KSME 인공지능머신연구회 고급인공지능 강습회, January 19-21, 2022

Self Supervised Learning

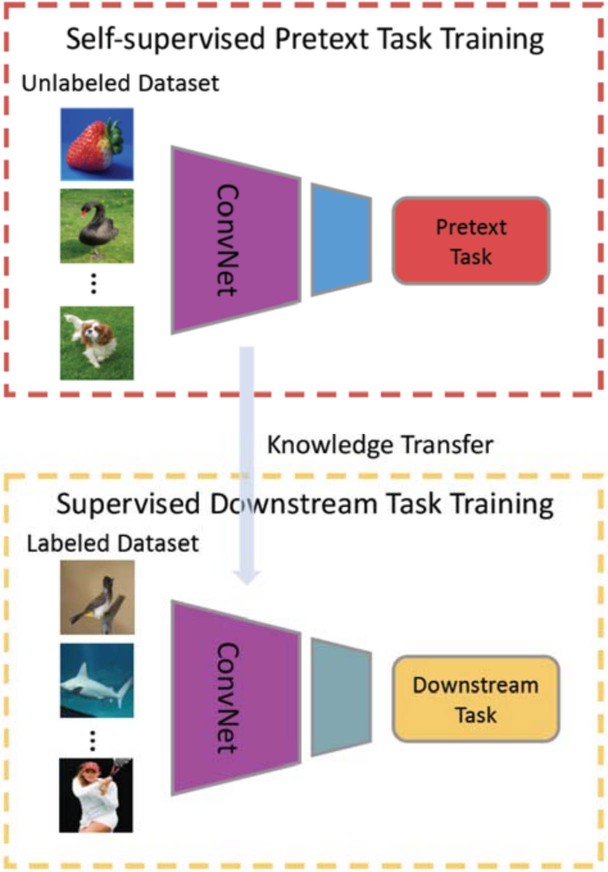
Prof. Hyunseok Oh

School of Mechanical Engineering
Gwangju Institute of Science and Technology

Self-supervised Learning

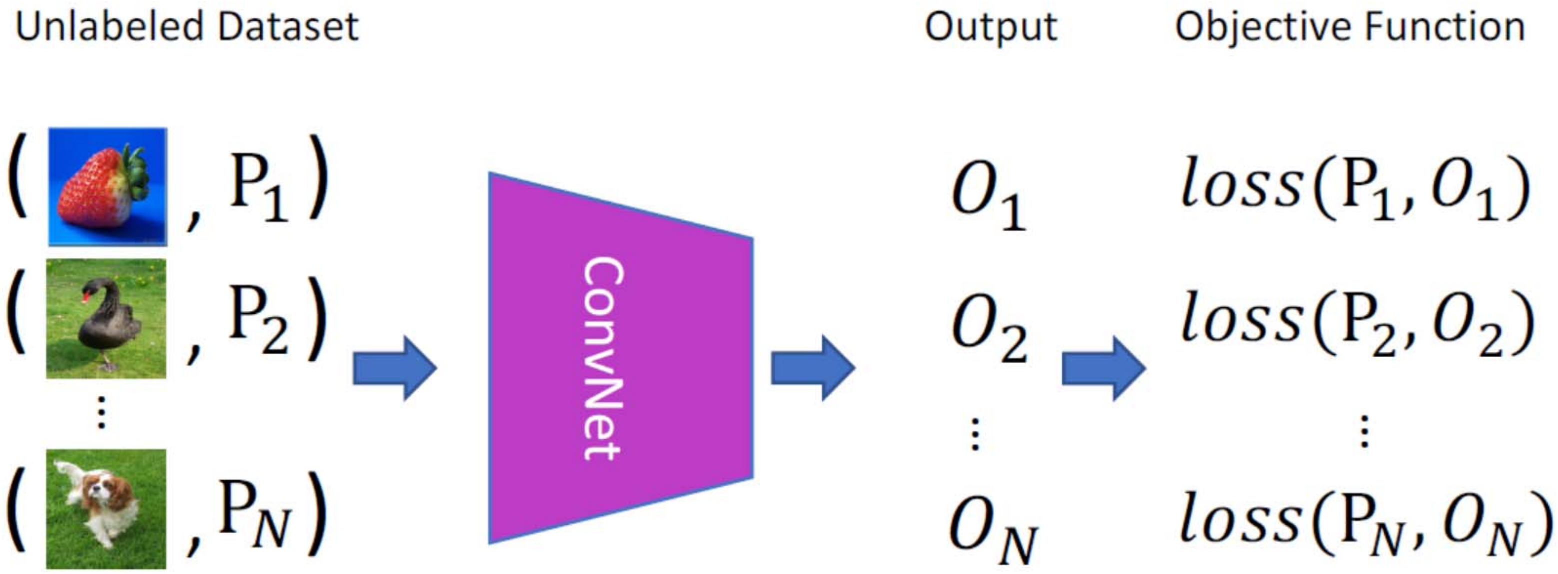
- Supervised learning is powerful. However, it requires a "large" amount of "labeled" data. To relieve the burden,
  - Transfer learning
  - Semi-supervised learning
  - Weakly-supervised
  - Unsupervised learning
- Self-supervised learning (SSL) is
  - Sub-class of unsupervised learning
  - To define pretext tasks which can be formulated using only unlabeled data. It requires higher-level semantic understanding in order to solve.
  - The features obtained from pretext tasks can be successfully transferred to classification, object detection, and segmentation tasks.

General Pipeline of SSL



- Visual Features are learned by pretext tasks.
- The learned parameters serve as a pre-trained model.
- They are transferred to other downstream. Then, fine-tuned.
- The performance of downstream tasks is used to evaluate the quality of learned features.

Pretext Task



Pretext Task: Generation-based Methods

- “Colorful image colorization”, ECCV, 2016.

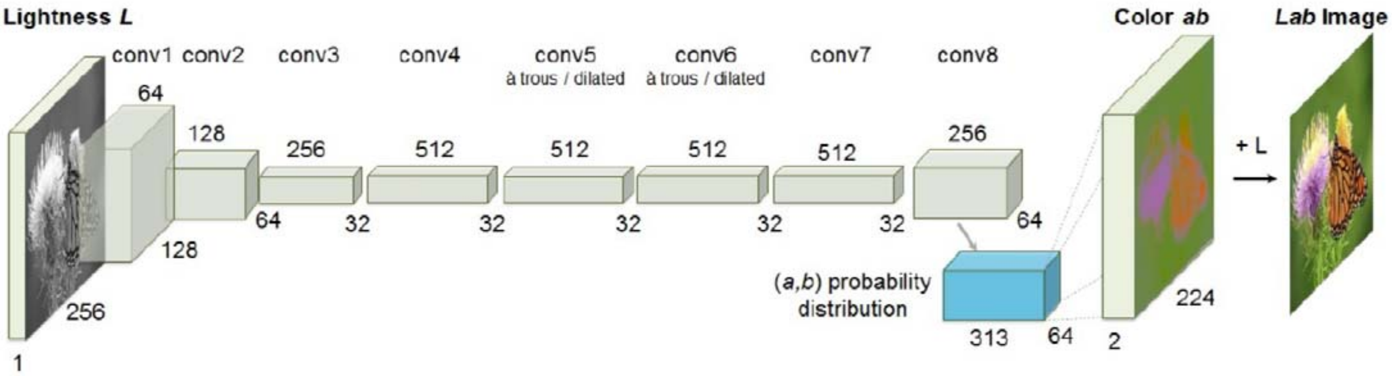
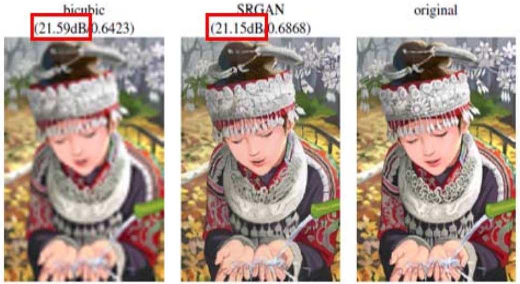


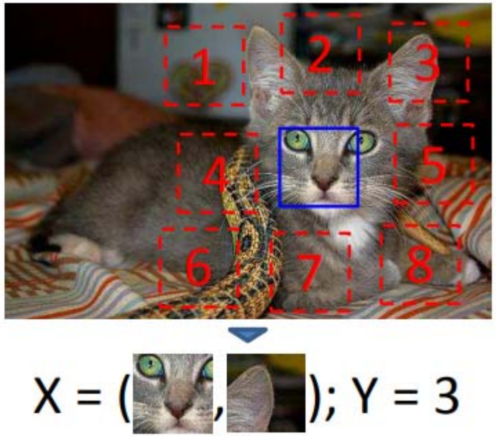
Image colorization



Super-resolution

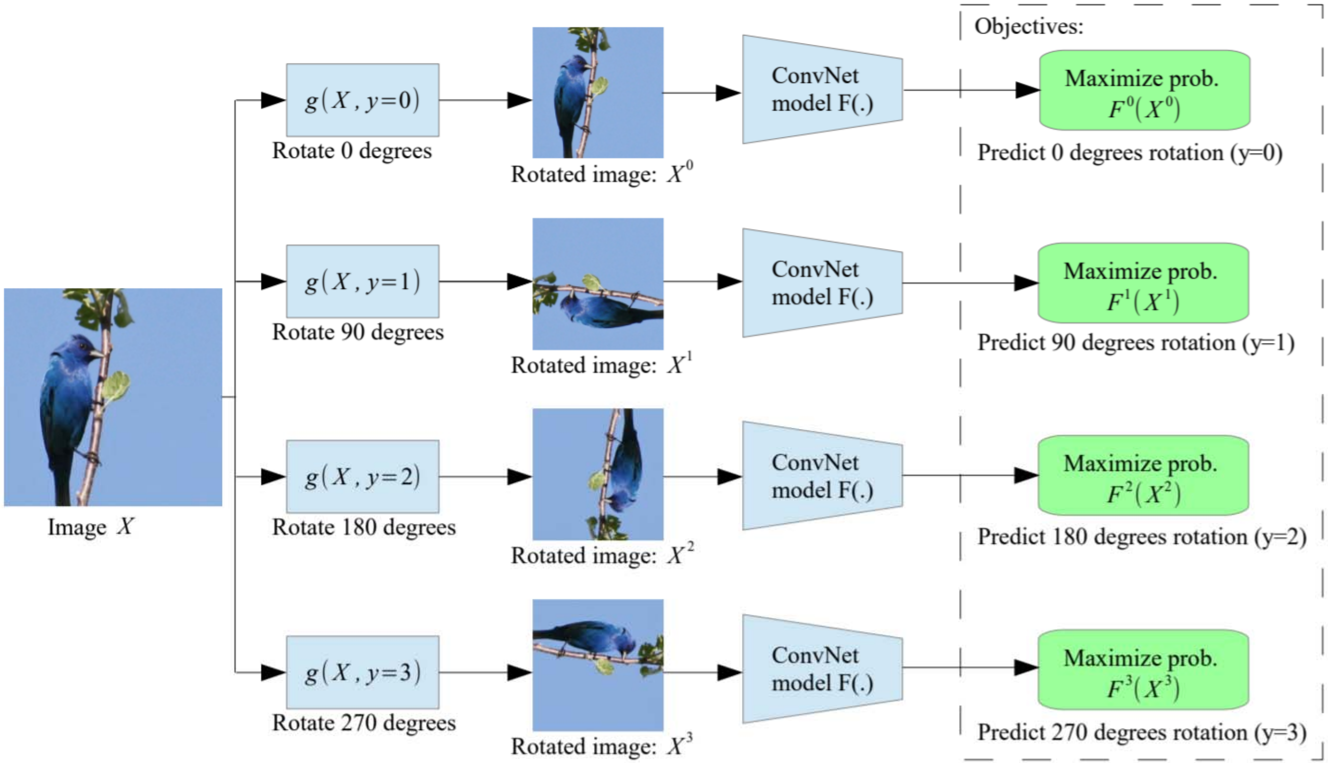
Pretext Task: Context-based Methods

- "Unsupervised learning of visual representations by solving jigsaw puzzles," ECCV, 2016.
  - Generate image patches
  - Shuffled image patches
  - Correct order of the sampled 9 patches



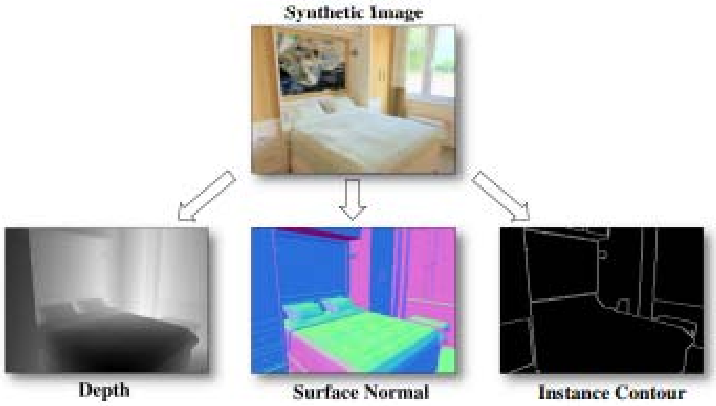
Pretext Task: Context-based Methods

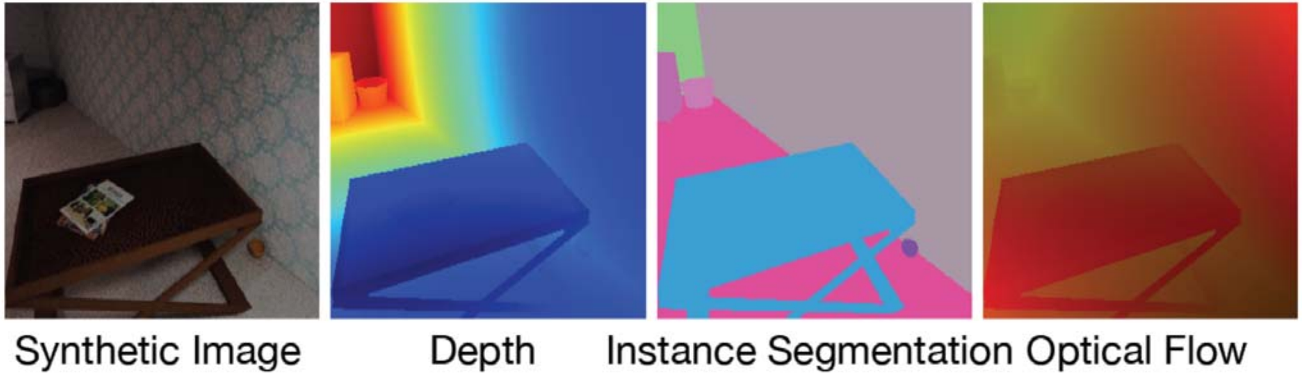
- "Unsupervised representation learning by predicting image rotation", ICLR, 2018.



Pretext Task: Free Semantic Label-based Methods

- Learning with labels generated by game engines.

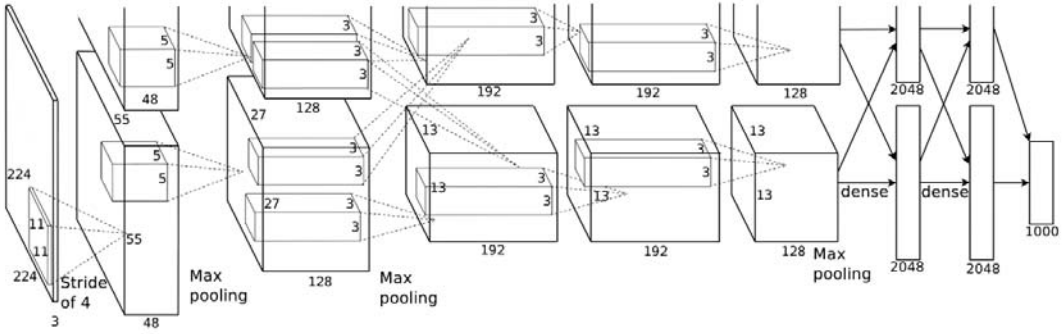




Performance Evaluation (1/2)

- Classification problems on ImageNet and Places datasets.
- The linear classifier is trained based on the nth convolutional layer of AlexNet.

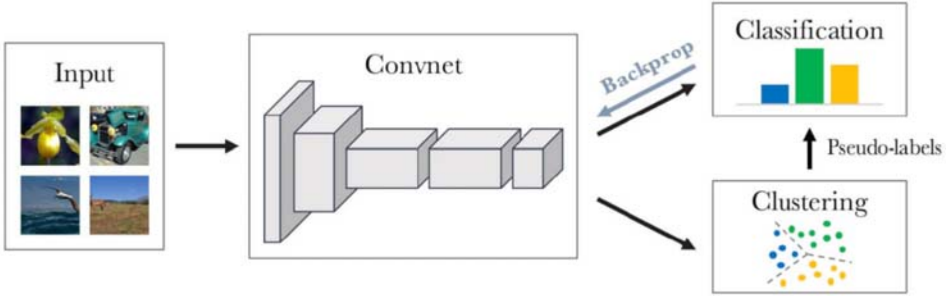
Method	Pretext Tasks	ImageNet conv1	ImageNet conv2	ImageNet conv3	ImageNet conv4	ImageNet conv5	Places conv1	Places conv2	Places conv3	Places conv4	Places conv5
Places labels [8]	—	—	—	—	—	—	22.1	35.1	40.2	43.3	44.6
ImageNet labels [8]	—	19.3	36.3	44.2	48.3	50.5	22.7	34.8	38.4	39.4	38.7
Random(Scratch) [8]	—	11.6	17.1	16.9	16.3	14.1	15.7	20.3	19.8	19.1	17.5
ColorfulColorization [18]	Generation	12.5	24.5	30.4	31.5	30.3	16.0	25.7	29.6	30.3	29.7
BiGAN [122]	Generation	17.7	24.5	31.0	29.9	28.0	21.4	26.2	27.1	26.1	24.0
SplitBrain [42]	Generation	17.7	29.3	35.4	35.2	32.8	21.3	30.7	34.0	34.1	32.5
ContextEncoder [19]	Context	14.1	20.7	21.0	19.8	15.5	18.2	23.2	23.4	21.9	18.4
ContextPrediction [41]	Context	16.2	23.3	30.2	31.7	29.6	19.7	26.7	31.9	32.7	30.9
Jigsaw [20]	Context	**18.2**	28.8	34.0	33.9	27.1	23.0	32.1	35.5	34.8	31.3
Learning2Count [130]	Context	18.0	30.6	34.3	32.5	25.7	**23.3**	**33.9**	36.3	34.7	29.6
DeepClustering [44]	**Context**	13.4	**32.3**	**41.0**	**39.6**	**38.2**	19.6	33.2	**39.2**	**39.8**	**34.7**



AlexNet
(2012)

Performance Evaluation (2/2)

Method	Pretext Tasks	Classification	Detection	Segmentation
ImageNet Labels [8]	—	79.9	56.8	48.0
Random(Scratch) [8]	—	57.0	44.5	30.1
ContextEncoder [19]	Generation	56.5	44.5	29.7
BiGAN [122]	Generation	60.1	46.9	35.2
ColorfulColorization [18]	Generation	65.9	46.9	35.6
SplitBrain [42]	Generation	67.1	46.7	36.0
RankVideo [38]	Context	63.1	47.2	35.4†
PredictNoise [46]	Context	65.3	49.4	37.1†
JigsawPuzzle [20]	Context	67.6	53.2	37.6
ContextPrediction [41]	Context	65.3	51.1	—
Learning2Count [130]	Context	67.7	51.4	36.6
DeepClustering [44]	**Context**	**73.7**	**55.4**	**45.1**
WatchingVideo [81]	Free Semantic Label	61.0	52.2	—
CrossDomain [30]	Free Semantic Label	68.0	52.6	—
AmbientSound [154]	Cross Modal	61.3	—	—
TiedToEgoMotion [95]	Cross Modal	—	41.7	—
EgoMotion [94]	Cross Modal	54.2	43.9	—

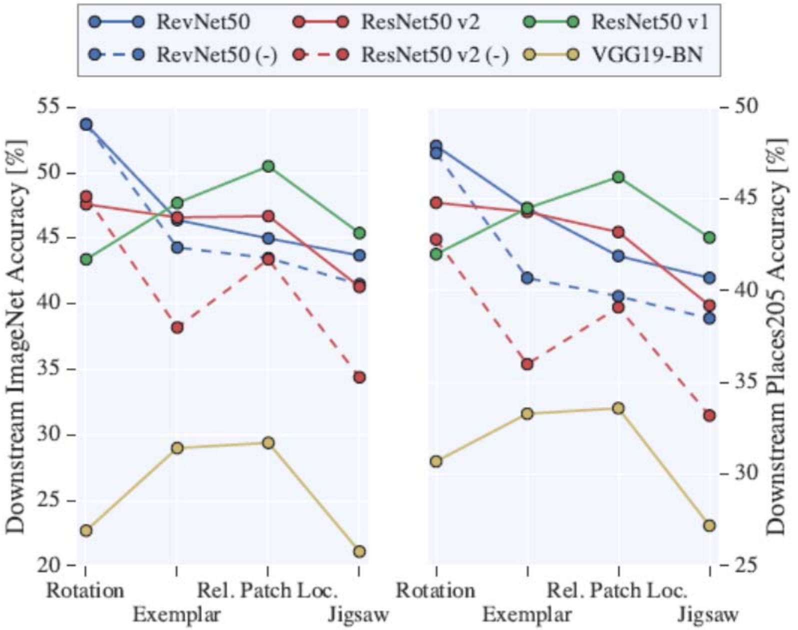


Deep clustering (2018)

Revisiting Self-supervised Learning

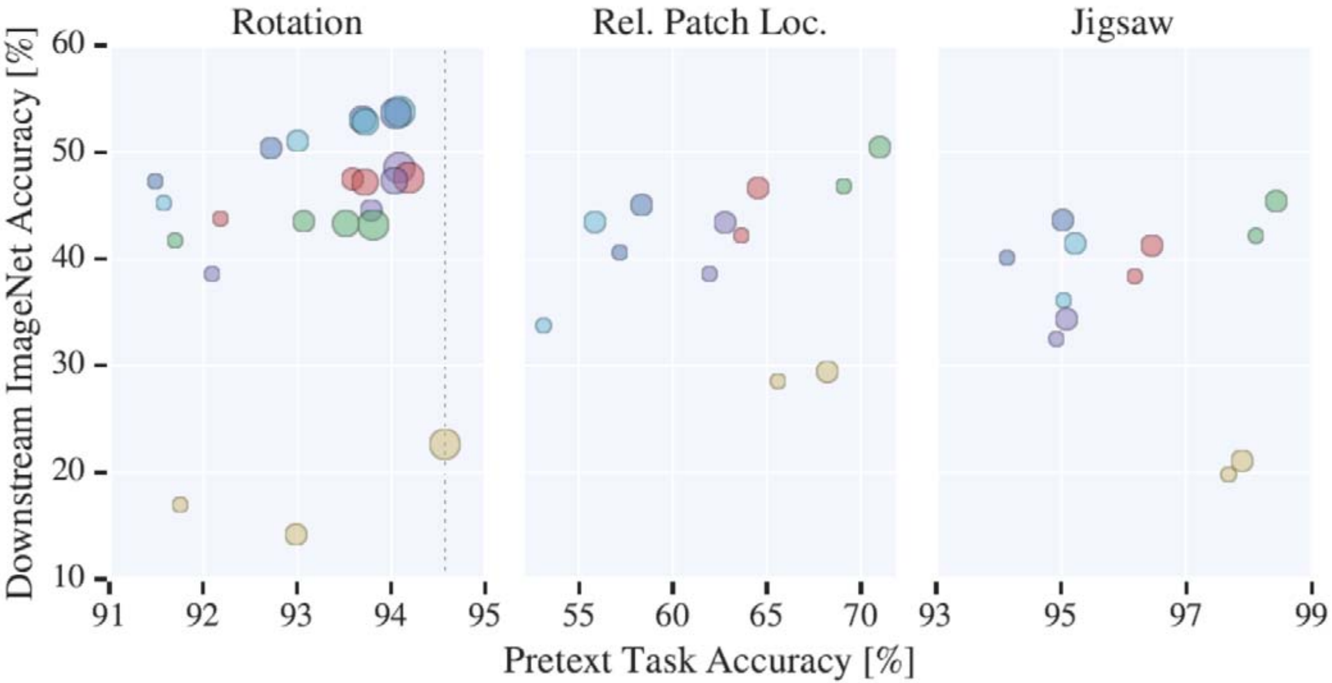
- "Revisiting self-supervised visual representation learning", CVPR, 2019.

Family		ImageNet		Places205	
		Prev.	Ours	Prev.	Ours
A	Rotation[11]	38.7	**55.4**	35.1	**48.0**
R	Exemplar[8]	31.5	46.0	-	42.7
R	Rel. Patch Loc.[8]	36.2	51.4	-	45.3
A	Jigsaw[34, 51]	34.7	44.6	35.5	42.2
V	CC+vgg-Jigsaw++[36]	37.3	-	37.5	-
A	Counting[35]	34.3	-	36.3	-
A	Split-Brain[51]	35.4	-	34.1	-
V	DeepClustering[3]	**41.0**	-	**39.8**	-
R	CPC[37]	48.7[†]	-	-	-
R	Supervised RevNet50	74.8	74.4	-	58.9
R	Supervised ResNet50 v2	76.0	75.8	-	61.6
V	Supervised VGG19	72.7	75.0	58.9	61.5



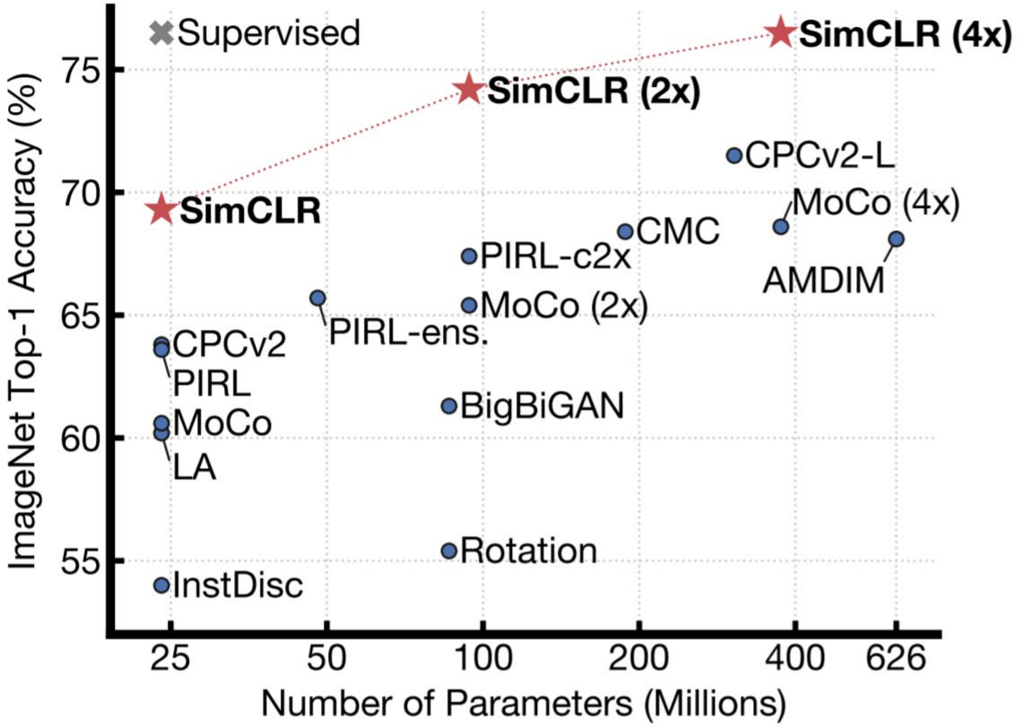
Revisiting Self-supervised Learning

- "Revisiting self-supervised visual representation learning", CVPR, 2019.
  - According to pretext accuracy, the widest VGG model is the best one for Rotation, but it performs poorly on the downstream task.



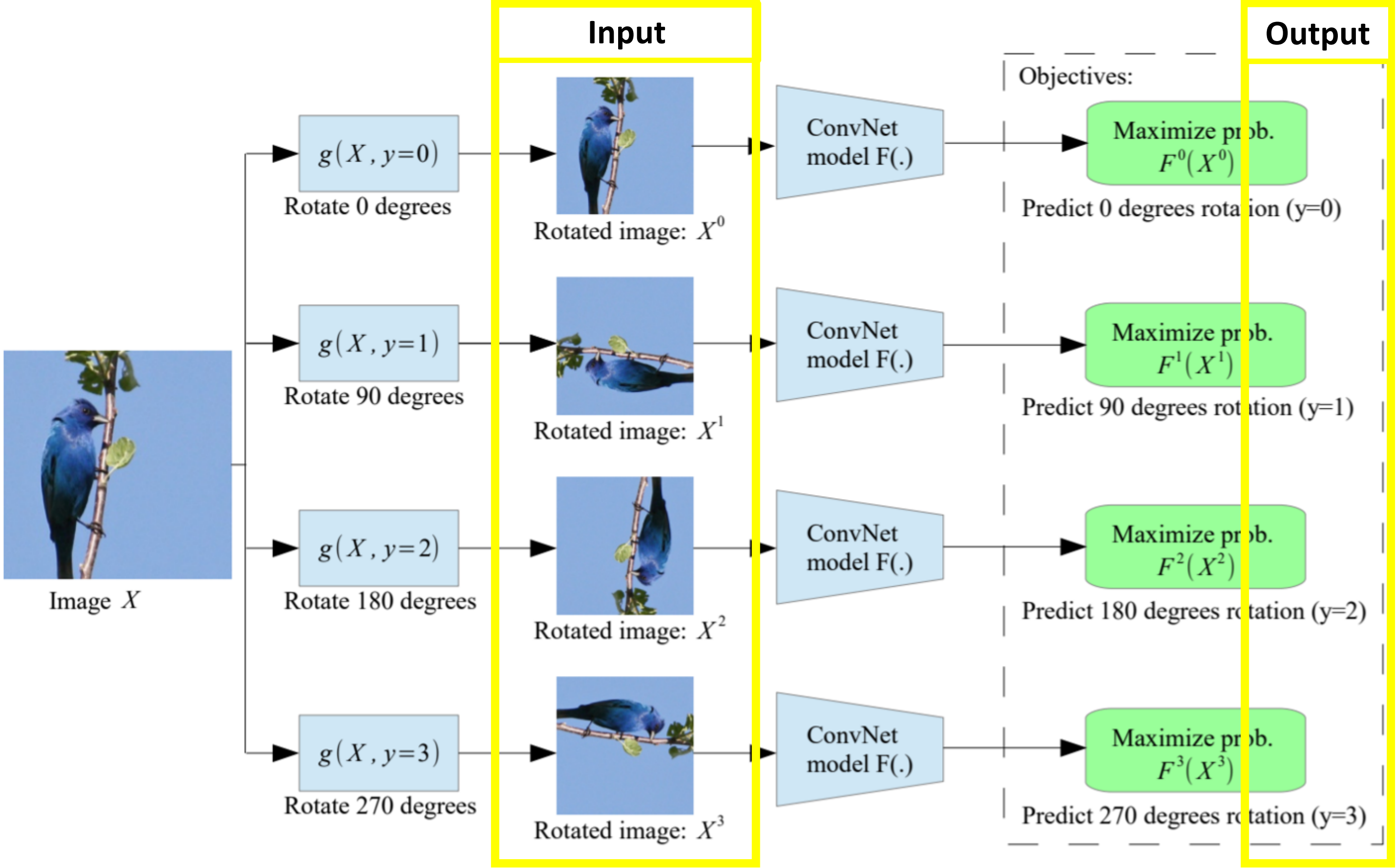
SimCLR

- "Simple framework for contrastive learning of visual representations (SimCLR)", 2020.



Demo

RotNet



Input
Output
Objectives:
Image X
$g(X, y=0)$
Rotate 0 degrees
$g(X, y=1)$
Rotate 90 degrees
$g(X, y=2)$
Rotate 180 degrees
$g(X, y=3)$
Rotate 270 degrees
Rotated image: X^0
Rotated image: X^1
Rotated image: X^2
Rotated image: X^3
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
Maximize prob. $F^0(X^0)$
Predict 0 degrees rotation (y=0)
Maximize prob. $F^1(X^1)$
Predict 90 degrees rotation (y=1)
Maximize prob. $F^2(X^2)$
Predict 180 degrees rotation (y=2)
Maximize prob. $F^3(X^3)$
Predict 270 degrees rotation (y=3)

Dataset

1. Import library

```
import tensorflow as tf
import numpy as np

from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

import matplotlib.pyplot as plt
```

2. Load MNIST dataset ¶

```
#Load MNIST dataset
(X_train, Y_train), (X_test, Y_test) = keras.datasets.mnist.load_data()

# using 1000 data for test
X_train=X_train[:1000]
Y_train=Y_train[:1000]

# using 300 data for test
X_test=X_test[:300]
Y_test=Y_test[:300]

print(X_train.shape)
```

Rotation

- Rotation can be implemented by flip and transpose

90 degrees

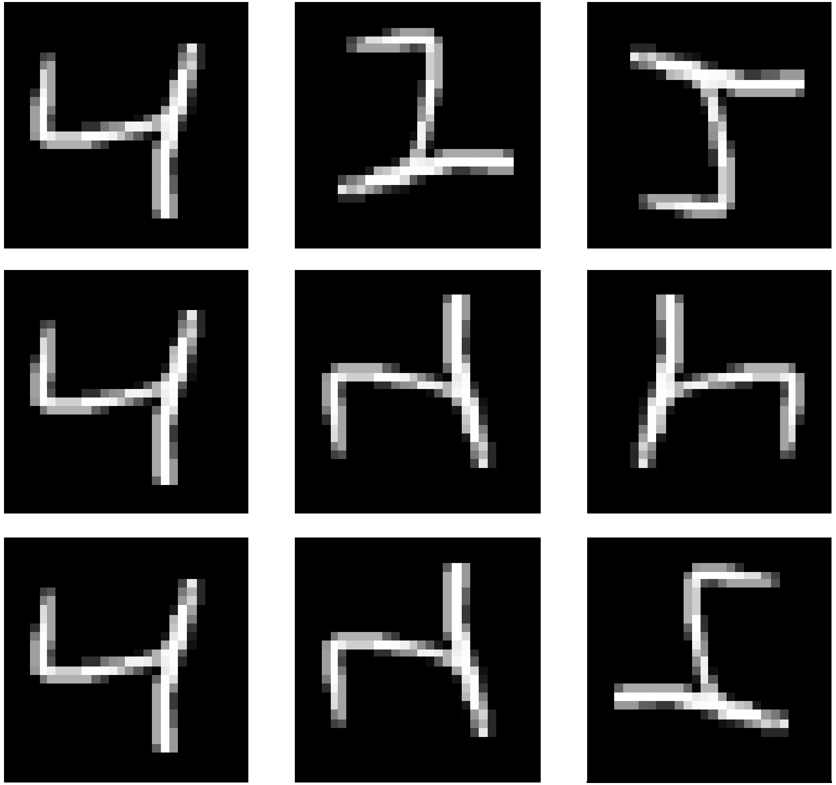
→ Transpose + Vertical flip

180 degrees

→ Vertical flip + Horizontal flip

270 degrees

→ Vertical flip + Transpose



Pretext Task Model

- ConvNet model for rotation detection
  - Model : 3 Conv layer + 1 FC layer
  - Optimizer : Stochastic Gradient Descent(SGD)

3 Pretext-task Model

```python

layer1 = Conv2D(64, kernel_size=(3, 3), strides=(2, 2), padding='same',
                activation='relu',kernel_initializer='random_normal')
layer2 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))
layer3 = Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='same',
                activation='relu' ,kernel_initializer='random_normal')
layer4 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))
layer5 = Conv2D(16, kernel_size=(3, 3), strides=(2, 2), padding='same',
                activation='relu',kernel_initializer='random_normal')

layer6 = Flatten()
layer7= Dense(4, activation='softmax',kernel_initializer='random_normal')

model = keras.Sequential([keras.Input(shape=(28,28,1)),
                          layer1, layer2, layer3,layer4,layer5,
                          layer6,layer7])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 14, 14, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 7, 7, 32)          18464
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 2, 16)          4624
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 4)                 260
=================================================================
Total params: 23,988
Trainable params: 23,988
Non-trainable params: 0
_________________________________________________________________
```

Pretext Task Training

```python
sgd = keras.optimizers.SGD(learning_rate = 0.001,momentum = 0.9)
model.compile(loss = 'categorical_crossentropy', optimizer = sgd, metrics = ['accuracy'])
hist= model.fit(X_rotate, Y_rotate, batch_size = 192, epochs = 50,verbose = 2, shuffle=False)
```

```python
# Freeze the pretext model
model.trainable=False
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 14, 14, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 7, 7, 32)          18464
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 2, 16)          4624
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 4)                 260
=================================================================
Total params: 23,988
Trainable params: 23,988
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 14, 14, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 7, 7, 32)          18464
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 2, 16)          4624
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 4)                 260
=================================================================
Total params: 23,988
Trainable params: 0
Non-trainable params: 23,988
_________________________________________________________________
```

Transfer

- Deep layers are specified only for pretext task

Model	ConvB1	ConvB2	ConvB3	ConvB4	ConvB5
RotNet with 3 conv. blocks	85.45	88.26	62.09	-	-
RotNet with 4 conv. blocks	85.07	89.06	86.21	61.73	-
RotNet with 5 conv. blocks	85.04	**89.76**	86.82	74.50	50.37

Downstream Task

- ConvNet model for digit classification
  - Model: 2 Conv layer + 1 FC layer
  - Optimizer : Stochastic Gradient Descent(SGD)
  - Since pretext task model freezes and transfers, It has only 2,890 trainable parameters

```python
layer9=Flatten()
layer10 = Dense(10,activation = 'softmax',kernel_initializer='random_normal')
# new layer to classify 10 numbers

model2 = keras.Sequential([keras.Input(shape=(28,28,1)),
                            layer1, layer2, layer3,layer4,layer9,layer10,])
model2.summary()
```

3 Pretext-task Model

```python
layer1 = Conv2D(64, kernel_size=(3, 3), strides=(2, 2), padding='same',
                activation='relu',kernel_initializer='random_normal')
layer2 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))
layer3 = Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='same',
                activation='relu' ,kernel_initializer='random_normal')
layer4 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))
layer5 = Conv2D(16, kernel_size=(3, 3), strides=(2, 2), padding='same',
                activation='relu',kernel_initializer='random_normal')

layer6 = Flatten()
layer7= Dense(4, activation='softmax',kernel_initializer='random_normal')

model  = keras.Sequential([keras.Input(shape=(28,28,1)),
                           layer1, layer2, layer3,layer4,layer5,
                           layer6,layer7])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 14, 14, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 7, 7, 32)          18464
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 288)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2890
=================================================================
Total params: 21,994
Trainable params: 2,890
Non-trainable params: 19,104
_________________________________________________________________
```

Supervised Model

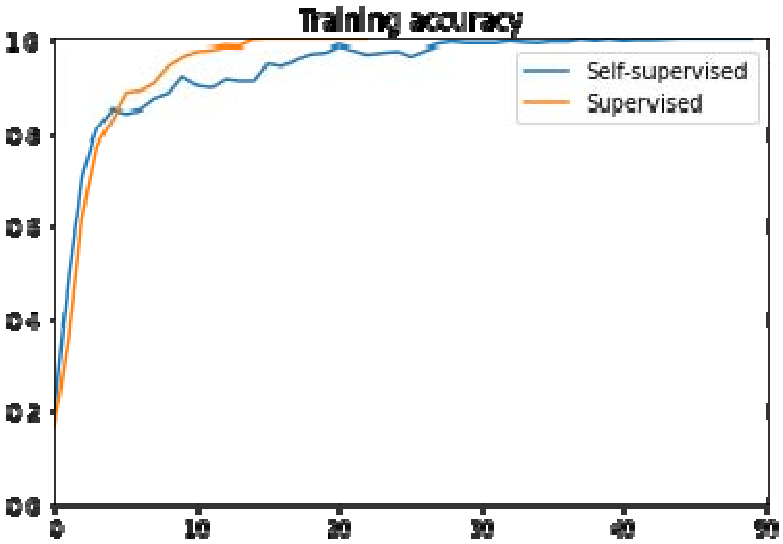
- ConvNet model for digit classification
  - It has same model architecture with downstream model
  - Optimizer : Stochastic Gradient Descent(SGD)
  - The number of total parameter is same with down stream model, but it has 0 Non-trainable parameters

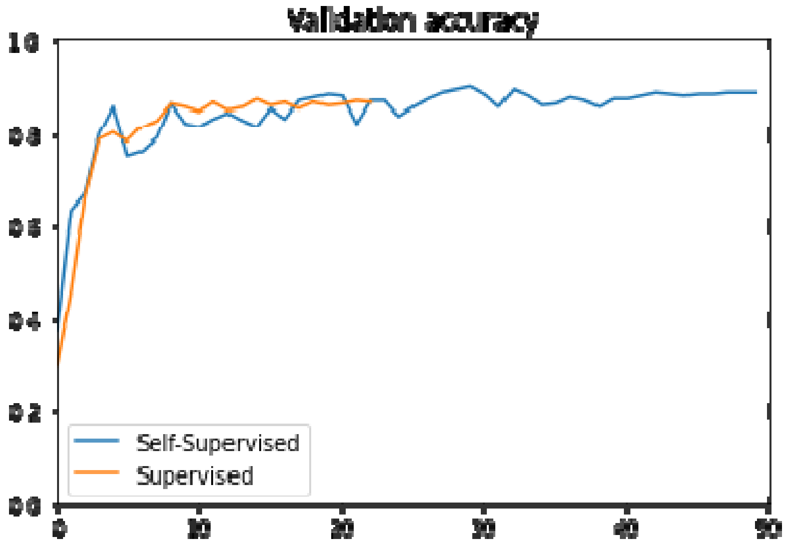
5 Supervised Model

```
#supervised model
model3 = Sequential()

model3.add(Conv2D(64, kernel_size=(3, 3), strides=(2, 2),activation='relu', padding='same',
                 kernel_initializer='random_normal',input_shape=(28,28,1)))
model3.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model3.add(Conv2D(32, kernel_size=(3, 3),strides=(1, 1), activation='relu', padding='same',
                 kernel_initializer='random_normal'))
model3.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model3.add(Flatten())
model3.add(Dense(10, activation='softmax',kernel_initializer='random_normal'))

model3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 14, 14, 64)        640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 32)          18464
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 288)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2890
=================================================================
Total params: 21,994
Trainable params: 21,994
Non-trainable params: 0
_________________________________________________________________
```

Training Result





Test Result

```
eval_self = model_down.evaluate(X_test,Y_test,batch_size = 64,steps =10,verbose = 2)
```

```
10/10 - 0s - loss: 1.8287 - accuracy: 0.8967
```

```
eval_super = model_super.evaluate(X_test,Y_test,batch_size = 64,steps =10, verbose = 2)
```

```
10/10 - 0s - loss: 0.4288 - accuracy: 0.8933
```