AI VIETNAM
All-in-One Course

Extra Class

Advanced CNN Architecture

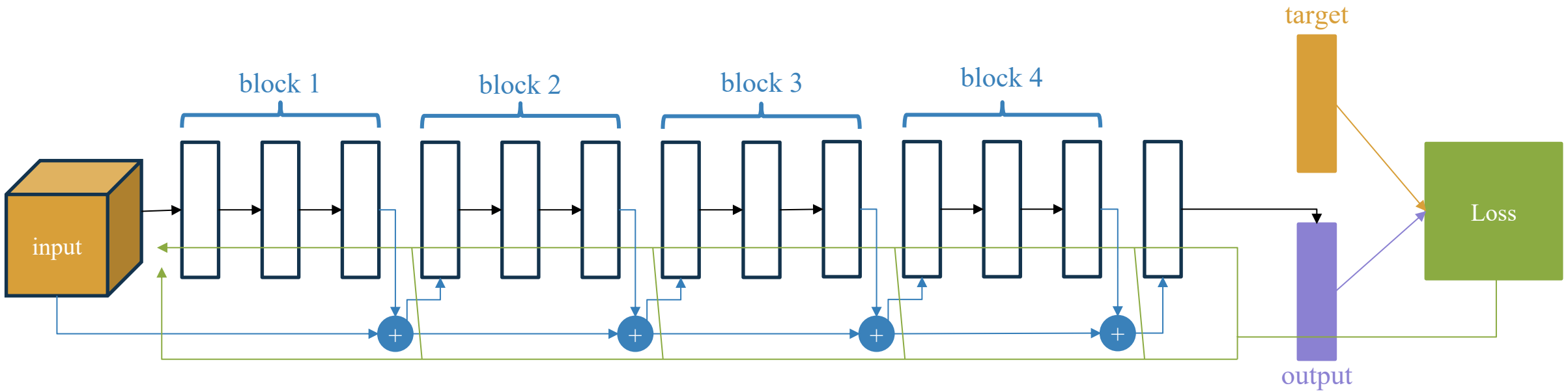
Nguyen Quoc Thai

CONTENT

AI VIET NAM
@aivietnam.edu.vn

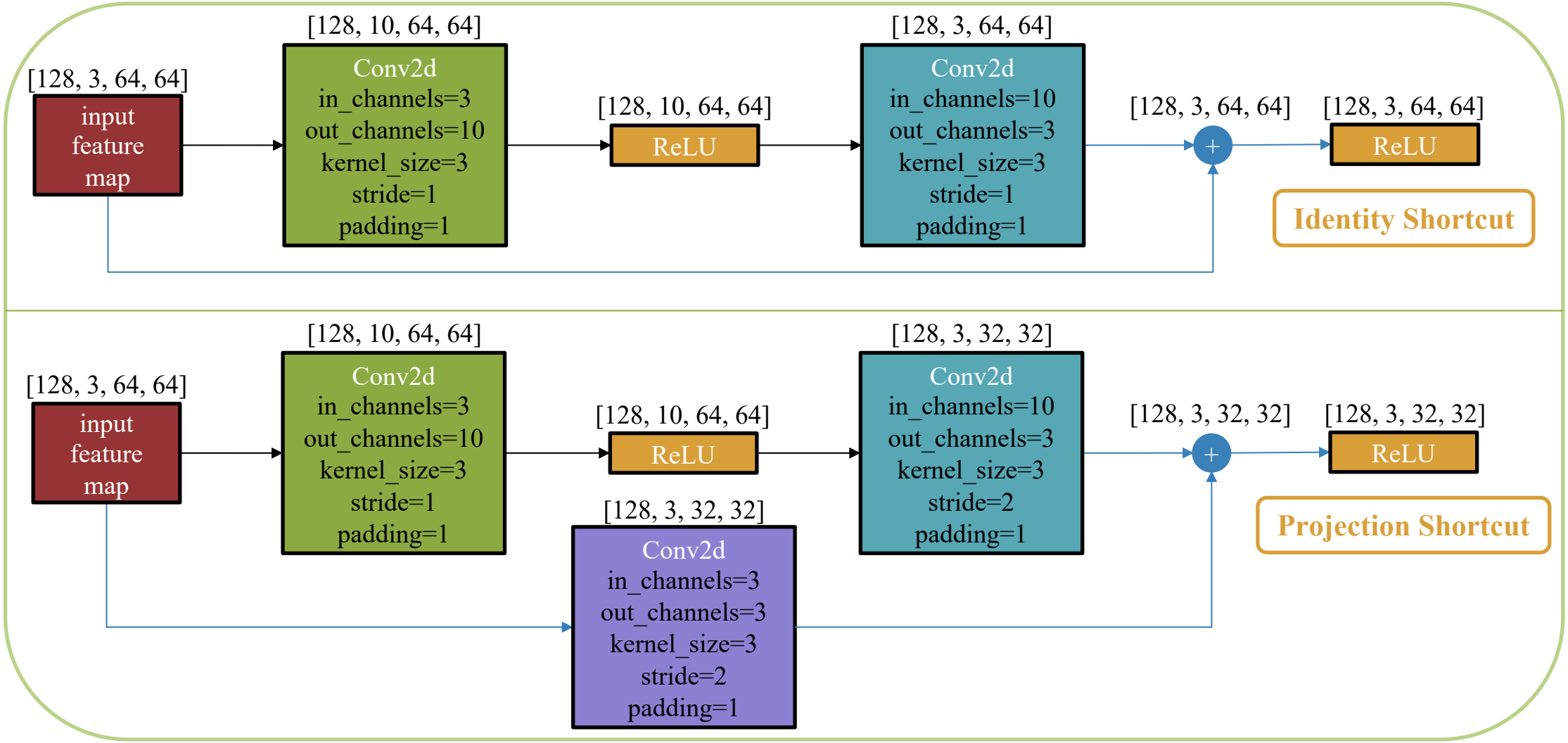
1 – ResNet

Review: Skip Connection



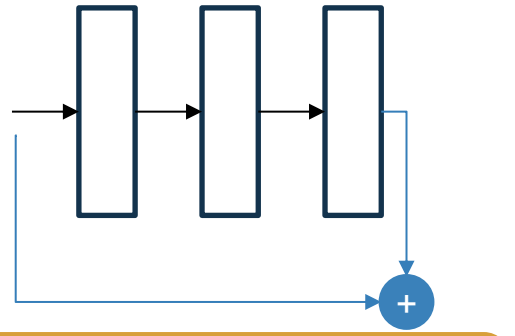
1 – ResNet

Review: Skip Connection

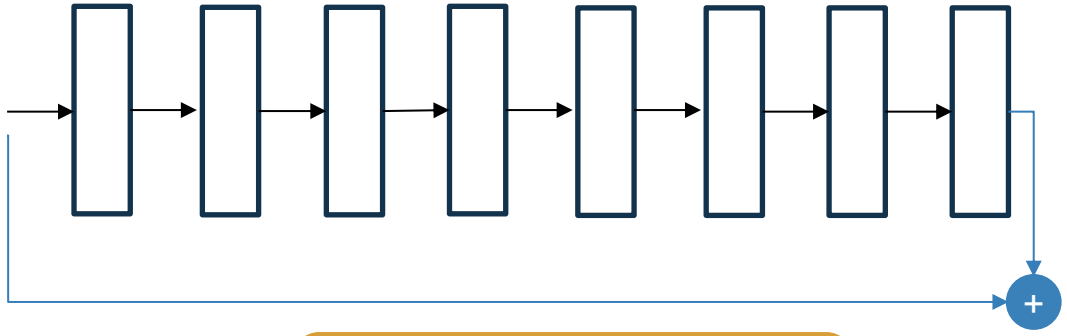


1 – ResNet

Review: Skip Connection



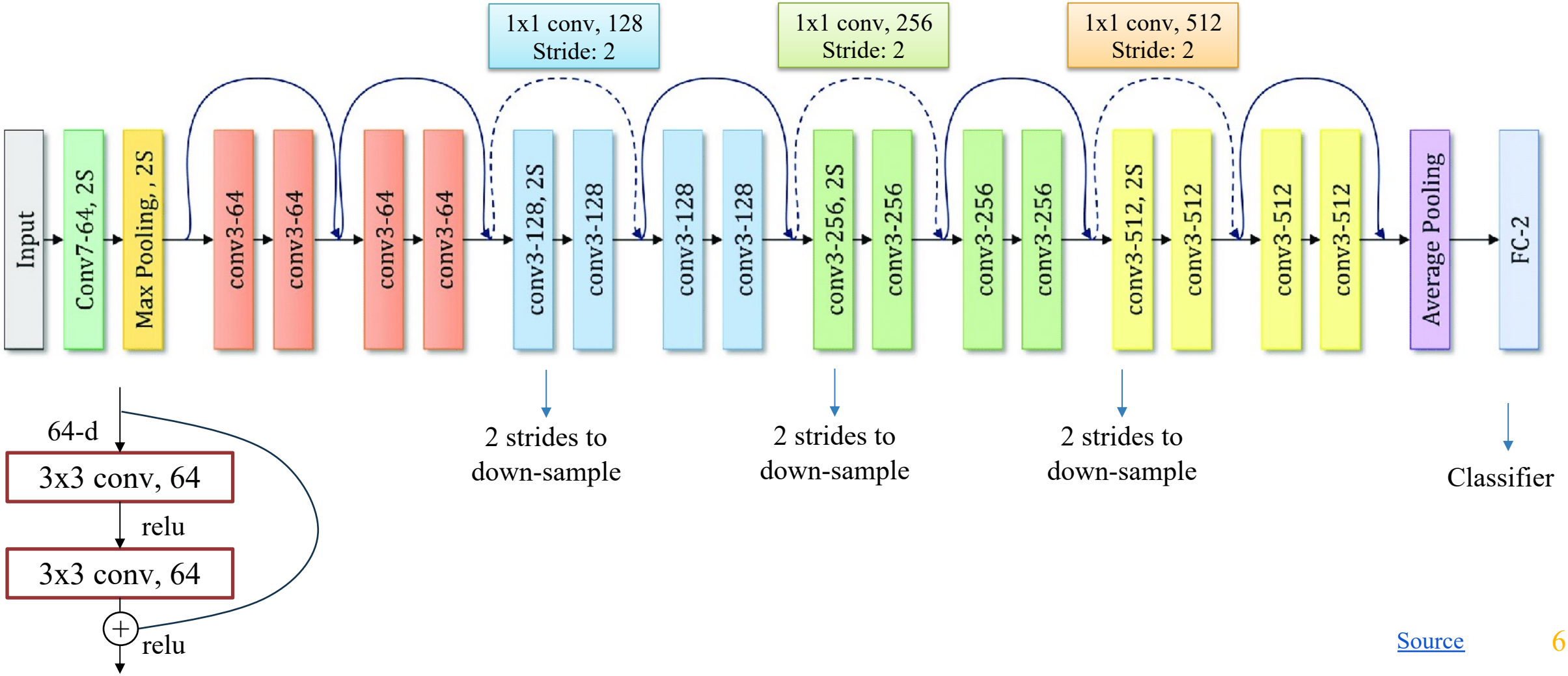
Short Skip Connection
(ResNet, …)



Long Skip Connection
(UNet, …)

1 – ResNet

ResNet18



Source

1 – ResNet

ResNet18

```
import torch

from torchvision import models
from torchsummary import summary
```

```
resnet_model = models.resnet18(weights=None)
```

```
==========================================
Total params: 11,689,512
Trainable params: 11,689,512
Non-trainable params: 0
------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 62.79
Params size (MB): 44.59
Estimated Total Size (MB): 107.96
------------------------------------------
```

1 – ResNet

ResNet (18 – 34 – 50 – 101 – 152)

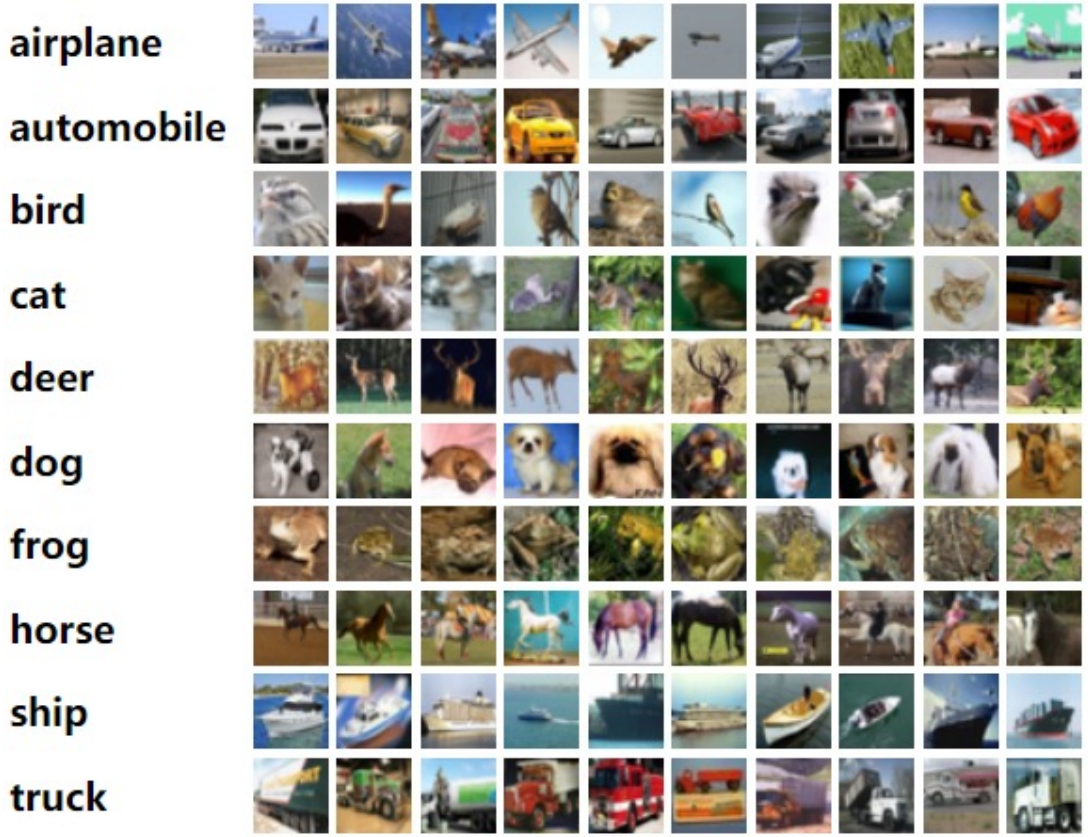
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				

Source

1 – ResNet

CIFAR10 Classification using ResNet18

- Training: 50,000 images. Testing: 10,000 images
- Classes: 10



1 – ResNet

CIFAR10 Classification using ResNet18 – Demo

- Load Dataset

```
ROOT = './data'

train_data = datasets.CIFAR10(
    root=ROOT,
    train=True,
    download=True,
    transform=transforms.ToTensor()
)

test_data = datasets.CIFAR10(
    root=ROOT,
    train=False,
    download=True,
    transform=transforms.ToTensor()
    )
```

1 – ResNet

CIFAR10 Classification using ResNet18 – Demo

- Preprocessing

```
train_data_stack = torch.stack([img for img, _ in train_data], dim=3)
```

```
train_data_stack.shape
```

```
torch.Size([3, 32, 32, 50000])
```

```
mean= train_data_stack.view(3,-1).mean(dim=1)
std= train_data_stack.view(3,-1).std(dim=1)
mean, std
```

```
(tensor([0.4914, 0.4822, 0.4465]), tensor([0.2470, 0.2435, 0.2616]))
```

```
data_transforms = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(mean, std)
])
```

```
train_data.transform = data_transforms
test_data.transform = data_transforms
```

```
BATCH_SIZE = 512

train_dataloader = data.DataLoader(
    train_data,
    shuffle=True,
    batch_size=BATCH_SIZE
)

test_dataloader = data.DataLoader(
    test_data,
    batch_size=BATCH_SIZE
)
```

1 – ResNet

CIFAR10 Classification using ResNet18 – Demo

- Model

```
model = models.resnet18(weights=None)
in_features = model.fc.in_features
model.fc = nn.Linear(in_features, len(train_data.classes))
model.to(device)
```

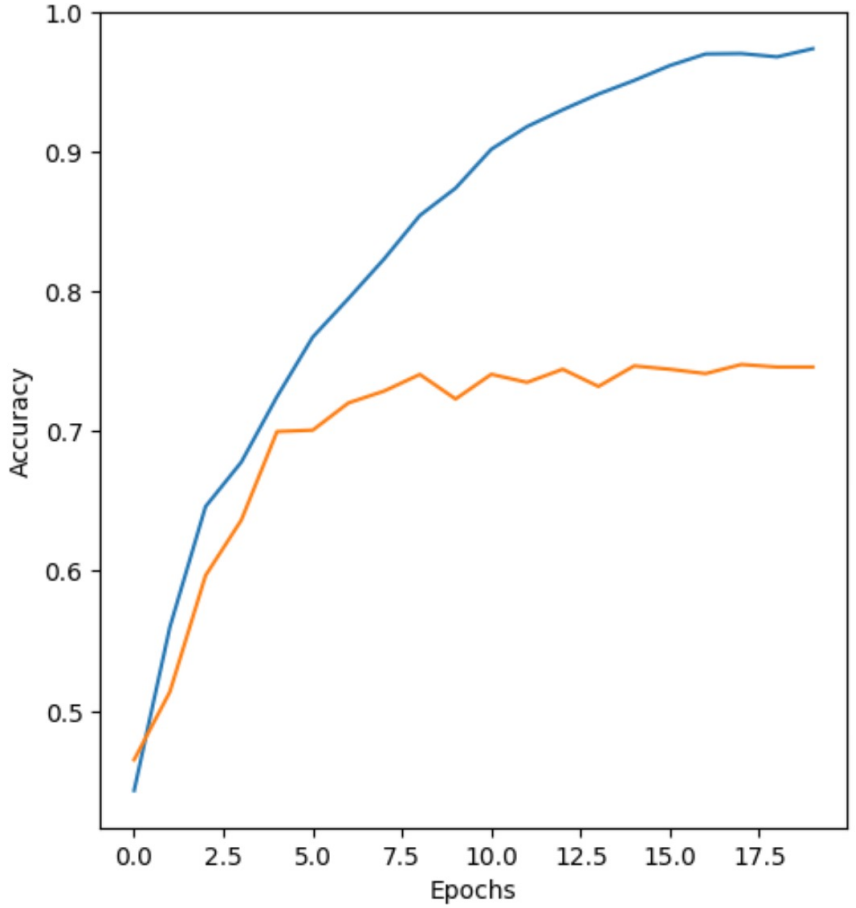
```
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=512, out_features=1000, bias=True)
```

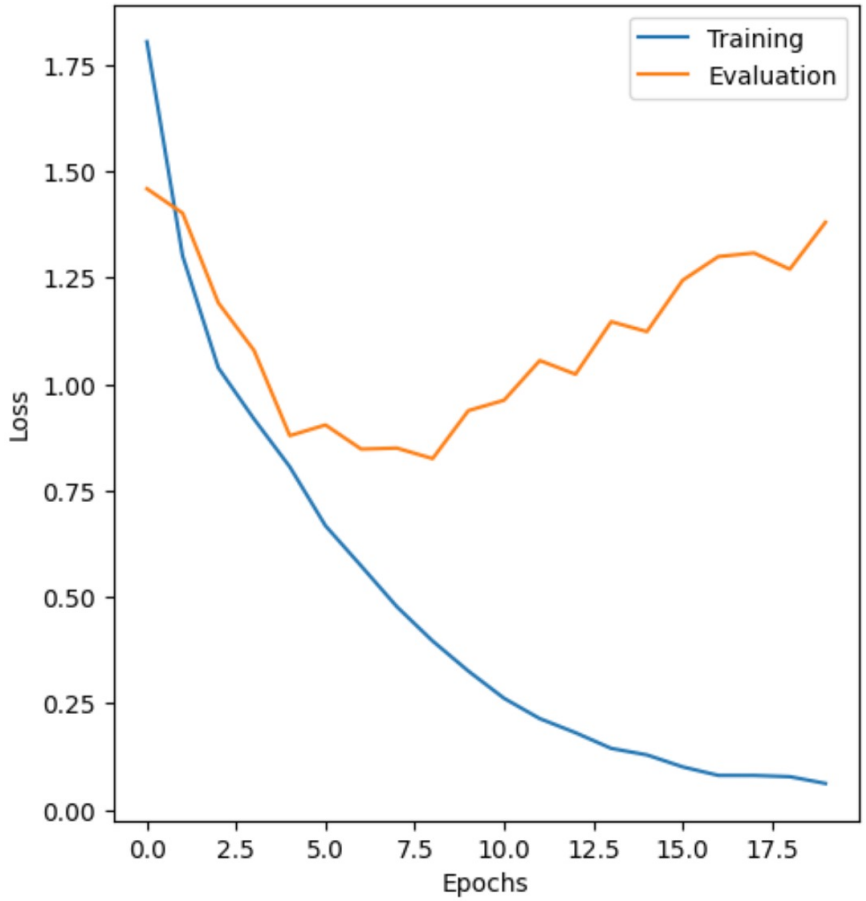
```
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=512, out_features=10, bias=True)
```

1 – ResNet

CIFAR10 Classification using ResNet18 – Demo

- Training

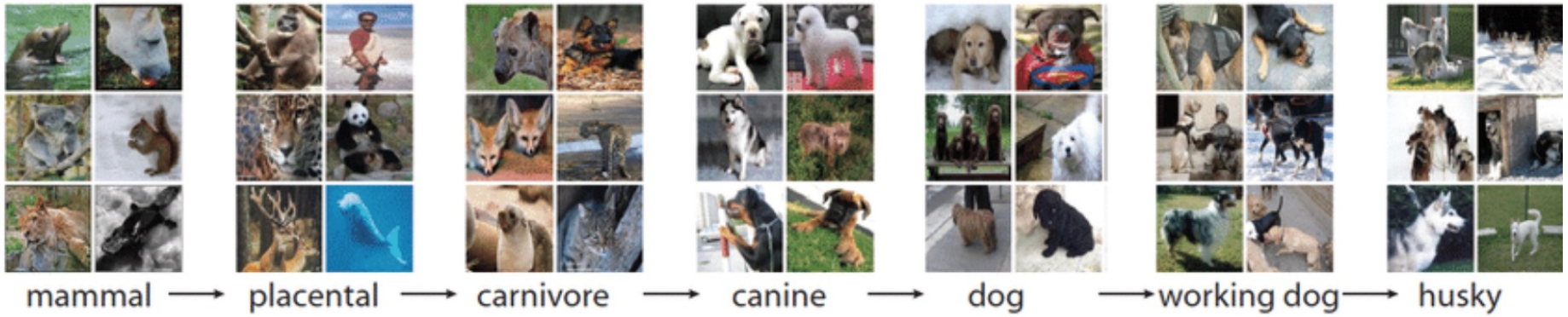




2 – Pretrained Models

ImageNet

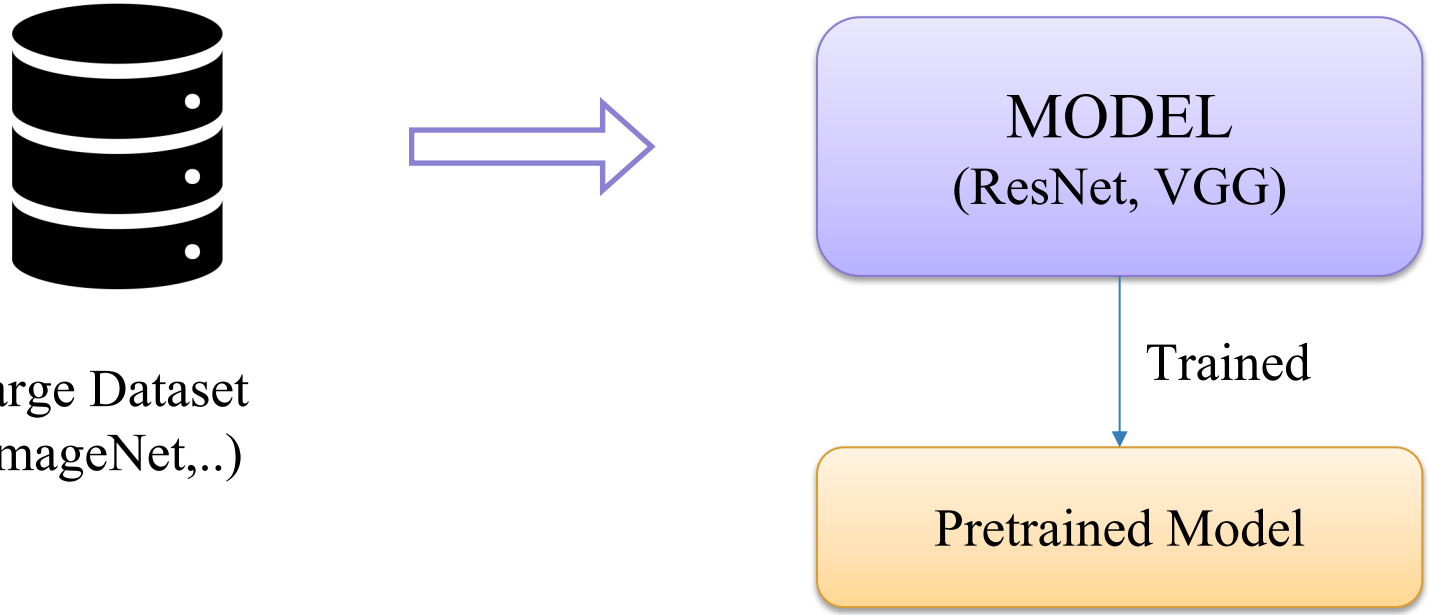
- Training: 1,281,167 images. Validation: 50,000 images. Testing: 100,000 images
- Object classes: 1,000



2 – Pretrained Models

ImageNet

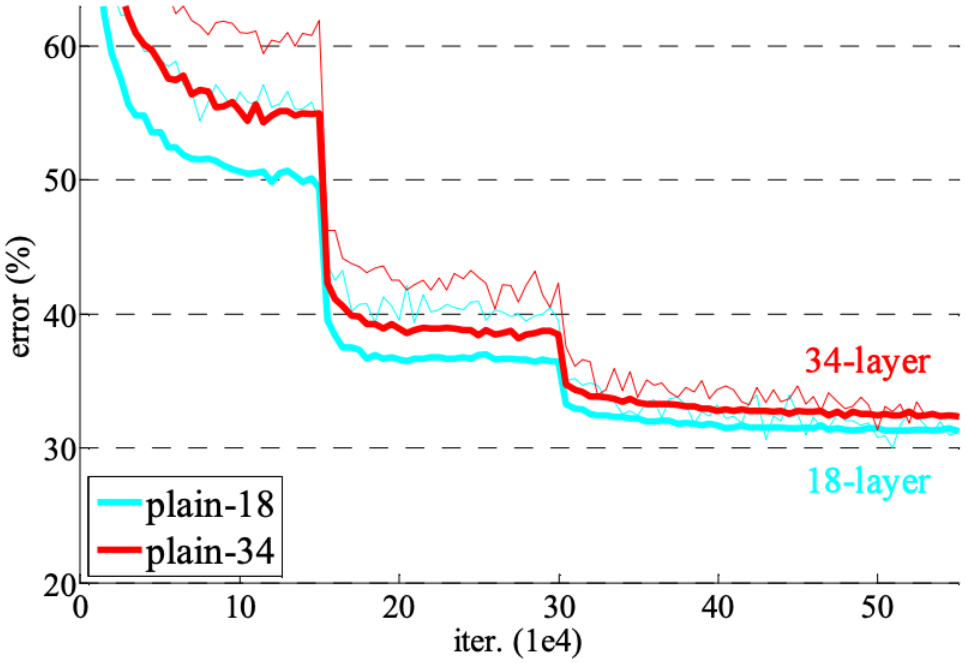
- Training: 1,281,167 images. Validation: 50,000 images. Testing: 100,000 images
- Object classes: 1,000

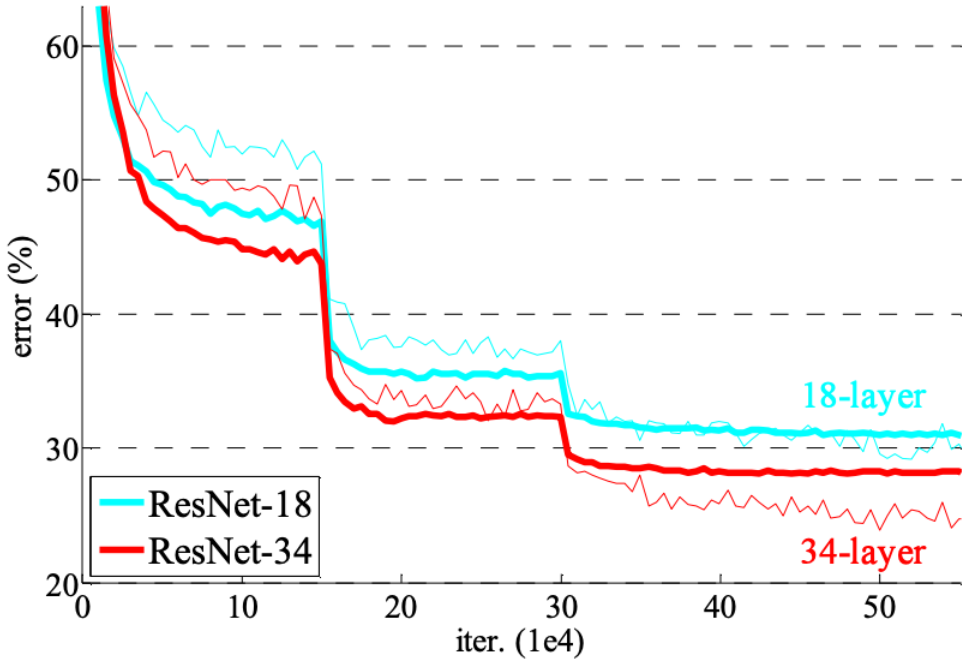


2 – Pretrained Models

ImageNet

- Training: 1,281,167 images. Validation: 50,000 images. Testing: 100,000 images
- Object classes: 1,000



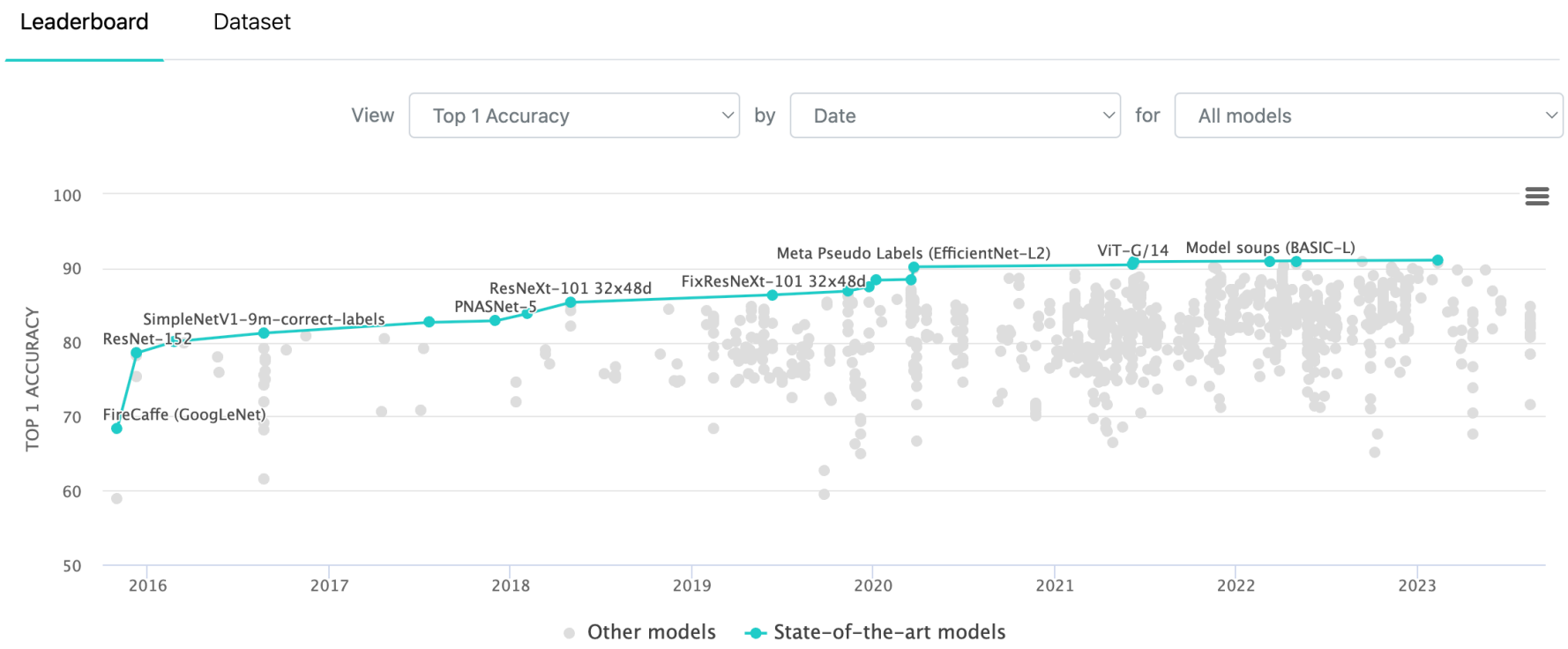


Source

2 – Pretrained Models

ImageNet

- Training: 1,281,167 images. Validation: 50,000 images. Testing: 100,000 images
- Object classes: 1,000

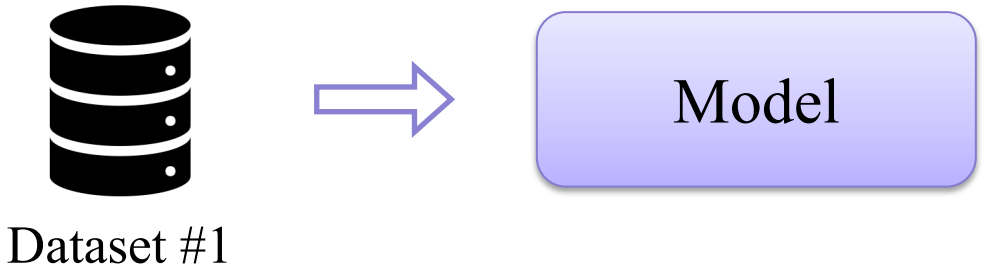


Source

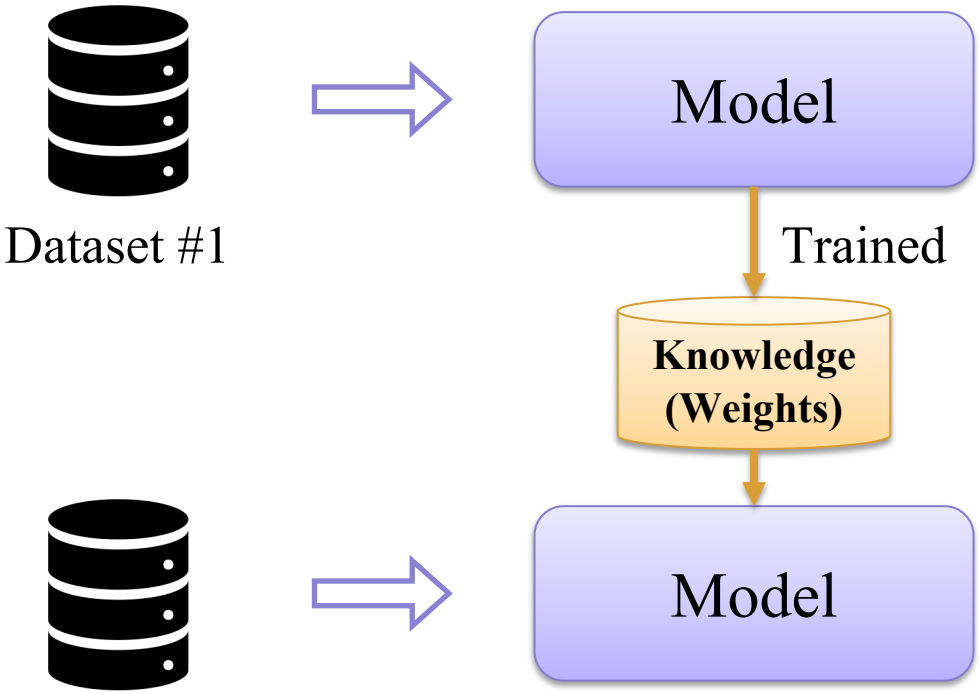
2 – Pretrained Models

Transfer Learing

- Traditional Learning: Isolated, single task learning



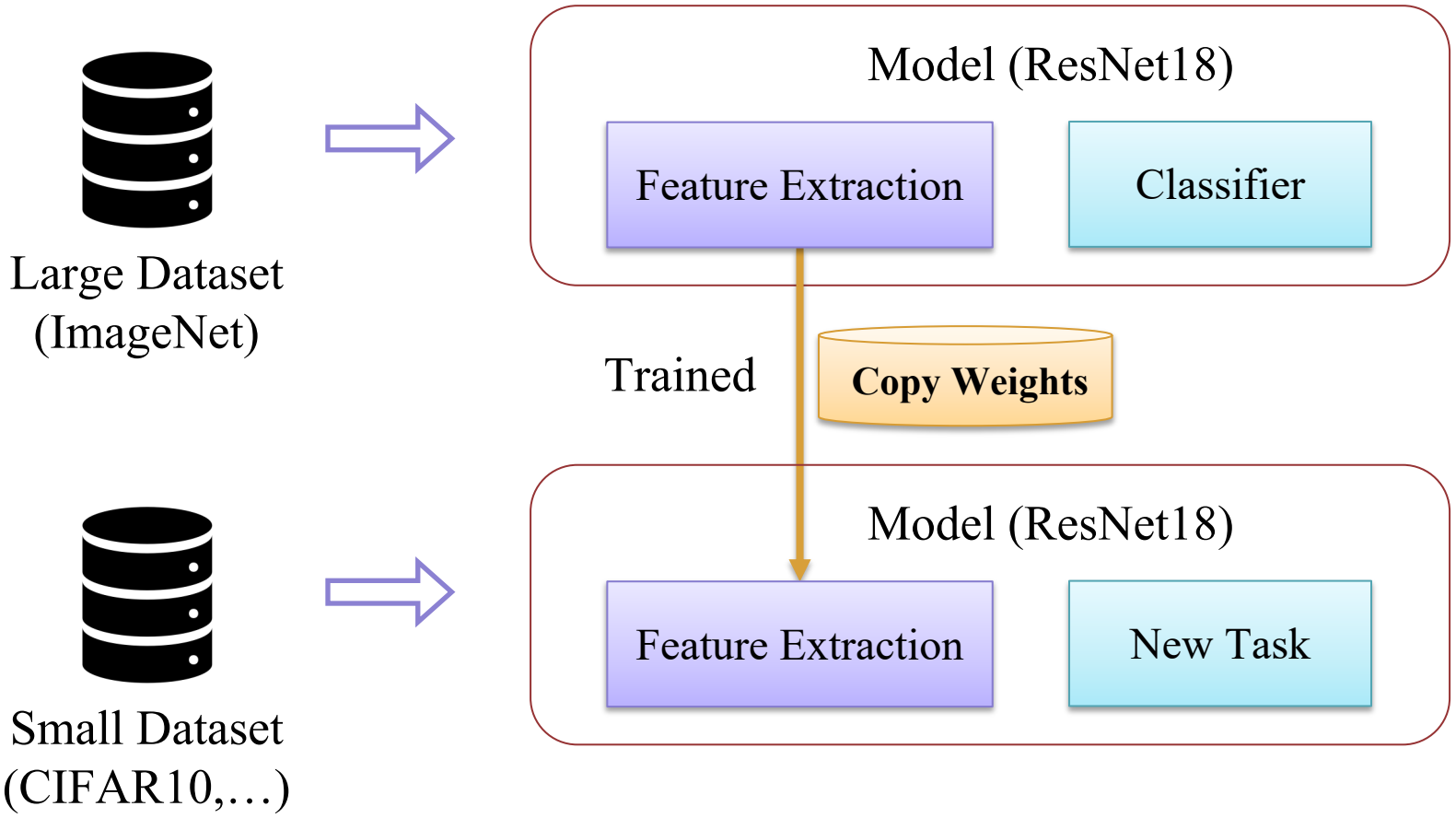
- Transfer Learning: Learning of a new tasks relies on the previous learned tasks



2 – Pretrained Models

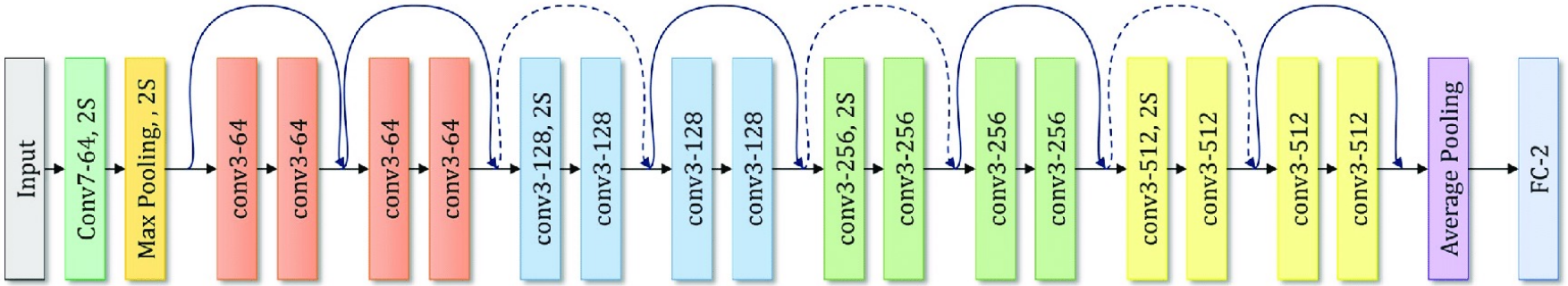
Transfer Learing

- Transfer Learning: Feature Extractor



2 – Pretrained Models

Feature Extraction using Pretrained Models



Feature Extractor

Pretraied Model (ResNet18)

Vector
Matrix

New Task

2 – Pretrained Models

Feature Extraction using Pretrained Models - Demo

```
pretrained_model = models.resnet18(
    weights=models.ResNet18_Weights.IMAGENET1K_V1
)
```

```
Downloading: "https://download.pytorch.org/models/re
100%|██████████| 44.7M/44.7M [00:00<00:00, 135MB/s]
```

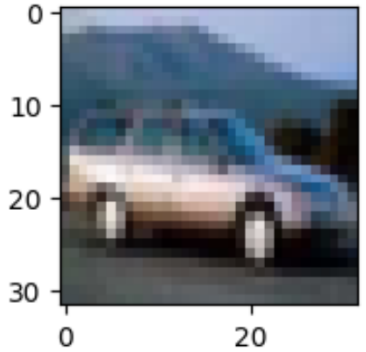
```
# remove classifier layer (block)
resnet_feature_extractor = nn.Sequential(
    *list(pretrained_model.children())[:-1]
)

resnet_feature_extractor.eval()
```

2 – Pretrained Models

Feature Extraction using Pretrained Models - Demo

```
plt.figure(figsize=(2, 2))
plt.imshow(train_data.data[4])
```

```
<matplotlib.image.AxesImage at 0x7ddf7dbf86d0>
```




```
train_data.data[4].shape
```

```
(32, 32, 3)
```

```
processed_image = data_transforms(train_data.data[4])
```

```
processed_image.shape
```

```
torch.Size([3, 32, 32])
```

```
prediction = resnet_feature_extractor(processed_image.unsqueeze(0))
```

```
prediction.squeeze().shape
```

```
torch.Size([512])
```

3 – Image Similarity

1 Similarity between two images

Feature Extractor

Feature Extractor

Similarity/
Distance Function

Similarity Score /
Distance

MSE
L1 – Distance
Cosine Similarity
Normalized Cross-Correlation

3 – Image Similarity

1 Similarity between two images - MSE

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$Y = [y_1, y_2, y_3, \ldots, y_n]$$
$$\hat{Y} = [\hat{y}_1, \hat{y}_2, \hat{y}_3, \ldots, \hat{y}_n]$$

```
1 def mse(tensor1, tensor2):
2     return torch.mean((tensor1 - tensor2) ** 2)
```

3 – Image Similarity

1 Similarity between two images - MSE

1.0	2.0	4.0	5.0

\-

1.2	2.4	3.4	5.6

$=$ $($ | -0.2 | -0.4 | 0.6 | -0.6 | $)^2 \rightarrow$ | 0.04 | 0.16 | 0.36 | 0.36 | $\xrightarrow{\text{mean}}$ 0.23

100	105	98	110

\-

102	107	97	108

$=$ $($ | -2 | -2 | 1 | ? | $)^2 \rightarrow$ | 4 | 4 | 1 | ? | $\xrightarrow{\text{mean}}$?

3 – Image Similarity

5

Similarity between two images – on raw images

MSE: 0.4063

MSE: 0.3265

3 – Image Similarity

3

Similarity between two images –L1 Distance

$$L1 = \sum_{i=1}^{N} |p_i - q_i|$$

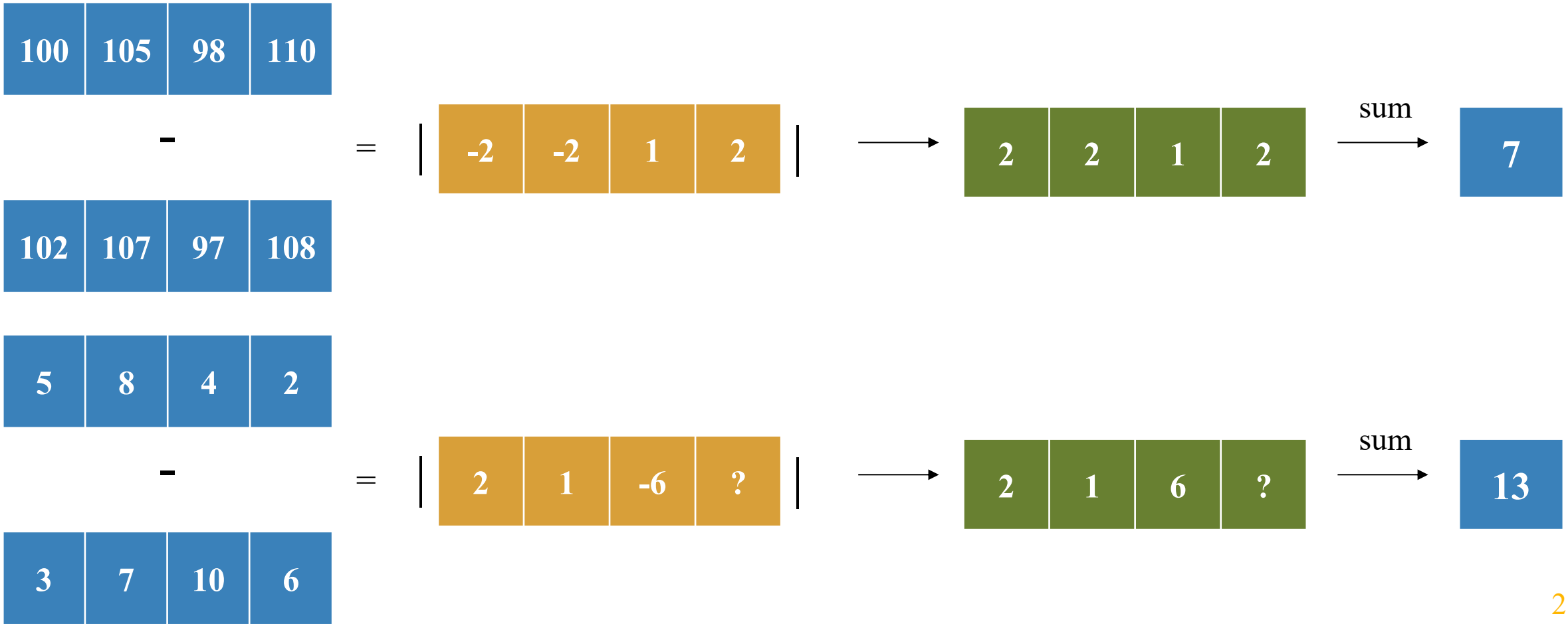
$$Y = [y_1, y_2, y_3, \ldots, y_n]$$
$$\hat{Y} = [\hat{y}_1, \hat{y}_2, \hat{y}_3, \ldots, \hat{y}_n]$$

```python
def l1_distance(tensor1, tensor2):
    return torch.sum(torch.abs(tensor1 - tensor2))
```

3 – Image Similarity

3

Similarity between two images –L1 Distance



3 – Image Similarity

5

Similarity between two images – on raw images

L1: 51128.43

L1: 40339.5

3 – Image Similarity

5

Similarity between two images – on raw images

L1: 46423.3
MSE: 0.3477

L1: 40339.5
MSE: 0.3265

Problem?

3 – Image Similarity

2

Similarity between two images – Cosine Similarity

$$cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

```python
def cosine_similarity(tensor1, tensor2):
    return torch.nn.functional.cosine_similarity(tensor1, tensor2, dim=0)
```

```python
def cosine_similarity_v2(tensor1, tensor2):
    tensor1_norm  = torch.norm(tensor1)
    tensor2_norm  = torch.norm(tensor2)
    return torch.dot(tensor1, tensor2) / (tensor1_norm * tensor2_norm)
```

$$A = [a_1, a_2, a_3, \dots, a_n]$$
$$B^T = [b_1, b_2, b_3, \dots, b_n]$$

$$A.B = a_1b_1 + a_2b_2 + \cdots a_nb_n$$

$$||A|| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$$

$$||B|| = \sqrt{b_1^2 + b_2^2 + \cdots + b_n^2}$$

3 – Image Similarity

2

Similarity between two images – Cosine Similarity

$$cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

1.0	2.0	3.0	4.0

$*$

5.0
6.0
7.0
8.0

$$= \frac{1*5+2*6+3*7+4*8}{\sqrt{1^2+2^2+3^2+4^2}\sqrt{5^2+6^2+7^2+8^2}} = 0.97$$

3	2	4	6

$*$

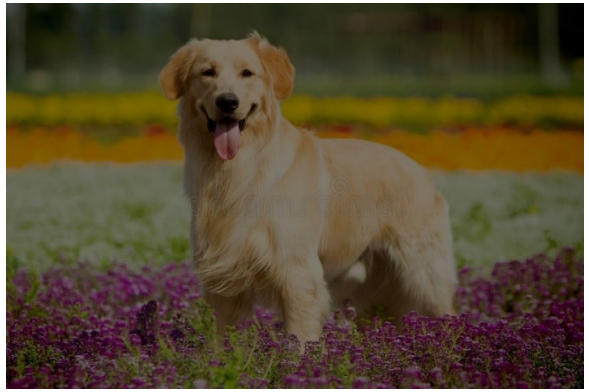
9
3
2
6

$$= \frac{?}{\sqrt{?}*\sqrt{?}} = ?$$

3 – Image Similarity

5

Similarity between two images – on raw images



Cosine sim: 1.0

Cosine sim: 0.83

3 – Image Similarity

4 Similarity between two images – Normalized Cross-Correlation

Formula

$$NCC = \frac{1}{IJ}\sum_{j=1}^{J}\sum_{i=1}^{I}\frac{(A_{i,j}-a)(B_{i,j}-b)}{\sigma_A\sigma_B}$$

$$a = \frac{1}{IJ}\sum_{j=1}^{J}\sum_{i=1}^{I}A_{i,j} \qquad b = \frac{1}{IJ}\sum_{j=1}^{J}\sum_{i=1}^{I}B_{i,j}$$

$$\sigma_A = \sqrt{\frac{1}{IJ}\sum_{j=1}^{J}\sum_{i=1}^{I}(A_{i,j}-a)^2} \qquad \sigma_B = \sqrt{\frac{1}{IJ}\sum_{j=1}^{J}\sum_{i=1}^{I}(B_{i,j}-b)^2}$$

3 – Image Similarity

4

Similarity between two images – Normalized Cross-Correlation

```python
def normalized_cross_correlation(tensor1, tensor2):
    mean1 = torch.mean(tensor1)
    mean2 = torch.mean(tensor2)

    # Calculate the centered tensors
    centered1 = tensor1 - mean1
    centered2 = tensor2 - mean2

    # Calculate the NCC
    return torch.sum(centered1 * centered2) / (torch.sqrt(torch.sum(centered1 ** 2)) * torch.sqrt(torch.sum(centered2 ** 2)))
```

3 – Image Similarity

4 **Similarity between two images – Normalized Cross-Correlation**

3 – Image Similarity

4 Similarity between two images – Normalized Cross-Correlation

-3	-2	0	5

* =

-4.5	?	0	-2.5

1.5	0.5	-1.5	-0.5

$$\frac{-8}{\sqrt{38} * \sqrt{5}} = ?$$

9	4	0	25

Sum → 38

2.25	0.25	2.25	0.25

Sum → 5

3 – Image Similarity

5

Similarity between two images – on raw images

NCC: 0.99

NCC: 0.3937

3 – Image Similarity

5

Similarity between two images – on raw images

Method	Similarity
L1 Distance	51128.43
MSE	0.4063
Cosine similarity	0.7620
Normalized Cross Correlation	-0.3564

3 – Image Similarity

5

Similarity between two images – on raw images

Method	Similarity
L1 Distance	40339.5
MSE	0.3265
Cosine similarity	0.8294
Normalized Cross Correlation	0.3937

3 – Image Similarity

5

Similarity between two images – on raw images

Method	Similarity
L1 Distance	46423.3
MSE	0.3477
Cosine similarity	1.0000
Normalized Cross Correlation	0.9999

3 – Image Similarity

5

Similarity between two images – on raw images

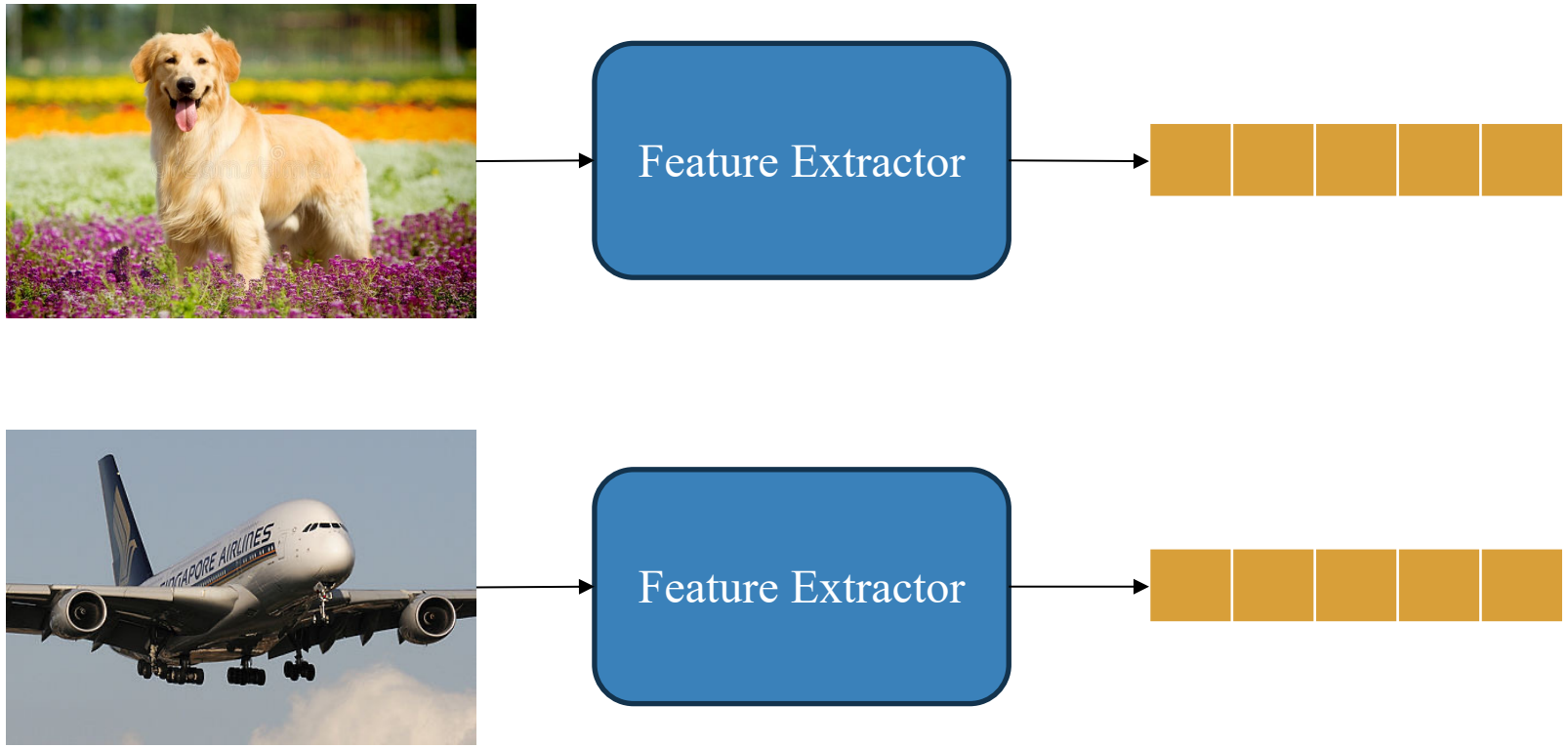
Method	Similarity
L1 Distance	30073.23
MSE	0.2542
Cosine similarity	0.8691
Normalized Cross Correlation	0.3812

Problem?

3 – Image Similarity

6

Similarity between two images – using resnet18



3 – Image Similarity

6 Similarity between two images – using resnet18

Load the model →

```
1 import torchvision
2 from torchvision import models
3
4 model = torchvision.models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)
```

```
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/t
100%|██████████| 44.7M/44.7M [00:00<00:00, 127MB/s]
```

```
1 tensor_dog1.shape
```

```
torch.Size([3, 224, 224])
```

```
1 model.fc = nn.Identity()
```

Compute feature →

```
1 dog1_feature = model(tensor_dog1.unsqueeze(0))
2 dog2_feature = model(tensor_dog2.unsqueeze(0))
3 dog3_feature =  model(tensor_dog3.unsqueeze(0))
4 airplane_feature =  model(tensor_airplane.unsqueeze(0))
```

3 – Image Similarity

6 Similarity between two images – using resnet18

Feature Extractor

Feature Extractor

Similarity/
Distance Function

Similarity Score /
Distance

3 – Image Similarity

6

Similarity between two images – using resnet18

Method	Similarity
L1 Distance	53.34
MSE	0.1329
Cosine similarity	0.8999
Normalized Cross Correlation	0.4541

3 – Image Similarity

6

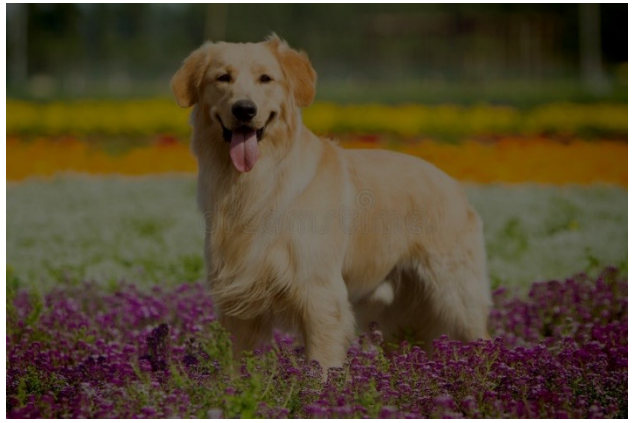
Similarity between two images – using resnet18

Method	Similarity
L1 Distance	49.47
MSE	0.123
Cosine similarity	0.912
Normalized Cross Correlation	0.595

3 – Image Similarity

6

Similarity between two images – using resnet18



Method	Similarity
L1 Distance	4.344
MSE	0.011
Cosine similarity	0.9993
Normalized Cross Correlation	0.9961

3 – Image Similarity

5

Similarity between two images – on raw images

Method	Similarity
L1 Distance	22.1067
MSE	0.0549
Cosine similarity	0.9983
Normalized Cross Correlation	0.7959

AI VIET NAM

@aivietnam.edu.vn

Thanks!

Any questions?