On-Device AI 실습: Knowledge Distillation

Dongkun Shin
Intelligent Embedded Systems Lab.
Sungkyunkwan University

Overview

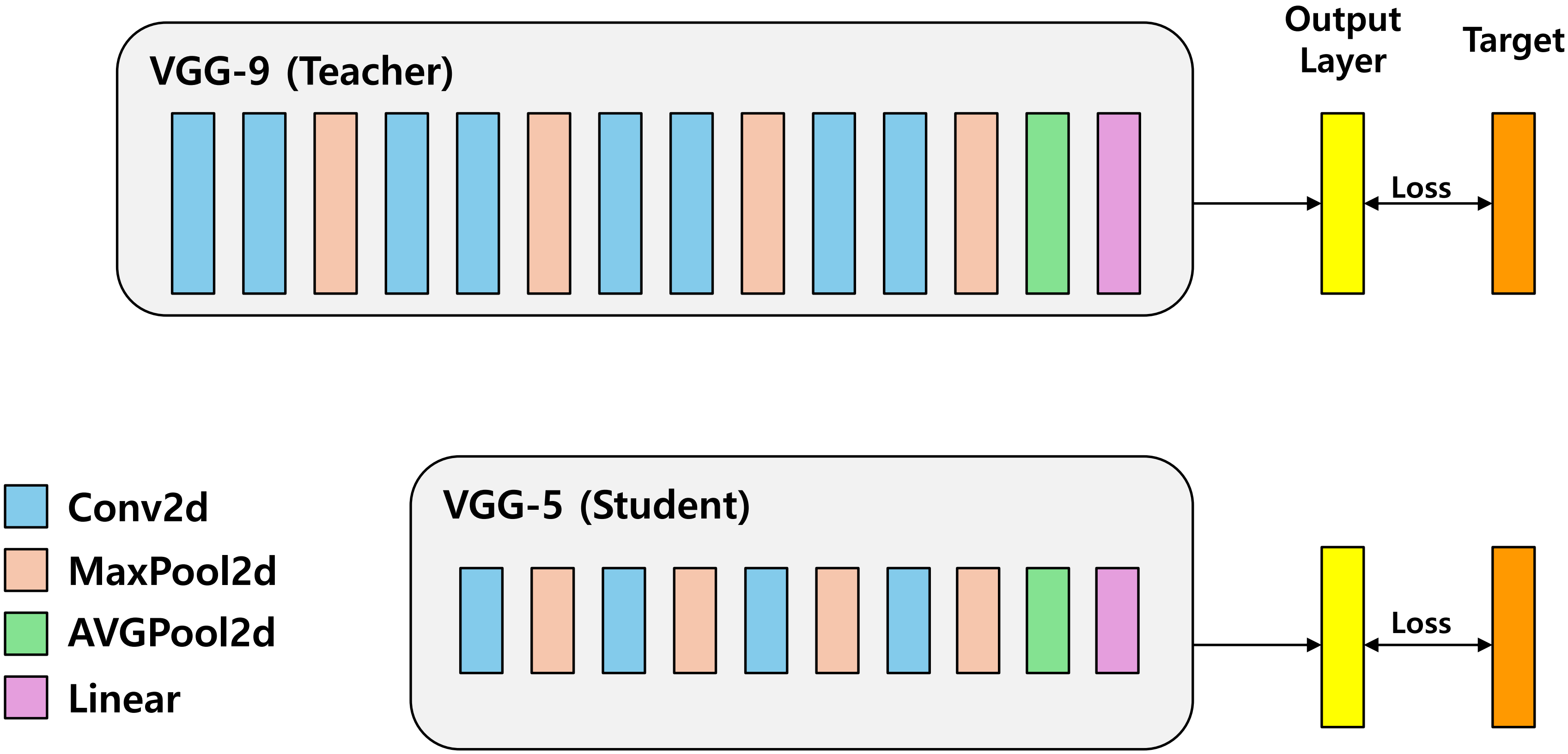
1. **Baseline 학습 (Cross-Entropy Loss)**
   – Teacher 모델과 Student 모델을 각각 Cross-Entropy Loss만으로 학습시켜 정확도를 비교합니다.

2. **Knowledge Distillation (Soft Targets)**
   – Teacher의 softmax 출력을 활용한 Knowledge Distillation을 적용하고, temperature 및 loss weight에 따른 영향을 분석합니다.

3. **Cosine Loss Minimization (Cosine Loss)**
   – Teacher와 Student의 convolutional feature를 추출하여, CosineEmbeddingLoss를 적용해 내부 표현 유사도를 증가시키는 방식으로 학습합니다.

4. **Intermediate Regressor (Regressor + MSE)**
   – Teacher의 feature map과 Student의 regressed feature map을 MSE로 정렬하며, 중간 표현을 직접 학습합니다.

Baseline 학습 (Cross-Entropy Loss)



Cross-Entropy Loss

https://docs.pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

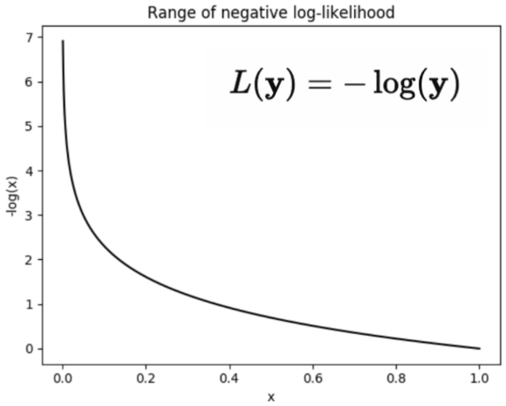
- **Definition:**

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} \log \boxed{\frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})}} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

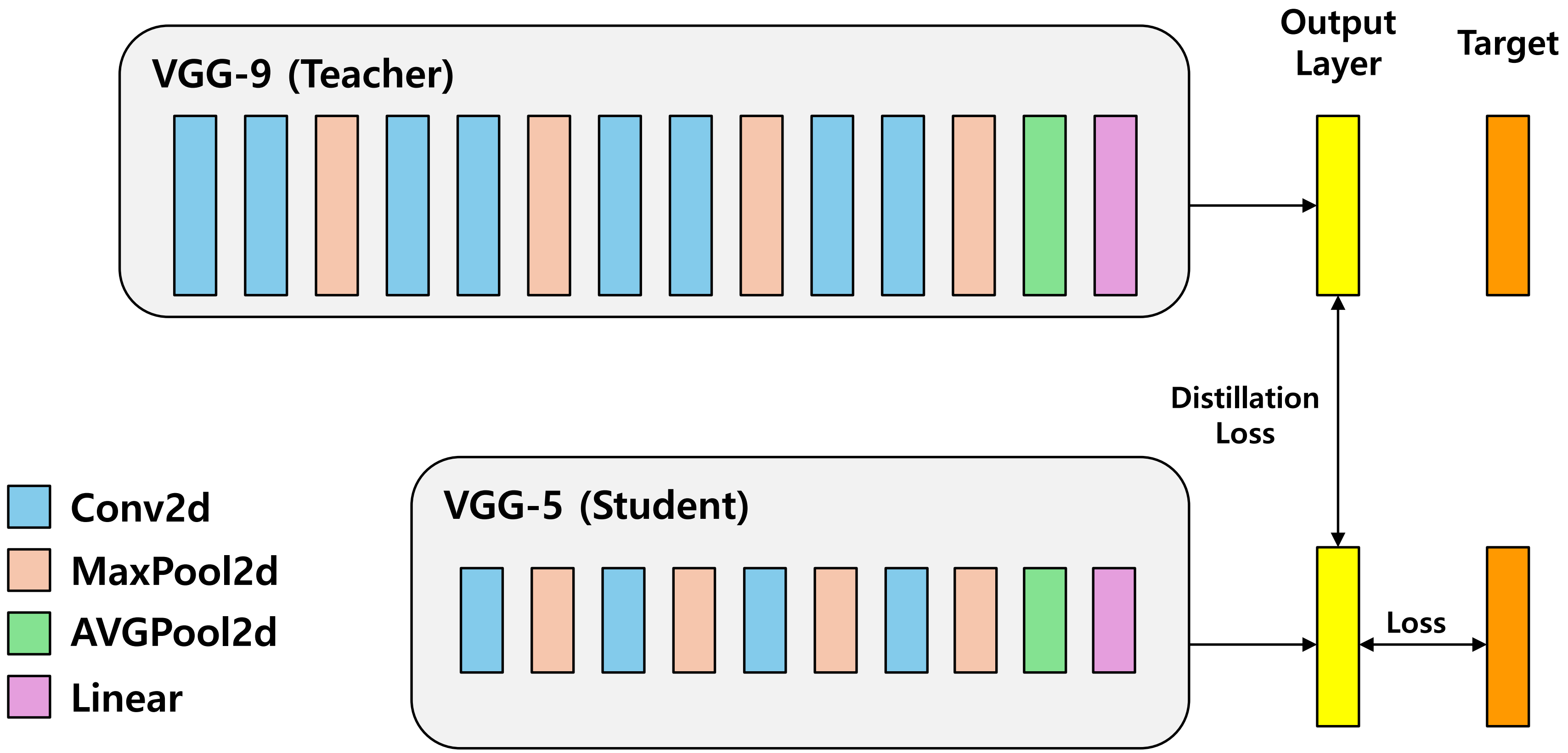
softmax

- **Example: nn.CrossEntropyLoss**

```
>>> # Example of target with class indices
>>> loss = nn.CrossEntropyLoss()
>>> input = torch.randn(3, 5, requires_grad=True)
>>> target = torch.empty(3, dtype=torch.long).random_(5)
>>> output = loss(input, target)
>>> output.backward()
>>>
>>> # Example of target with class probabilities
>>> input = torch.randn(3, 5, requires_grad=True)
>>> target = torch.randn(3, 5).softmax(dim=1)
>>> output = loss(input, target)
>>> output.backward()
```



Knowledge Distillation (Soft Targets)



KL (Kullback-Leibler) Divergence Loss

https://docs.pytorch.org/docs/stable/generated/torch.nn.KLDivLoss.html

- **Definition:** $D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}$

$$L(y_{\text{pred}},\ y_{\text{true}}) = y_{\text{true}} \cdot \log \frac{y_{\text{true}}}{y_{\text{pred}}} = y_{\text{true}} \cdot (\log y_{\text{true}} - \log y_{\text{pred}})$$

- **Example: nn.KLDivLoss**

```
>>> kl_loss = nn.KLDivLoss(reduction="batchmean")
>>> # input should be a distribution in the log space
>>> input = F.log_softmax(torch.randn(3, 5, requires_grad=True), dim=1)
>>> # Sample a batch of distributions. Usually this would come from the dataset
>>> target = F.softmax(torch.rand(3, 5), dim=1)
>>> output = kl_loss(input, target)
>>>
>>> kl_loss = nn.KLDivLoss(reduction="batchmean", log_target=True)
>>> log_target = F.log_softmax(torch.rand(3, 5), dim=1)
>>> output = kl_loss(input, log_target)
```

Distillation Loss 계산

- **For Knowledge Distillation loss**
  - Use temperature: T

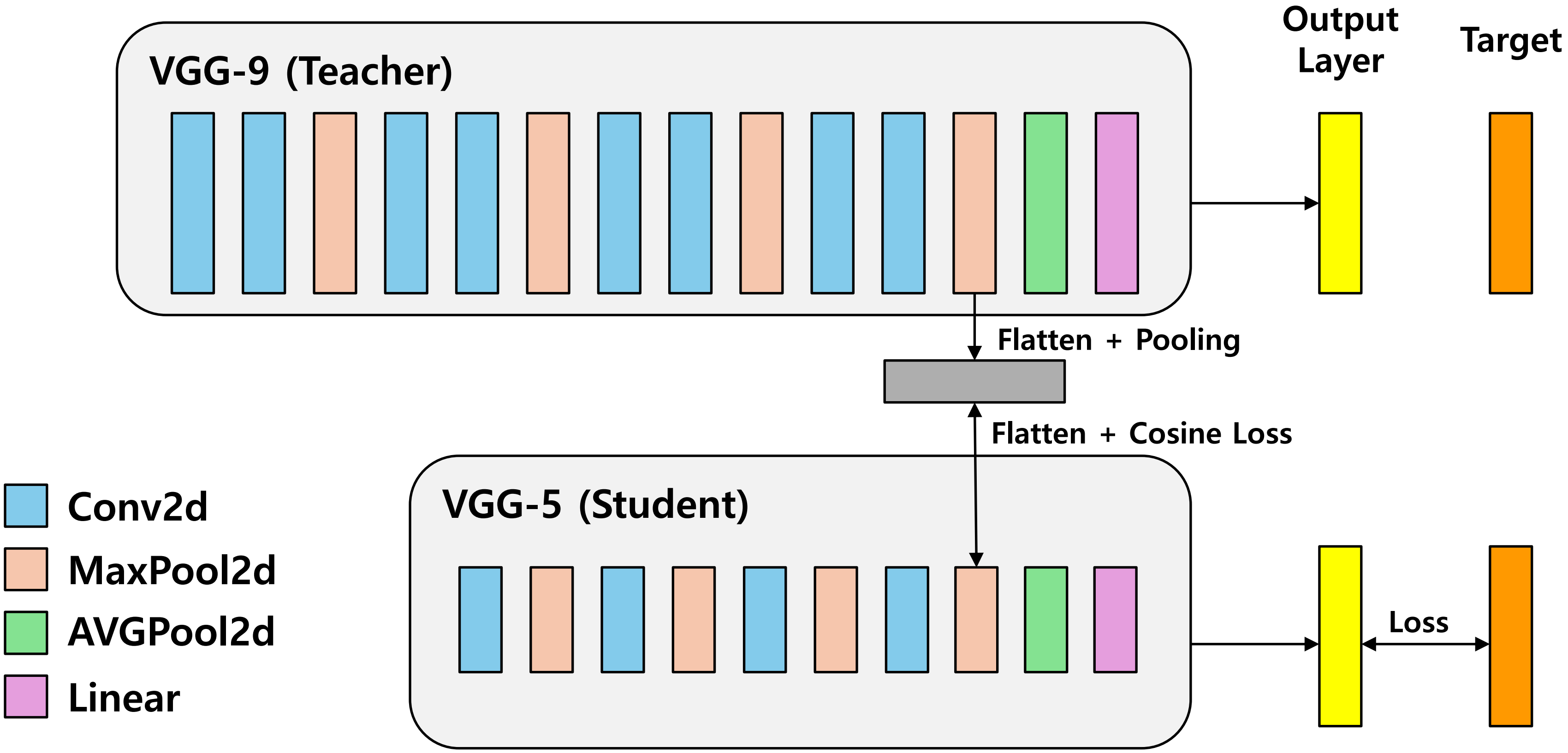
- **Probability calculation:**
  - $P(x,\ T) = softmax\left(\frac{x}{T}\right) \Rightarrow y_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$

- **Distillation Loss:**
  - $\frac{D_{\mathrm{KL}}(P \| Q)}{N} \times T^2$
    - N: batch size
    - T: temperature (loss의 scale이 T^2에 반비례하므로 T^2을 곱해 loss를 보정)

[실습 1] Knowledge Distillation 학습 함수 정의


```
###################### YOUR CODE STARTS HERE ######################
# 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
# Hint: nn.functional.softmax()
soft_targets = nn.functional.softmax(teacher_logits / T, dim=-1)
student_prob = nn.functional.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
soft_targets_loss = torch.sum(soft_targets * (soft_targets.log() - student_prob.log())) / student_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
###################### YOUR CODE ENDS HERE ######################
```

Cosine Loss Minimization (Cosine Loss)



Cosine Embedding Loss

https://docs.pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html

- **Definition:**

$$\text{loss}(x, y) = \begin{cases} 1 - \cos(x_1, x_2), & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2) - \text{margin}), & \text{if } y = -1 \end{cases}$$

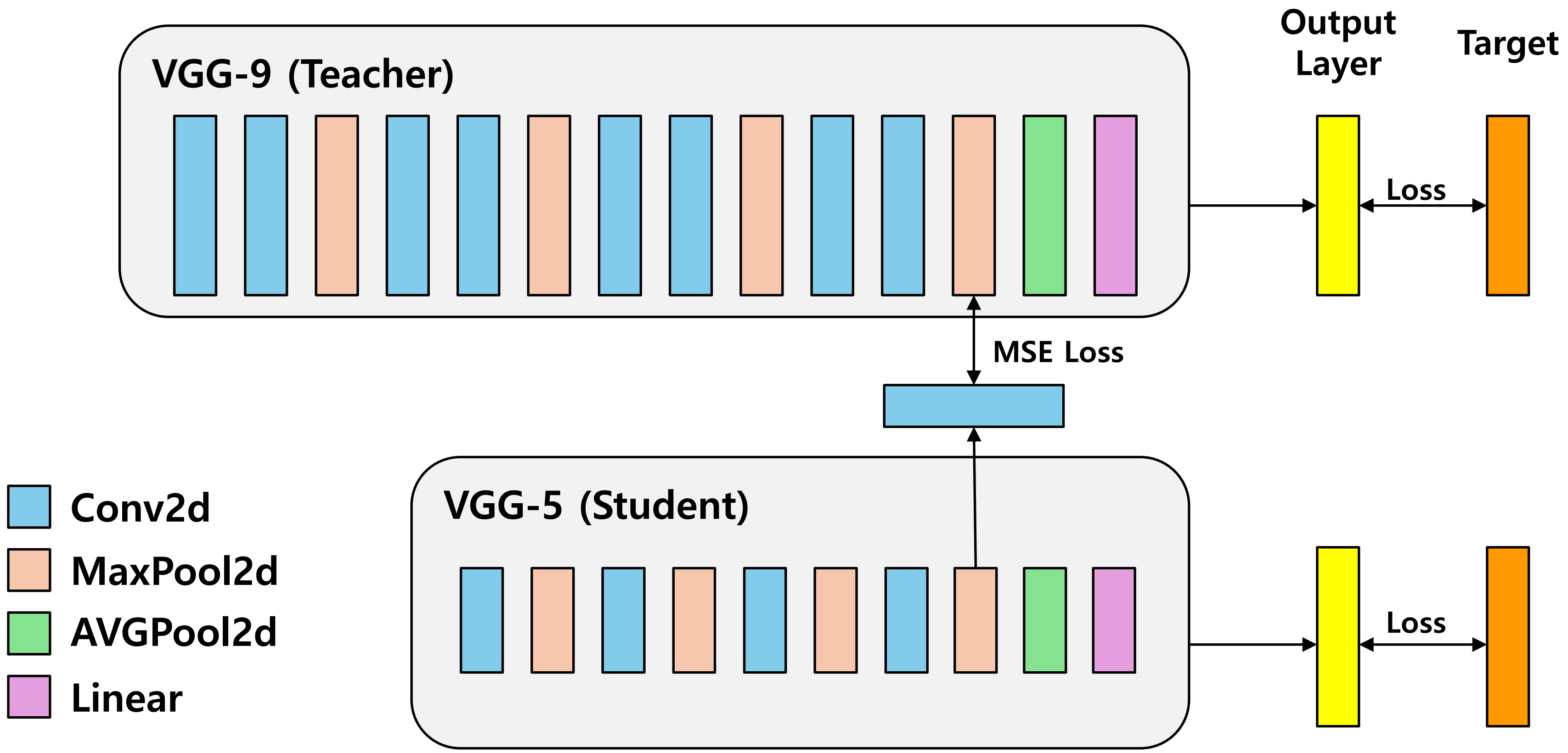
- **Example: nn.CosineEmbeddingLoss**

```
>>> loss = nn.CosineEmbeddingLoss()
>>> input1 = torch.randn(3, 5, requires_grad=True)
>>> input2 = torch.randn(3, 5, requires_grad=True)
>>> target = torch.ones(3)
>>> output = loss(input1, input2, target)
>>> output.backward()
```

[실습 2] Cosine Similarity 기반 KD 학습 함수 정의


```
###################### YOUR CODE STARTS HERE ######################
# Forward pass with the teacher model and keep only the hidden representation
with torch.no_grad():
    _, teacher_hidden_representation = teacher(inputs)

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

# Calculate the cosine loss. Target is a vector of ones. From the loss formula above we can see that is
# the case where loss minimization leads to cosine similarity increase.
# Hint: cosine_loss(x, y, target)에서 target은 1로 이루어진 vector이며, torch.ones(inputs.size(0)).cuda())를 사용
hidden_rep_loss = cosine_loss(student_hidden_representation, teacher_hidden_representation,
                              target=torch.ones(inputs.size(0)).cuda())

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

# Weighted sum of the two losses
loss = hidden_rep_loss_weight * hidden_rep_loss + ce_loss_weight * label_loss
###################### YOUR CODE ENDS HERE ########################
```

Intermediate Regressor (Regressor + MSE)



MSE (Mean Squared Error) Loss

https://docs.pytorch.org/docs/stable/generated/torch.nn.MSELoss.html

- **Definition:**

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = (x_n - y_n)^2$$

- **Example: nn.MSELoss**

```
>>> loss = nn.MSELoss()
>>> input = torch.randn(3, 5, requires_grad=True)
>>> target = torch.randn(3, 5)
>>> output = loss(input, target)
>>> output.backward()
```

[실습 3] Hint-based Knowledge Distillation 학습 함수 정의 (MSE Loss 기반)


```
##################### YOUR CODE STARTS HERE #####################
# Again ignore teacher logits
with torch.no_grad():
    _, teacher_feature_map = teacher(inputs)

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

# Calculate the loss
hidden_rep_loss = mse_loss(regressor_feature_map, teacher_feature_map)

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

# Weighted sum of the two losses
loss = feature_map_weight * hidden_rep_loss + ce_loss_weight * label_loss
##################### YOUR CODE ENDS HERE #######################
```