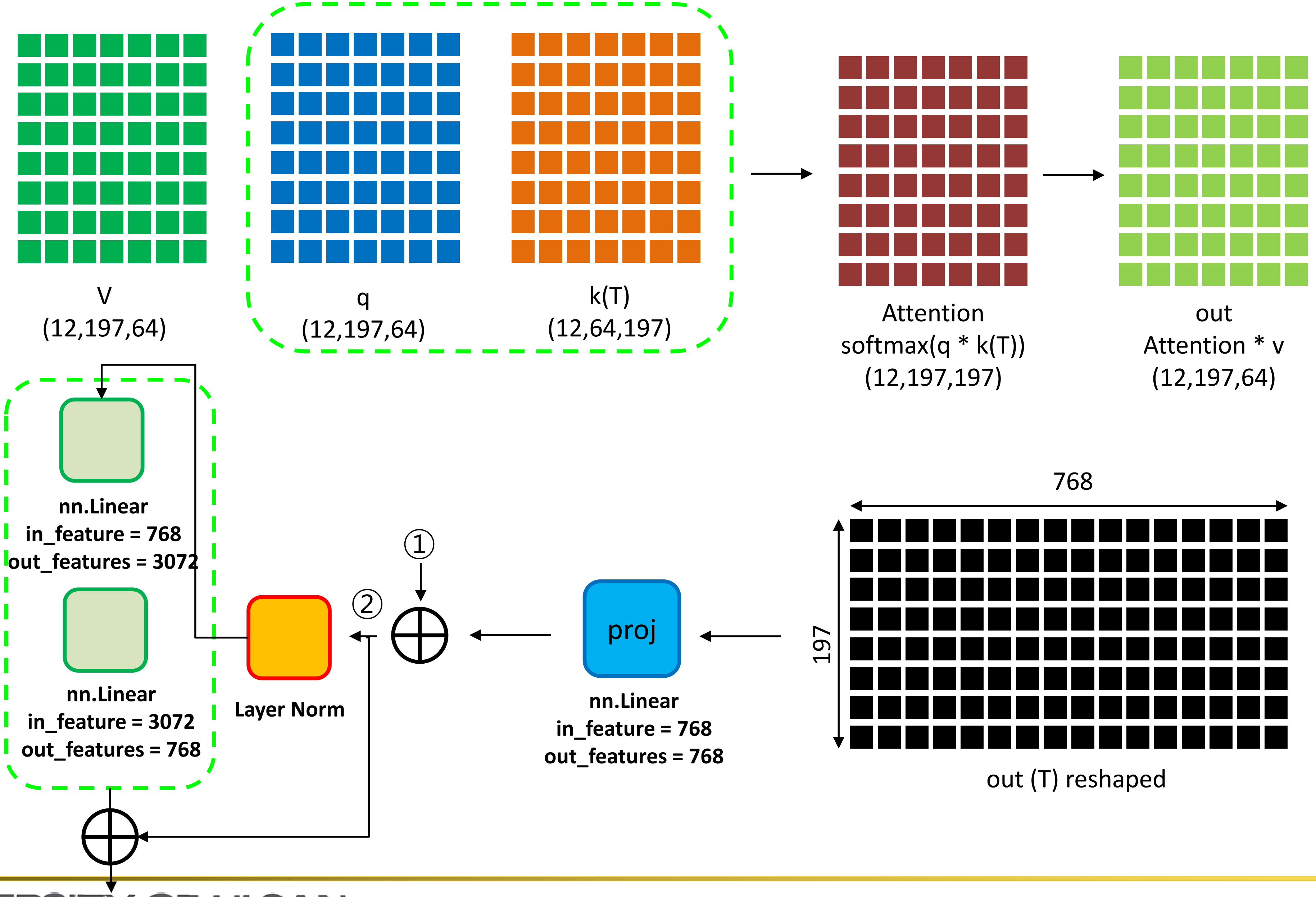


A. Dosovitskiy, L. Beyer, A. Kolesnikov, "An Image is Worth 16 x 16 Words: Transformers for Image Recognition at Scale", 2021, ICLR

Vision Transformer

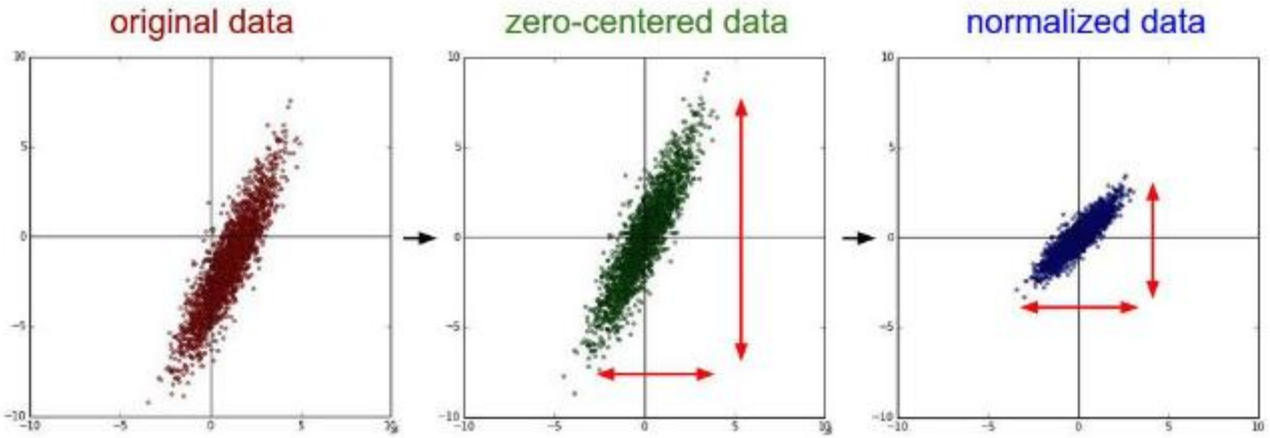


Vision Transform



Batch Normalization without Shifting(1/2)

- Advantage(Batch Normalization)
  - Speed up training time
  - Reduce sensitivity of weight initialization
  - Model regularization
  - Data normalization between 0 and 1



Normalization (Standardization)

Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“, ICML2015

Batch Normalization without Shifting(2/2)

- Input: Values of x over a mini-batch: $\mathcal{B} = x_{1\ldots m}$

  Parameters to be learned: γ, β
- Output: $y_i = \mathbf{BN}_{\gamma,\beta}(x_i)$

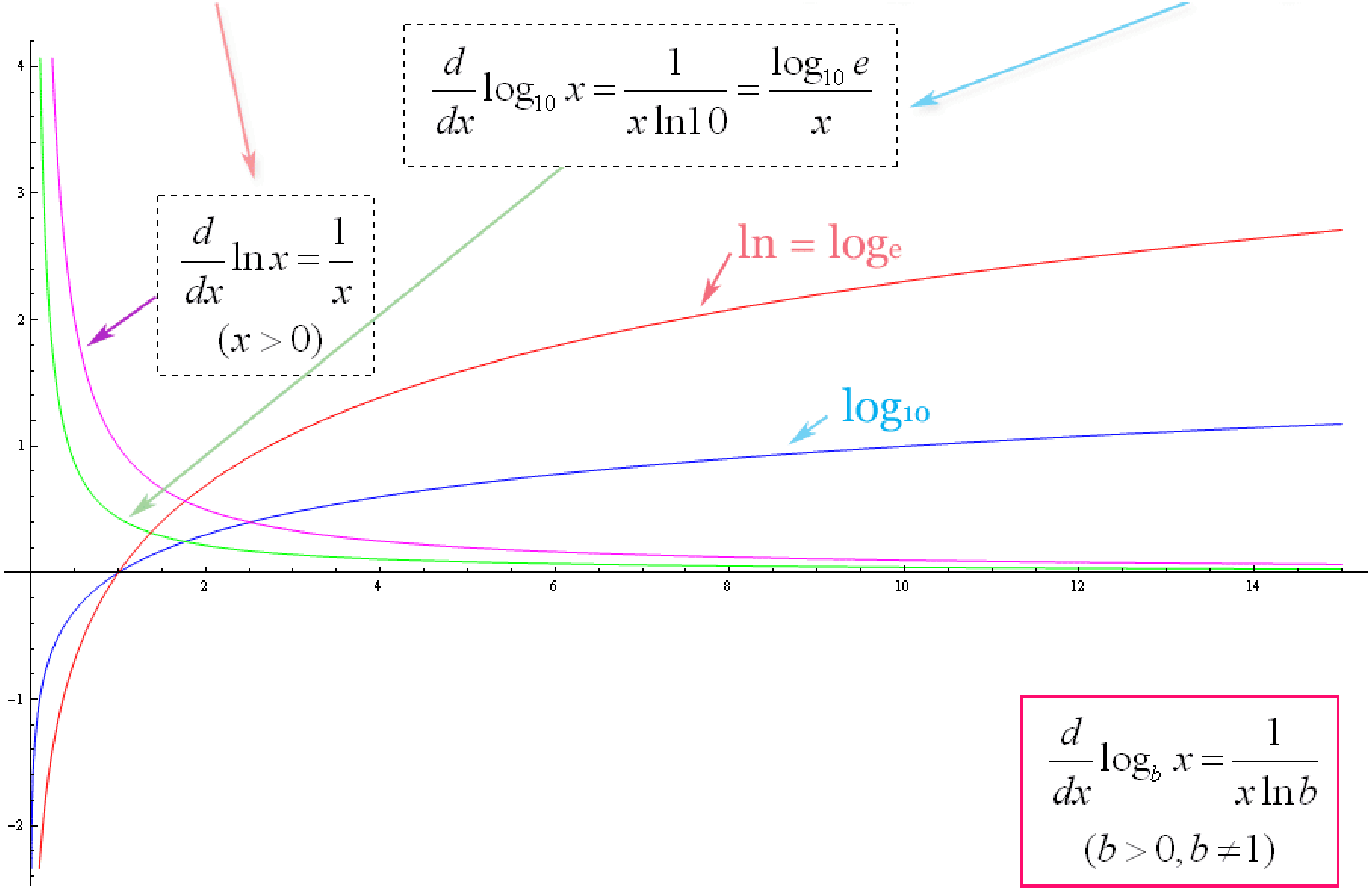
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$ Mini-batch mean

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$ Mini-batch variance

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$ Normalize(zero-centered)
:zero score normalization

$$y_i \leftarrow \gamma\hat{x}_i + \cancel{\beta} \equiv \mathrm{BN}_{\gamma,\cancel{\beta}}(x_i)$$ Scale and ~~shift~~

Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML2015

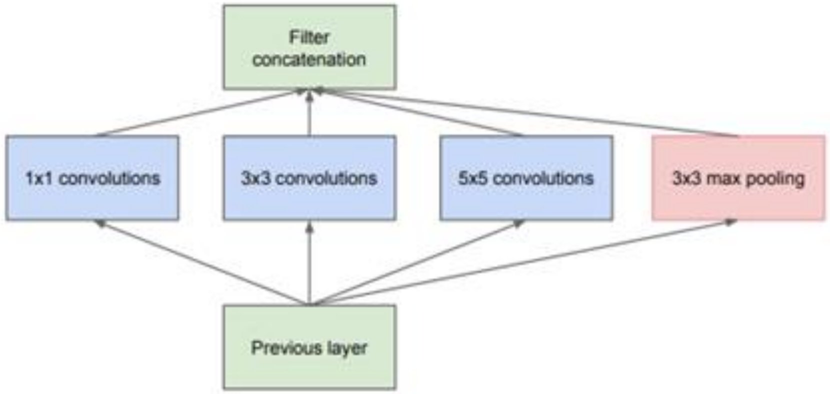


$$\frac{d}{dx}\log_{10} x = \frac{1}{x\ln 10} = \frac{\log_{10} e}{x}$$

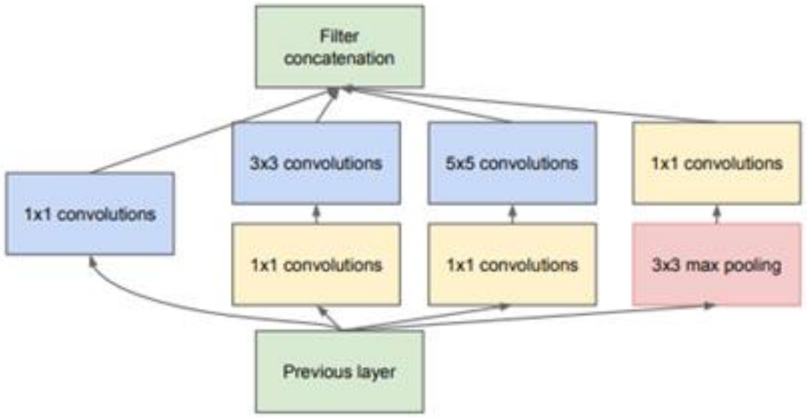
$$\frac{d}{dx}\ln x = \frac{1}{x}$$
$$(x > 0)$$

$$\frac{d}{dx}\log_b x = \frac{1}{x\ln b}$$
$$(b > 0, b \neq 1)$$

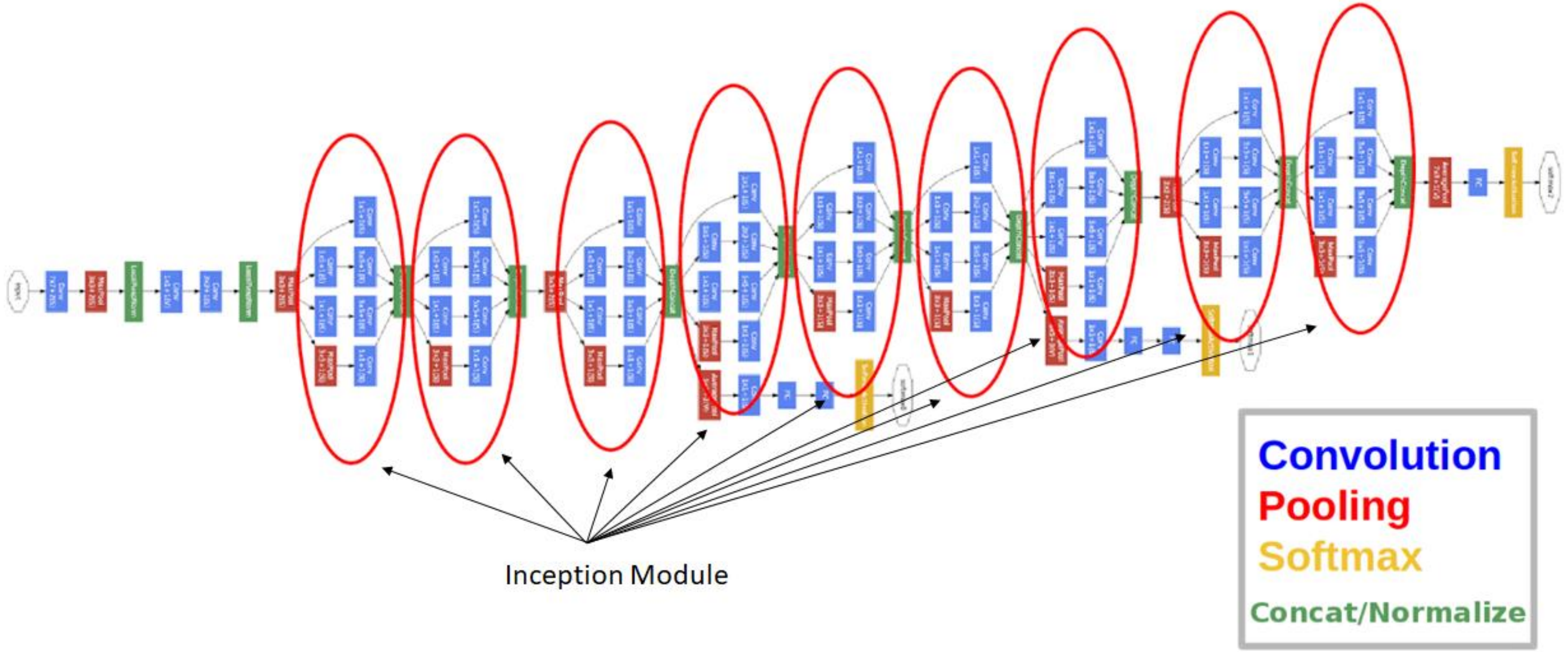
InceptionNet



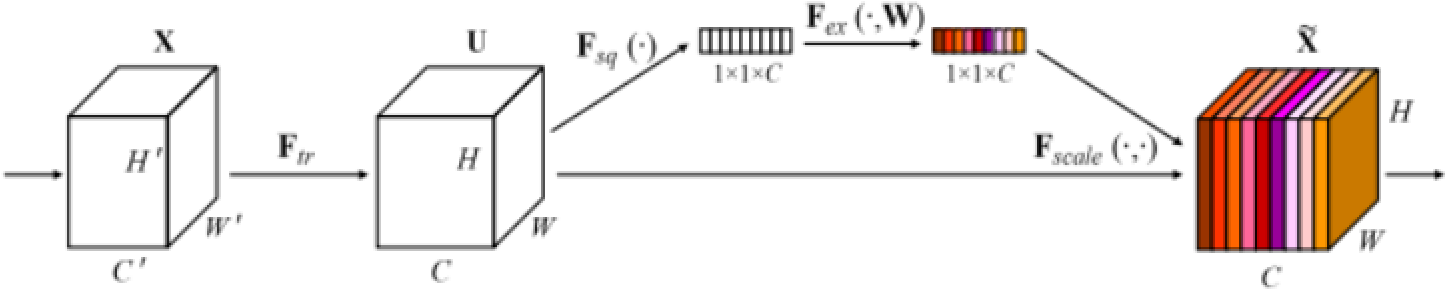
(a) Inception module, naïve version



(b) Inception module with dimensionality reduction



Squeeze and Excitation Block



1. Squeeze: Global Information Embedding

$\mathbf{F}_{sq}(\cdot)$: global average pooling

2. Excitation: Adaptive Recalibration

$$\mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1\mathbf{z}))$$

$\rightarrow$ two fc layers with ReLU

$$\tilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c\mathbf{u}_c\text{: dimensionality-increasing}$$

J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

SE-Inception Module

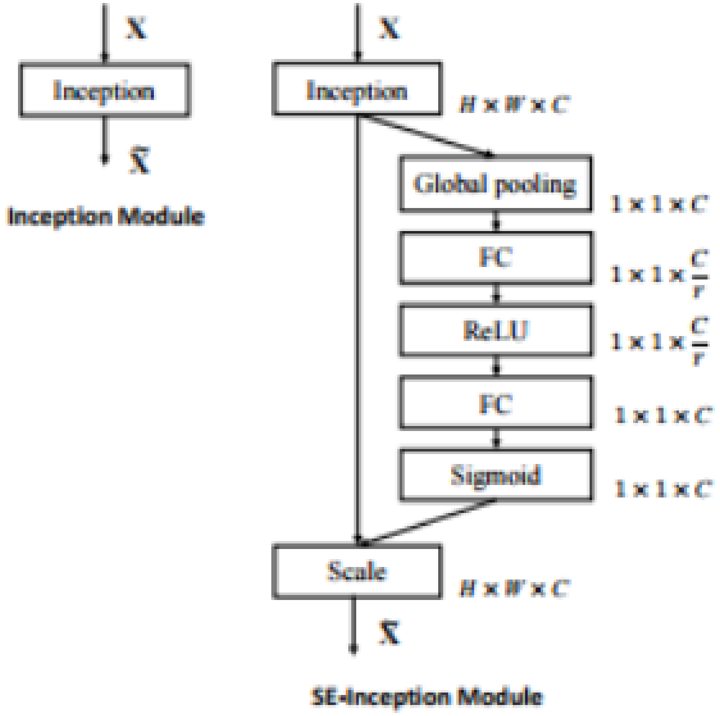


Fig. 2. The schema of the original Inception module (left) and the SE-Inception module (right).

J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Residual and SE-Residual Module

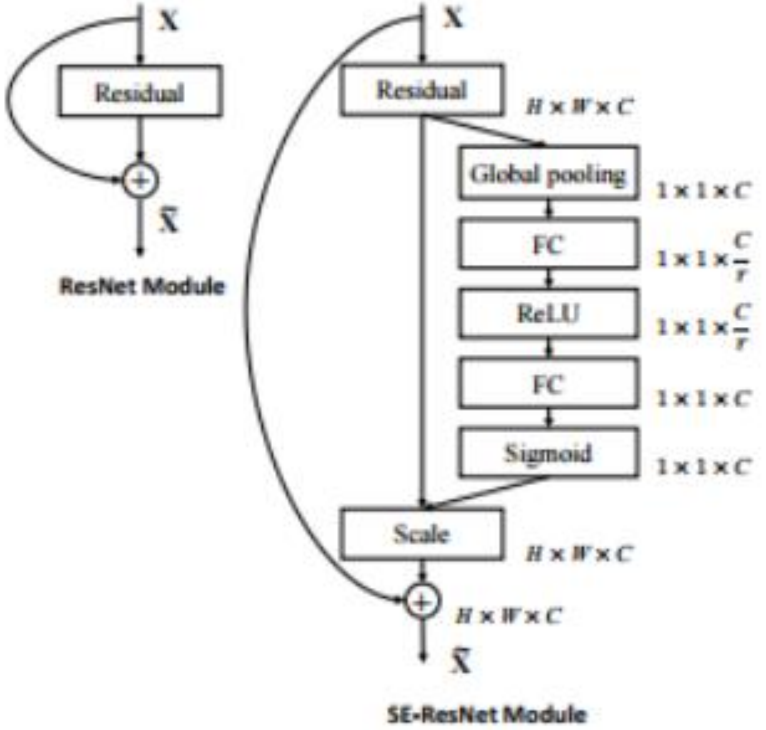


Fig. 3. The schema of the original Residual module (left) and the SE-ResNet module (right).

J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

ResNeXt deeper/wider test in Original Paper

	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

Table 3. Ablation experiments on ImageNet-1K. (**Top**): ResNet-50 with preserved complexity (~4.1 billion FLOPs); (**Bottom**): ResNet-101 with preserved complexity (~7.8 billion FLOPs). The error rate is evaluated on the single crop of 224×224 pixels.

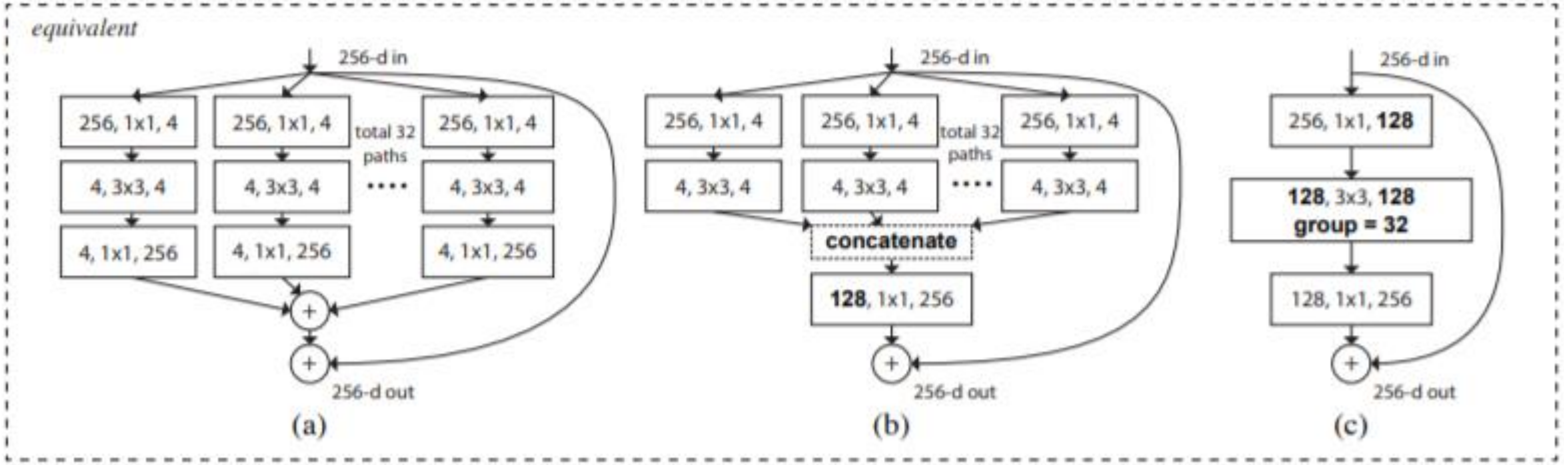


Figure 3. Equivalent building blocks of ResNeXt. (**a**): Aggregated residual transformations, the same as Fig. 1 right. (**b**): A block equivalent to (a), implemented as early concatenation. (**c**): A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold**

S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, "Aggregated Residual Transformations for Deep Neural Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 5987-5995.

ResNet50 vs ResNeXt-50

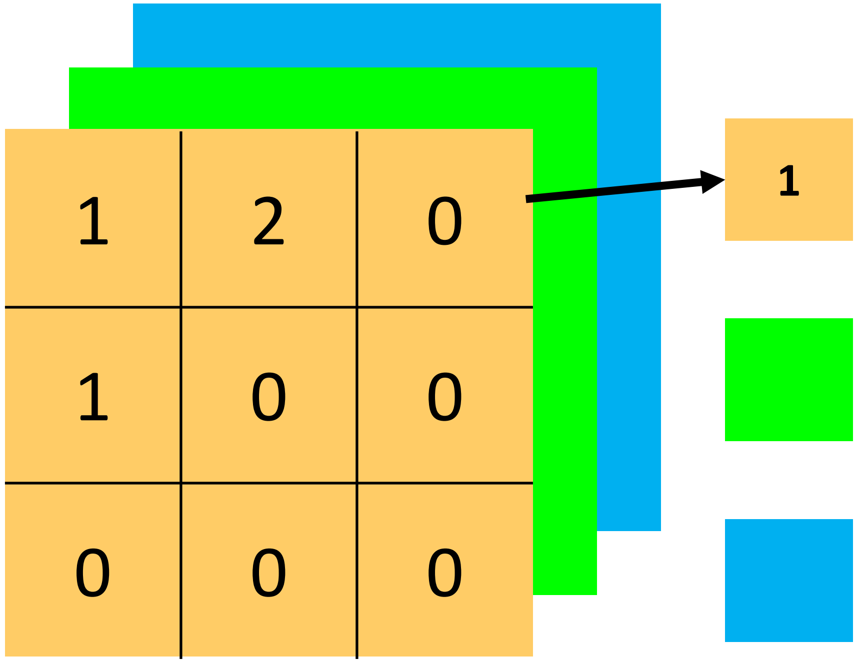
stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**$\times10^6$	**25.0**$\times10^6$
FLOPs		**4.1**$\times10^9$	**4.2**$\times10^9$

- **ResNeXt50**
- **Deeper depth for convolution**
- **But less computation**

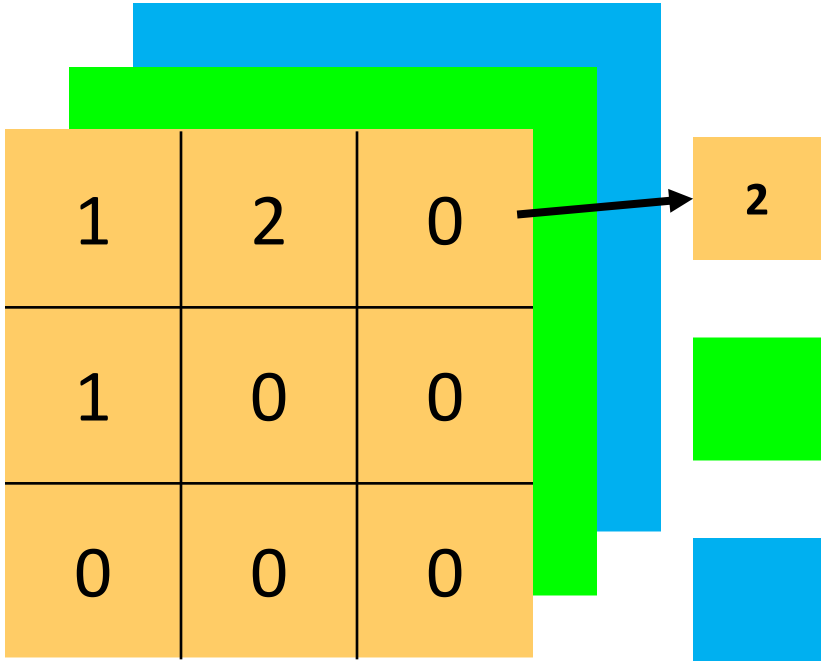
S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, "Aggregated Residual Transformations for Deep Neural Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 5987-5995.

Global Average Pooling vs Global Max Pooling

- Why Global pooling?
  - Replace fully connected layer -> overfitting and expensive computation
  - Structural regularization / confidence maps in classification
  - Others? -> Dropout, fully convolution layer



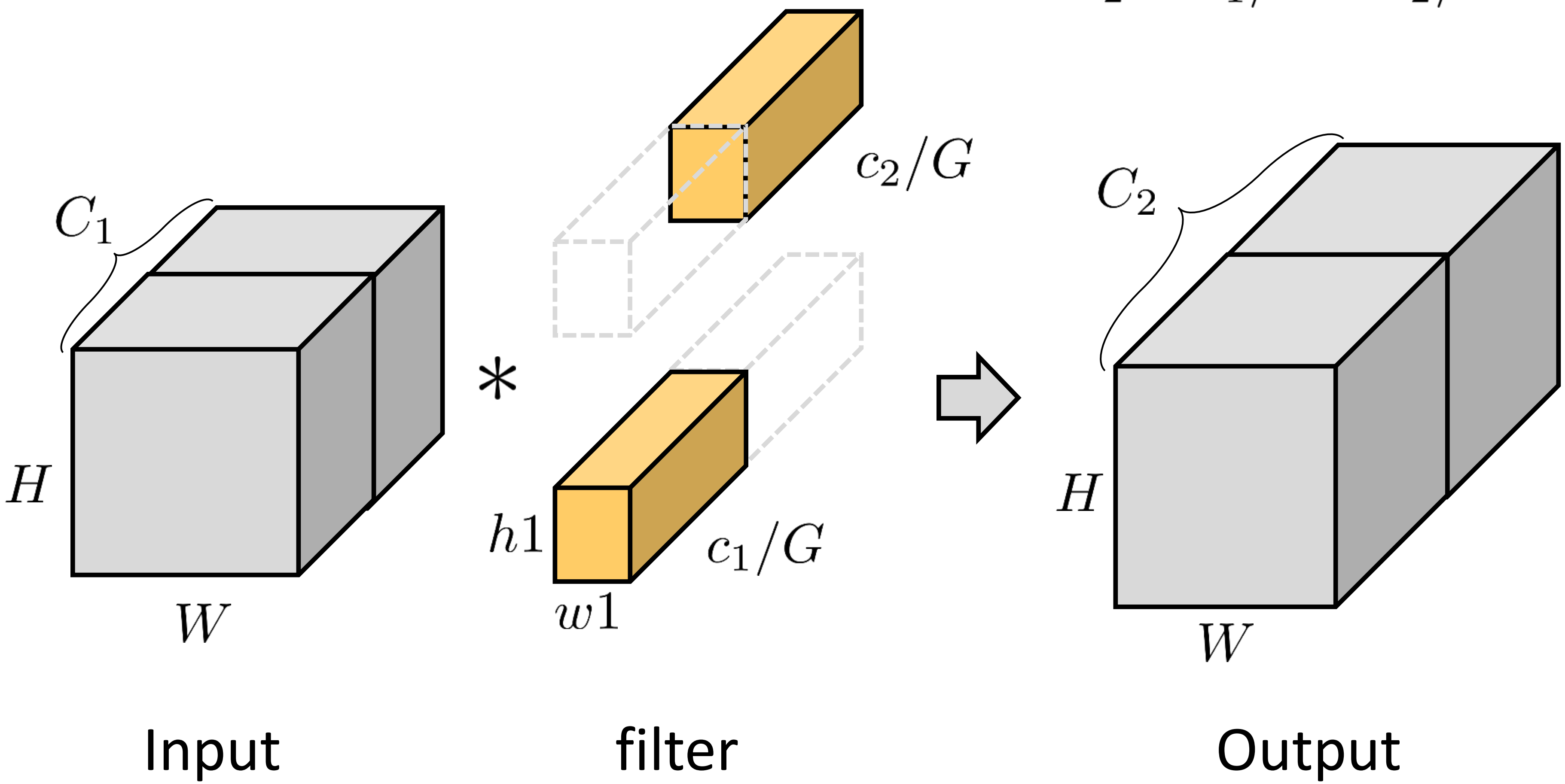
a) Activation of conv layer and global average pooling



b) Activation of conv layer and global max pooling

- G: number of group

$$C_2 = c_1/G + c_2/G$$

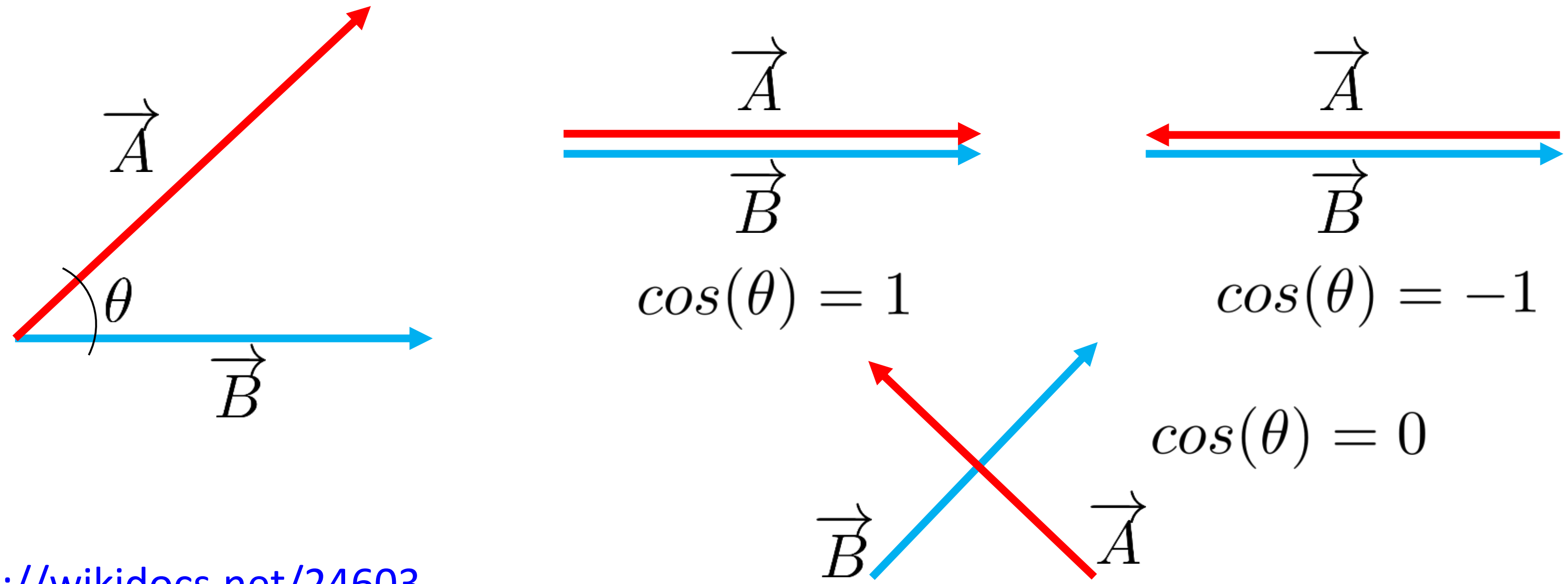


Cosine similarity(1)

- According to the direction of vector, between two vectors, compute the similarity

$$A \cdot B = \|A\|\|B\|cos(\theta)$$

$$\text{similarity} = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$



https://wikidocs.net/24603

1. A = (3,0), B=(6,0)

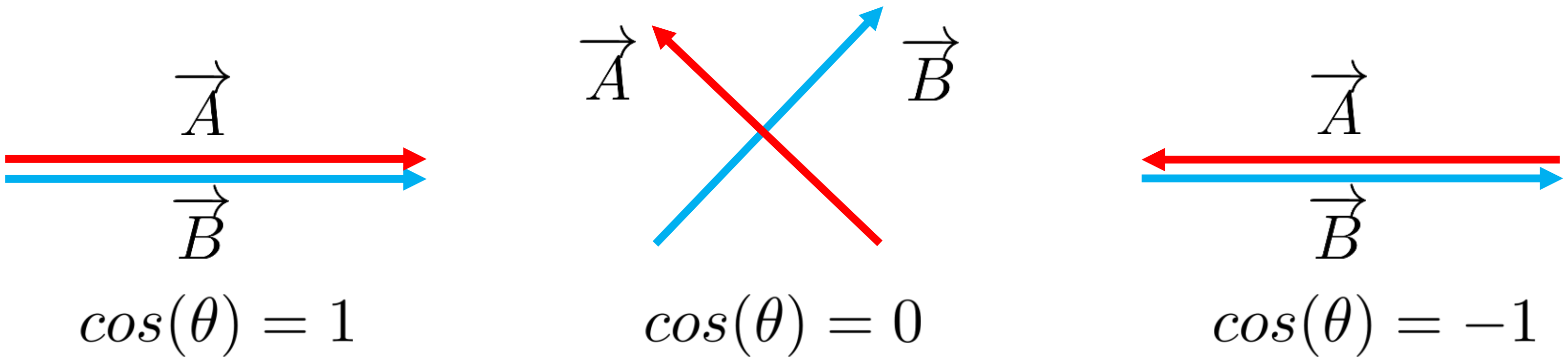
$$A \cdot B = 18, \|A\| \times \|B\| = 18 \rightarrow cos\theta = 1$$

2. A = (3,0), B=(0,6)

$$A \cdot B = 0, \|A\| \times \|B\| = 18 \rightarrow cos\theta = 0$$

3. A = (3,0), B=(-4,0)

$$A \cdot B = -12, \|A\| \times \|B\| = 12 \rightarrow cos\theta = -1$$



Cosine similarity(3)

$$cos(\theta) = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\|\overrightarrow{A}\|\|\overrightarrow{B}\|}$$

$$\|\overrightarrow{A}\| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$$

$$\|\overrightarrow{B}\| = \sqrt{b_1^2 + b_2^2 + \cdots + b_n^2}$$

Grad-CAM(1/5)

- Based on the fundamental framework of Grad-CAM[3]
  - Gradient-weighted Class Activation Mapping

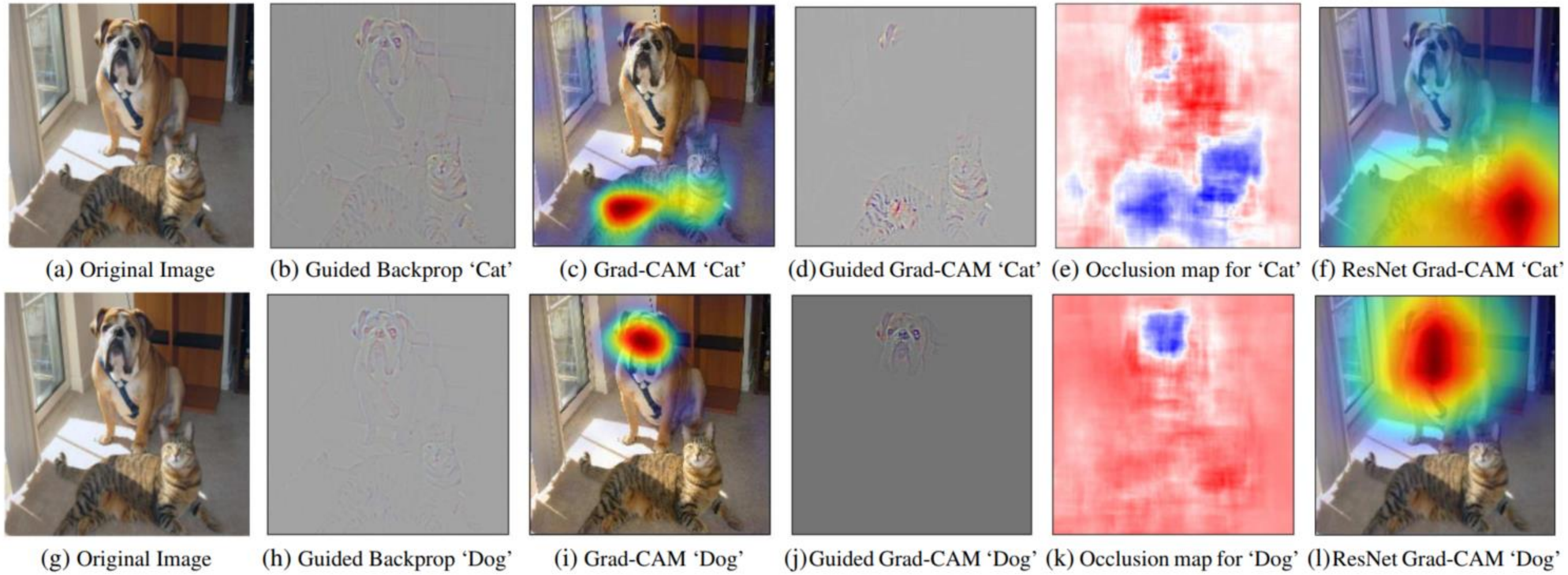
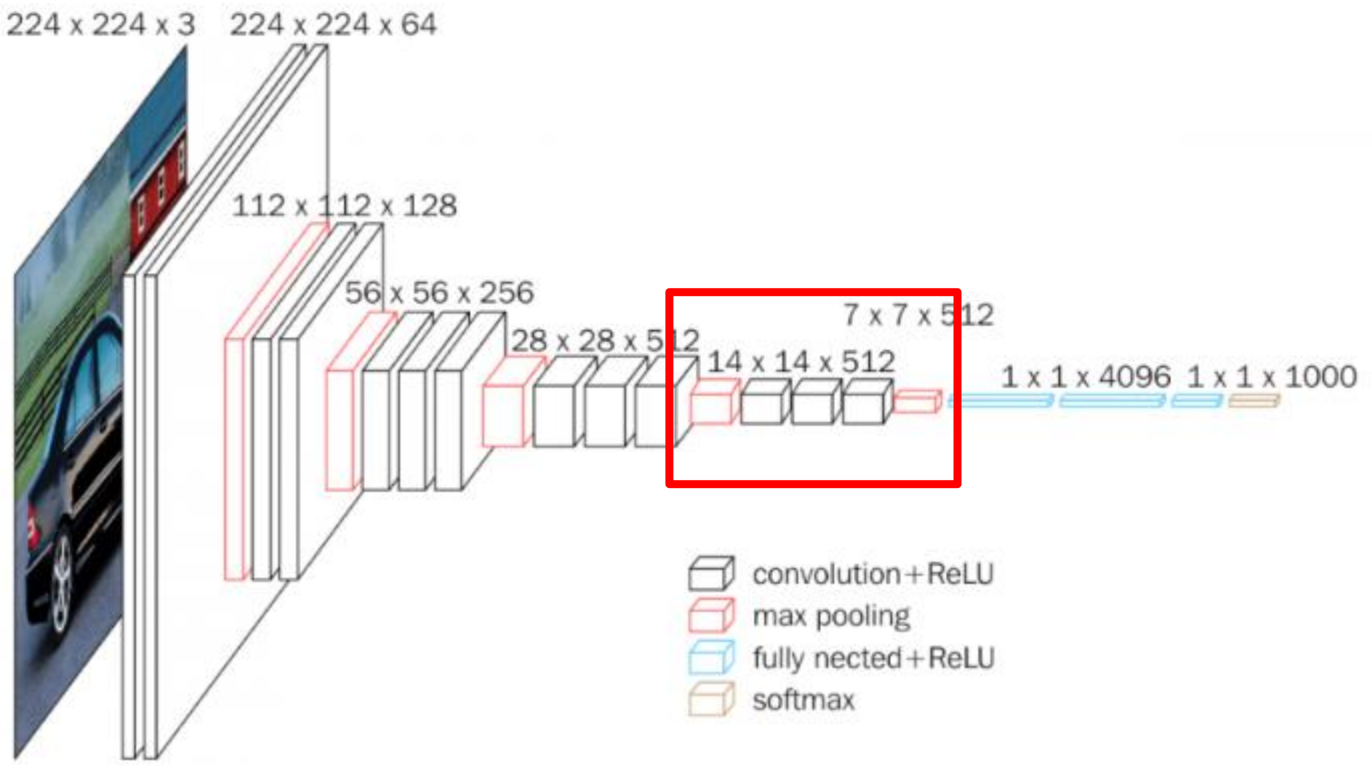


Figure 1: (a) Original image with a cat and a dog. (b-f) Support for the cat category according to various visualizations for VGG-16 and ResNet. (b) Guided Backpropagation [42]: highlights all contributing features. (c, f) Grad-CAM (Ours): localizes class-discriminative regions, (d) Combining (b) and (c) gives Guided Grad-CAM, which gives high-resolution class-discriminative visualizations. Interestingly, the localizations achieved by our Grad-CAM technique, (c) are very similar to results from occlusion sensitivity (e), while being orders of magnitude cheaper to compute. (f, l) are Grad-CAM visualizations for ResNet-18 layer. Note that in (c, f, i, l), red regions corresponds to high score for class, while in (e, k), blue corresponds to evidence for the class. Figure best viewed in color.

[3]R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017

Implementation Details

- CNN model: VGG16 and ResNet101
- Proposed model: last two convolutional layers modified as to have the stride equal to 1 instead of 2 in the original networks
- dilated conv-> rate=2, 4(to enlarge the receptive field)



- Deeper representations in a CNN capture higher-level visual construct[4,5]
- Convolution features naturally retain **spatial information which is lost in full-connected layers** so we expect the **last convolutional layers to have the best compromise between high-level semantics and detailed spatial information**
- In last layer**, look for semantic class-specific information** in the image

[4] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013
[5] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, pages 1–23, 2016

- To obtain the class discriminative localization map

$$L^c_{Grad-CAM} \in \mathbb{R}^{u \times v}$$

- Compute the gradient of the score for class c,

$$y^c\text{: before softmax, fc 1000}$$

- Respect to feature maps A^k of a convolutional layer

$$\frac{\partial y^c}{\partial A^k} = \frac{\partial y^c}{\partial fc4000_2 layer} \times \frac{\partial fc4000_2 layer}{\partial fc4000_1 layer} \times \frac{\partial fc4000_1 layer}{\partial A^k}$$

- Gradients flowing back are global-average pooling to obtain the neuron importance weights α^c_k

$$\alpha^c_k = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A^k_{ij}}}_{\text{gradients via backprop}}$$

Grad-CAM(4/5)

- Weighted combination of forward activation maps and follow it by a ReLU to obtain

$$L^c_{Grad-CAM} = ReLU\left(\sum_k \alpha^c_k A^k\right)$$

- Result in coarse heat-map of the same size as the convolutional feature maps(14x14, last convolutional layers of VGG and AlexNet)
- Why they use ReLU
  - Only interested in the features that have a positive influence on the class of interest
  - Pixels whose intensity should be increased in order to increase

y^c: fc 1000 layer

  - **Negative** pixels are likely to **belong to other categories** in the image

- Prediction Score for class c

$$S^c = \sum_k \underbrace{w_k^c}_{\text{class feature weights}} \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{A_{ij}^k}_{\text{feature map}}$$

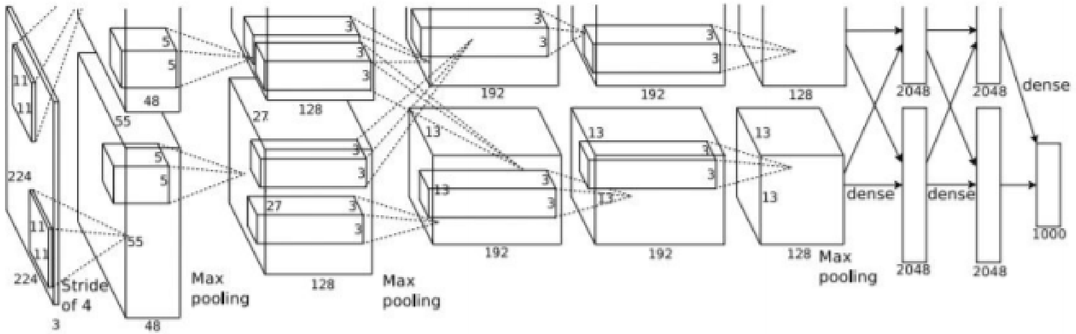
- Able to interchange the prediction score

$$\alpha_k^c = w_k^c$$

$$S^c = \frac{1}{Z}\sum_i \sum_j \underbrace{\sum_k w_k^c A_{ij}^k}_{L_{\text{CAM}}^c}$$

Case Study: AlexNet

[Krizhevsky et al. 2012]



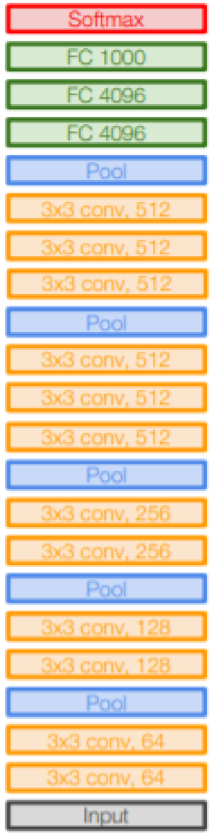
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

VGG16 Architecture

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



VGG16

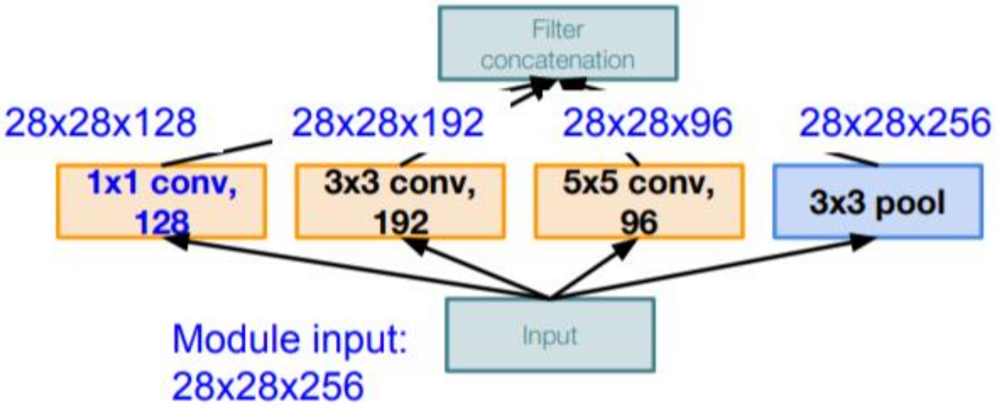
Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3:What is output size after filter concatenation?

28x28x(128+192+96+256) = 28x28x672



Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

Conv Ops:

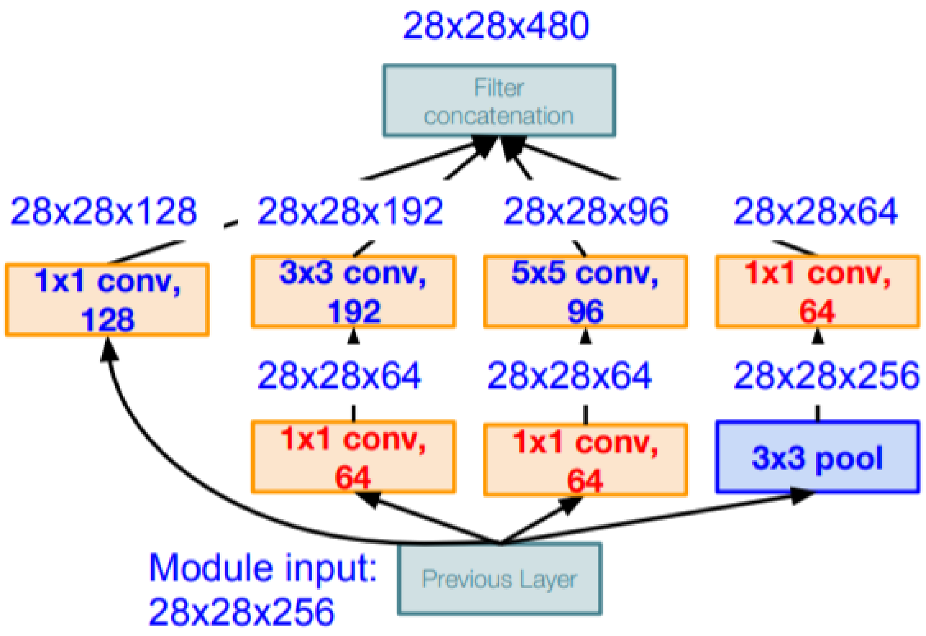
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Very expensive compute

Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

Case Study: GoogLeNet

[Szegedy et al., 2014]



Inception module with dimension reduction

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

Conv Ops:

[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops

Compared to 854M ops for naive version
Bottleneck can also reduce depth after pooling layer

Interaction with feature and attention mask

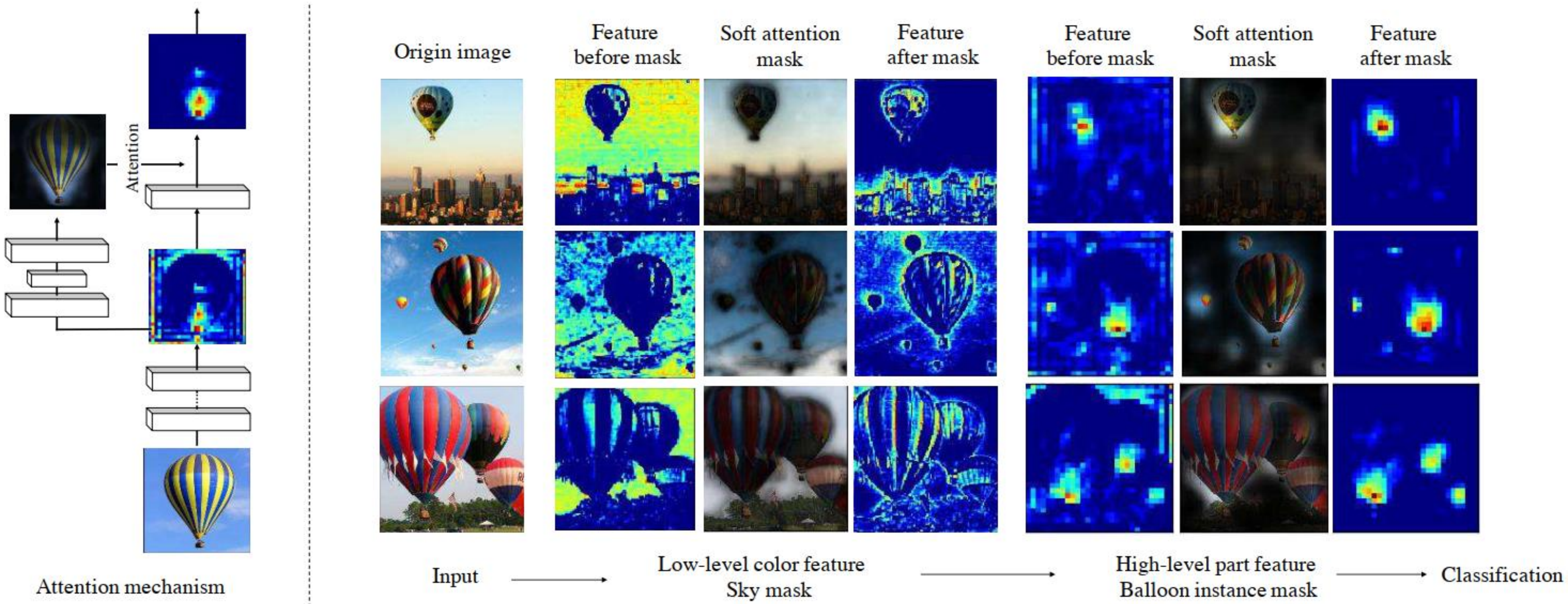


Figure 1: **Left:** an example shows the interaction between features and attention masks. **Right:** example images illustrating that different features have different corresponding attention masks in our network. The sky mask diminishes low-level background blue color features. The balloon instance mask highlights high-level balloon bottom part features.

Network Architecture

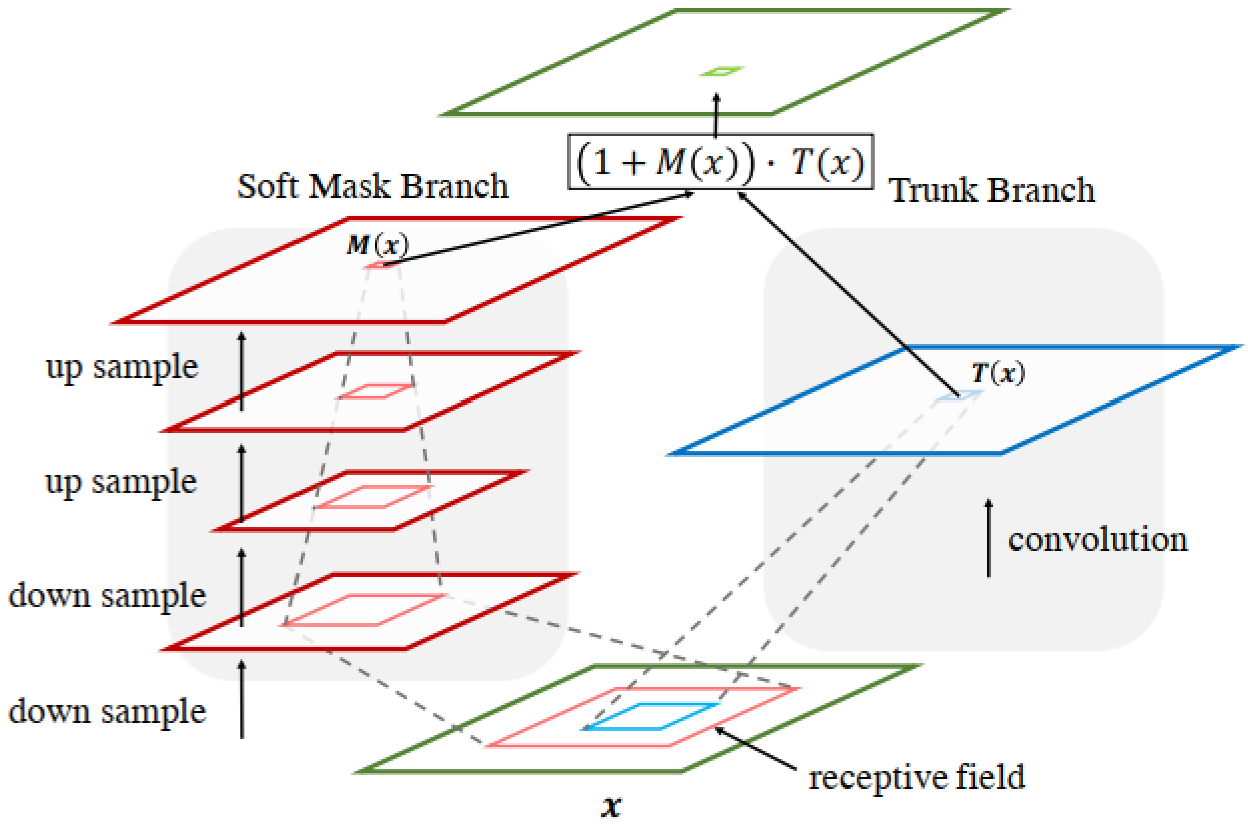
- Stacking multiple attention modules
- Two parts in attention module
  - Trunk branch, T(x)
  - Mask branch, M(x)
- Attention module

$$H_{i,c}(x) = M_{i,c}(x) * T_{i,c}(x)$$

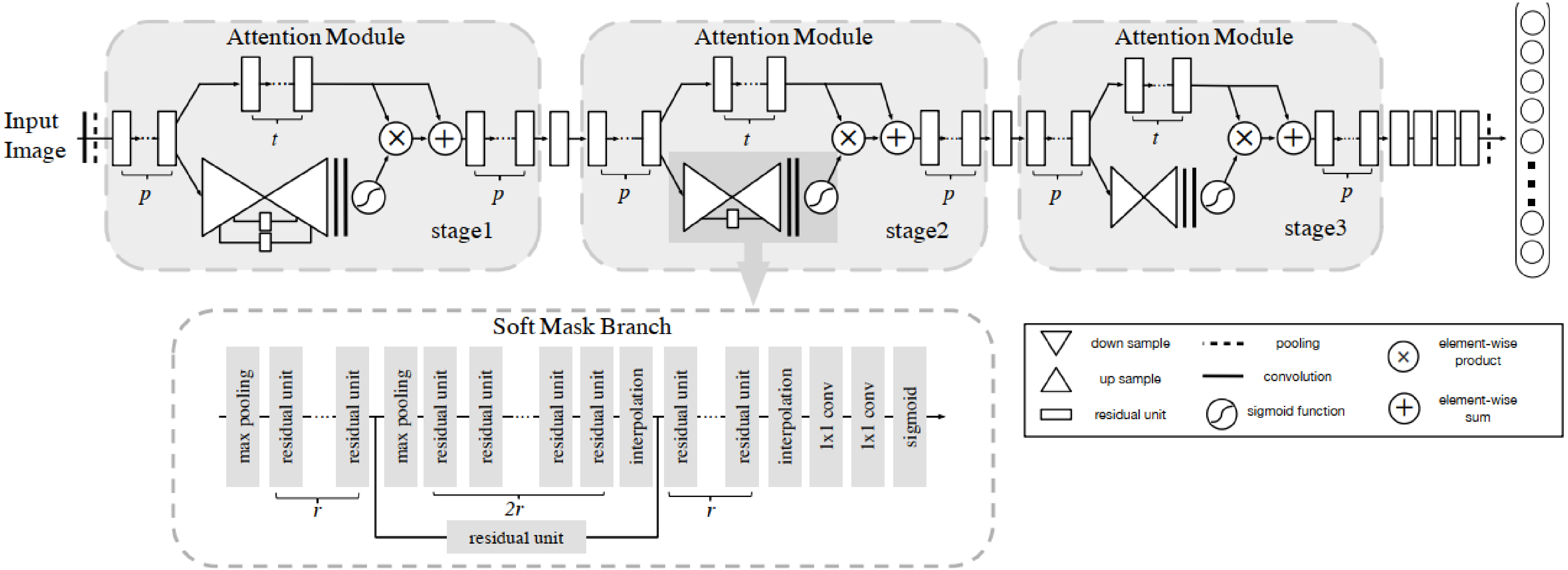
  - i: all spatial position, c: channel

Process of Soft Mask(Attention) & Trunk Branch

- Trunk: keep convolution same size with input
- Soft mask: generate mask information in top-down and bottom-up process



Proposed Architecture



p: The number of pre-processing Residual Unit before splitting into trunk and mask branch

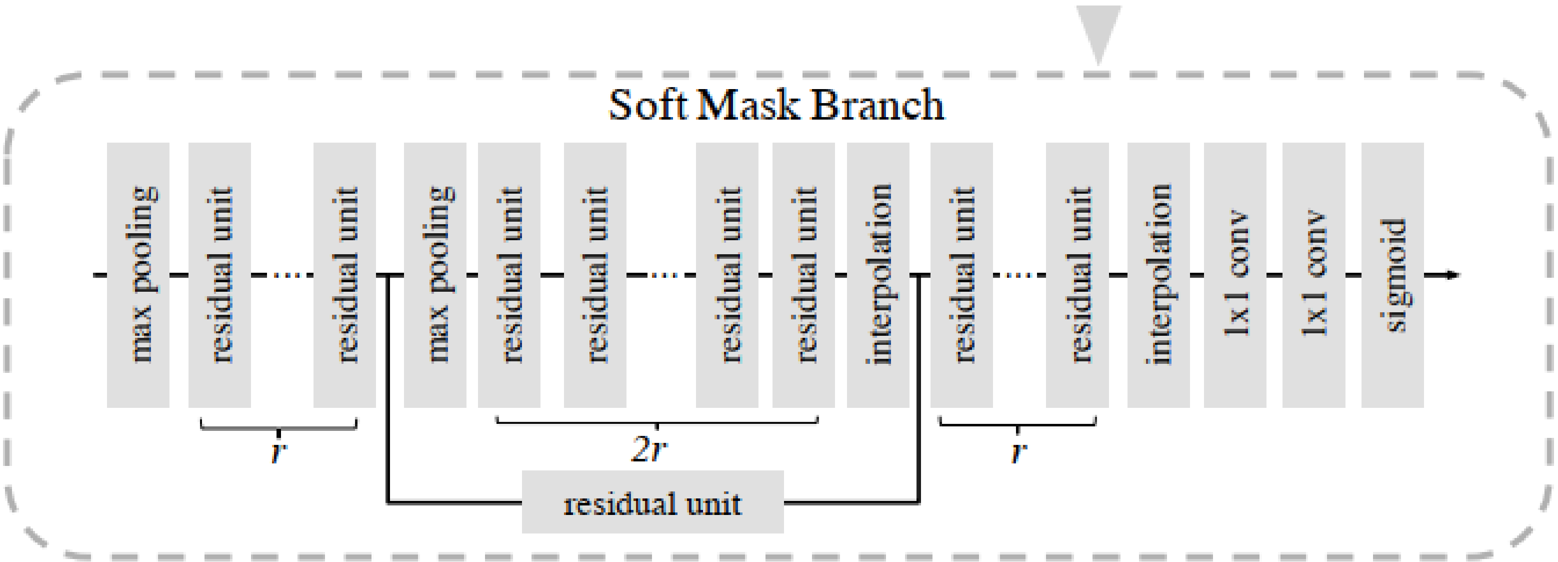
t: The number of Residual Units in trunk branch

r: The number of Residual Units between adjacent pooling layer in the mask branch

$p = 1, t = 2, r = 1$

Between stage, there is residual unit with stride, 2

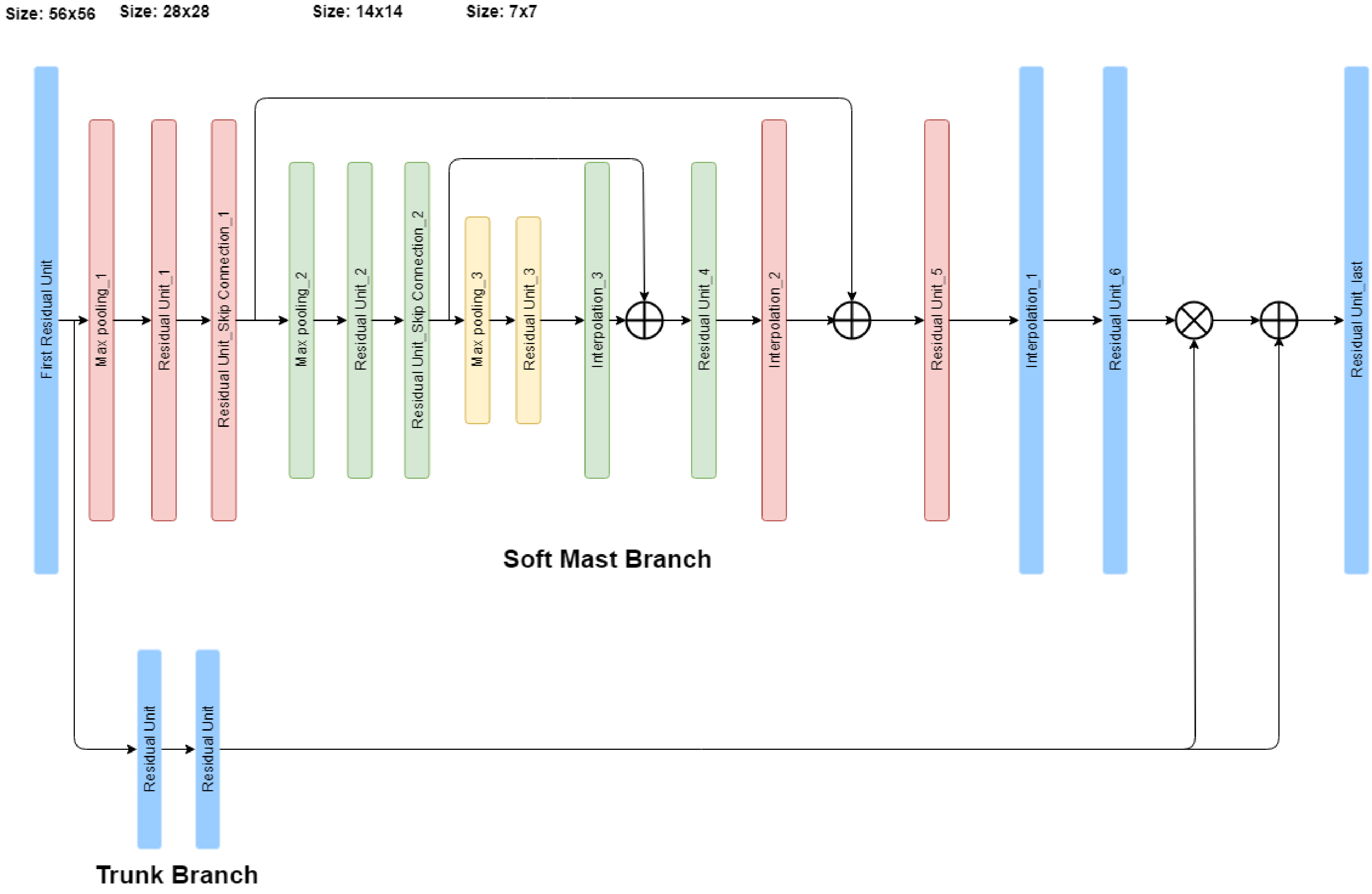
Soft Mask Branch



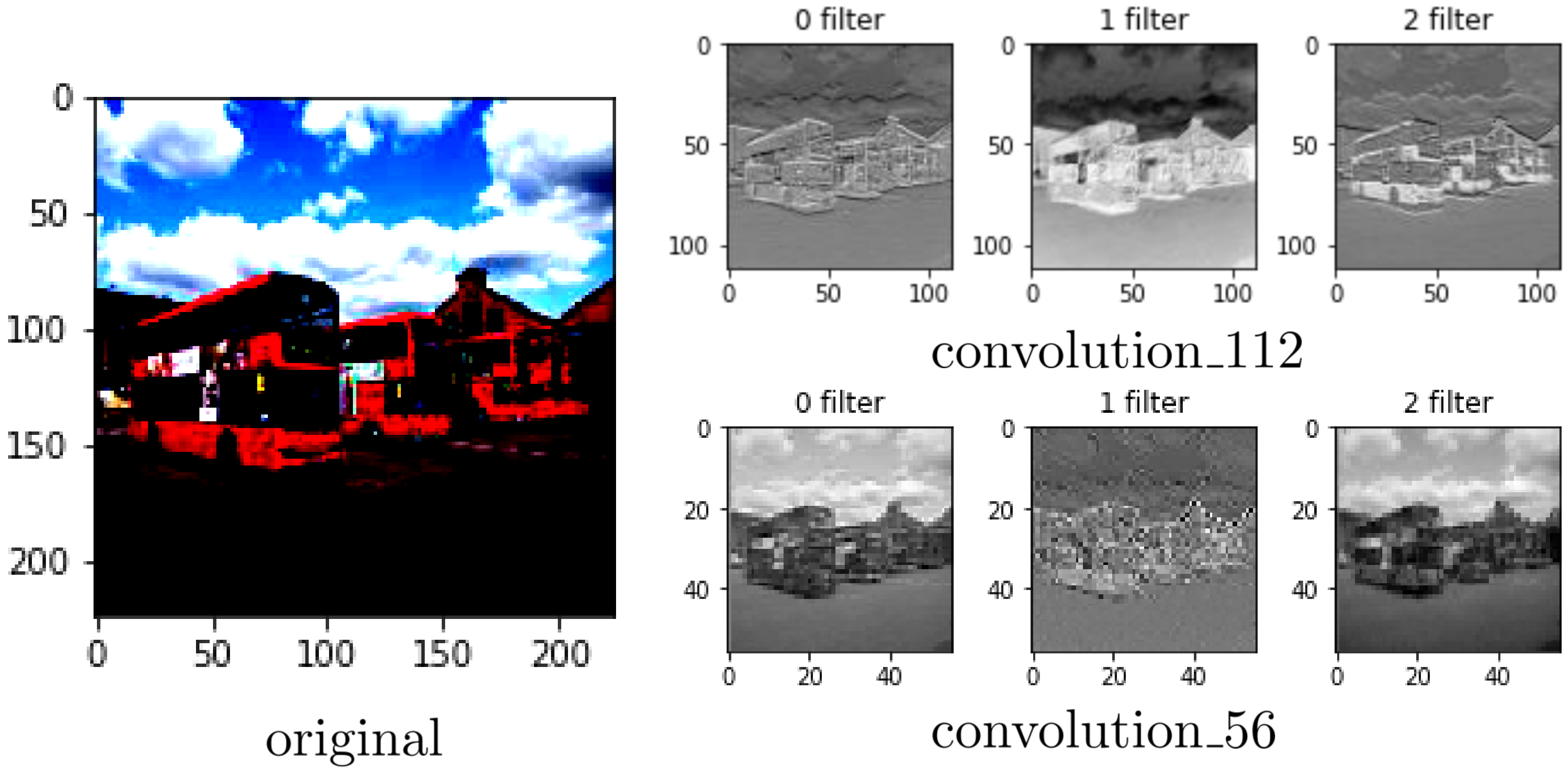
- Improve the trunk branch features with mask branch

interpolation: upsample&bilinear interpolation

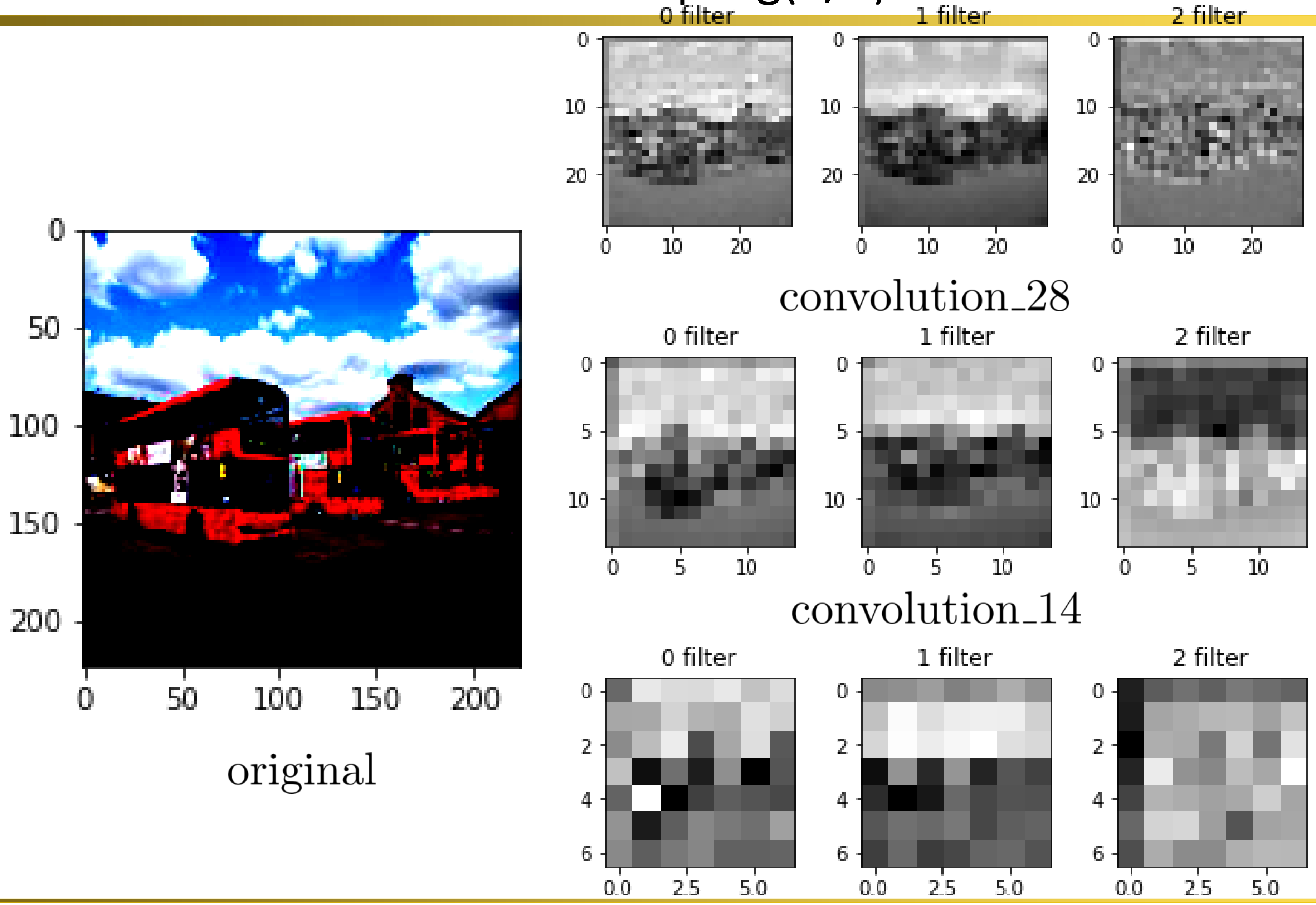
Detail of Soft Mask & Trunk Branch(stage1)



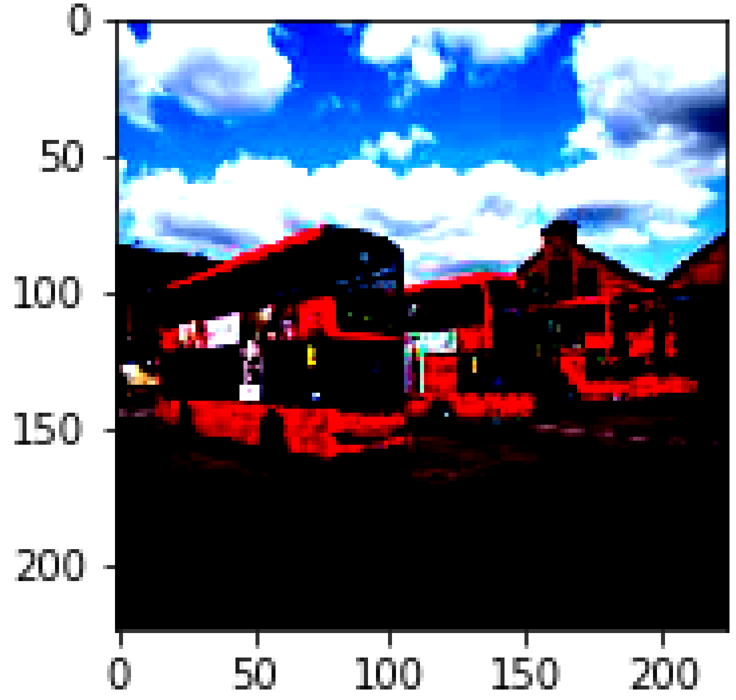
Downsampling(1/2)



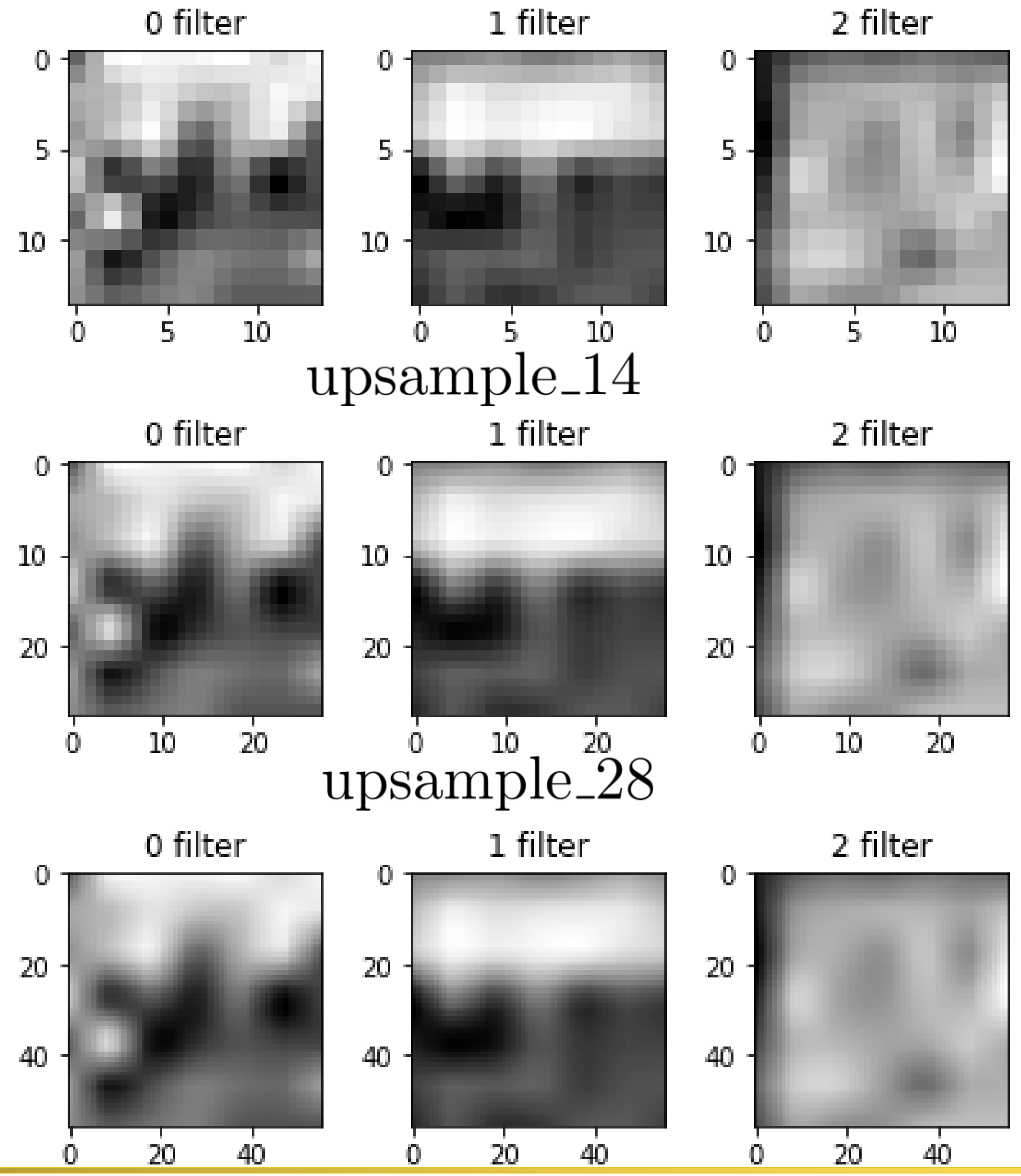
Downsampling(2/2)



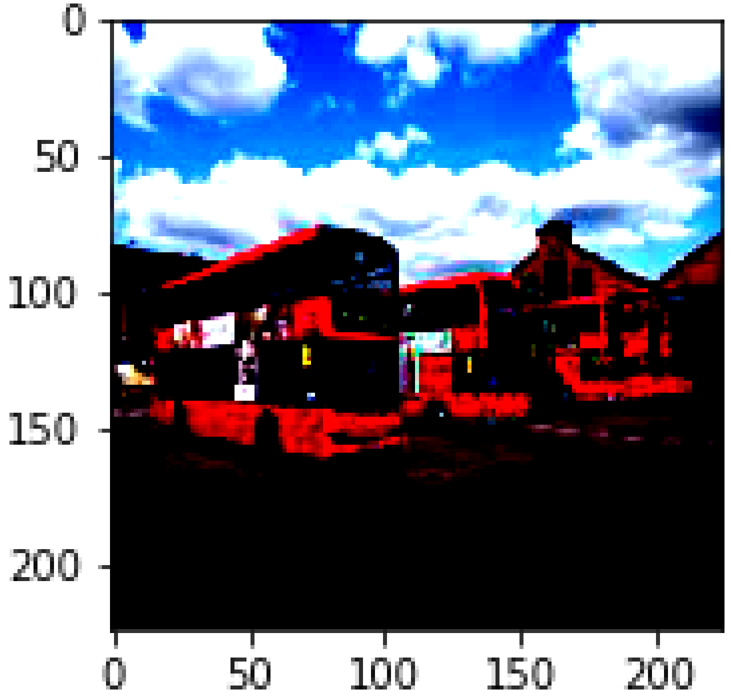
Upsampling(1/2)



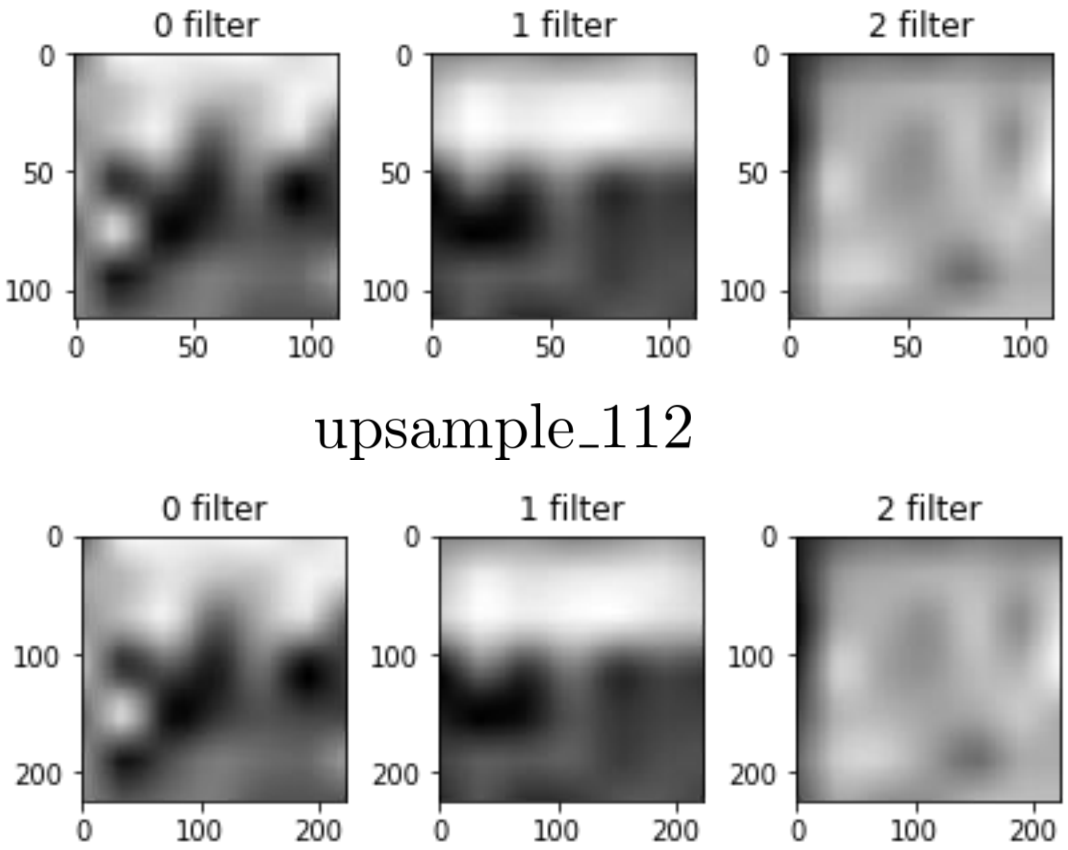
original



Upsampling(2/2)



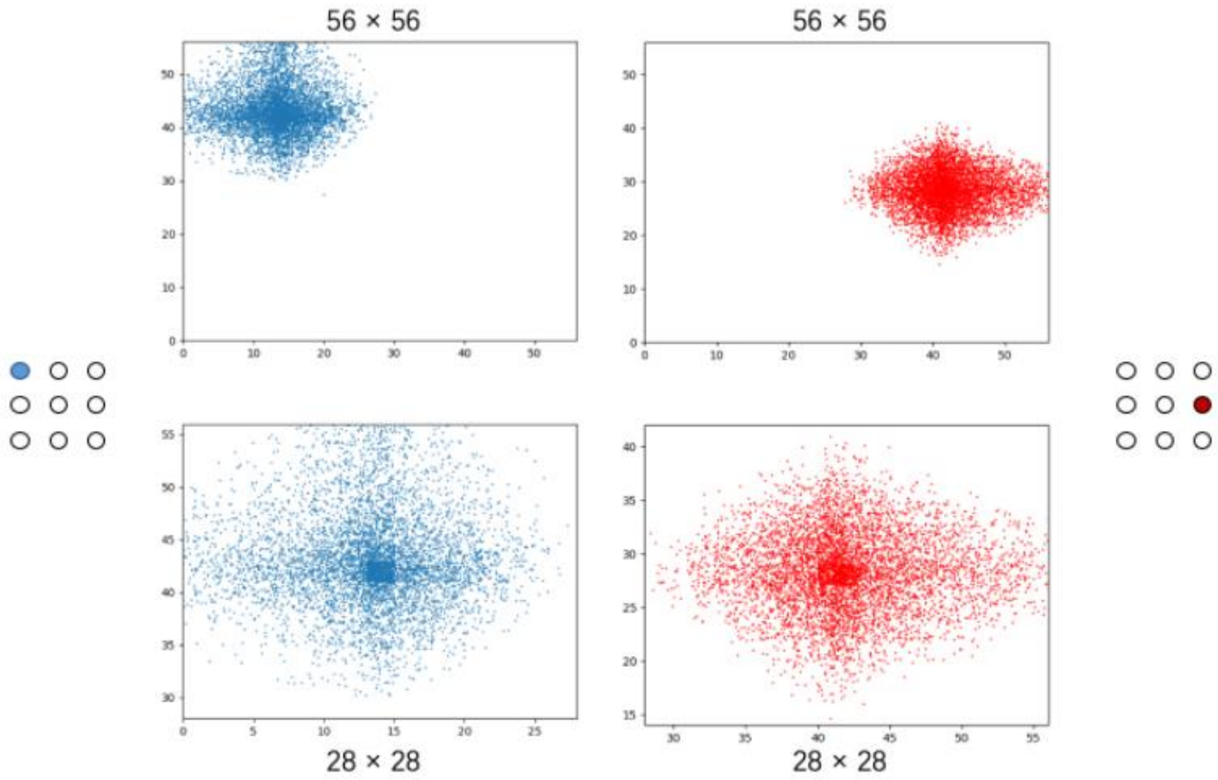
original



upsample_112

upsample_224

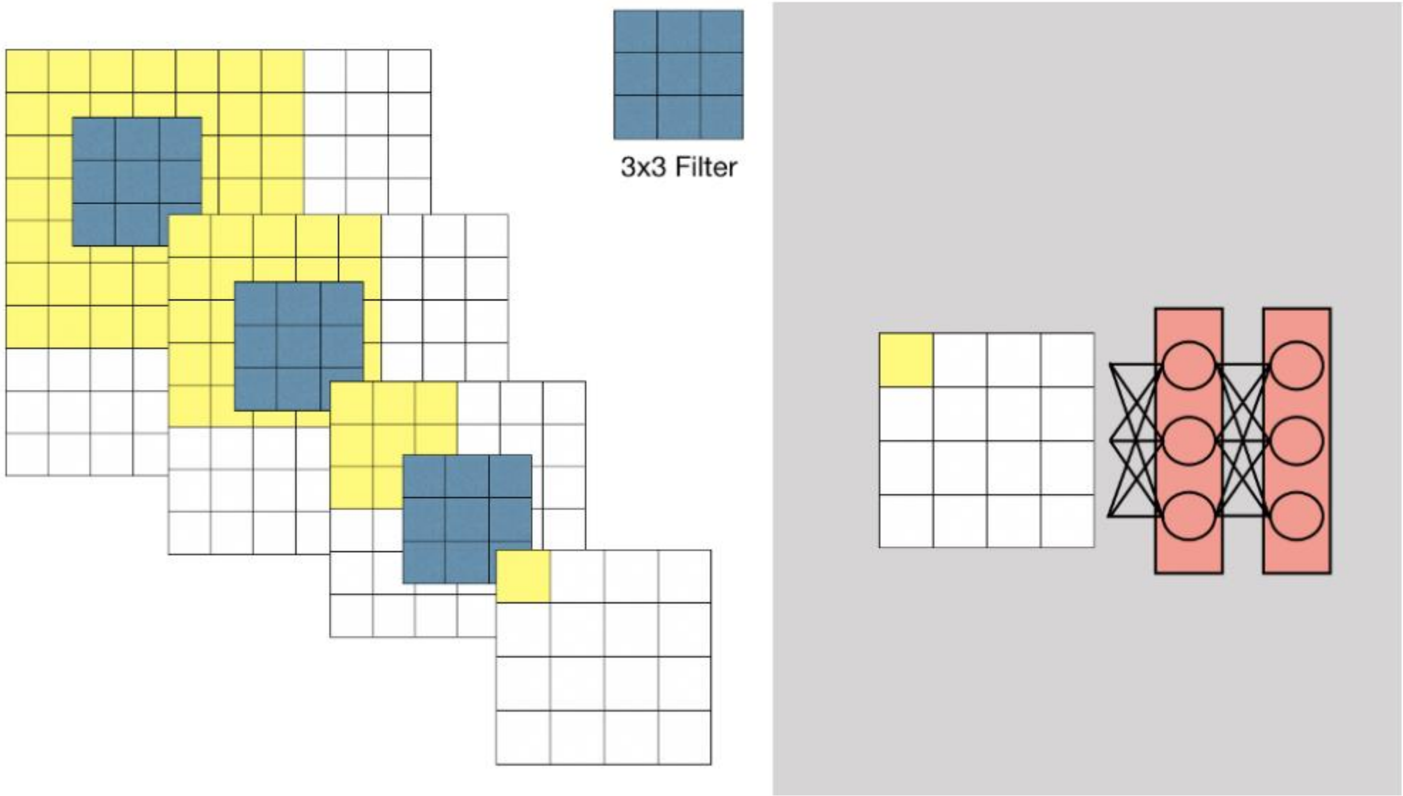
3 Grid R-CNN Plus



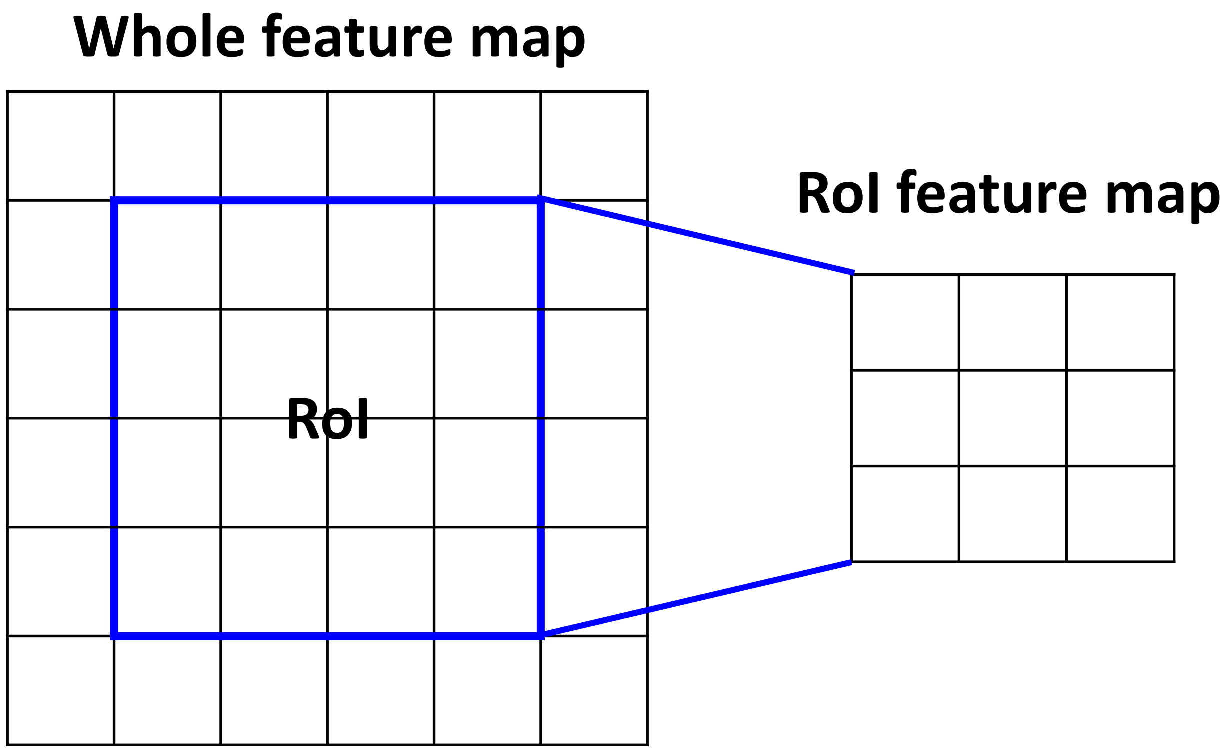
https://www.groundai.com/project/grid-r-cnn-plus-faster-and-better/1

Fully Convolutional Network vc Fully Connected Layer

- FC layer lost the location information



https://medium.com/@msmapark2/fcn-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-fully-convolutional-networks-for-semantic-segmentation-81f016d76204



- RoIPool: quantized bins + pooling
- RoIAlign: continuous + bilinear interpolation + pooling
  -> better preserved spatial correspondence

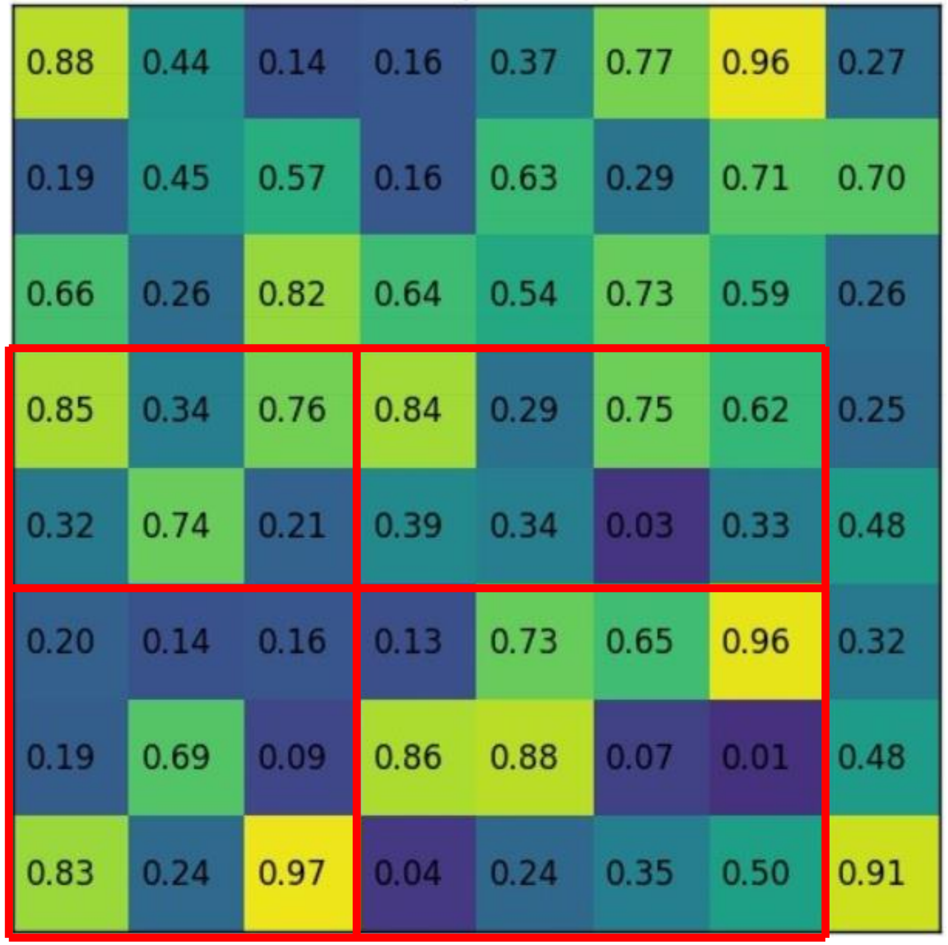
- RoIPool: quantized bins + pooling

0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

a) Input activation

Source: https://blog.deepsense.ai/region-of-interest-pooling-explained/

- RoIPool: quantized bins + pooling

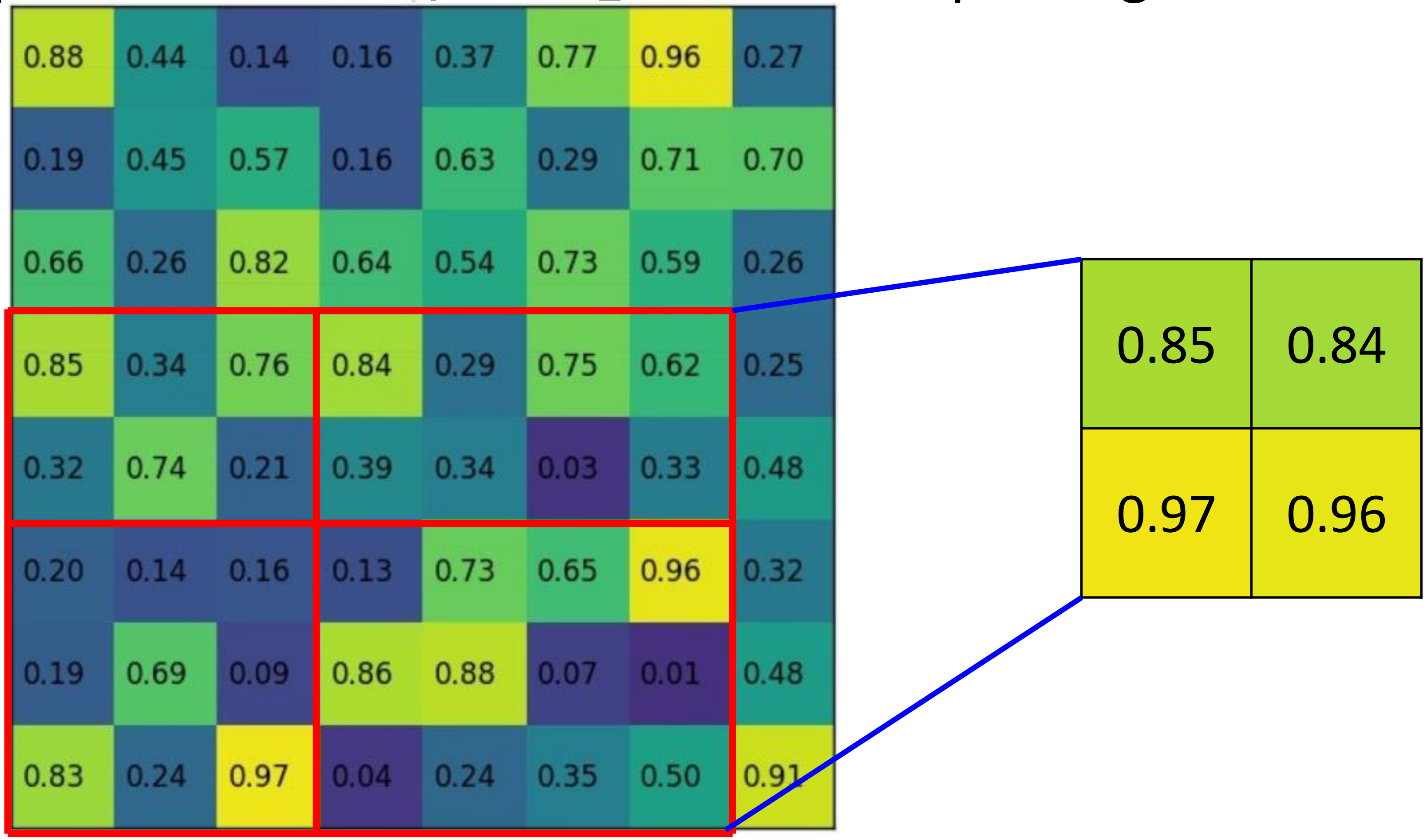


Region projection and pooling sections

Source: https://blog.deepsense.ai/region-of-interest-pooling-explained/

RoIPool

- RoIPool: quantized bins + pooling -> 2x2 max pooling

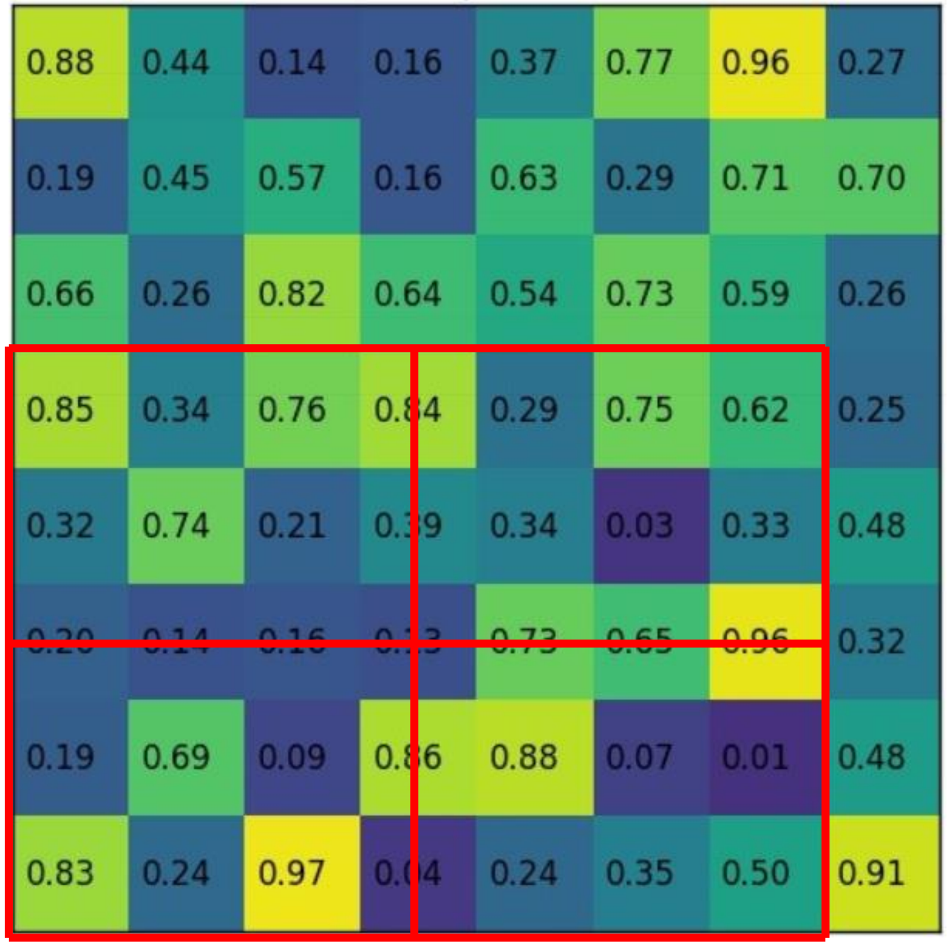


Region projection and pooling sections

Source: https://blog.deepsense.ai/region-of-interest-pooling-explained/

RoIAlign

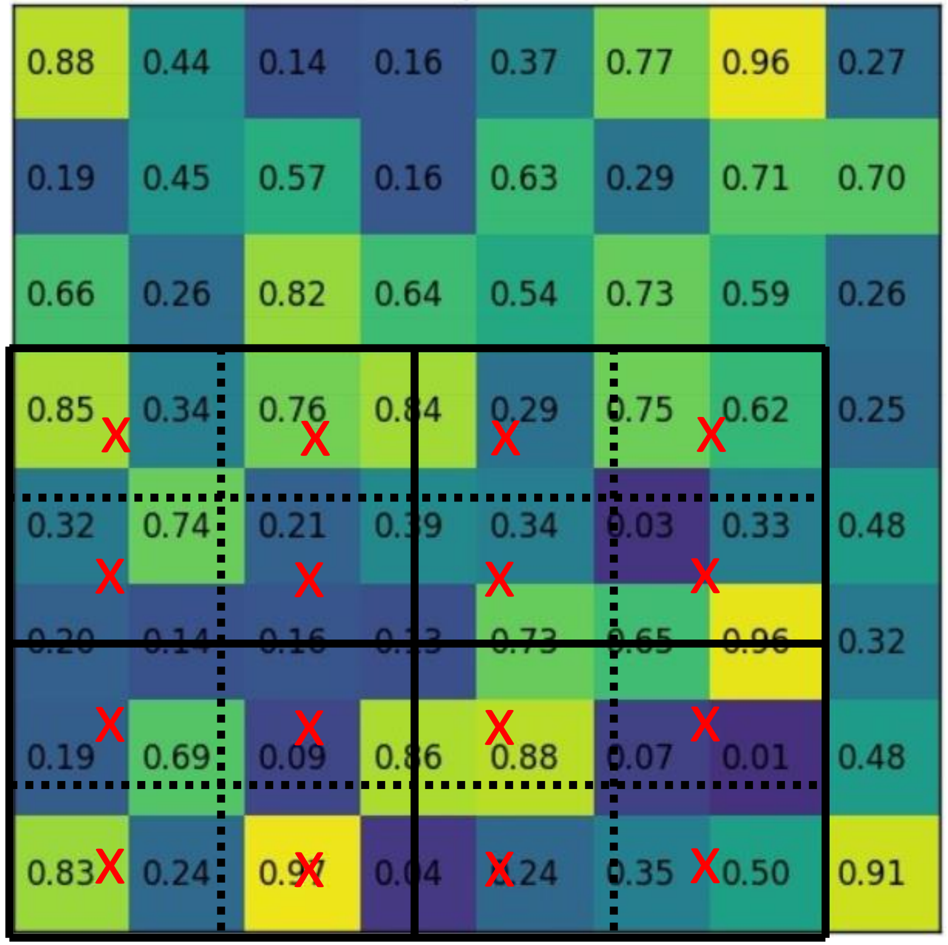
- RoIAlign: continuous + bilinear interpolation + max pooling



Region projection and pooling sections

Source: https://blog.deepsense.ai/region-of-interest-pooling-explained/

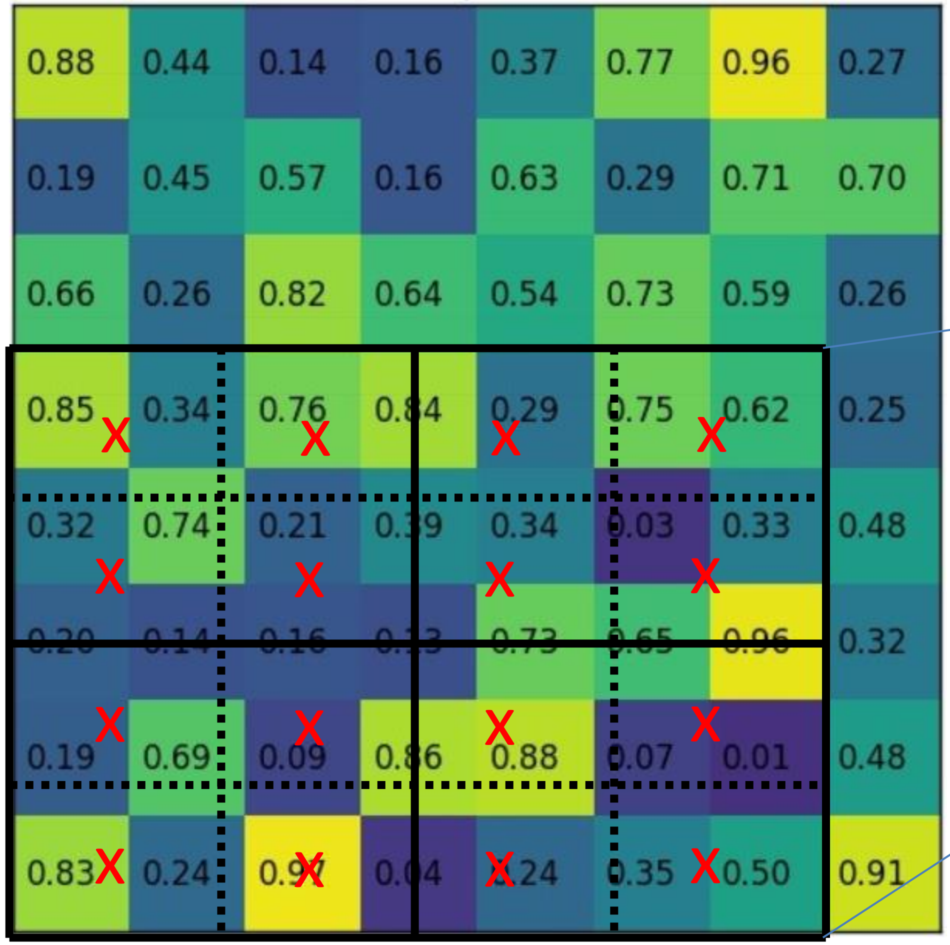
- RoIAlign: continuous + bilinear interpolation + max pooling



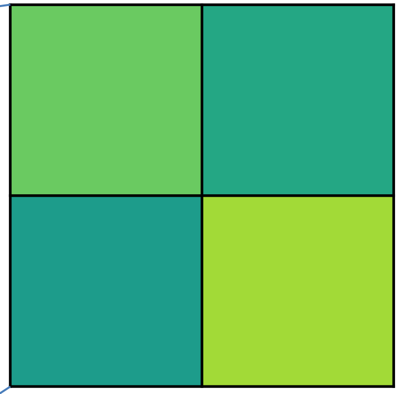
Sampling locations

Source: https://blog.deepsense.ai/region-of-interest-pooling-explained/

- RoIAlign: continuous + bilinear interpolation + max pooling



Sampling locations

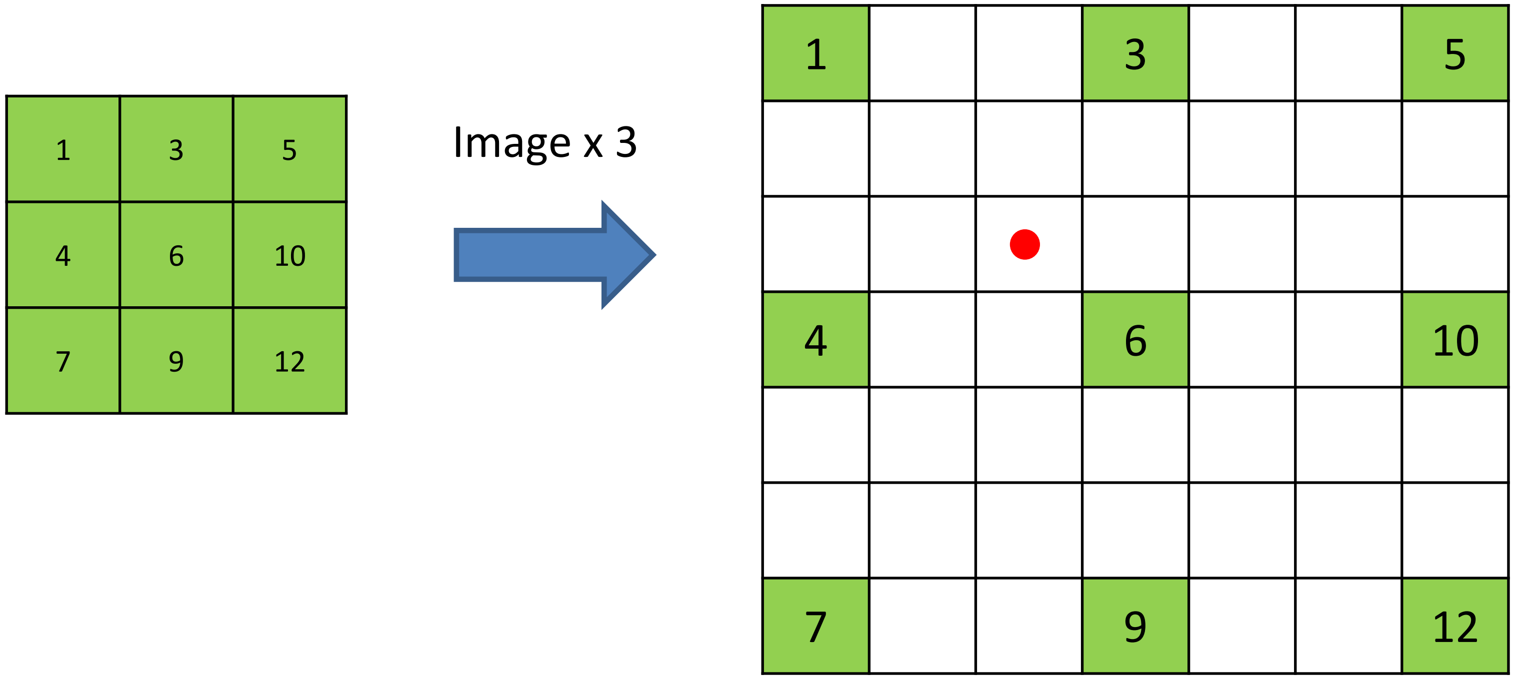


Max pooling output

Source: https://blog.deepsense.ai/region-of-interest-pooling-explained/

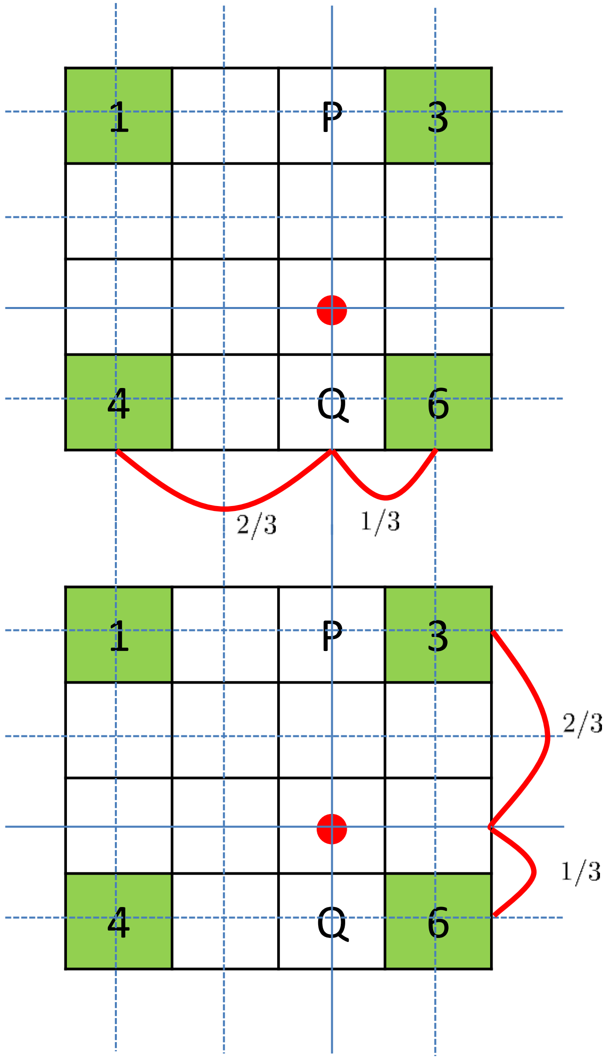
Bilinear Interpolation

- Extension of linear interpolation on rectilinear 2D grid
- As the intensity value is in linear relationship



- What is the red point intensity value?
  - Bilinear interpolation can get the intensity

- Extension of linear interpolation on rectilinear 2D grid

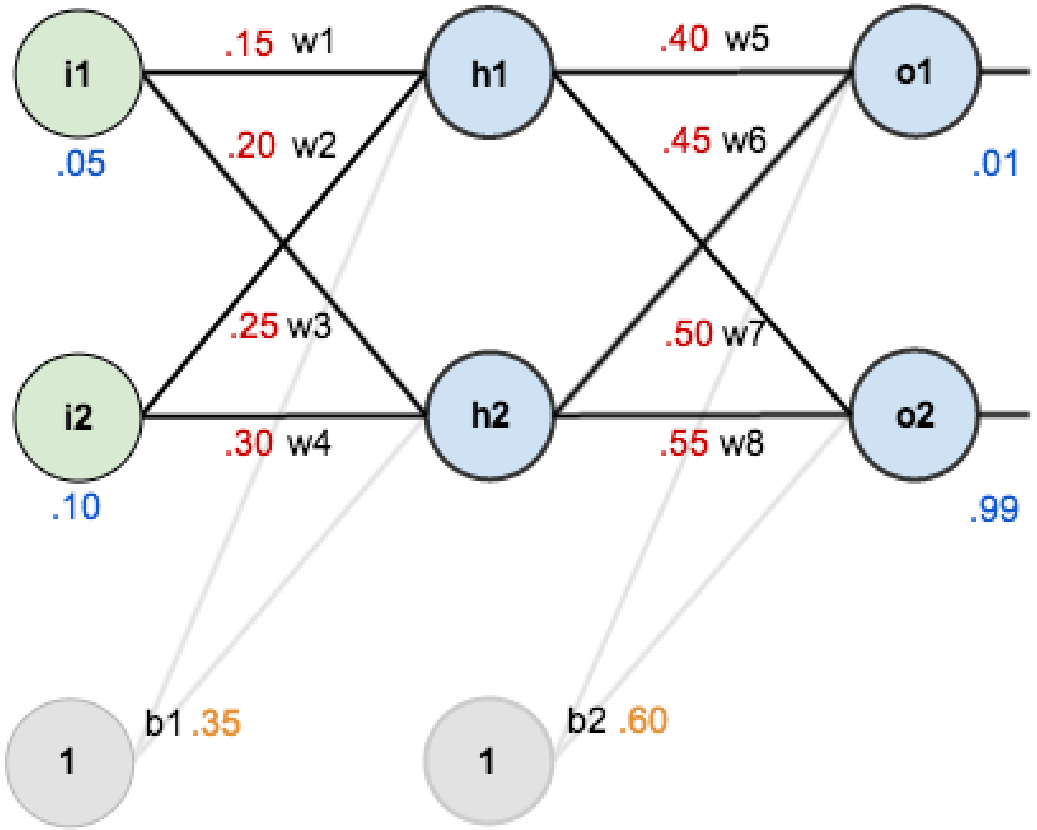


- In case of x axis
  - P value
    A=1, B=3
    P =(1/3) x A + (2/3) x B = 2.33
  - Q value
    C=4, D=6
    Q = (1/3) x 4 + (2/3) x 6 = 5.33
- In case of y axis
  - P = 2.33, Q = 5.33
    red = (1/3) x P + (2/3) x Q = 4.33

Derive for Back Propagation

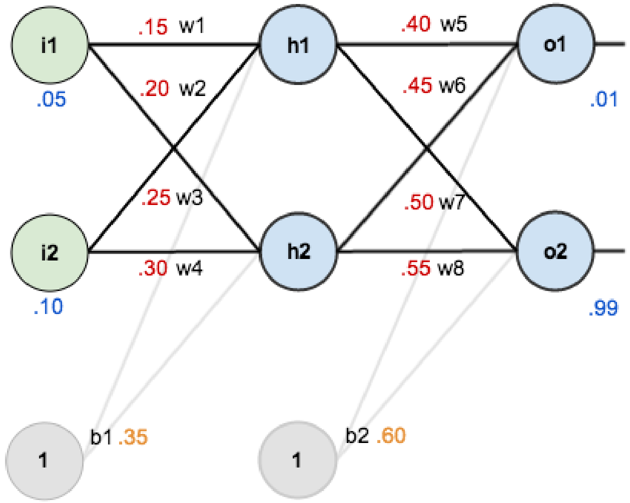
Youlkyeong Lee

Structure



- Training Input/output
- Initial weights
- biases

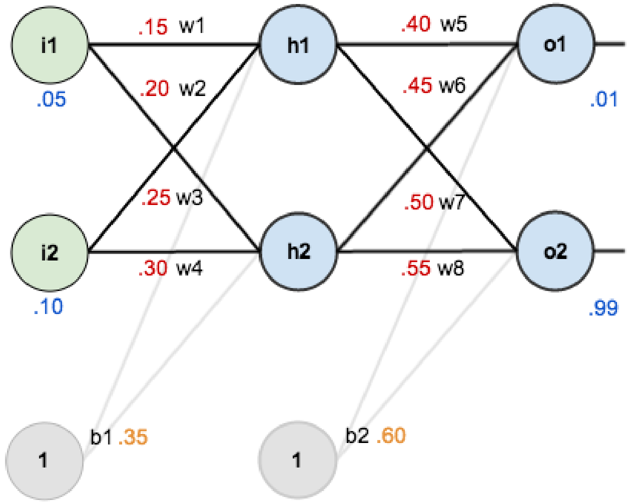
Feed Forward Computation(1/3)



$$\mathrm{net}_{h1} = w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1 \quad \mathrm{net}_{h1} = 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 \times 1 = 0.3775$$

$$\text{logistic function} \quad \mathrm{out}_{h1} = \frac{1}{1 + exp^{-\mathrm{net}_{h1}}} = \frac{1}{1 + exp^{-0.3775}} = 0.593269992$$

$$\mathrm{out}_{h2} = 0.596884378$$



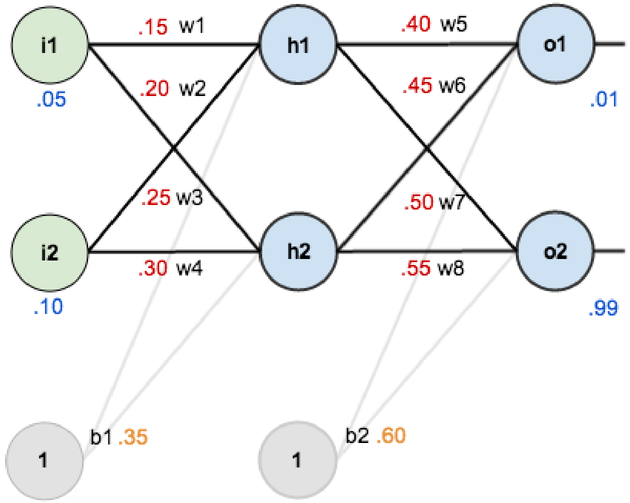
$$\text{net}_{o1} = w_5 \times \text{out}_{h1} + w_6 \times \text{out}_{h2} + b_2 \times 1$$

$$\text{net}_{o1} = 0.4 \times 0.593269992 + 0.45 \times 0.596884378 + 0.6 \times 1 = 1.105905967$$

$$\text{logistic function} \quad \text{out}_{o1} = \frac{1}{1 + exp^{-\text{net}_{o1}}} = \frac{1}{1 + exp^{-1.105905967}} = 0.75146507$$

$$\text{out}_{o2} = 0.772928465$$

- Calculating the t



$$E_{total} = \sum \frac{1}{2}(\text{target} - \text{output})^2$$

$$E_{o1} = \sum \frac{1}{2}(\text{target} - \text{output})^2 = \frac{1}{2}(0.01 - 0.75146507)^2 = 0.274811083$$

$$E_{o2} = 0.023560026$$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

Backwards Computation(1/3)

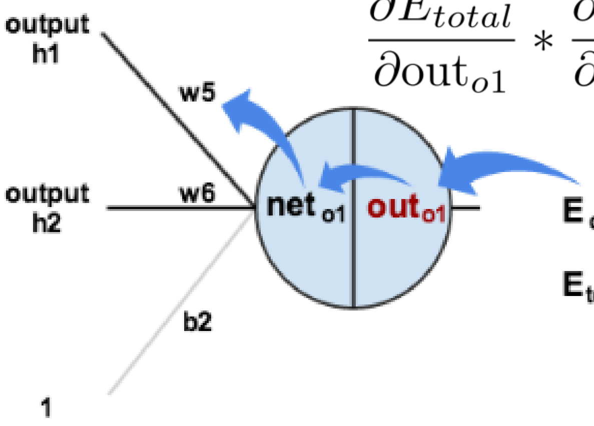
- Consider w_5, how much a change in w_5 affects the total error

$$\rightarrow \frac{\partial E_{total}}{\partial w_5}$$

- In chain rule

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial \mathrm{out}_{o1}} * \frac{\partial \mathrm{out}_{o1}}{\partial \mathrm{net}_{o1}} * \frac{\partial \mathrm{net}_{o1}}{\partial w_5}$$

Backwards Computation(1/5)



$$\frac{\partial E_{total}}{\partial \mathrm{out}_{o1}} * \frac{\partial \mathrm{out}_{o1}}{\partial \mathrm{net}_{o1}} * \frac{\partial \mathrm{net}_{o1}}{\partial w_5} = \frac{\partial E_{total}}{\partial w_5}$$

$E_{o1} = \frac{1}{2}(\mathrm{target}_{o1} - \mathrm{out}_{o1})^2$

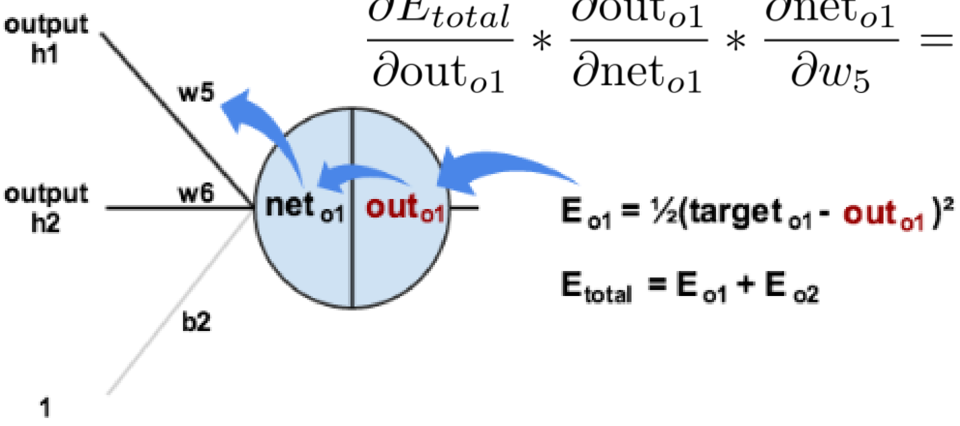
$E_{total} = E_{o1} + E_{o2}$

$$E_{total} = \frac{1}{2}(\mathrm{target}_{o1} - \mathrm{out}_{o1})^2 + \frac{1}{2}(\mathrm{target}_{o2} - \mathrm{out}_{o2})^2$$

$$\frac{\partial E_{total}}{\partial \mathrm{out}_{o1}} = 2 \times \frac{1}{2}(\mathrm{target}_{o1} - \mathrm{out}_{o1})^{2-1} \times (-1) + 0$$

$$\frac{\partial E_{total}}{\partial \mathrm{out}_{o1}} = -(\mathrm{target}_{o1} - \mathrm{out}_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

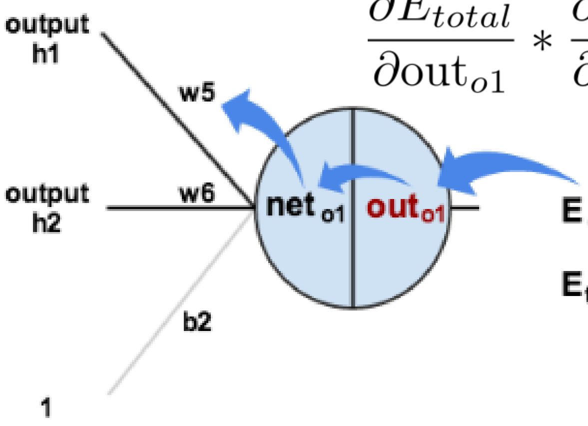
$$\frac{\partial E_{total}}{\partial \text{out}_{o1}} * \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} * \frac{\partial \text{net}_{o1}}{\partial w_5} = \frac{\partial E_{total}}{\partial w_5}$$



$$\text{out}_{o1} = \frac{1}{1 + exp^{-net_{o1}}}$$

$$\frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = \text{out}_{o1}(1 - \text{out}_{o1}) = 0.75146507(1 - 0.75146507) = 0.186815602$$

Backwards Computation(3/5)



$$\frac{\partial E_{total}}{\partial \text{out}_{o1}} * \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} * \frac{\partial \text{net}_{o1}}{\partial w_5} = \frac{\partial E_{total}}{\partial w_5}$$

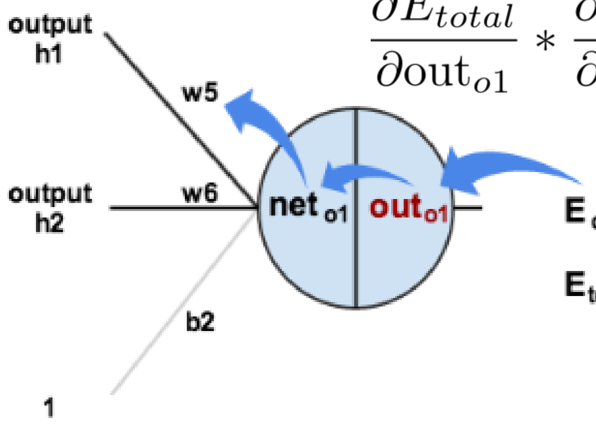
$E_{o1} = \frac{1}{2}(\text{target}_{o1} - \text{out}_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

$$\text{net}_{o1} = w_5 \times \text{out}_{h1} + w_6 \times \text{out}_{h2} + b_2 \times 1$$

$$\frac{\partial \text{net}_{o1}}{\partial w_5} = 1 \times \text{out}_{h1} \times w_5^{(1-1)} + 0 + 0 = \text{out}_{h1} = 0.593269992$$

Backwards Computation(4/5)



$$\frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5} = \frac{\partial E_{total}}{\partial w_5}$$

$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$= 0.74136507 \times 0.186815602 \times 0.593269992 = 0.082167041$$

- To decrease the error, subtract this value from the current weight

$$w_5^+ = w_5 - \eta \times \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 \times 0.082167041 = 0.35891648$$

$$\eta = 0.5$$

First order Markov chains

- A sequence of random variables $x_1, x_2, \ldots$

$$\boxed{x_1} \rightarrow \boxed{x_2} \rightarrow \boxed{x_3} \rightarrow \boxed{x_4} \rightarrow$$

- x_t is the state of the model at time t
- Markov assumption:
  each state is dependency of only on the previous one
  dependency given by a **conditional probability**

$$p(x_t | x_{t-1})$$

- **A first-order Markov chain**
- **Nth-order Markov chain:**

$$p(x_t | x_{t-1}, \ldots, x_{t-N})$$

Computation of Convolution Image Size

$$\text{Output Size(Width)} = \frac{W - F_w + 2 * P}{S} + 1$$

$$\text{Output Size(Height)} = \frac{H - F_h + 2 * P}{S} + 1$$

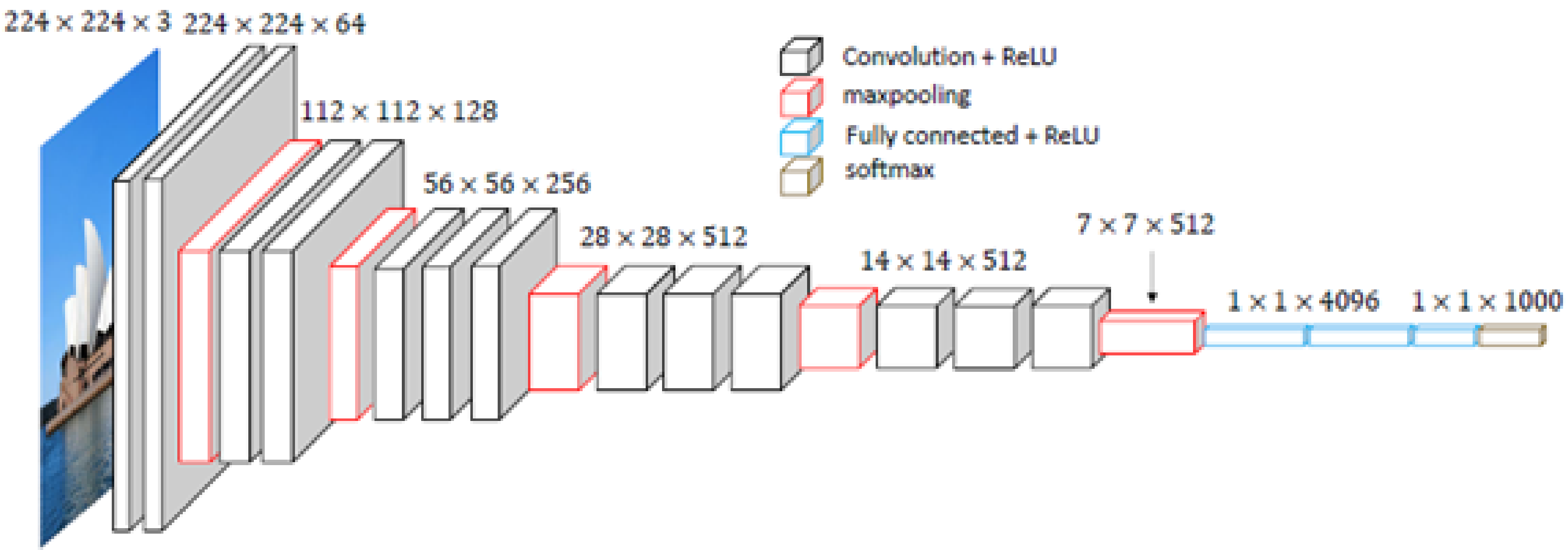
W : Input Size(Width) H : Input Size(Height)

F_w : Filter Width F_h : Filter Height

P : Padding

S : Stride

VGG-16 Structure



VGG-16 Structure

- Number of layers:
  weight layers(conv layers + fully connected layers),
  no count max pooling

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

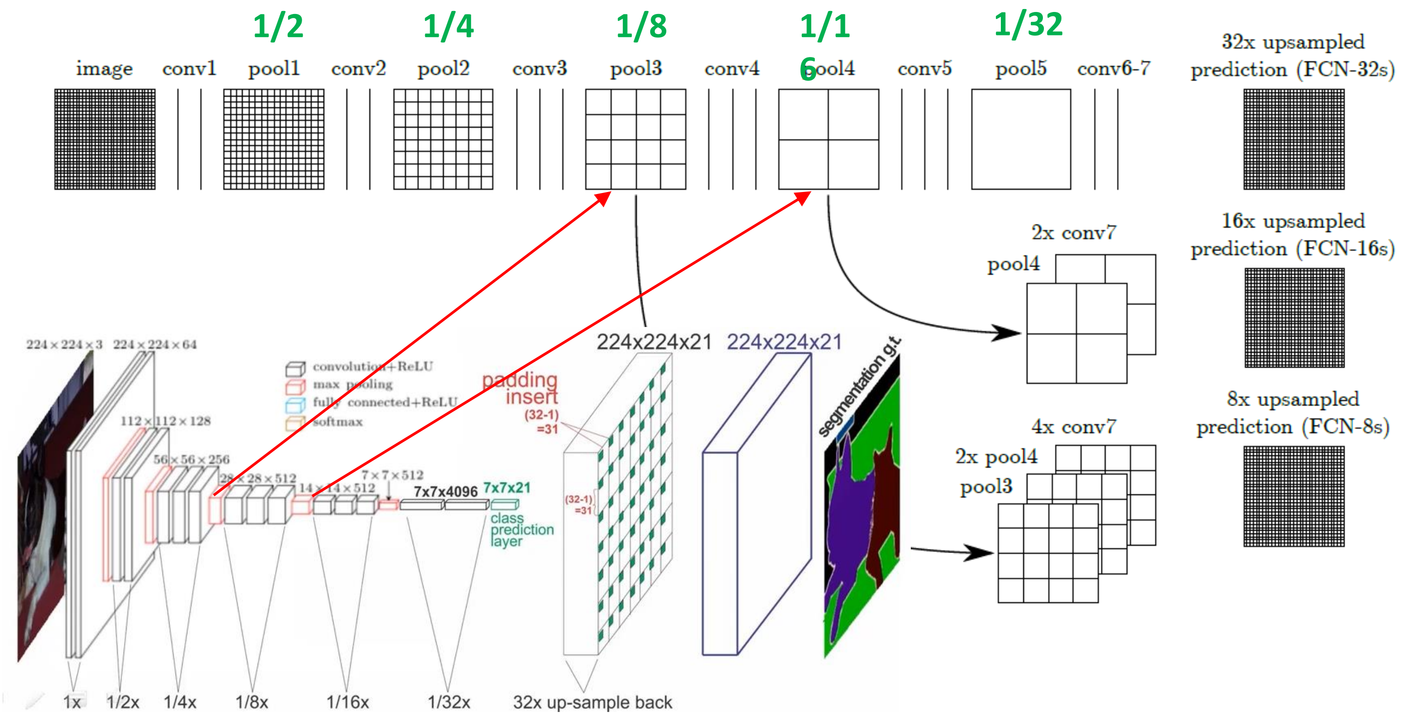
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

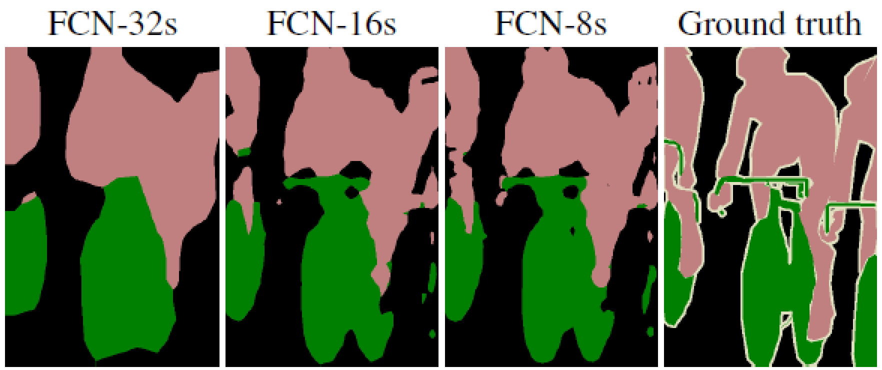
- Previous pixel-level classification
  - Generate local window
  - Object classify within window
  - Center point of window = class
  - Problem: limitation search in local information and computation
- FCN
  - Use CNN architecture
  - Global computation
  - Pixel-level classification
  - Classify at each pixel

FCN architecture



Source: https://youtu.be/UdZnhZrM2vQ

FCN Result



	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

Source: Jonathan Long, Evan Shelhamer, Trevor Darrell, "Fully Convolutional Networks for Semantic Segmentation", TPAMI, Vol. 39, Issue 4, 2016

Experiments

- Hyperparameter : suing existing Faster R-CNN
- Backbone architectures: ResNet50, ResNet101, FPN(Feature Pyramid Networks)
- Input image: resized into 800px for its shorter size
- GPU: 8 GPU @2 images on training
- Training time: 32 hours(ResNet50-FPN), 44 Hours (ResNet101-FPN), not end-to-end training
- Testing time (ResNet101-FPN): 105ms per image on an Nvidia Tesla M40 GPU(plus 15ms CPU time resizing the outputs to the original resolution)
- Dataset: MS COCO(80k train, 35k val, 5k test)
- YOLOv2, SSD->GeForce GTX Titan X(3073 cores, 12Gb)
- Mask R-CNN->Nvidia Tesla M40(3072 cores, 12Gb)

Result

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Result

- Compare with Faster R-CNN in COCO dataset

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

Result

- Mask R-CNN in Cityscapes dataset

	training data	AP [val]	AP	AP$_{50}$	person	rider	car	truck	bus	train	mcycle	bicycle
InstanceCut [23]	fine + coarse	15.8	13.0	27.9	10.0	8.0	23.7	14.0	19.5	15.2	9.3	4.7
DWT [4]	fine	19.8	15.6	30.0	15.1	11.7	32.9	17.1	20.4	15.0	7.9	4.9
SAIS [17]	fine	-	17.4	36.7	14.6	12.9	35.7	16.0	23.2	19.0	10.3	7.8
DIN [3]	fine + coarse	-	20.0	38.8	16.5	16.7	25.7	20.6	30.0	23.4	17.1	10.1
SGN [29]	fine + coarse	29.2	25.0	44.9	21.8	20.1	39.4	24.8	33.2	30.8	17.7	12.4
Mask R-CNN	fine	31.5	26.2	49.9	30.5	23.7	46.9	22.8	32.2	18.6	19.1	16.0
Mask R-CNN	fine + COCO	**36.4**	**32.0**	**58.1**	**34.8**	**27.0**	**49.1**	**30.1**	**40.9**	**30.9**	**24.1**	**18.7**

Table 7. Results on Cityscapes val ('AP [val]' column) and test (remaining columns) sets. Our method uses ResNet-50-FPN.

Result

- Human Pose Estimation

description	backbone	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
original baseline	R-50-FPN	64.2	86.6	69.7	58.7	73.0
+ updated baseline	R-50-FPN	65.1	86.6	70.9	59.9	73.6
+ deeper	R-101-FPN	66.1	87.7	71.7	60.5	75.0
+ ResNeXt	X-101-FPN	67.3	88.0	73.3	62.2	75.6
+ data distillation [35]	X-101-FPN	**69.1**	**88.9**	**75.3**	**64.1**	**77.1**
+ test-time augm.	X-101-FPN	**70.4**	**89.3**	**76.8**	**65.8**	**78.1**

Table 9. **Enhanced keypoint results** of Mask R-CNN on COCO `minival`. Each row adds an extra component to the above row. Here we use only keypoint annotations but no mask annotations. We denote ResNet by 'R' and ResNeXt by 'X' for brevity.

OBJECT DETECTION

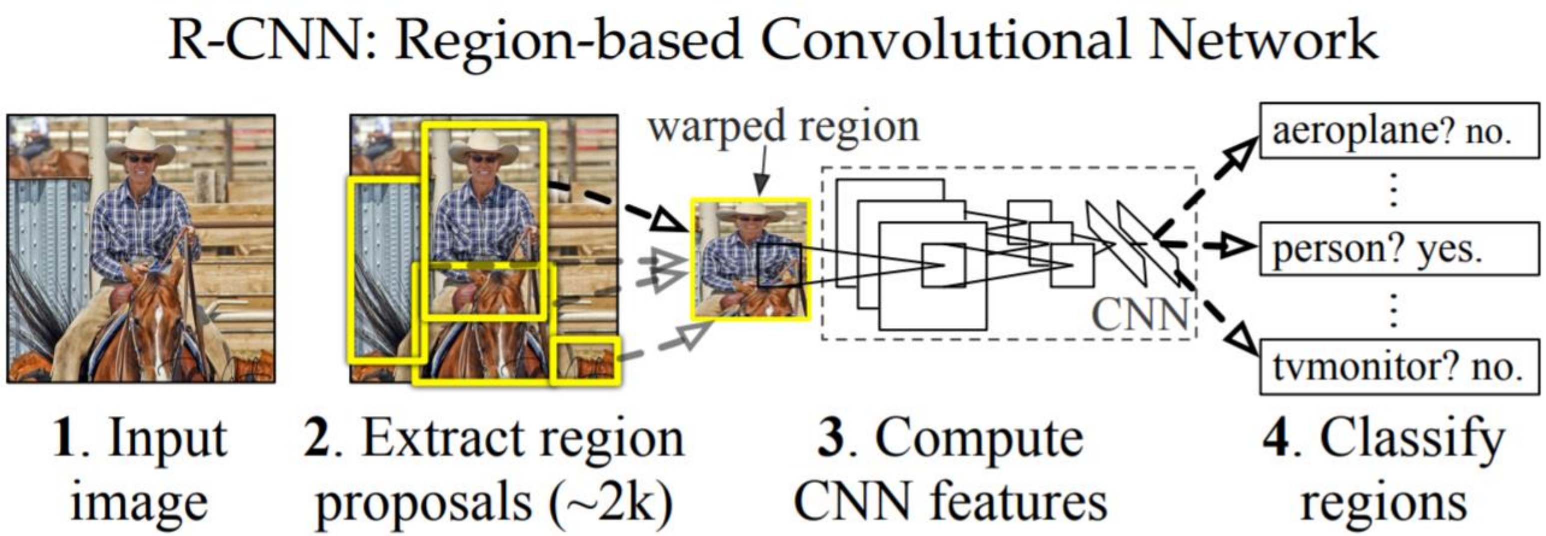
Selective Search

- Selective Search by Hierarchical grouping
  - Create initial regions
  - Greedy algorithm to iteratively group regions
    - Similarities between all neighbouring regions
      - $R = r_{,} ..., r_n$: initial regions $s(r_i, r_j)$: similarity
    - Group together with two similar regions
    - Calculate new similarities between resulting region and its neighbors
  - Repeat the process of grouping the most similar regions until the whole image becomes a single region

Hierarchical Grouping Results

R-CNN Architecture

- Takes an input image
- Extracts around 2000 bottom-up region proposals(by selective search)
- Computes features for each proposal using a large convolutional network
- Classifies each region using class-specific linear SVMs

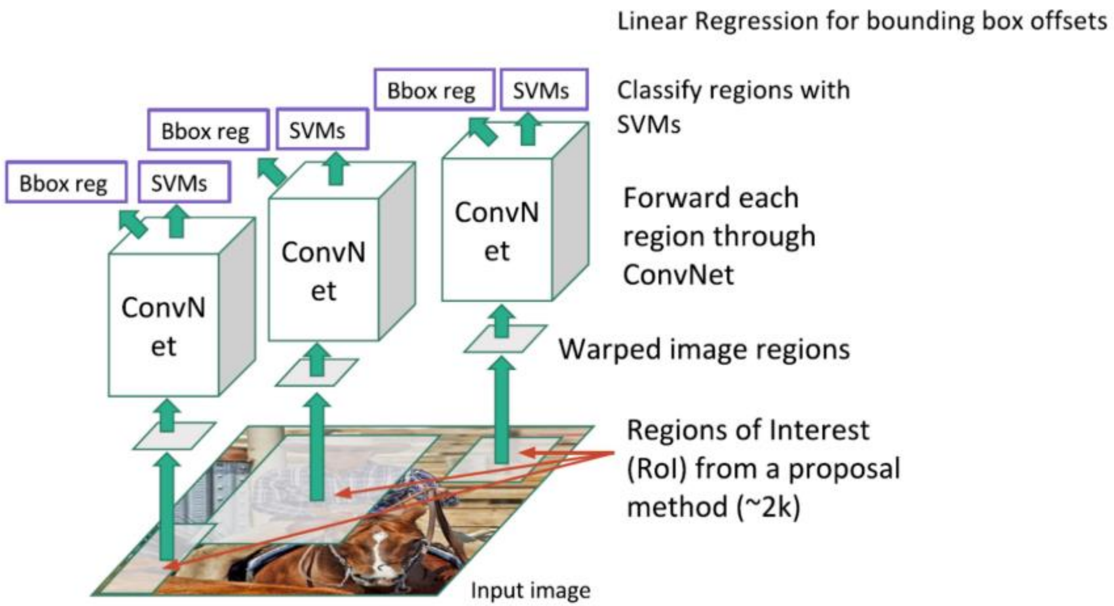


R-CNN, object detection system overview

Source: Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, "Region-based Convolutional Networks for Accurate Object Detection and Segmentation", PAMI, Vol 38, Issue 1, 2015

R-CNN Architecture

- Regional Proposal + CNN
- Regional Proposal by Selective search
- Compute the CNN at each proposal region(many computation)
- Improve the detection accuracy by bounding box regression



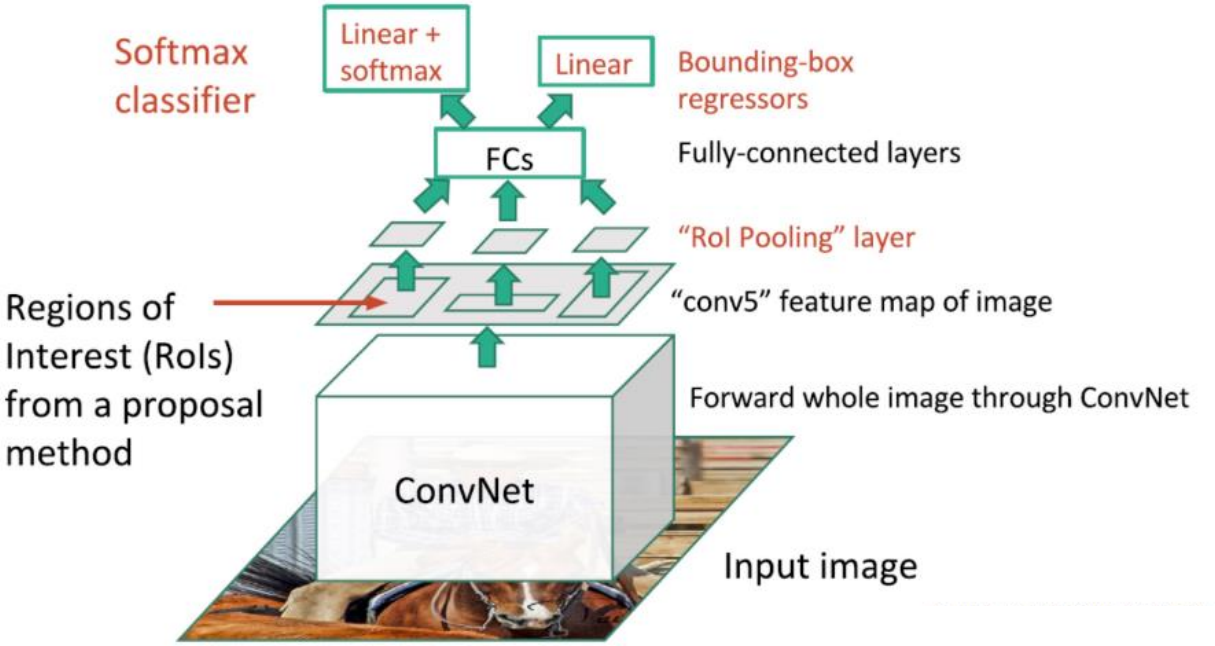
R-CNN Architecture

- Problem
  - Slow test
    - 13s/image in GPU
    - 53s/image in CPU
  - Learning process complex
    - 84 hours in GPU

Source: https://jamiekang.github.io/2017/05/28/faster-r-cnn/

Fast R-CNN

- Initial RoI by Selective search
- CNN for whole image
- After CNN, RoI apply with feature map
- Different from R-CNN that is no convolution computation at each RoI

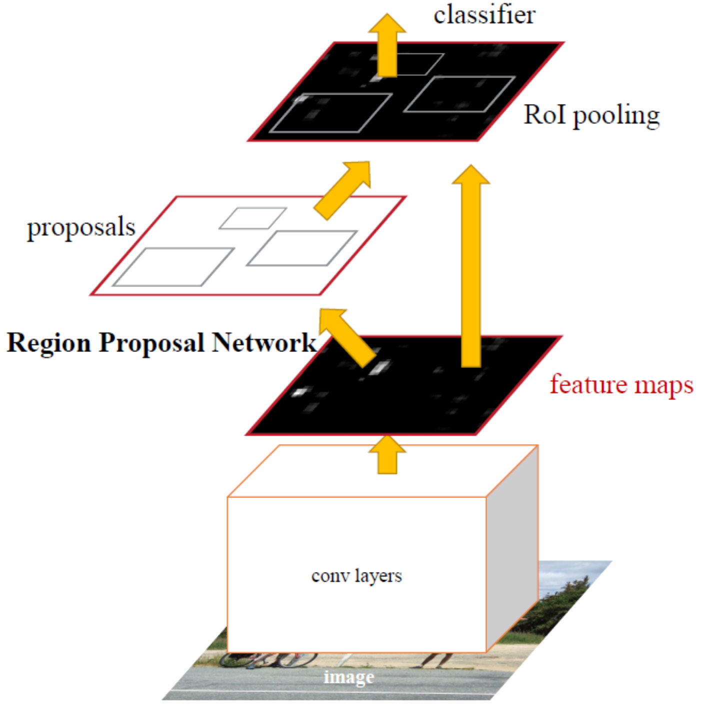


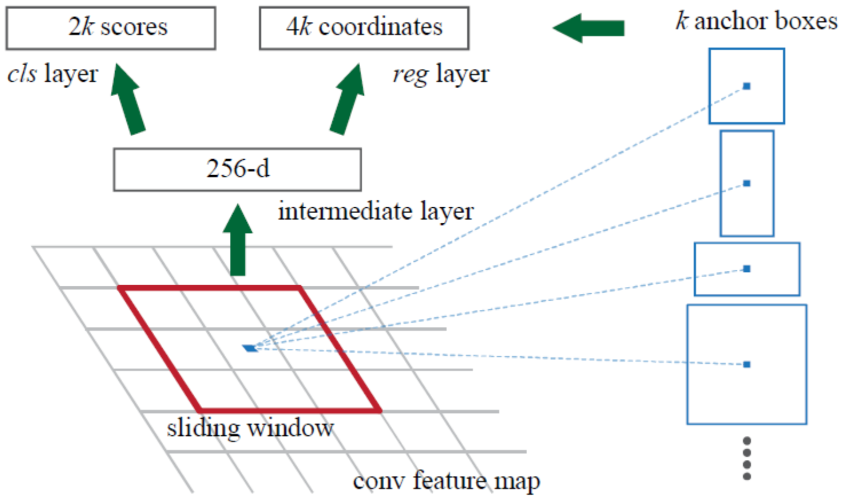
Fast R-CNN Architecture

Source: https://jamiekang.github.io/2017/05/28/faster-r-cnn/

Faster R-CNN

- Region Proposal Networks
  - No use selective search
  - RoIpooling, classifier and bounding box regressor

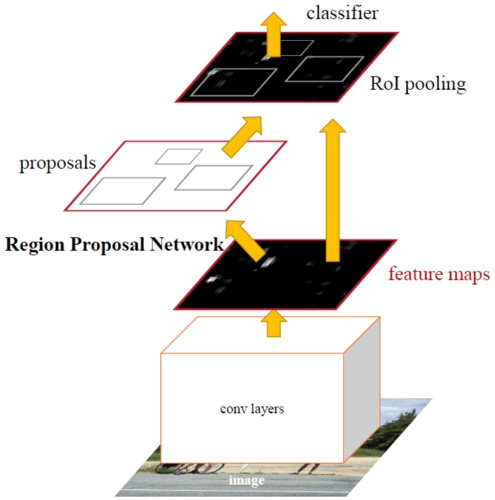


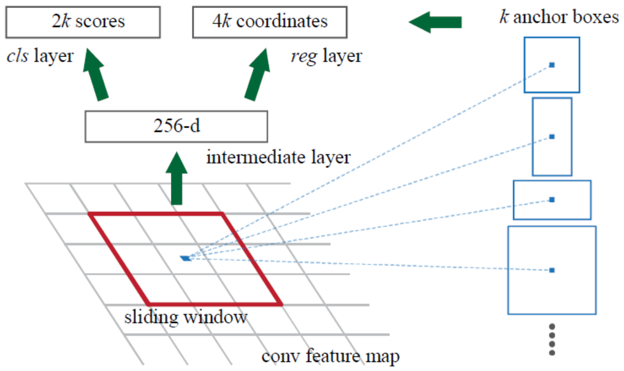


Source: https://jamiekang.github.io/2017/05/28/faster-r-cnn/

Faster R-CNN

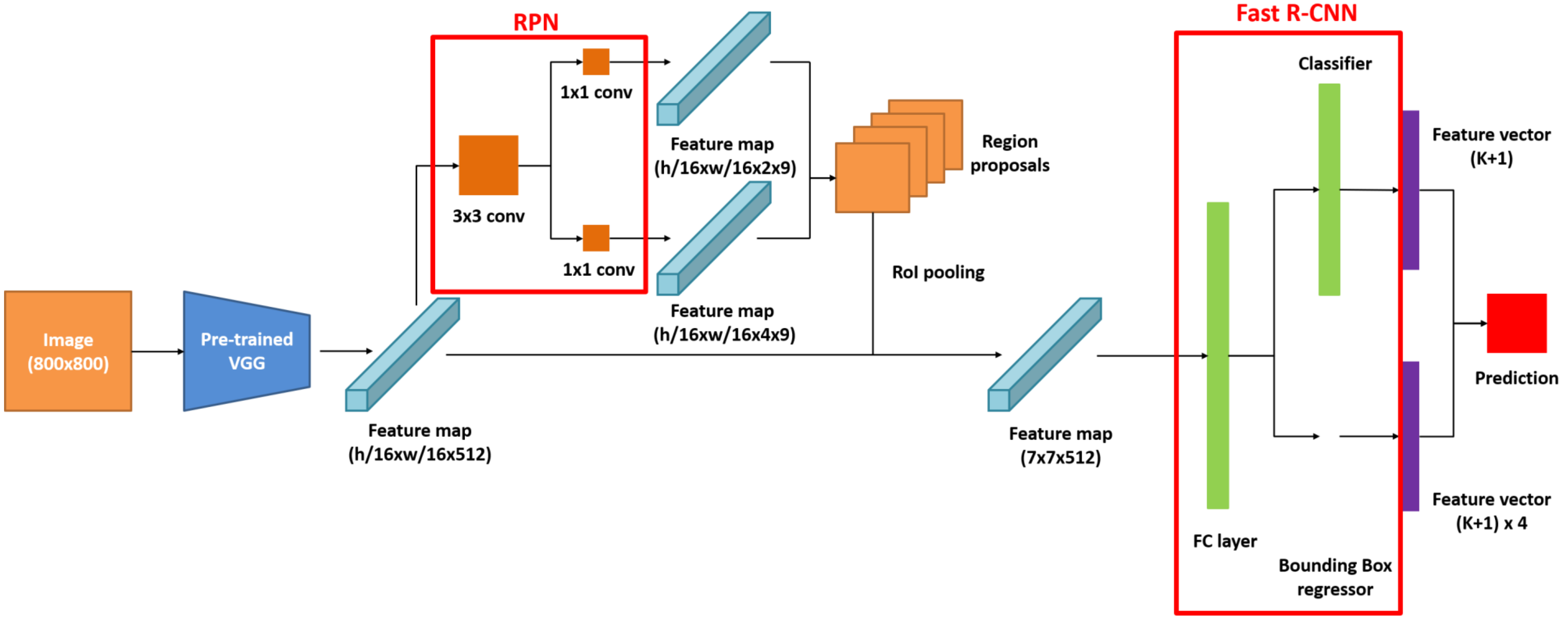
- Region Proposal Networks
  - K proposals are parameterized relative to k reference boxes, which are called *anchor*
  - *K=9(3 scales(128, 256, 512) and 3 aspect ratio(1:1, 1:2, 2:1))*
  - *Image resize: PASAL VOC 2007(500x375) -> 1000x600*
  - 2k scores(there is object in bounding box or not)
  - 4k coordinates: x, y(top-left), H, W



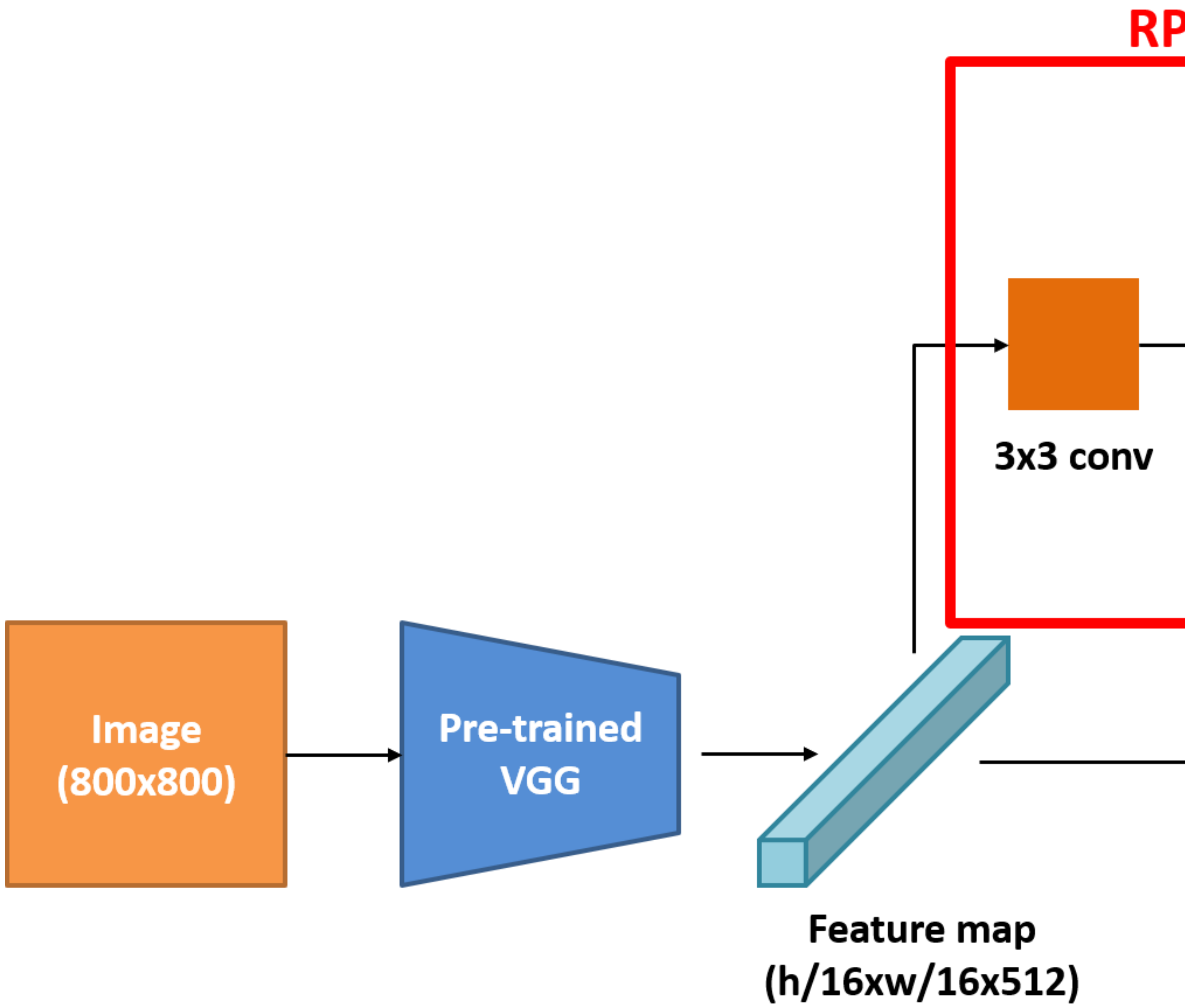


Source: https://jamiekang.github.io/2017/05/28/faster-r-cnn/

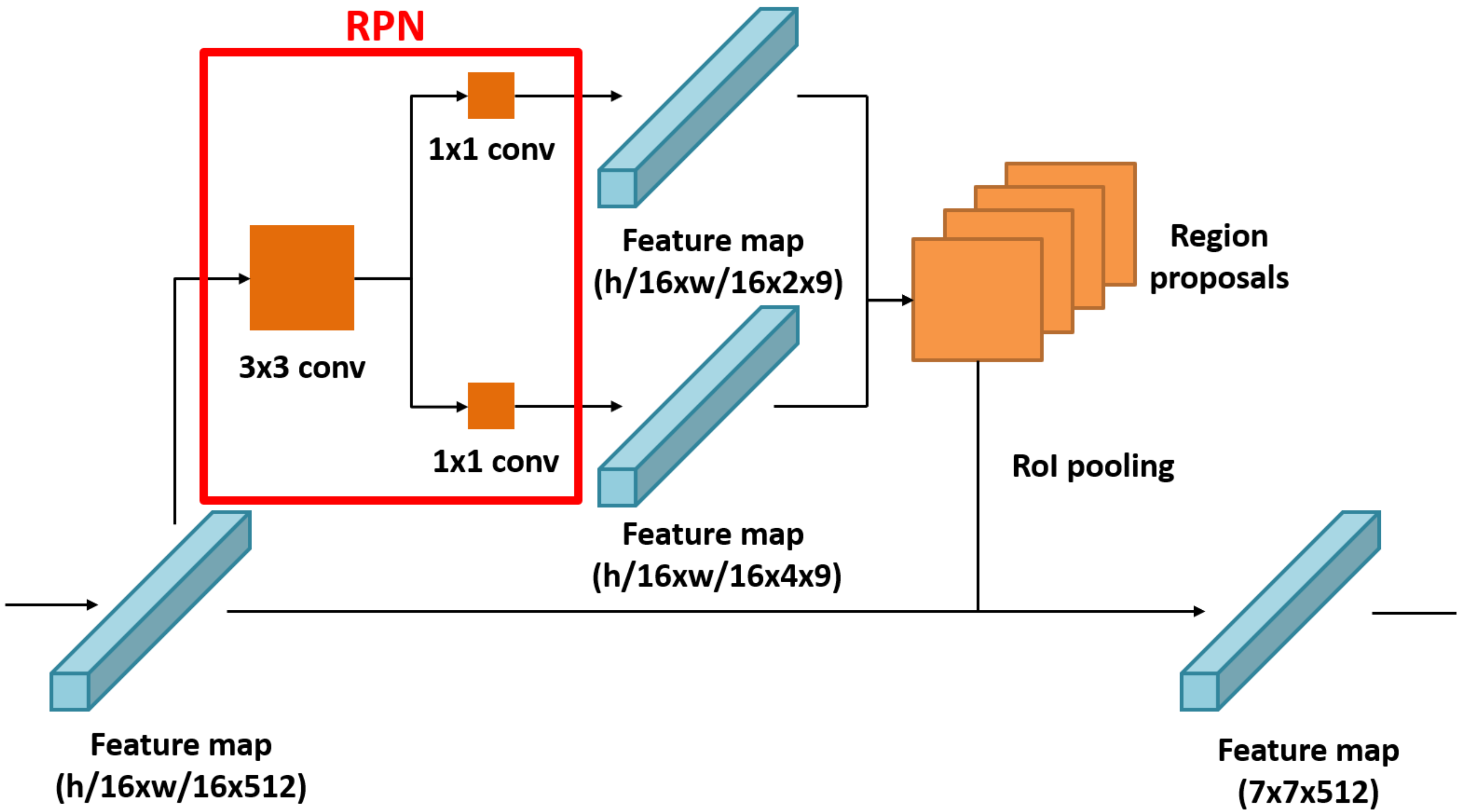
Architecture of Faster R-CNN



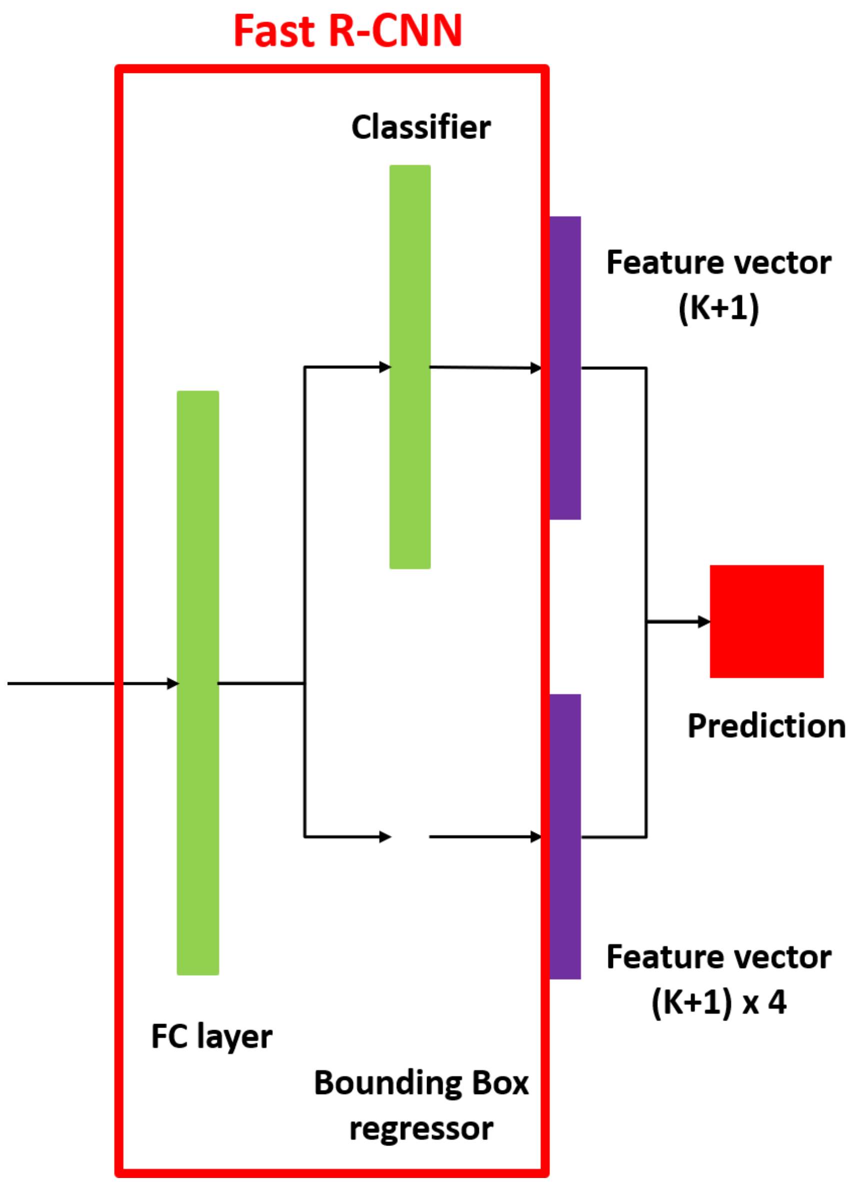
Part of Feature Extraction

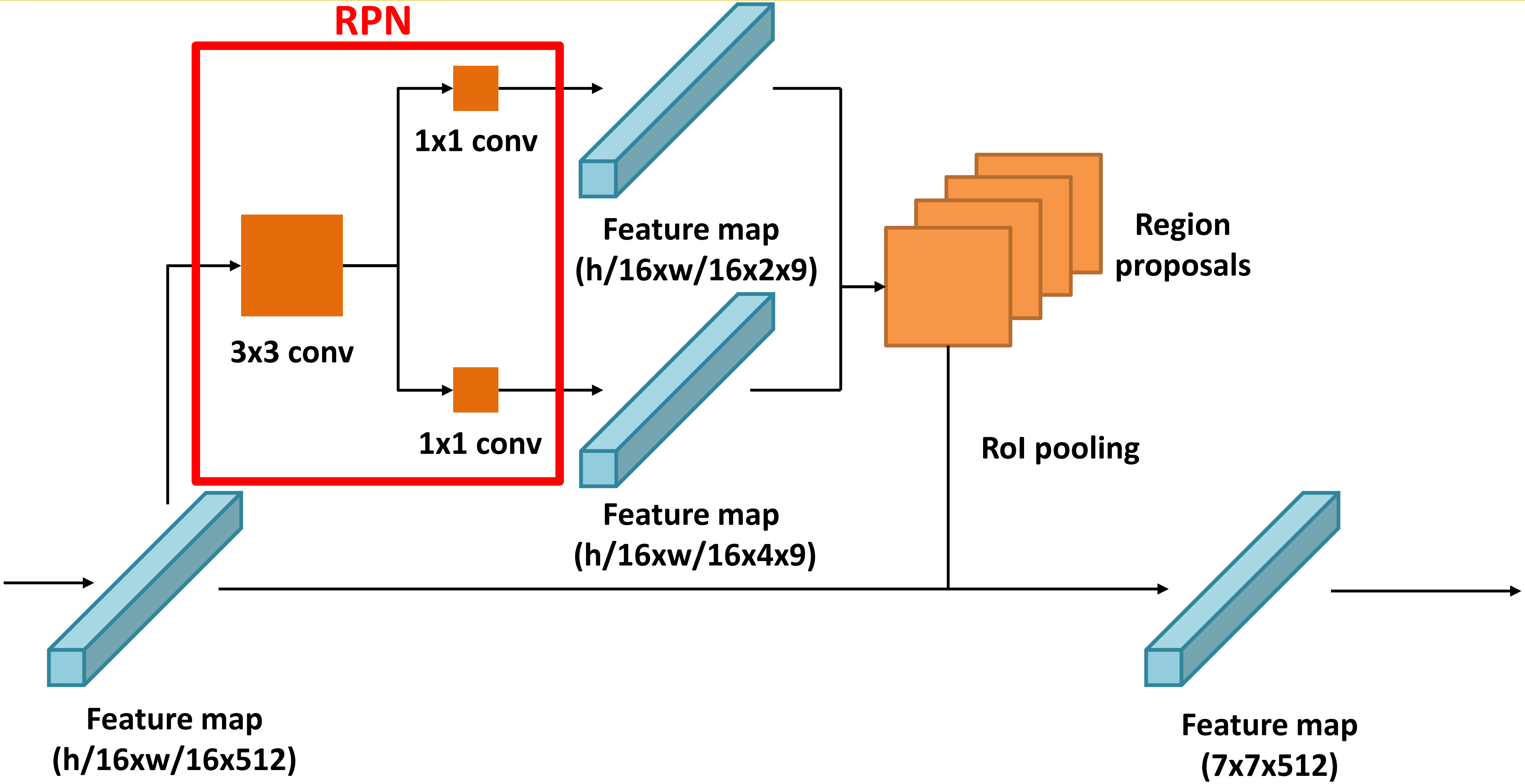


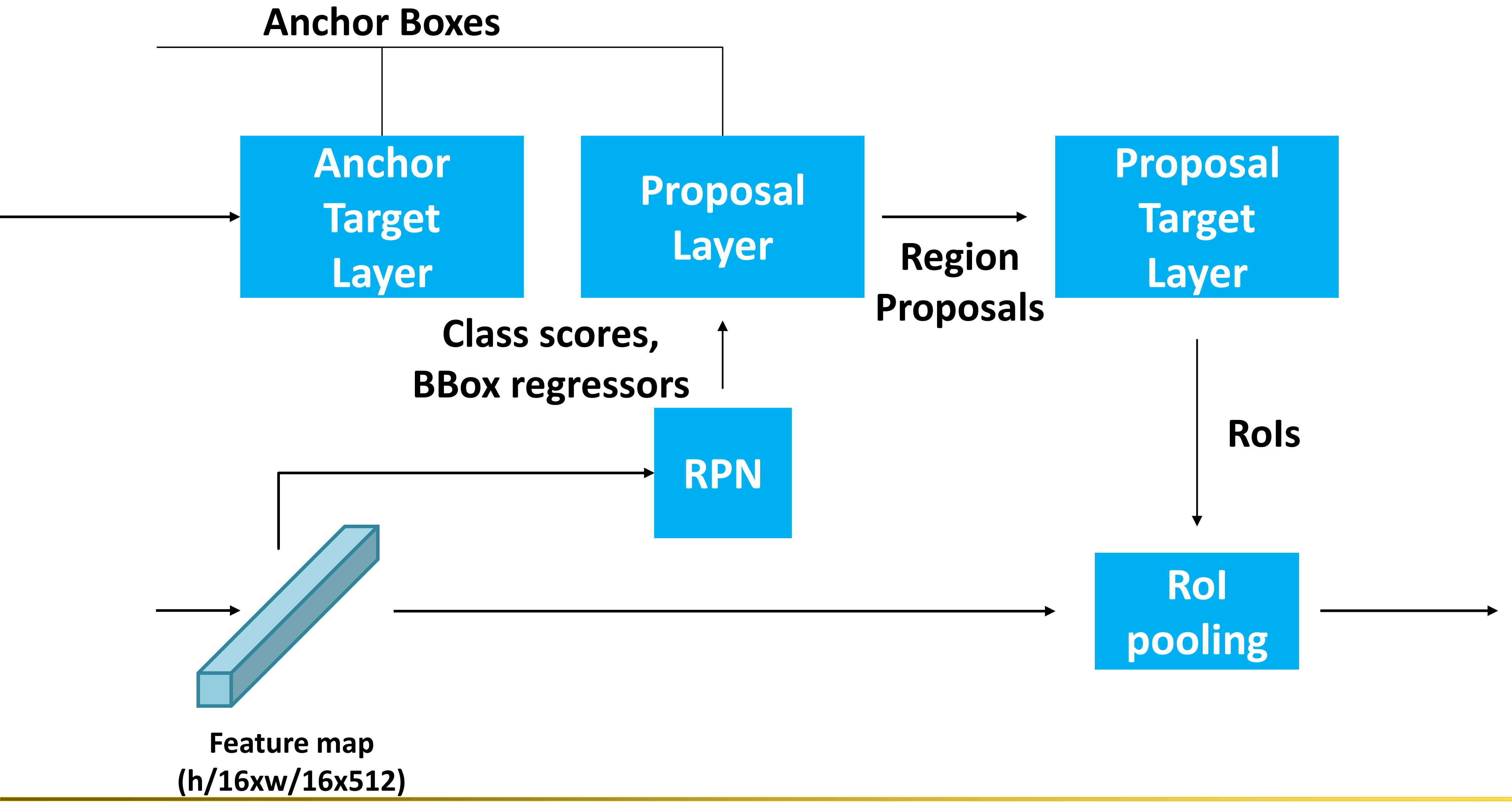
Part of Region Proposal Network(RPN)

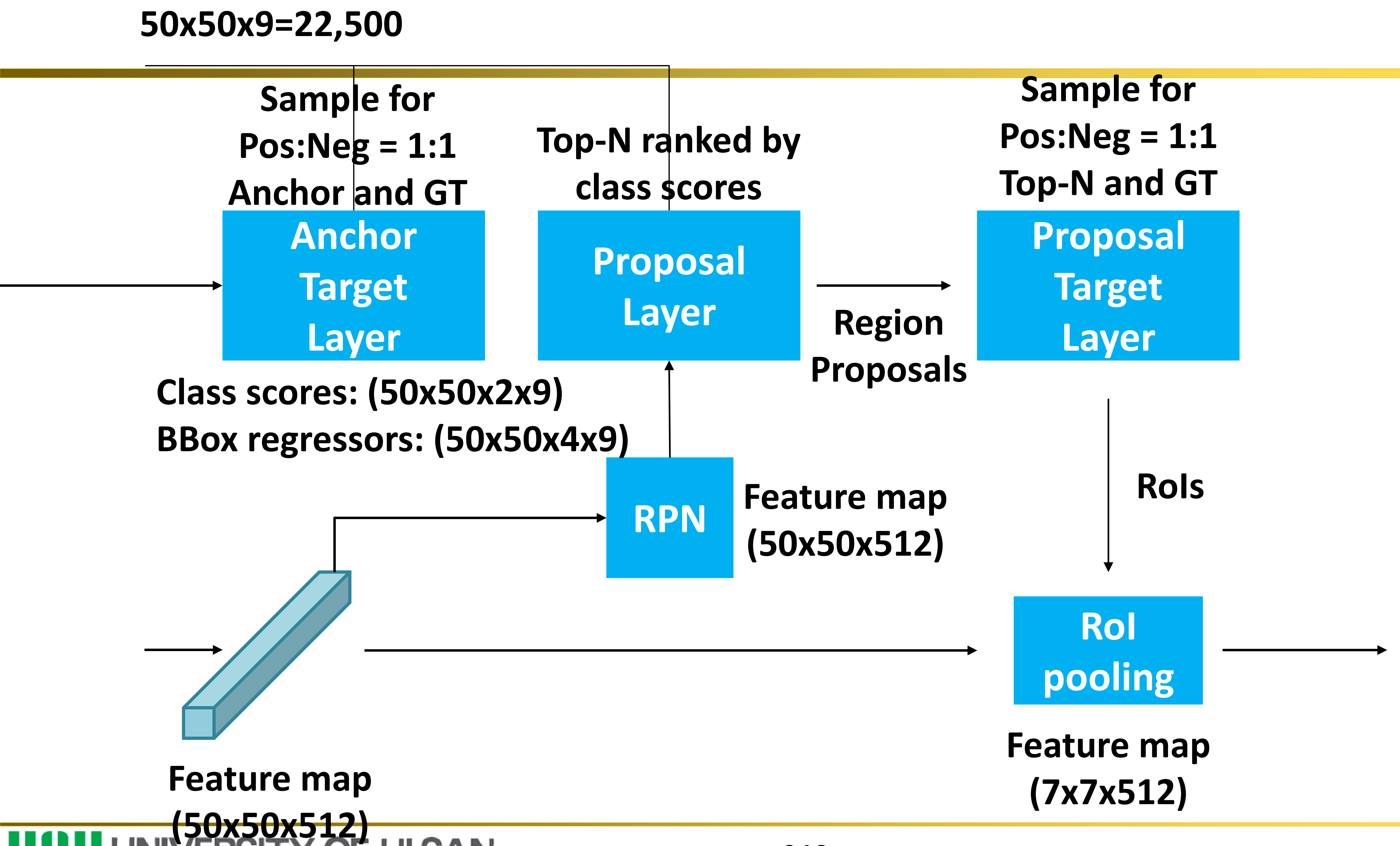


Part of Prediction




RPN
1x1 conv
3x3 conv
1x1 conv
Feature map
(h/16xw/16x2x9)
Feature map
(h/16xw/16x4x9)
Region
proposals
RoI pooling
Feature map
(h/16xw/16x512)
Feature map
(7x7x512)
FC layer


Anchor Boxes
Anchor Target Layer
Proposal Layer
Region Proposals
Proposal Target Layer
Class scores, BBox regressors
RPN
RoIs
RoI pooling
Feature map (h/16xw/16x512)


50x50x9=22,500
Sample for
Pos:Neg = 1:1
Anchor and GT
Top-N ranked by
class scores
Sample for
Pos:Neg = 1:1
Top-N and GT
Anchor
Target
Layer
Proposal
Layer
Region
Proposals
Proposal
Target
Layer
Class scores: (50x50x2x9)
BBox regressors: (50x50x4x9)
RPN
Feature map
(50x50x512)
RoIs
RoI
pooling
Feature map
(50x50x512)
Feature map
(7x7x512)

Multi-Task Loss

- Training classifier and bounding box regressor at the same time

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

- Predicted probability of anchor i being an object

$$p^* = (p_0^*, p_1^*, \ldots, p_i^*)$$

- Ground-truth label: $p = (p_0, p_1, \ldots, p_i)$
- Classification loss: Log loss

$$L_{cls}(p_i, p_i^*) = -(p_i^* log(p_i) + (1 - p_i^*) log(1 - p_i))$$

- Training classifier and bounding box regressior at the same time

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*)$$

- Coordinates of the predicted bounding box

$$t^* = (t_0^*, t_1^*, \ldots, t_i^*)$$

- Ground-truth bounding box: $t = (t_0, t_1, \ldots, t_i)$
- Smooth L1 loss function:

$$L_{reg}(t_i,t_i^*) = R(t_i - t_i^*) = \sum_{i\in\{x,y,w,h\}} smooth_{L_1}(t_i - t_i^*)$$

$$smooth_{L_1}(t_i - t_i^*) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

$$\lambda = 1: \ \text{balancing hyperparameter}$$

- Converting coordinate as follows
- Ground-truth bounding box: $t = (t_0, t_1, \ldots, t_i)$

$$t_x = (x - x_a)/w_a \quad t_y = (y - y_a)/h_a$$

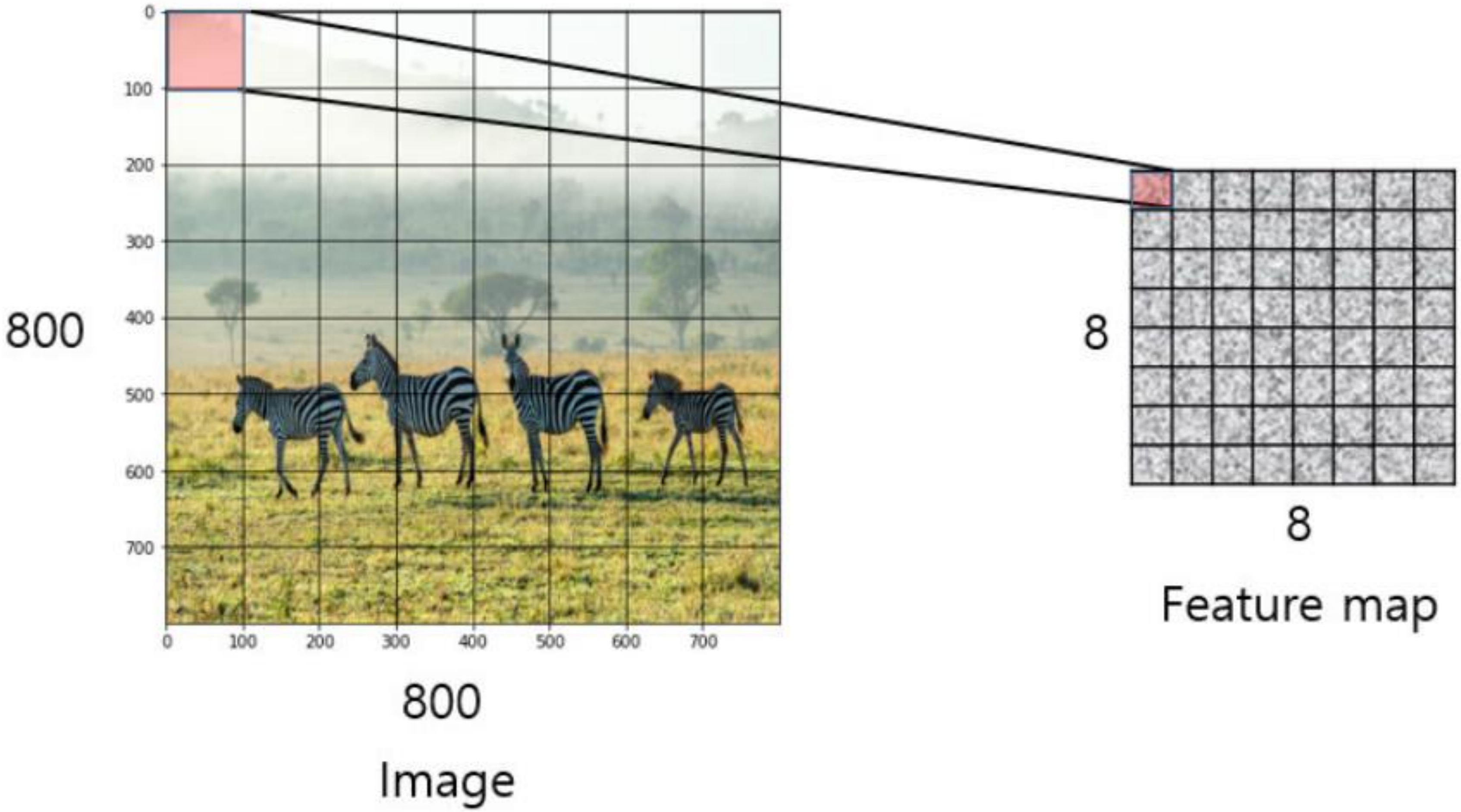
$$t_w = \log(w/w_a) \qquad t_h = \log(h/h_a)$$

- Coordinates of the predicted bounding box: $t^* = (t_0^*, t_1^*, \ldots, t_i^*)$

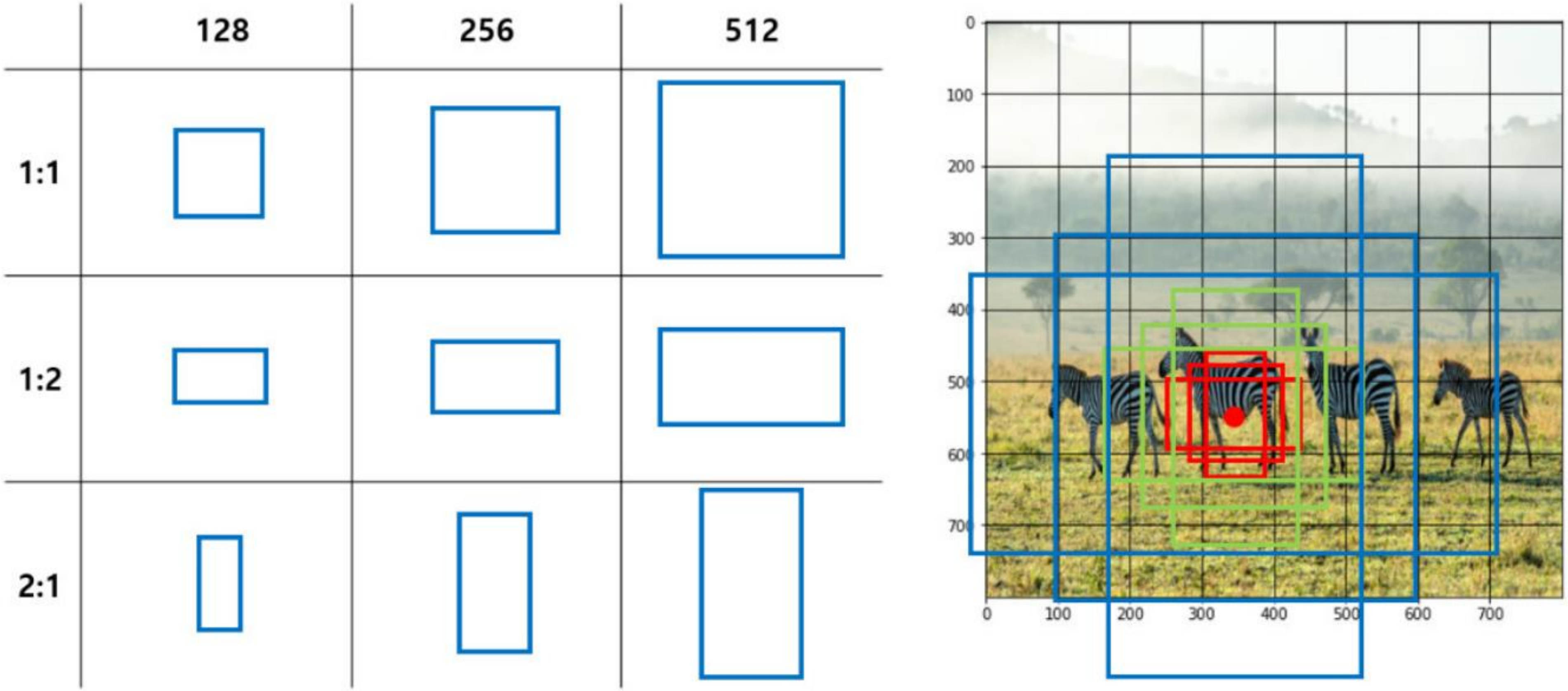
$$t_x^* = (x^* - x_a)/w_a \quad t_y^* = (y^* - y_a)/h_a$$

$$t_w^* = \log(w^*/w_a) \qquad t_h^* = \log(h^*/h_a)$$

- Generating the dense sampling for proposal region



- Suggested three kinds of scales and aspect ratio for anchor box



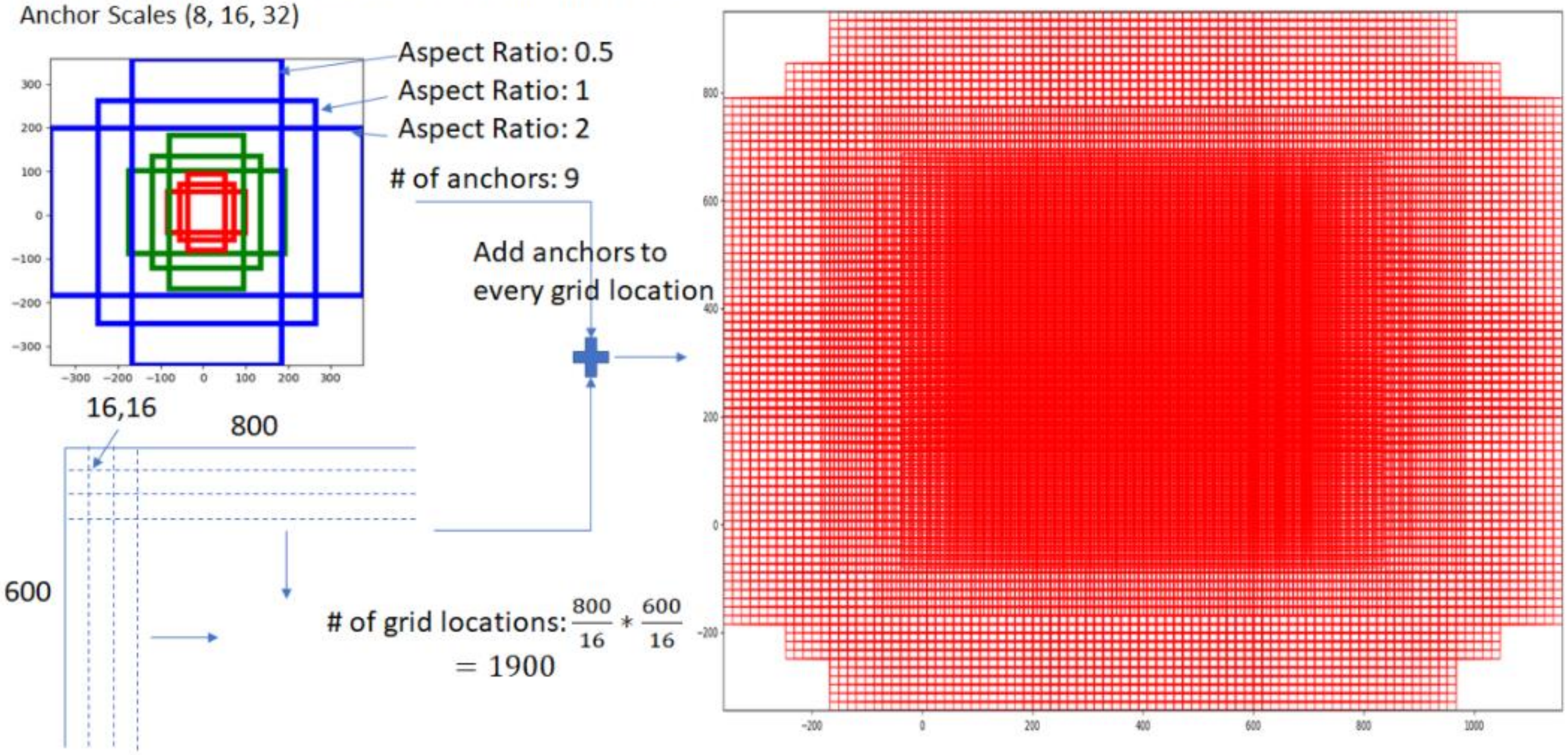
Anchor Box (3/3)

Generate Anchors

Given:

- Set of aspect ratios (0.5, 1, 2)
- Stride length (downscaling performed by resnet head: 16)
- Anchor Scales (8, 16, 32)

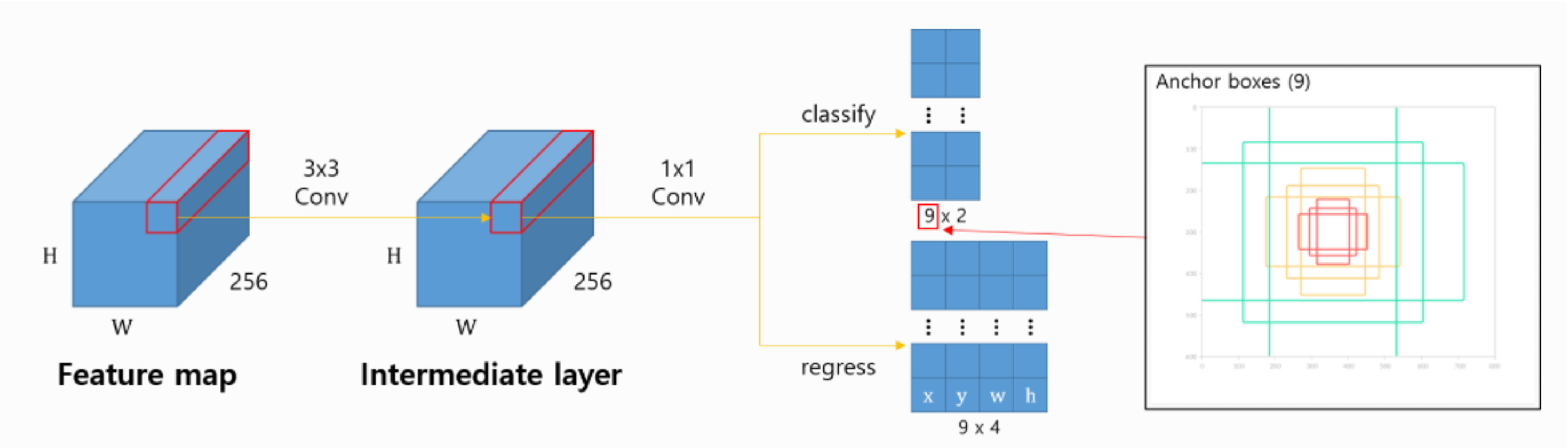
Total number of anchors: 1900*9 = 17100
Some boxes lie outside the image boundary



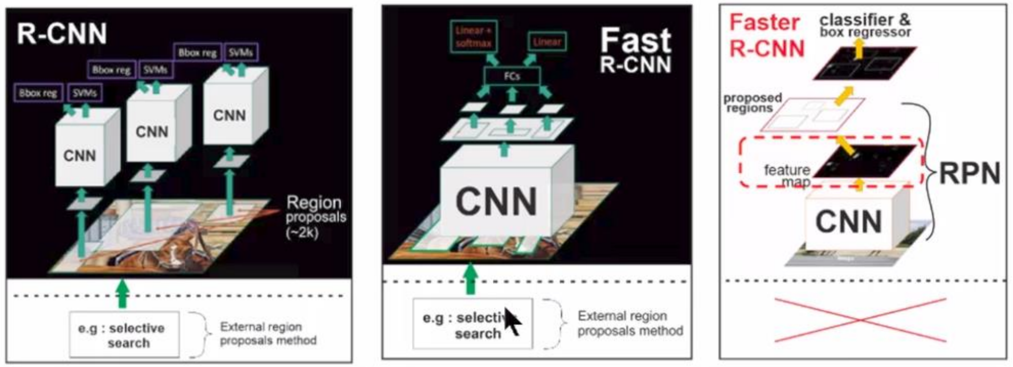
$$\text{\# of grid locations: } \frac{800}{16} * \frac{600}{16} = 1900$$

Create uniformly spaced grid with spacing = stride length

- Positive sample(128):Negative sample(128) = 1:1
- To make it balance of proposed bbox class and regression(position)



R-CNN vs Fast vs Faster R-CNN



	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

* Standford lecture notes on CNN by Fei Fei Li and Andrej Karpathy

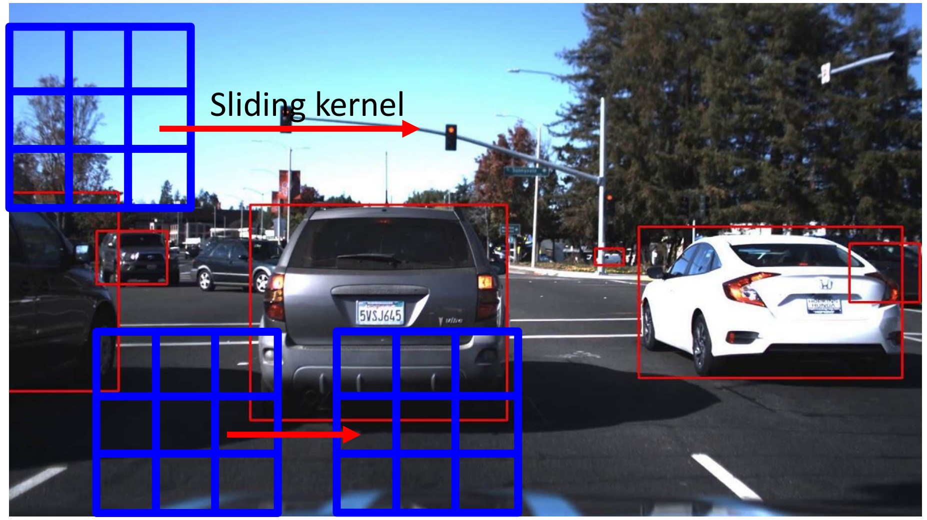
5

By Ardian Umam

Source: https://www.youtube.com/watch?v=v5bFVbQvFRk

Bounding Box Predictions

- By sliding the kernel, it is a convolution process with ground truth
- Some of kernels are matched with ground truth
- How much match kernel with ground truth



Source: https://www.google.co.kr/search?q=udacity+dataset&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjFitKu8-jaAhVElZQKHY0QCjUQ_AUICigB&biw=1474&bih=750#imgrc=Fi2QqVQZN63kMM: , UDACITY dataset

DATASETS

IMAGENET Dataset

- ImageNet 2012(Classification)
  - ImageNet Large Scale Visual Recognition Challenge(ILSVRC)
  - 1.28 million training images
  - 50k validation images
  - 100k test images
  - 1,000 categories

https://image-net.org/index.php

Datasets

- MS COCO: Microsoft COCO: Common Objects in Context
  - Object segmentation
  - Recognition in context
  - Superpixel stuff segmentation
  - 330K images
  - 1.5 million object instances
  - 80 object categories
  - 91 stuff categories
  - 5 captions per image
  - 250,000 people with keypoints

PASCAL VOC

- PASCAL VOC project
  - Pattern Analysis, Statistical Modelling and Computational Learning(PASCAL),
  - VOC(Visual Object Classes)
  - Provides standardized image data sets for object class recognition
  - Provides a common set of tools for accessing the data sets and annotations
  - Enables evaluation and comparison of different methods
  - Ran challenges evaluating performance on object class recognition (from 2005-2012, now finished)
- Organizers
  - Mark Everingham (University of Leeds)
  - Luc van Gool (ETHZ, Zurich)
  - Chris Williams (University of Edinburgh)
  - John Winn (Microsoft Research Cambridge)
  - Andrew Zisserman (University of Oxford)

PASCAL VOC 2007/2012

- Classes: 20
  - Person(1): person
  - Animal(6): bird, cat, cow, dog, horse, sheep
  - Vehicle(7): aeroplane, bicycle, boat, bus, car, motorbike, train
  - Indoor(8): bottle, chair, dining, table, potted, plant, sofa, tv/monitor
- Train/validation/test
  - 9,963image containing 24,640 annotated objects

- Classes: 20
- Train/validation/test
  - 11,530 images containing 27,450 RoI annotated objects and 6,929 segmentations

CITYSCAPES

- Semantic Understanding of Urban Street Scenes
- Labs: Daimler AG R&D, Max Planck Institute for Informatics, TU Darmstadt Visual Inference Group, Germany
- Volume
  - 2048x1024 pixels
  - 5,000 annotated images with fine annotations
    - https://www.cityscapes-dataset.com/examples/#fine-annotations
  - 20,000 annotated images with coarse annotations
    - https://www.cityscapes-dataset.com/examples/#coarse-annotations
- Metadata
  - Preceding and trailing video frames. Each annotated image is the 20th image from a 30 frame video snippets
  - Corresponding right stereo views
  - GPS coordinates
  - Ego-motion data from vehicle odometry
  - Outside temperature from vehicle sensor
- Benchmark suite and evaluation server
  - Pixel-level semantic labeling
  - Instance-level semantic labeling

CITYSCAPES

- Features
  - Annotations
    - Semantic
    - Instance-wise
    - Dense pixel annotations
- 30 classes
  - Flat: road, sidewalk, parking, rail track
  - Human: person, rider
  - Vehicle: car, truck, bus, on rails, motorcycle, bicycle, caravan, trailer
  - Construction: building, wall, fence, guard rail, bridge, tunnel
  - Object, pole, pole group, traffic sign, traffic light
  - Nature: vegetation, terrain
  - Sky: sky
- Diversity
  - 50cities
  - Several months(spring, summer, fall)
  - Daytime
  - Good/medium weather conditions
  - Manually selected frames
    - Large number of dynamic objects
    - Varying scene layout
    - Varying background

ADE20K Dataset

- ADE20K Dataset
  - Massachusetts Institute of Technology , USA, University of Toronto, Canada
  - ADE stands for Adela Barriuso who did handedly annotated entire dataset
  - 150 object and stuff classes / train: 25,574, val: 2,000
  - Segmentation, image size: 512 x 512

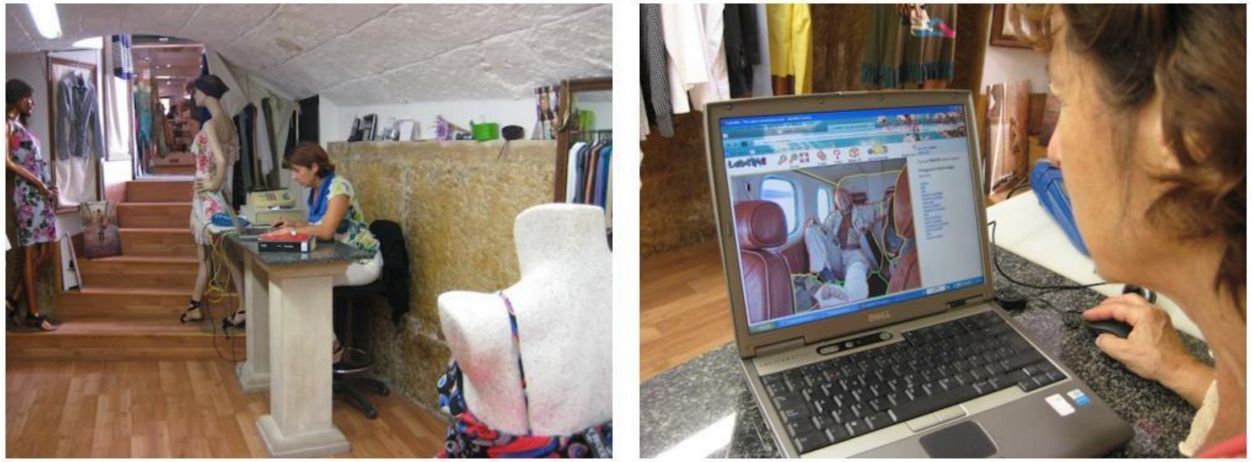


Figure 2: *The image annotation context. All the labeling was done inside a clothing shop named* Transparencia *in the heart of Palma de Mallorca, Spain.*

Brainwash Dataset

- Brainwash dataset in 2015
  - Head detection
  - Train/val: 11,917 images, 91,146 annotated heads
  - Image size: 480x640

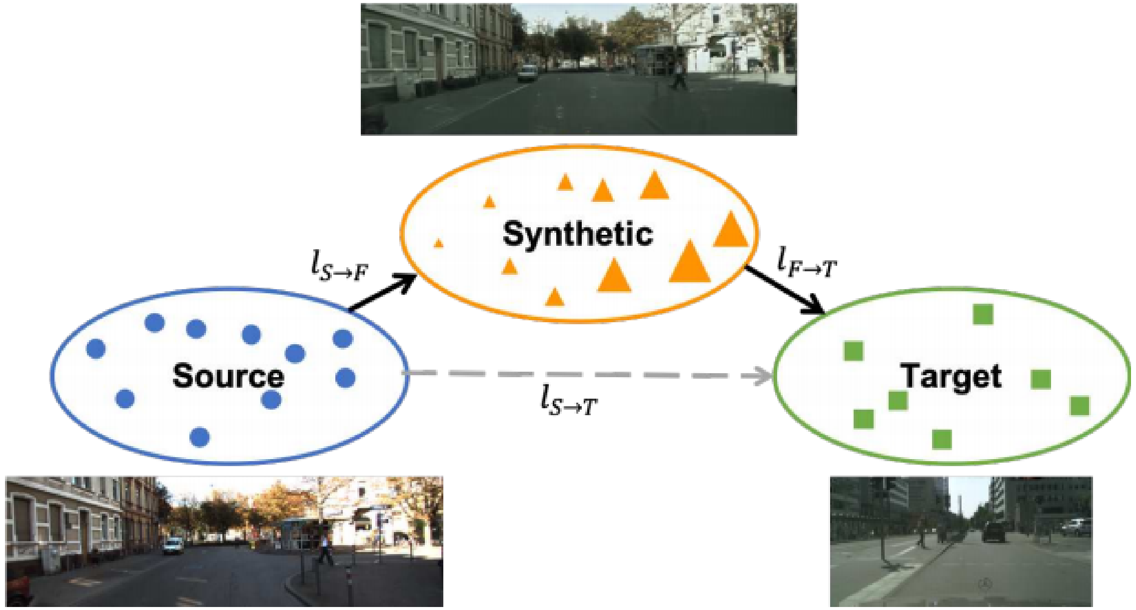
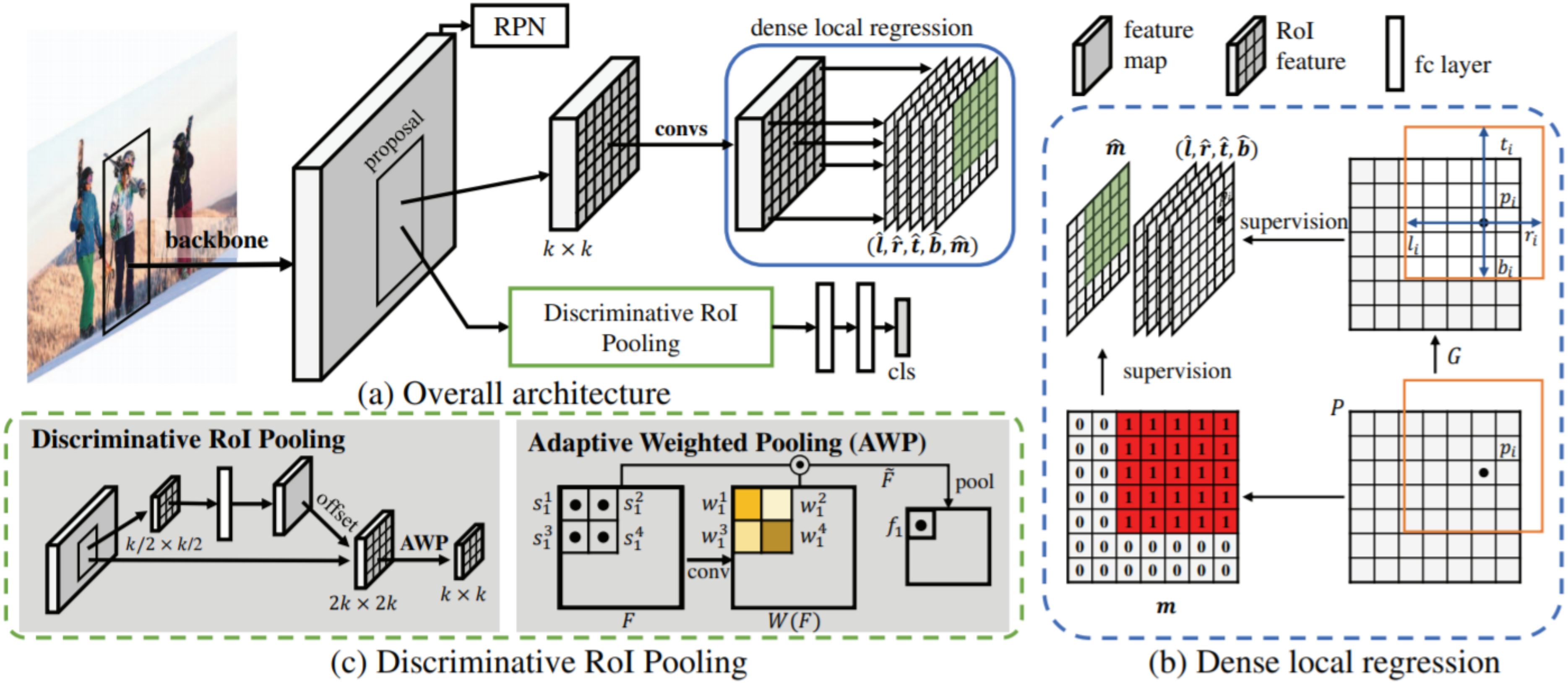
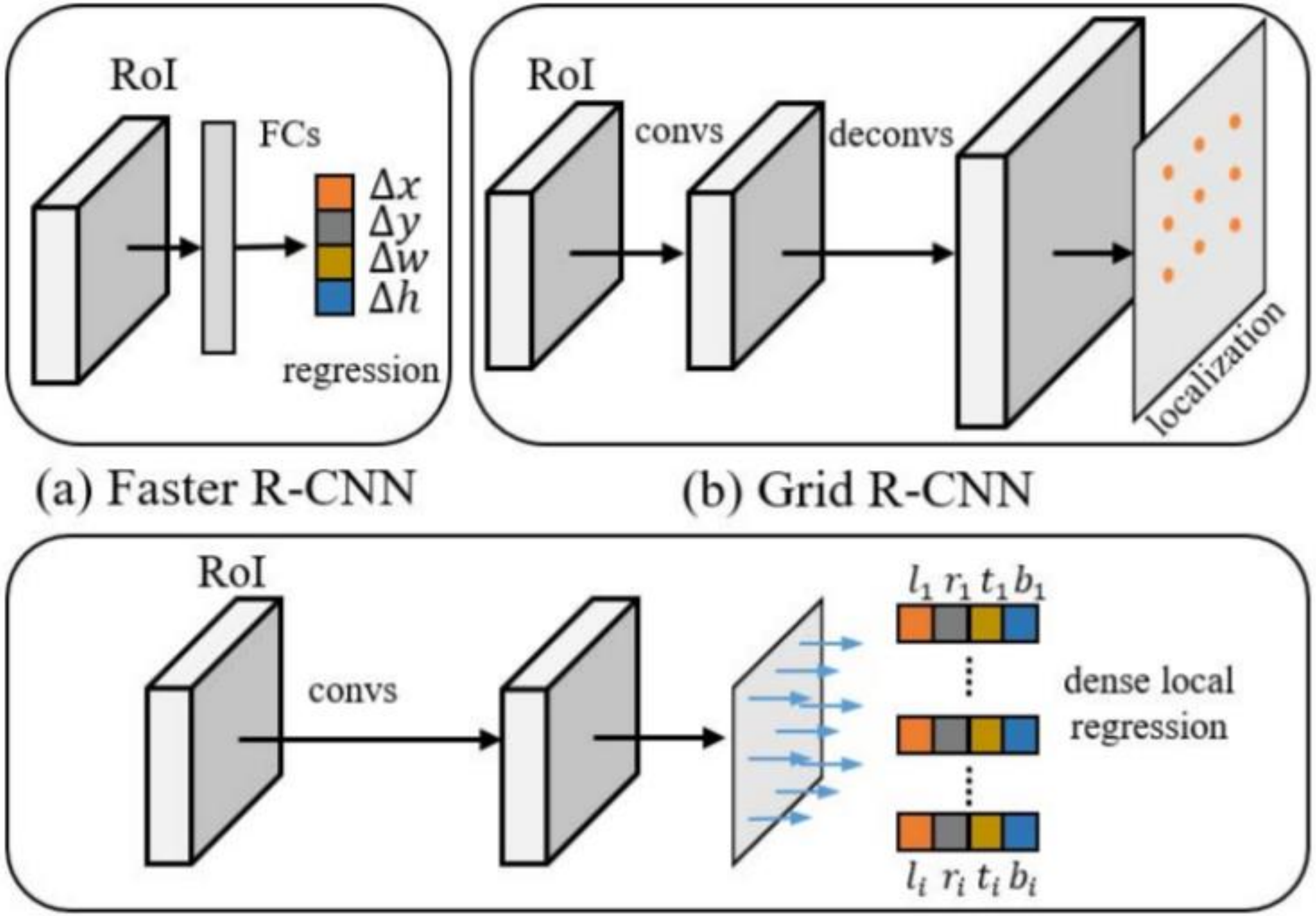


Figure 1. An illustration of our progressive adaptation method. Conventional domain adaptation aims to solve domain-shift problem from source to target domain, which is denoted as $l_{\mathbb{S}\to\mathbb{T}}$. We propose to bridge this gap with an intermediate synthetic domain that allows us to gradually solve separate subtasks with smaller gaps (shown as $l_{\mathbb{S}\to\mathbb{F}}$ and $l_{\mathbb{F}\to\mathbb{T}}$). In addition, we treat each image in the synthetic domain unequally based on its quality with respect to the target domain, where the size of the yellow triangles stand for their weights (i.e., the closer to target, the higher of the weight).

- D2Det redesigns both regression and classification branches of the traditional two-stage R-CNN detectors by dense local regression and discriminative classification



Comparison of Dense Local Regression with Traditional Regression



(a) Faster R-CNN (b) Grid R-CNN

(c) Our approach

FCN vs FC

- Traditional regression in Faster R-CNN
  - Candidate object proposal: $P = (x_P, y_P, w_P, h_P)$
  - Ground Truth box: G$= (x_G, y_G, w_G, h_G)$
  - Box offsets: $\Delta_x = (x_G - x_P)/w_P, \Delta_x = (y_G - y_P)/h_P$

$$\Delta_w = \log(\frac{w_G}{w_P}),\ \Delta_h = \log\left(\frac{h_G}{h_P}\right)$$

where (x, y) = box centers and (w, h) = width and height

  - Pooling = [RoIPool or RoIAlign]

Regression Parameters in Proposed method

- Regression in this paper
  - Distance of each local feature: $p_i = (x_i, y_i)$
  - Top-left and bottom-right from GT: $(x_l, y_t), (x_r, y_b)$
  - Ground-truth offsets: $l_i = (x_i - x_l)/w_P, t_i = (y_i - y_t)/h_P$

$$r_i = (\mathrm{x_r} - \mathrm{x_i})/w_P,\ b_i = (y_b - y_i)/h_P$$

$$m_i = \begin{cases} 1, & if\ p_i \in G; \\ 0, & otherwise \end{cases}$$

$$\hat{m}_i = \{\hat{m}_i : i \in [1, k^2]\} \rightarrow \sigma(\hat{m}_i): sigmoid\ function$$

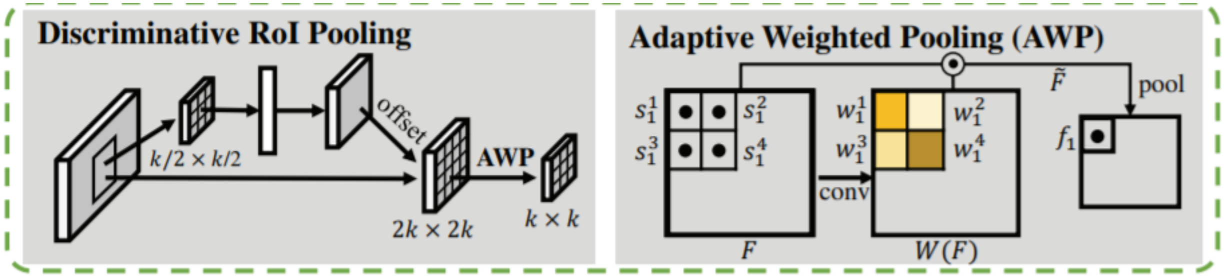
$\sigma(\hat{m}_i)$>0.5: ignore the predicted box

$, y_i)$

$$m_i = \{m_i : i \in [1, k^2]\}$$

$$k \times k: RoI\ pooling\ size, 7 \times 7$$

- To classify the object, it adopts the discriminative RoI pooling



(c) Discriminative RoI Pooling

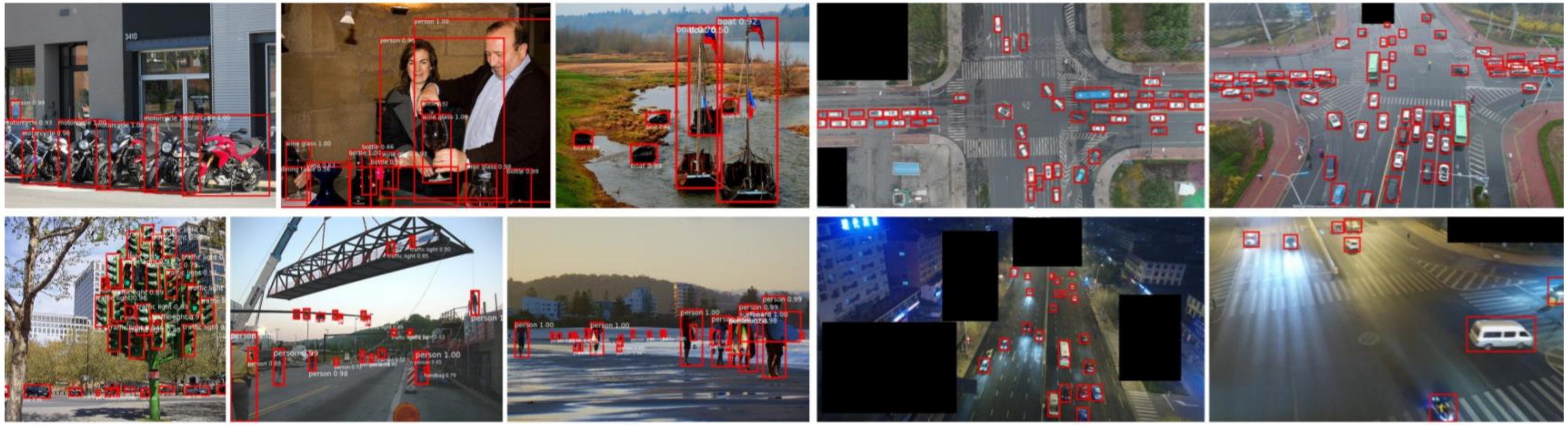
- To use the [illegible] $=\frac{k}{2} \times \frac{k}{2}$ and fully connected layer
- Weighted RoI feature

$$F \in R^{2k \times 2k}, \mathrm{W}(F) \in R^{2k \times 2k}$$

$$\tilde{F} = W(F) \odot F$$

$\odot$: Hadamard product(=element-wise product)

Experiment Results(2/2)



(a) MS COCO (b) UAVDT

DATA AUGMENTATION

- Improved Regularization of Convolutional Neural Networks with Cutout
- Area of cutout: zero-value -> similar with dropout

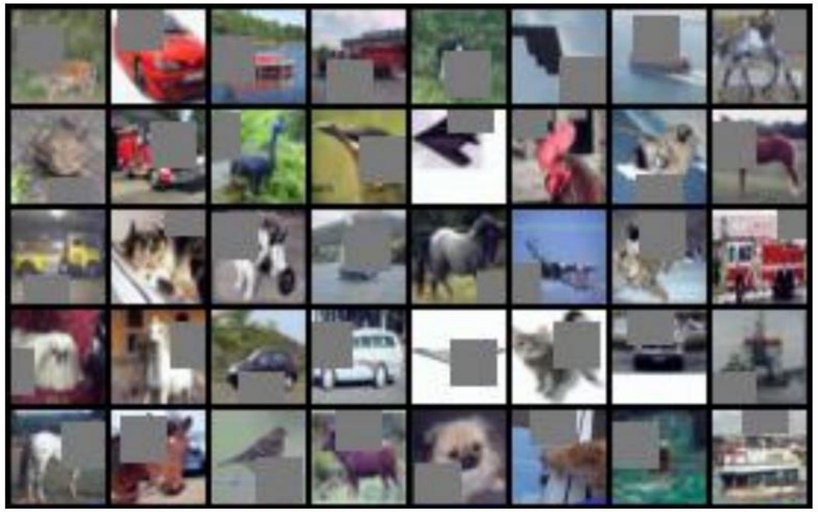
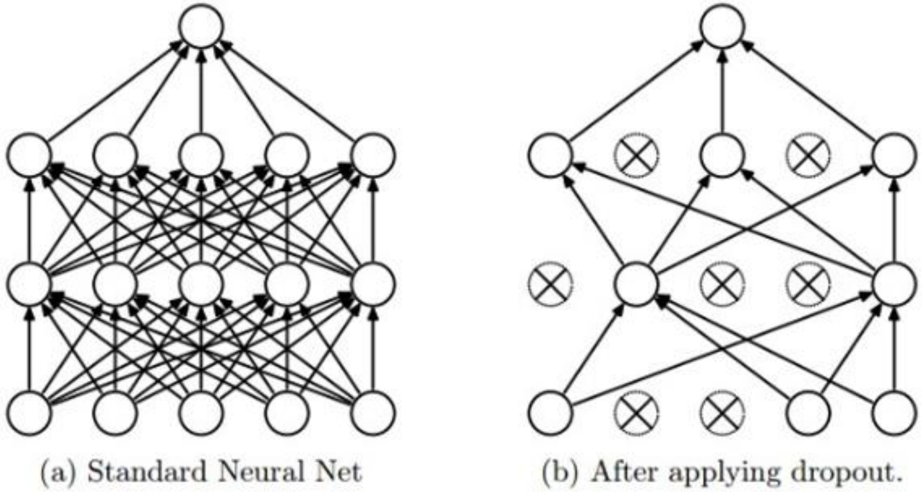


Figure 1: Cutout applied to images from the CIFAR-10 dataset.

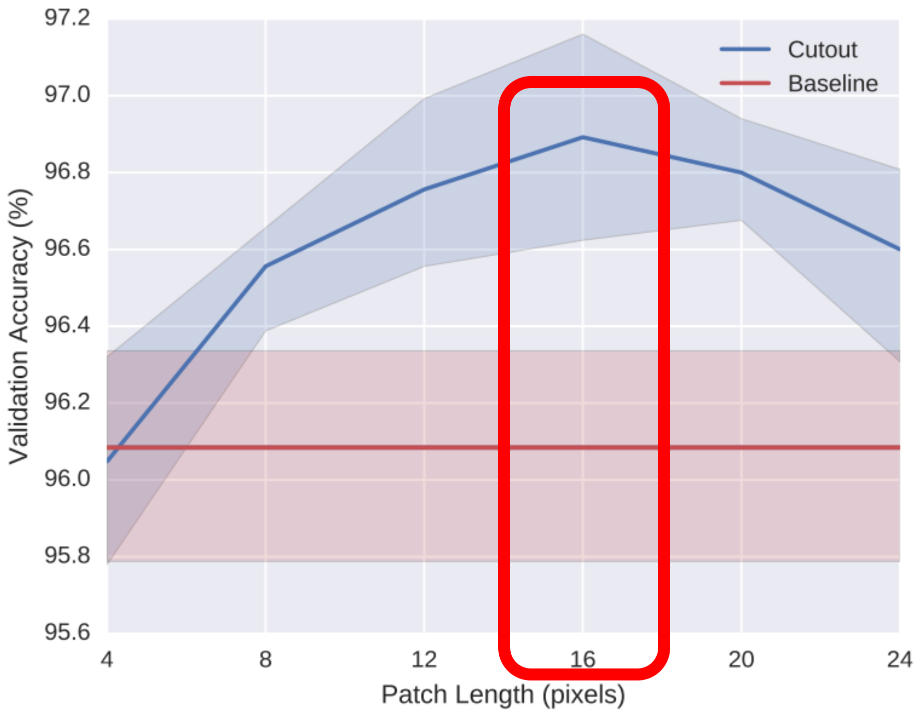


Dropout

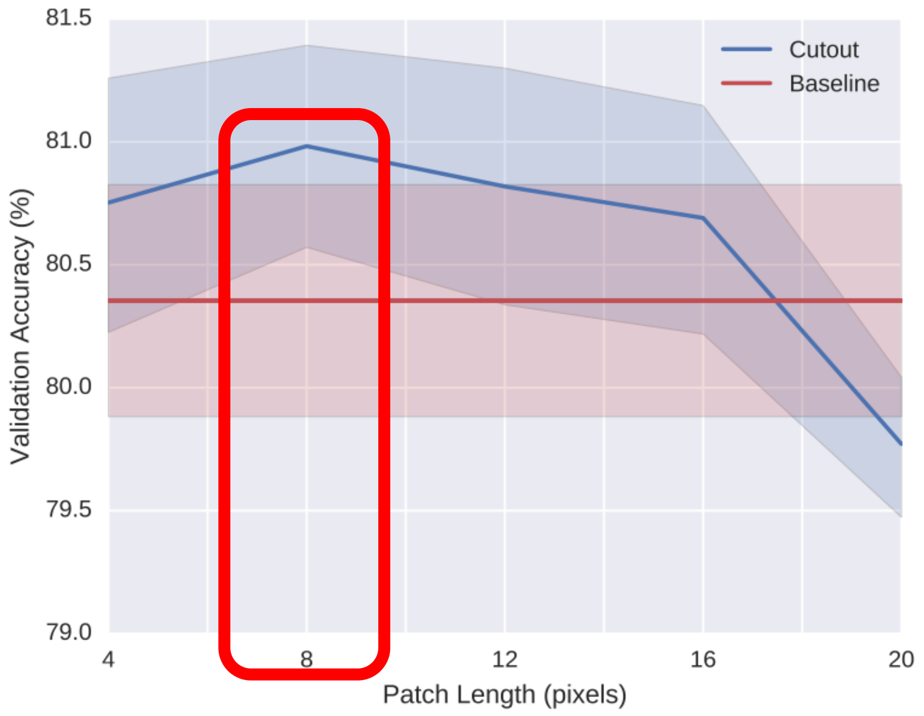
Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.

Augmentation: Cutout(2/2)

- Affecting the size of cutout area for the performance
- In this experiments, defaulted shape => square



(a) CIFAR-10

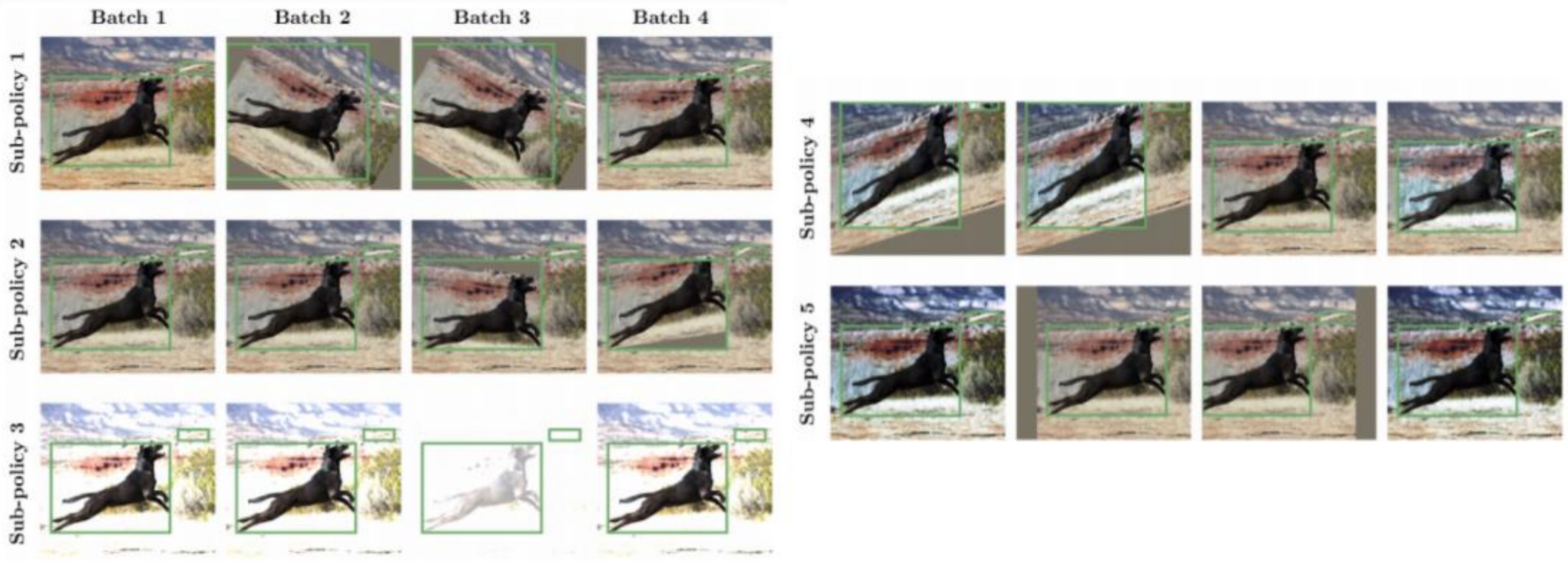


(b) CIFAR-100

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.

Search Space(Auto Augmentation)

- Search Space
  - Color Operations: Equalize, Contrast, Brightness, etc
  - Geometric Operations: Rotate, TranslationX, TranslationY, etc
  - Bounding Box Operations: Bbox_Only_Equalize, Bbox_Only_Rotate



Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection. In ECCV, 2020

- Image-level Augmentation
- Commonly used Image Pyramid
- Expensive computation for training original multi-scale
- Random crop for Zoom-In
- Change P, M for every iteration
- P(probability): [0, 0.1, 0.2, 0.3, 0.4, 0.5]
- M(magnitude): Zoom ratio function

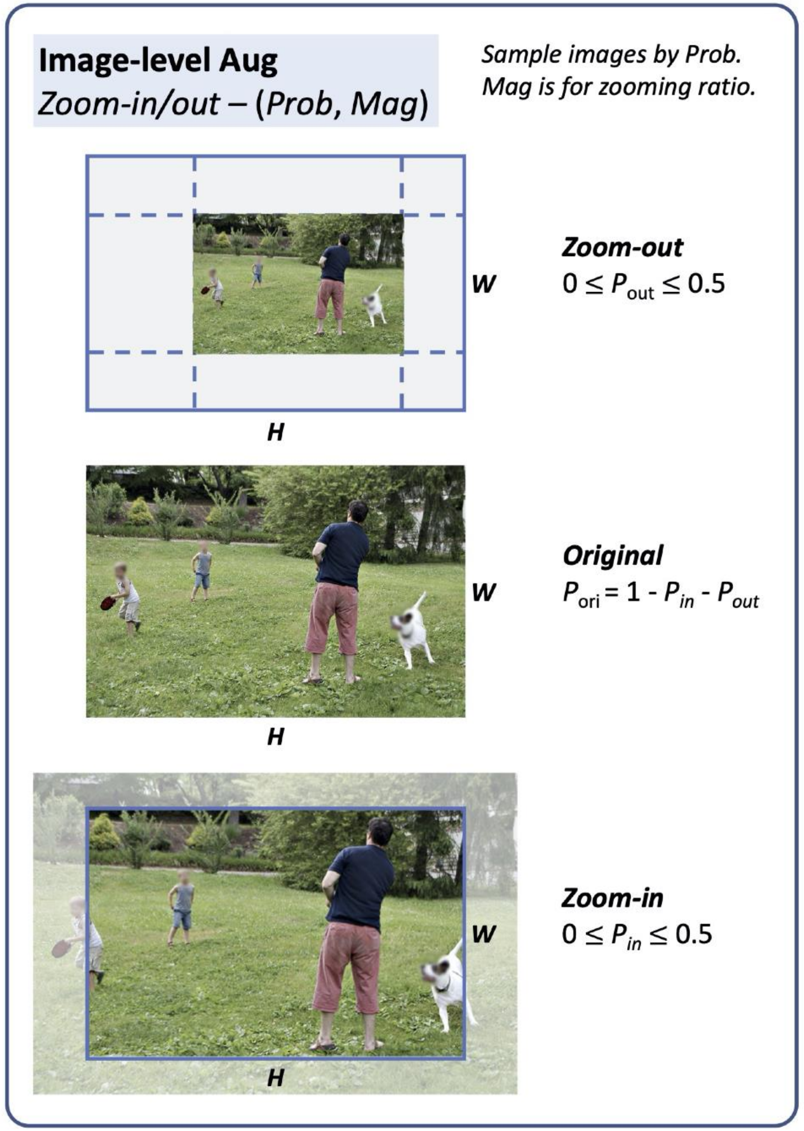
$$M_{\text{Zoom-In}} = [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$$

$$M_{\text{Zoom-Out}} = [1.0, 1.1, 1.2, 1.3, 1.4, 1.5]$$

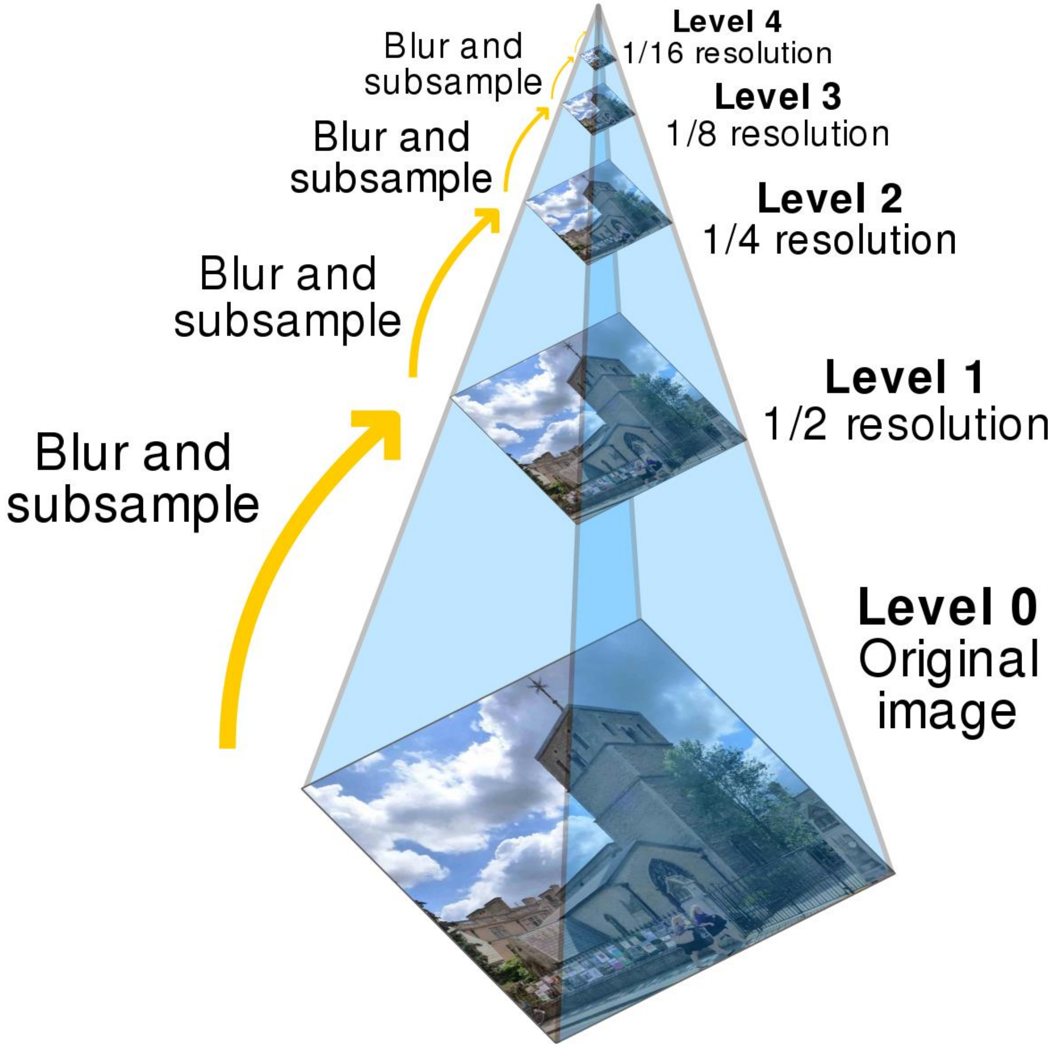
$$P = \{P_{In}, P_{out}, P_{ori}\}$$

- Total:

$$10 \text{ images} = 3, P_{In} + 4, P_{Out} + 3, P_{ori}$$



- Downscale for every level



Scale-aware Search Space(Box-level Augmentation)

- Box-level Augmentation
  - Conducted augmentation for each object box
  - Different from box object area, used gaussian map for each object
  - Blended original and transformed pixels with spatial wise Gaussian map
  - Affected context information for detection result
    - Removed background pixels
    - Drop the AP_s, 25.2->18.0

Table 1: Analysis on the context for scales. On well-trained ResNet-101 detectors, AP_s drops and AP_l increases consistently if contexts are removed in validation images.

	with context	AP	AP_s	AP_m	AP_l
Faster R-CNN	✓	41.4	25.2	44.8	53.0
	✗	40.5	18.0	45.7	56.1
	Δ	-0.9	**-7.2**	+0.9	**+3.1**
RetinaNet	✓	40.3	23.3	44.0	53.3
	✗	39.8	16.7	44.4	57.7
	Δ	-0.5	**-6.6**	+0.4	**+4.4**

- Box-level Augmentation
  - Augment area adaptive to object sizes, area ratio
  - Blended original and transformed pixels with spatial wise Gaussian map

$$\alpha(x,y) = \exp\left(-\left(\frac{(x-x_c)^2}{2\sigma_x^2} + \frac{(y-y_c)^2}{2\sigma_y^2}\right)\right)$$

  - A: augmented region

$$A = \alpha(x,y) \cdot I + (1-\alpha(x,y)) \cdot T$$

  - I: input, T: transformation function
  - V: area for gaussian map, HxW: image height x width

$$V = \int_0^H \int_0^W \alpha(x,y)\,dx\,dy \approx 2\pi\sigma_x\sigma_y \qquad \sigma_x = h\sqrt{\frac{W/H}{2\pi}r} \qquad \sigma_y = w\sqrt{\frac{H/W}{2\pi}r}$$

Box-level Aug
Color/Geometric – (Prob, Mag, Area)

Large object
e.g. $r(s_{box}) < 1$

Middle object
e.g. $r(s_{box}) > 1$

Small object
e.g. $r(s_{box}) > 1$

Bounding box ***Augmented area***

Box-level Aug policy contains Color and Geometric operations.

	Aug types	***Prob.***	***Mag.***	***Area ratio***
Color	*Brightness*	P_1	M_1	$r(s_{box})$
	Color	P_2	M_2	
	Contrast	P_3	M_3	
	Cutout	P_4	M_4	
	Equalize	P_5	M_5	
	Sharpness	P_6	M_6	
	Solarize	P_7	M_7	
	SolarizeAdd	P_8	M_8	
Geometric	*Hflip*	P_9	M_9	$r(s_{box})$
	Rotate	P_{10}	M_{10}	
	ShearX	P_{11}	M_{11}	
	ShearY	P_{12}	M_{12}	
	TranslateX	P_{13}	M_{13}	
	TranslateY	P_{14}	M_{14}	

$r(s_{box})$ is the ratio of aug area and box size. It varies for different scale of objects.

Scale-aware Box-level Aug

Scale-aware Search Space(Box-level Augmentation)

- Box-level Augmentation
  - 8 color operations and 6 geometric operations
    - Probability range: 6 discrete values, [0, 0.2, 0.4, 0.6, 0.8, 1.0]
    - Magnitude range: 6 discrete values, [0, 2, 4, 6, 8, 10]
  - Area ratio
    - Area ratio type: small, middle, large
    - Area ratio range: 10 discrete values [0.2, 0.4, 0.6, 0.8, 1.0, 2, 4, 6, 8, 10]
  - Total candidate policies box(5 policies), image(1 policies)
    - Image-level augmentation 6 ** 4
      - Zoom-in: 6*6
      - Zoom-out: 6*6
    - Box-level augmentation (10 ** 3) * ((8 * 6 * 6) * (6 * 6 * 6)) ** 5
      - Area ratio: 10*10*10
      - Color operations: 8*6*6
      - Geometric operations: 6*6*6

$$(6^2)^2 \times \left(((6 \times 6^2) \times (8 \times 6^2))^5 \times 10^3\right) = 1.2 \times 10^{30}$$

Scale-aware Search Space(Box-level Augmentation)

- Compared with Auto Augmentation 2 times combinations

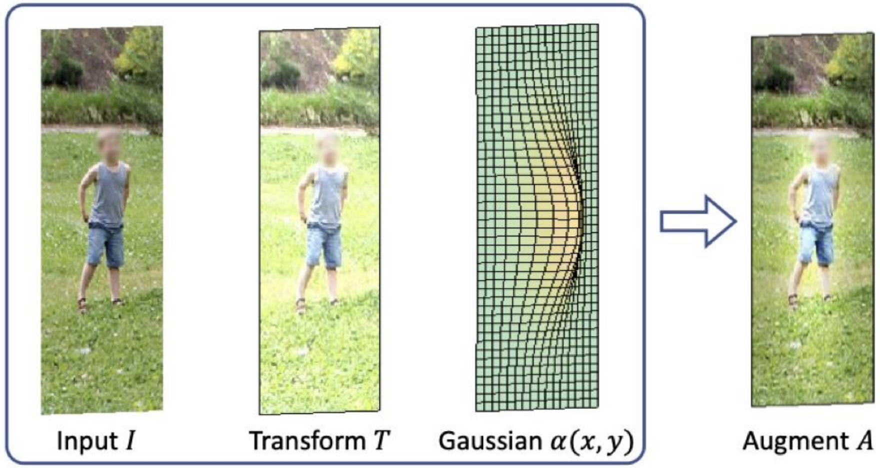
$$(22 \times 6 \times 6)^{2\times 5} \approx 9.6 \times 10^{28}$$

- Total candidate policies box(5 policies), image(1 policies)
  - Image-level augmentation 6 ** 4
    - Zoom-in: 6*6
    - Zoom-out: 6*6
  - Box-level augmentation (10 ** 3) * ((8 * 6 * 6) * (6 * 6 * 6)) ** 5
    - Area ratio: 10*10*10
    - Color operations: 8*6*6
    - Geometric operations: 6*6*6

$$(6^2)^2 \times \left(((6 \times 6^2) \times (8 \times 6^2))^5 \times 10^3\right) = 1.2 \times 10^{30}$$

Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection. In ECCV, 2020

Scale-aware Search Space(Box-level Augmentation)



(a) Comparison between square and gaussian transform.

(b) Gaussian-based transform process.

Figure 3: An example of Gaussian-based box-level augmentation. It removes the original hard boundary and the augmented areas are adjustable to the Gaussian variance.

Scale-aware Search Space(Box-level Augmentation)

Table A - 3. Details about box-level operations with their description and magnitude ranges.

Operation	Description	Magnitude range
Brightness	Control the object brightness. Magnitude = 0 represents the black, while magnitude = 1.0 means the original.	[0.1, 1.9]
Color	Control the color balance. Magnitude = 0 represents a black & white object, while magnitude = 1.0 means the original.	[0.1, 1.9]
Contrast	Control the contrast of the object. Magnitude = 0 represents a gray object, while magnitude = 1.0 means the original object.	[0.1, 1.9]
Cutout	Randomly set a square area of pixels to be gray. Magnitude represents the side length.	[0, 60]
Equalize	Equalize the histogram of the object area.	-
Sharpness	Control the sharpness of the object. Magnitude = 0 represents a blurred object, while magnitude = 1.0 means the original object.	[0.1, 1.9]
Solarize	Invert all pixels above a threshold value. Magnitude represents the threshold.	[0,256]
SolarizeAdd	For pixels less than 128, add an amount to them. Magnitude represents the amount.	[0,110]
Hflip	Flip the object horizontally.	-
Rotate	Rotate the object to a degree. Magnitude represents the degree.	[-30,30]
ShearX/Y	Shear the object along the horizontal or vertical axis with a magnitude.	[-0.3, 0.3]
TranslateX/Y	Translate the object in the horizontal or vertical direction by magnitude pixels.	[-150, 150]

Scale-aware Search Space(Box-level Augmentation)

Figure A - 2. Examples on different box-level operations with magnitudes random sampled.

Scale-aware Search Space(Box-level Augmentation)

(a) Original image

(b) Image with context pixels removed

Figure A - 3. An example image of removing context.

Table 9: Searched augmentation policy.

Image-level	(Zoom-in, 0.2, 4)	(Zoom-out, 0.4, 10)
Box-level	*Color operations*	*Geometric operations*
Sub-policy 1.	(Color, 0.4, 2)	(TranslateX, 0.4, 4)
Sub-policy 2.	(Brightness, 0.2, 4)	(Rotate, 0.4, 2)
Sub-policy 3.	(Sharpness, 0.4, 2)	(ShearX, 0.2, 6)
Sub-policy 4.	(SolarizeAdd, 0.2, 2)	(Hflip, 0.3, 0)
Sub-policy 5.	Original	(TranslateY, 0.2, 8)
Area ratio	Small - 6 Middle - 2 Large - 0.4	

Scale-aware Estimation Metric

- In common, adaptive random augmentation in dataset
  - Evaluated with accuracy for object detection
- In this paper, balanced optimization over different scales
  - Evaluated with accumulated loss and accuracy

- MS(Multi-scale) training baseline
- Model: Faster R-CNN, RetinaNet, FCOS, Mask R-CNN

Table 2: Improvement details on RetinaNet ResNet-50.

	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
MS Baseline	38.2	57.3	40.5	23.0	41.6	50.3
Ours image-level	40.1	59.8	43.3	24.0	44.1	53.1
+ box-level	40.6	60.4	43.6	24.1	44.4	53.5
+ scale-aware area	**41.3**	**61.0**	**44.1**	**25.2**	**44.5**	**54.6**

Table 5: Improvements across detection frameworks.

Models	*policy*	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
RetinaNet:							
ResNet-50[3]	Baseline	36.6	55.7	39.1	20.8	40.2	49.4
	MS Baseline	38.2	57.3	40.5	23.0	41.6	50.3
	Ours	**41.3**	61.0	44.1	25.2	44.5	54.6
ResNet-101	Baseline	38.8	59.1	42.3	21.8	42.7	50.2
	MS Baseline	40.3	59.8	42.9	23.2	44.0	53.2
	Ours	**43.1**	62.8	46.0	26.2	46.8	56.7
Faster R-CNN:							
ResNet-50	Baseline	37.6	57.8	41.0	22.2	39.9	48.4
	MS Baseline	39.1	60.8	42.6	24.1	42.3	50.3
	Ours	**41.8**	63.3	45.7	26.2	44.7	54.1
ResNet-101	Baseline	39.8	61.3	43.5	23.1	43.2	52.3
	MS Baseline	41.4	60.4	44.8	25.0	45.5	53.1
	Ours	**44.2**	65.6	48.6	29.4	47.9	56.7
FCOS:							
ResNet-50	MS Baseline	40.8	59.6	43.9	26.2	44.9	51.9
	Ours	**42.6**	61.2	46.0	28.2	46.4	54.3
ResNet-101	MS Baseline	41.8	60.3	45.3	25.6	47.7	56.1
	Ours	**44.0**	62.7	47.3	28.2	47.8	56.1

Table 6: Improvements across tasks on Mask R-CNN.

Models	*policy*	$AP^{m/k}$	$AP^{m/k}_{50}$	$AP^{m/k}_{75}$	AP^{b}	AP^{b}_{50}	AP^{b}_{75}
Instance Segmentation:							
ResNet-50	MS Baseline	36.4	58.8	38.7	40.4	61.9	44.0
	Ours	**38.1**	60.9	40.8	**42.8**	64.4	46.9
ResNet-101	MS Baseline	37.9	60.4	40.4	42.3	63.8	46.6
	Ours	**40.0**	63.2	42.9	**45.3**	66.4	49.8
Keypoint Estimation:							
ResNet-50	MS Baseline	64.1	85.9	69.7	53.5	82.7	58.4
	Ours	**65.7**	86.6	71.7	**55.5**	84.2	60.9
ResNet-101	MS Baseline	65.1	86.5	71.2	54.8	83.2	60.0
	Ours	**66.4**	87.5	72.7	**56.5**	84.6	62.1

Experiment

Table 8: Comparison with state-of-the-art data augmentation methods for object detection.

Method	Detector	Backbone	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Hand-crafted:								
Dropblock [18]	RetinaNet	ResNet-50	38.4	56.4	41.2	-	-	-
Mix-up [49]	Faster R-CNN	ResNet-101	41.1	-	-	-	-	-
PSIS* [45]	Faster R-CNN	ResNet-101	40.2	61.1	44.2	22.3	45.7	51.6
Stitcher [8]	Faster R-CNN	ResNet-101	42.1	-	-	26.9	45.5	54.1
GridMask [6]	Faster R-CNN	ResNeXt-101	42.6	65.0	46.5	-	-	-
InstaBoost* [17]	Mask R-CNN	ResNet-101	43.0	64.3	47.2	24.8	45.9	54.6
SNIP (MS test)* [41]	Faster R-CNN	ResNet-101-DCN-C4	44.4	66.2	49.9	27.3	47.4	56.9
SNIPER (MS test)* [42]	Faster R-CNN	ResNet-101-DCN-C4	46.1	67.0	51.6	29.6	48.9	58.1
Automatic:								
AutoAug-det [51]	RetinaNet	ResNet-50	39.0	-	-	-	-	-
AutoAug-det [51]	RetinaNet	ResNet-101	40.4	-	-	-	-	-
AutoAug-det† [51]	RetinaNet	ResNet-50	40.3	60.0	43.0	23.6	43.9	53.8
AutoAug-det† [51]	RetinaNet	ResNet-101	41.8	61.5	44.8	24.4	45.9	55.9
RandAug [10]	RetinaNet	ResNet-101	40.1	-	-	-	-	-
RandAug† [10]	RetinaNet	ResNet-101	41.4	61.4	44.5	25.0	45.4	54.2
Ours:								
Scale-aware AutoAug	RetinaNet	ResNet-50	41.3	61.0	44.1	25.2	44.5	54.6
Scale-aware AutoAug	RetinaNet	ResNet-101	43.1	62.8	46.0	26.2	46.8	56.7
Scale-aware AutoAug	Faster R-CNN	ResNet-101	44.2	65.6	48.6	29.4	47.9	56.7
Scale-aware AutoAug (MS test)	Faster R-CNN	ResNet-101-DCN-C4	47.0	68.6	52.1	32.3	49.3	60.4
Scale-aware AutoAug	FCOS	ResNet-101	44.0	62.7	47.3	28.2	47.8	56.1
Scale-aware AutoAug	FCOS‡	ResNeXt-32x8d-101-DCN	48.5	67.2	52.8	31.5	51.9	63.0
Scale-aware AutoAug (1200 size)	FCOS‡	ResNeXt-32x8d-101-DCN	49.6	68.5	54.1	35.7	52.5	62.4
Scale-aware AutoAug (MS test)	FCOS‡	ResNeXt-32x8d-101-DCN	**51.4**	69.6	57.0	37.4	54.2	65.1

Experiment

- Scale invariant for using multi-scale training
- Better performance for using Scale aware Auto Augmentation

Table 10: Scale variation issue on a clean Faster R-CNN.

	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
ResNet-50-C4	34.7	55.7	37.1	18.2	38.8	48.3
with MS train	34.8	55.6	37.3	18.9	39.2	47.6
with FPN	36.7	58.4	39.6	21.1	39.8	48.1
with Ours	**36.8**	58.0	39.5	21.0	41.2	49.1

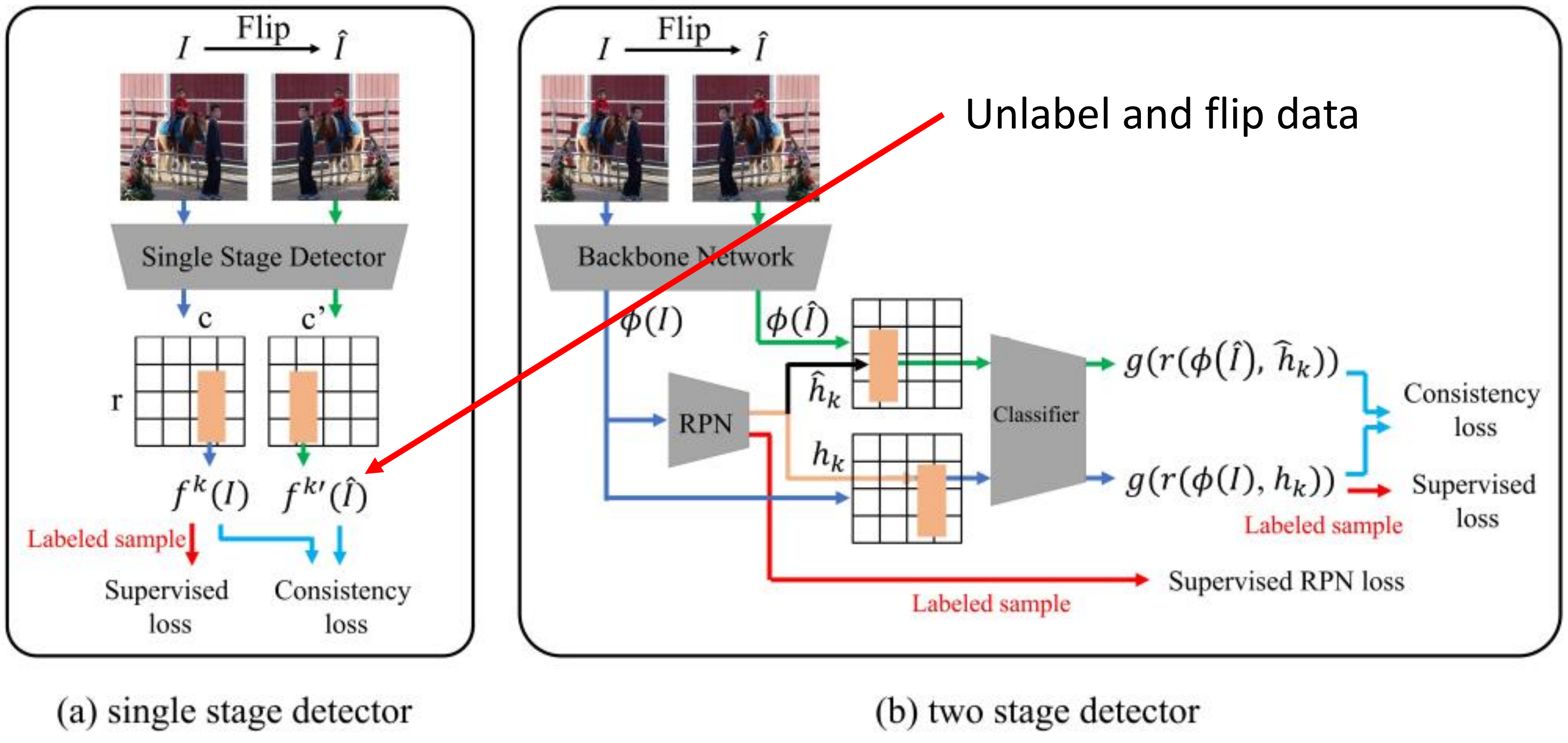
SEMI-SUPERVISED LEARNING

- Semi-supervised learning
- Using horizontally flip image as unlabel data, to improve consistency regularization(CR)

Consistency-based Semi-supervised Learning for Object Detection

Jisoo Jeong*, Seungeui Lee*, Jeesoo Kim & Nojun Kwak
Department of Transdisciplinary Studies
Graduate School of Convergence Science and Technology
Seoul National University
Seoul, Korea
{soo3553, seungeui.lee, kimjiss0305, nojunk}@snu.ac.kr

Jisoo Jeong, Seungeui Lee, Jeesoo Kim, and Nojun Kwak. Consistency-based semi-supervised learning for object detection. In Advances in Neural Information Processing Systems, pages 10759–10768, 2019.

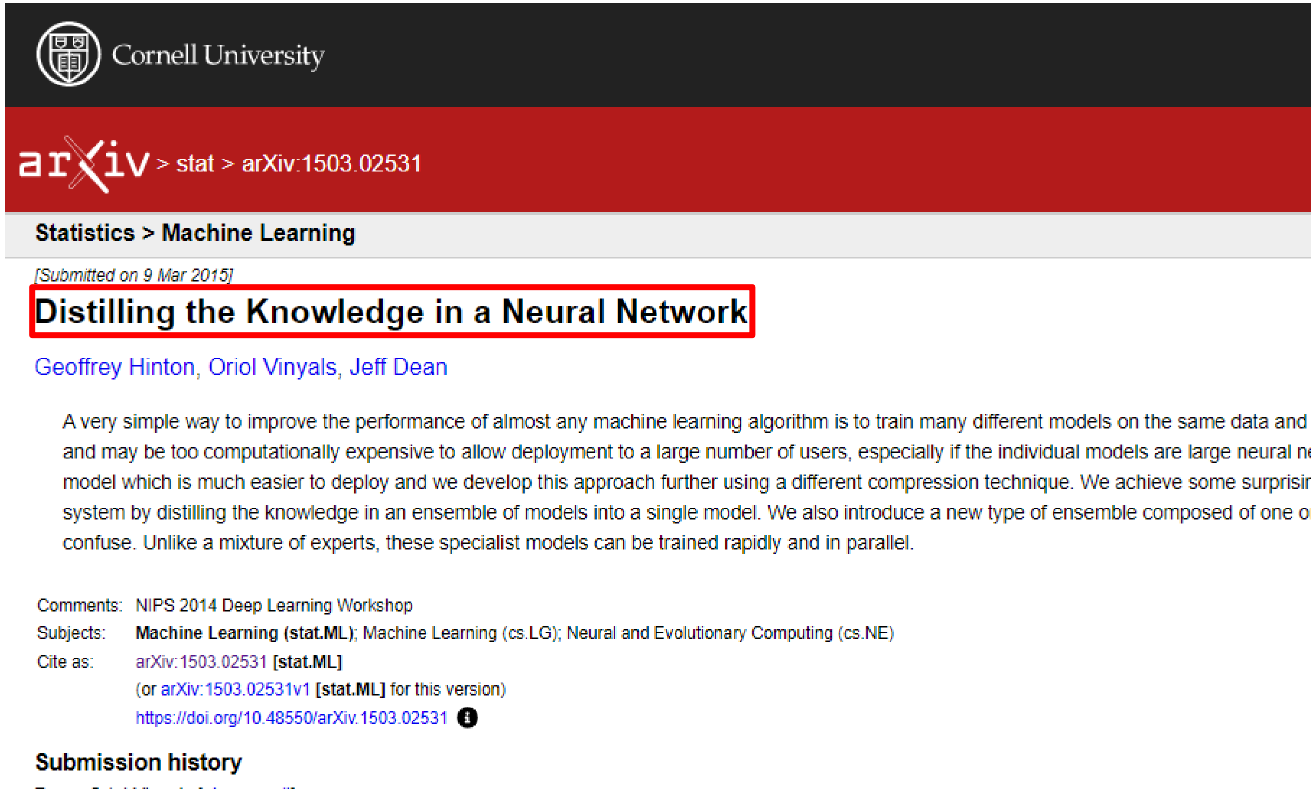


(a) single stage detector

(b) two stage detector

Figure 2: Overall structure of our proposed method. (a) $f^k(I)$ and $f^{k'}(\hat{I})$ are extracted by a single stage detector from image I and flipped image $\hat{I}$ respectively. The supervised loss is computed between $f^k(I)$ and the ground truth for labeled data and the consistency loss is computed between $f^k(I)$ and $f^{k'}(\hat{I})$ for labeled and unlabeled data. (b) $\phi(I)$ and $\phi(\hat{I})$ originate from the backbone network and the RoI is computed only from $\phi(I)$. $\hat{h}_k$ is obtained by flipping h_k to associate two corresponding boxes and supervised and consistency losses are calculated in the same way as for the single stage detector.

- Googleplex(Google+complex) in Mountain View, California
- Edward Kasner, America Mathematician
- Googleplex from Googolplex $10^{100} \rightarrow 10^{googol}$


Cornell University

arXiv > stat > arXiv:1503.02531

Statistics > Machine Learning

[Submitted on 9 Mar 2015]

Distilling the Knowledge in a Neural Network

Geoffrey Hinton, Oriol Vinyals, Jeff Dean

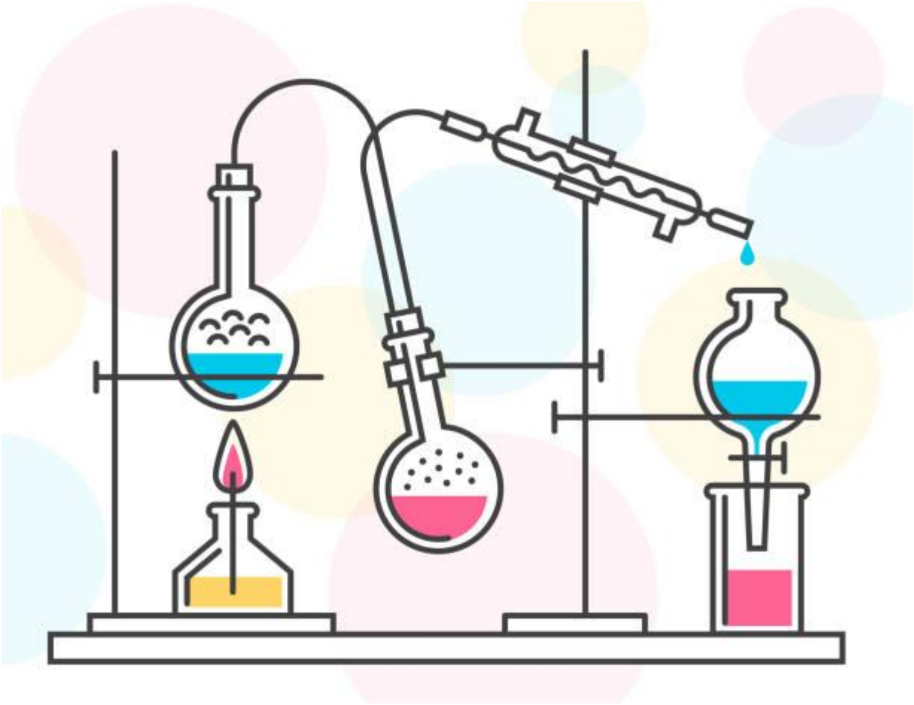
A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and
and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural ne
model which is much easier to deploy and we develop this approach further using a different compression technique. We achieve some surprisir
system by distilling the knowledge in an ensemble of models into a single model. We also introduce a new type of ensemble composed of one o
confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

Comments:	NIPS 2014 Deep Learning Workshop
Subjects:	**Machine Learning (stat.ML)**; Machine Learning (cs.LG); Neural and Evolutionary Computing (cs.NE)
Cite as:	arXiv:1503.02531 **[stat.ML]**
	(or arXiv:1503.02531v1 **[stat.ML]** for this version)
	https://doi.org/10.48550/arXiv.1503.02531

Submission history

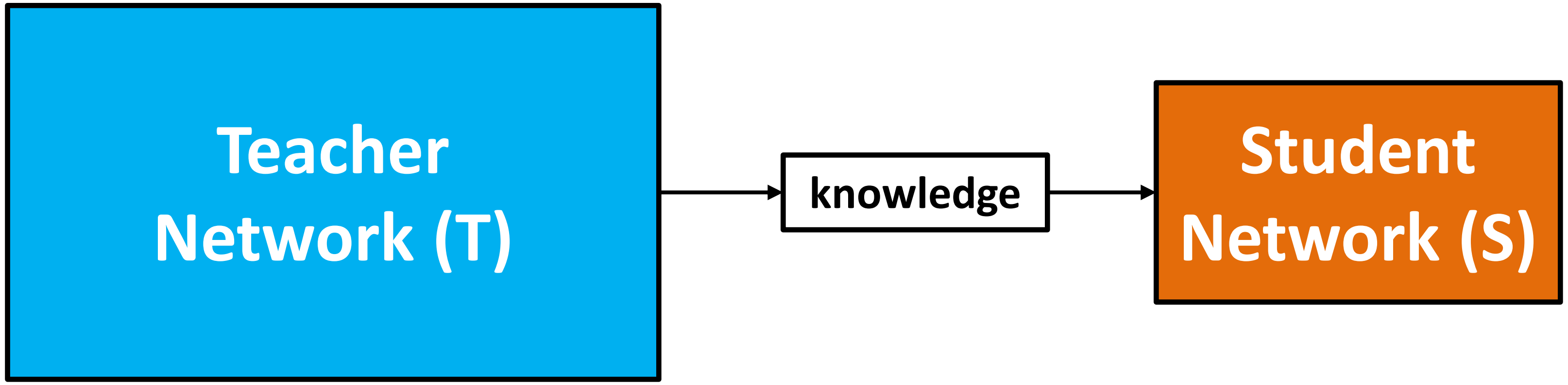
Geoffrey Hinton, Oriol Vinyals, Jeff Dean, “Distilling the Knowledge in a Neural Network”, NIPS, 2014, arXiv:1503.02531

- Contribution
  - Complex model(T): acc -> 99%, inference time -> 3 hours
  - Simple model(S): acc -> 90%, inference time -> 3 mins



Geoffrey Hinton, Oriol Vinyals, Jeff Dean, “Distilling the Knowledge in a Neural Network”, NIPS, 2014, arXiv:1503.02531

- Contribution
  - Teacher Network(T)
    - Cumbersome model: ensemble / a large generalized model
    - (pros) excellent performance
    - (cons) computationally expensive
  - Student Network(S)
    - Small model
    - (pros) fast inference
    - (cons) lower performance than Teacher Network



Geoffrey Hinton, Oriol Vinyals, Jeff Dean, “Distilling the Knowledge in a Neural Network”, NIPS, 2014, arXiv:1503.02531

Knowledge Distillation

- Already predicted the dog from model
- But still room to study from other results
- Room to learn from the soft label than hard label
- Possible to figure the difference out between cat and (cow and car)

$$q_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

cow	dog	cat	car	
0	1	0	0	Original label =hard label

cow	dog	cat	car	
10^{-6}	0.9	0.1	10^{-9}	Output softmax =soft label

※ difficult to study with small value

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, "Distilling the Knowledge in a Neural Network", NIPS, 2014, arXiv:1503.02531

- Enlarge the softmax value with T(temperature)
  - If T is getting large, easy to understand the value
  - **Because of T, called distillation**

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

cow	dog	cat	car
10^{-6}	0.9	0.1	10^{-9}

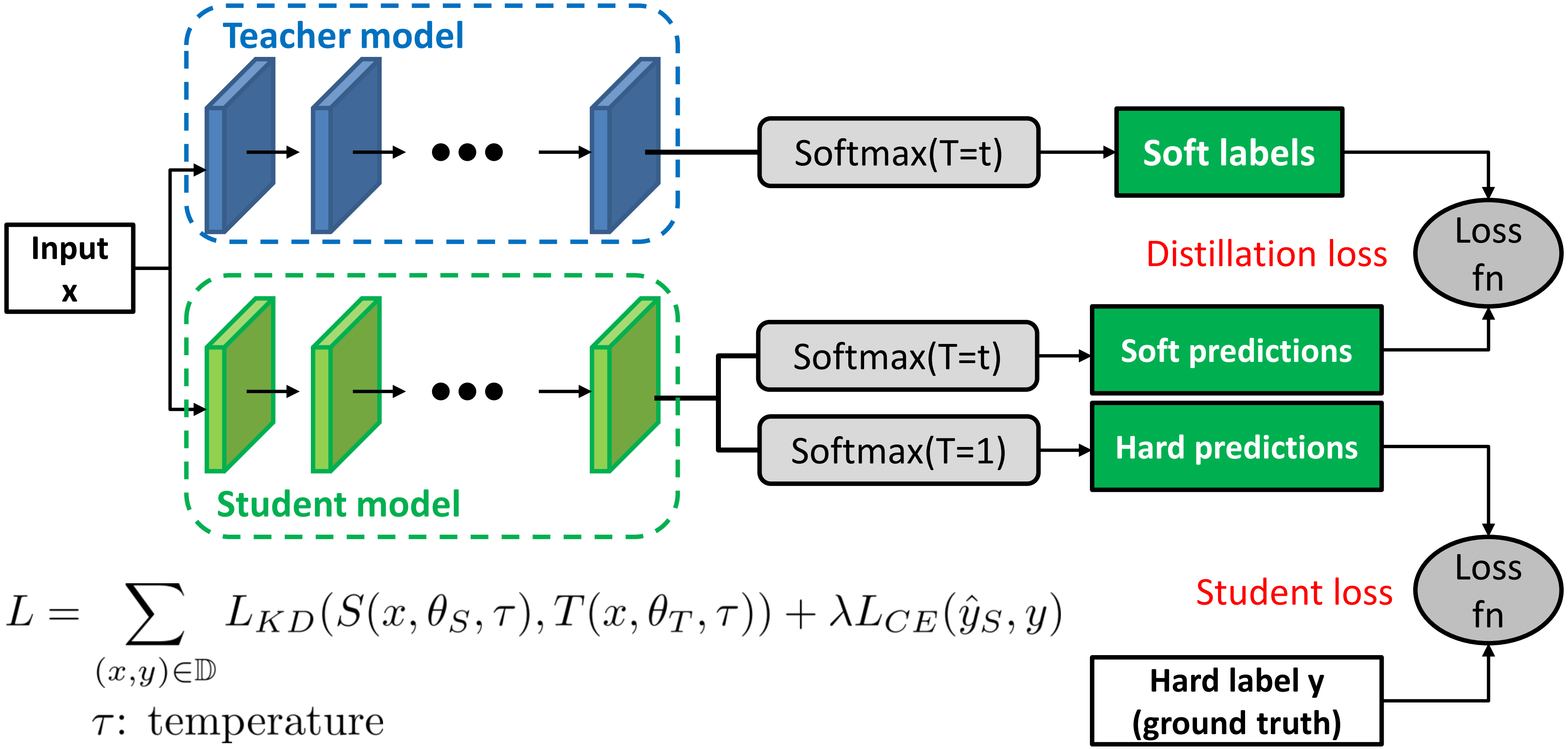
Output softmax
=soft label

cow	dog	cat	car
0.05	0.3	0.2	0.005

Softened output
softmax
=soft label

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, "Distilling the Knowledge in a Neural Network", NIPS, 2014, arXiv:1503.02531

- Distillation loss(how to update Teacher -> Student)



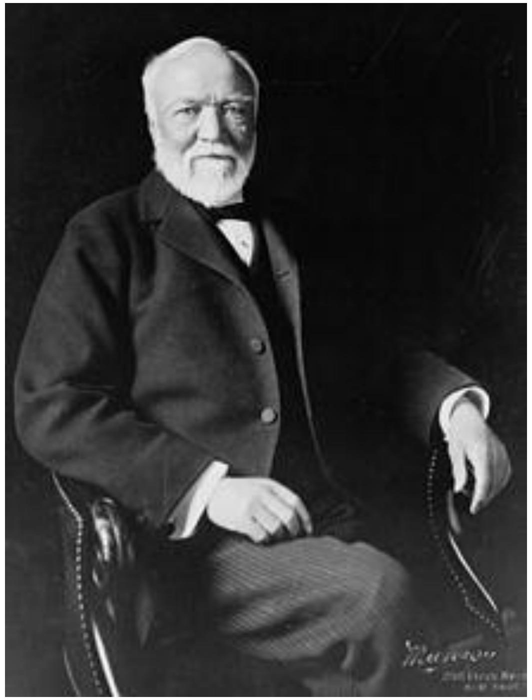
$$L = \sum_{(x,y)\in\mathbb{D}} L_{KD}(S(x,\theta_S,\tau), T(x,\theta_T,\tau)) + \lambda L_{CE}(\hat{y}_S, y)$$

τ: temperature

θ: parameters

UNIVERSITY

- Founded in 1900, Carnegie Technical Schools
- Founder: Andrew Carnegie(Nov 25th, 1835 – Aug 11th, 1919)
  - Steel industry / richest Americans in history
- In 1967, Carnegie Institute of Technology and the Mellon Institute of Industrial Research



Andrew Carnegie in 1913