Computer Vision

Annotation data efficient learning

Tae-Hyun Oh (오태현)
전자전기공학과
POSTECH

Slide by Jongha Kim (김종하)
TAs: {Dongmin Choi , Jongha Kim, Juyong Lee, Sungbin Kim} (in alphabetic order)

boostcamp AI Tech

1. Data augmentation

1.1 Learning representation of dataset
1.2 Data augmentation
1.3 Various data augmentation methods
1.4 Modern augmentation techniques

2. Leveraging pre-trained information

2.1 Transfer learning
2.2 Knowledge distillation

3. Leveraging unlabeled dataset for training

3.1 Semi-supervised learning
3.2 Self-training

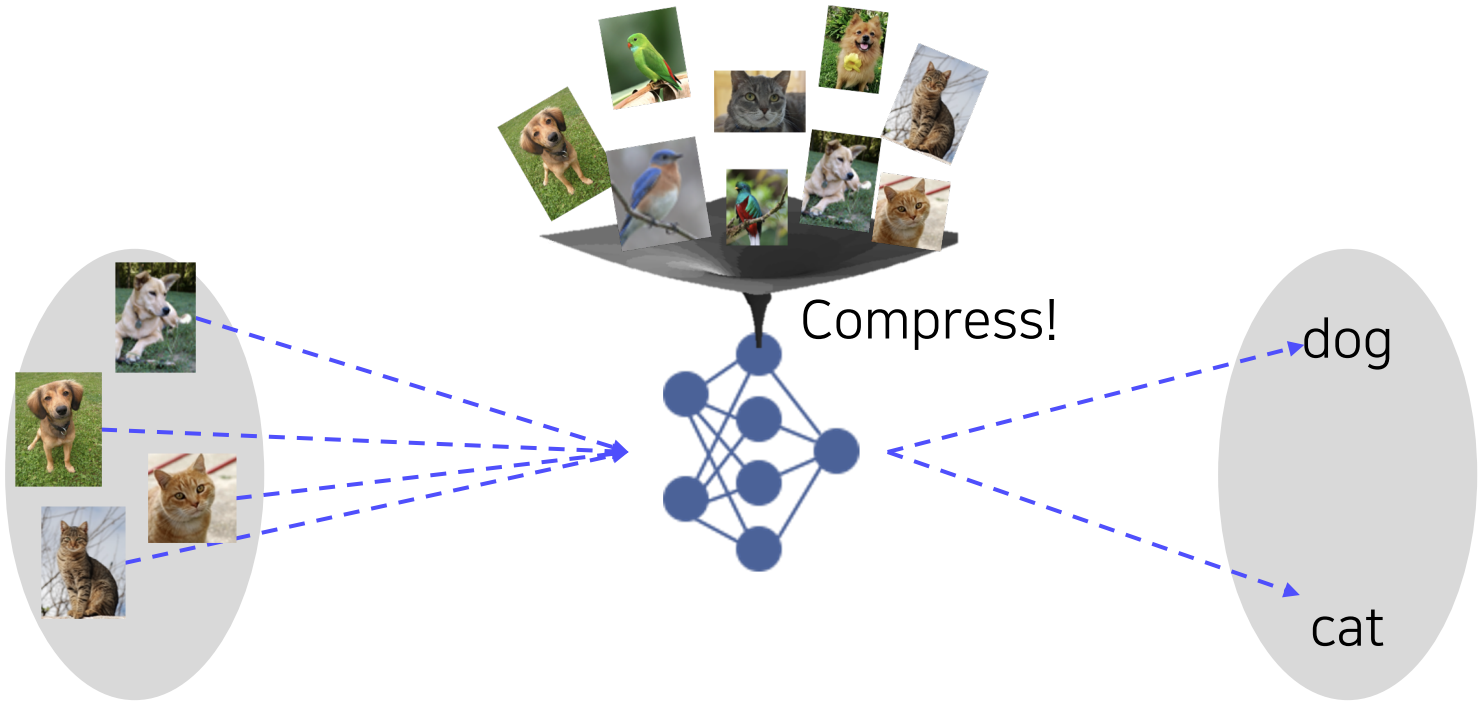
1.

Data augmentation

1.1 Learning representation of dataset

Learning representation from a dataset

- Neural networks learn compact features (information) of a dataset

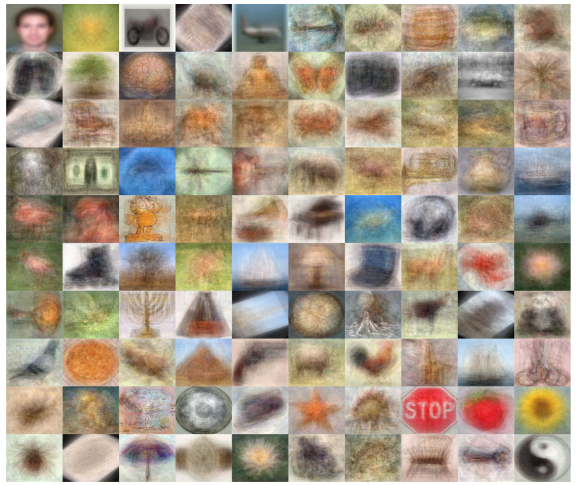


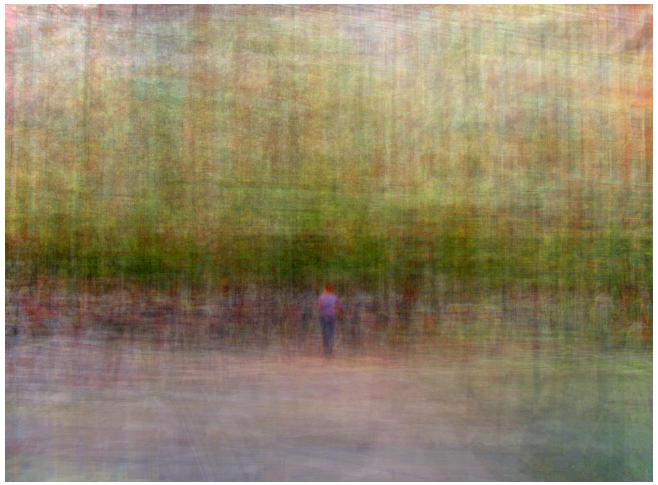
[https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html]

1.1 Learning representation of dataset

Dataset is (almost) always biased

- Images taken by camera (training data) ≠ real data





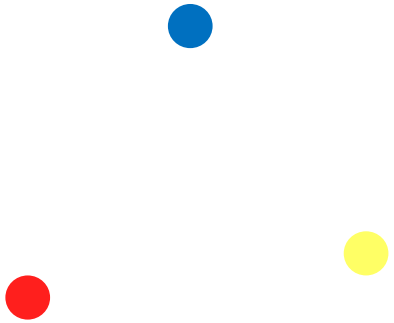
Mean images of each theme

[https://groups.csail.mit.edu/vision/torralbalab/]

1.1 Learning representation of dataset

The training dataset is sparse samples of real data

- The training dataset contains only fractional part of real data



Samples in the training set

Real data distribution

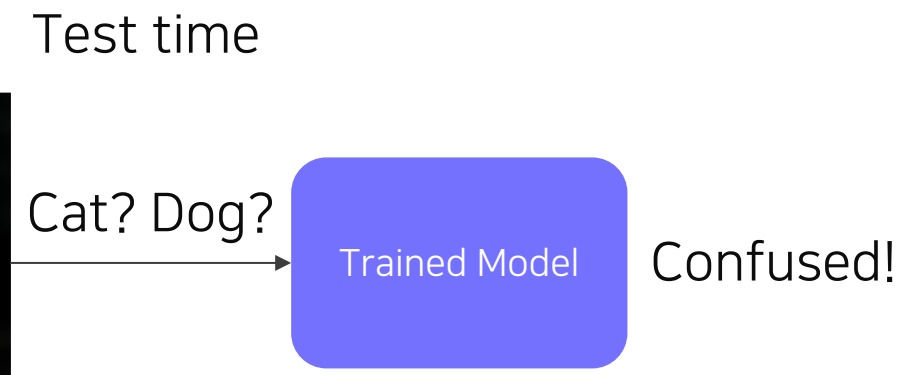
1.1 Learning representation of dataset

The training dataset and real data always have a gap

- Suppose a training dataset has only bright images
- During test time, if a dark image is fed as input, the trained model may be confused
- Problem: Datasets do not fully represent real data distribution!

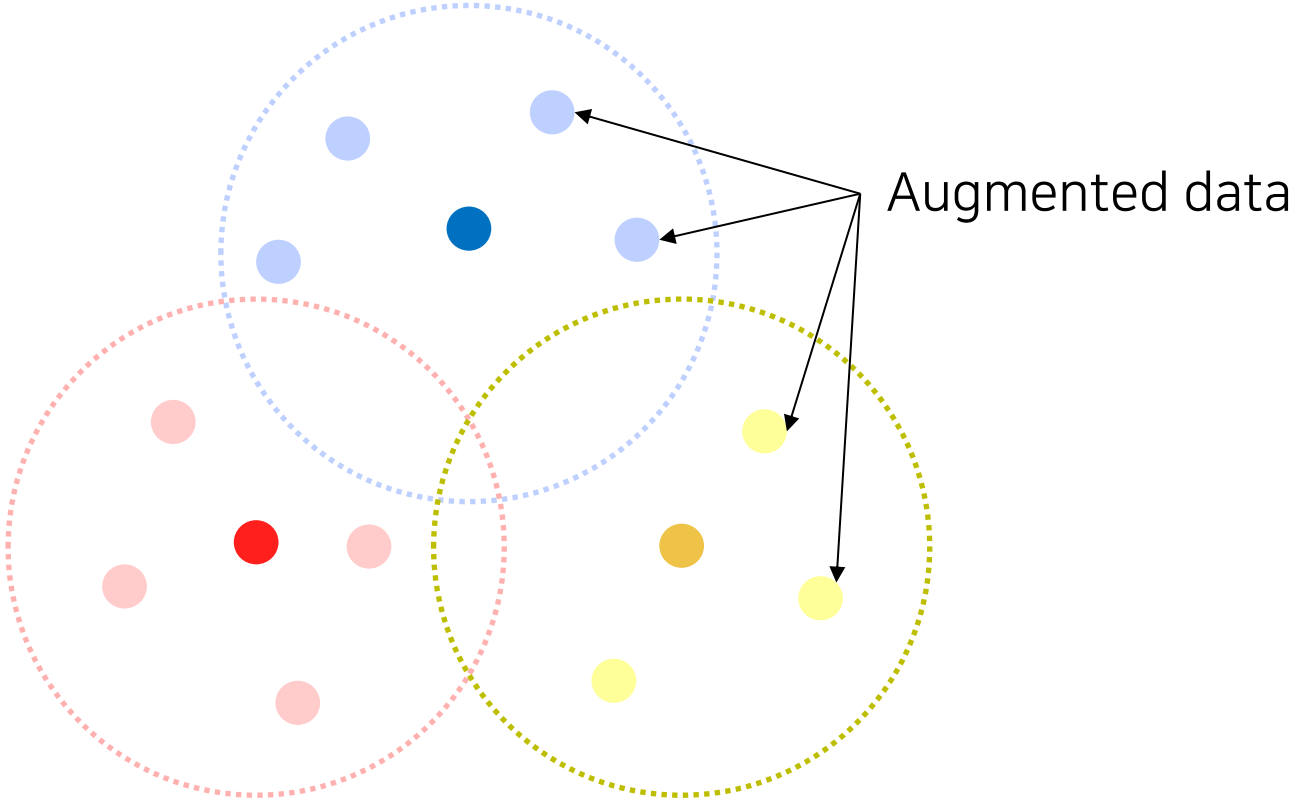
Train dataset

Input image



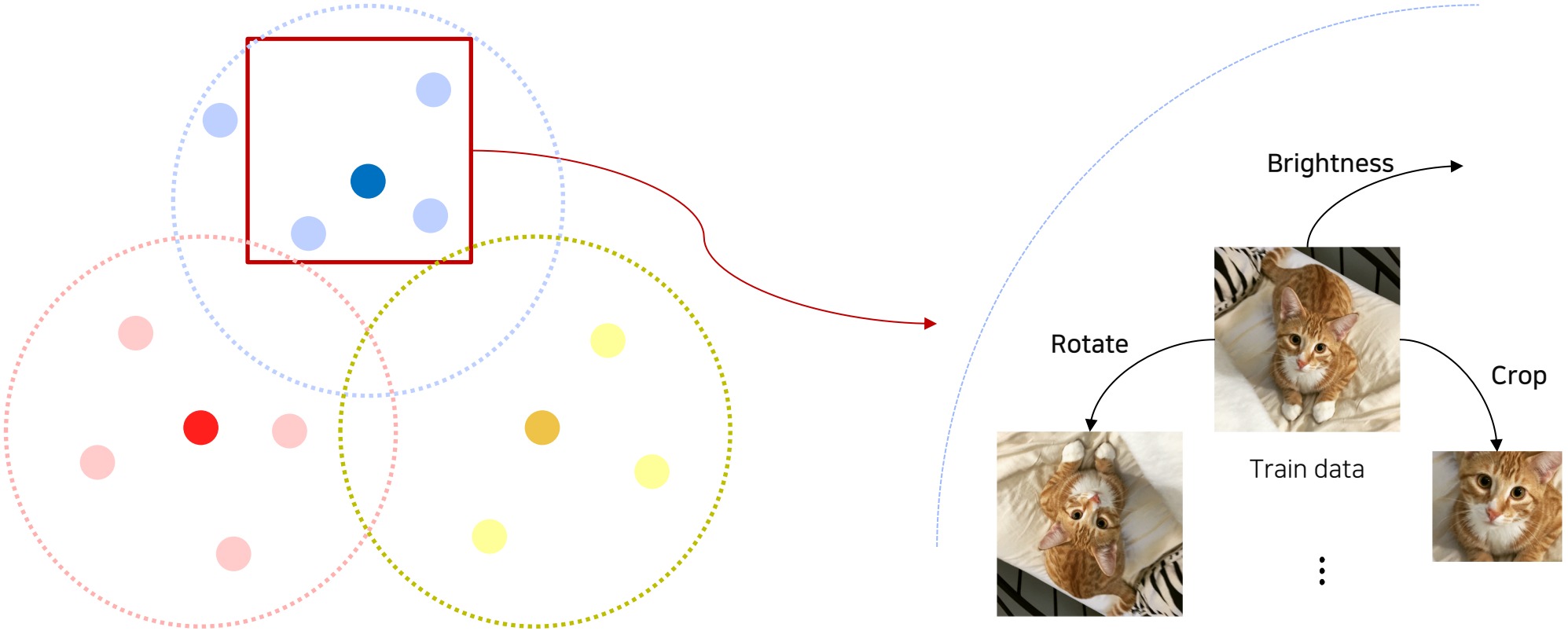
1.1 Learning representation of dataset

Augmenting data to fill more space and to close the gap



1.1 Learning representation of dataset

Examples of augmentations to make a dataset denser



And many other augmentations available!

1.2 Data augmentation

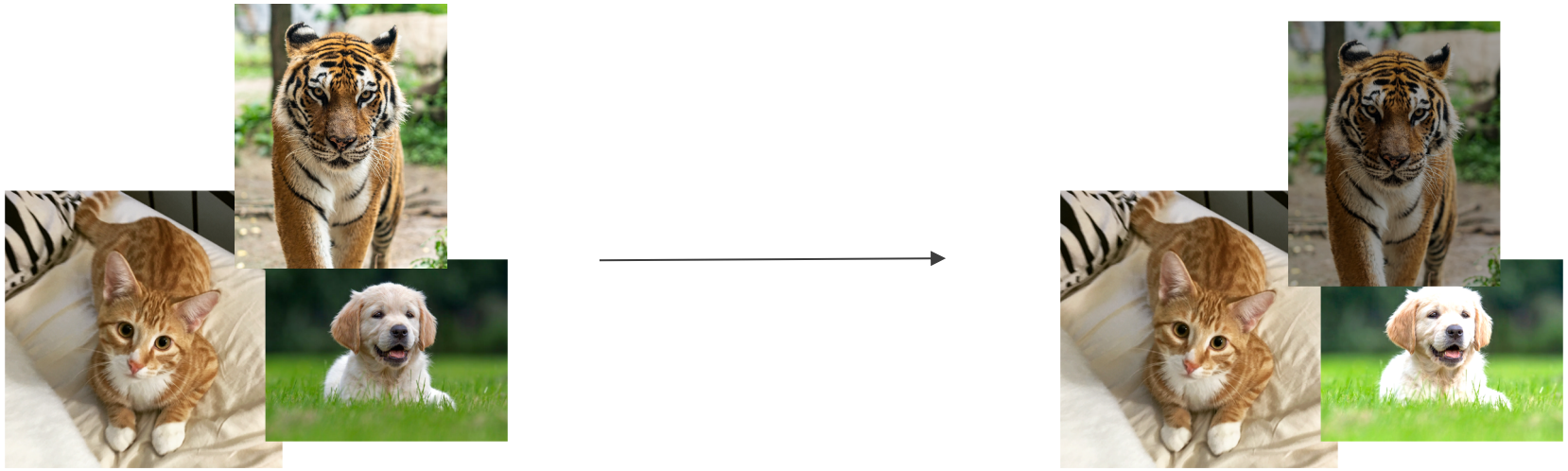
Image data augmentation

- Applying various image transformations to the dataset
  - Crop, Shear, Brightness, Perspective, Rotate ⋯
- OpenCV and NumPy have various methods useful for data augmentation
- Goal: Make training dataset's distribution similar with real data distribution

1.3 Various data augmentation methods

Brightness adjustment

- Various brightness in dataset



Original data samples

Augmented data

1.3 Various data augmentation methods

Brightness adjustment

- Brightness adjustment (brightening) using NumPy

```
def brightness_augmentation(img):
    # numpy array img has RGB value(0~255) for each pixel
    img[:,:,0] = img[:,:,0] + 100 # add 100 to R value
    img[:,:,1] = img[:,:,1] + 100 # add 100 to G value
    img[:,:,2] = img[:,:,2] + 100 # add 100 to B value

    img[:,:,0][img[:,:,0]>255] = 255  # clip R values over 255
    img[:,:,1][img[:,:,1]>255] = 255  # clip G values over 255
    img[:,:,2][img[:,:,2]>255] = 255  # clip B values over 255
    return img
```

1.3 Various data augmentation methods

Rotate, flip

- Diverse angles in dataset

Original data samples

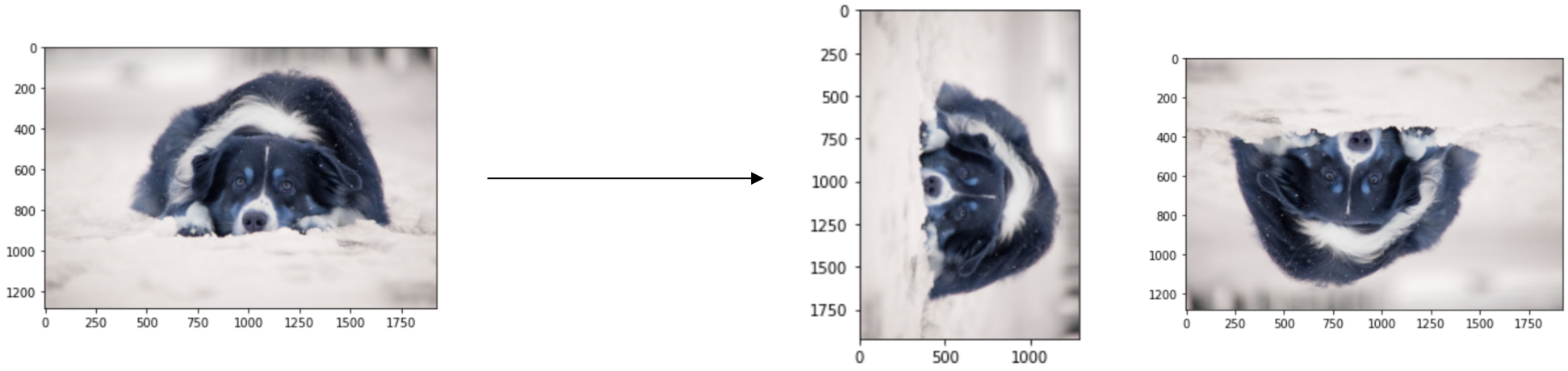
Augmented data

1.3 Various data augmentation methods

Rotate, flip

- Rotating (flipping) image using OpenCV

```
img_rotated = cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)
img_flipped = cv2.rotate(image, cv2.ROTATE_180)
```



1.3 Various data augmentation methods

Crop

- Learning with only part of images

Original data samples

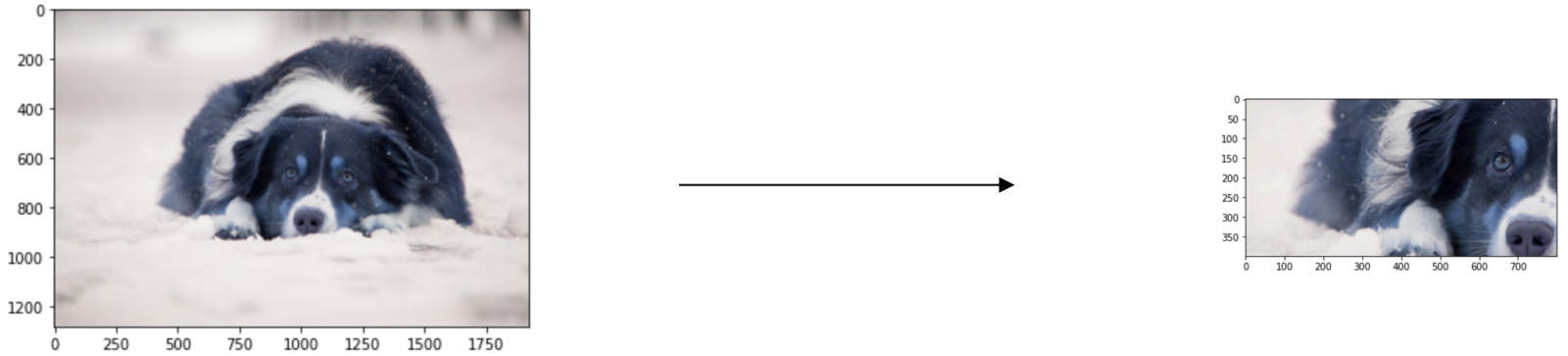
Augmented data

1.3 Various data augmentation methods

Crop

- Cropping image using NumPy

```
y_start = 500      # y pixel to start cropping
crop_y_size = 400 # cropped image's height
x_start = 300      # x pixel to start cropping
crop_x_size = 800 # cropped image's width
img_cropped = image[y_start : y_start + crop_y_size, x_start : x_start + crop_x_size, :]
```



1.3 Various data augmentation methods

Affine transformation

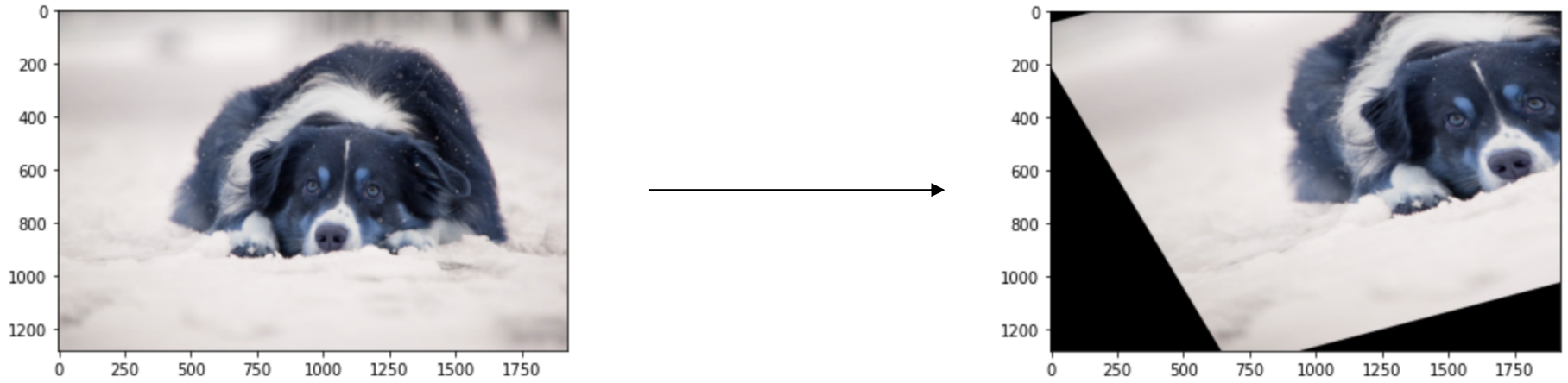
- Preserves ‘line’, ‘length ratio’, and ‘parallelism’ in image
- For example, transforming a rectangle into a parallelogram
  (See the shear transform example below)

1.3 Various data augmentation methods

Affine transformation

- Affine transformation (shear) using OpenCV

```
rows, cols, ch = image.shape
pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])
M = cv2.getAffineTransform(pts1,pts2)
shear_img = cv2.warpAffine(image, M, (cols,rows))
```



1.4 Modern augmentation techniques

CutMix

[Yun et al., ICCV 2019]

- 'Cut' and 'Mix' training example to help model better localize objects

1.4 Modern augmentation techniques

Generating new training image

[Yun et al., ICCV 2019]

- Mixing both images and labels

x_1 x_2 x_{new}




y_1 y_2 y_{new}

$[1,0]$ $[0,1]$ $[0.7,0.3]$

1.4 Modern augmentation techniques

RandAugment

[Cubuk et al., CVPRW 2020]

- Many augmentation methods exist. Hard to find best augmentations to apply
- Automatically finding the best sequence of augmentations to apply
- Random sample, apply, and evaluate augmentations

- identity
- rotate
- posterize
- sharpness
- translate-x
- autoContrast
- solarize
- contrast
- shear-x
- translate-y
- equalize
- color
- brightness
- shear-y

1.4 Modern augmentation techniques

Example of augmented images in RandAug

[Cubuk et al., CVPRW 2020]

- Augmentation policy has two parameters
  - Which augmentation to apply
  - Magnitude of augementation to apply (how much to augment)

- Parameters used in the above example
  - Which augmentation to apply : 'ShearX' & 'AutoContrast'
  - Magnitude of augmentation to apply : 9

1.4 Modern augmentation techniques

Randomly testing augmentation policies

[Cubuk et al., CVPRW 2020]

- Finding the best augmentation policy
  - Sample a policy : Policy = {N augmentations to apply} by random sampling
  - Train with a sampled policy, and evaluate the accuracy

- identity
- rotate
- posterize
- sharpness
- translate-x
- autoContrast
- solarize
- contrast
- shear-x
- translate-y
- equalize
- color
- brightness
- shear-y

1.4 Modern augmentation techniques

Augmentation helps model learning

[Cubuk et al., CVPRW 2020]

- Higher test accuracy than training without augmentation

	baseline	PBA	Fast AA	AA	RA
CIFAR-10					
Wide-ResNet-28-2	94.9	-	-	**95.9**	95.8
Wide-ResNet-28-10	96.1	**97.4**	97.3	**97.4**	97.3
Shake-Shake	97.1	**98.0**	**98.0**	**98.0**	**98.0**
PyramidNet	97.3	**98.5**	98.3	**98.5**	**98.5**
CIFAR-100					
Wide-ResNet-28-2	75.4	-	-	**78.5**	78.3
Wide-ResNet-28-10	81.2	**83.3**	82.7	82.9	**83.3**
SVHN (core set)					
Wide-ResNet-28-2	96.7	-	-	98.0	**98.3**
Wide-ResNet-28-10	96.9	-	-	98.1	**98.3**
SVHN					
Wide-ResNet-28-2	98.2	-	-	**98.7**	**98.7**
Wide-ResNet-28-10	98.5	98.9	98.8	98.9	**99.0**

2.

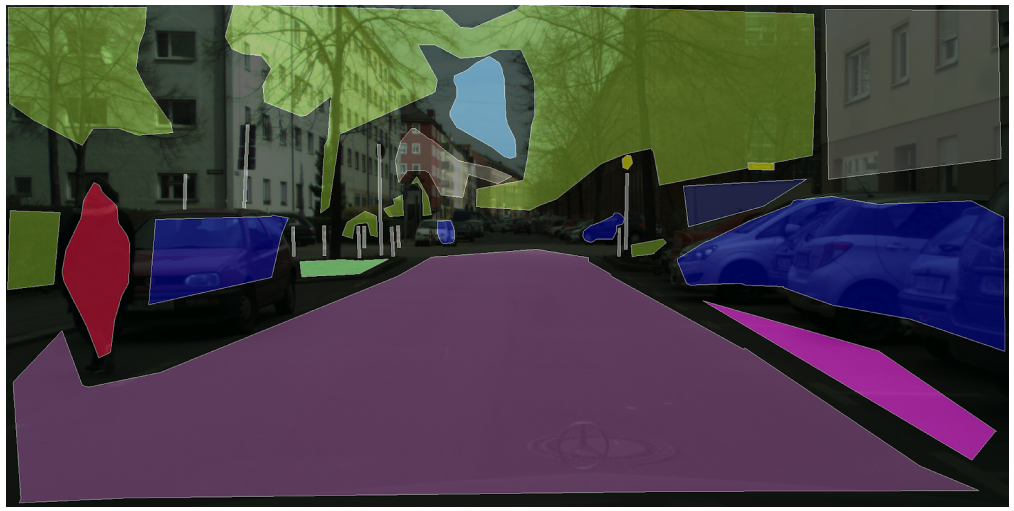
Leveraging pre-trained information

2.1 Transfer learning

The high-quality dataset is expensive and hard to obtain

- Supervised learning requires a very large-scale dataset for training
- Annotating data is very expensive, and its quality is not ensured
- Transfer learning : A practical training method with a small dataset!

Ideal segmentation label we imagine



Actual human-labeled data

Benefits when using transfer learning

By transfer learning, we can easily

adapt to a new task by leveraging pre-trained knowledge (feature)!

2.1 Transfer learning

Motivational observation : Similar datasets share common information

- E.g., 4 distinct datasets with similar images
- Knowledge learned from one dataset can be applied to other datasets!

(a) ImageNet Synset: One sample image from each category

(b) Corel-1000 Dataset: Sample images from each category

(c) Caltech-256 Dataset: One sample image from each category

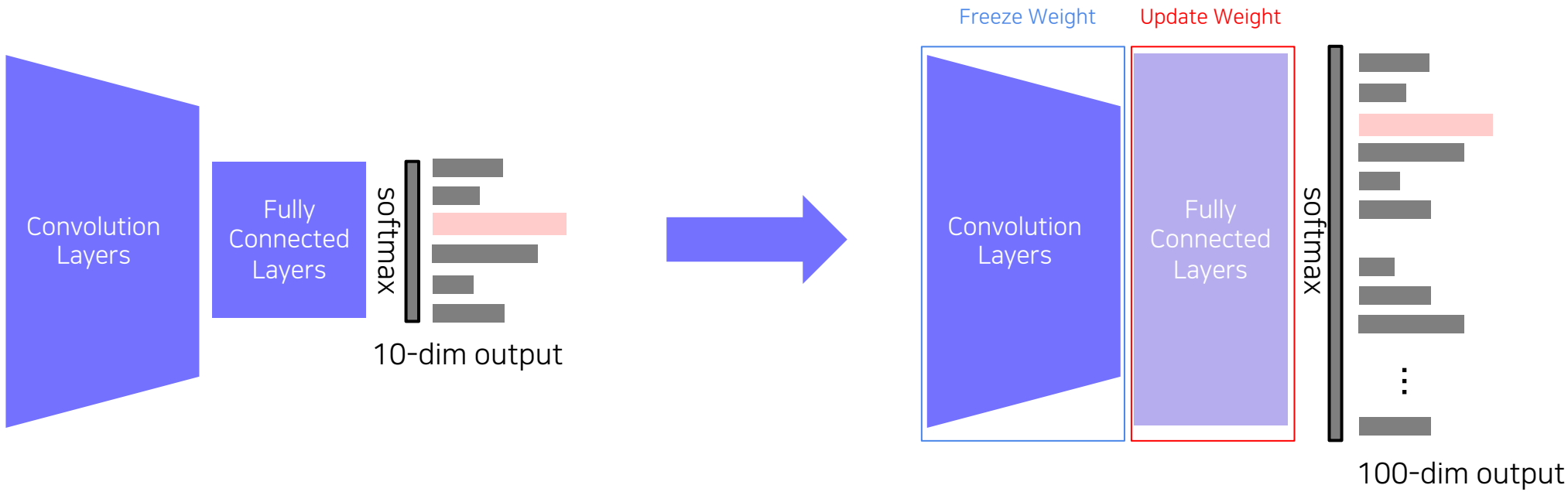
(d) Caltech-101 Dataset: One sample image from each category

Images from [Ahmed et al., Applied Intelligence 2017]

2.1 Transfer learning

Approach 1 : Transfer knowledge from a pre-trained task to a new task

- Given a model pre-trained on a 10-class dataset,
- Chop off the final layer of the pre-trained model, add and only re-train a new FC layer
- Extracted features preserve all the knowledge from pre-training

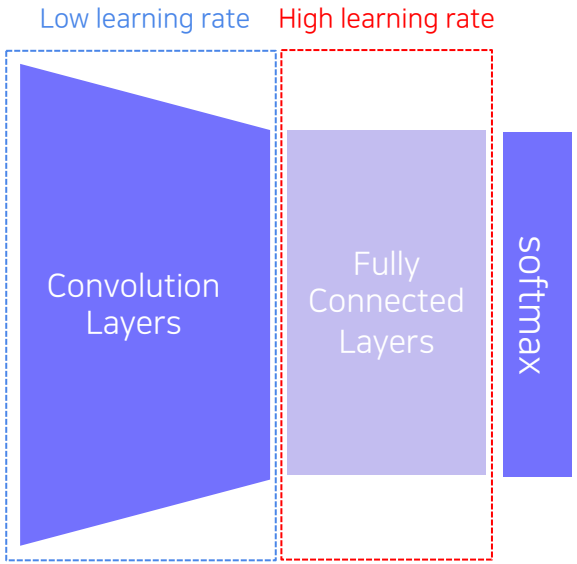


2.1 Transfer learning

Approach 2 : Fine-tuning the whole model

[Oquab et al., CVPR 2015]

- Given a model pre-trained on a dataset
- Replace the final layer of the pre-trained model to a new one, and re-train the whole model
- Set learning rates differently



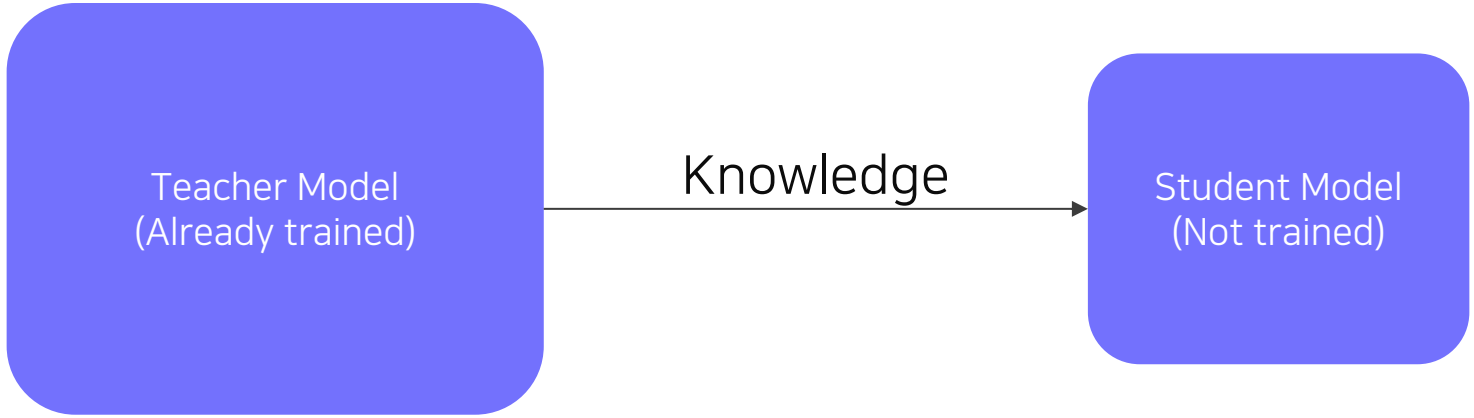
Update the whole model

2.2 Knowledge distillation

Passing what model learned to 'another' smaller model (Teacher-student learning)

[Hinton et al., NIPS deep learning workshop 2015]

- 'Distillate' knowledge of a trained model into another smaller model
- Used for model compression (Mimicking what a larger model knows)
- Also, used for pseudo-labeling (Generating pseudo-labels for an unlabeled dataset)

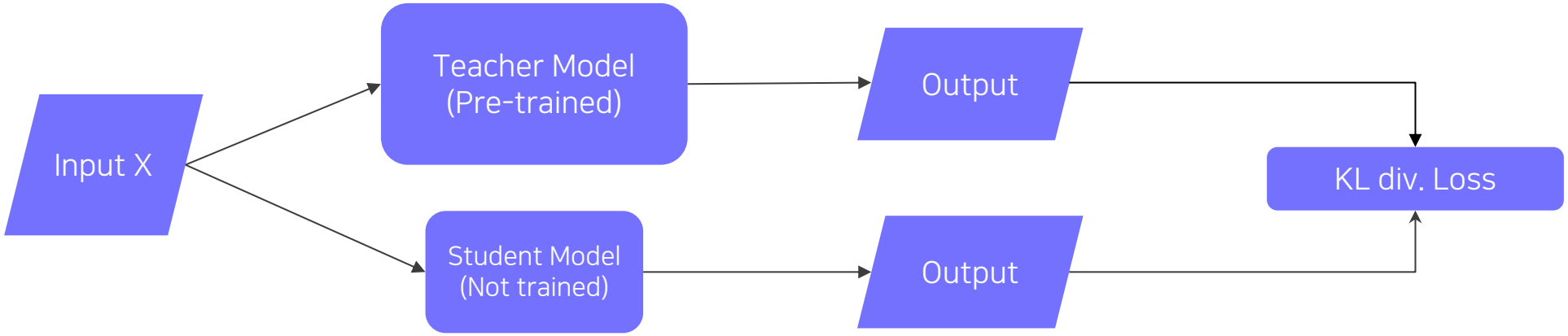


2.2 Knowledge distillation

Teacher-student network structure

[Hinton et al., NIPS deep learning workshop 2015]

- The student network learns what the teacher network knows
- The student network mimics outputs of the teacher network
- Unsupervised learning, since training can be done only with unlabeled data

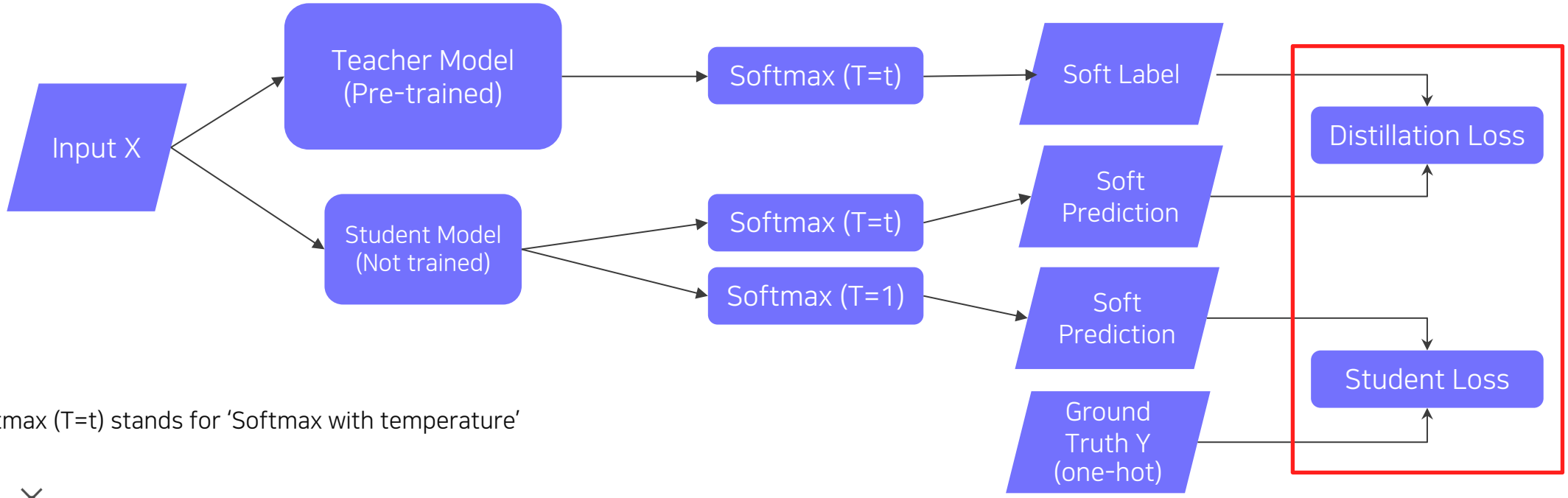


2.2 Knowledge distillation

Knowledge distillation

[Hinton et al., NIPS deep learning workshop 2015]

- When labeled data is available, can leverage labeled data for training (Student Loss)
- Distillation loss to 'predict similar outputs with the teacher model'



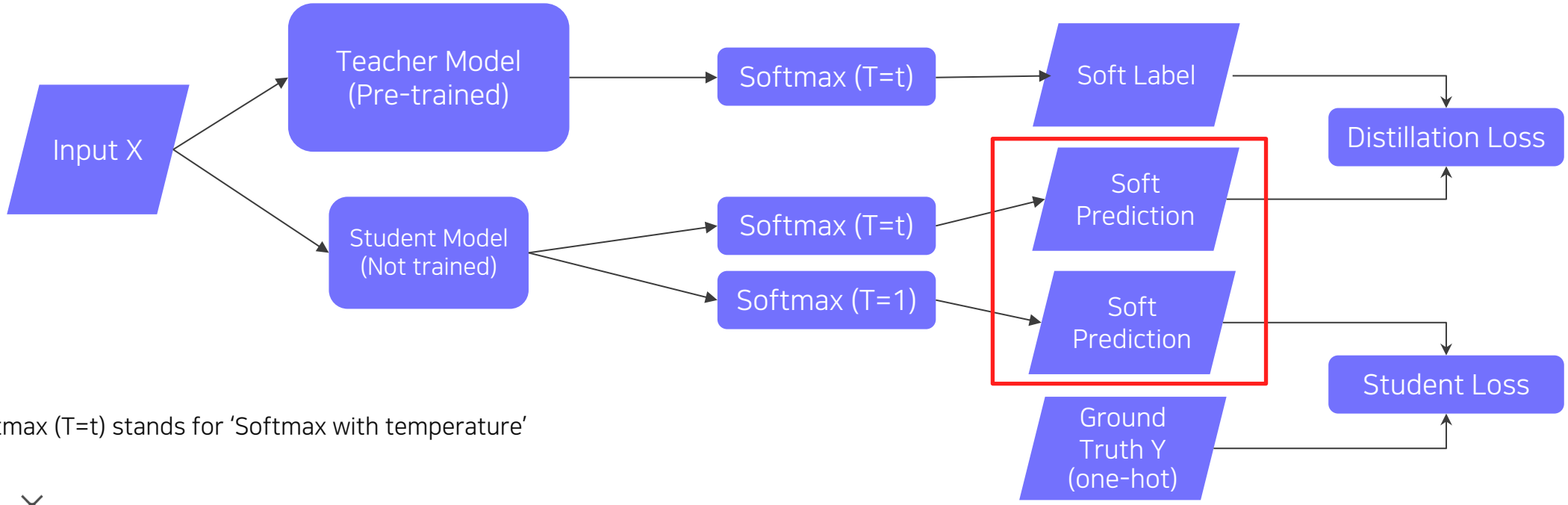
* Softmax (T=t) stands for 'Softmax with temperature'

2.2 Knowledge distillation

Knowledge distillation

[Hinton et al., NIPS deep learning workshop 2015]

- When labeled data is available, can leverage labeled data for training (Student Loss)
- Distillation loss to 'predict similar outputs with the teacher model'



* Softmax (T=t) stands for 'Softmax with temperature'

2.2 Knowledge distillation

Hard label vs. Soft label

- Hard label (One-hot vector)
  - Typically obtained from the dataset
  - Indicates whether a class is 'true answer' or not
- Soft label
  - Typicall output of the model (=inference result)
  - Regard it as "knowledge". Useful to observe how the model thinks

$$\begin{pmatrix} Bear \\ Cat \\ Dog \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} Bear \\ Cat \\ Dog \end{pmatrix} = \begin{pmatrix} 0.14 \\ 0.8 \\ 0.06 \end{pmatrix}$$

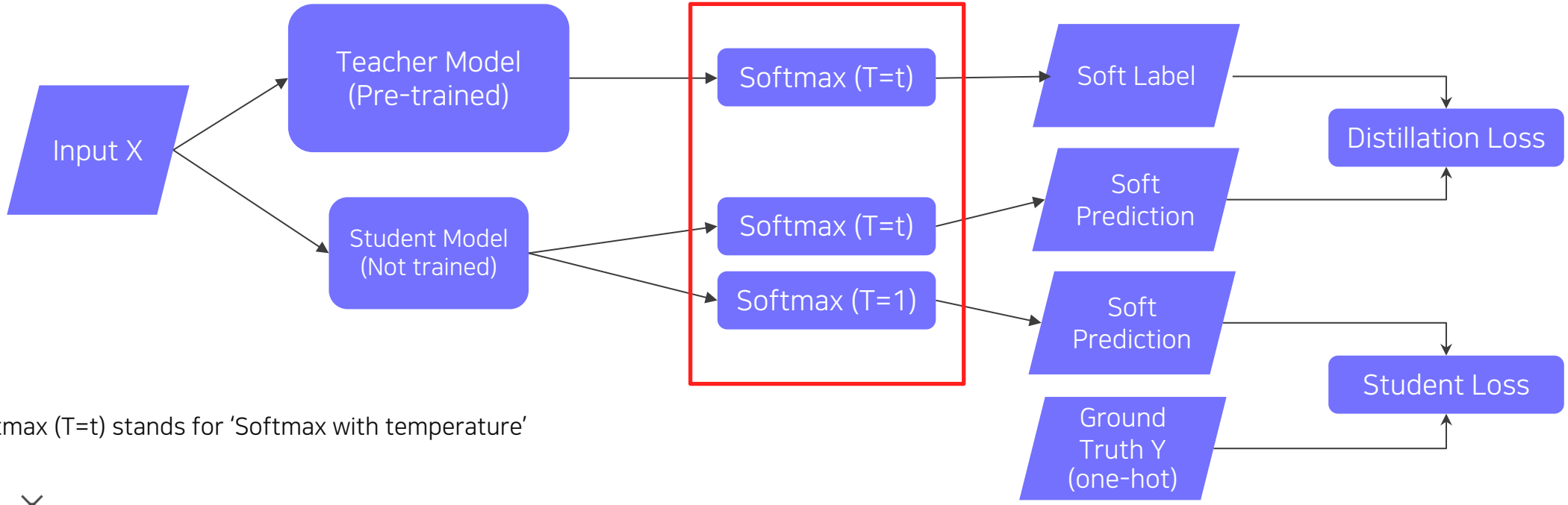
Hard label　　Soft label

2.2 Knowledge distillation

Knowledge distillation

[Hinton et al., NIPS deep learning workshop 2015]

- When labeled data is available, can leverage labeled data for training (Student Loss)
- Distillation loss to 'predict similar outputs with the teacher model'



* Softmax (T=t) stands for 'Softmax with temperature'

2.2 Knowledge distillation

Softmax with temperature (T)

- Softmax with temperature : controls difference in output between small & large input values
- A large T smoothens large input value differences
- Useful to synchronize the student and teacher models' outputs

Normal Softmax (=Hard Prediction)

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

$softmax(5,10) = (0.0067, 0.9933)$

Softmax with temperature T (=Soft Prediction)

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

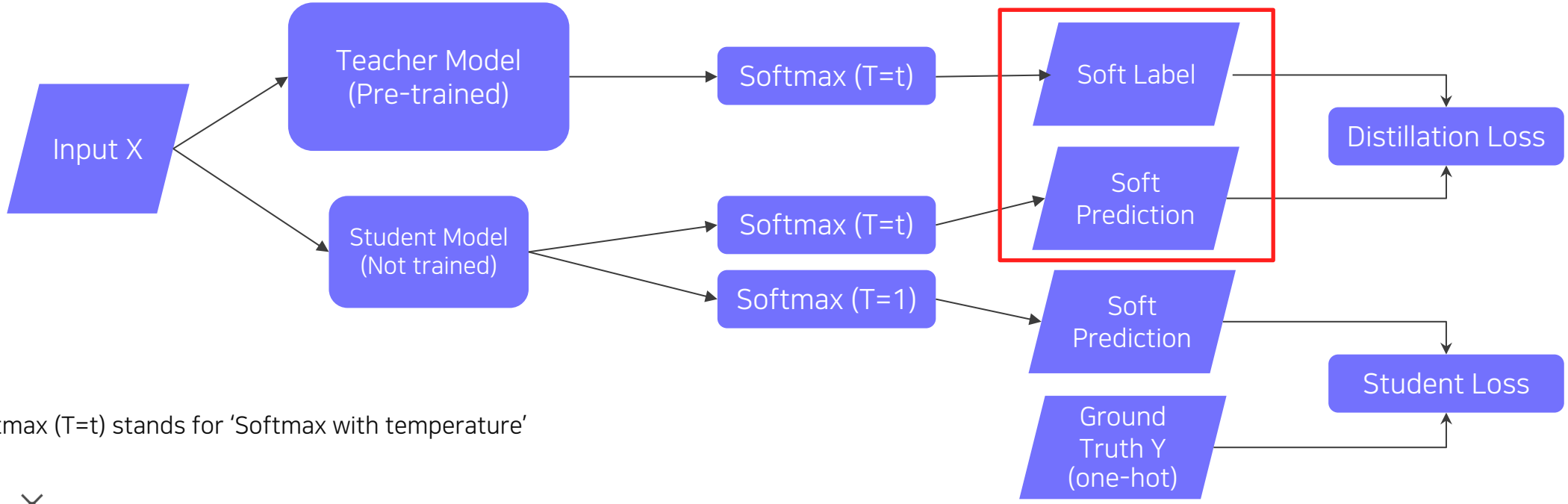
$softmax_{(t=100)}(5,10) = (0.4875, 0.5125)$

2.2 Knowledge distillation

Knowledge distillation

[Hinton et al., NIPS deep learning workshop 2015]

- When labeled data is available, can leverage labeled data for training (Student Loss)
- Distillation loss to 'predict similar outputs with the teacher model'
- Semantic information is not considered in distillation



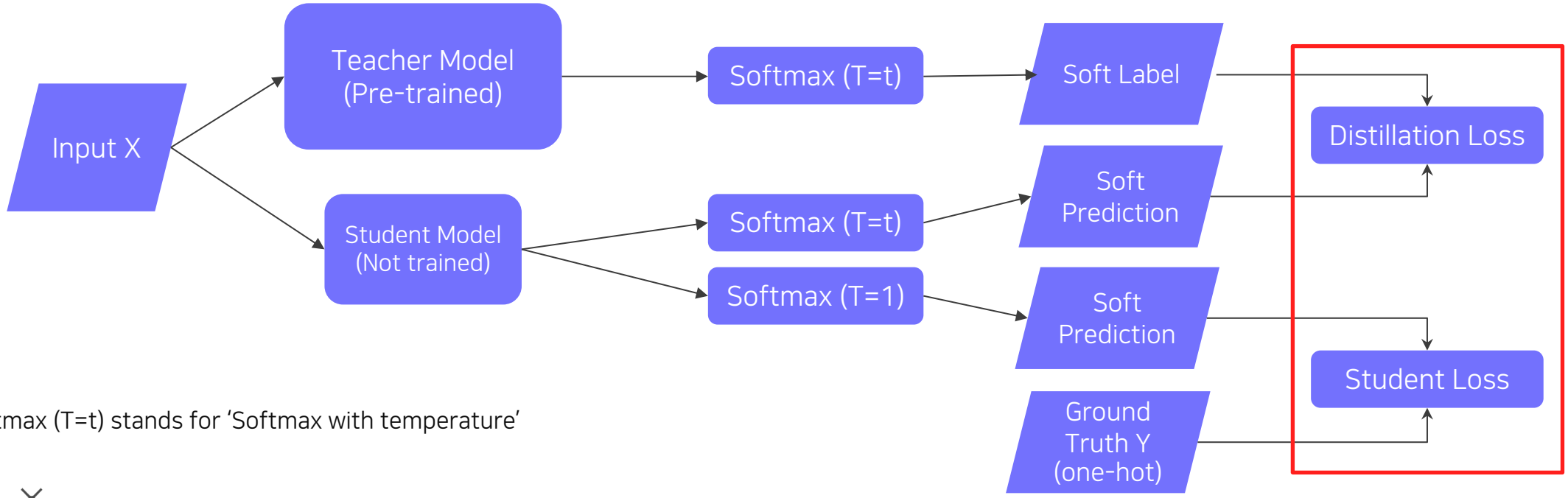
* Softmax (T=t) stands for 'Softmax with temperature'

2.2 Knowledge distillation

Knowledge distillation

[Hinton et al., NIPS deep learning workshop 2015]

- When labeled data is available, can leverage labeled data for training (Student Loss)
- Distillation loss to 'predict similar outputs with the teacher model'
- Semantic information is not considered in distillation



* Softmax (T=t) stands for 'Softmax with temperature'

2.2 Knowledge distillation

Intuition about distillation loss and student loss

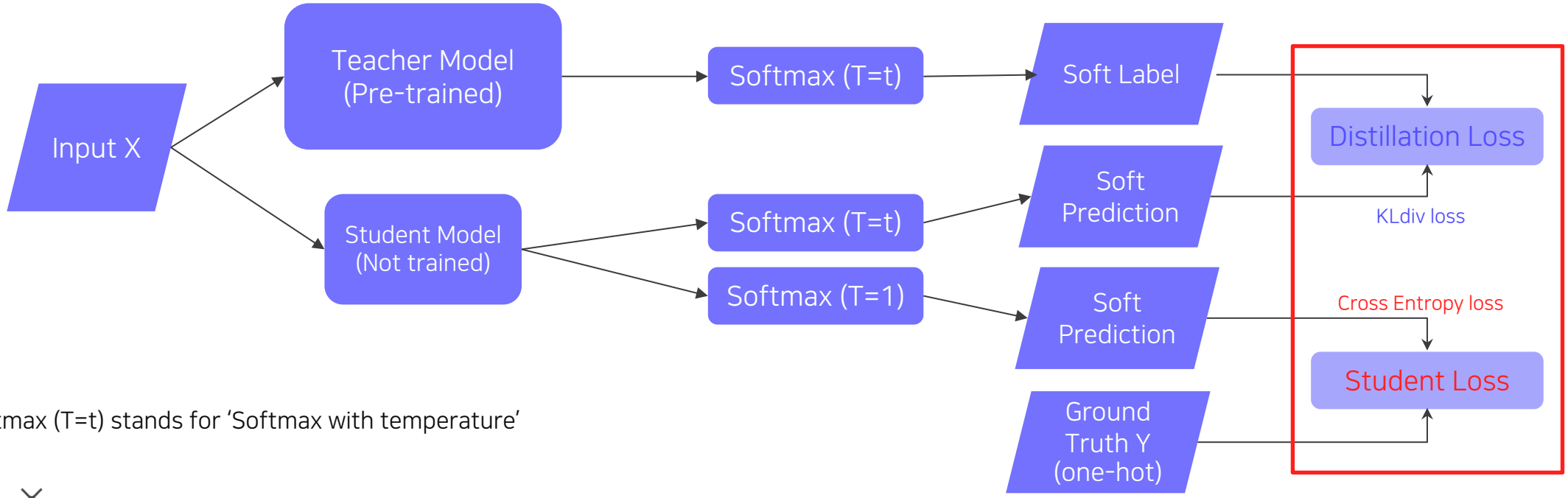
- Distillation Loss
  - KLdiv(Soft label, Soft prediction)
  - Loss = difference between the teacher and student network's inference
  - Learn what teacher network knows by mimicking

- Student Loss
  - CrossEntropy(Hard label, Soft prediction)
  - Loss = difference between the student network's inference and true label
  - Learn the "right answer"

2.2 Knowledge distillation

Loss functions in knowledge distillation

- Weighted sum of "Distillation loss" and "Student loss"



* Softmax (T=t) stands for 'Softmax with temperature'

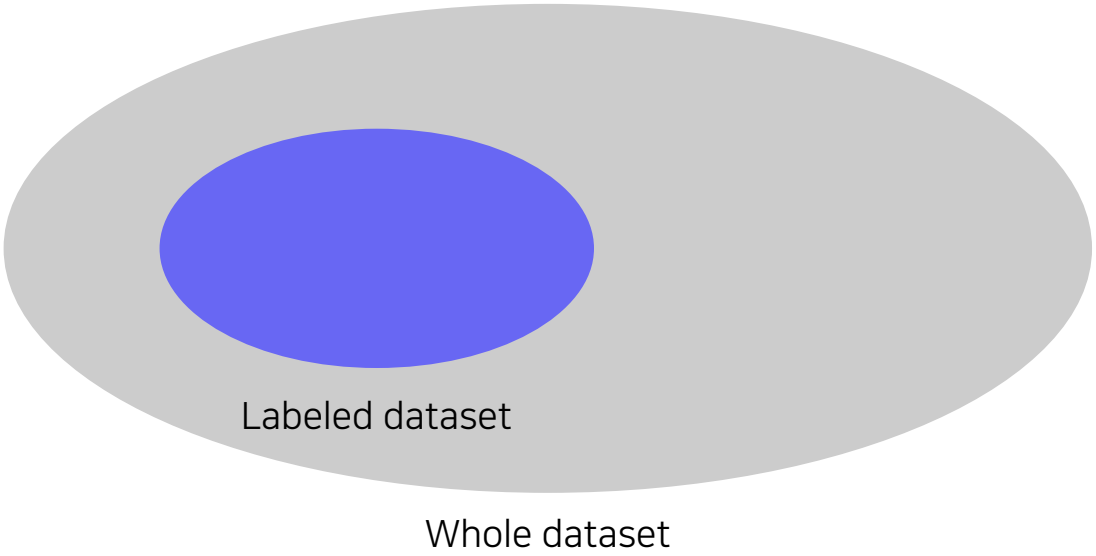
3.

Leveraging unlabeled dataset for training

3.1 Semi-supervised learning

There are lots of unlabeled data

- Typically, only a small portion of data is labeled
- Is there any way to learn from unlabeled data?
- Semi-supervised learning : Unsupervised (No label) + Fully Supervised (fully labeled)

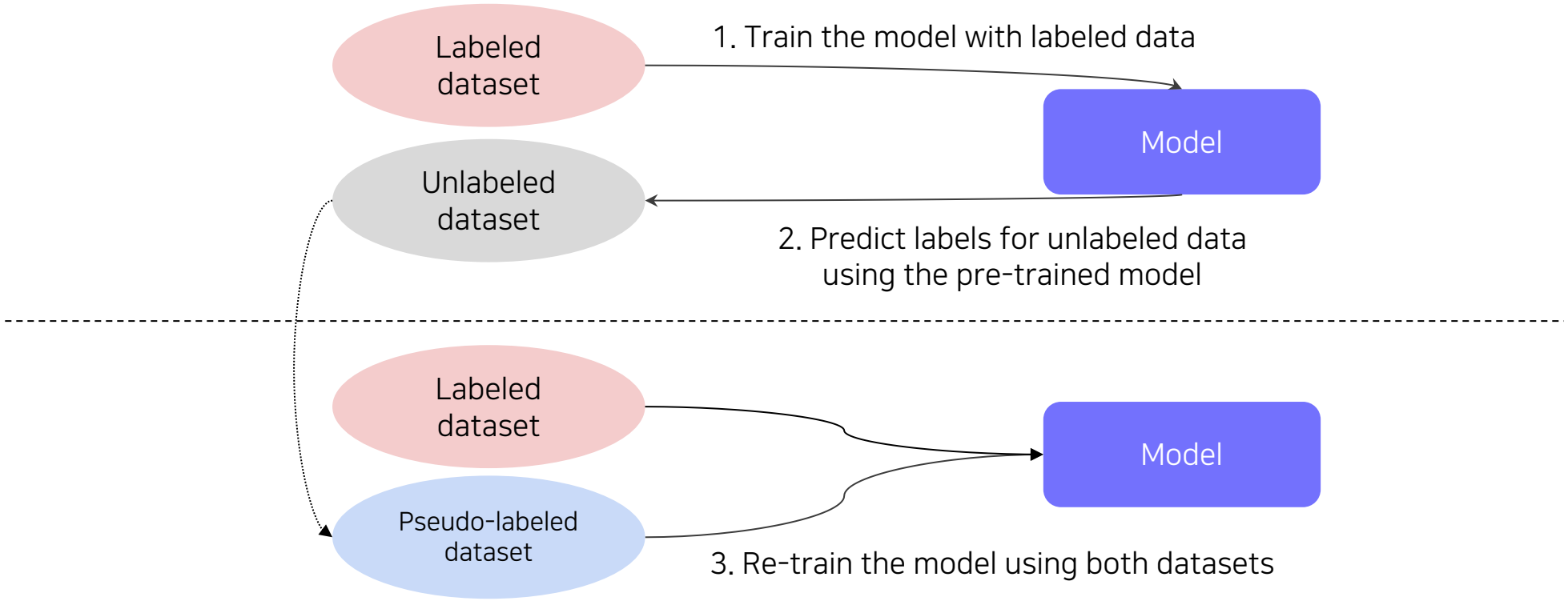


3.1 Semi-supervised learning

Semi-supervised learning with pseudo labeling

[Lee, ICML workshop 2013]

- Pseudo-labeling unlabeled data using a pre-trained model, then use for training



3.2 Self-training

Recap : Data efficient learning methods so far

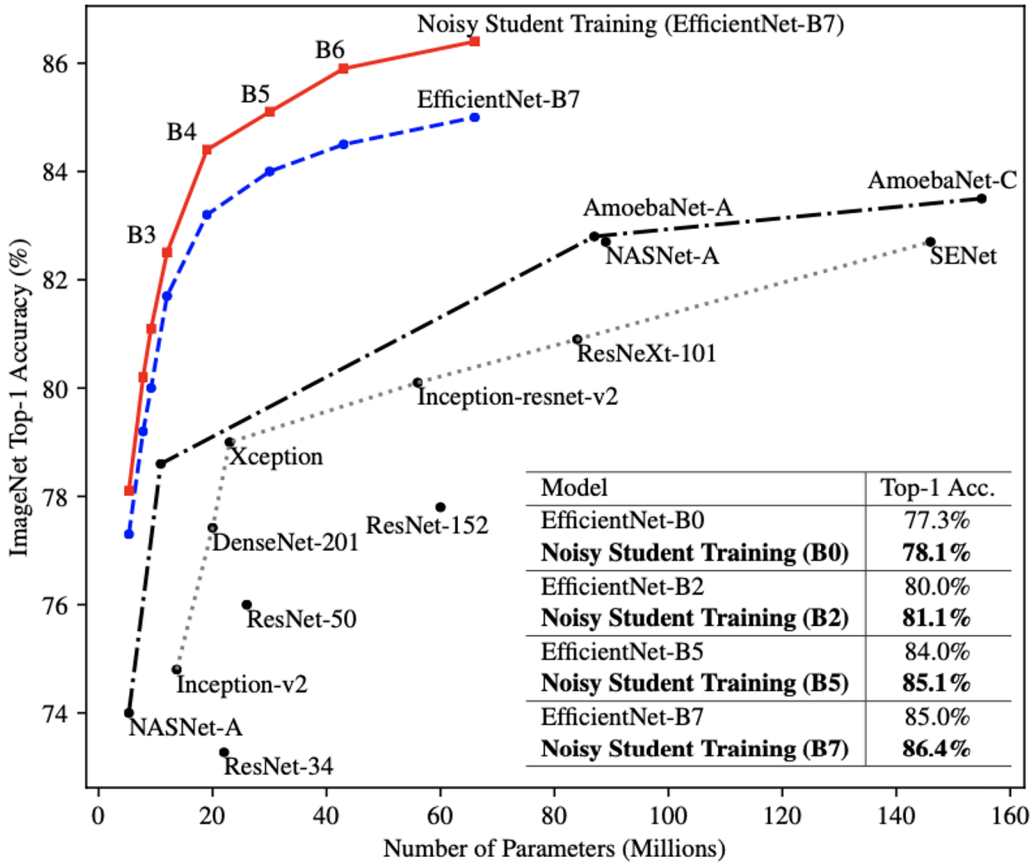
- Data Augmentation
  - Augment a dataset to make the dataset closer to real data distribution
- Knowledge distillation
  - Train a student network to imitate a teacher network
  - Transfer the teacher network's knowledge to the student network
- Semi-supervised learning (Pseudo label-based method)
  - Pseudo-label an unlabeled dataset using a pre-trained model, then use for training
  - Leveraging an unlabeled dataset for training!

3.2 Self-training

Self-training

[Xie et al., CVPR 2020]

- Augmentation + Teacher-Student networks + semi-supervised learning
- SOTA ImageNet classification, 2019

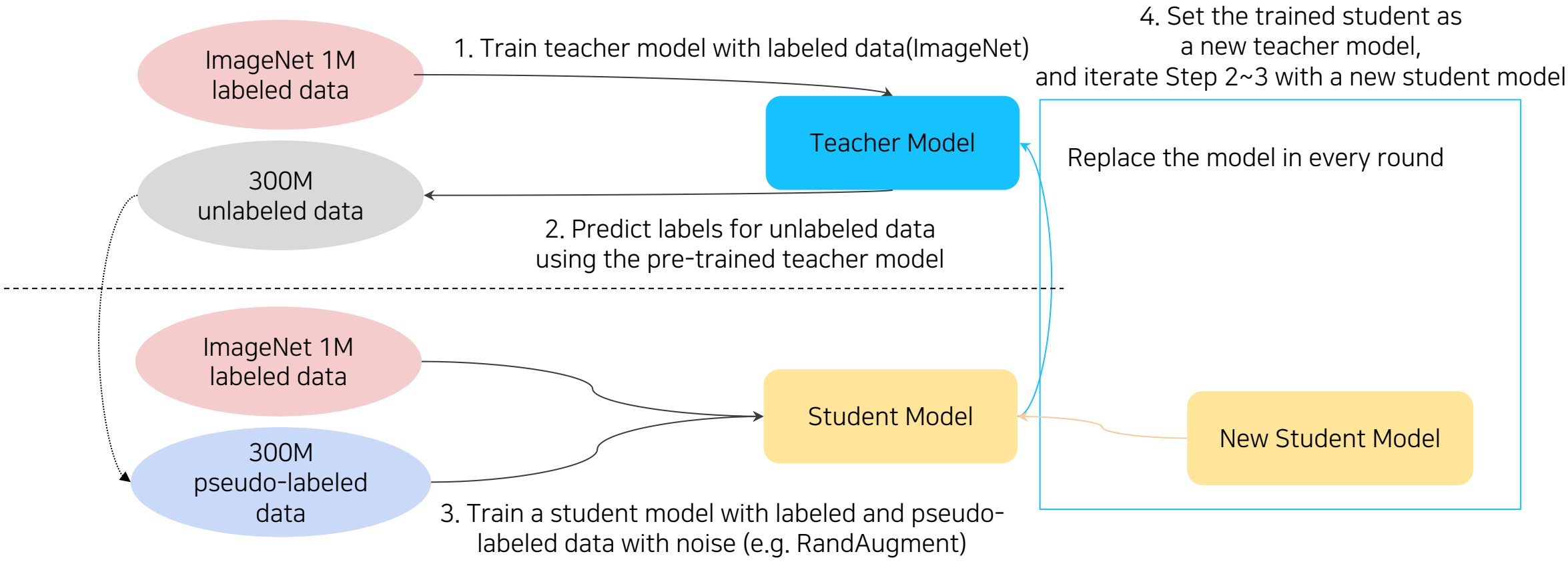


Model	Top-1 Acc.
EfficientNet-B0	77.3%
Noisy Student Training (B0)	**78.1%**
EfficientNet-B2	80.0%
Noisy Student Training (B2)	**81.1%**
EfficientNet-B5	84.0%
Noisy Student Training (B5)	**85.1%**
EfficientNet-B7	85.0%
Noisy Student Training (B7)	**86.4%**

3.2 Self-training

Self-training with noisy student

[Xie et al., CVPR 2020]

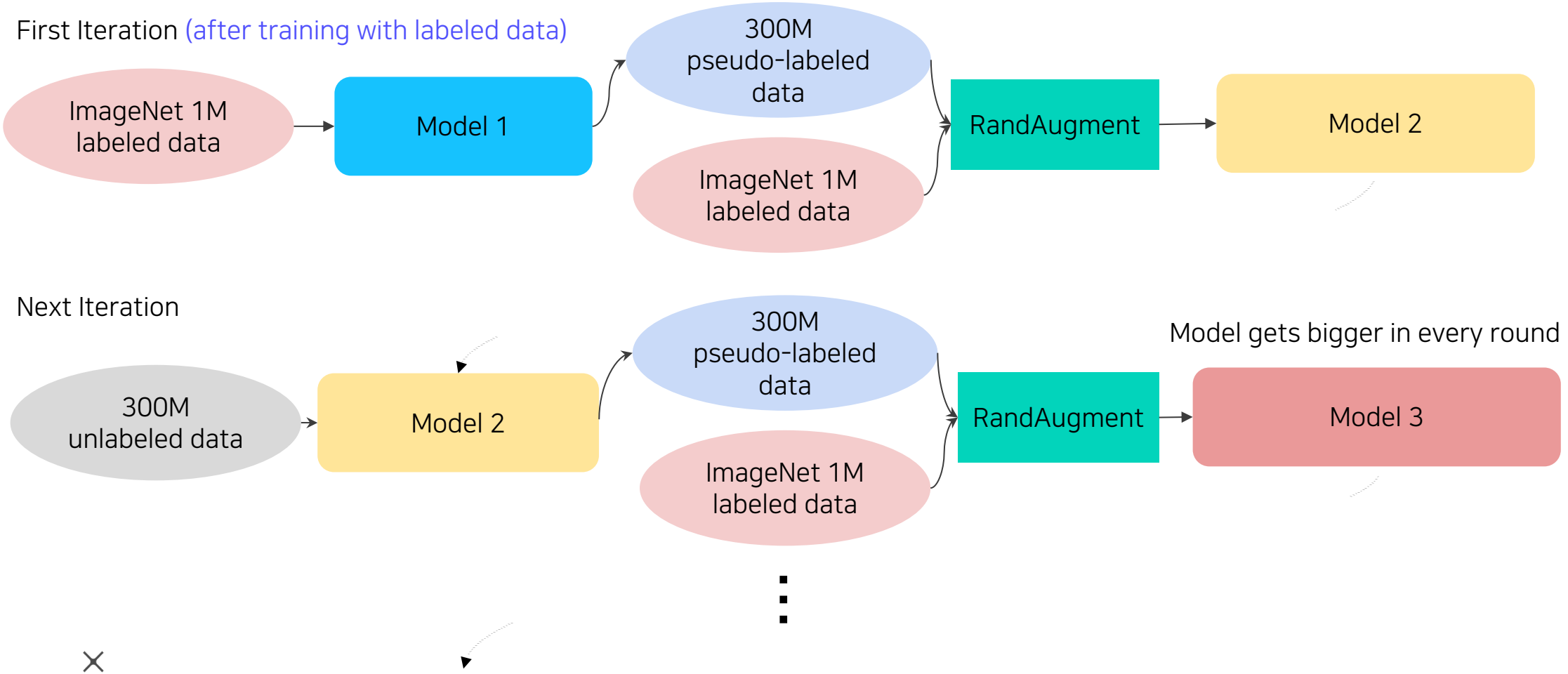


3.2 Self-training

Leveraging unlabeled dataset for training

Iteratively training noisy student network using teacher network

[Xie et al., CVPR 2020]



3.2 Self-training

Brief overview of self-training

[Xie et al., CVPR 2020]

1. Train initial teacher model with labeled data
2. Pseudo-label unlabeled data using teacher model
3. Train student model with both lableled and unlabeled data with augmentation
4. Set the student model as a new teacher, and set new model (bigger) as a new student
5. Repeat 2~4 with new teacher/student models

Reference

1. Data augmentation

- Yun et al., CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features, ICCV 2019
- Cubuk et al., Randaugment: Practical automated data augmentation with a reduced search space, CVPRW 2020

2. Leveraging pre-trained information

- Ahmed et al., Fusion of local and global features for effective image extraction, Applied Intelligence 2017
- Oquab et al., Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks, CVPR 2015
- Hinton et al., Distilling the Knowledge in a Neural Network, NIPS deep learning workshop 2015
- Li & Hoiem, Learning without Forgetting, TPAMI 2018

3. Leveraging unlabeled dataset for training

- Lee, Pseudo-label : The simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks, ICML Workshop 2013
- Xie et al., Self-training with Noisy Student improves ImageNet classification, CVPR 2020

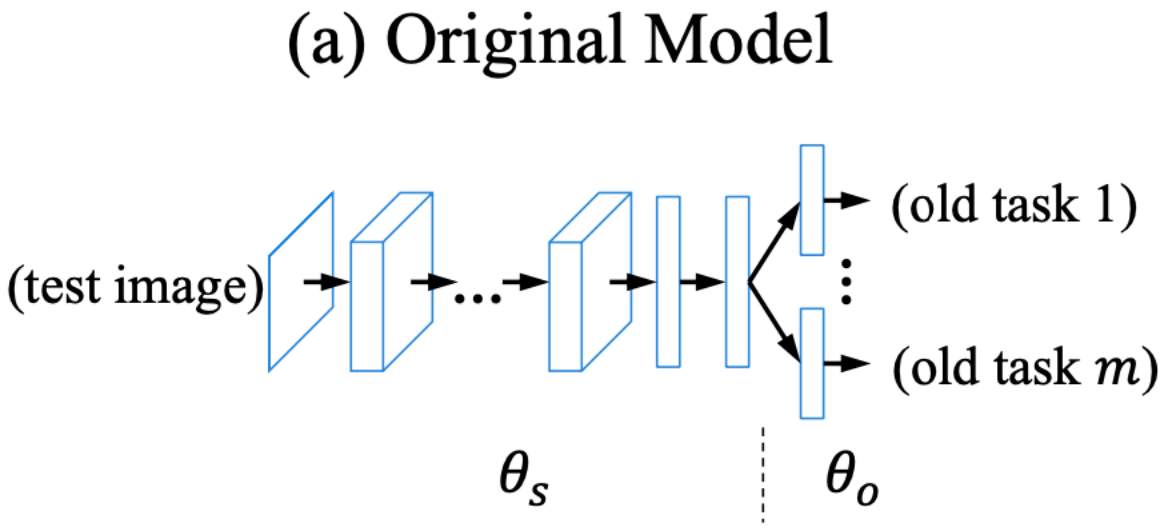
End of Document
Thank You.

2.3 Learning without Forgetting

Fine-tuning a model for both old and new tasks

[Li & Hoiem, TPAMI 2018]

- Given a model pre-trained for a specific task (old task)
- Goal : Fine-tuning a model that performs well in both old and new tasks
- Fine-tuning?

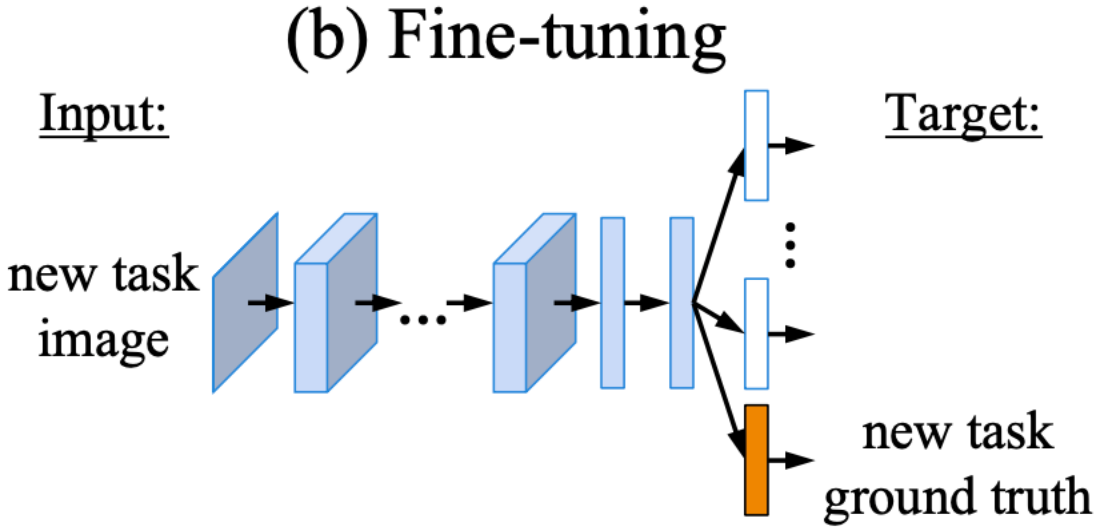


2.3 Learning without Forgetting

Fine-tuning forgets old tasks

[Li & Hoiem, TPAMI 2018]

- When fine-tuning, the whole model is updated
- Therefore, the feature extractor is fitted for the new task
- The updated feature extractor is not compatible to the old tasks' classifiers (Forgetting)
- (Naïve approach) Train using both old & new datasets

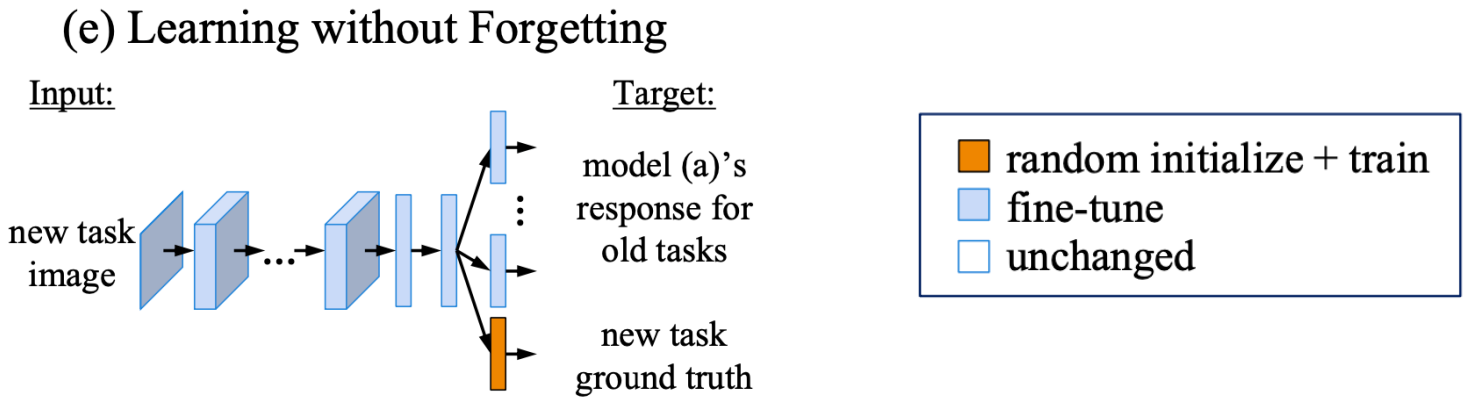


2.3 Learning without Forgetting

Learning without Forgetting

[Li & Hoiem, TPAMI 2018]

- Fine-tuning a model to perform well on both 'old' and 'new' tasks without the old dataset
- Learning without Forgetting
  - Standard training for the new task
  - Training the old task branches to follow the pre-trained model's output with the new task data
- The intuition is similar to distillation; learning what other models know!



2.3 Learning without Forgetting

Loss functions in Learning without Forgetting

[Li & Hoiem, TPAMI 2018]

- Weighted sum of 'Old task loss' and 'New task loss'