AI VIETNAM
VLAI Seminar

Knowledge Distillation

Source: https://pytorch.org/tutorials/beginner/knowledge_distillation_tutorial.html

Dinh-Thang Duong

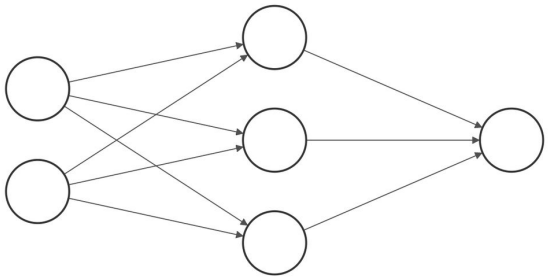
Outline

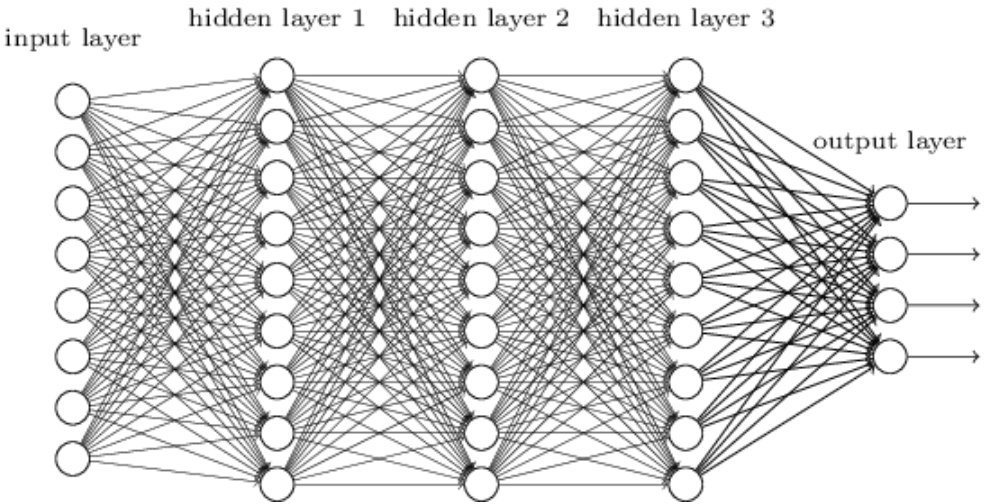
- Introduction
- Knowledge Distillation
- Experimental Results
- Question

Introduction

Introduction

❖ Deep Learning

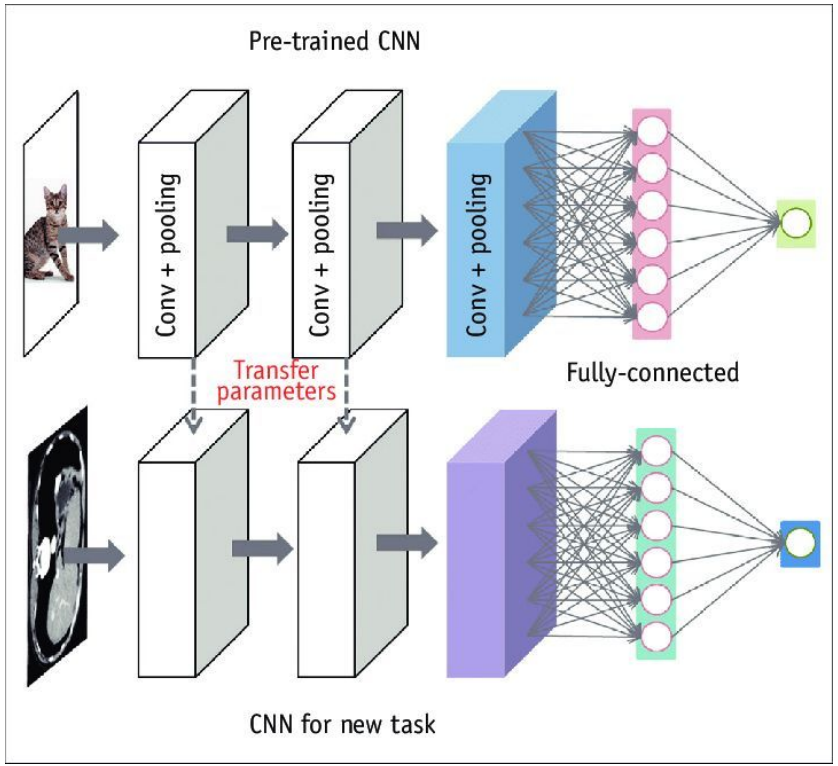




https://www.excella.com/insights/top-3-most-popular-neural-networks
https://medium.com/make-computer-science-fun-again/learning-from-small-neural-networks-6bc5ffc2f3d3

Introduction

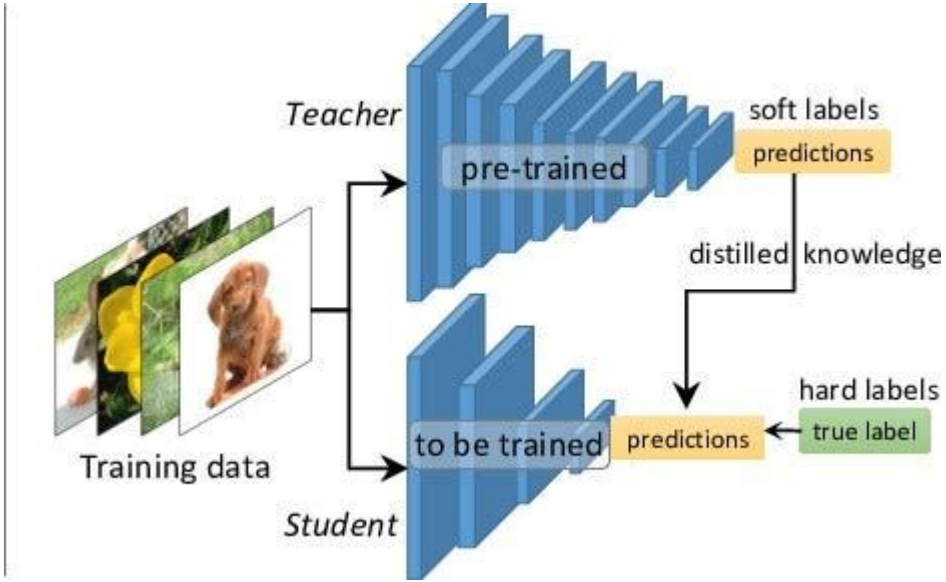
- **Transfer learning**



https://in.pinterest.com/pin/923730573552168592/

Introduction

❖ Knowledge Distillation



Knowledge distillation: A technique in machine learning where a smaller, less complex model (the Student) is trained to replicate the behavior of a larger, more complex model (the Teacher).

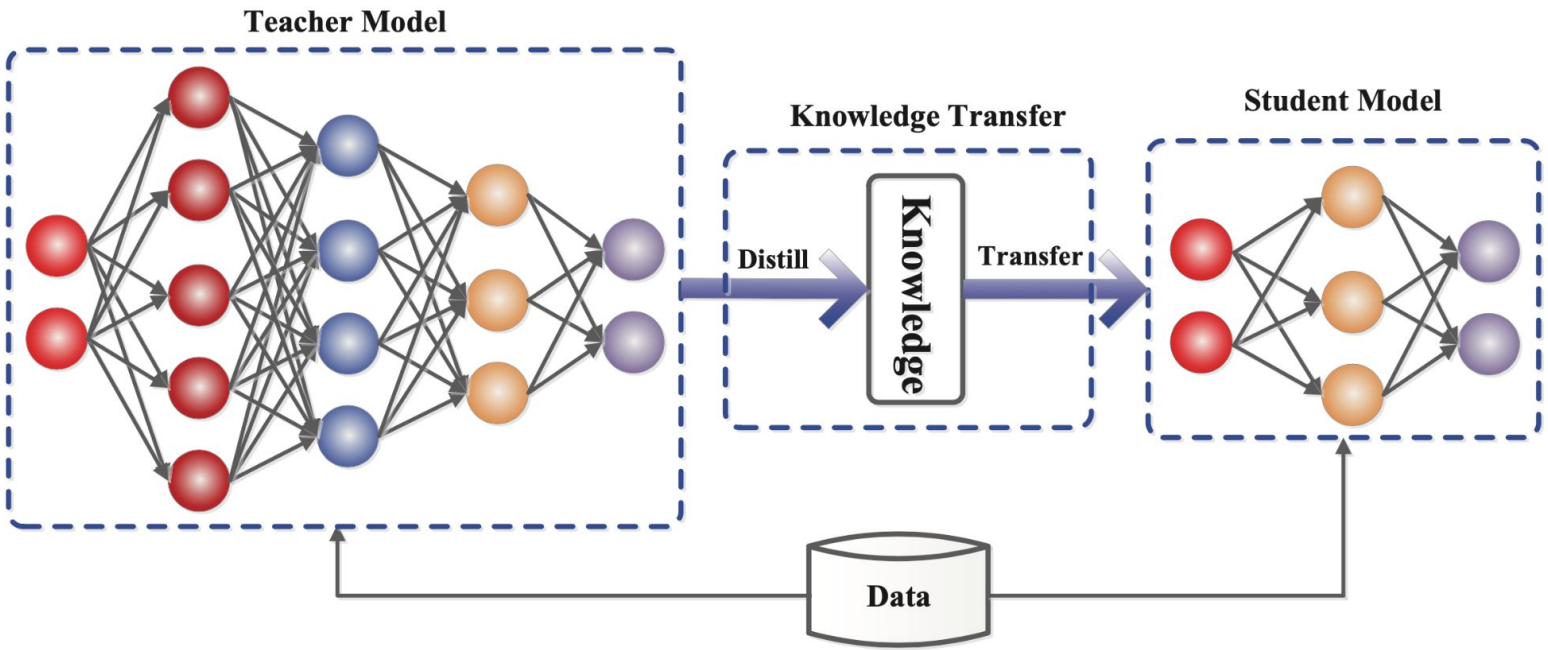
This process involves transferring the knowledge encoded in the Teacher model's predictions and internal representations to the Student model, enabling it to achieve similar performance with reduced computational requirements.

https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

Knowledge Distillation

Knowledge Distillation

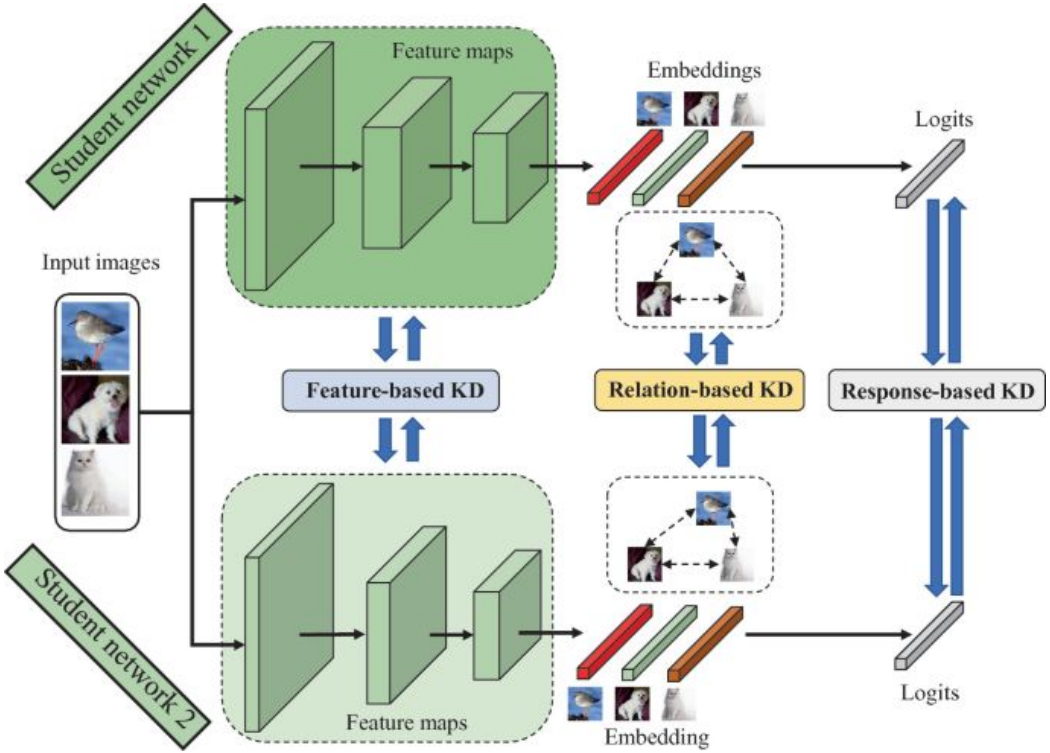
❖ KD Framework



https://neptune.ai/blog/knowledge-distillation

Knowledge Distillation

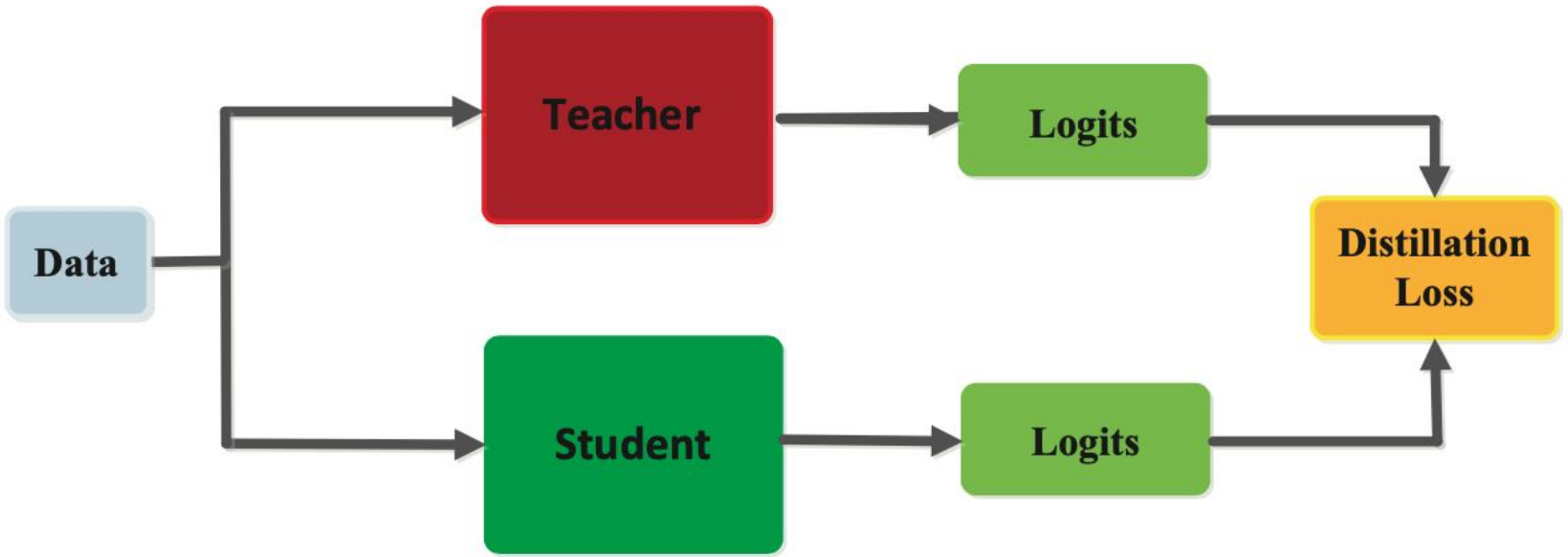
❖ Neural Networks Knowledge



https://link.springer.com/chapter/10.1007/978-3-031-32095-8_1

Knowledge Distillation

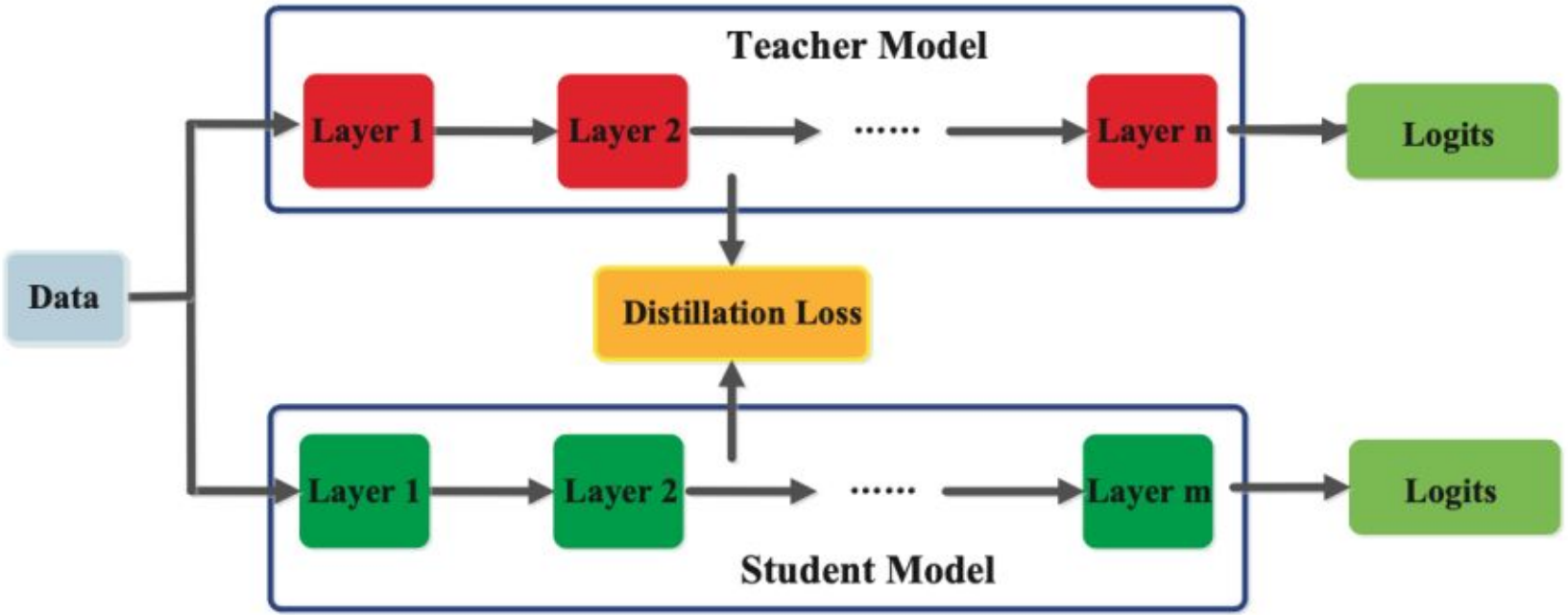
❖ Neural Networks Knowledge: Response-based



https://neptune.ai/blog/knowledge-distillation

Knowledge Distillation

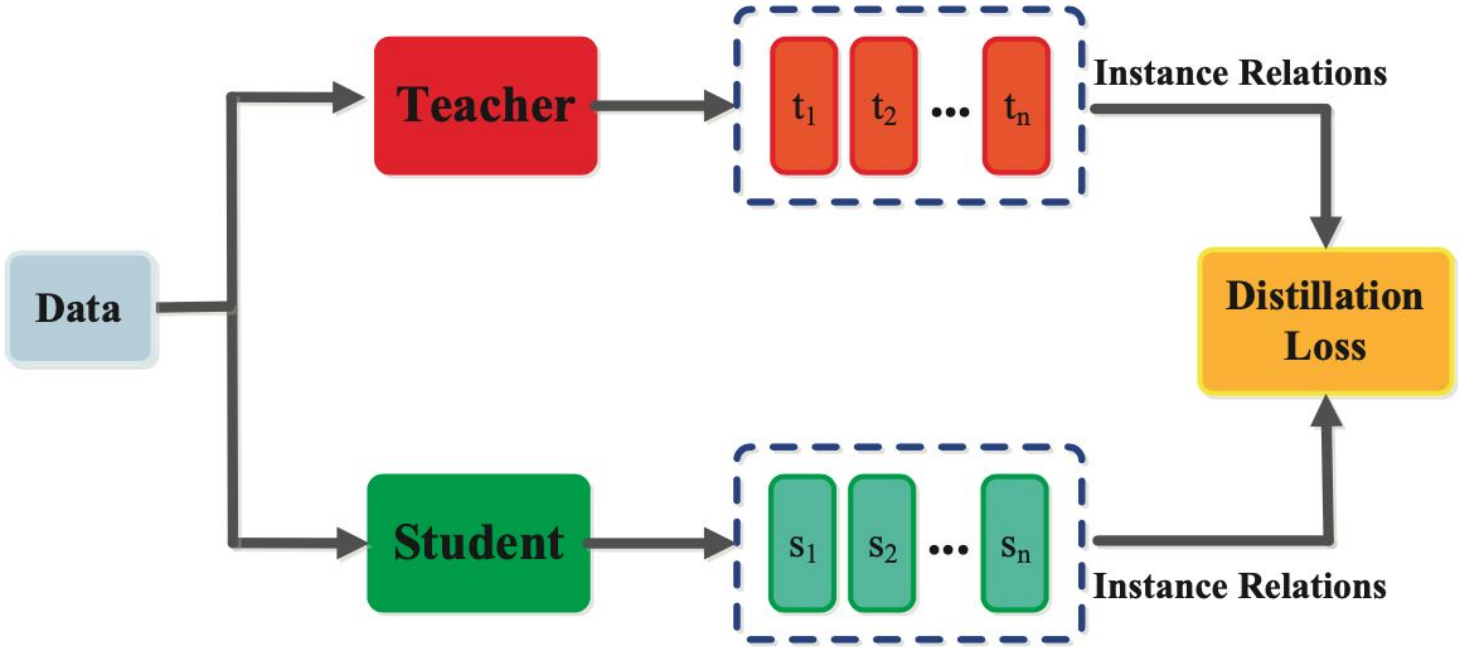
❖ Neural Networks Knowledge: Feature-based



https://neptune.ai/blog/knowledge-distillation

Knowledge Distillation

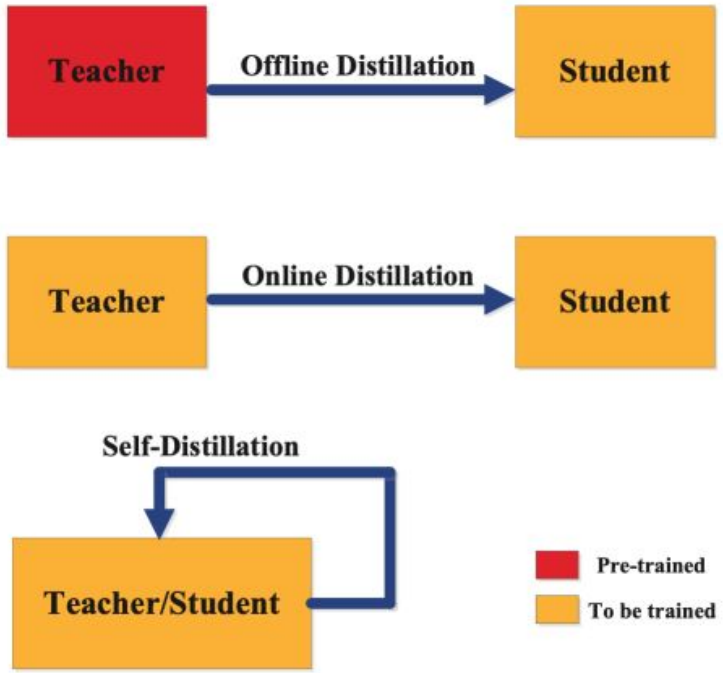
❖ Neural Networks Knowledge: Relation-based



https://neptune.ai/blog/knowledge-distillation

Knowledge Distillation

❖ Training KD

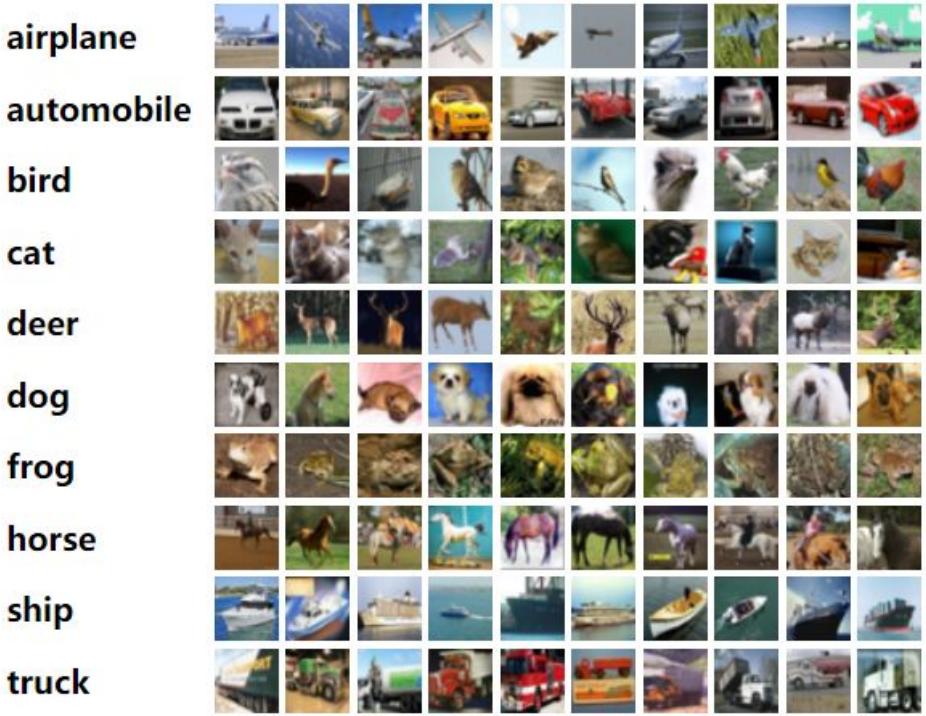


https://neptune.ai/blog/knowledge-distillation

Experimental Results

Experimental Results

❖ Experimental Setup: CIFAR10

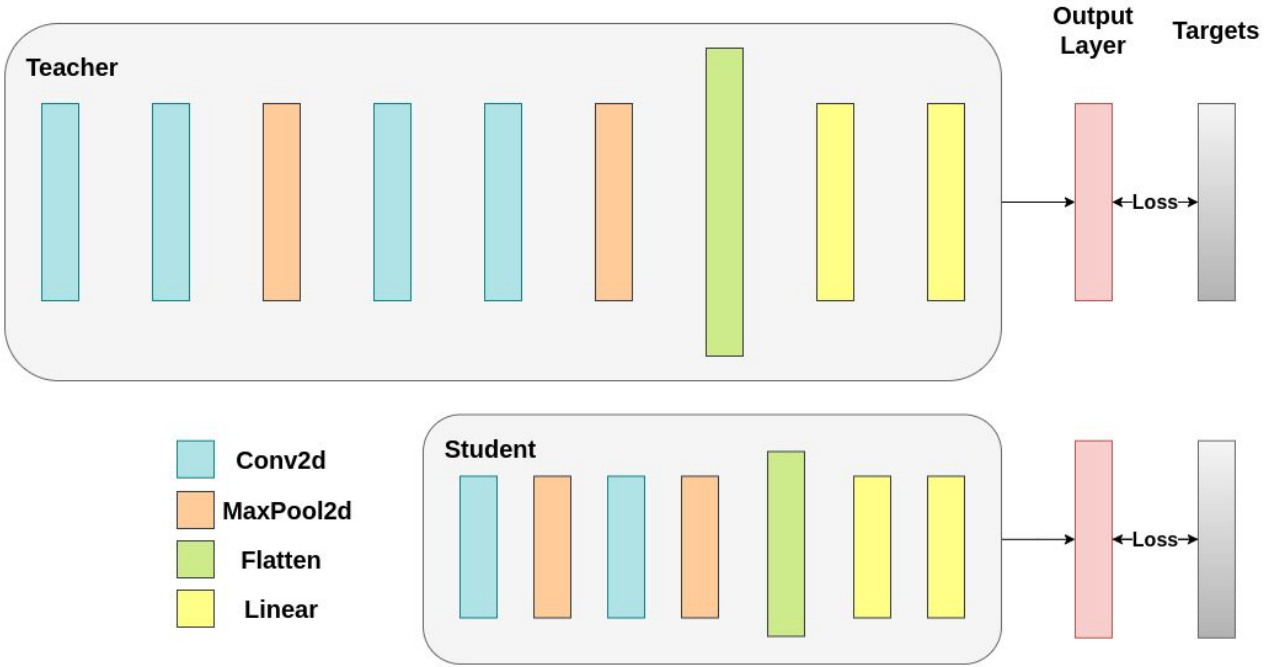


- The CIFAR-10 dataset (Canadian Institute for Advanced Research, 10 classes) is a subset of the Tiny Images dataset and consists of 60000 32x32 color images.
- The images are labelled with one of 10 mutually exclusive classes.
- There are 6000 images per class with 5000 training and 1000 testing images per class.

https://paperswithcode.com/dataset/cifar-10

Experimental Results

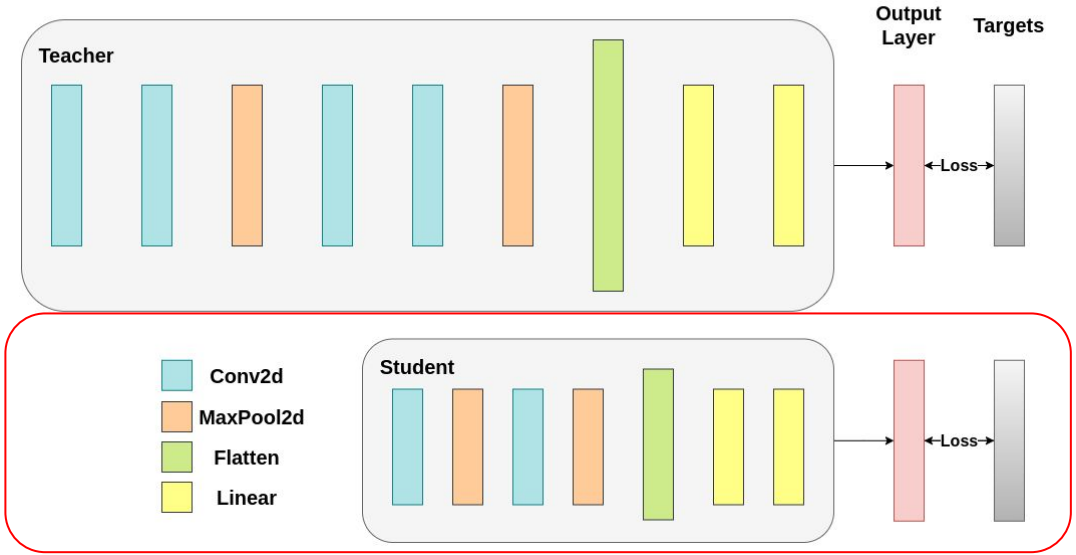
❖ Experimental Setup



Experimental Results

❖ Experimental Setup: Student model

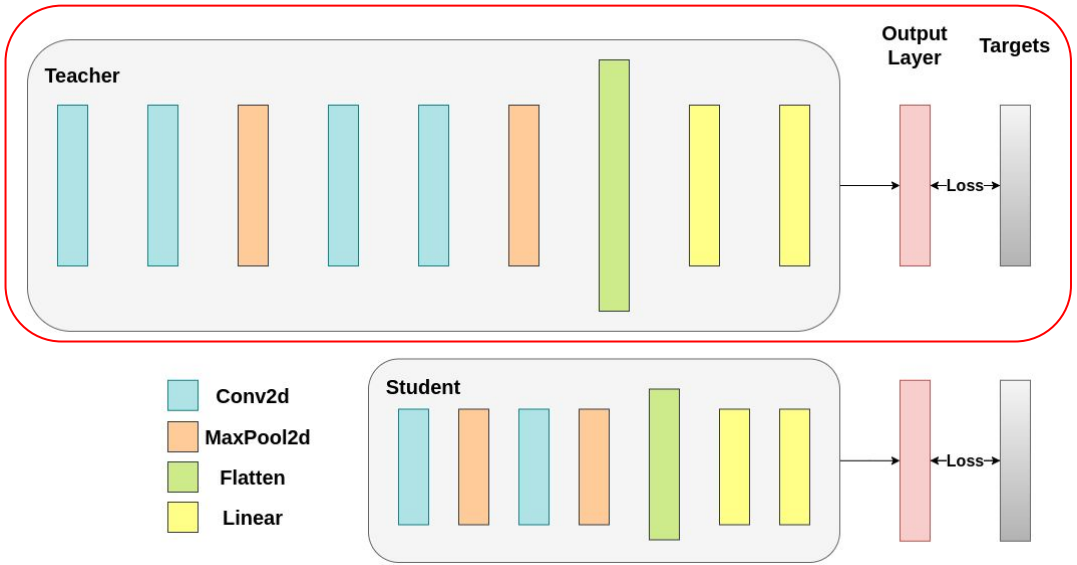
```python
class StudentCNN(nn.Module):
    def __init__(self, n_classes):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 16, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(16, 16, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.classifier = nn.Sequential(
            nn.Linear(1024, 256),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.Linear(256, n_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)

        return x
```



Experimental Results

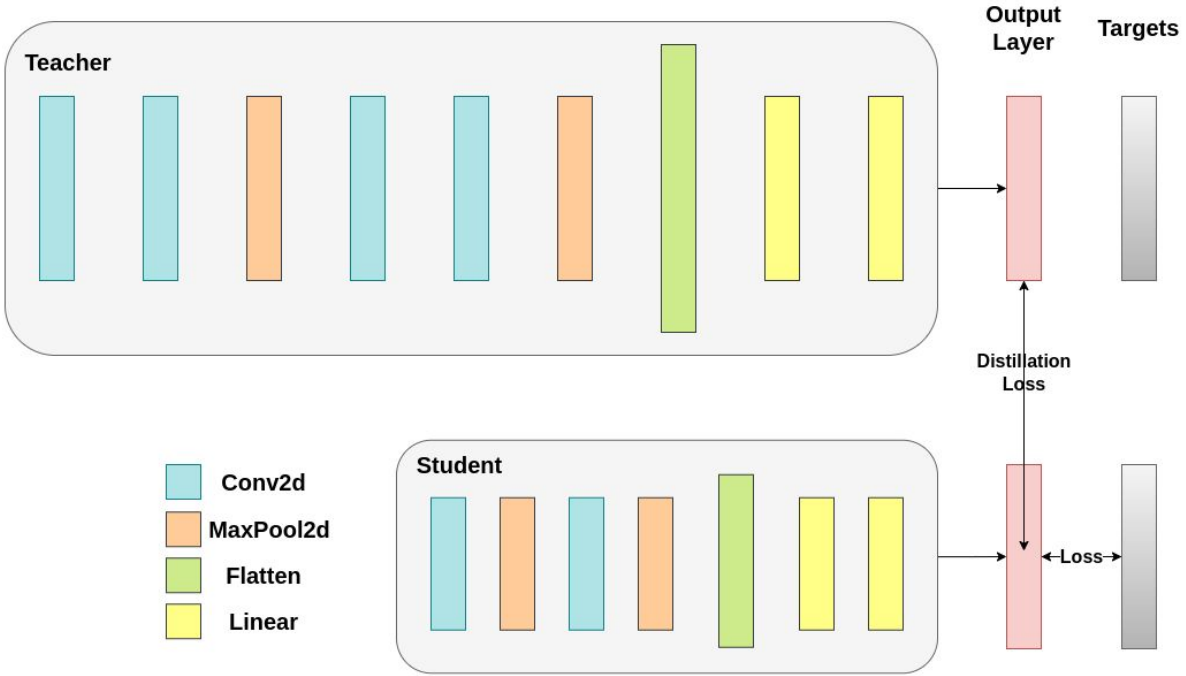
❖ Experimental Setup: Student model

```python
class TeacherCNN(nn.Module):
    def __init__(self, n_classes):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(128, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(64, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(64, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.classifier = nn.Sequential(
            nn.Linear(2048, 512),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.Linear(512, n_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)

        return x
```



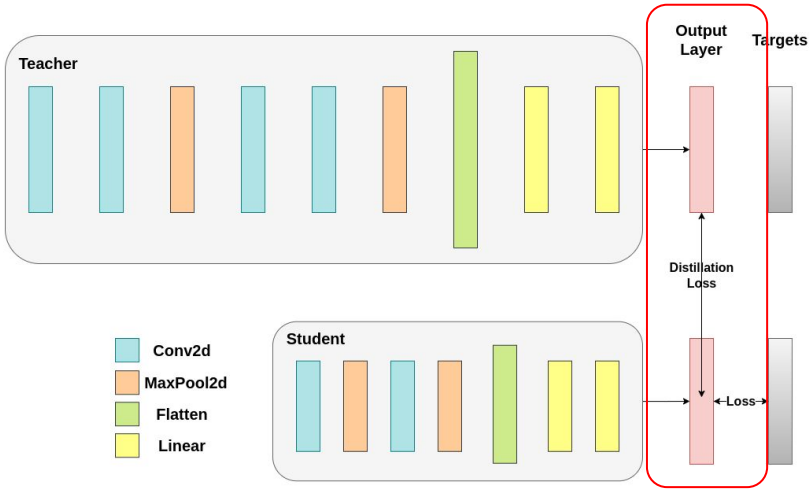
Experimental Results

❖ Experimental Setup: KD

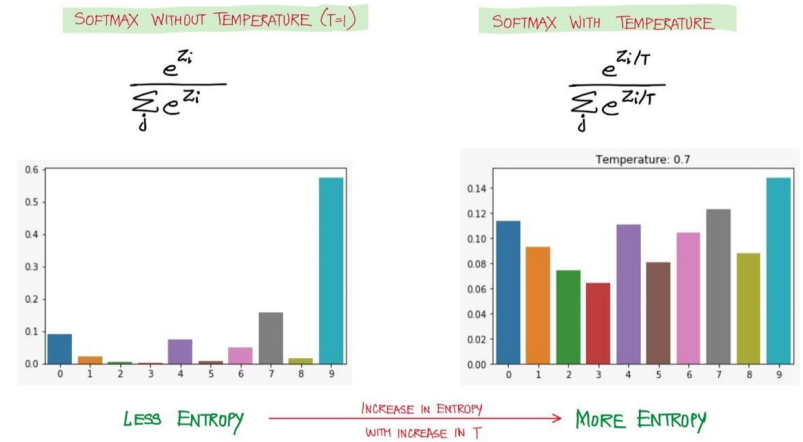


Experimental Results

❖ Experimental Setup: KD



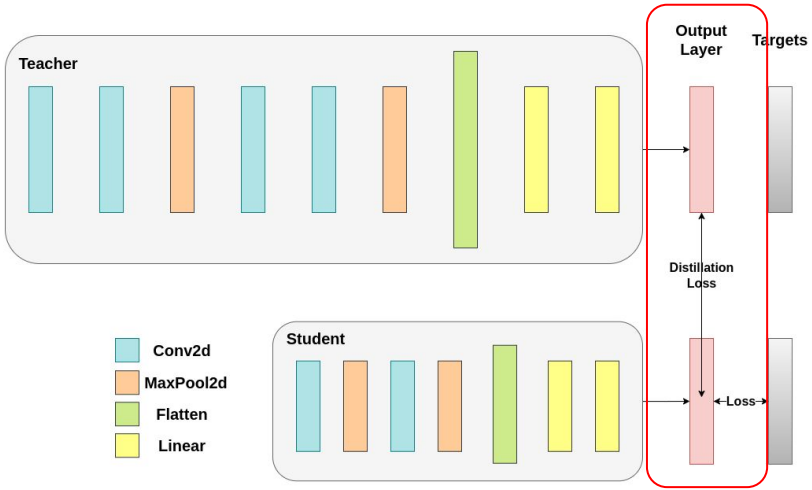
Hard targets	0	1	0	0
	cow	dog	cat	car
Soft targets	10^{-6}	0.9	0.1	10^{-9}



https://medium.com/@harshit158/softmax-temperature-5492e4007f71

Experimental Results

❖ Experimental Setup: KD

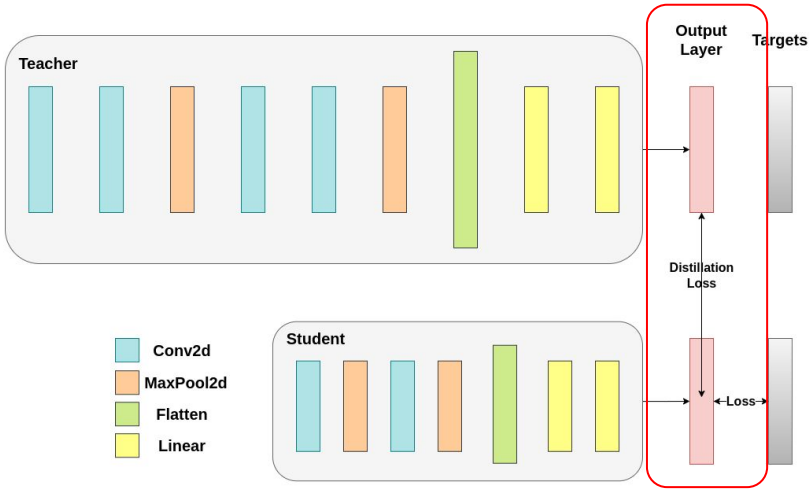


$$\text{Loss} = \alpha \cdot \text{CrossEntropy}(y,p) + (1-\alpha) \cdot T2 \cdot \text{KL}(q/T, p/T)$$

https://medium.com/@harshit158/softmax-temperature-5492e4007f71

Experimental Results

❖ Experimental Setup: KD



$$KL(P||Q) = \frac{\sum_x P(x) log \frac{P(x)}{Q(x)}}{n}$$

KL-Divergence Loss

https://medium.com/@harshit158/softmax-temperature-5492e4007f71

Experimental Results

Implementation

```
# Forward pass with the teacher model - do not save gradients here as we do not change the teacher's weights
with torch.no_grad():
    teacher_logits = teacher(inputs)

# Forward pass with the student model
student_logits = student(inputs)

#Soften the student logits by applying softmax first and log() second
soft_targets = nn.functional.softmax(teacher_logits / T, dim=-1)
soft_prob = nn.functional.log_softmax(student_logits / T, dim=-1)

# Calculate the soft targets loss. Scaled by T**2 as suggested by the authors of the paper "Distilling the knowledge in a neural network"
soft_targets_loss = torch.sum(soft_targets * (soft_targets.log() - soft_prob)) / soft_prob.size()[0] * (T**2)

# Calculate the true label loss
label_loss = ce_loss(student_logits, labels)

# Weighted sum of the two losses
loss = soft_target_loss_weight * soft_targets_loss + ce_loss_weight * label_loss

loss.backward()
optimizer.step()
```

Experimental Results

- **Implementation**

```
Epoch 1/10, Loss: 2.4240907564455147
Epoch 2/10, Loss: 1.9099933942969964
Epoch 3/10, Loss: 1.7070190565926688
Epoch 4/10, Loss: 1.5723673555315758
Epoch 5/10, Loss: 1.4511103690886984
Epoch 6/10, Loss: 1.364636446748461
Epoch 7/10, Loss: 1.2888724171385473
Epoch 8/10, Loss: 1.2154909116881234
Epoch 9/10, Loss: 1.157678938641 84
Epoch 10/10, Loss: 1.103281553910703
Teacher accuracy: 0.73%
Student accuracy without teacher: 0.66%
Student accuracy with CE + KD: 0.66%
```

Question

THANK YOU