

2020~21 겨울특강

텐서플로 기반 딥러닝

동양미래대학교 컴퓨터정보공학과 강환수 교수

Fashion MNIST 데이터셋

CNN 적용

실습 파일

- 14W-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

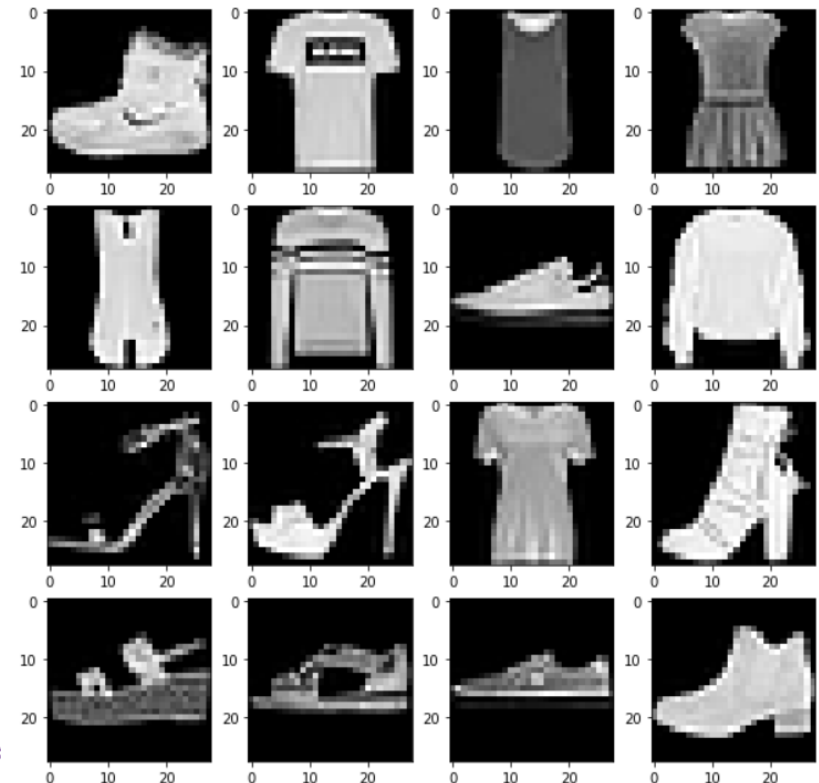
```
import matplotlib.pyplot as plt

# 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 field)
 - (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')
])
```

• 패러미터 수 계산 방법

컨볼루션 패러미터 수 = 커널사이즈² * 커널수 * 채널(색상) + 커널수(bias)
 일반 완전 연결층 패라미터 수 = (입력수 + 1) * 출력수

Model: "sequential"

Layer (type)	Output Shape	Param #	
conv2d (Conv2D)	(None, 26, 26, 16)	160	$\text{커널 수}(K) * \text{커널 사이즈}(F)^2 * (\text{채널:색상 수}(D)) + \text{커널 수}(K)$ $16 * 3 * 3 * 1 + 16 = 160$
conv2d_1 (Conv2D)	(None, 24, 24, 32)	4640	$\text{커널 수}(K) * \text{커널 사이즈}(F)^2 * (\text{채널:색상 수}(D)) + \text{커널 수}(K)$ $32 * 3 * 3 * 16 + 32 = 4,640$
conv2d_2 (Conv2D)	(None, 22, 22, 64)	18496	$\text{커널 수}(K) * \text{커널 사이즈}(F)^2 * (\text{채널:색상 수}(D)) + \text{커널 수}(K)$ $64 * 3 * 3 * 32 + 64 = 18,496$
flatten (Flatten)	(None, 30976)	0	
dense (Dense)	(None, 128)	3965056	$(\text{이전 출력 노드 수} + 1) * \text{노드 수}$ $(30976 + 1) * 128$
dense_1 (Dense)	(None, 10)	1290	$(128 + 1) * 10$

Total params: 3,989,642

Trainable params: 3,989,642

Non-trainable params: 0

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											

학습

• 기본 batch_size=32

```
[7] history = model.fit(train_X, train_Y, epochs=10, validation_split=0.25)
```

```
Epoch 1/10
1407/1407 [=====] - 19s 9ms/step - loss: 0.6057 - accuracy: 0.7896 - val_loss: 0.3937 - val_accuracy: 0.8618
Epoch 2/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.3348 - accuracy: 0.8775 - val_loss: 0.3654 - val_accuracy: 0.8703
Epoch 3/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.2858 - accuracy: 0.8929 - val_loss: 0.3897 - val_accuracy: 0.8707
Epoch 4/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.2369 - accuracy: 0.9143 - val_loss: 0.4075 - val_accuracy: 0.8651
Epoch 5/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.2100 - accuracy: 0.9219 - val_loss: 0.4072 - val_accuracy: 0.8677
Epoch 6/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.1760 - accuracy: 0.9361 - val_loss: 0.4739 - val_accuracy: 0.8668
Epoch 7/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.1588 - accuracy: 0.9412 - val_loss: 0.5035 - val_accuracy: 0.8638
Epoch 8/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.1413 - accuracy: 0.9474 - val_loss: 0.5917 - val_accuracy: 0.8597
Epoch 9/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.1341 - accuracy: 0.9486 - val_loss: 0.5798 - val_accuracy: 0.8633
Epoch 10/10
1407/1407 [=====] - 12s 9ms/step - loss: 0.1147 - accuracy: 0.9562 - val_loss: 0.6886 - val_accuracy: 0.8608
```

$[60,000 * (3/4) / 32] + 1$

검증 데이터의 손실과 정확도

컨볼루션 신경망 모델 학습

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

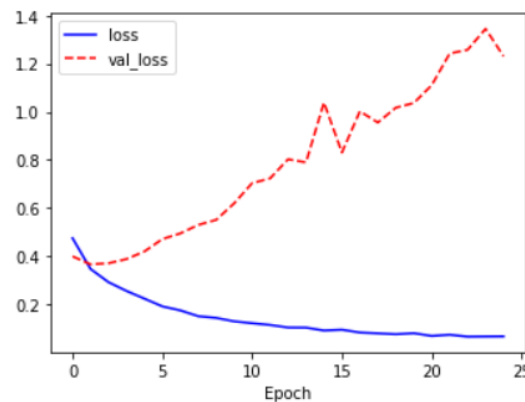
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))
```

```
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()
```

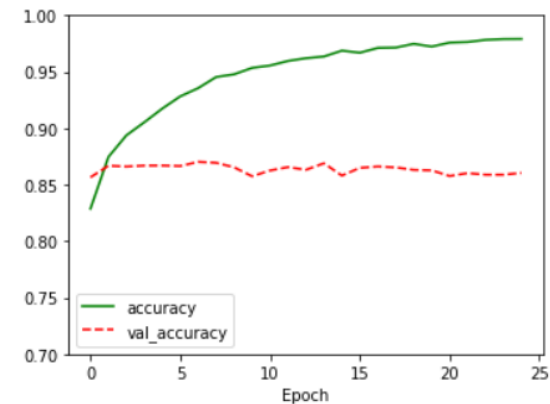
```
plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'r--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.7, 1)
plt.legend()
```

```
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 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_10 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_5 (MaxPooling2D)	(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)
```

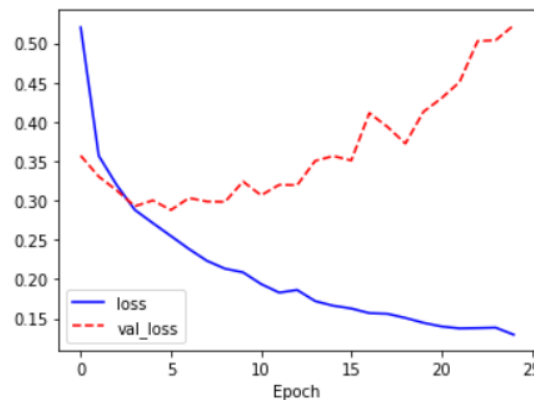
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))
```

```
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()
```

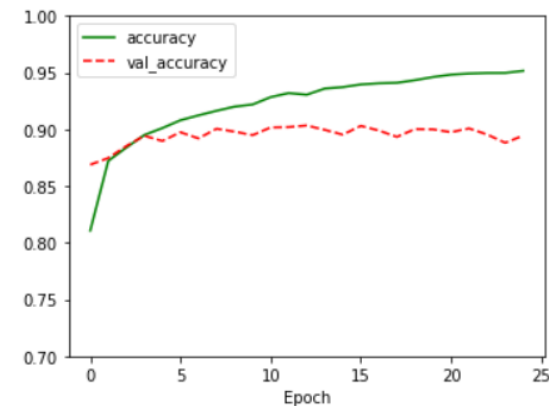
```
plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'r--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.7, 1)
plt.legend()
```

```
plt.show()
```

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

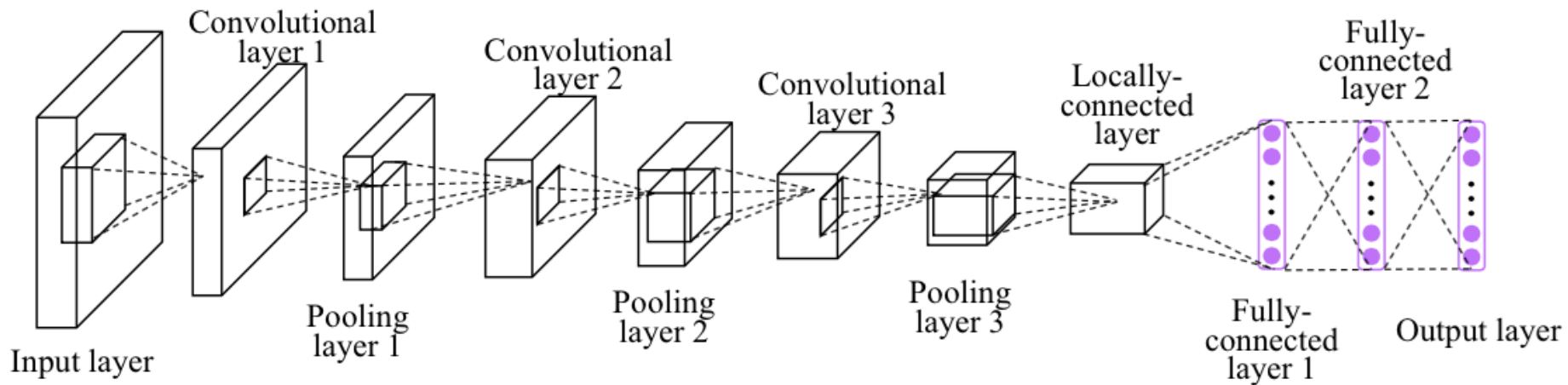
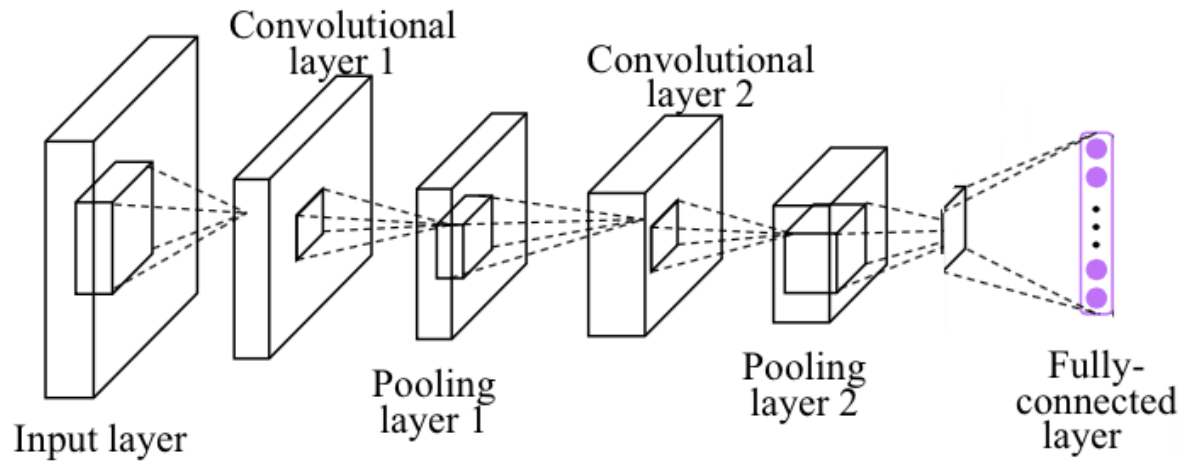


[0.5307855606079102, 0.8906999826431274]



CNN(합성곱) 다양한 구조

다양한 CNN 구조



Deep CNN 모델 적용

```
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()
```

패러미터 수

- 5백2십만개 이상

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 28, 28, 32)	320
conv2d_13 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 64)	0
dropout_2 (Dropout)	(None, 14, 14, 64)	0
conv2d_14 (Conv2D)	(None, 14, 14, 128)	73856
conv2d_15 (Conv2D)	(None, 12, 12, 256)	295168
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 256)	0
dropout_3 (Dropout)	(None, 6, 6, 256)	0
flatten_4 (Flatten)	(None, 9216)	0
dense_8 (Dense)	(None, 512)	4719104
dropout_4 (Dropout)	(None, 512)	0
dense_9 (Dense)	(None, 256)	131328
dropout_5 (Dropout)	(None, 256)	0
dense_10 (Dense)	(None, 10)	2570

=====
 Total params: 5,240,842
 Trainable params: 5,240,842
 Non-trainable params: 0
 =====

정확도 92%

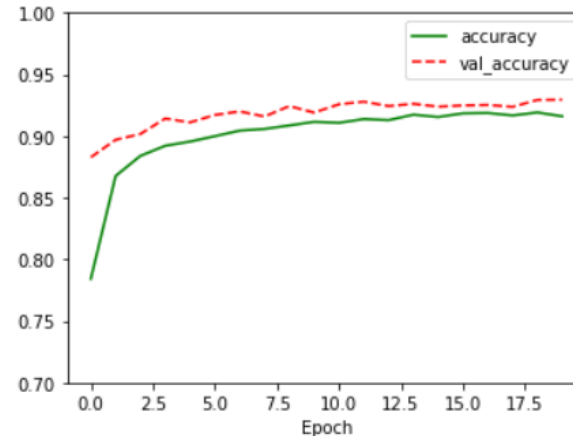
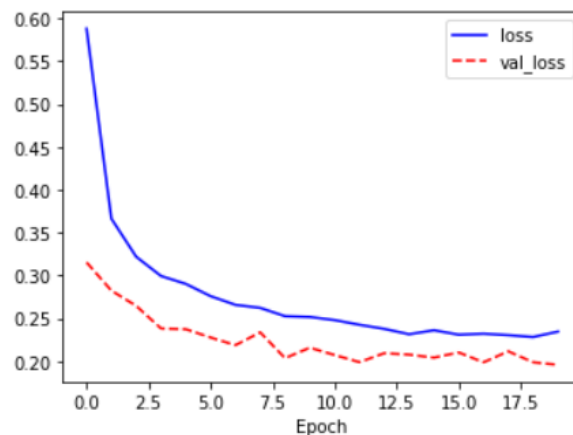
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

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)
```



[0.21626901626586914, 0.9236999750137329]