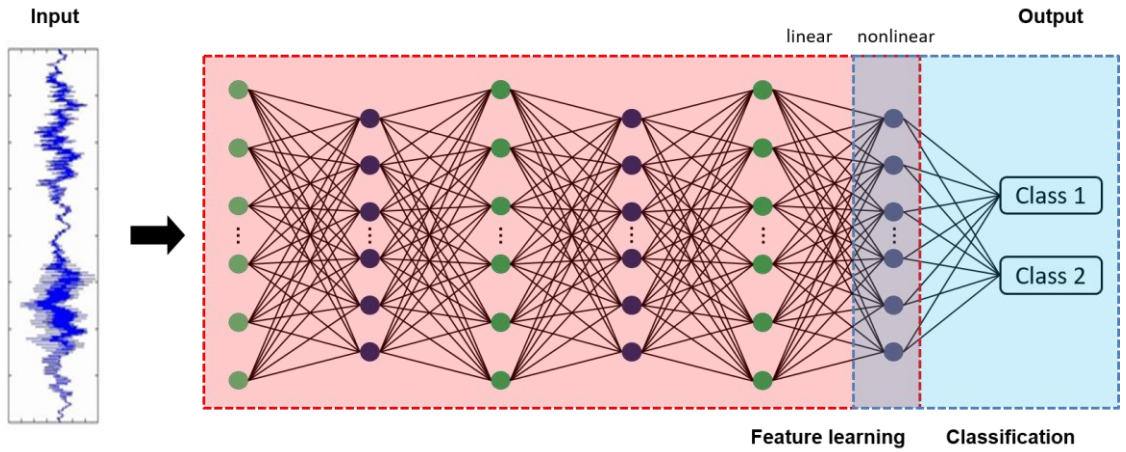
Transfer Learning

Industrial AI Lab.

Prof. Seungchul Lee

Pre-trained Models

- Training a model on ImageNet from scratch takes days or weeks.
- Many models trained on ImageNet and their weights are publicly available!
- Transfer learning
  - Use pre-trained weights, remove last layers to compute representations of images
  - The network is used as a generic feature extractor
  - Train a classification model from these features on a new classification task
  - Pre- trained models can extract more general image features that can help identify edges, textures, shapes, and object composition
  - Better than handcrafted feature extraction on natural images



Transfer Learning

- We assume that these model parameters contain the knowledge learned from the source data set and that this knowledge will be equally applicable to the target data set.
- We will train the output layer from scratch, while the parameters of all remaining layers are fine tuned based on the parameters of the source model.
- Or initialize all weights from pre-trained model, then train them with target data

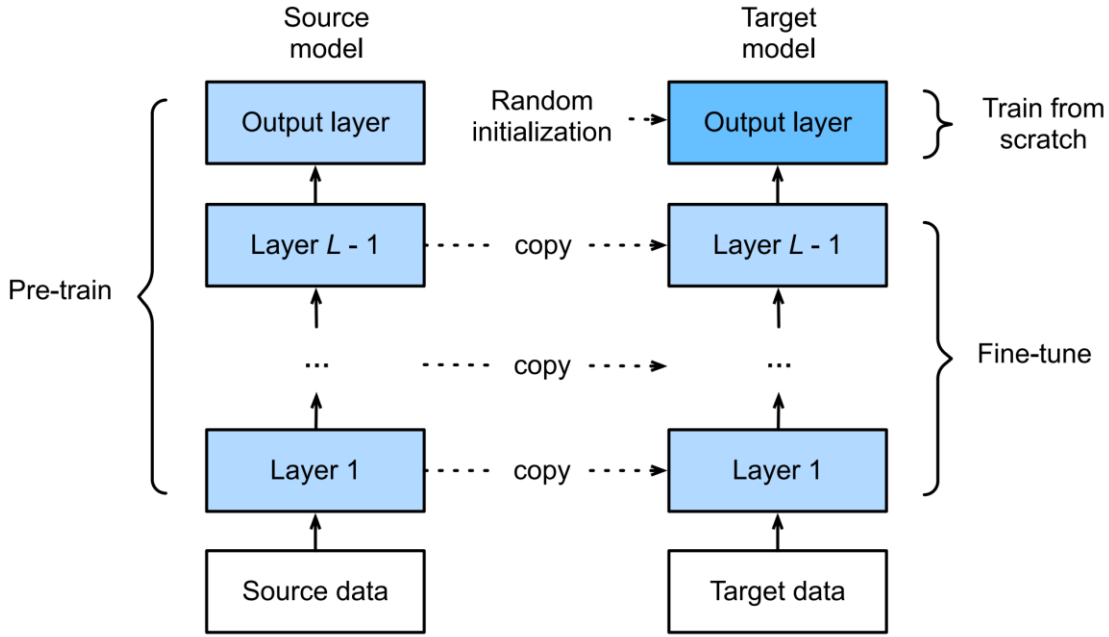


Image Classification with VGG16

- Target data
  - 5 classes

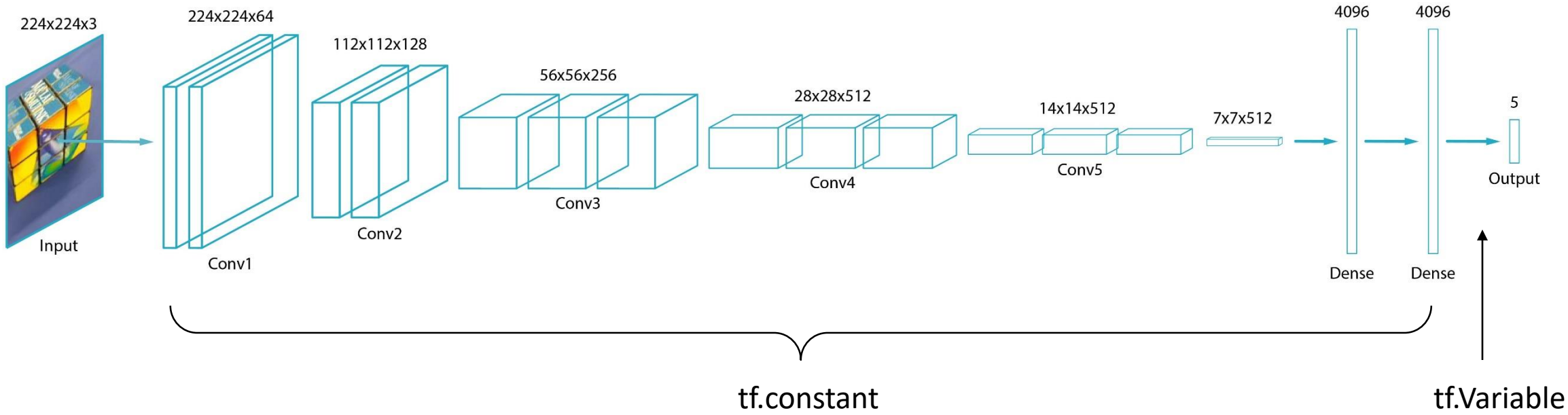
```
Dict = ['Hat','Cube','Card','Torch','screw']
```

- Target data to VGG16
  - Poor performance

1. mosquito_net: 4.66%
2. toilet_tissue: 4.00%
3. envelope: 2.29%
4. carton: 2.20%
5. photocopier: 1.86%

Label : Cube

Transfer Learning Structure



Transfer Learning Implementation

```
vgg16_weights = model.get_weights()

weights = {
    'conv1_1' : tf.constant(vgg16_weights[0]),
    'conv1_2' : tf.constant(vgg16_weights[2]),

    'conv2_1' : tf.constant(vgg16_weights[4]),
    'conv2_2' : tf.constant(vgg16_weights[6]),

    'conv3_1' : tf.constant(vgg16_weights[8]),
    'conv3_2' : tf.constant(vgg16_weights[10]),
    'conv3_3' : tf.constant(vgg16_weights[12]),

    'conv4_1' : tf.constant(vgg16_weights[14]),
    'conv4_2' : tf.constant(vgg16_weights[16]),
    'conv4_3' : tf.constant(vgg16_weights[18]),

    'conv5_1' : tf.constant(vgg16_weights[20]),
    'conv5_2' : tf.constant(vgg16_weights[22]),
    'conv5_3' : tf.constant(vgg16_weights[24]),

    'fc1' : tf.constant(vgg16_weights[26]),
    'fc2' : tf.constant(vgg16_weights[28]),
    # train from scratch
    'out' : tf.Variable(tf.random_normal([4096, 5], stddev = 0.1))
}
```

```
biases = {
    'conv1_1' : tf.constant(vgg16_weights[1]),
    'conv1_2' : tf.constant(vgg16_weights[3]),

    'conv2_1' : tf.constant(vgg16_weights[5]),
    'conv2_2' : tf.constant(vgg16_weights[7]),

    'conv3_1' : tf.constant(vgg16_weights[9]),
    'conv3_2' : tf.constant(vgg16_weights[11]),
    'conv3_3' : tf.constant(vgg16_weights[13]),

    'conv4_1' : tf.constant(vgg16_weights[15]),
    'conv4_2' : tf.constant(vgg16_weights[17]),
    'conv4_3' : tf.constant(vgg16_weights[19]),

    'conv5_1' : tf.constant(vgg16_weights[21]),
    'conv5_2' : tf.constant(vgg16_weights[23]),
    'conv5_3' : tf.constant(vgg16_weights[25]),

    'fc1' : tf.constant(vgg16_weights[27]),
    'fc2' : tf.constant(vgg16_weights[29]),
    # train from scratch
    'out' : tf.Variable(tf.random_normal([5], stddev = 0.1))
}
```

Testing

```
Dict = ['Hat','Cube','Card','Torch','screw']
```

```
Prediction : Cube
Probability : [0.   0.99 0.01 0.   0.  ]
```