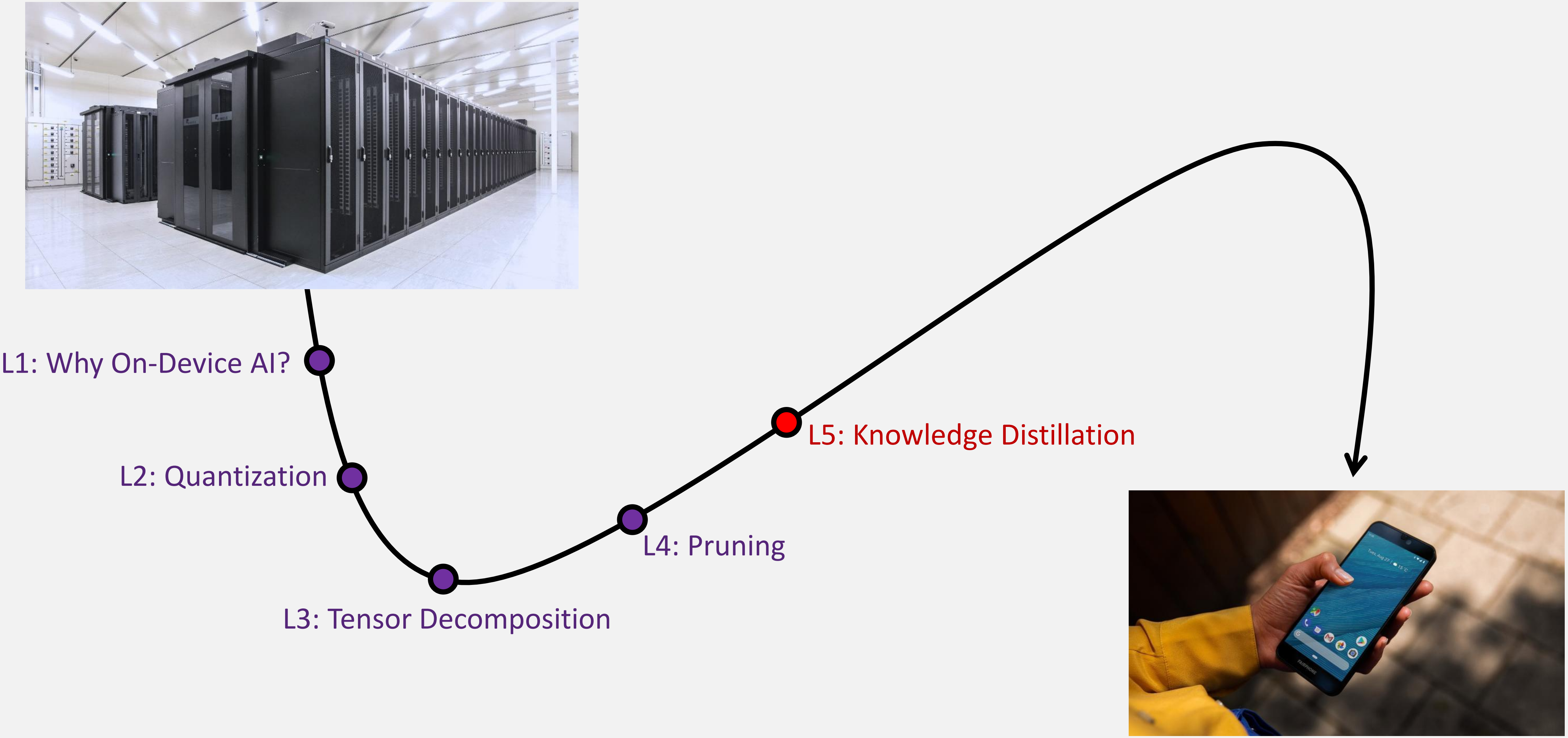
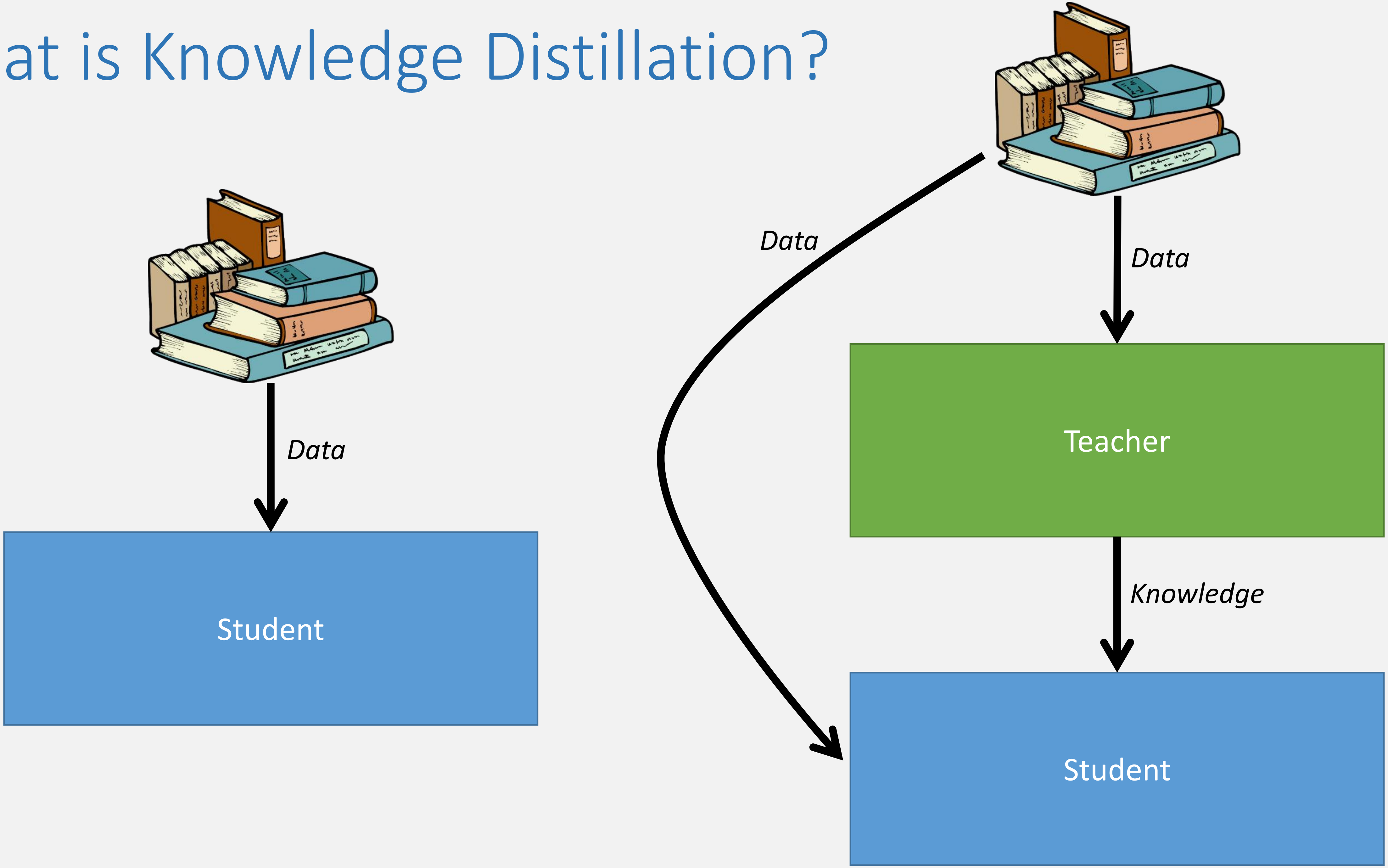
Lecture 5: Knowledge Distillation

On-Device AI Journey



What is Knowledge Distillation?



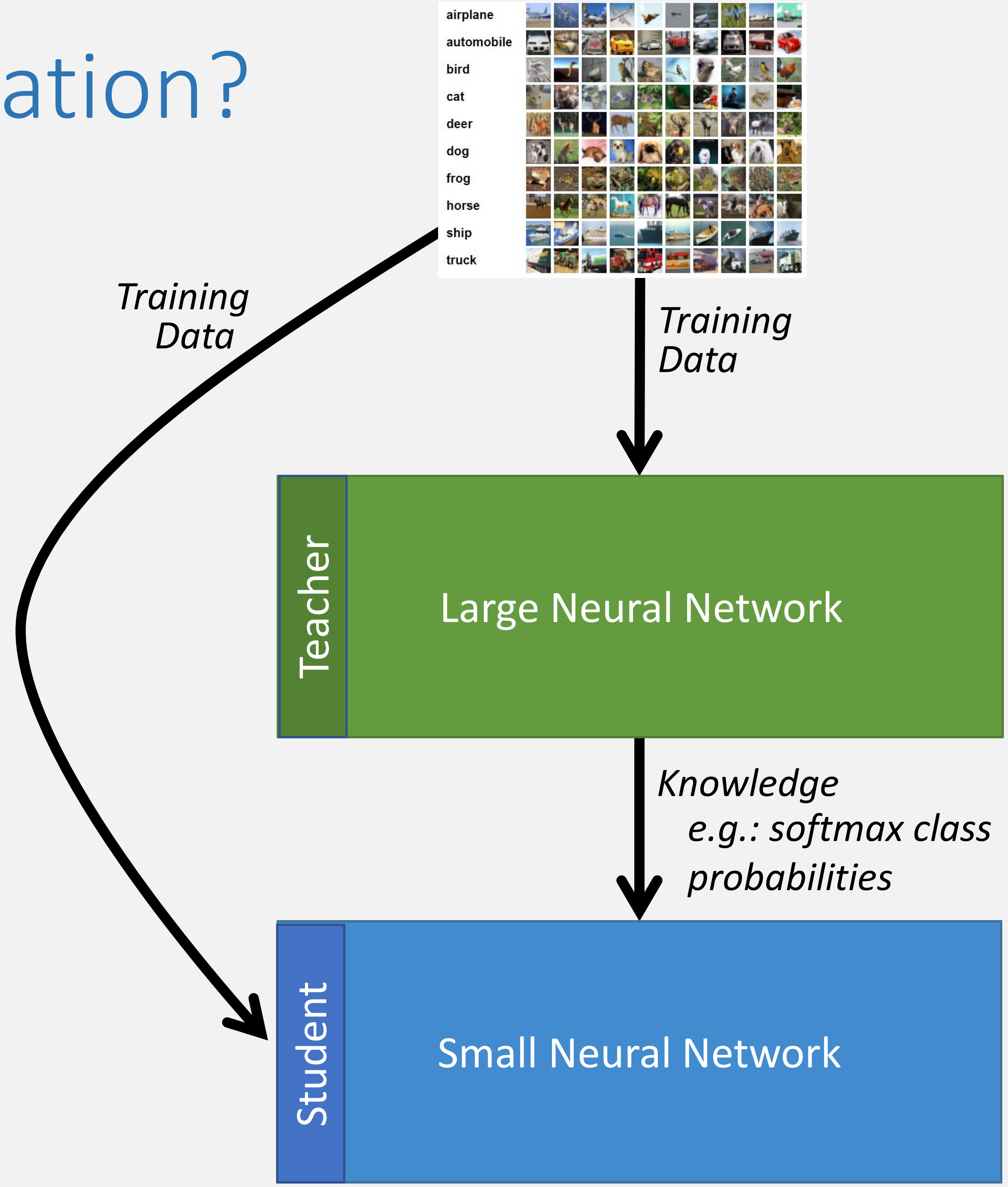
What is Knowledge Distillation?

- Distill "knowledge" from a large neural network to a small one.
  - E.g. ResNet101 → MobileNet

- Larger DNNs are easier to train
- Small DNNs are easier to deploy

- Knowledge?
  - Classification: Softmax class probabilities

- Proposed by Caruna et al. (2006)
- Generalized by Hinton et al. (2015)



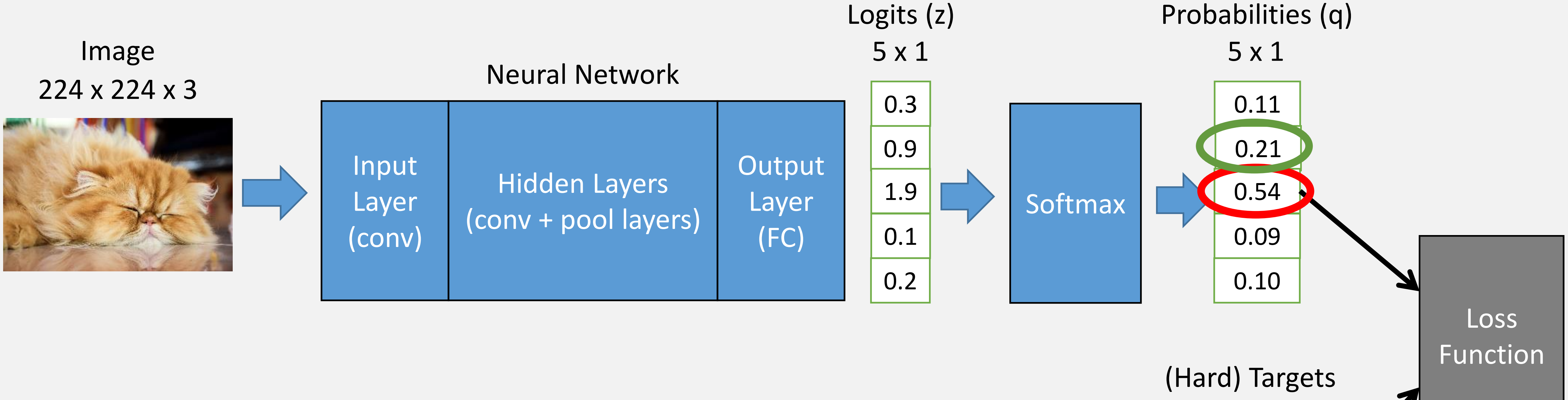
Learning Outcomes

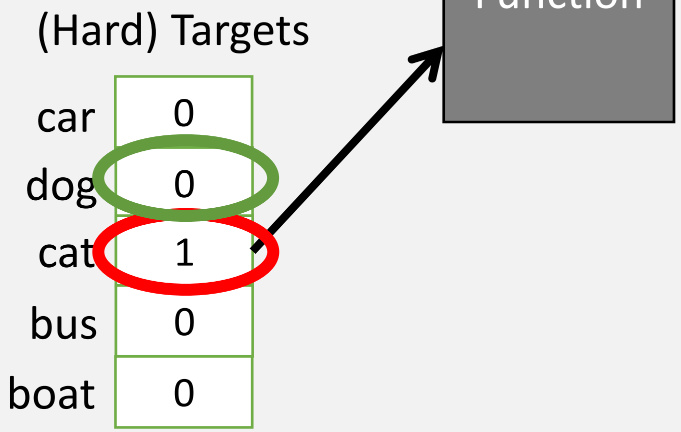
Understand what knowledge distillation is and why it is needed.

Write code to perform knowledge distillation

Understand advanced knowledge distillation techniques and open problems

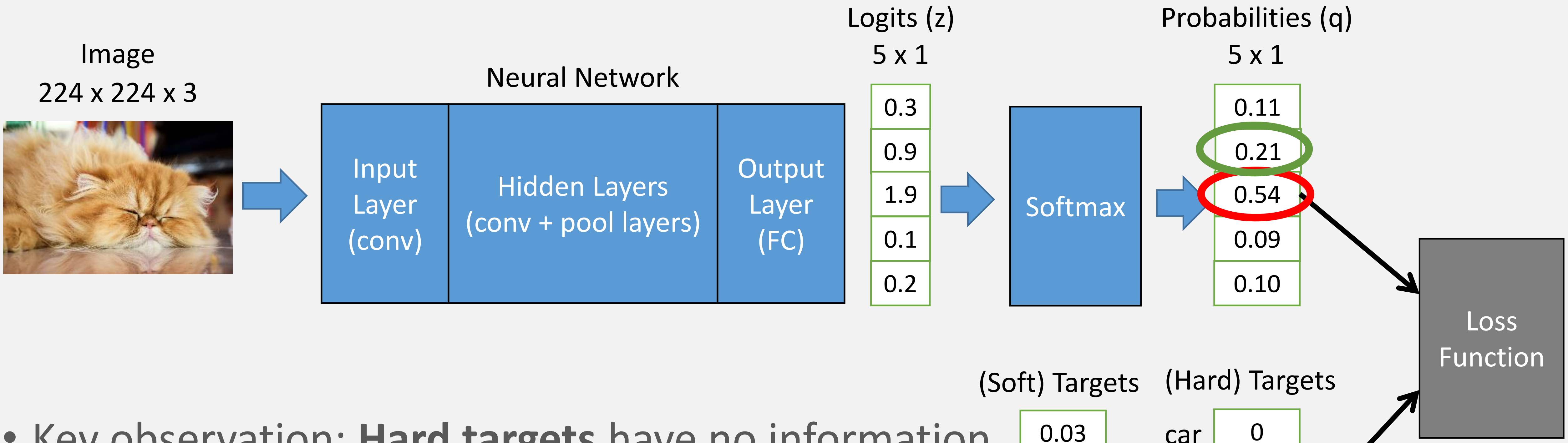
Training a neural network

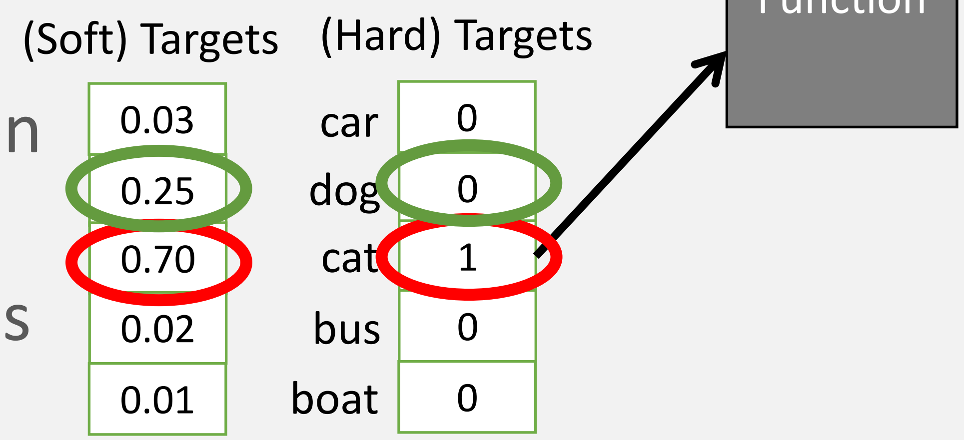




- Key observation: **Hard targets** have no information about wrong classes

Training a neural network



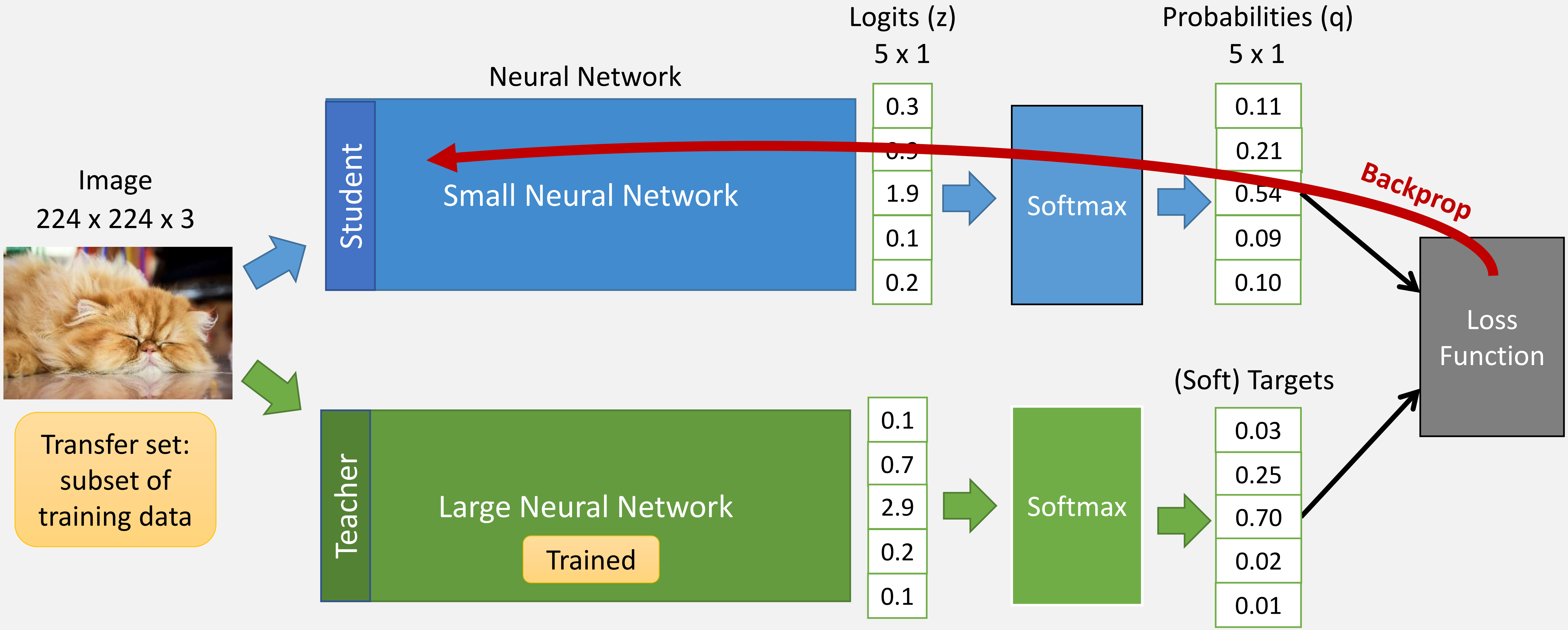


- Key observation: **Hard targets** have no information about wrong classes
- **Soft targets** have information about wrong classes
  - Where do we get them from?

Where do we get soft targets?

1. From labeled data by clustering similar classes
2. Using an equation
3. From a trained neural network
4. Using expert annotation

Knowledge Distillation

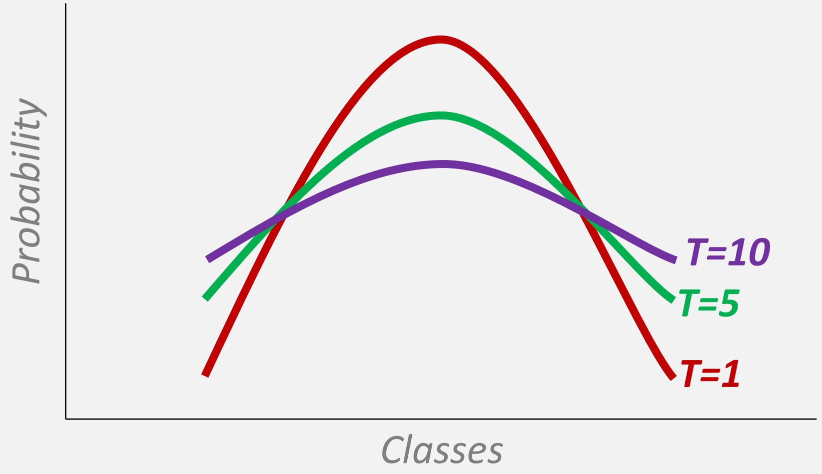


Softmax Temperature

Recall: Softmax function: $$q_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

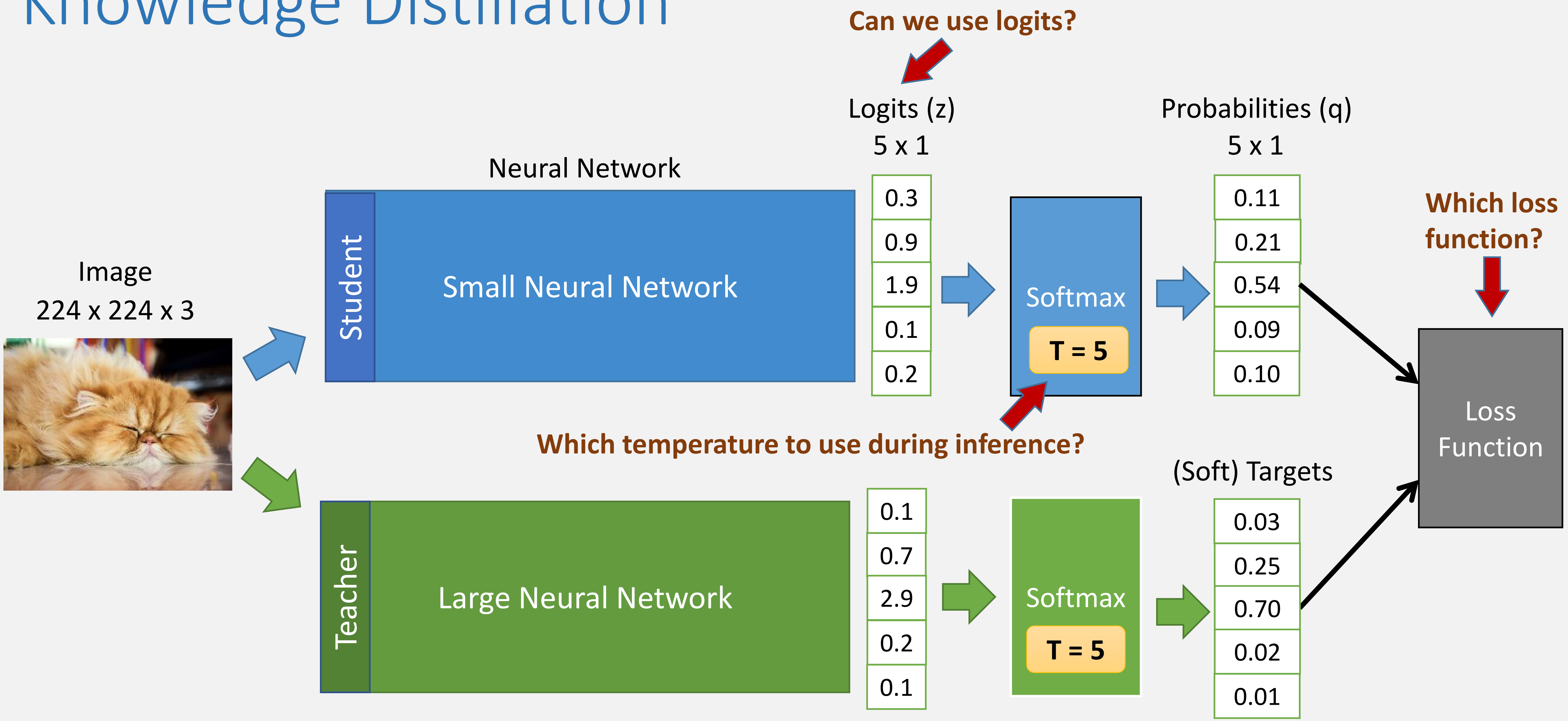
Softmax with temperature: $$q_i = \frac{e^{z_i / T}}{\sum_j e^{z_j / T}}$$

z	q [T=1]	q [T=10]
-1.1	0.007	0.171
1.4	0.087	0.219
3.7	0.880	0.276
0.1	0.024	0.193
-3.0	0.001	0.141



Additional hyper-parameter: **T** → exposes more "dark knowledge"

Knowledge Distillation



What temperature to use during inference?

1. Same as training (T = 5)
2. T = 1 (Standard softmax)
3. T = 0
4. T = 10

Distillation Loss + Student Loss

$$L(x;W) = \boldsymbol{\alpha} * H\big(y, \sigma(z_s; T=1)\big) + \boldsymbol{\beta} * H\big(\sigma(z_t; T=\boldsymbol{t}), \sigma(z_s; T=\boldsymbol{t})\big)$$

Image
224 x 224 x 3

Neural Network

Student: Small Neural Network

Teacher: Large Neural Network

Softmax T = 1

Hard prediction

0.02
0.11
0.84
0.01
0.02

Hard labels

0	0	1	0	0

Loss Function

Student Loss

Softmax T = 5

Soft predictions

0.11
0.21
0.54
0.09
0.10

Softmax T = 5

(Soft) Targets

0.03
0.25
0.70
0.02
0.01

Loss Function

Distillation Loss

Distillation Loss + Student Loss

$$\begin{aligned} L(x;W) = \; & \boldsymbol{\alpha} * H\big(y, \sigma(z_s; T = 1)\big) \\ & + \boldsymbol{\beta} * H(\sigma(z_t; T = \boldsymbol{t}), \sigma(z_s; T = \boldsymbol{t})) \end{aligned}$$

Student loss

Distillation loss

$x: input$
$W: student\ model\ parameters$
$H: loss\ function$
$y: hard\ targets$
$\sigma: softmax\ function$
$T: softmax\ temperature$
$z_s: student\ soft\ predictions$
$z_t: teacher\ soft\ targets$

```python
import torch
import torch.nn.functional as F
import torchvision

# load model + init optimizer
model = torchvision.models.mobilenet_v2()
optimizer = torchvision.optim.SGD(net.parameters(), lr=0.01, momentum=0.9)

for epoch in range(NUM_EPOCHS):

    for inputs,labels in trainloader:

        # forward
        outputs = net(inputs)

        # loss
        loss = F.cross_entropy(outputs, labels)

        # backward + optimize
        loss.backward()
        optimizer.step()
```

```python
# load model + init optimizer
student = torchvision.models.mobilenet_v2()
optimizer = torchvision.optim.SGD(net.parameters(), lr=0.01, momentum=0.9)
```

Load teacher + set hyper-params

```python
for epoch in range(NUM_EPOCHS):
    for inputs,labels in trainloader:
        # forward
        outputs_s = student(inputs)
```

Compute loss

```python
        # backward + optimize
        loss.backward()
        optimizer.step()
```

```python
# load model + init optimizer
student = torchvision.models.mobilenet_v2()
optimizer = torchvision.optim.SGD(net.parameters(), lr=0.01, momentum=0.9)
teacher = torchvision.models.resnet101() # load teacher model
T, ALPHA = 5, 0.3 # distillation hyper-params

for epoch in range(NUM_EPOCHS):
    for inputs,labels in trainloader:
        # forward
        outputs_s = student(inputs)
        outputs_t = teacher(inputs)  # teacher forward pass

        hard_loss = F.cross_entropy(outputs, labels) # hard loss with G.T. labels

        #distillation loss
        p, q = F.softmax(outputs_s/T, dim=1), F.softmax(outputs_t/T, dim=1)
        dist_loss = F.kl_div(p, q)

        loss = ALPHA * dist_loss + (1. - ALPHA) * hard_loss # combined hard + distillation loss

        # backward + optimize
        loss.backward()
        optimizer.step()
```

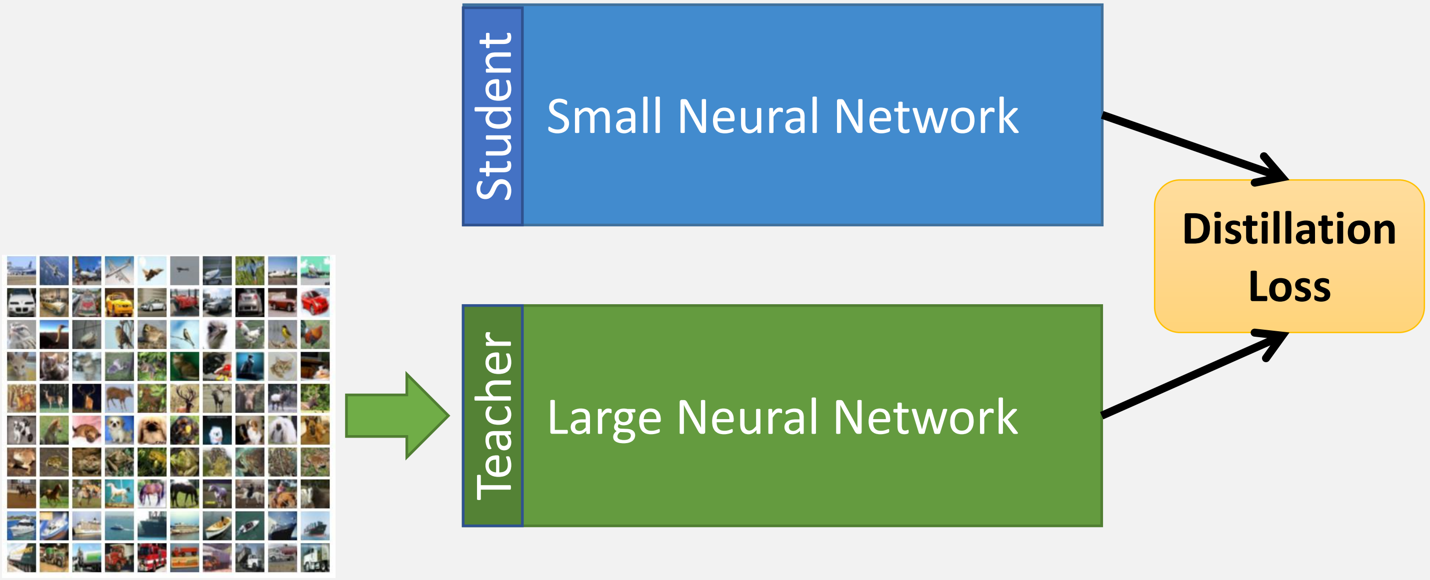
Learning Outcomes

Understand what knowledge distillation is and why it is needed.

Write code to perform knowledge distillation

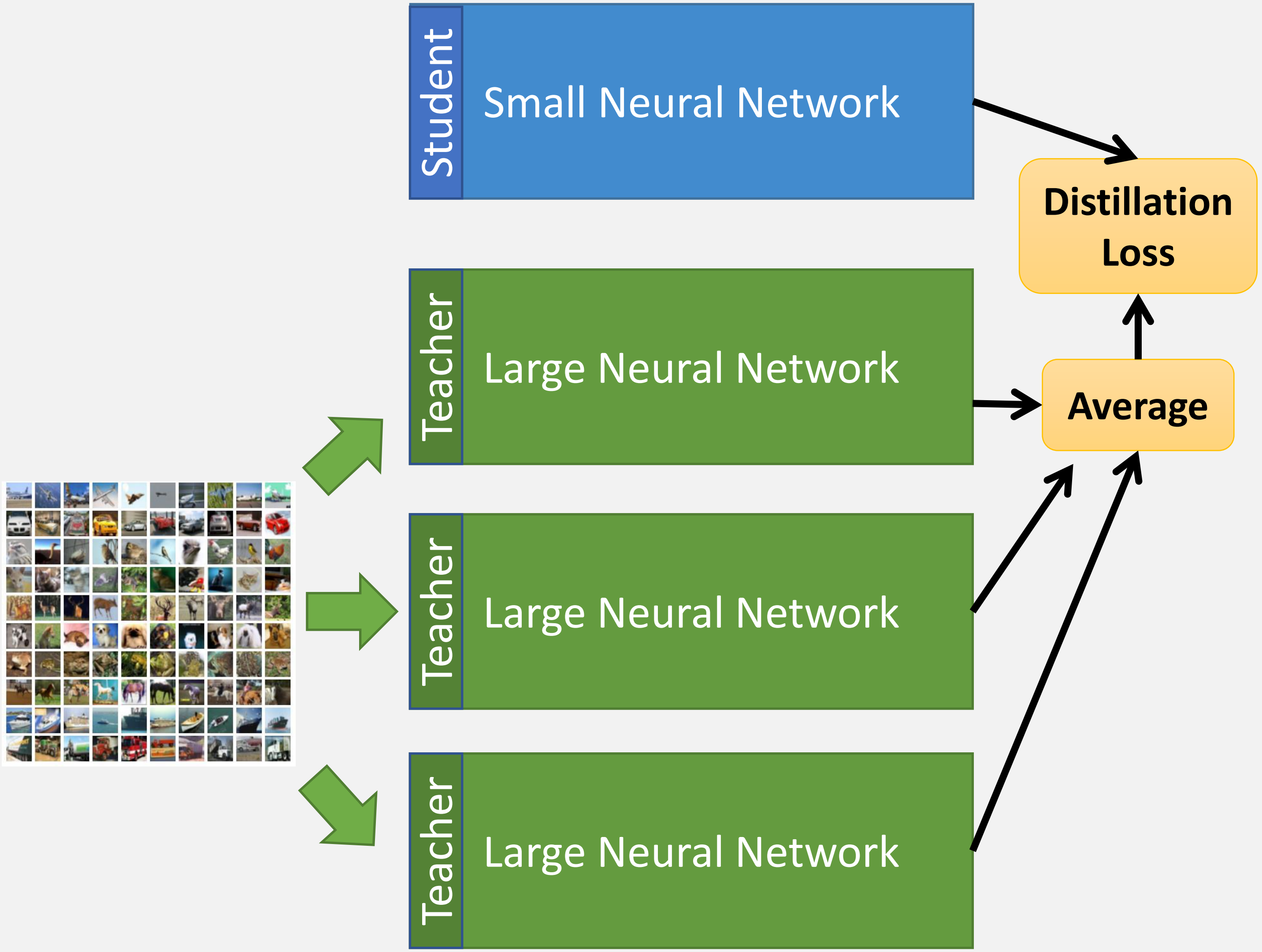
Understand advanced knowledge distillation techniques and open problems

Ensembles and Specialists



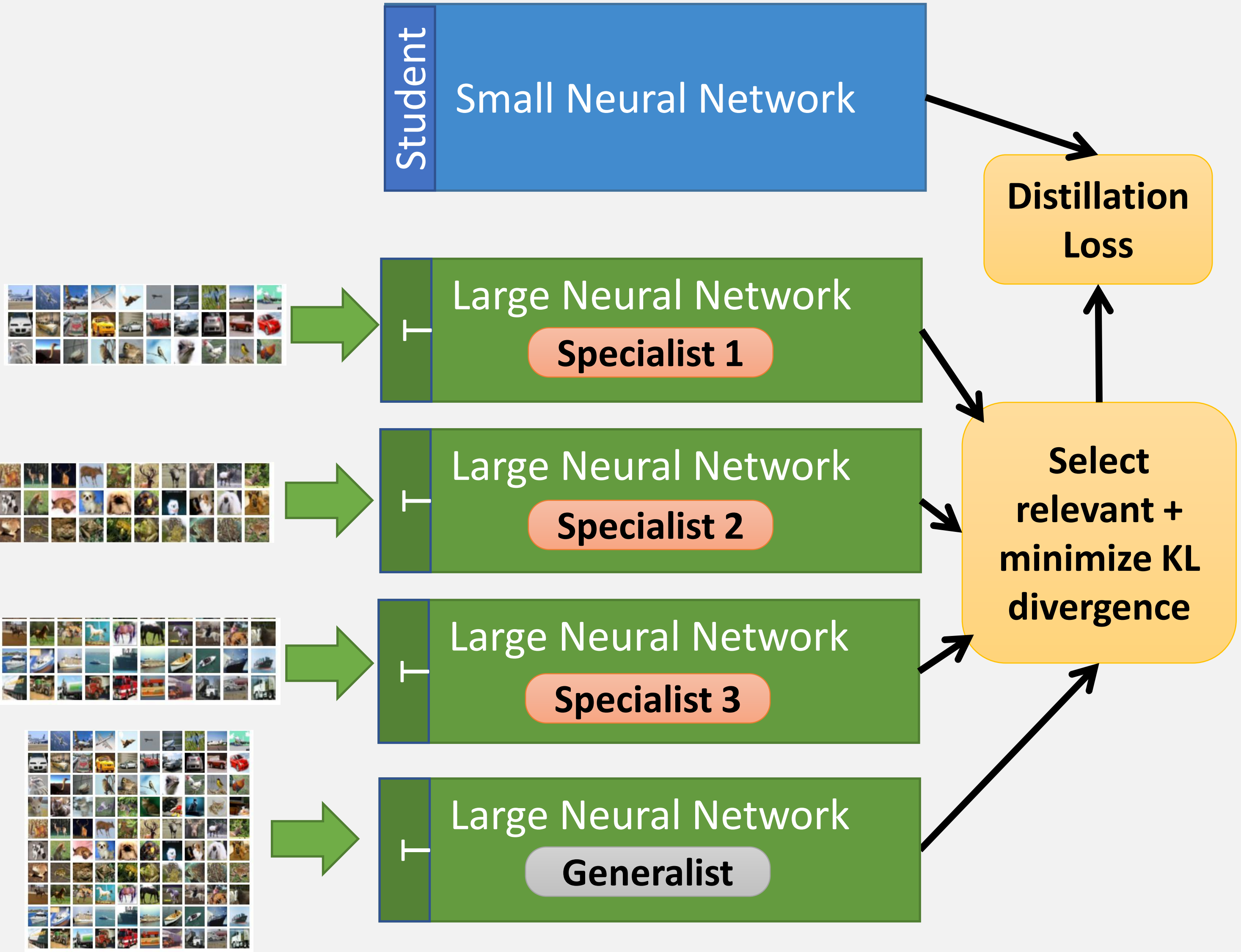
- Teacher architecture can be more complicated – boost accuracy

Ensembles and Specialists



- Teacher architecture can be more complicated – boost accuracy
- Ensembles:
  - Different initializations
  - Different model architectures

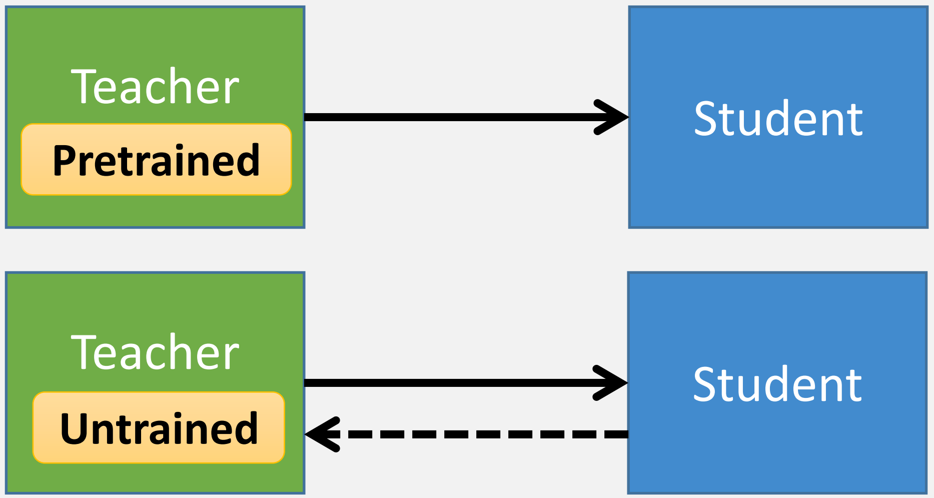
Ensembles and Specialists



- Teacher architecture can be more complicated – boost accuracy
- Ensembles:
  - Different initializations
  - Different model architectures
- Specialists [1]:
  - Divide classes to different model
    - Google JFT dataset has 15000 classes
  - One generalist NN on all data
  - Top-k classes from generalist are further *refined* by specialists
  - How to choose specialist classes?

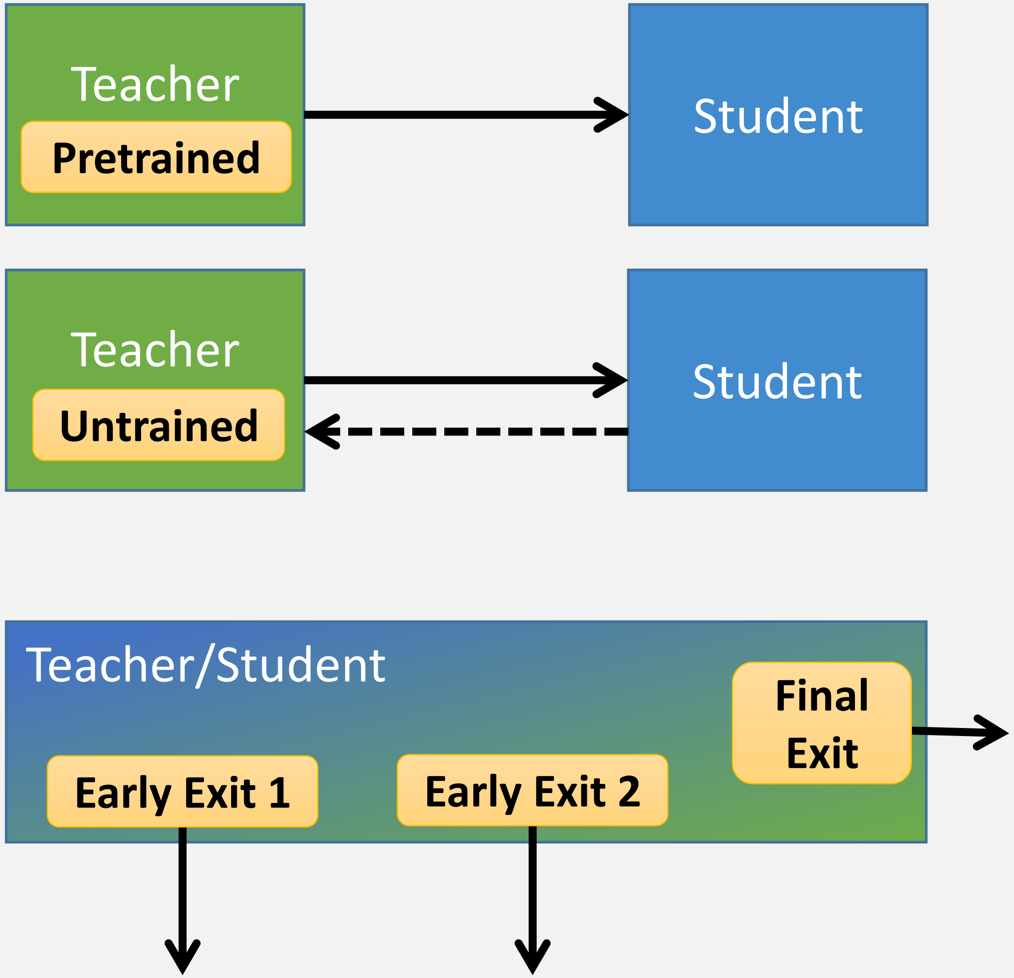
Distillation Types

- Offline: Pretrained teacher used to add distillation loss during student training
- Online: Both teacher and student are trained simultaneously
  - Collaborative/mutual learning



Distillation Types

- Offline: Pretrained teacher used to add distillation loss during student training
- Online: Both teacher and student are trained simultaneously
  - Collaborative/mutual learning
- Self distillation:
  - E.g. Progressive hierarchical inference



Knowledge Types

- Response-based:
  - Output probabilities as soft targets (as we have already seen)

$$L_{ResD}(z_t, z_s) = \mathcal{L}_R(z_t, z_s)$$

- Feature-based:
  - Output/weights of 1 or more “hint layers” and minimize e.g. MSE loss
  - More advanced: minimize difference in attention maps between student/teacher

$$L_{FeaD}\big(f_t(x), f_s(x)\big) = \mathcal{L}_F\big(\Phi_t(f_t(x)), \Phi_s(f_s(x))\big)$$

- Relation-based:
  - Correlations between feature maps: e.g. Gramian between two features

$$L_{RelD}(f_t, f_s) = \mathcal{L}_{R^1}\big(\Psi_t(\hat{f}_t, \check{f}_t), \Psi_s(\hat{f}_s, \check{f}_s)\big)$$

Distillation Algorithms

- Survey paper [2] has a thorough overview of different distillation variations
- Adverserial: Teacher also acts as discriminator in GAN to supplement training data to "teach" true data distribution
- Cross-modal: Teacher trained on RGB distills information to student learning on heat maps. Unlabeled image pairs needed.
- Quantized distillation: Use full-precision network to transfer knowledge to quantized network.
- … more in the paper!

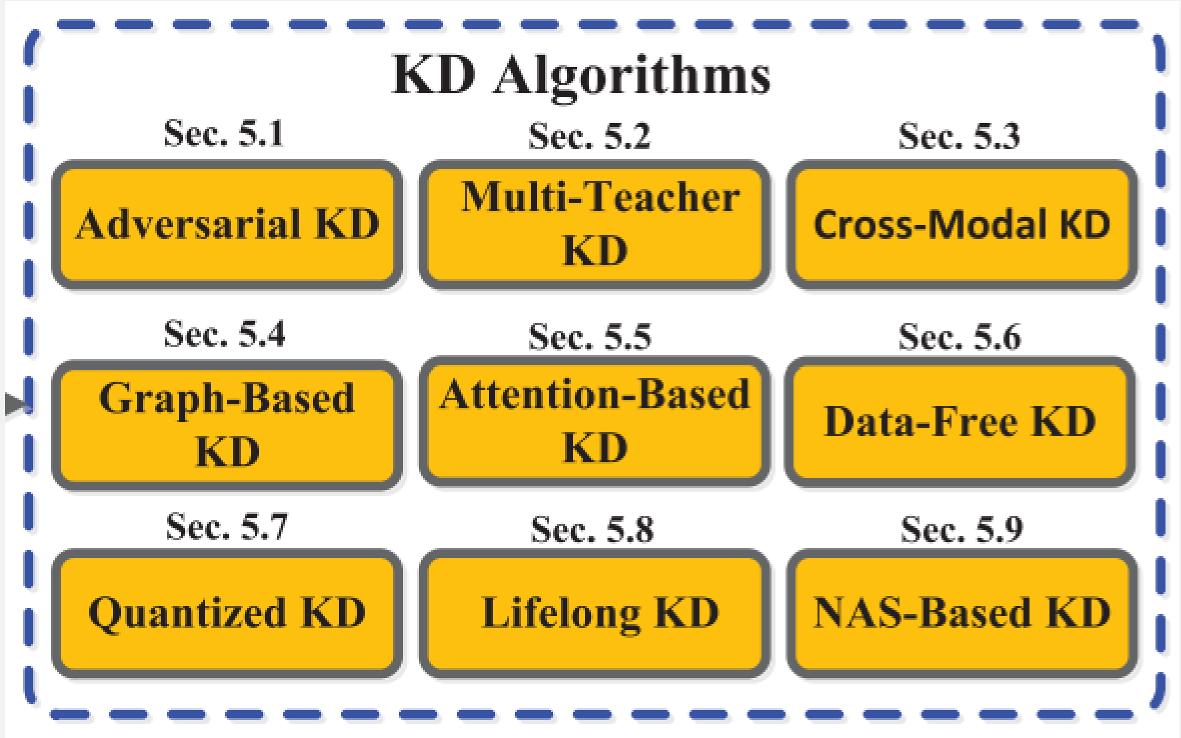


Image from [2]

Reading

[1] G. Hinton et al., **Distilling the knowledge in a neural network**, *Neural Information Processing Systems (NeurIPS)*, 2015.

[2] J. Guo et al., **Knowledge Distillation: A Survey**, *ArXiV preprint*, 2021.

Learning Outcomes

Understand what knowledge distillation is and why it is needed.

Write code to perform knowledge distillation

Understand advanced knowledge distillation techniques and open problems

Greater Impact

- Environmental impact: Smaller models, lower carbon footprint
  - Beware Jevons paradox!
- Data privacy: Smaller models, on-device AI, data stays on your device
- If you are Google:
  - Power bill $$
- If you are Samsung:
  - AWS bill $$


That's all Folks!