March 26th, 2021

# AI+BD ML Lab. Day 5

## Transfer Learning

YoungIn Kim

<youngkim21@postech.edu>

MoNet Laboratory

# Goals

1. Understanding **Transfer Learning**

2. Understanding more about CNN Deep model

3. Make Code!

# Backgrounds

For your information & reminding

# Transfer Learning

More Data,
High Intelligence

# Transfer Learning

Collecting data is
the most difficult thing

# Transfer Learning

Collecting data is
the most difficult thing

How to solve?

# Transfer Learning

Collecting data is
the most difficult thing

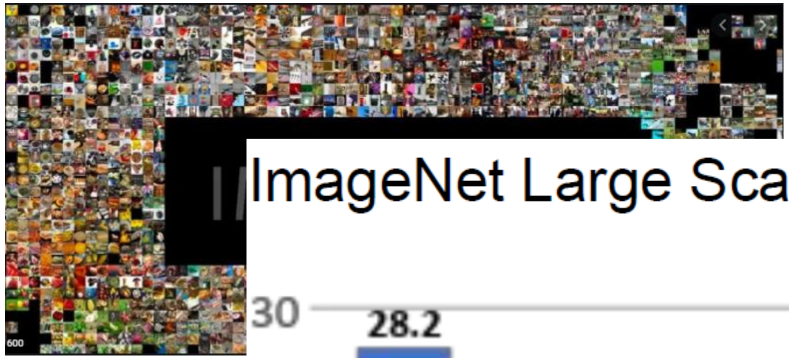
How to solve?

- Transfer Learning
- Self-supervised Learning
- Semi-supervised Learning
- …

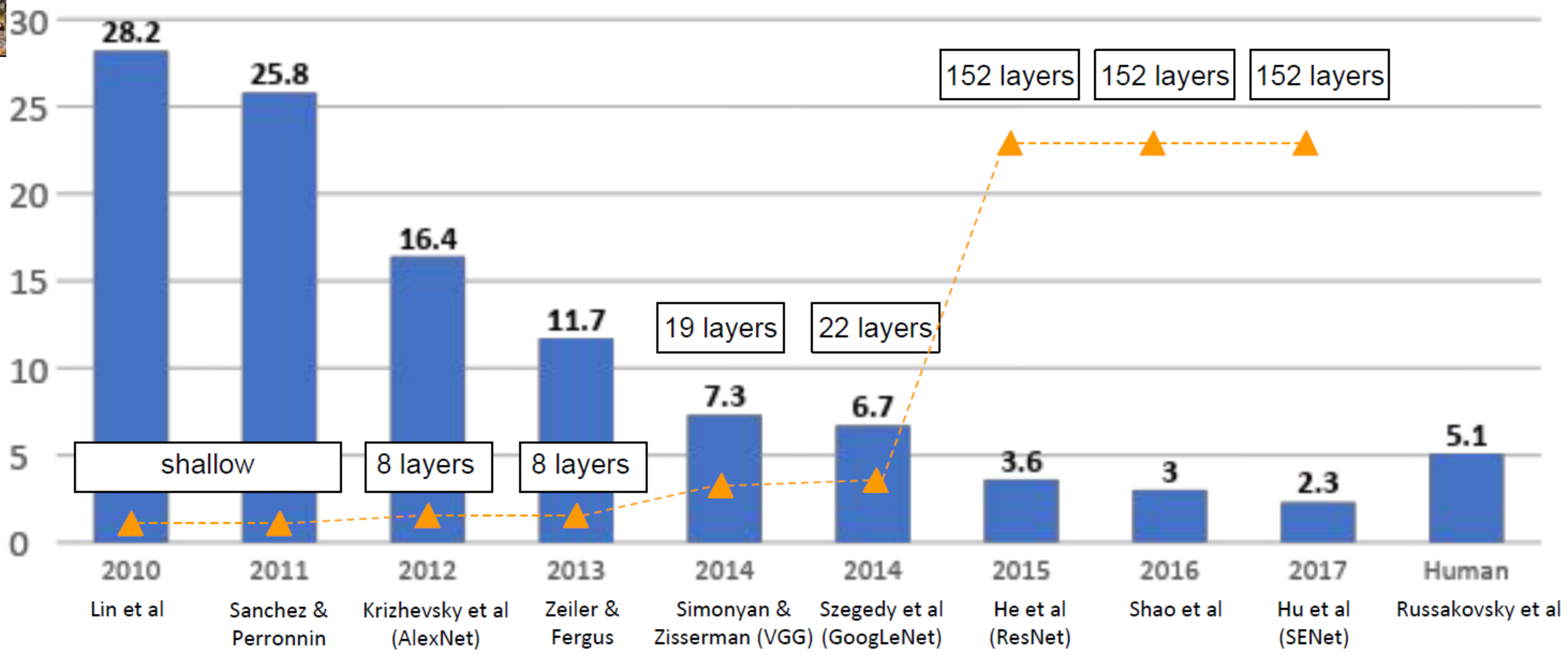
# Transfer Learning

1000 Class, 14 million Data

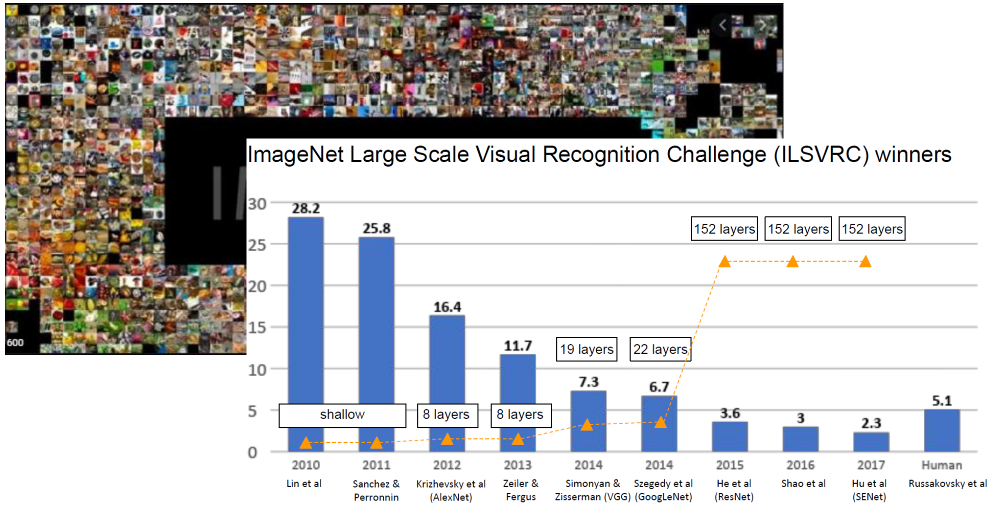
# Transfer Learning



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



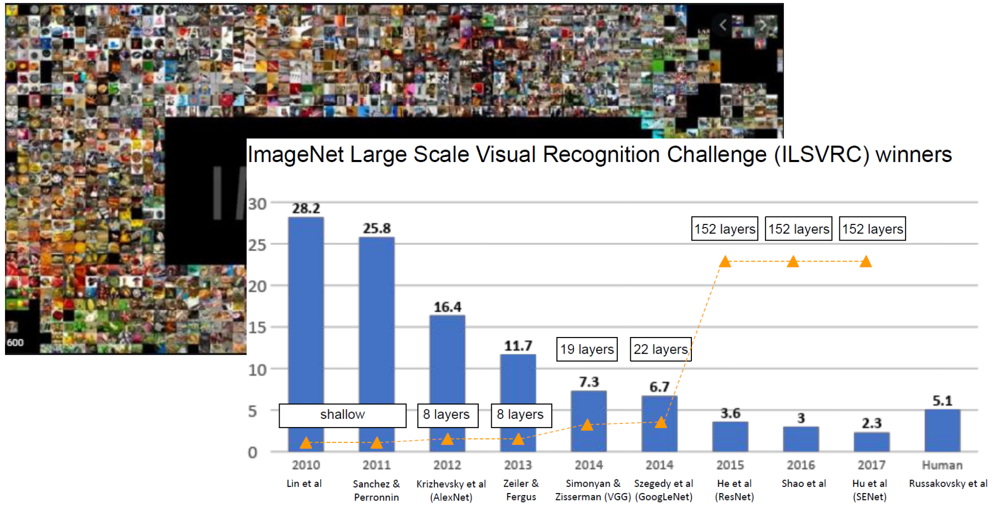
# Transfer Learning



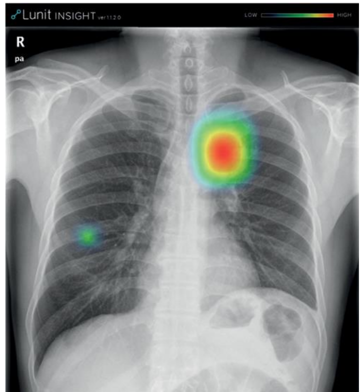
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

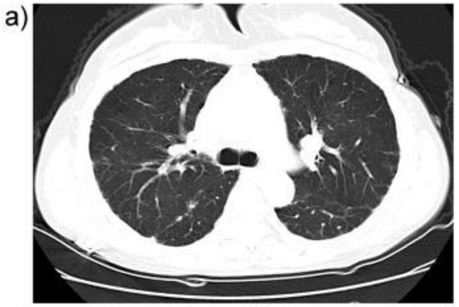
Why don't we REUSE this model?

# Transfer Learning

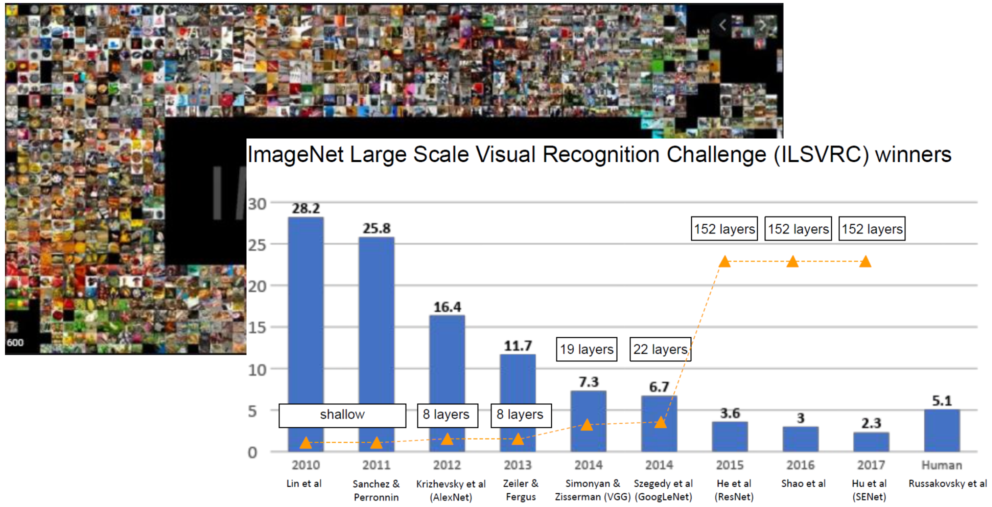


Why don’t we <u>REUSE</u> this model?





# Transfer Learning

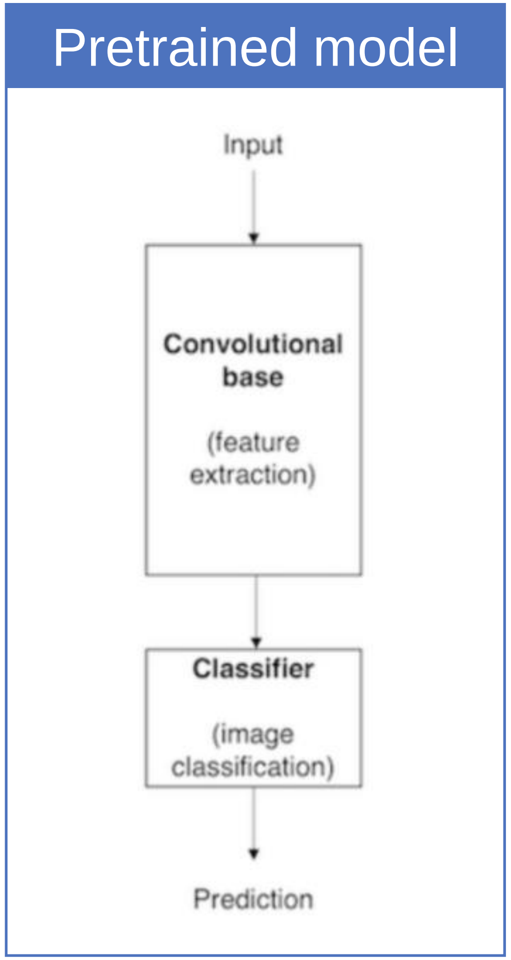


ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Why don’t we REUSE this model?

Fine Tuning!

# Transfer Learning



Pretrained model

Input

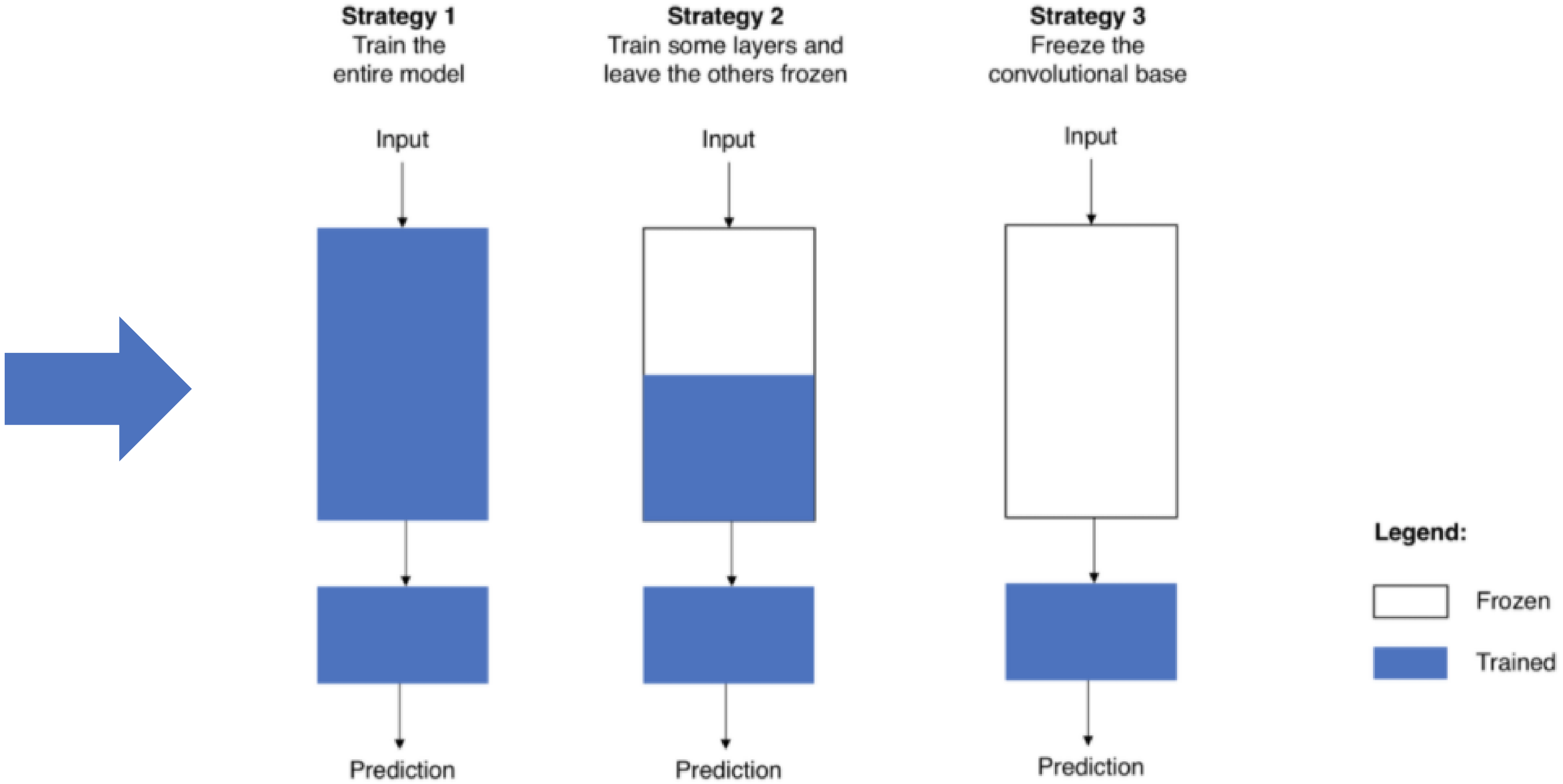
**Convolutional base**

(feature extraction)

**Classifier**

(image classification)

Prediction



**Strategy 1**
Train the entire model

**Strategy 2**
Train some layers and leave the others frozen

**Strategy 3**
Freeze the convolutional base

Input

Prediction

**Legend:**

□ Frozen

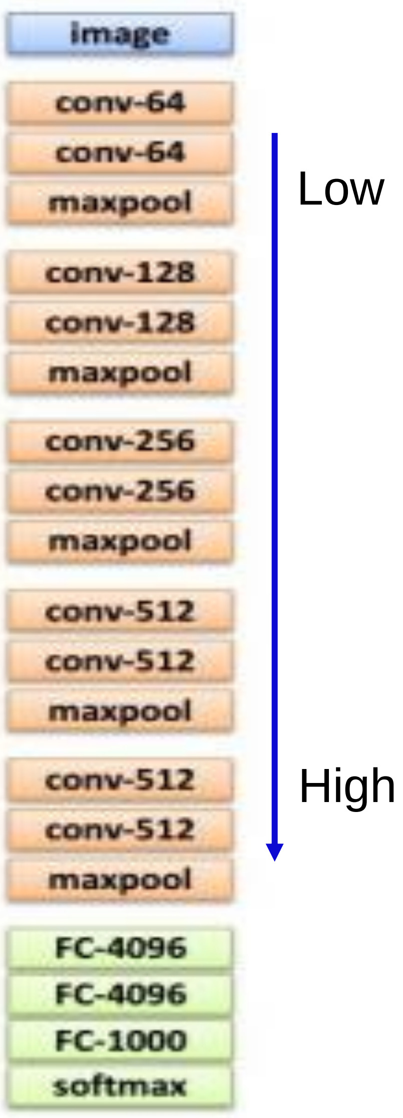
■ Trained

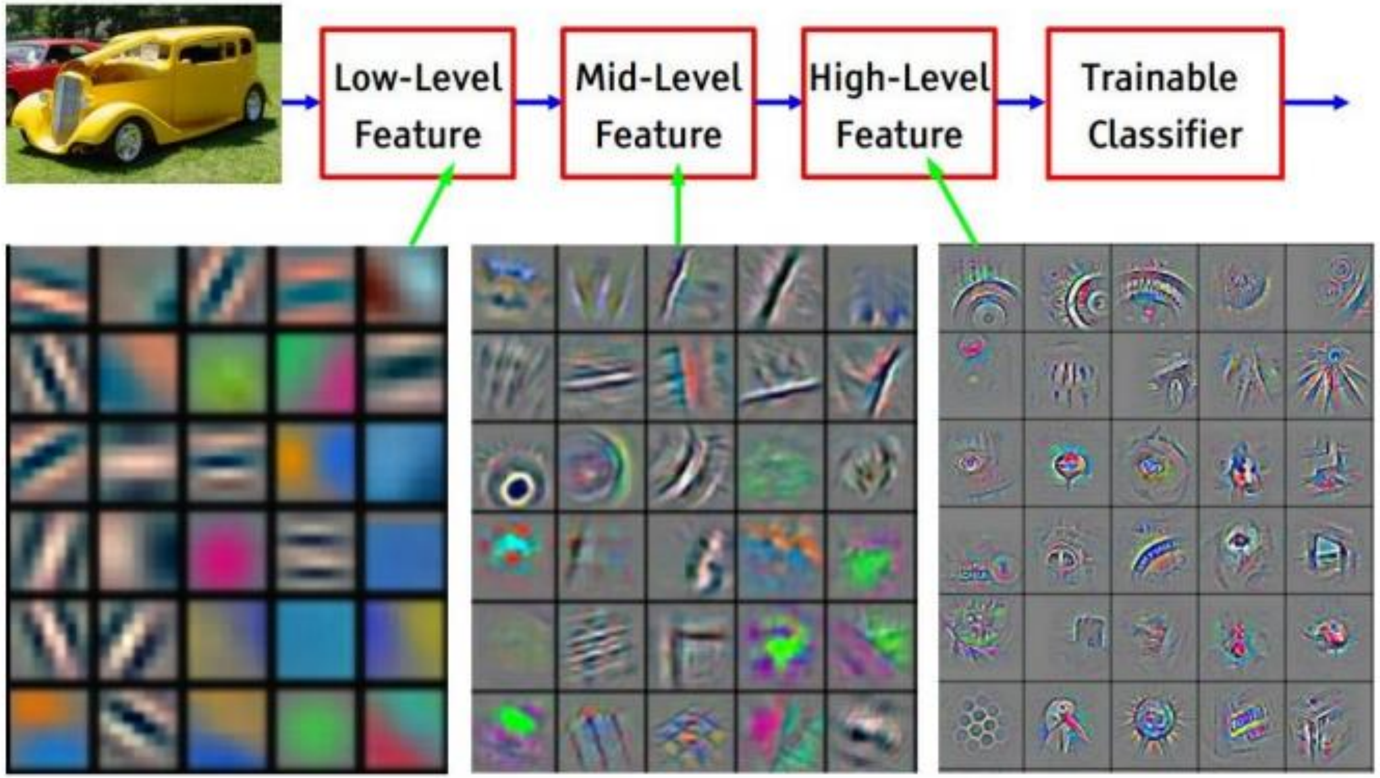
Fine Tuning

https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

# Transfer Learning

CNN Architecture

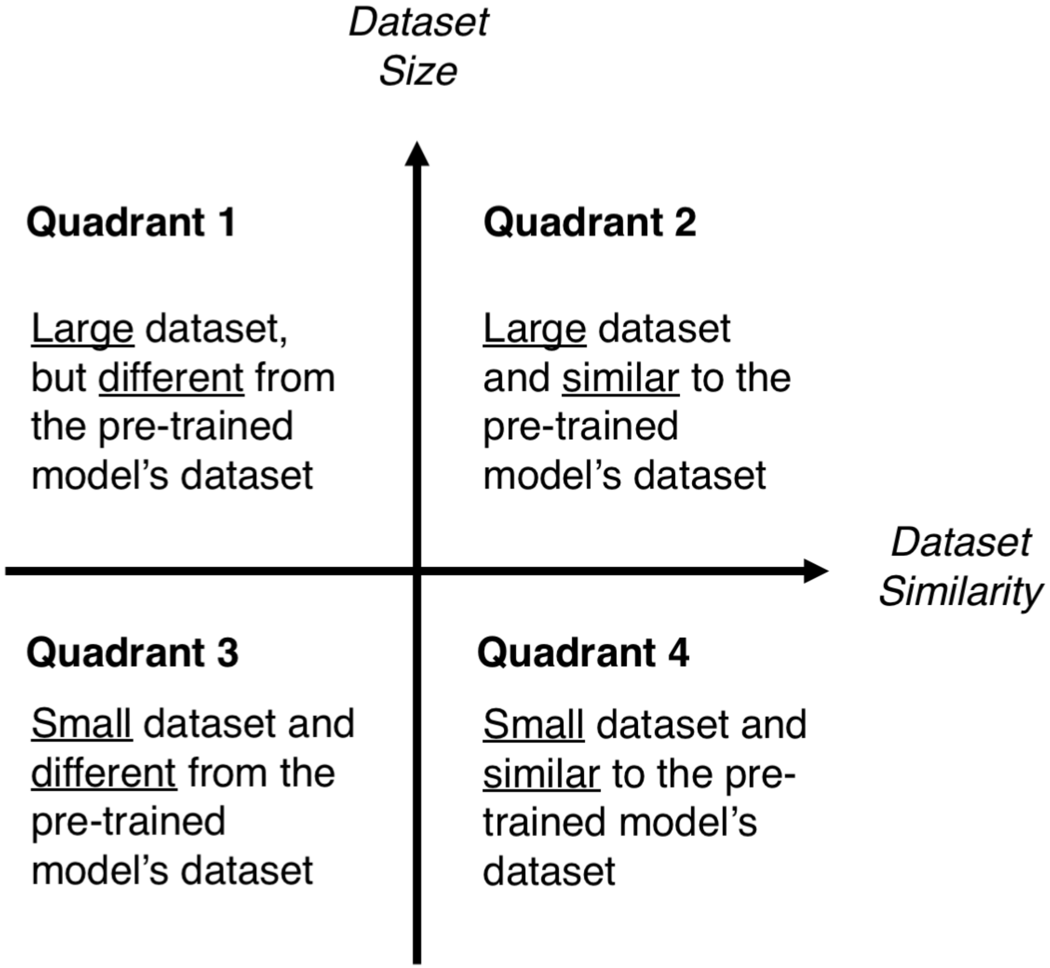




Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

https://yosinski.com/deepvis

# Transfer Learning



**Quadrant 1**

Large dataset, but different from the pre-trained model's dataset

**Quadrant 2**

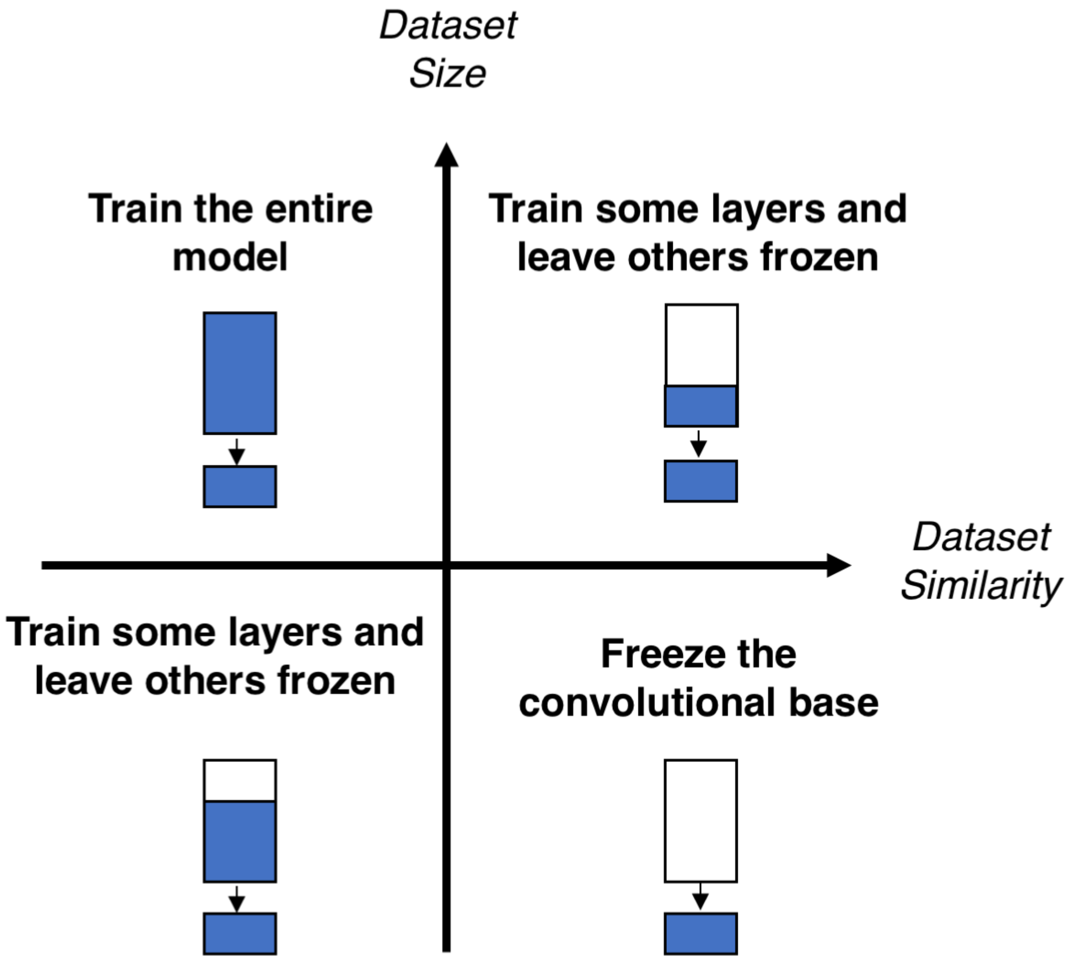
Large dataset and similar to the pre-trained model's dataset

**Quadrant 3**

Small dataset and different from the pre-trained model's dataset

**Quadrant 4**

Small dataset and similar to the pre-trained model's dataset



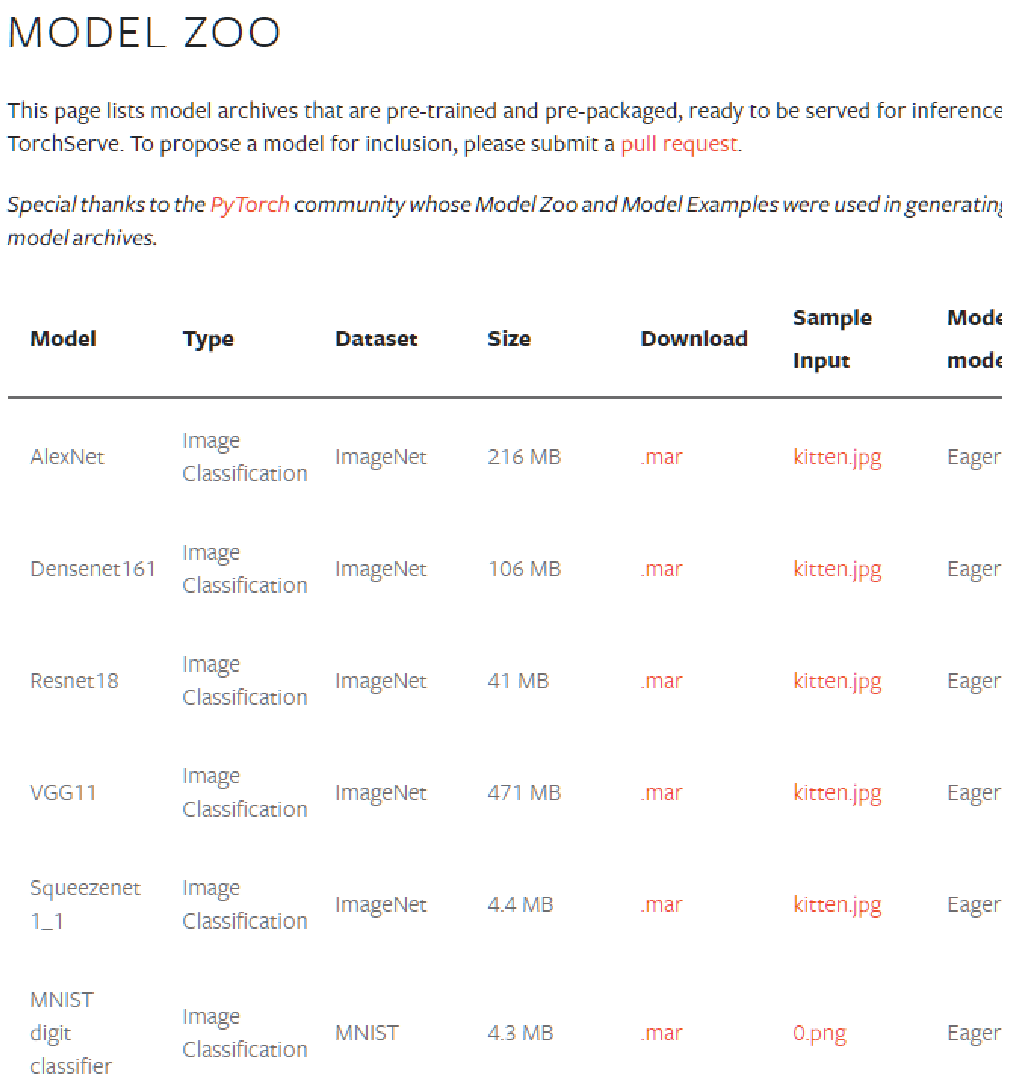
**Train the entire model**

**Train some layers and leave others frozen**

**Train some layers and leave others frozen**

**Freeze the convolutional base**

# Transfer Learning



## MODEL ZOO

This page lists model archives that are pre-trained and pre-packaged, ready to be served for inference TorchServe. To propose a model for inclusion, please submit a pull request.

*Special thanks to the PyTorch community whose Model Zoo and Model Examples were used in generatin model archives.*

| Model | Type | Dataset | Size | Download | Sample Input | Mode mode |
|---|---|---|---|---|---|---|
| AlexNet | Image Classification | ImageNet | 216 MB | .mar | kitten.jpg | Eager |
| Densenet161 | Image Classification | ImageNet | 106 MB | .mar | kitten.jpg | Eager |
| Resnet18 | Image Classification | ImageNet | 41 MB | .mar | kitten.jpg | Eager |
| VGG11 | Image Classification | ImageNet | 471 MB | .mar | kitten.jpg | Eager |
| Squeezenet 1_1 | Image Classification | ImageNet | 4.4 MB | .mar | kitten.jpg | Eager |
| MNIST digit classifier | Image Classification | MNIST | 4.3 MB | .mar | 0.png | Eager |

https://pytorch.org/serve/model_zoo.html

# It’s coding time

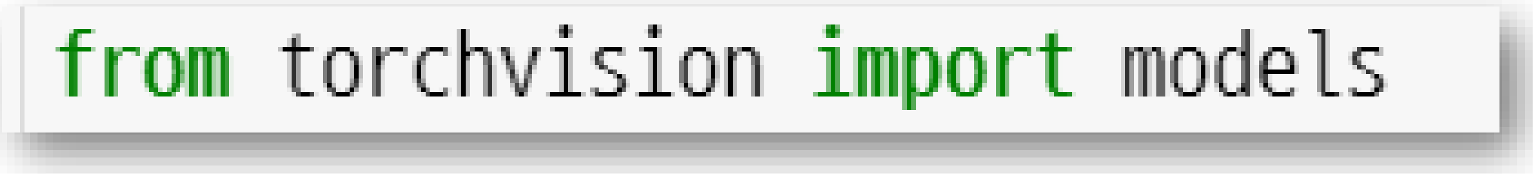
Let’s fill the I.P.Y.N.B

# Full code

**Full code :**

https://git.io/aibd-tl-5-full

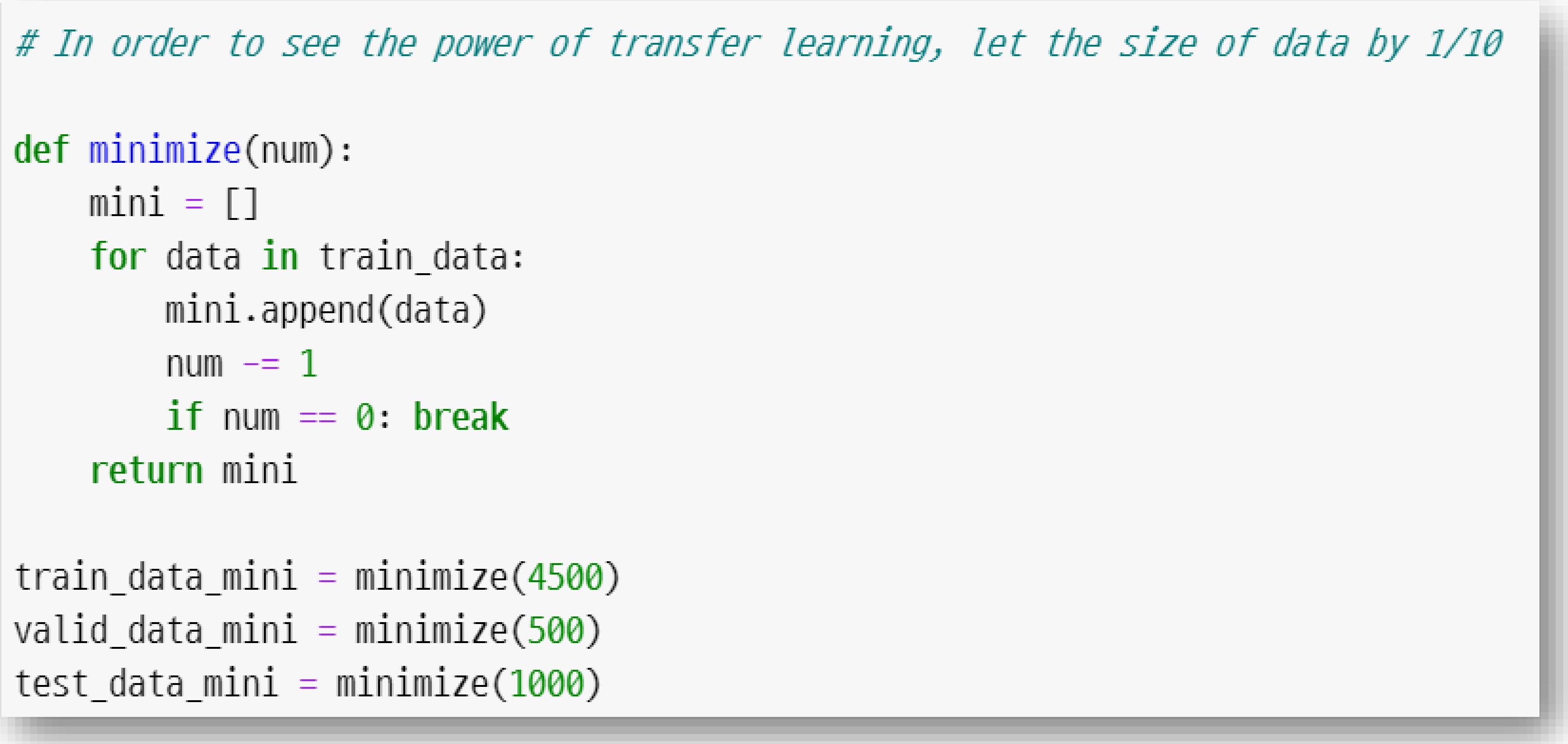
# Full code



```
from torchvision import models
```

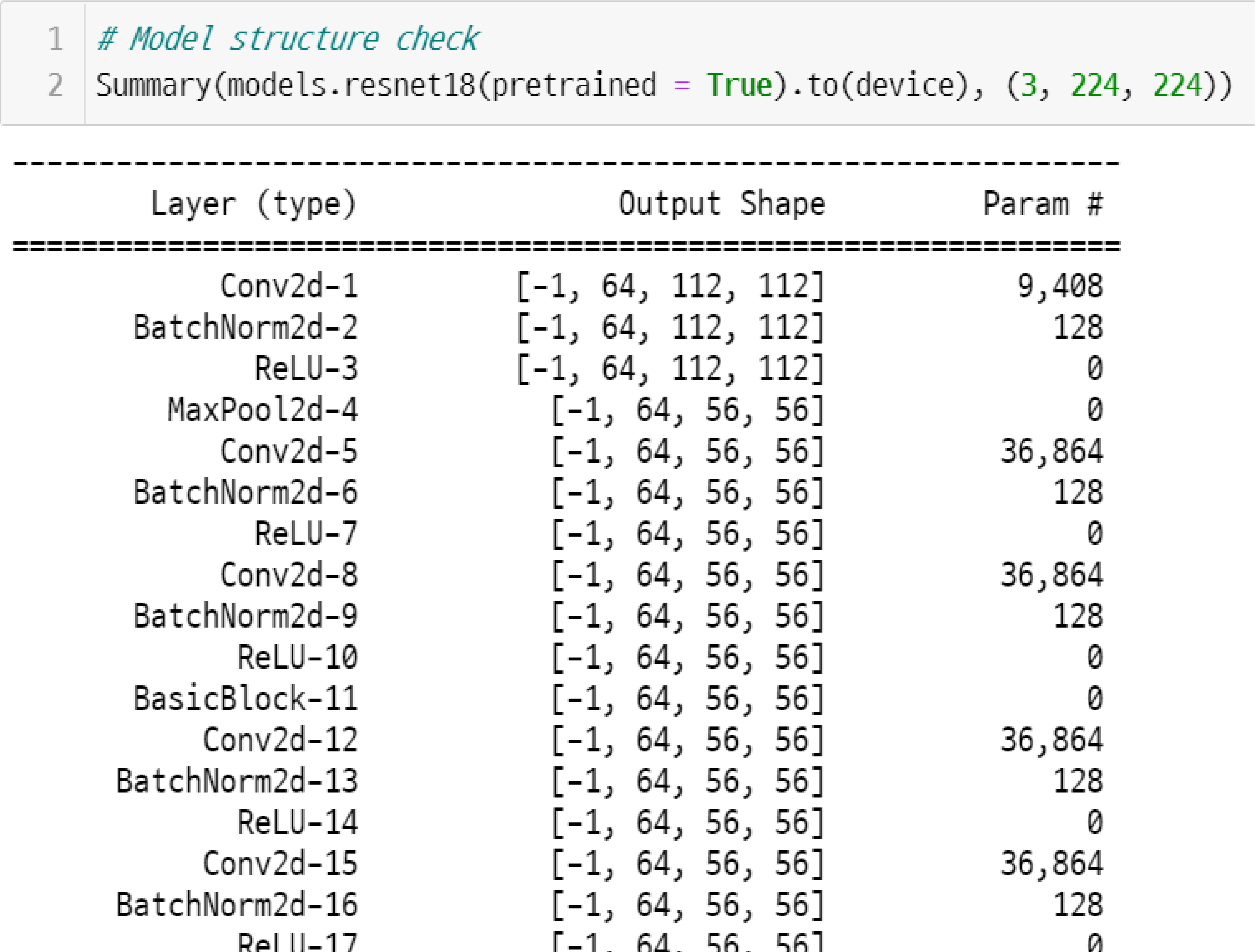
https://pytorch.org/vision/stable/models.html

# Full code



```
# In order to see the power of transfer learning, let the size of data by 1/10

def minimize(num):
    mini = []
    for data in train_data:
        mini.append(data)
        num -= 1
        if num == 0: break
    return mini

train_data_mini = minimize(4500)
valid_data_mini = minimize(500)
test_data_mini = minimize(1000)
```

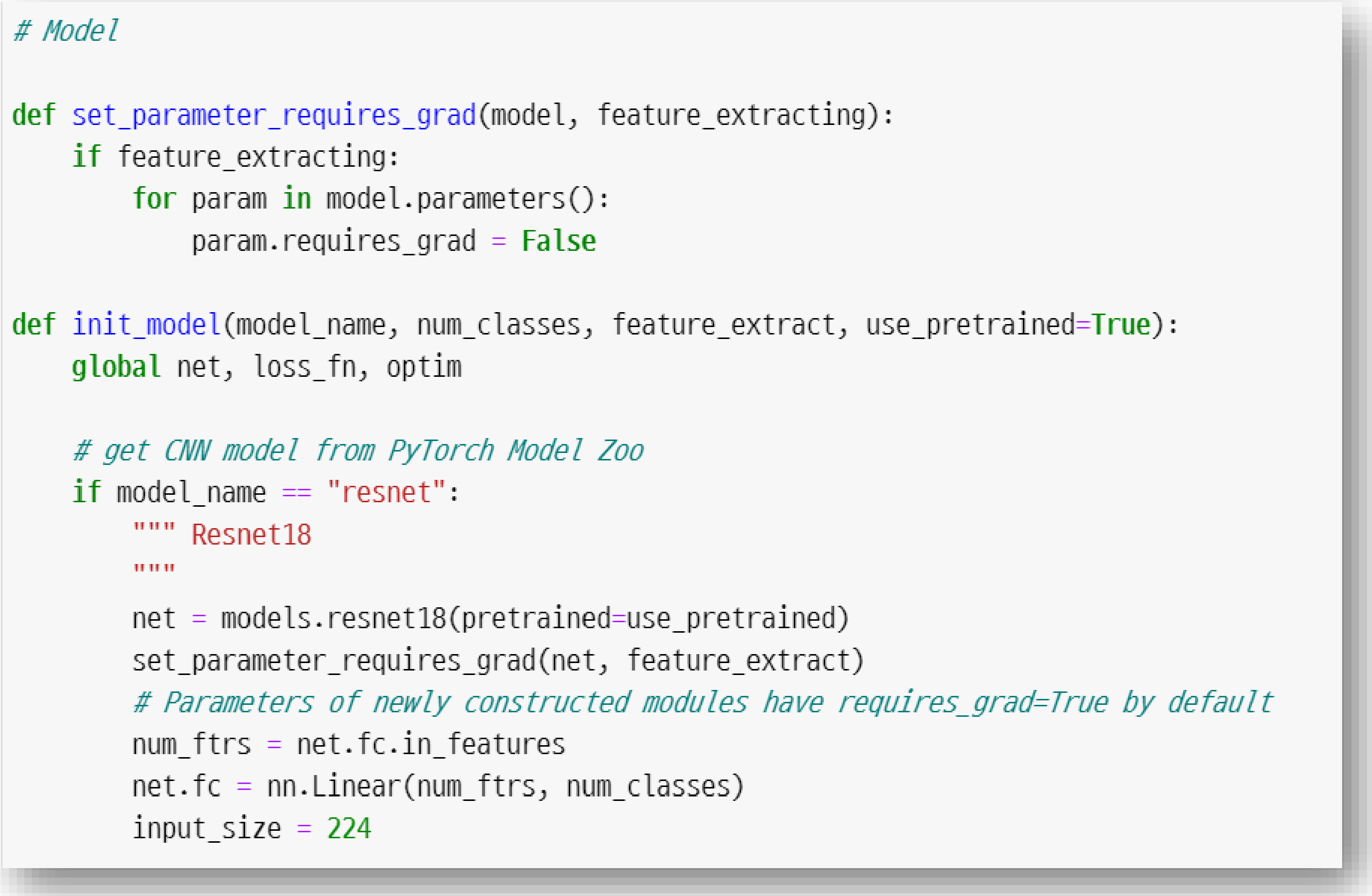
# Full code



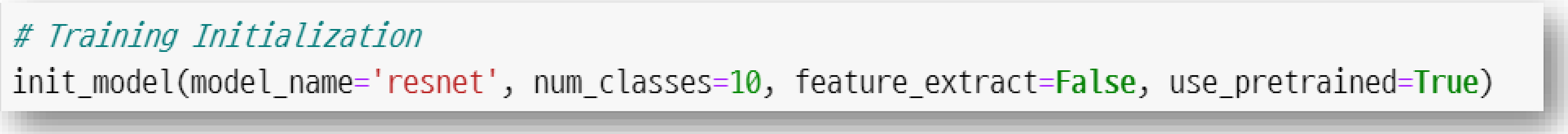
```python
# Model structure check
Summary(models.resnet18(pretrained = True).to(device), (3, 224, 224))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           9,408
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]          36,864
       BatchNorm2d-6           [-1, 64, 56, 56]             128
              ReLU-7           [-1, 64, 56, 56]               0
            Conv2d-8           [-1, 64, 56, 56]          36,864
       BatchNorm2d-9           [-1, 64, 56, 56]             128
             ReLU-10           [-1, 64, 56, 56]               0
       BasicBlock-11           [-1, 64, 56, 56]               0
           Conv2d-12           [-1, 64, 56, 56]          36,864
      BatchNorm2d-13           [-1, 64, 56, 56]             128
             ReLU-14           [-1, 64, 56, 56]               0
           Conv2d-15           [-1, 64, 56, 56]          36,864
      BatchNorm2d-16           [-1, 64, 56, 56]             128
             ReLU-17           [-1, 64, 56, 56]               0
```

# Full code

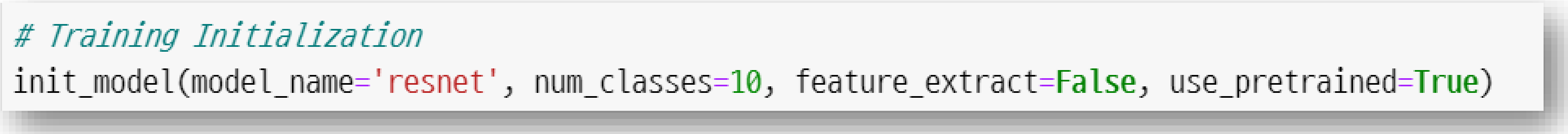


```python
# Model

def set_parameter_requires_grad(model, feature_extracting):
    if feature_extracting:
        for param in model.parameters():
            param.requires_grad = False

def init_model(model_name, num_classes, feature_extract, use_pretrained=True):
    global net, loss_fn, optim

    # get CNN model from PyTorch Model Zoo
    if model_name == "resnet":
        """ Resnet18
        """
        net = models.resnet18(pretrained=use_pretrained)
        set_parameter_requires_grad(net, feature_extract)
        # Parameters of newly constructed modules have requires_grad=True by default
        num_ftrs = net.fc.in_features
        net.fc = nn.Linear(num_ftrs, num_classes)
        input_size = 224
```



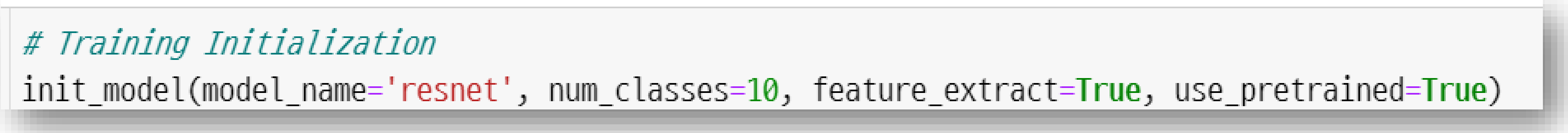
```python
# Training Initialization
init_model(model_name='resnet', num_classes=10, feature_extract=False, use_pretrained=True)
```

# Full code



```
# Training Initialization
init_model(model_name='resnet', num_classes=10, feature_extract=False, use_pretrained=True)
```

```
Test accuracy = 0.814484126984127
Test loss = 0.7271402364685422
```



```
# Training Initialization
init_model(model_name='resnet', num_classes=10, feature_extract=True, use_pretrained=True)
```

```
Test accuracy = 0.7847222222222222
Test loss = 0.6246340693462462
```