13주차(2/2)

기계학습 오픈 프레임워크

파이썬으로 배우는 기계학습

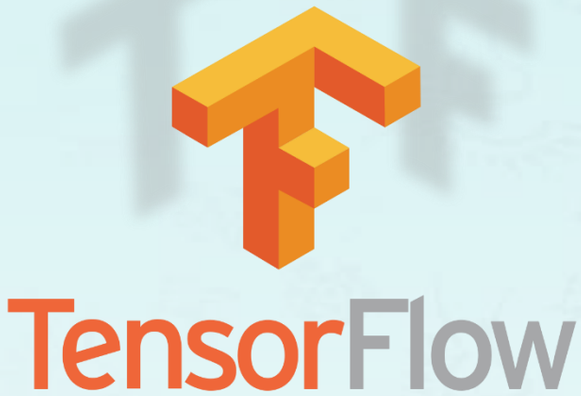
한동대학교
김영섭 교수

기계학습 오픈 프레임워크

- 학습 목표
  - 기계학습을 위한 오픈 프레임워크는 무엇이 있는지 알아본다.
  - **TensorFlow, Keras, PyTorch**가 무엇인지 이해한다.
  - **CNN**을 이용하여 **MNIST** 데이터를 **3**가지 프레임워크로 다뤄본다.

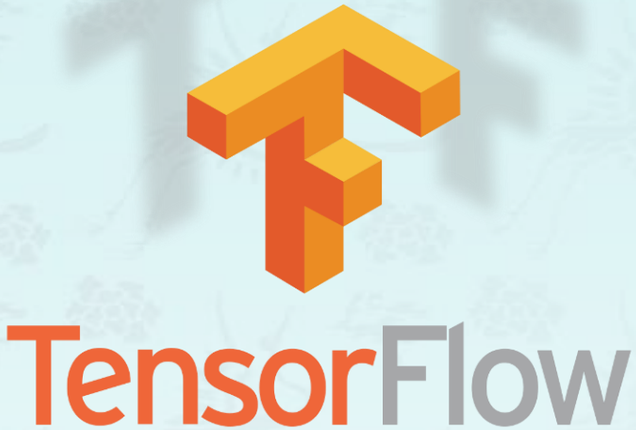
- 학습 내용
  - 기계학습을 위한 오픈 프레임워크
  - **TensorFlow, Keras, PyTorch**
  - **CNN**을 이용한 **MNIST** 데이터셋 분석

1. 오픈 프레임워크 종류: 6가지 종류



1. 오픈 프레임워크 종류: TensorFlow

- **C++, Python** 기반
- 합성곱 신경망**(CNN)**과
  순환 신경망**(RNN)** 구현
- **CPU, GPU** 환경에서 모두 동작



1. 오픈 프레임워크 종류: Keras

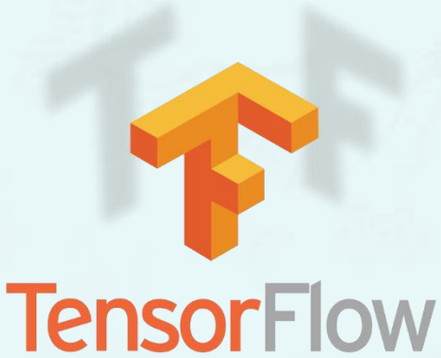
- **Python** 기반
- **TensorFlow, CNTK** 기반
- 문법이 간단하고 직관적
- **CNN**과 **RNN** 구현
- **CPU, GPU** 환경에서 모두 동작

1. 오픈 프레임워크 종류: PyTorch

- 간결하고 구현이 빠름
- 파이썬과 높은 호환성
- **Numpy** 와 유사한 **Tensor**
- 페이스북 인공지능팀 개발

2. MNIST 데이터 분석: 데이터 읽어오기 - TensorFlow

```
1  from tensorflow.examples.tutorials.mnist
2                  import input_data
3  mnist = input_data.read_data_sets(
4                  "MNIST_data/",one_hot=True)
```



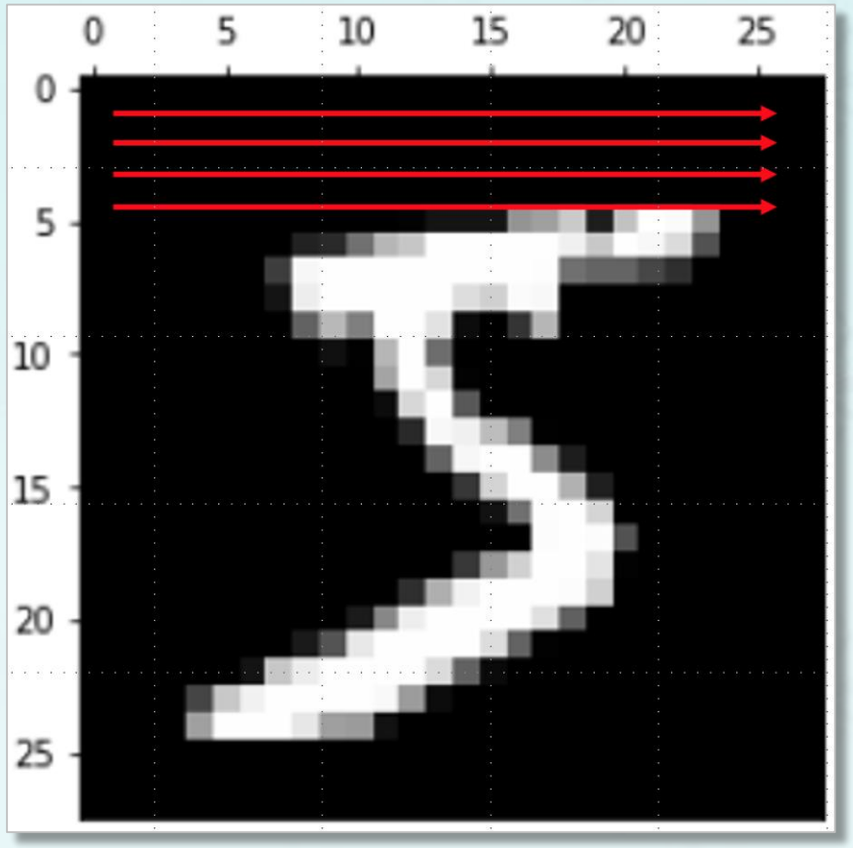
2. MNIST 데이터 분석: 데이터 읽어오기 - Keras

```
1  from keras.datasets import mnist
2  (X, y),(Xtest,ytest) = mnist.load_data()
```

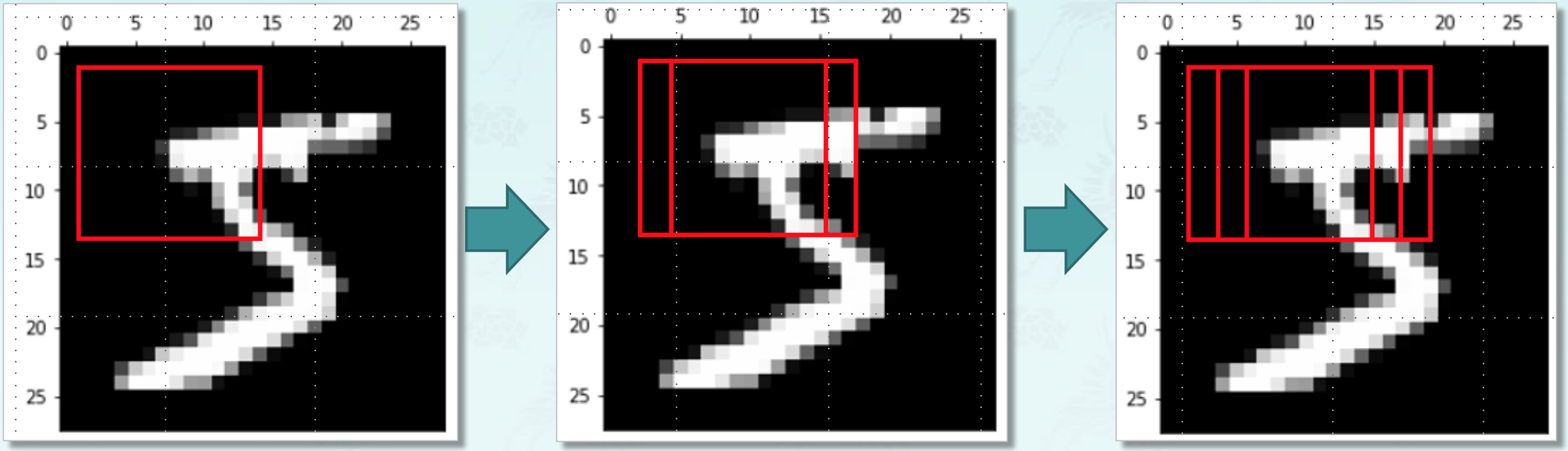
2. MNIST 데이터 분석: 데이터 읽어오기 - PyTorch

```
1  import torch
2  import torchvision.datasets as dset
3
4  root='./data'
5  train_set = dset.MNIST(root=root, train=True,
6                         download=True)
7  test_set = dset.MNIST(root=root, train=False,
8                        download=True)
```

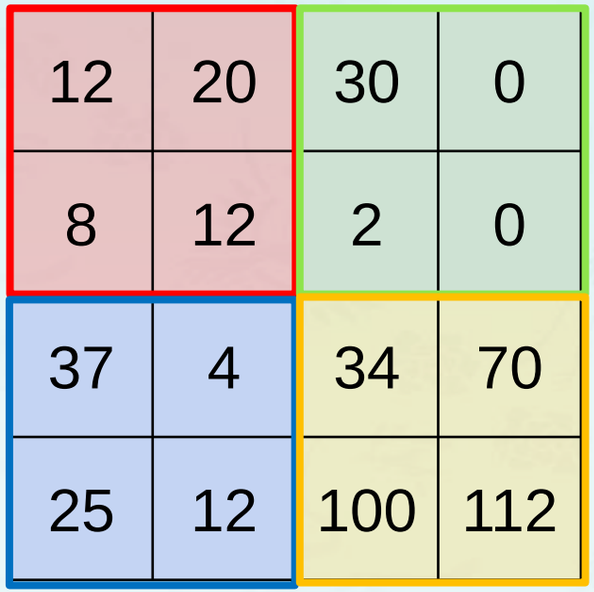
3. CNN 구현: 이미지 처리 방법



3. CNN 구현: Convolutional Layer

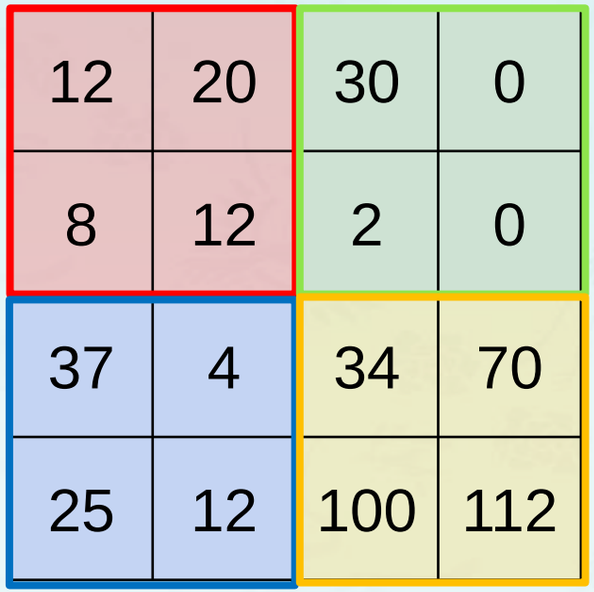


3. CNN 구현: Pooling Layer



- Max Pooling Layer
- Global Average Pooling Layer

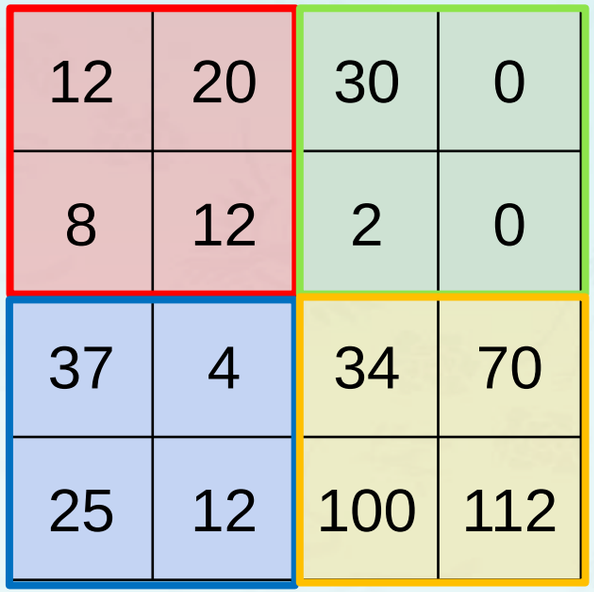
3. CNN 구현: Pooling Layer



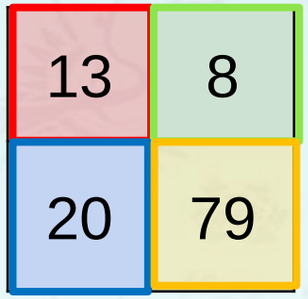
- **Max Pooling Layer**

20	30
37	112

3. CNN 구현: Pooling Layer



- **Global Average Pooling Layer**



3. CNN 구현: 신경망 구축 - TensorFlow

```python
import tensorflow as tf
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
dropout_ratio = tf.placeholder(tf.float32)

x_train = tf.reshape(x, [-1, 28, 28, 1])

W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

h_conv1 = tf.nn.relu
          (conv2d(x_train, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu
          (conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu
        (tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
h_fc1_drop = tf.nn.dropout(h_fc1, dropout_ratio)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_hat = tf.nn.softmax
       (tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

3. CNN 구현: 신경망 학습 - TensorFlow

```python
import tensorflow as tf
x = tf.placeholder(tf.float32, [None, 784])
```

```python
CE = tf.reduce_mean(
        -tf.reduce_sum(y * tf.log(y_hat),
        reduction_indices=[1])
        )
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(CE)
```

경사하강법

학습률

교차 엔트로피

```python
b_conv1 = bias_variable([32])

h_conv1 = tf.nn.relu(conv2d(x_train, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
h_fc1_drop = tf.nn.dropout(h_fc1, dropout_ratio)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_hat = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

3. CNN 구현: 신경망 학습 - TensorFlow

```
import tensorflow as tf
x = tf.placeholder(tf.float32, [None, 784])
```

```
CE = tf.reduce_mean(
```

신경망 출력: 예측값

클래스 레이블

```
correct_prediction = tf.equal(tf.argmax(y_hat,1), tf.argmax(y,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
```

```
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
h_fc1_drop = tf.nn.dropout(h_fc1, dropout_ratio)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_hat = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

3. CNN 구현: 신경망 학습 - TensorFlow

```
for i in range(10):
    batch = mnist.train.next_batch(32)
    if i%100 == 0:
        train_accuracy = accuracy.eval(session=sess, feed_dict={
            x:batch[0], y: batch[1], dropout_ratio: 1.0
        })
        print("step %d, training accuracy %g"%(i, train_accuracy))
    train_step.run(feed_dict=
                   {x: batch[0], y: batch[1], dropout_ratio: 0.2})

acc = sess.run(accuracy, feed_dict=
               {x:mnist.test.images, y:mnist.test.labels,
                dropout_ratio: 1.0})
print("Test accuracy: {}".format(acc))
```

```
h_fc1_drop = tf.nn.dropout(h_fc1, dropout_ratio)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_hat = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

3. CNN 구현: 신경망 학습 - TensorFlow

```
for i in range(10):
        batch = mnist.train.next_batch(32)
        if i%100 == 0:
            train_accuracy = accuracy.eval(session=sess, feed_dict={
                x:batch[0], y: batch[1], dropout_ratio: 1.0
            })
            print("step %d, training accuracy %g"%(i, train_accuracy))
        train_step.run(feed_dict=
                       {x: batch[0], y: batch[1], dropout_ratio: 0.2})

    acc = sess.run(accuracy, feed_dict=
                   {x:mnist.test.images, y:mnist.test.labels,
                    dropout_ratio: 1.0})
    print("Test accuracy: {}".format(acc))
```

```
h_fc1_drop = tf.nn.dropout(h_fc1, dropout_ratio)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_hat = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

3. CNN 구현: 학습 결과 - TensorFlow

```python
import tensorflow as tf
x = tf.placeholder(tf.float32, [None, 784])

b_c
h_c
h_p

W_conv2
b_conv2

h_conv2
h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
h_fc1_drop = tf.nn.dropout(h_fc1, dropout_ratio)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_hat = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

```python
CE = tf.reduce_mean(
```

```python
correct_prediction = tf.equal(tf.argmax(y_hat,1), tf.argmax(y,1))
```

```python
for i in range(10):
    batch = mnist.train.next_batch(50)
    if i%100 == 0:
        train_accuracy = accuracy.eval(session=sess, feed_dict={
            x:batch[0], y: batch[1], dropout_ratio: 1.0
        })
        print("step %d,
    train_step.run(feed_
                                                    2})

acc = sess.run(accuracy, feed_dict={x:mnist.test.images, y:mnist.test.labels, dropout_ratio: 1.0})
print("Test accuracy: {}".format(acc))
```

```
Test accuracy: 94%
```

3. CNN 구현: 신경망 구축 - Keras

```
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import GlobalAveragePooling2D
from keras.layers import Dropout, Flatten, Dense
from keras.models import Sequential

# define the model
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2,
                 padding='valid', activation='relu',
                 input_shape=(28, 28, 1)))
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=2,
                 padding='valid',activation='relu'))
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=2,
                 padding='valid',activation='relu'))
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))

# summarize the model
model.summary()
```

3. CNN 구현: 신경망 구축 - Keras

```
from keras.layers import Conv2D, MaxPooling2D
```

```
model.compile(
    loss='categorical_crossentropy',
    optimizer='rmsprop',
    metrics=['accuracy']
)
```

```
                input_shape=(28, 28, 1)))
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=2,
                 padding='valid',activation='relu'))
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=2,
                 padding='valid',activation='relu'))
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))

# summarize the model
model.summary()
```

3. CNN 구현: 신경망 구축 - Keras

```
from keras.layers import Conv2D, MaxPooling2D
```

```
model.compile(
```

```
from keras.callbacks import ModelCheckpoint

checkpointer = ModelCheckpoint(
            filepath='mnist.model.best.hdf5',
            verbose=1,
            save_best_only=True
)
model.fit(X, y,
          batch_size=128, epochs=10,
          validation_split=0.2,
          callbacks=[checkpointer],
          verbose=1, shuffle=True
)
```

```
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))

# summarize the model
model.summary()
```

3. CNN 구현: 신경망 학습 - Keras

```
from keras.layers import Conv2D, MaxPooling2D
```

```
model.compile(
```

```
from keras.callbacks import ModelCheckpoint
```

```
model.load_weights('mnist.model.best.hdf5')

loss_and_metrics = model.evaluate(Xtest, ytest)
accuracy = 100 * loss_and_metrics[1]

print("Test accuracy: {}%".format(accuracy))
```

```
            callbacks=[checkpointer],
            verbose=1, shuffle=True
)
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))

# summarize the model
model.summary()
```

3. CNN 구현: 학습 결과 - Keras

```
from keras.layers import Conv2D, MaxPooling2D
```

```
model.compile(
```

```
from keras.callbacks import ModelCheckpoint
```

```
model.load_weights('mnist.model.best.hdf5')

loss_and_metrics = model.evaluate(Xtest, ytest)
accuracy = 100 * loss_and_metrics[1]

print("Test accuracy
```

```
Test accuracy: 98.79%
```

```
            batch_size=
            validation_split=0.2,
            callbacks=[checkpointer],
            verbose=1, shuffle=True
)
model.add(Dropout(0.2))
model.add(MaxPooling2D(pool_size=2))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))

# summarize the model
model.summary()
```

3. CNN 구현: 신경망 구축 - PyTorch

```python
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 16, kernel_size=2)
        self.conv1_drop = nn.Dropout2d(0.2)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=2)
        self.conv2_drop = nn.Dropout2d(0.2)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=2)
        self.conv3_drop = nn.Dropout2d(0.2)

        self.fc1 = nn.Linear(256, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1_drop(self.conv1(x)), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv3(x)), 2))
        x = x.view(-1, 256)
        x = self.fc1(x)

        return F.log_softmax(x)
```

3. CNN 구현: 신경망 구축 - PyTorch

```
import torch.nn as nn
```

```
n_epochs = 10
learning_rate = 0.01
random_seed = 1
log_interval = 10

network = Net()
optimizer = optim.RMSprop(network.parameters(), lr=learning_rate)
```

```
        self.fc1 = nn.Linear(256, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1_drop(self.conv1(x)), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv3(x)), 2))
        x = x.view(-1, 256)
        x = self.fc1(x)

        return F.log_softmax(x)
```

3. CNN 구현: 신경망 구축 - PyTorch

```python
import torch.nn as nn
```

```python
n_epochs = 10
learning_rate = 0.01
```

```python
def train(epoch):
    network.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()
        output = network(data)
        loss = F.cross_entropy(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
            epoch, batch_idx * len(data), len(train_loader.dataset),
            100. * batch_idx / len(train_loader), loss.item()))
```

```python
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv3(x)), 2))
        x = x.view(-1, 256)
        x = self.fc1(x)

        return F.log_softmax(x)
```

PyTorch

3. CNN 구현: 신경망 구축 - PyTorch

```python
import torch.nn as nn
```

```python
n_epochs = 10
```

```python
def train(epoch):
    network.train()
```

```python
def test():
    network.eval()
    test_loss = 0
    correct = 0
    for data, target in test_loader:
        output = network(data)
        test_loss += F.cross_entropy(output, target, size_average=False).item()
        pred = output.data.max(1, keepdim=True)[1]
        correct += pred.eq(target.data.view_as(pred)).sum()
        test_loss /= len(test_loader.dataset)
    print('Test Accuracy: {}%\n'.format(
        100. * correct / len(test_loader.dataset)))
```

```python
    x = self.fc1(x)

    return F.log_softmax(x)
```

3. CNN 구현: 학습 결과 - PyTorch

```
import torch.nn as nn
```

```
n_epochs = 10
```

```
def train(epoch):
```

```
def test():
```

```
for epoch in range(1, n_epochs + 1):
    train(epoch)
    test()
```

```
                                                    e=False).item()
```

Test Accuracy: 97%

```
            test_loss /= len(test_loader.dataset)
        print('Test Accuracy: {}%\n'.format(
            100. * correct / len(test_loader.dataset)))
```

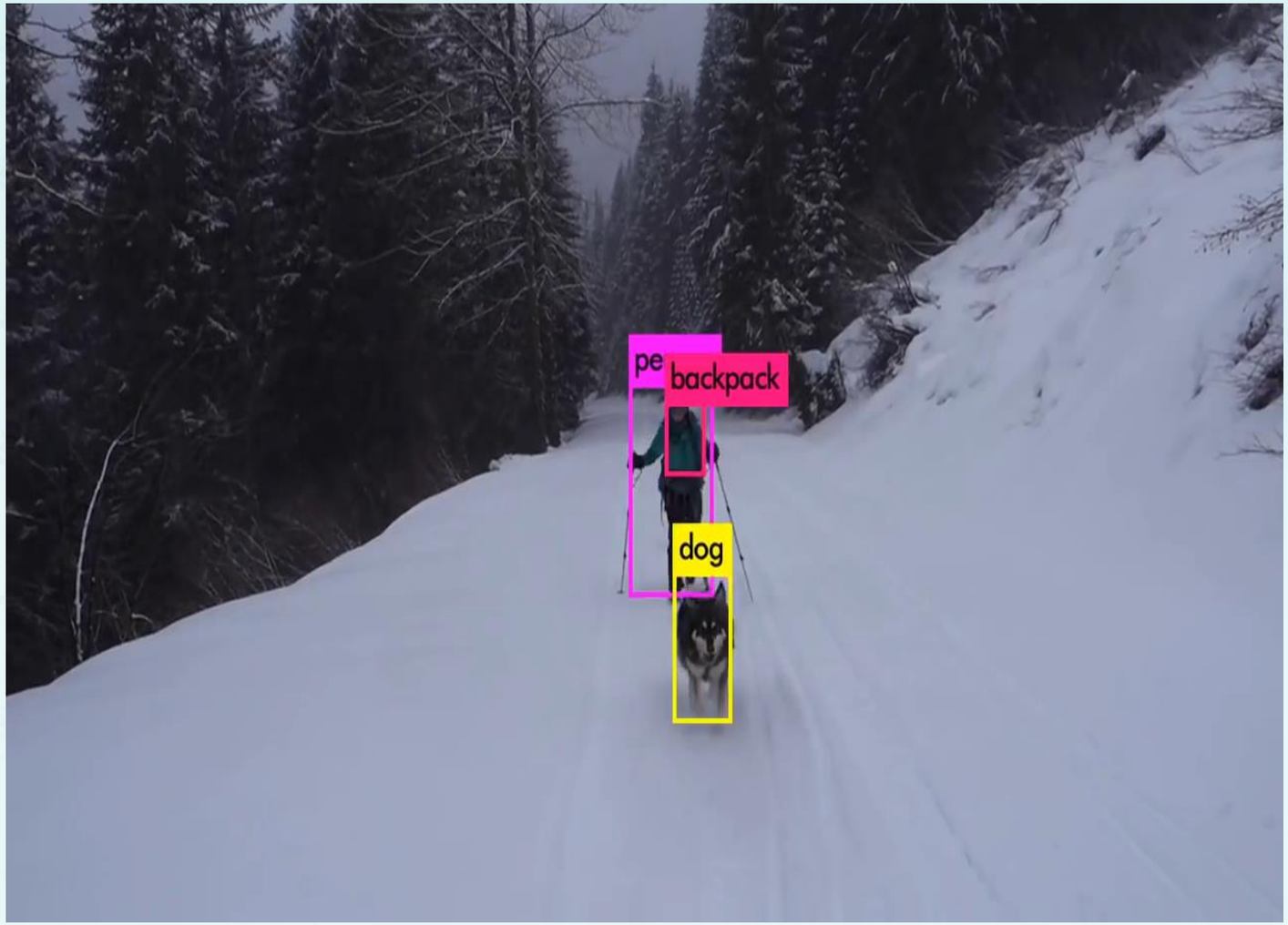
```
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv3(x)), 2))
        x = x.view(-1, 256)
        x = self.fc1(x)

        return F.log_softmax(x)
```

PyTorch

4. 기계학습 모델:YOLO

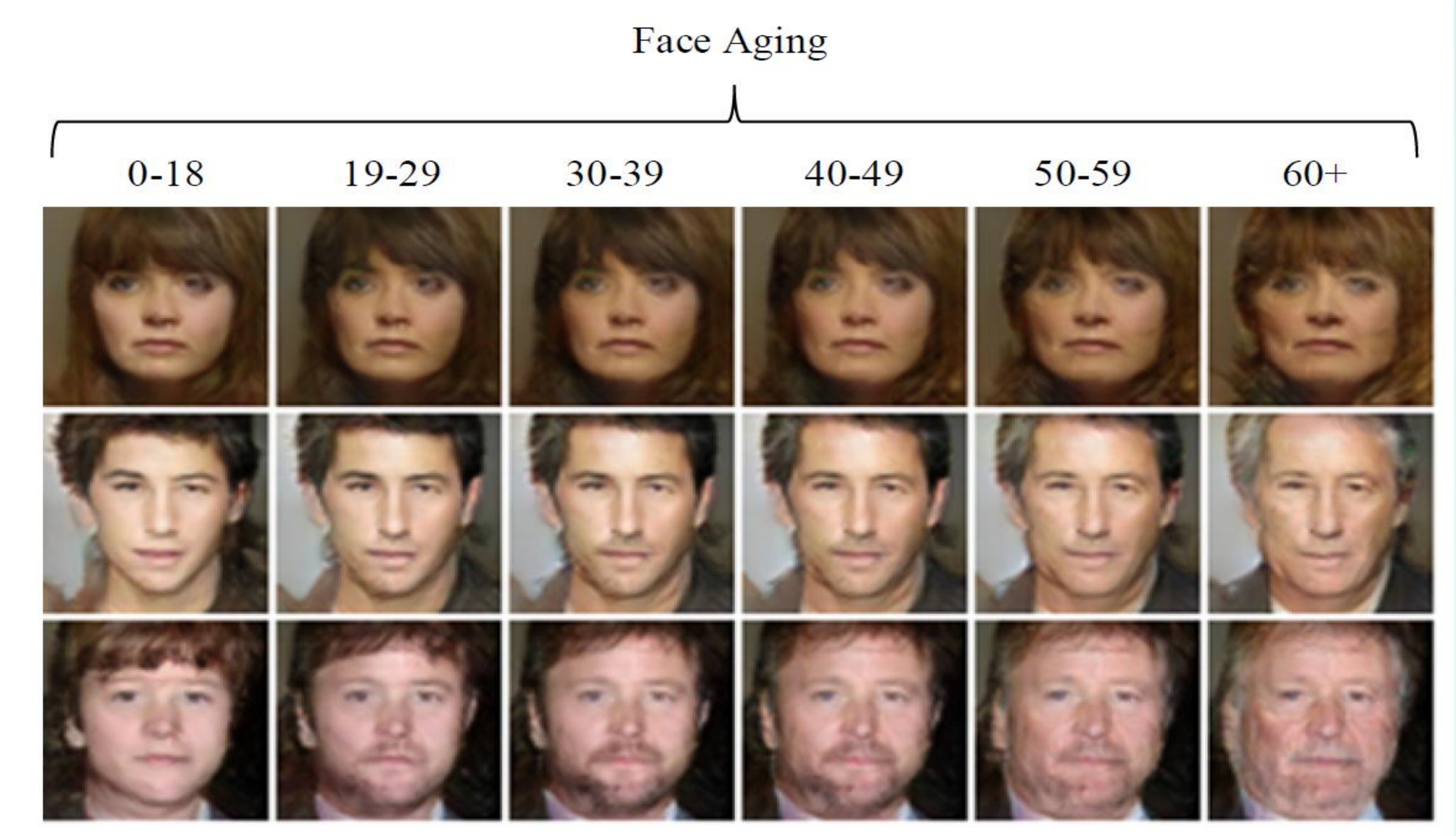
- 실시간 물체인식에 사용
- **You Only Look Once**
- **1**초에 **45**장 이미지 분석 **(45 fps)**



4. 기계학습 모델: GAN – 생성적 적대 신경망

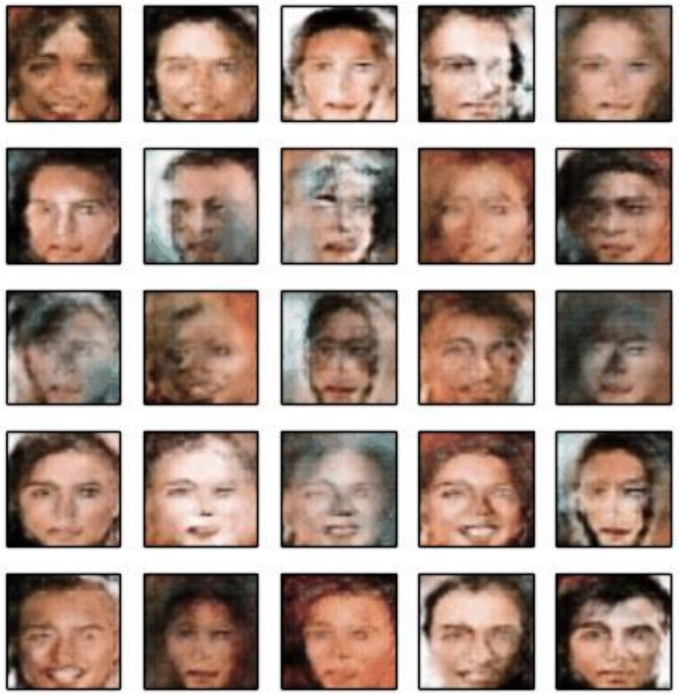
- **Generative Adversarial Networks**
- 이미지 직접 생성
- 예 : 현재 얼굴 사진 학습하여 미래
  얼굴 이미지 생성

4. 기계학습 모델: GAN – 생성적 적대 신경망

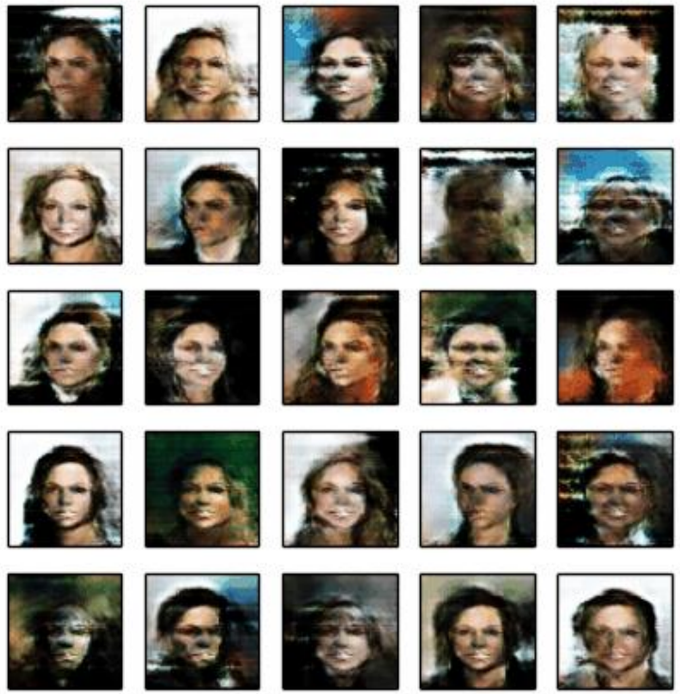


4. 기계학습 모델: GAN – 생성적 적대 신경망

- **DCGAN (Deep Convolutional GAN)**



Epoch 1



Epoch 1

Epoch 1

기계학습 오픈 프레임워크

- 학습 정리
  - 기계학습을 위한 오픈 프레임워크
  - **TensorFlow, Keras, PyTorch**
  - **CNN**을 이용한 **MNIST** 데이터셋 분석
  - 기계학습 모델 **GAN**