6.3 Fashion MNIST 데이터셋에 적용하기

파일

- **ch6_3_Fashion_MNIST_with_CNN.ipynb**

데이터 로드와 정규화

```
import tensorflow as tf
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_X, train_Y), (test_X, test_Y) = fashion_mnist.load_data()

train_X = train_X / 255.0
test_X = test_X / 255.0
```

Conv2D 레이어를 위한 모양 변형

- **Conv2D 레이어로 컨볼루션 연산을 수행**
  - 이미지는 보통 채널을 가짐
    - **컬러 이미지는 RGB의 3채널, 흑백 이미지는 1채널**
  - Conv2D 레이어는 채널을 가진 형태의 데이터를 받도록 기본적으로 설정
    - **채널을 갖도록 데이터의 Shape를 변형**
    - **Fashion MNIST 데이터를 구성하는 흑백 이미지는 1개의 채널을 갖음**
      - **reshape() 함수를 사용해 데이터의 가장 뒤쪽에 채널 차원을 추가**

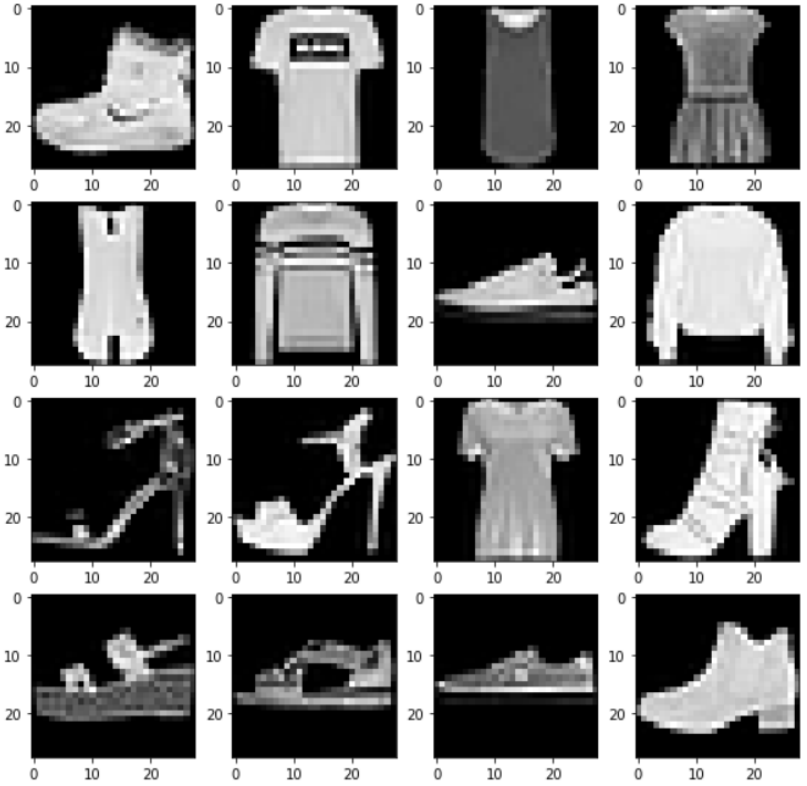
```
# reshape 이전
print(train_X.shape, test_X.shape)

train_X = train_X.reshape(-1, 28, 28, 1)
test_X = test_X.reshape(-1, 28, 28, 1)

# reshape 이후
print(train_X.shape, test_X.shape)
```

```
(60000, 28, 28) (10000, 28, 28)
(60000, 28, 28, 1) (10000, 28, 28, 1)
```

데이터 확인 시각화

```
import matplotlib.pyplot as plt

# 전체 그래프의 크기를 width = 10, height = 10으로 지정합니다.
plt.figure(figsize=(10, 10))
for c in range(16):
  # 4행 4열로 지정한 그리드에서 c+1번째의 칸에 그래프를 그립니다. 1~16번째 칸을 채우게 됩니다.
  plt.subplot(4,4,c+1)
  plt.imshow(train_X[c].reshape(28,28), cmap='gray')

plt.show()

# 훈련 데이터이 1~16번째 까지의 라벨 프린트합니다.
print(train_Y[:16])
```

라벨	범주
0	티셔츠/상의
1	바지
2	스웨터
3	드레스
4	코트
5	샌들
6	셔츠
7	운동화
8	가방
9	부츠



[9 0 0 3 0 2 7 2 5 5 0 9 5 5 7 9]

컨볼루션 신경망 모델에서의 패러미터 수

- **Conv2D의 인수**
  - kernel_size
    - **필터 행렬의 크기, 수용 영역(receptive filed)**
      - **(높이, 너비)**
  - filters
    - **필터의 개수**

```
model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(input_shape=(28,28,1),
                         kernel_size=(3,3), filters=16),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=32),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=64),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(units=128, activation='relu'),
  tf.keras.layers.Dense(units=10, activation='softmax')
])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 16)        160
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        4640
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 22, 22, 64)        18496
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3965056
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 3,989,642
Trainable params: 3,989,642
Non-trainable params: 0
_________________________________________________________________
```

(None, 28, 28, 1)

커널 수(K) * 커널 사이즈(F)^2 * (채널:색상 수(D)) + 커널 수(K)
16 * 3*3 * 1 + 16 = 160

커널 수(K) * 커널 사이즈(F)^2 * (채널:색상 수(D)) + 커널 수(K)
32 * 3*3 * 16 + 32 = 4,640

커널 수(K) * 커널 사이즈(F)^2 * (채널:색상 수(D)) + 커널 수(K)
64 * 3*3 * 32 + 64 = 18,496

(이전 출력 노드 수 + 1) * 노드 수
(30976 + 1) * 128

(128 + 1) * 10

컨볼루션 신경망 모델 정의

- **컨볼루션 신경망 모델**
  - 풀링 레이어 또는 드롭아웃 없이 정의된 모델

```
model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(input_shape=(28,28,1),
                         kernel_size=(3,3), filters=16),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=32),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=64),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(units=128, activation='relu'),
  tf.keras.layers.Dense(units=10, activation='softmax')

])

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

GPU 사용 설정

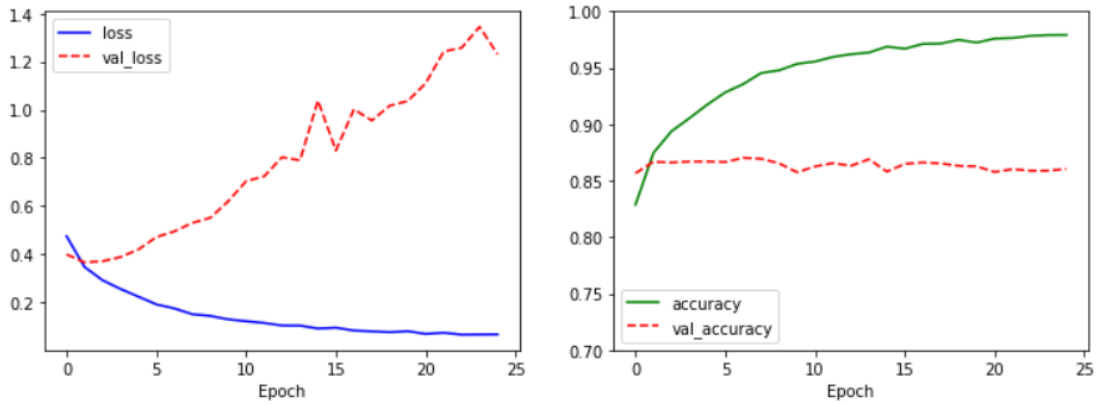
- **구글 코랩에서는 무료로 GPU를 사용**
  - [메뉴]-[런타임]-[런타임 유형 변경]-[하드웨어 가속기]-[GPU]로 지정
- **GPU 확인**
  - 구글 코랩에서 지원하는 GPU의 성능
    - **Tesla K80나 Tesla T4를 사용**

```
[5]   1 !nvidia-smi
```

```
Fri Aug  7 00:41:15 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.57       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   45C    P0    27W /  70W |    293MiB / 15079MiB |      0%      Default |
|                               |                      |                 ERR! |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

컨볼루션 신경망 모델 학습

```
history = model.fit(train_X, train_Y, epochs=25, validation_split=0.25)


import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1,2,1)
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'r--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.7, 1)
plt.legend()

plt.show()

model.evaluate(test_X, test_Y, verbose=0)
```




```
[1.3350350856781006, 0.855400025844574]
```

풀링 레이어, 드롭아웃 레이어 추가

- **과적합을 줄이는 데 기여**
  - MaxPool2D(strides=(2,2)), Dropout(rate=0.3)
    - **strides: 필터가 계산 과정에서 한 스텝마다 이동하는 크기**
      - **기본값은 (1,1)이고, (2,2) 등으로 설정할 경우 한 칸씩 건너뛰면서 계산**
    - **rate: 제외할 뉴런의 비율**

```
model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(input_shape=(28,28,1), kernel_size=(3,3), filters=32),
  tf.keras.layers.MaxPool2D(strides=(2,2)),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=64),
  tf.keras.layers.MaxPool2D(strides=(2,2)),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=128),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(units=128, activation='relu'),
  tf.keras.layers.Dropout(rate=0.3),
  tf.keras.layers.Dense(units=10, activation='softmax')
])
```

풀링 레이어 추가로 패러미터 수 감소

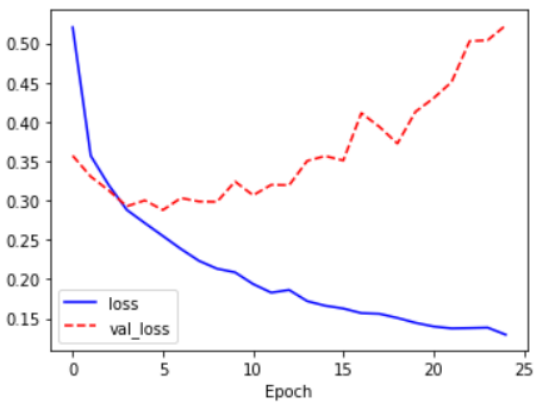
- **Params의 개수**
  - 기존 3,989,642에서 241,546으로 대폭 감소
    - **Flatten에 들어오는 Params의 개수가 기존 (None, 30976)에 비해 (None, 1152)**

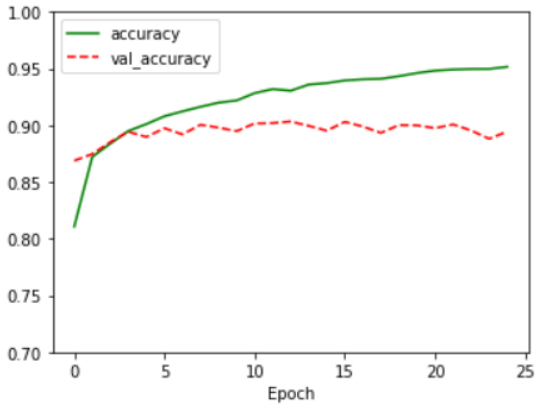
```
model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(input_shape=(28,28,1), kernel_size=(3,3), filters=32),
  tf.keras.layers.MaxPool2D(strides=(2,2)),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=64),
  tf.keras.layers.MaxPool2D(strides=(2,2)),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=128),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(units=128, activation='relu'),
  tf.keras.layers.Dropout(rate=0.3),
  tf.keras.layers.Dense(units=10, activation='softmax')

])

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 3, 3, 128)         73856
_________________________________________________________________
flatten_3 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               147584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
=================================================================
Total params: 241,546
Trainable params: 241,546
Non-trainable params: 0
_________________________________________________________________
```

훈련과 시각화

```
history = model.fit(train_X, train_Y, epochs=25, validation_split=0.25)

import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1,2,1)
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'r--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.7, 1)
plt.legend()

plt.show()

model.evaluate(test_X, test_Y, verbose=0)
```






```
[0.5307855606079102, 0.8906999826431274]
```

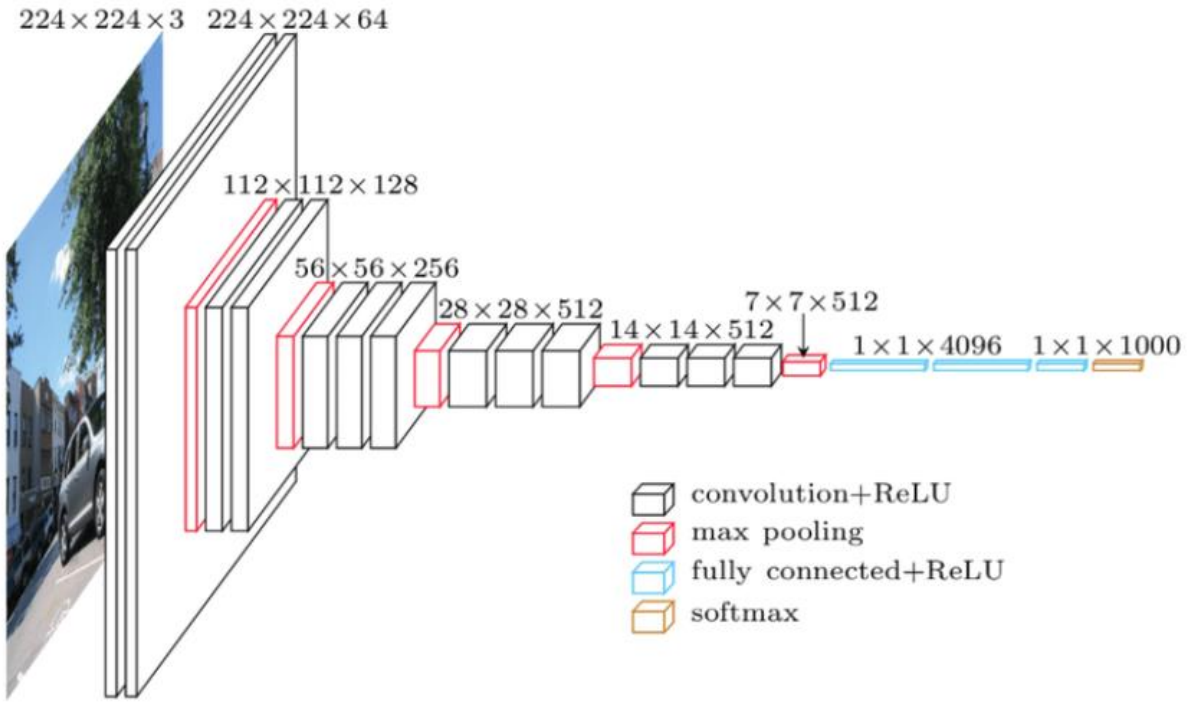
CNN(합성곱) 성능 높이기

파일

- ch6_4_improve_performance.ipynb

많은 층 쌓기

- **VGGNet 컨볼루션 신경망 응용**



VGGNet architecture [19]

데이터 불러오기 및 정규화

```
import tensorflow as tf
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_X, train_Y), (test_X, test_Y) = fashion_mnist.load_data()

train_X = train_X / 255.0
test_X = test_X / 255.0

# reshape 이전
print(train_X.shape, test_X.shape)

train_X = train_X.reshape(-1, 28, 28, 1)
test_X = test_X.reshape(-1, 28, 28, 1)

# reshape 이후
print(train_X.shape, test_X.shape)
```

```
(60000, 28, 28) (10000, 28, 28)
(60000, 28, 28, 1) (10000, 28, 28, 1)
```

CNN 모델 정의

```
import tensorflow as tf

model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(input_shape=(28,28,1), kernel_size=(3,3), filters=32,
                         padding='same', activation='relu'),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=64, padding='same', activation='relu'),
  tf.keras.layers.MaxPool2D(pool_size=(2,2)),
  tf.keras.layers.Dropout(rate=0.5),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=128, padding='same', activation='relu')
,
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=256, padding='valid', activation='relu'
),
  tf.keras.layers.MaxPool2D(pool_size=(2,2)),
  tf.keras.layers.Dropout(rate=0.5),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(units=512, activation='relu'),
  tf.keras.layers.Dropout(rate=0.5),
  tf.keras.layers.Dense(units=256, activation='relu'),
  tf.keras.layers.Dropout(rate=0.5),
  tf.keras.layers.Dense(units=10, activation='softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

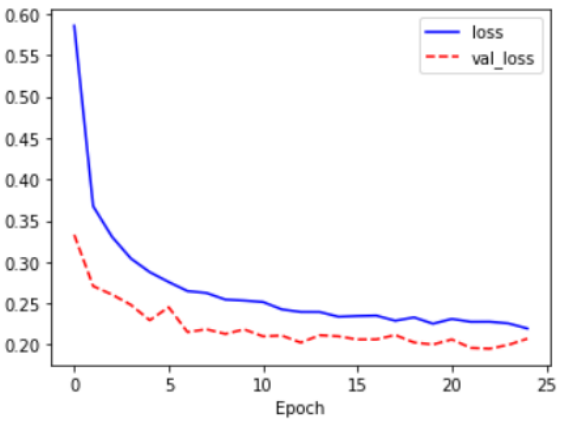
VGG-7

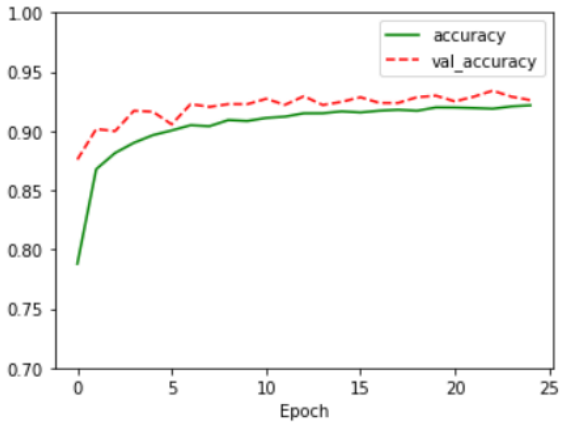
- **다음 패턴을 2차례 반복**
  - 컨볼루션 레이어를 2개 겹치고 풀링 레이어를 1개 사용
  - 과적합을 방지의 드롭아웃 레이어
    - **풀링 레이어 다음에 드롭아웃 레이어 배치**
- **Flatten 층, 드롭아웃 레이어를 배치**
  - 다음에 이어지는 3개의 Dense 레이어 사이
- **VGG-7 레이어**
  - 컨볼루션 레이어와 Dense 레이어의 개수

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 14, 14, 128)       73856
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
dropout_5 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               4719104
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                2570
=================================================================
Total params: 5,240,842
Trainable params: 5,240,842
Non-trainable params: 0
_________________________________________________________________
```

학습과 시각화

```
history = model.fit(train_X, train_Y, epochs=25, validation_split=0.25)

import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1,2,1)
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'r--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.7, 1)
plt.legend()

plt.show()

model.evaluate(test_X, test_Y, verbose=0)
```





[0.23273345828056335, 0.9169999957084656]

이미지 보강

- **훈련데이터를 보강하는 기법**
  - 훈련데이터에 없는 이미지를 새롭게 만들어 사용
- **ImageDataGenerator**
  - Tensorflow 이미지 보강 작업
    - **rotation_range: 이미지 회전 범위 (degrees)**
    - **width_shift, height_shift: 그림을 수평 또는 수직으로 랜덤하게 평행 이동시키는 범위 (원본 가로, 세로 길이에 대한 비율 값)**
    - **rescale: 원본 영상은 0-255의 RGB 계수로 구성되는데, 이 같은 입력값은 모델을 효과적으로 학습시키기에 너무 높습니다 (통상적인 learning rate를 사용할 경우). 그래서 이를 1/255로 스케일링하여 0-1 범위로 변환시켜줍니다. 이는 다른 전처리 과정에 앞서 가장 먼저 적용됩니다.**
    - **shear_range: 임의 전단(기울기) 변환 (shearing transformation) 범위**
    - **zoom_range: 임의 확대/축소 범위**
    - **horizontal_flip: True로 설정할 경우, 50% 확률로 이미지를 수평으로 뒤집습니다. 원본 이미지에 수평 비대칭성이 없을 때 효과적입니다. 즉, 뒤집어도 자연스러울 때 사용하면 좋습니다.**
    - **fill_mode 이미지를 회전, 이동하거나 축소할 때 생기는 공간을 채우는 방식**

ImageDataGenerator 생성

- 인자 값 지정

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np

image_generator = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.10,
    shear_range=0.5,
    width_shift_range=0.10,
    height_shift_range=0.10,
    horizontal_flip=True,
    vertical_flip=False)
```

ImageDataGenerator 이미지 생성

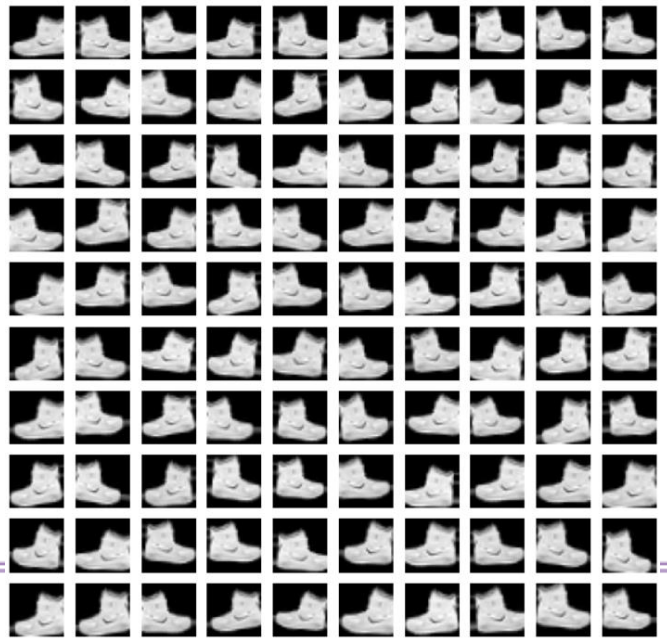
- **메소드 flow()**
  - 데이터 및 라벨 배열, 증가 된 데이터의 일괄 처리를 생성
    - **입력자료 x, 정답(레이블) y**

```
flow(
    x, y=None, batch_size=32, shuffle=True, sample_weight=None, seed=None,
    save_to_dir=None, save_prefix='', save_format='png', subset=None
)
```

```
augment_size = 100

x_augmented = image_generator.flow(
  np.tile(train_X[0].reshape(28*28), 100).reshape(-1, 28, 28, 1),
  np.zeros(augment_size),
  batch_size=augment_size, shuffle=False).next()[0]
```

ImageDataGenerator 생성 그림 그리기

```
# 새롭게 생성된 이미지 표시
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 10))
for c in range(100):
  plt.subplot(10, 10, c+1)
  plt.axis('off')
  plt.imshow(x_augmented[c].reshape(28, 28), cmap='gray')

plt.show()
```



3만개 이미지 보강

- **augment_size=30000으로 설정**
  - 훈련 데이터의 50%인 30,000개의 이미지를 추가하기 위해
- **0~59,999 범위의 정수 중에서 30,000개의 정수 난수**
  - 이미지를 변형할 원본 이미지를 찾기 위해 np.random.randint()함수를 활용
    - **정수는 중복 가능**
  - 중복을 원치 않으면
    - **np.random.randint() 대신에 np.random.choice() 함수를 사용**
      - **replace 인수를 False로 설정**
- **copy() 함수를 사용하여 원본은 보전**
- **image_generator.flow() 함수를 사용**
  - 30,000개의 새로운 이미지를 생성
- **np.concatenate()**
  - 훈련 데이터에 보강 이미지를 추가

이미지 보강 소스

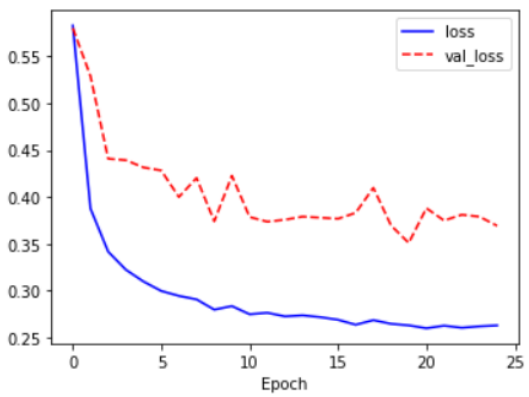
```
import tensorflow as tf
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_X, train_Y), (test_X, test_Y) = fashion_mnist.load_data()

train_X = train_X / 255.0
test_X = test_X / 255.0

# reshape 이전
print(train_X.shape, test_X.shape)

train_X = train_X.reshape(-1, 28, 28, 1)
test_X = test_X.reshape(-1, 28, 28, 1)

# reshape 이후
print(train_X.shape, test_X.shape)

image_generator = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.10,
    shear_range=0.5,
    width_shift_range=0.10,
    height_shift_range=0.10,
    horizontal_flip=True,
    vertical_flip=False)

augment_size = 30000

randidx = np.random.randint(train_X.shape[0], size=augment_size)
x_augmented = train_X[randidx].copy()
y_augmented = train_Y[randidx].copy()
x_augmented = image_generator.flow(x_augmented, np.zeros(augment_size), batch_size=augment_size, shuffle=False).next()[0]

# 원래 데이터인 x_train에 이미지 보강된 x_augmented를 추가합니다.
train_X = np.concatenate((train_X, x_augmented))
train_Y = np.concatenate((train_Y, y_augmented))

print(train_X.shape)
```

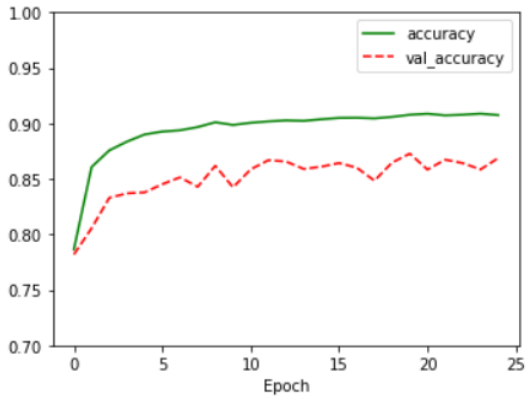
모델 정의

```
import tensorflow as tf

model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(input_shape=(28,28,1), kernel_size=(3,3), filters=32,
                         padding='same', activation='relu'),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=64, padding='same', activation='relu'),
  tf.keras.layers.MaxPool2D(pool_size=(2,2)),
  tf.keras.layers.Dropout(rate=0.5),
  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=128, padding='same', activation='relu'),

  tf.keras.layers.Conv2D(kernel_size=(3,3), filters=256, padding='valid', activation='relu'),

  tf.keras.layers.MaxPool2D(pool_size=(2,2)),
  tf.keras.layers.Dropout(rate=0.5),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(units=512, activation='relu'),
  tf.keras.layers.Dropout(rate=0.5),
  tf.keras.layers.Dense(units=256, activation='relu'),
  tf.keras.layers.Dropout(rate=0.5),
  tf.keras.layers.Dense(units=10, activation='softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# model.summary()
history = model.fit(train_X, train_Y, epochs=25, validation_split=0.25)
```

훈련 시각화

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1,2,1)
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'r--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.7, 1)
plt.legend()

plt.show()

model.evaluate(test_X, test_Y, verbose=0)
```



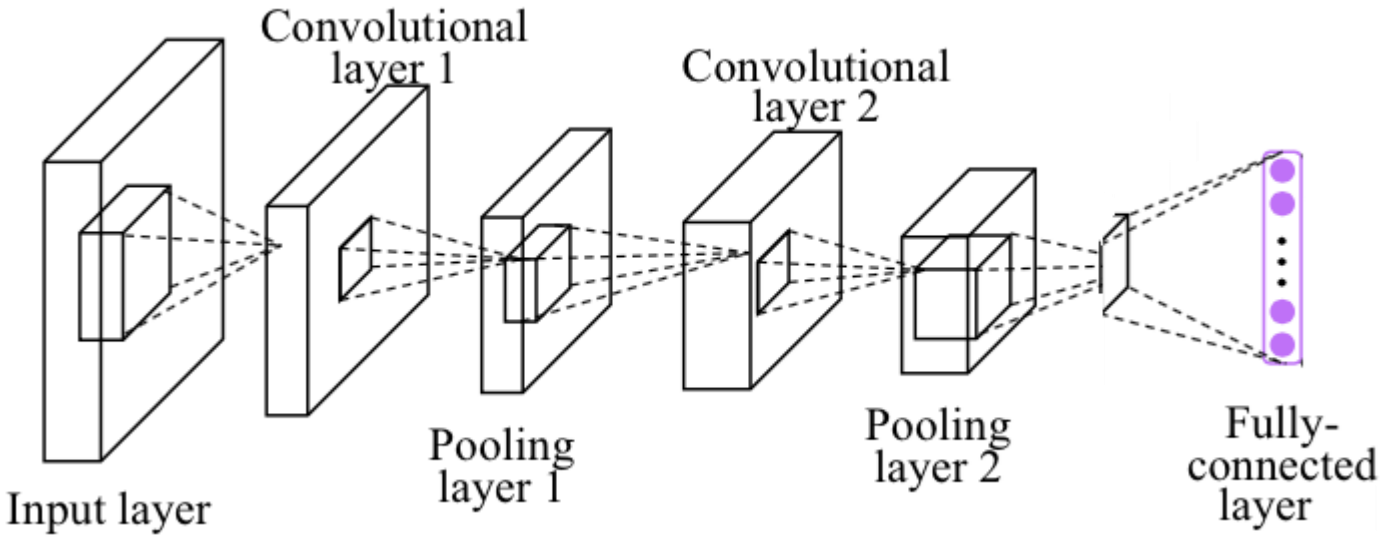


[0.2110530138015747, 0.9251000285148621]

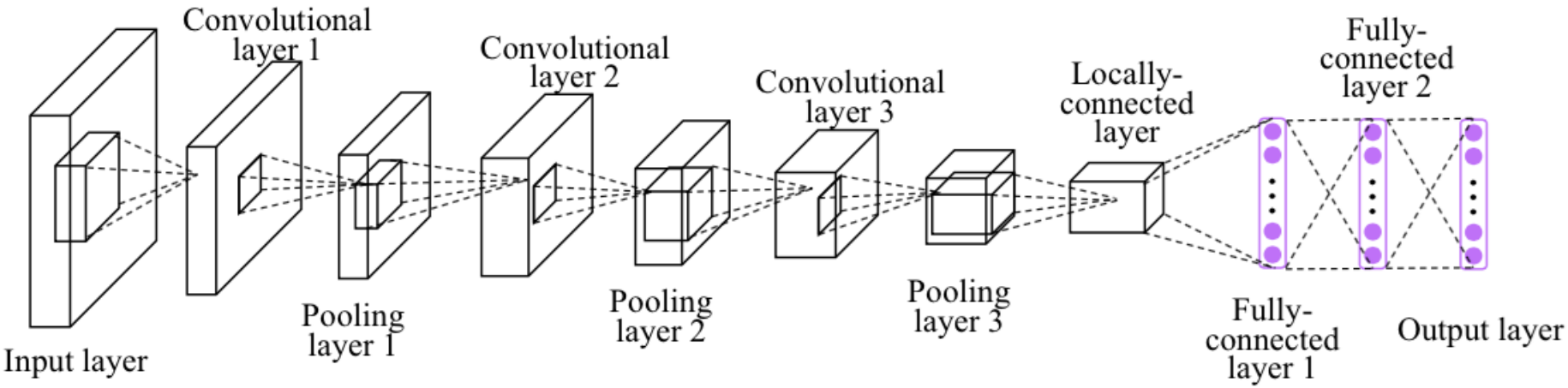
CNN(합성곱) 텐서플로 구현

MNIST simple CNN

- 정확도 98.6%



Deep CNN


Convolutional layer 1
Convolutional layer 2
Convolutional layer 3
Locally-connected layer
Fully-connected layer 2
Input layer
Pooling layer 1
Pooling layer 2
Pooling layer 3
Fully-connected layer 1
Output layer

패라미터(W와 b) 수 계산

W와 b의 수 계산

- 컨볼루션 패라미터 수 = 커널사이즈^2 * 커널수 * 채널(색상) + 커널수(bias)
- 일반 완전 연결층 패라미터 수 = (입력수 + 1) * 출력수

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_26 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 13, 13, 32)        0
_________________________________________________________________
flatten_14 (Flatten)         (None, 5408)              0
_________________________________________________________________
dense_31 (Dense)             (None, 128)               692352
_________________________________________________________________
dense_32 (Dense)             (None, 10)                1290
=================================================================
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
```

(3*3 * 32 * 1) + 32

(5408 + 1) * 128

```python
# 2. 모델 구성하기
model = Sequential()
    #컨볼루션 층
model.add(Conv2D(32, input_shape=(28,28,1), kernel_size=(3, 3), activation='relu'))
        #맥스플링
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
    #완전 연결층
model.add(Dense(128, activation='relu'))
    #출력층
model.add(Dense(10, activation='softmax'))
```