MNIST 손글씨 예측과 오류 확인

테스트 데이터의 첫 번째 손글씨 예측 결과를 확인

- **model.predict(input)**
  - 모델의 fit(), evaluate()에 입력과 같은 형태가 필요
- **첫 번째 손글씨만 알아보더라도 3차원 배열로 입력**
  - 슬라이스해서 사용
    - **x_test[:1]**
  - 예측
    - **pred_result = model.predict(x_test[:1])**
- **결과**
  - 정수 ?
  - 실제
    - **(1, 10)의 이차원 배열**
  - 결과
    - **10개의 0~1의 실수**

```
[33]  1 # 테스트 데이터의 첫 번째 손글씨 예측 결과를 확인
      2 print(x_test[:1].shape)
      3
      4 pred_result = model.predict(x_test[:1])
      5 print(pred_result.shape)
      6 print(pred_result)
      7 print(pred_result[0])
```

```
(1, 28, 28)
(1, 10)
[[8.7629097e-12 4.7056760e-14 2.5735870e-12 1.3529770e-07 1.9923079e-21
  1.6554103e-12 2.3112234e-21 9.9999988e-01 2.5956004e-10 3.6446388e-10]]
[8.7629097e-12 4.7056760e-14 2.5735870e-12 1.3529770e-07 1.9923079e-21
 1.6554103e-12 2.3112234e-21 9.9999988e-01 2.5956004e-10 3.6446388e-10]
```

이게 과연 무엇 일까?

정답으로 나온 10개의 실수 예측

- **0~1**
  - 확률 값?
  - 10 개 합이 1
- **One hot encoding**
  - Indicator/dummy ?
- **argmax() 로**

```
import numpy as np

# 10 개의 수를 더하면?
one_pred = pred_result[0]
print(one_pred.sum())

# 혹시 가장 큰 수가 있는 첨자가 결과
one = np.argmax(one_pred)
print(one)
```

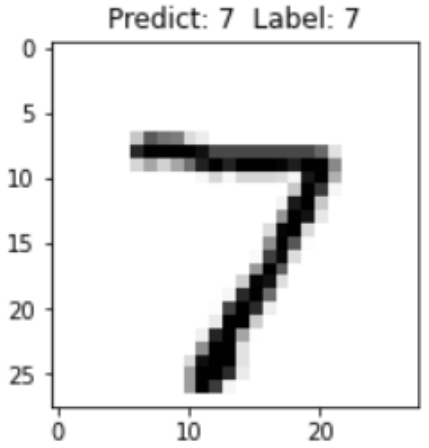
```
[8.7629097e-12      0
4.7056760e-14       1
2.5735870e-12       2
1.3529770e-07       3
1.9923079e-21       4
1.6554103e-12       5
2.3112234e-21       6
9.9999988e-01       7
2.5956004e-10       8
3.6446388e-10]      9
```

실제 손글씨를 그려 결과와 비교

- **맞은 결과 7**
  - 예측
    - **predict()**
    - **예측 결과는 원핫 인코딩**
      - **다시 argmax()로 변환**

```
import numpy as np

# 10 개의 수를 더하면?
one_pred = pred_result[0]
print(one_pred.sum())

# 혹시 가장 큰 수가 있는 첨자가 결
one = np.argmax(one_pred)
print(one)

import matplotlib.pyplot as plt

plt.figure(figsize=(5, 3))
tmp = "Predict: " + str(one) + "  Label: " + str(y_test[0])
plt.title(tmp)
plt.imshow(x_test[0], cmap='Greys')
```

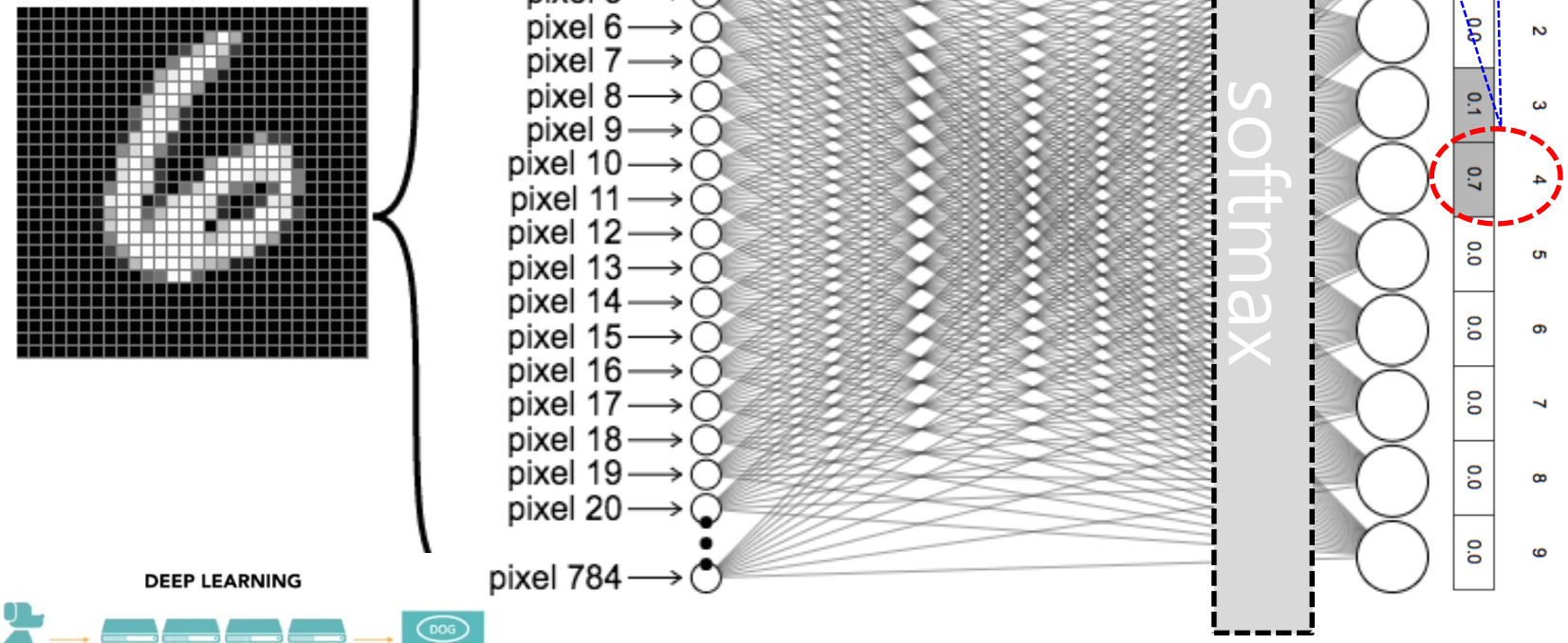
```
1.0
7
<matplotlib.image.AxesImage at 0x7f4a7890f2b0>
```



활성화 함수 softmax()

```
# 층을 차례대로 쌓아 tf.keras.models.Sequential 모델을 생성
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

소프트맥스의 결과가 가장 큰 것이 결과, 즉 4로 예측



원핫 인코딩과 flatten

- 메소드 np.argmax()

```
[46]  1 import numpy as np
      2
      3 ##############################################
      4 # 원핫 인코딩과 argmax 학습
      5 print(np.argmax([5, 4, 10, 1, 2]))
      6 print(np.argmax([3, 1, 4, 9, 6, 7, 2]))
      7 print(np.argmax([[0.1, 0.8, 0.1], [0.7, 0.2, 0.1], [0.2, 0.1, 0.7]], axis=1))
```

```
2
3
[1 0 2]
```

- 메소드 arr.flatten()

```
[50]  1 ##############################################
      2 # 간단한 자료 처리
      3 import numpy as np
      4
      5 x = np.array([2, 3, 254, 5, 6, 3])
      6 x = x / 255.0
      7 print(x)
      8
      9 x = x.reshape(2, 3)
     10 print(x)
     11
     12 x = x.flatten()
     13 print(x)
```

```
[0.00784314 0.01176471 0.99607843 0.01960784 0.02352941 0.01176471]
[[0.00784314 0.01176471 0.99607843]
 [0.01960784 0.02352941 0.01176471]]
[0.00784314 0.01176471 0.99607843 0.01960784 0.02352941 0.01176471]
```

임의의 20개 예측 값과 정답

- **예측 값과 20개의 첨자 구하기**
  - 리스트 pred_result
    - **모델의 예측 결과, 확률 값**
  - 리스트 pred_labels
    - **모델의 예측 결과, 정수**
  - 리스트 samples
    - **출력할 20개의 첨자 리스트**

```
from random import sample
import numpy as np

# 예측한 softmax의 확률이 있는 리스트 pred_result
pred_result = model.predict(x_test)

# 실제 예측한 정답이 있는 리스트 pred_labels
pred_labels = np.argmax(pred_result, axis=1)

#랜덤하게 20개의 훈련용 자료를 예측 값과 정답, 그림을 그려 보자.
samples = sorted(sample(range(len(x_test)), nrows * ncols)) # 출력할 첨자 선정
```

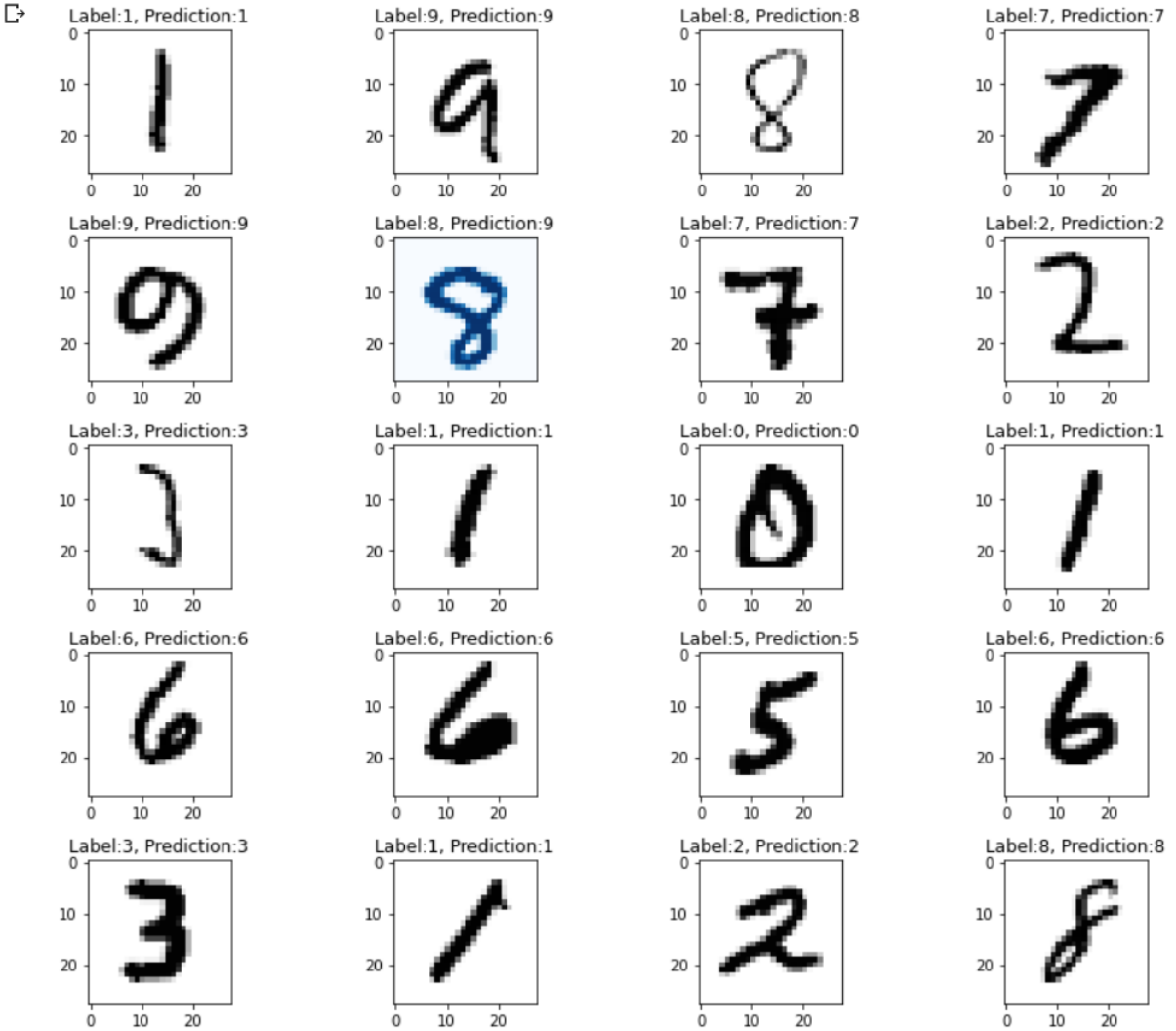
임의의 20개 예측 값과 정답, 손글씨 그리기

- **pred_labels[n] == y_test[n]**
  - 예측이 맞는 경우
  - 리스트 pred_labels
    - **모델의 예측 결과, 정수**
  - 리스트 y_test
    - **훈련 데이터 정답**
- **예측이 틀린 것은 'Blues'로 그리기**

```
# 임의의 20개 그리기
count = 0
nrows, ncols = 5, 4
plt.figure(figsize=(12,10))
for n in samples:
    count += 1
    plt.subplot(nrows, ncols, count)
    # 예측이 틀린 것은 파란색으로 그리기
    cmap = 'Greys' if ( pred_labels[n] == y_test[n]) else 'Blues'
    plt.imshow(x_test[n].reshape(28, 28), cmap=cmap, interpolation='nearest')
    tmp = "Label:" + str(y_test[n]) + ", Prediction:" + str(pred_labels[n])
    plt.title(tmp)

plt.tight_layout()
plt.show()
```

임의의 20개 샘플 예측 값과 정답 그리기

- 8을 9로 예측



임의의 20개 샘플 예측 값과 정답 그리기 소스

```
################################################
from random import sample
import numpy as np

# 예측한 softmax의 확률이 있는 리스트 pred_result
pred_result = model.predict(x_test)

# 실제 예측한 정답이 있는 리스트 pred_labels
pred_labels = np.argmax(pred_result, axis=1)

#랜덤하게 20개의 훈련용 자료를 예측 값과 정답, 그림을 그려 보자.
samples = sorted(sample(range(len(x_test)), nrows * ncols)) # 출력할 첨자 선정

# 임의의 20개 그리기
count = 0
nrows, ncols = 5, 4
plt.figure(figsize=(12,10))
for n in samples:
    count += 1
    plt.subplot(nrows, ncols, count)
    # 예측이 틀린 것은 파란색으로 그리기
    cmap = 'Greys' if ( pred_labels[n] == y_test[n]) else 'Blues'
    plt.imshow(x_test[n].reshape(28, 28), cmap=cmap, interpolation='nearest')
    tmp = "Label:" + str(y_test[n]) + ", Prediction:" + str(pred_labels[n])
    plt.title(tmp)

plt.tight_layout()
plt.show()
################################################
```

예측이 틀린 20개 찾기

- 틀린 것을 임의의 20개를 찾아 첨자를 리스트 samples에 저장

```
from random import sample
import numpy as np

################################################
# 예측 틀린 것 첨자를 저장할 리스트
mispred = []
# 예측한 softmax의 확률이 있는 리스트 pred_result
pred_result = model.predict(x_test)

# 실제 예측한 정답이 있는 리스트 pred_labels
pred_labels = np.argmax(pred_result, axis=1)

for n in range(0, len(y_test)):
    if pred_labels[n] != y_test[n]:
        mispred.append(n)
print('정답이 틀린 수', len(mispred))

# 랜덤하게 틀린 것 20개의 첨자 리스트 생성
samples = sample(mispred, 20)
print(samples)
```

```
정답이 틀린 수 195
[6625, 6093, 3946, 5676, 9587, 8311, 3520, 9679, 3558, 4571, 2953, 1112,
3503, 5642, 2369, 6400, 4551, 1247, 3943, 5734]
```

예측이 잘못된 20개 샘플로 그리기

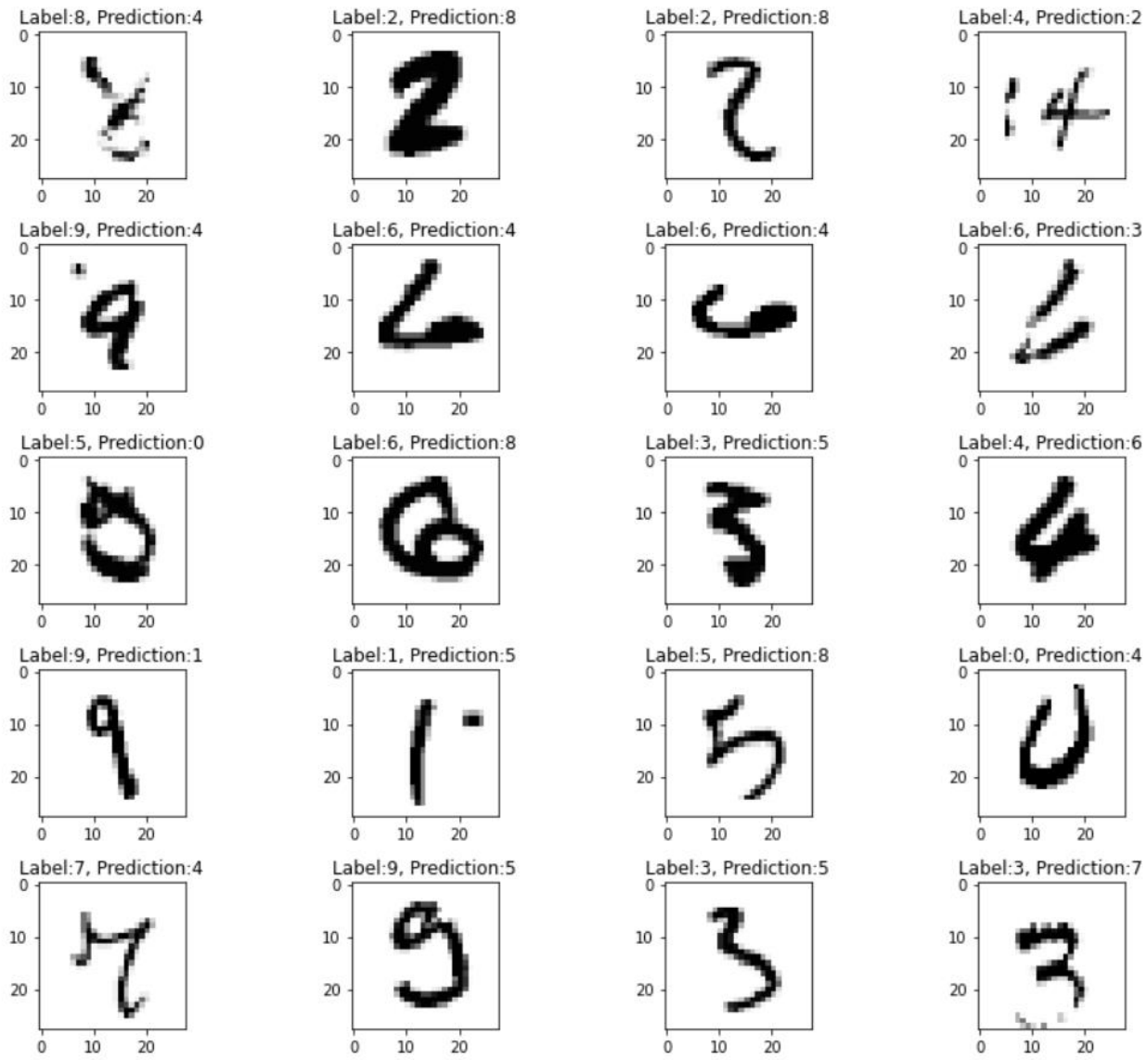
- **틀린 첨자 저장**
  - mispred
  - 196개 중 랜덤하게 20개 선택
- **5행 4열로 그리기**

```
# 틀린 것 20개 그리기
count = 0
nrows, ncols = 5, 4
plt.figure(figsize=(12,10))
for n in samples:
    count += 1
    plt.subplot(nrows, ncols, count)
    plt.imshow(x_test[n].reshape(28, 28), cmap='Greys', interpolation='nearest')
    tmp = "Label:" + str(y_test[n]) + ", Prediction:" + str(pred_labels[n])
    plt.title(tmp)

plt.tight_layout()
plt.show()
################################################
```

```
정답이 틀린 수 195
[6625, 6093, 3946, 5676, 9587, 8311, 3520, 9679, 3558, 4571, 2953, 1112,
3503, 5642, 2369, 6400, 4551, 1247, 3943, 5734]
```

예측이 잘못된 20개 샘플

정답이 틀린 수 195
[6625, 6093, 3946, 5676, 9587, 8311, 3520, 9679, 3558, 4571, 2953, 1112, 3503, 5642, 2369, 6400, 4551, 1247, 3943, 5734]



예측이 잘못된 20개 그리기 소스

```
from random import sample
import numpy as np

##############################################
# 예측 틀린 것 첨자를 저장할 리스트
mispred = []
# 예측한 softmax의 확률이 있는 리스트 pred_result
pred_result = model.predict(x_test)

# 실제 예측한 정답이 있는 리스트 pred_labels
pred_labels = np.argmax(pred_result, axis=1)

for n in range(0, len(y_test)):
    if pred_labels[n] != y_test[n]:
        mispred.append(n)
print('정답이 틀린 수', len(mispred))

# 랜덤하게 틀린 것 20개의 첨자 리스트 생성
samples = sample(mispred, 20)
print(samples)

# 틀린 것 20개 그리기
count = 0
nrows, ncols = 5, 4
plt.figure(figsize=(12,10))
for n in samples:
    count += 1
    plt.subplot(nrows, ncols, count)
    plt.imshow(x_test[n].reshape(28, 28), cmap='Greys', interpolation='nearest')
    tmp = "Label:" + str(y_test[n]) + ", Prediction:" + str(pred_labels[n])
    plt.title(tmp)

plt.tight_layout()
plt.show()
##############################################
```