삼성 DS-KAIST AI Expert 프로그램
Transfer & Multi-task Learning

Hankook Lee
KAIST ALIN Lab.
July 23, 2019

Overview

- **This tutorial consists of two parts:**

  Part 1. Transfer Learning
  - Knowledge Distillation
  - Attention Transfer

  Part 2. Multi-task Learning
  - Shared Architectures

Overview

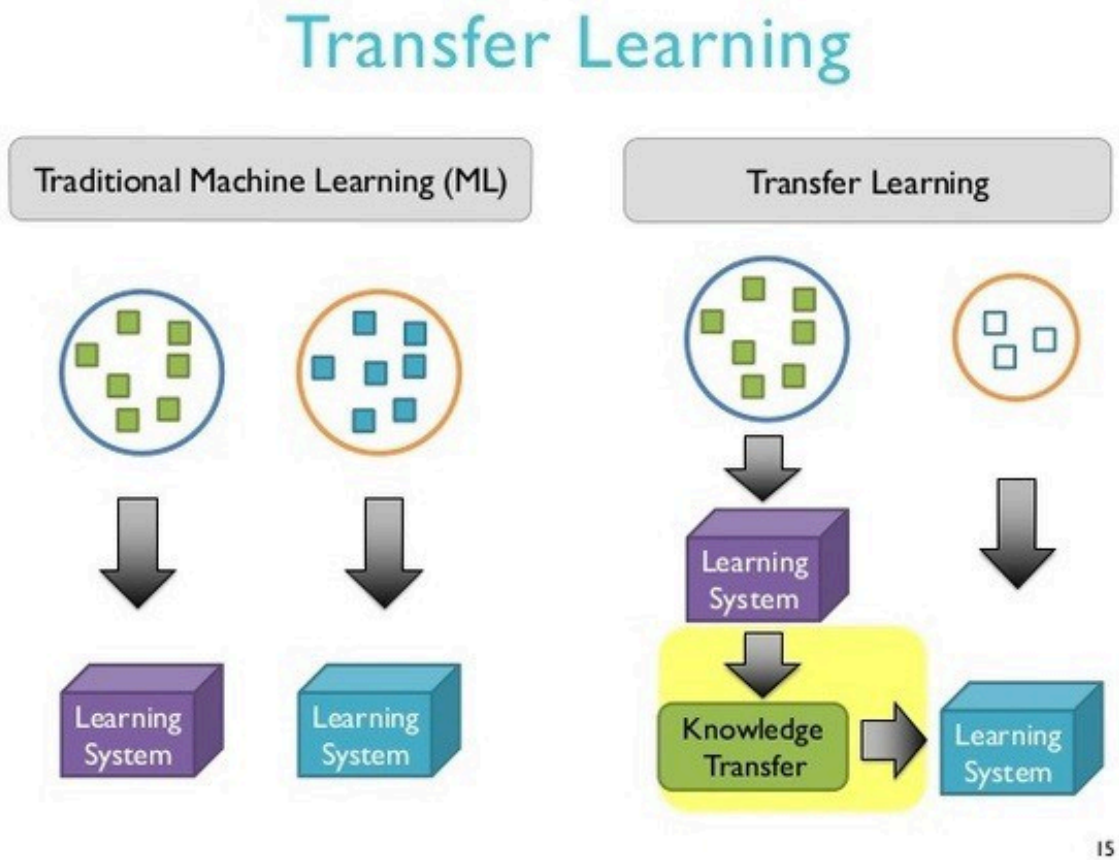
- **Dependencies**
  - python3
  - tensorflow-gpu >= 1.14.0
  - jupyter-notebook
  - matplotlib

- **Please download the material in the github:**
  - https://github.com/bbuing9/Samsung-AI-KAIST

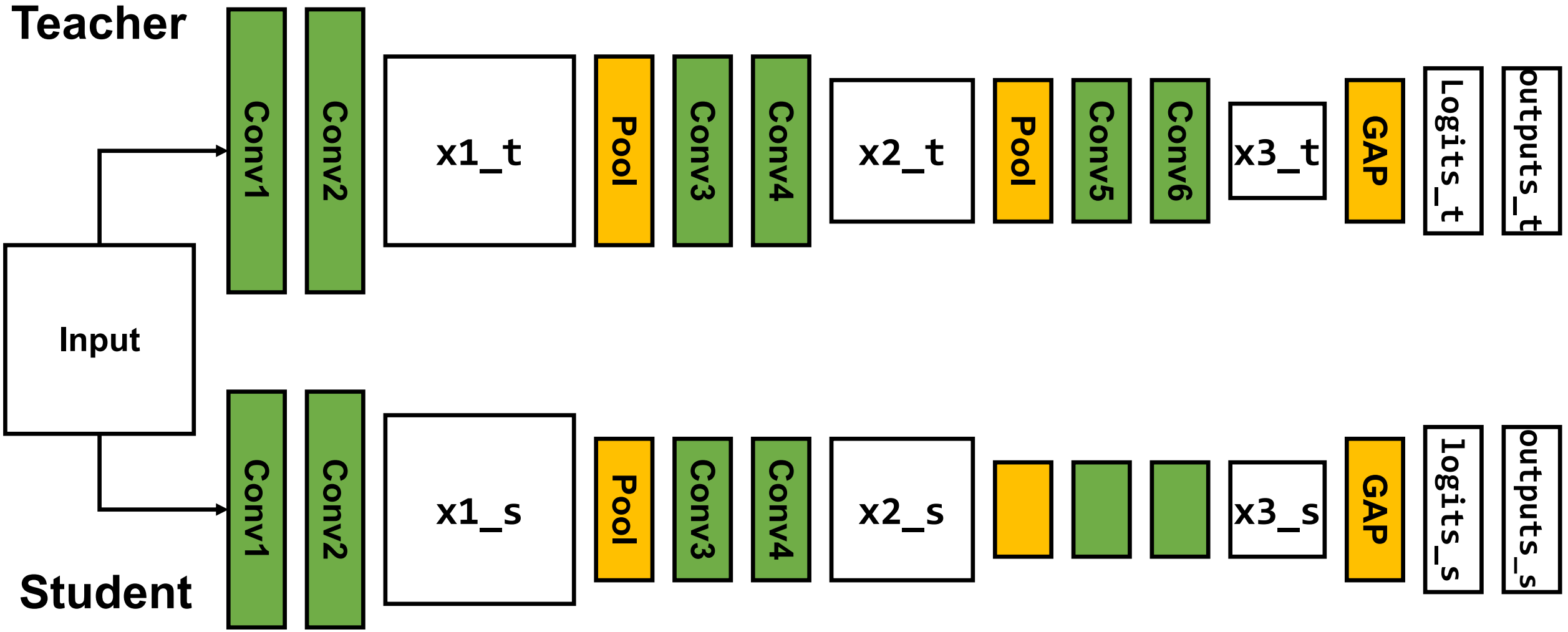
Transfer Learning

What is Transfer Learning?

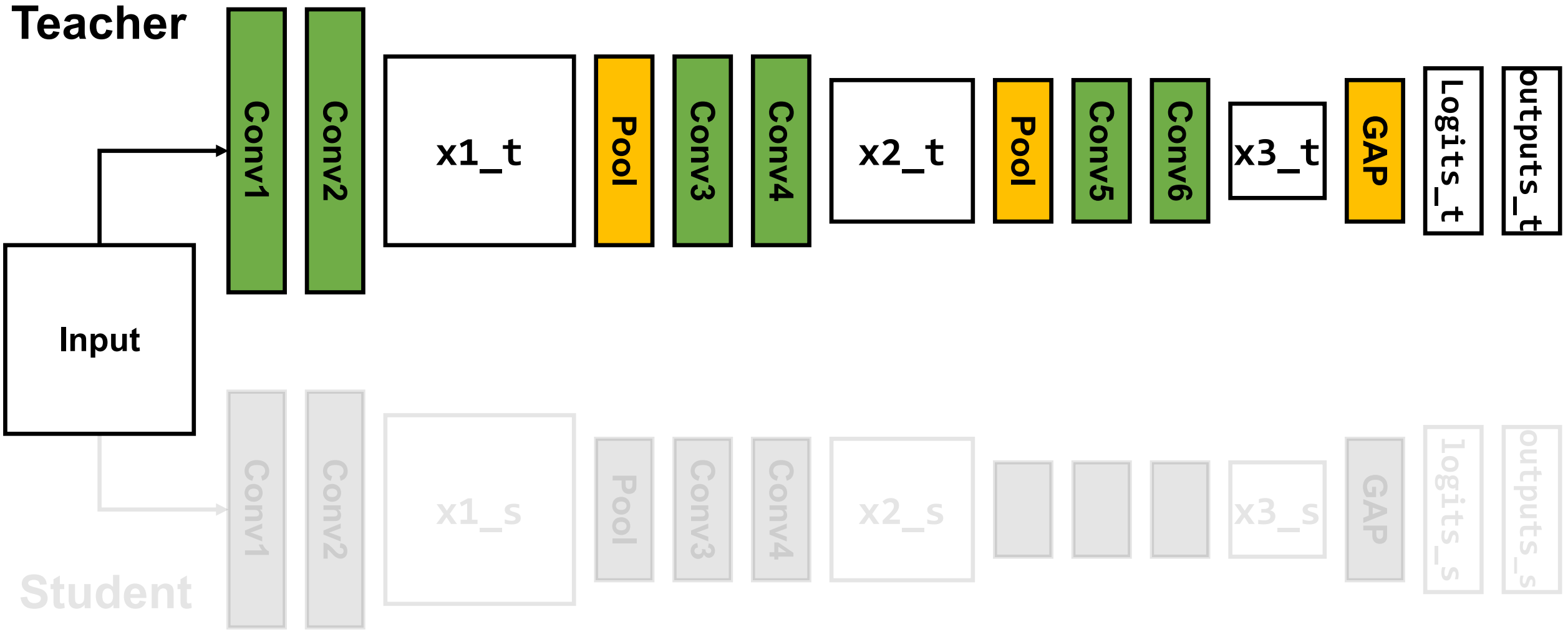
- Transfer ″Knowledge” from pre-trained models to new models



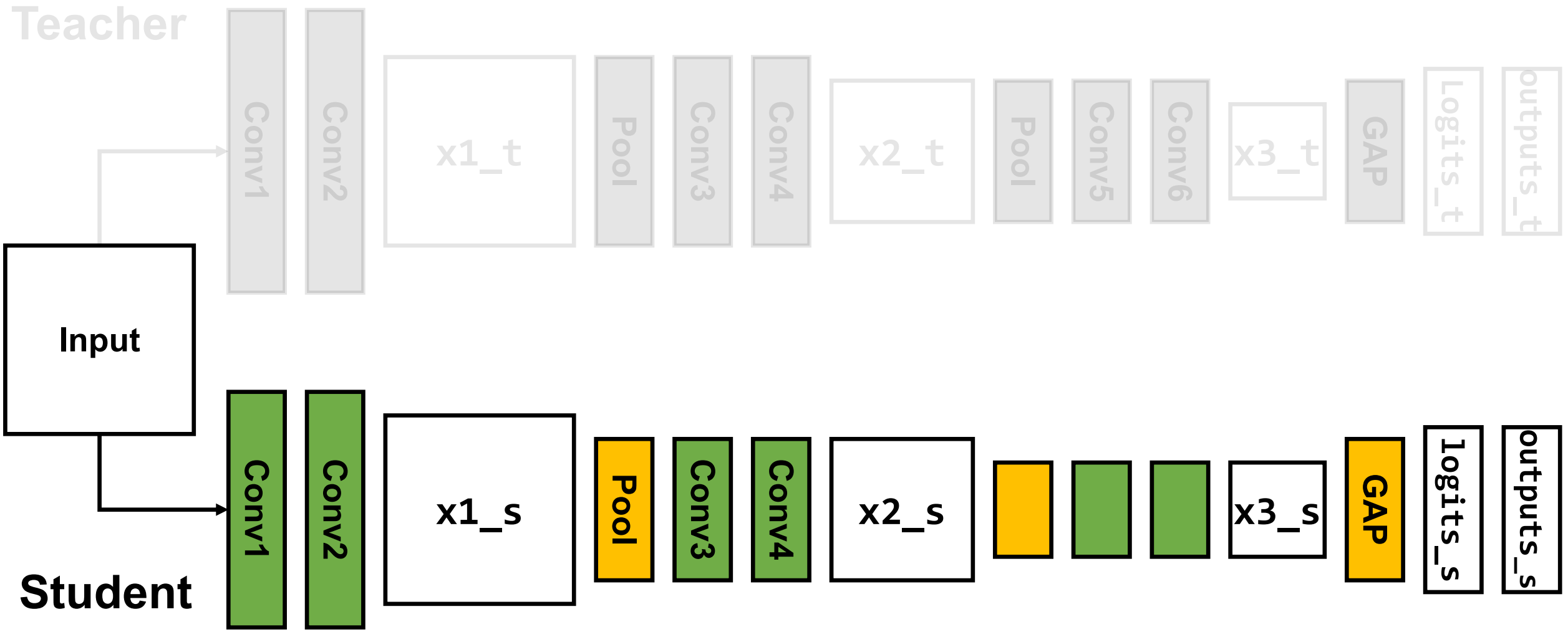
Define Teacher & Student Model



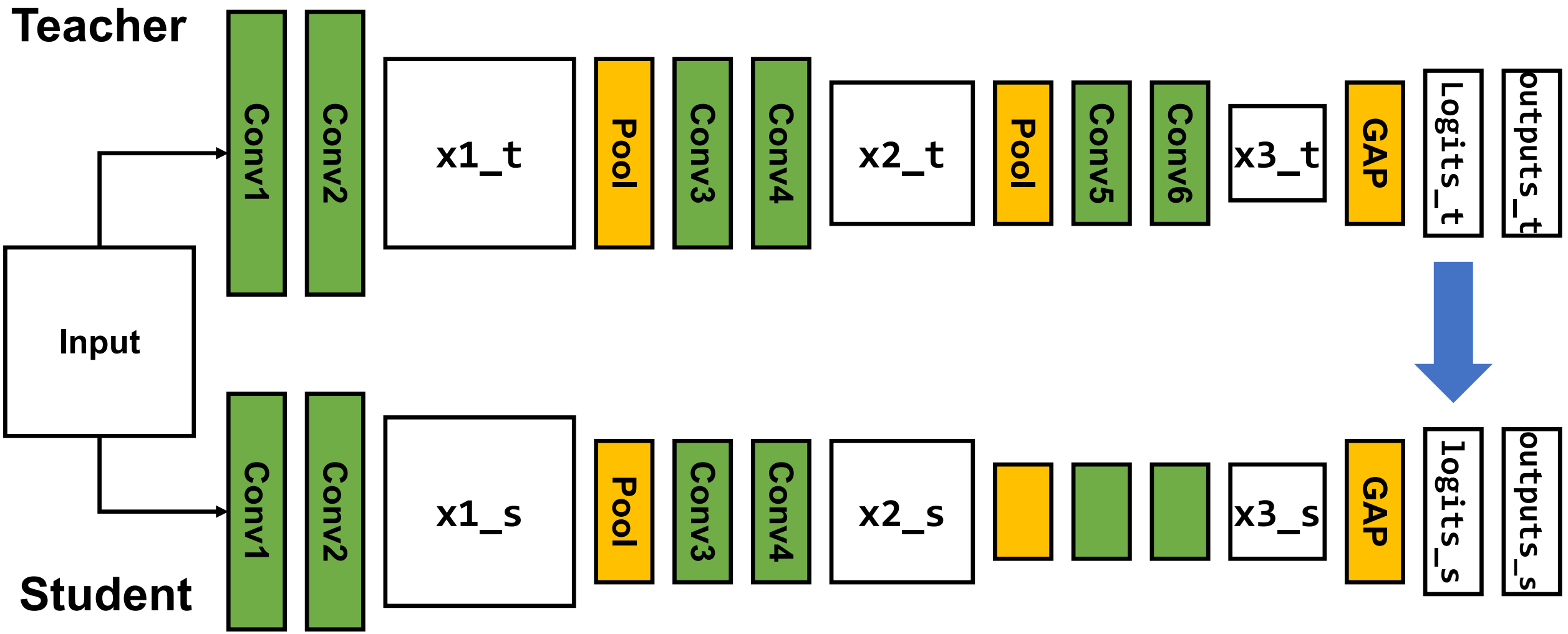
Train Teacher



Q1. Train Student Model without Transfer



Q2. Knowledge Distillation



Q2. Knowledge Distillation

- Convert the below equation to Tensorflow scripts

$$p_t = \mathrm{softmax}(l_t/T)$$

$$p_s = \mathrm{softmax}(l_s/T)$$

$$\mathcal{L}_{\mathrm{KD}} = \mathrm{constant} - \sum_{i=1}^{C} p_t^{(i)} \log p_s^{(i)}$$

Q2. Knowledge Distillation

- Convert the below equation to Tensorflow scripts

```
p_t = tf.nn.softmax(l_t / T)
p_s = tf.nn.log_softmax(l_s / T)
```

$$\mathcal{L}_{\mathrm{KD}} = \mathrm{constant} - \sum_{i=1}^{C} p_t^{(i)} \log p_s^{(i)}$$

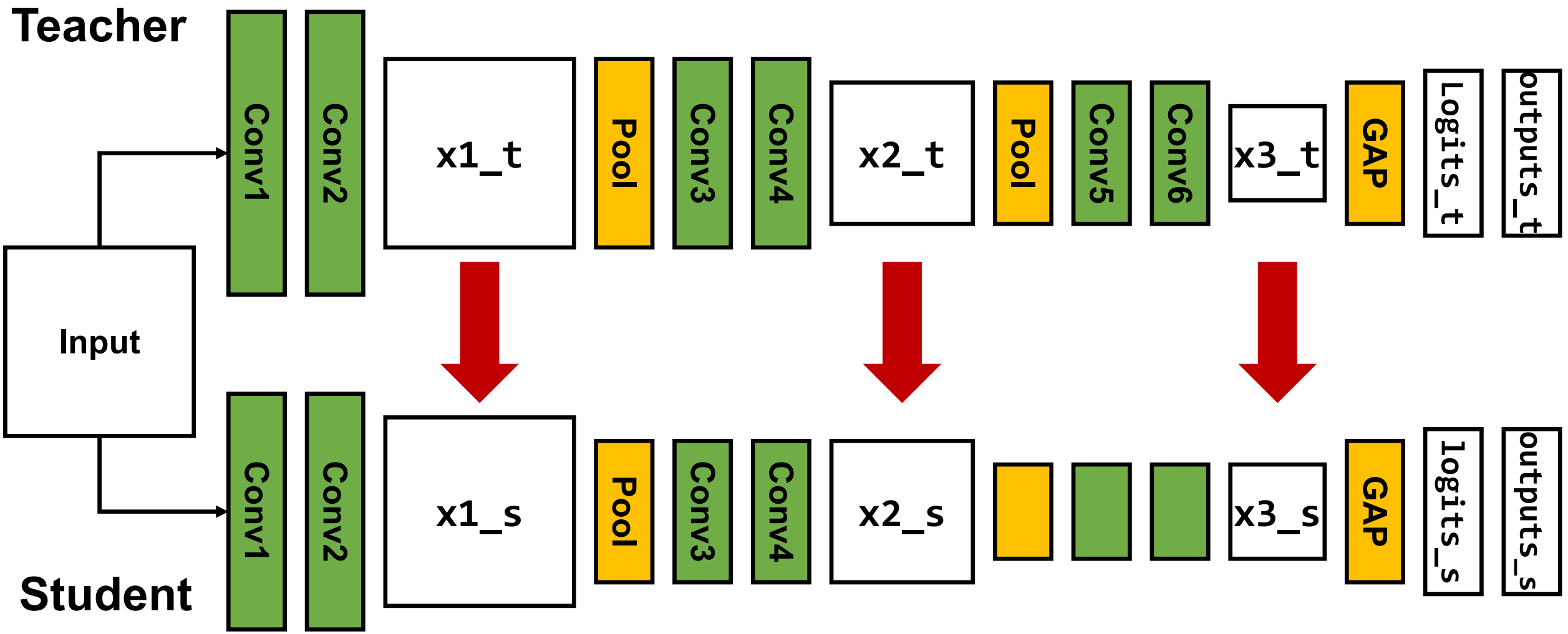
Q2. Knowledge Distillation

- Convert the below equation to Tensorflow scripts

```
p_t = tf.nn.softmax(l_t / T)
p_s = tf.nn.log_softmax(l_s / T)

loss = - tf.math.reduce_sum(p_t * p_s, axis=1)
loss = tf.math.reduce_mean(loss)
```

Q3. Attention Transfer

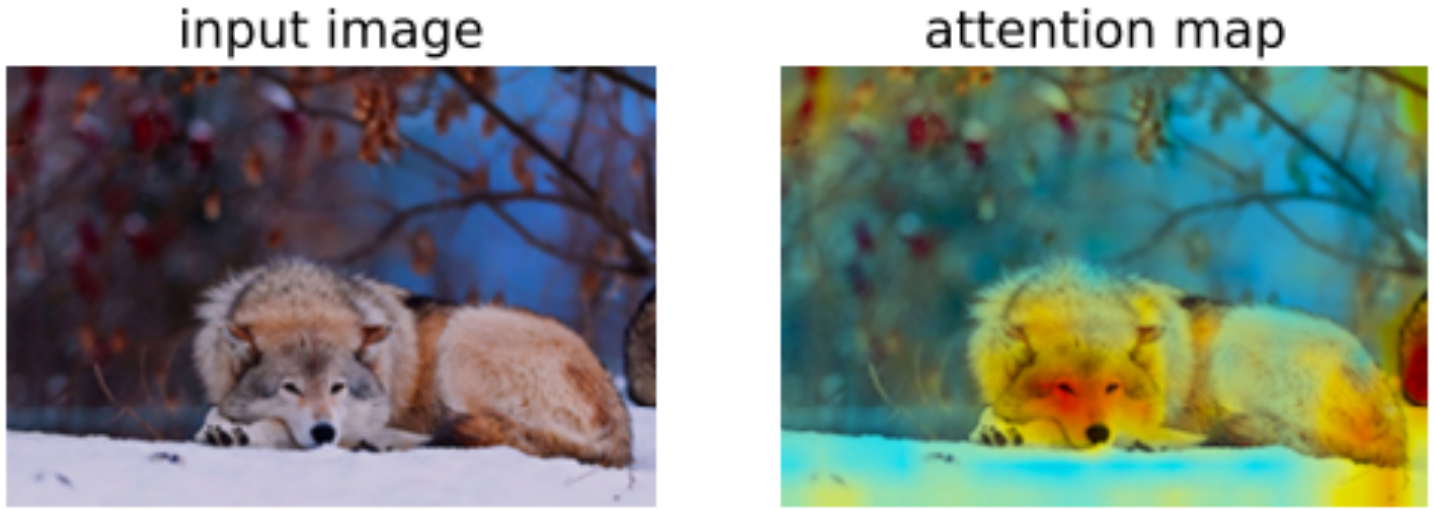


Q3. Attention Transfer

- Convert the below equation to Tensorflow scripts
- B

$$a_t = \sum_{i=1}^{C} |x_t^{(i)}|$$

$$a_s = \sum_{i=1}^{C} |x_s^{(i)}|$$



$$\mathcal{L}_{\mathrm{AT}}(x_t, x_s) = \left\| \frac{\mathrm{vec}(a_t)}{\|\mathrm{vec}(a_t)\|_2} - \frac{\mathrm{vec}(a_s)}{\|\mathrm{vec}(a_s)\|_2} \right\|_2$$

Q3. Attention Transfer

- Convert the below equation to Tensorflow scripts

```
a_t = tf.math.reduce_sum(tf.math.abs(x_t), axis=3)
a_s = tf.math.reduce_sum(tf.math.abs(x_s), axis=3)
```

$$\mathcal{L}_{\mathrm{AT}}(x_t, x_s) = \left\| \frac{\mathrm{vec}(a_t)}{\|\mathrm{vec}(a_t)\|_2} - \frac{\mathrm{vec}(a_s)}{\|\mathrm{vec}(a_s)\|_2} \right\|_2$$

Q3. Attention Transfer

- Convert the below equation to Tensorflow scripts

```
a_t = tf.math.reduce_sum(tf.math.abs(x_t), axis=3)
a_s = tf.math.reduce_sum(tf.math.abs(x_s), axis=3)
vec_t = tf.keras.layers.Flatten()(a_t)
vec_t = tf.math.l2_normalize(vec_t, axis=1)
vec_s = tf.keras.layers.Flatten()(a_s)
vec_s = tf.math.l2_normalize(vec_s, axis=1)
```

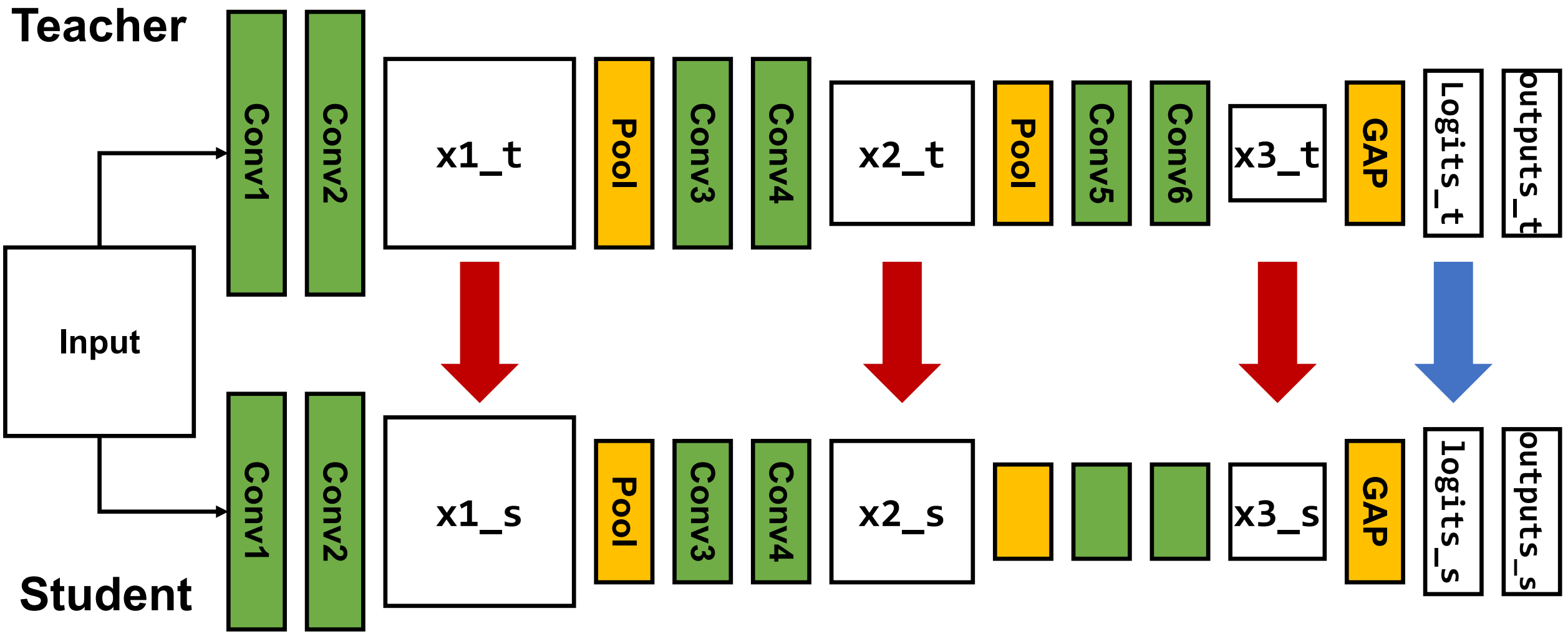
$$\mathcal{L}_{\mathrm{AT}}(x_t, x_s) = \left\| \frac{\mathrm{vec}(a_t)}{\|\mathrm{vec}(a_t)\|_2} - \frac{\mathrm{vec}(a_s)}{\|\mathrm{vec}(a_s)\|_2} \right\|_2$$

Q3. Attention Transfer

- Convert the below equation to Tensorflow scripts

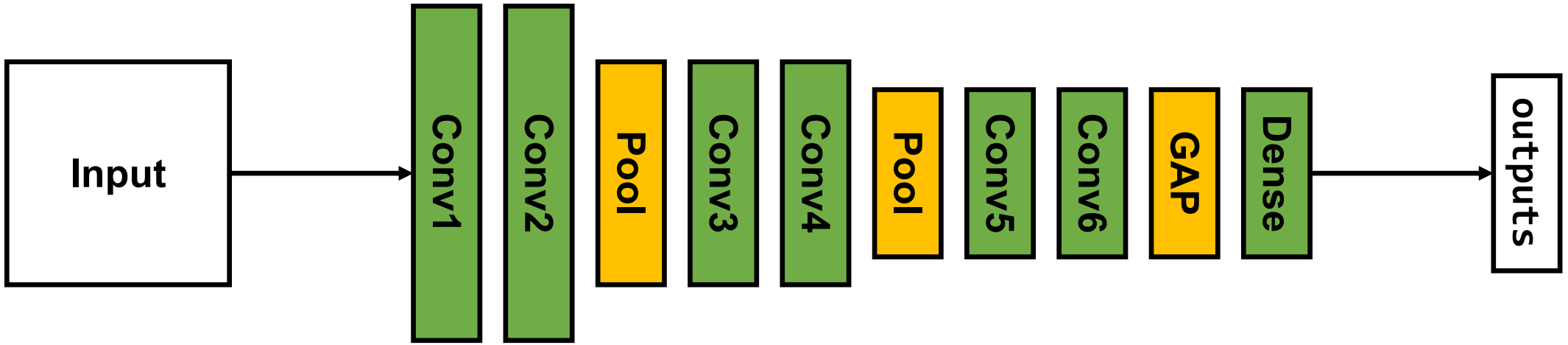
```
a_t = tf.math.reduce_sum(tf.math.abs(x_t), axis=3)
a_s = tf.math.reduce_sum(tf.math.abs(x_s), axis=3)
vec_t = tf.keras.layers.Flatten()(a_t)
vec_t = tf.math.l2_normalize(vec_t, axis=1)
vec_s = tf.keras.layers.Flatten()(a_s)
vec_s = tf.math.l2_normalize(vec_s, axis=1)
loss = tf.math.reduce_mean(tf.norm(a_t-a_s, axis=1))
```

A1. Train Stduent with KD & AT

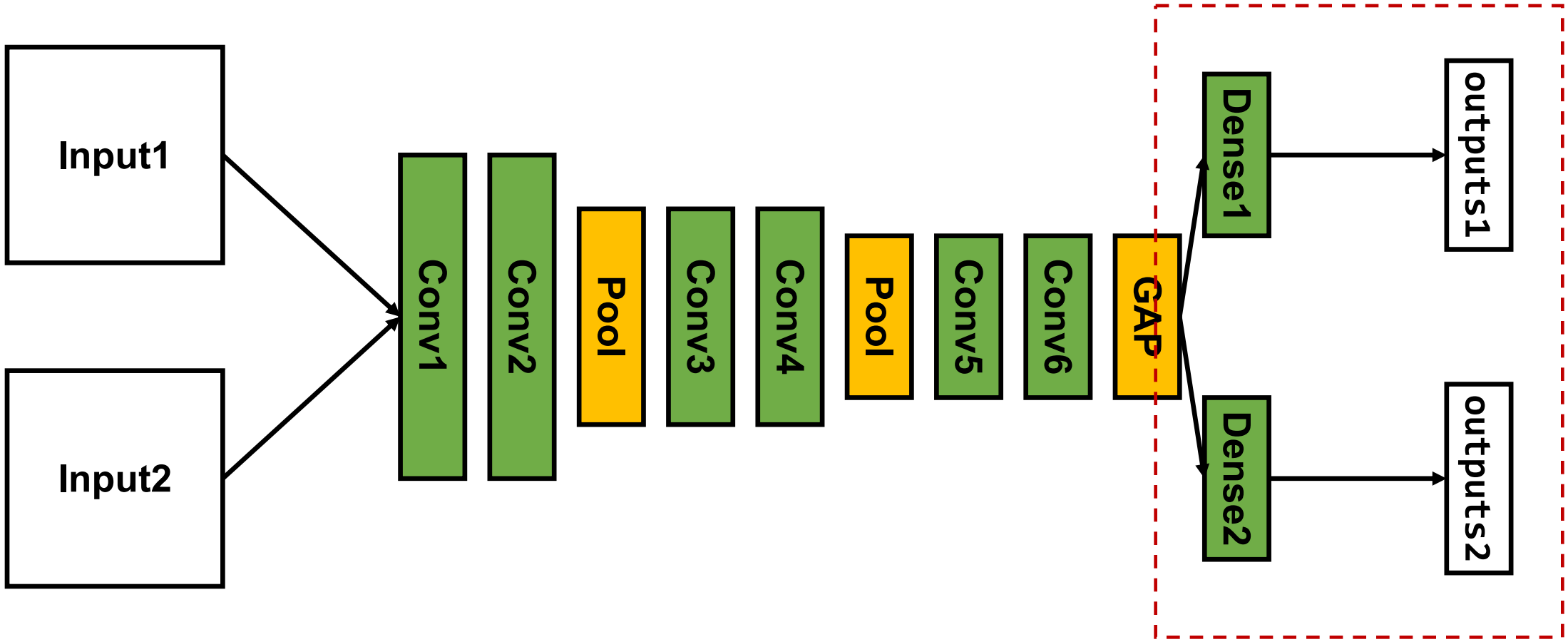


Multi-task Learning

Model for One Task



Q1. Model for Two Tasks

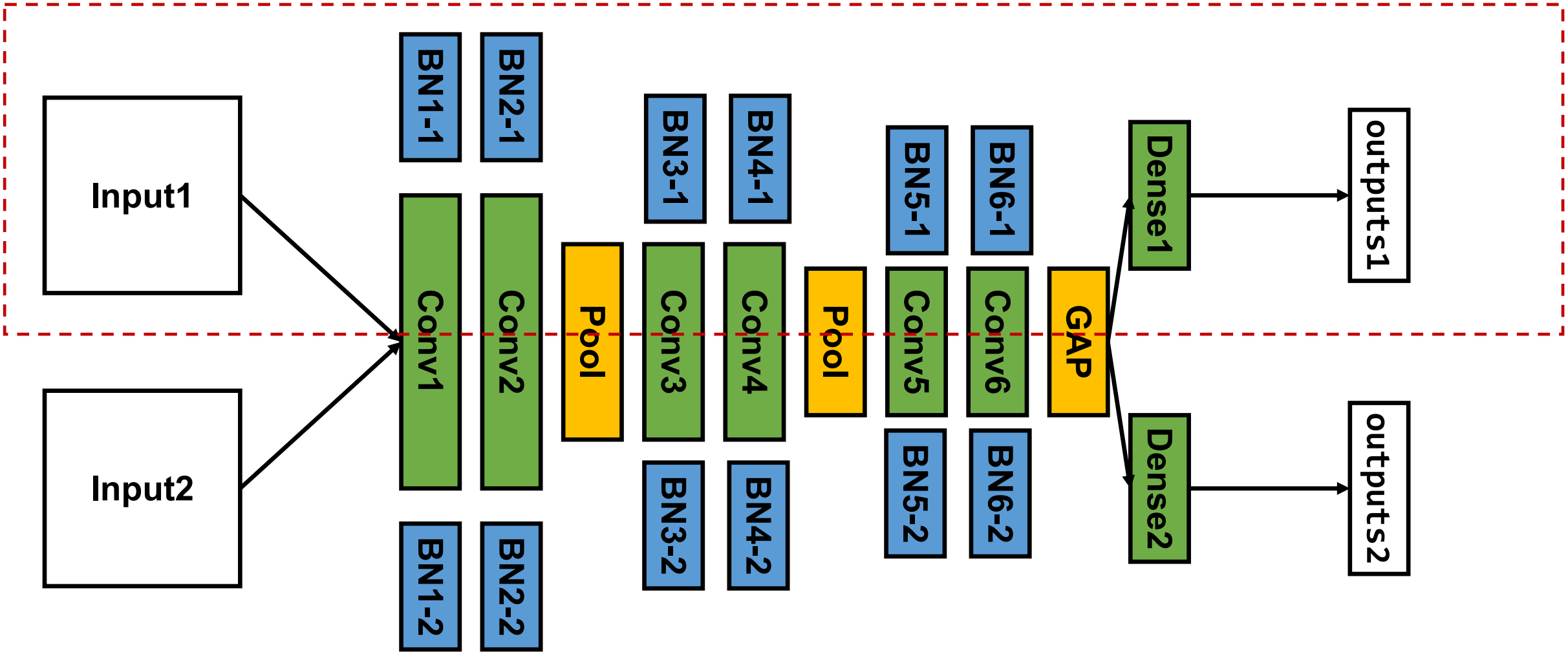


Q1. Model for Two Tasks

```
logits1 = Dense(num_classes1)(x1)
logits2 = Dense(num_classes2)(x2)

outputs1 = Activation('softmax',name='outputs1')(logits1)
outputs2 = Activation('softmax',name='outputs2')(logits2)
```

Q2. Independent BN for each Task



Q2. Independent BN for each Task

```
# define layers
bn1 = BatchNormalization()
bn2 = BatchNormalization()

# compute outputs
outputs1 = relu(bn1(conv(inputs1)))
outputs2 = relu(bn2(conv(inputs2)))
```

A2. Find Best Structure for Multiple Tasks