MNIST 손글씨
데이터 인식하기 2

목차

기본 신경망

```
from keras.datasets import mnist
```

```
# MNIST 데이터 불러오기
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

X_train = X_train.reshape(X_train.shape[0], 784).astype('float32') / 255
X_test = X_test.reshape(X_test.shape[0], 784).astype('float32') / 255

Y_train = np_utils.to_categorical(Y_train, 10)
Y_test = np_utils.to_categorical(Y_test, 10)
```

학습셋 이미지 수 : 60000
테스트셋 이미지 수 : 10000

기본 신경망

```
# 모델 프레임 설정
model = Sequential()
model.add(Dense(512, input_dim=784, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

기본 신경망

```
# 모델 실행 환경 설정
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

기본 신경망

```
# 모델 최적화 설정
MODEL_DIR = './model/'
if not os.path.exists(MODEL_DIR):
    os.mkdir(MODEL_DIR)

modelpath="./model/{epoch:02d}-{val_loss:.4f}.hdf5"
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1, save_best_only=True)
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=10)
```

기본 신경망

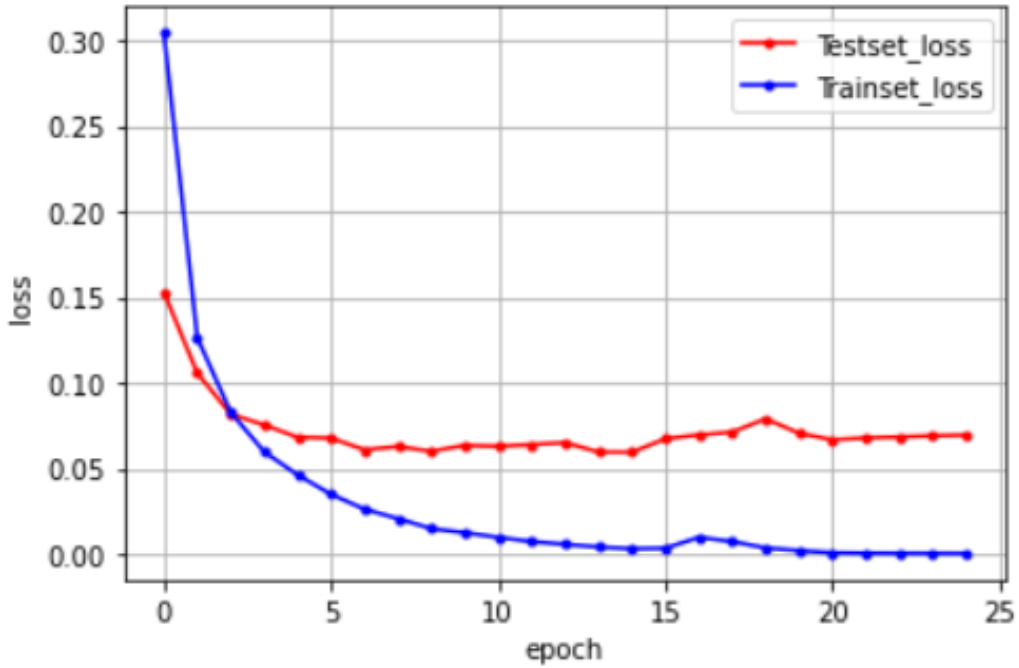
```
# 모델의 실행
history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test),
                    epochs=30, batch_size=200, verbose=0, callbacks=[early_stopping_callback,checkpointer])

# 테스트 정확도 출력
print("₩n Test Accuracy: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

기본 신경망

```
# 테스트 셋의 오차
y_vloss = history.history['val_loss']

# 학습셋의 오차
y_loss = history.history['loss']

# 그래프로 표현
x_len = numpy.arange(len(y_loss))
plt.plot(x_len, y_vloss, marker='.', c="red", label='Testset_loss')
plt.plot(x_len, y_loss, marker='.', c="blue", label='Trainset_loss')

# 그래프에 그리드를 주고 레이블을 표시
plt.legend(loc='upper right')
# plt.axis([0, 20, 0, 0.35])
plt.grid()
plt.xlabel('epoch')
plt.ylabel('loss')
plt.show()
```

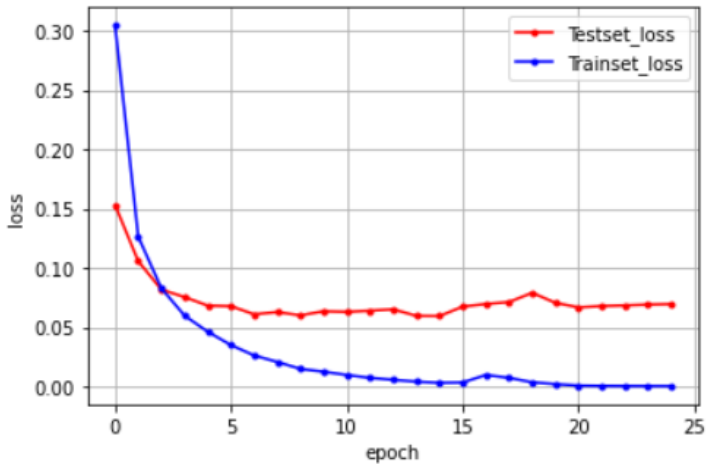
기본 신경망

```
Test Accuracy: 0.9838
```



기본 신경망

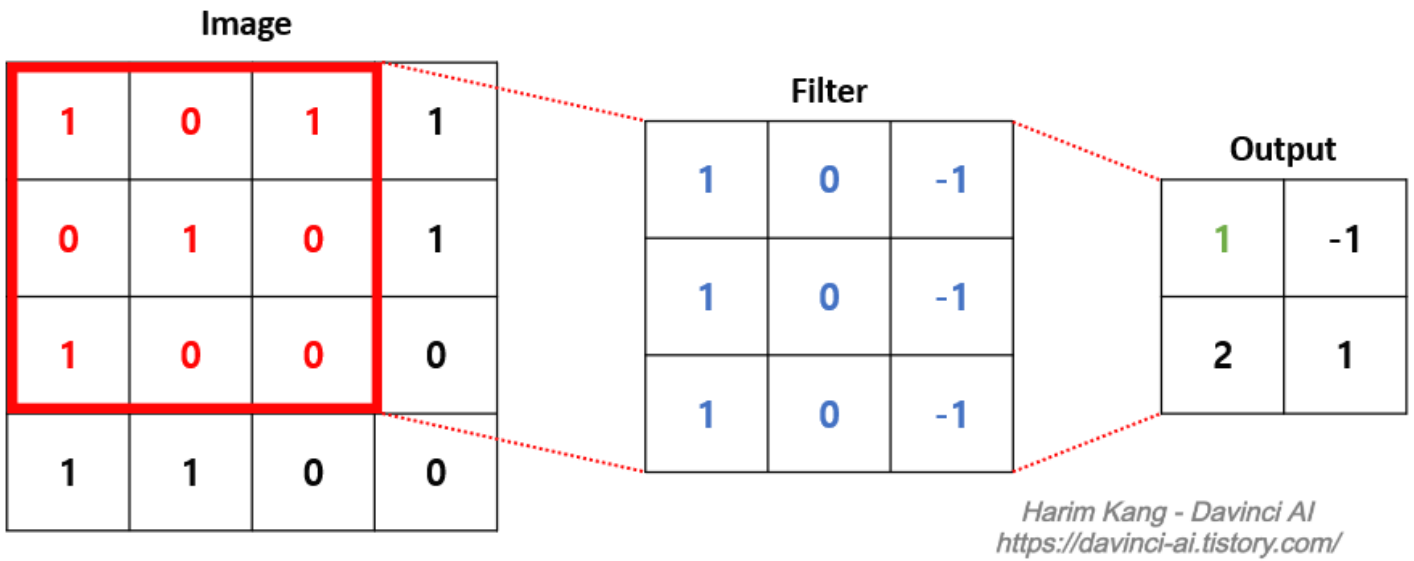
Test Accuracy: 0.9838




```
Epoch 00014: val_loss improved from 0.06000 to 0.05957, saving model to ./model₩14-0.0596.hdf5

Epoch 00015: val_loss improved from 0.05957 to 0.05943, saving model to ./model₩15-0.0594.hdf5

Epoch 00016: val_loss did not improve from 0.05943

Epoch 00017: val_loss did not improve from 0.05943

Epoch 00018: val_loss did not improve from 0.05943

Epoch 00019: val_loss did not improve from 0.05943

Epoch 00020: val_loss did not improve from 0.05943

Epoch 00021: val_loss did not improve from 0.05943

Epoch 00022: val_loss did not improve from 0.05943

Epoch 00023: val_loss did not improve from 0.05943

Epoch 00024: val_loss did not improve from 0.05943

Epoch 00025: val_loss did not improve from 0.05943
313/313 [==============================] - 1s 3ms/step - loss: 0.0695 - accuracy: 0.9838

 Test Accuracy: 0.9838
```

컨볼루션 신경망

CNN(Convolutional Neural Network, 컨볼루션 신경망, 합성곱 신경망)



1 x 1 + 0 x 0 + 1 x -1 + 0 x 1 + 1 x 0 + 0 x -1 + 1 x 1 + 0 x 0 + 0 x -1 = 1

컨볼루션 신경망

subsampling(서브 샘플링)

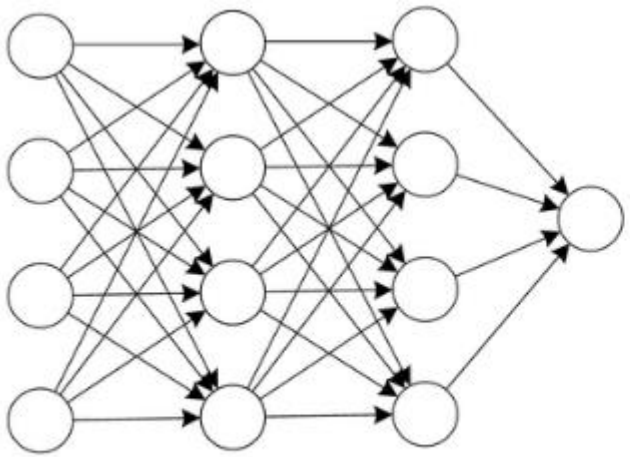
↓

Max pooling(맥스 풀링)

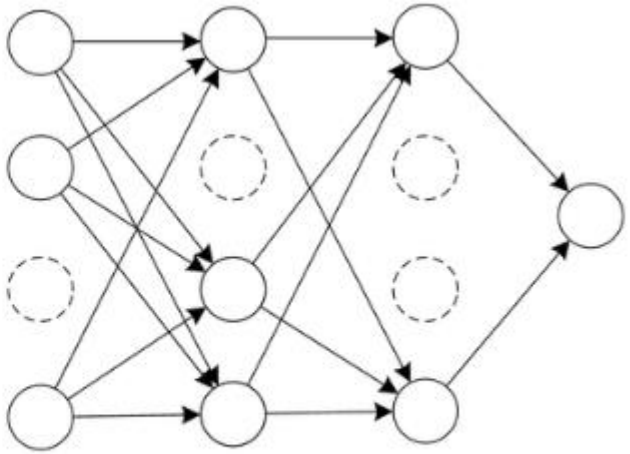
정해진 구역 안에서 가장 큰 값만 추출

컨볼루션 신경망

Drop out(드롭 아웃)



(a) Standard Neural Network



(b) Network after Dropout

컨볼루션 신경망

Flatten() 함수

```
a = np.array([[1, 2, 3],[4, 5, 6]])
a
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
a.flatten()
```

```
array([1, 2, 3, 4, 5, 6])
```

컨볼루션 신경망 적용

CNN(Convolutional Neural Network, 컨볼루션 신경망, 합성곱 신경망)

↓

```
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))
```

subsampling(서브 샘플링) / max pooling(맥스 풀링)

↓

```
model.add(MaxPooling2D(pool_size=2))
```

컨볼루션 신경망 적용

Drop out(드롭 아웃)

model.add(Dropout(0.25))

Flatten() 함수

model.add(Flatten())

컨볼루션 신경망 적용

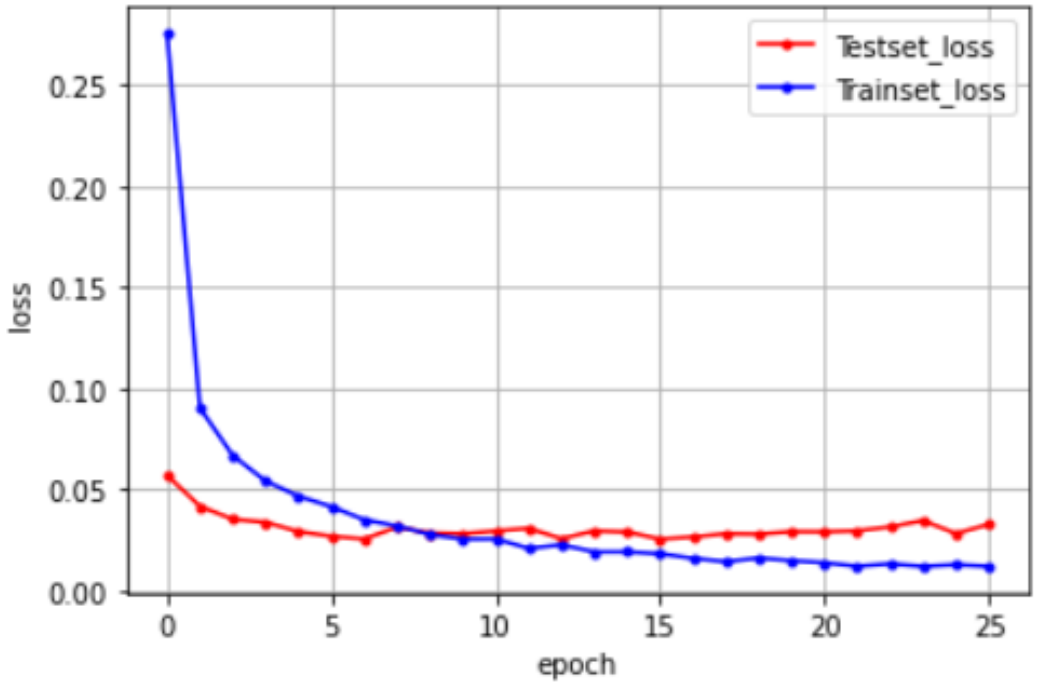
```
# 컨볼루션 신경망의 설정
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128,  activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

컨볼루션 신경망 적용

```
Epoch 00015: val_loss did not improve from 0.02573

Epoch 00016: val_loss improved from 0.02573 to 0.02567, saving model to ./model₩16-0.0257.hdf5

Epoch 00017: val_loss did not improve from 0.02567

Epoch 00018: val_loss did not improve from 0.02567

Epoch 00019: val_loss did not improve from 0.02567

Epoch 00020: val_loss did not improve from 0.02567

Epoch 00021: val_loss did not improve from 0.02567

Epoch 00022: val_loss did not improve from 0.02567

Epoch 00023: val_loss did not improve from 0.02567

Epoch 00024: val_loss did not improve from 0.02567

Epoch 00025: val_loss did not improve from 0.02567

Epoch 00026: val_loss did not improve from 0.02567
313/313 [==============================] - 5s 14ms/step - loss: 0.0329 - accuracy: 0.9925

 Test Accuracy: 0.9925
```

컨볼루션 신경망 적용

```
Test Accuracy: 0.9925
```



감사합니다