기계학습 최종 보고서

-CNN Fashion MNIST를 이용한 최적 모델 조건 찾기-

컴퓨터전자시스템공학부

201803964 황수민

201804439 권민지

201803334 정이든

한국외국어대학교

HANKUK UNIVERSITY OF FOREIGN STUDIES

목차

1. 프로젝트의 목적 및 동기

기계학습은 많은 경우에서 휴리스틱한 방법을 통해 가장 최적의 조건으로 학습의 정확도를 높이는 과정을 수행한다. 이러한 최적의 조건을 찾는 과정을 직접 수행해보고, 기계학습 수업을 통해 배운 이론들을 활용하기 위한 목적으로, 이번 프로젝트를 통해 keras에서 제공되는 fashion mnist 이미지셋을 이용하여 tensorflow를 이용한 CNN 코드를 작성하고, 이를 기준으로 활성함수와 레이어 개수, 테스트 데이터와 트레인 데이터 조절 등 CNN 학습에 영향을 주는 여러 요인들을 변경하고 수정하며 가장 높은 정확도를 가진 요소들을 조합하여 기존의 정확도와 비교/분석한다. 이러한 과정을 통해 결과적으로 가장 좋은 accuracy를 보여주는 조건들을 찾아내도록 한다.

최종적으로 이러한 과정을 통해 도출해낸 조건들을 이용하여, MNIST에서 제공하지 않는 다른 이미지들을 직접 테스트를 진행하여 실제로도 이미지가 제대로 구별되는지에 대하여 비교해보도록 한다.

2. 프로젝트의 기본 구조 및 실행 코드

-기본적인 모델 코드(relu 기준)

```
model = keras.Sequential([
 keras.layers.Conv2D(32, (3,3), padding="SAME", activation='relu',
 input_shape=(28,28,1)),
 keras.layers.MaxPooling2D((2,2)),
 keras.layers.Conv2D(64, (3,3), padding="SAME", activation='relu'),
 keras.layers.MaxPooling2D((2,2)),
 keras.layers.Conv2D(64, (3,3), padding="SAME", activation='relu'),
 keras.layers.MaxPooling2D((2,2)),
 keras.layers.Flatten(),
 keras.layers.Dense(64, activation='relu'),
 keras.layers.Dense(10, activation='softmax')
 ])
```

-기본적인 모델 코드(prelu 기준)

```
model = keras.Sequential([
keras.layers.Conv2D(32, (3,3), padding="SAME",input_shape=(28,28,1
)),
tf.keras.layers.PReLU(),
keras.layers.MaxPooling2D((2,2)),
```

```
keras.layers.Conv2D(64, (3,3), padding="SAME"),
tf.keras.layers.PReLU(),
keras.layers.MaxPooling2D((2,2)),
keras.layers.Conv2D(64, (3,3), padding="SAME"),
 tf.keras.layers.PReLU(),
keras.layers.MaxPooling2D((2,2)),
 keras.layers.Flatten(),
 keras.layers.Dense(64),
 tf.keras.layers.PReLU(),
 keras.layers.Dense(10, activation='softmax')
 ])
```

-모델 컴파일 코드

```
model.compile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
```

-이미지 사이즈 조절/reshape/흑백 이미지로 변경 코드

```
import tensorflow as tf
from keras_preprocessing import image

images = image.load_img("/content/Sandal2.png", target_size=(28,
28))
x = image.img_to_array(images)
x = tf.image.rgb_to_grayscale(x)
x = np.expand_dims(x, axis=0)
x = x/255.0
```

-이미지 prediction

```
model.predict(x)
print(model.predict_classes(x))
```

-matplotlib를 이용한 입력된 이미지 픽셀 시각적 확인 코드

```
import numpy as np
import tensorflow as tf
from keras_preprocessing import image

img = x
```

```
plt.figure()
plt.imshow(np.reshape(x,(28,28)))
plt.colorbar()
plt.grid(False)
plt.show()
```

-matplotlib를 이용한 입력된 이미지 분류 정확도 확인

```
def plot_value_array(i, predict_arr, true_label):
predict_arr, true_label = predict_arr[i], true_label[i]
plt.grid(False)
plt.xticks([])
plt.yticks([])
thisplot = plt.bar(range(10), predict_arr, color = "#777777" )
plt.ylim([0,1])
predicted_label = np.argmax(predict_arr)
thisplot[predicted_label].set_color('red')
```

```
pred = model.predict(x)
print(pred)
plot_value_array(0, pred,[x])
plt.xticks(range(10), class_names, rotation = 45)
plt.show()
```

3. 프로젝트의 실행 과정

3-1. 초기 실험

원활한 진행을 위한 기본 환경의 조건(train set, test set, epoch 등의 기준 개수와 횟수)를 구하기 위해 초기 실험을 진행하였을 때, sigmoid 와 tanh, ReLU 활성함수에서 EPOCH에 따른 과적합을 확인해본 결과, Tanh와 ReLU는 EPOCH가 50 이후에서부터 과적합이 발생하고, sigmoid에서는 30 이후부터 발생하기 시작하는 것을 확인할 수 있다. 또한 세 활성함수 모두 공통적으로 train 데이터가 60000개이고 test 데이터가 6000개일 때 가장 높은 정확도를 보여주고 있다. layer에서는 tanh는 원만한 정확도 개선을 보여주지만, sigmoid와 ReLU에서는 과적합이 발생하여 히든 레이어가 히든2+FC 2개 일 때 가장 좋은 결과가 나타나게 된다. 이러한 결과들을 종합하여, 다음 실험에서는 학습 횟수를 40으로 조정하도록 보완하며, 데이터의 갯수는 정확도 변화를 잘 구분할 수 있도록 60000보다는 작은 50000으로 설정하고 테스트 데이터의 개수는 6000개로 고정하여 다시 epoch와 layer 실험을 진행하도록 한다. 추가적으로 다른 Leaky ReLU, PReLU, ELU 활성함수를 조정된 데이터셋과 epoch로 테스트를 진행하도록 다시 계획을 수정하였다.

추후 진행될 개별 이미지들의 정확도를 시각적으로 파악할 수 있도록, 각각의 이미지가 어느 정도의 정확도로 구분되었는지에 대한 코드를 추가하여 최종 결과를 도출하도록 한다.

위의 실험에서는 keras에서 제공되는 fashion minist 이미지셋을 이용하여 tesortflow를 이용한 CNN 코드를 작성하고 이를 기준으로 sigmoid와, tanh, ReLU 활성함수와 레이어 개수, 테스트 데이터와 트레인 데이터 조절을 하여 CNN 학습에 영향을 주는 여러 요인들을 변경하고 수정하여 높은 정확도를 가진 요소들을 조합하여 기존의 정확도와 비교/분석하는 실험을 한 결과를 알아보았다. 이후 이러한 수정된 조건을 통한 실험 진행에 추가적으로 교수님의 피드백을 추가하여 여기에 동일한 구조에서 drop put, batch normalization 유무에 따른 학습 개선의 차이를 추가로 진행하였다.

3-2. 조정된 조건에서의 정확도 테스트

이전 실험에서 이용한 tanh, sigmoid, ReLu를 기존의 임의로 설정한 학습 데이터, 테스트 데이터, epoch로 실험한 결과 Tanh와 ReLU는 epoch가 50, sigmoid는 30 이후부터 과적합이 발생하고 train 데이터가 60000개이고 test 데이터가 6000개 일때 가장 높은 정확도를 보여주는 결과를 얻어 이를 바탕으로 기준 설정을 다음 실험에서 트레인 50000 테스트 6000 epoch 40으로 재설정하였다. 그리고 여기에 추가하여 ReLu를 개선한 모델인 leaky ReLu, PReLU, ELU를 같이 테스트 하여 기존에서 실제로 개선이 되었는지를 확인하였다.

기본:	tanh	MAXPOOL	train50000	test6000	epoch 40		
1차accur	1차time	2차 accuracy	2차 time	3차 accuracy	3차 time	avg accuracy	avg time
0.91717	207.8924444	0.90716666	200.2533059	0.909666657	201.2291796	0.911333323	203.1249766

기본:	sigmoid	MAXPOOL	train5000	test6000	epoch 40		
1차accura	1차time	2차 accuracy	2차 time	3차 accuracy	3차 time	avg accuracy	avg time
0.908833	225.2114024	0.909166694	223.8867638	0.913999975	227.2048581	0.910666664	225.4343414

기본:	RELU	MAXPOOL	train5000	test6000	epoch 40		
1차accur	1차time	2차 accuracy	2차 time	3차 accuracy	3차 time	avg accuracy	avg time
0.90867	212.2471321	0.910333335	208.0925238	0.913999975	282.3541934	0.910999993	234.2312831

기본:	Prelu	MAXPOOL	train5000	test6000	epoch 40		
1차accur	1차time	2차 accuracy	2차 time	3차 accuracy	3차 time	avg accuracy	avg time
0.9188	265.92957	0.906666696	271.805377	0.91833335	250.288521	0.914611121	262.6744893

기본:	leakyReLU	MAXPOOL	train5000	test6000	epoch 40	alpha = 0.1	
1차accur	1차time	2차 accuracy	2차 time	3차 accuracy	3차 time	avg accuracy	avg time
0.91267	219.1173835	0.910166681	219.5144999	0.910666645	225.8307216	0.911166668	221.487535

기본:	ELU	MAXPOOL	train5000	test6000	epoch 40	alpha = 0.1	
1차accura	1차time	2차 accuracy	2차 time	3차 accuracy	3차 time	avg accuracy	avg time
0.910833	244.0773172	0.913166642	248.6106629	0.914166689	252.1105571	0.91272223	248.2661791

이 결과 트레인셋 50000, 테스트셋 6000, epoch 40을 고정인 상태에서 기본 코드로동작된 6개의 활성함수에 대한 정확도 평균은 다음과 같이 나타났다. 현재의 수치상으로는 활성함수 PReLU 가 약 91.46% 가장 높은 정확도를 보이며, 동시에 가장 긴 학습 시간을 나타낸다. 가장 낮은 정확도는 91.09% sigmoid에서 나타난다.

이러한 결과를 통해, 개선된 활성함수를 시도함에 따라, 가장 높은 정확도를 보여주는 EPOCH가 50, 30, 20, 10, 10, 10으로 점점 줄어드는 것을 확인할 수 있다. MNIST의 이미지가 기본적으로 높은 학습 accuracy를 보여주기 때문에 정확도에 있어서 크게 차이가 나타나지는 않지만, 이러한 활성함수 변화에 따라 EPOCH 또한 줄어드는 경향을 보이는 것을 통해, 활성함수의 개선이 학습 횟수를 줄여도 높은 정확도가 나타날 수 있도록 보장한다는 것을 확인할 수 있다.

3-3. EPOCH 변경 테스트

첫 번째로는 epoch를 가지고 실험을 하는데 트레인 데이터와 테스트 데이터는 고정하고 Batch normalization과 Dropout 둘 다 없는 상태로 레이어 3개에 Fully connected layer 2개로 실험을 한다.

[부록-1(a)]참고-활성함수가 sigmoid일 때 Epoch를 변화 실험 시키면 위의 결과가 나오는데 이때 최고 정확도(accurarcy)는 epoch가 50일 때 91.10%가 나오고 최저 정확도는 epoch가 10일 때 88.93%가 나온다.

[부록-2(a)] 참고-다음으로 Tanh활성 함수를 사용할 때의 정확도(accuracy) 실험하였는데 최고 정확도는 epoch가 30일 때 91.11%가 나오고 최저 정확도는 epoch가 200일 때 90.48%가 나왔다.

[부록-3(a)] 참고-그 후 활성함수가 ReLU일 때의 Epoch 변화로 인한 정확도(accuracy)을 실험하였는데 최고 정확도는 epoch가 20일 때 91.50%가 나오고 최저 정확도는 epoch가 50일 때 90.85%가 나왔다.

[부록-4(a)] 참고-활성함수가 LeakyReLU일 때 최고 정확도는 epoch가 10일 때 91.58%가 나오고 최저 정확도는 epoch가 100일 때 91.00%가 나왔다.

[부록-5(a)] 참고-PReLU의 Epoch 실험 결과에서의 최고 정확도는 Epoch 10일 때 92.05%이고 최저 학습 정확도는 epoch가 70일 때 91.20%가 나왔다.

[부록-6(a)] 참고-마지막 ELU에서의 Epoch 변화에 따른 학습 정확도 실험에서는 Epoch가 110일 때 91.38%가 나와 최고 정확도를 가졌는데 Epoch가 10일 때 91.32%로 이와 비슷한 수치로 두번째 높은 정확도를 가졌다. 그리고 최저 정확도는 90.49%로 epoch가 190일 때이다.

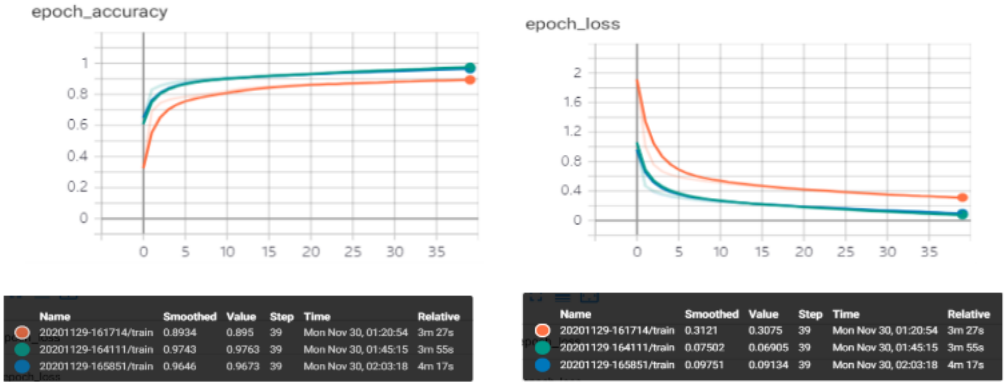
이 위의 실험 결과를 통해 가장 높은 정확도는 활성함수가 PReLU일 때 발생했고 가장 낮은 정확도는 활성함수가 Sigmoid일 때 발생했다. 위의 결과를 정리하면 Sigmoid 다음으로 Tanh 그리고 ReLU, ELU, Leaky ReLU, PReLU 이 순서로 정확도가 높아지는 것을 확인하였다. 이를 통해 실제로 활성함수의 변화에 따라 정확도가 개선되거나 낮아지는 것을 볼 수 있었다.

3-4. 레이어 개수 변화 테스트

두 번째로는 레이어 수만 조절하여 정확도가 어떻게 달라지는가에 대한 실험을 진행하였다. Fully connected layer 2개와 Hidden layer의 개수를 1개, 2개, 3개로 변화시키며 나오는 학습 정확도를 측정하였다.

(sigmoid 예)

sigmoid	layer	maxpool	train50000	test6000	batch없음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.870833337	219.9717414	0.882166684	220.2076147	0.883000016	218.8862057	0.878666679	219.6885206
2	0.906833351	252.1462181	0.90533334	246.9339039	0.906666696	243.9699278	0.906277796	247.6833499
3	0.911833346	272.9477696	0.905833304	274.1504169	0.911833346	266.2515218	0.909833332	271.1165694



[부록-2(b)], [부록-3(b)], [부록-4(b)], [부록-5(b)], [부록-6(b)] 참고-위의 실험으로 Sigmoid를 제외한 나머지 활성함수들은 공통적으로 hidden layer의 수가 2개이고 fully connected layer가 2개 일 때 가장 높은 학습 정확도를 가지고 hidden layer의 개수가 3개일 때 정확도가 다시 낮아지는 것을 볼 수 있었다.

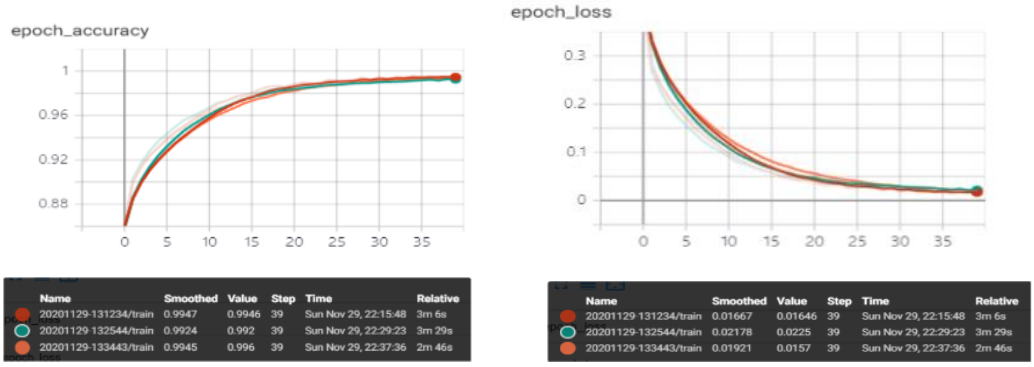
이와 같이, hidden layer의 개수를 조절하는 실험을 진행하였을 때, sigmoid 활성함수를 제외하고는 모두 hidden layer의 개수가 2개일 때 이후로 accuracy가 감소하는 경향을 보였다. 이러한 문제가 발생하는 이유를 찾아본 결과, 이는 레이어가 깊어질수록 local minimum에서 빠져나오기 힘든 문제로 인해 발생하기 때문이라는 것을 알 수 있었다.

3-5. 레이어 개수 변화 및, Batch normalization 테스트

위의 결과를 바탕으로 Layer의 수를 변화하는 실험에 Batch normalization이 있을 경우에 학습 정확도를 실험하였다. 이때 역시 Fully connected layer 2개와 Hidden layer의 개수를 1개, 2개, 3개로 변화시키며 Bach normalization이 있는 나오는 상태에서 학습 정확도를 측정하였다.

(sigmoid 예)

sigmoid	layer	maxpool	train50000	test6000	batch있음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.899833322	176.4152796	0.896166682	172.9733632	0.901833355	172.7727463	0.899277786	174.0537964
2	0.911333323	195.5940123	0.913666666	194.1058161	0.909333348	199.302633	0.911444445	196.3341538
3	0.915499985	217.5681834	0.904999971	218.528976	0.917500019	219.574502	0.912666659	218.5572205



[부록-2(c)], [부록-3(c)], [부록-4(c)], [부록-5(c)], [부록-6(c)] 참고-활성함수 Tanh, leakyReLU, ELU을 사용하여 실험하였을 때는 hidden layer 2개에 Fully connected layer 2개를 가지고 있을 때 가장 높은 정확도를 가졌고 hidden layer가 3개일 때 다시 낮아지는 결과가 나왔다. 하지만 나머지 활성함수의 경우 학습 정확도가 계속 높아지는 결과를 보여주었다. 앞의 실험과 다르게 Batch normalization이 있는 이 실험의 경우 이전의 결과와 비교했을 때 LeakyReLU와 ELU 활성함수를 제외한 모든 함수들은 Batch normalization이 존재했을 때 더 높은 학습 정확도가 나왔다. 여기서 ELU활성함수는 아주 미세한 차이로 Batch normalization이 없는 경우 학습 정확도가 높았다.

결과적으로 Tanh, leaky ReLU, ELU를 제외한 활성 함수들은 성공적으로 레이어가 증가함에 따라 accuracy도 증가하는 것을 보여주고 있다. Batch normalization은 정규화를 통해 학습을 더 빨리 하고 앞선 레이어가 증가함에 따라 local minimum에서 벗어나지 못할 확률을 줄여주는데, 이전에 정규화 과정이 없을 때의 가장 높은 정확도를 비교하였을 경

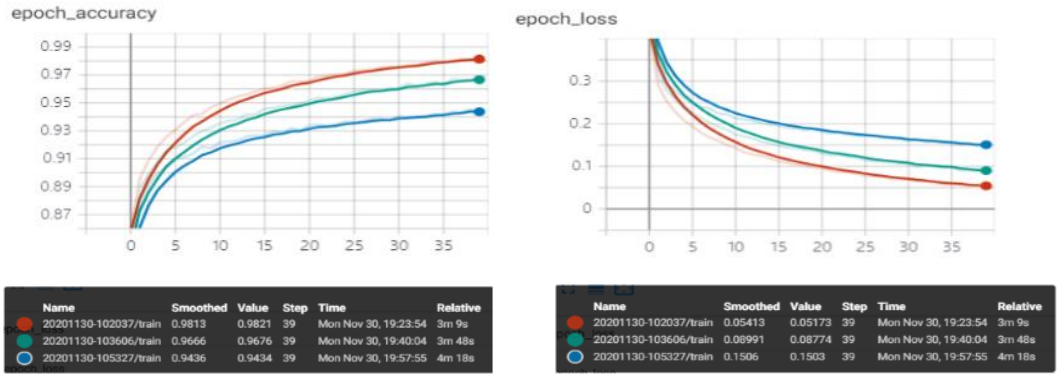
우, ELU를 제외하고는 모두 최고 정확도가 증가함을 볼 수 있었으며, ELU또한 이전의 최고 정확도와 아주 미세한 차이가 났기 때문에 충분한 횟수의 실험을 진행하게 된다면, 실제로는 모든 활성함수가 정규화 이후 accuracy가 증가할 것이라 예상한다.

3-6. 레이어 개수 변화 및, Batch normalization/Drop out 테스트

앞의 실험과 같이 Layer수를 변화를 주어 실험하는데 이번에는 Batch normalization과 Dropout을 포함하고 Hidden layer의 개수를 1개, 2개, 3개로 변화를 주며 실험을 하였다.

(sigmoid 예)

sigmoid	layer	maxpool	train50000	test6000	batch있음	드롭아웃있음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.905833304	202.5909991	0.906666696	198.2970502	0.904833317	197.1760683	0.905777772	199.3547059
2	0.919833362	234.6903243	0.917999983	234.9220512	0.922999978	237.677536	0.920277774	235.7633038
3	0.925499976	266.8513412	0.922833323	269.9921107	0.926833332	268.3070052	0.925055544	268.3834857



[부록-2(d)], [부록-3(d)], [부록-4(d)], [부록-5(d)], [부록-6(d)] 참고-위의 결과를 통해 6개의 활성함수 공통적으로 Batch normalization과 Dropout가 존재할 때, 모든 활성 함수가 정확도가 layer가 증가함에 따라 상승하고, 정규화만 진행되었을 때보다 최고 accuracy 또한 증가함을 볼 수 있었다. 원래 배치 정규화가 존재할 경우 이미 regularization 효과가 있기 때문에 drop out을 굳이 추가할 필요는 없지만, 이러한 규제를 activation 혹은 다른 활동이 일어난 뒤 drop out을 적용할 때 drop out이 accuracy가 상승하는 것에 영향을 미친다는 것을 알 수 있다. 드롭 아웃을 적용하는 방법을 참고한

https://stackoverflow.com/questions/39691902/ordering-of-batch-normalization-and-

dropout#answer-40295999 에서, 실제로 정규화와 drop out을 사용하였을 때, 성능이 개선된 다른 이들의 사례도 존재하였는데, drop out을 적용하게 됨에 따라 성능이 개선되는 장점이 분명 존재하지만, 그만큼 학습 시간이 오래 걸린다는 단점으로 인해, 상황에 따라 유동적인 판단으로 선택해야 할 필요성이 있다.

3-7. 트레인셋 개수 변화 테스트

마지막으로 위의 실험을 바탕으로 Batch normalization과 Dropout이 존재할 때의 Train set의 변화를 통해 학습 정확도를 측정하는 실험을 하였다. 여기서는 Hidden layer 3개와 Fully connected layer 2개에 Batch normalization과 Dropout 모두 존재할 때 Train set 개수를 20000, 40000, 6000개로 조절하여 학습 정확도를 측정하였다.

(sigmoid 예)

sigmoid	layer	maxpool	train변화		batch있음	드롭아웃있음		
train	1차accurac	1차 time	2차accurac	2차 time	3차accurac	3차time	accuracy av	time avg
20000	0.934	454.5294	0.929	450.3651	0.9205	449.3293	0.927833	451.4079
40000	0.927	453.9536	0.92525	453.1699	0.9295	458.5864	0.92725	455.2366
60000	0.917167	455.5461	0.923	452.3802	0.921833	455.1628	0.920667	454.363

[부록-2(e)], [부록-3(e)], [부록-4(e)], [부록-5(e)], [부록-6(e)] 참고-위의 결과로 보면 Sigmoid를 제외한 모든 활성함수 에서 모두 Train set이 증가할 때 학습 정확도 역시 증가하는 것을 확인할 수 있었다. 이는 Train set이 늘어나면서 데이터가 증가하는데 너무 단순한 모델이기 때문에 underfitting 발생하여 낮은 정확도를 가질 것이라는 추측을 해보았다.

이 실험 결과를 통해 가장 높은 정확도는 활성함수 PReLU가 Train set 60000개 존재할 때 93.25%가 나온다. 이는 Tanh 활성함수의 가장 높은 정확도인 91.47%보다 1.78%높은

학습 정확도의 결과를 가졌다.

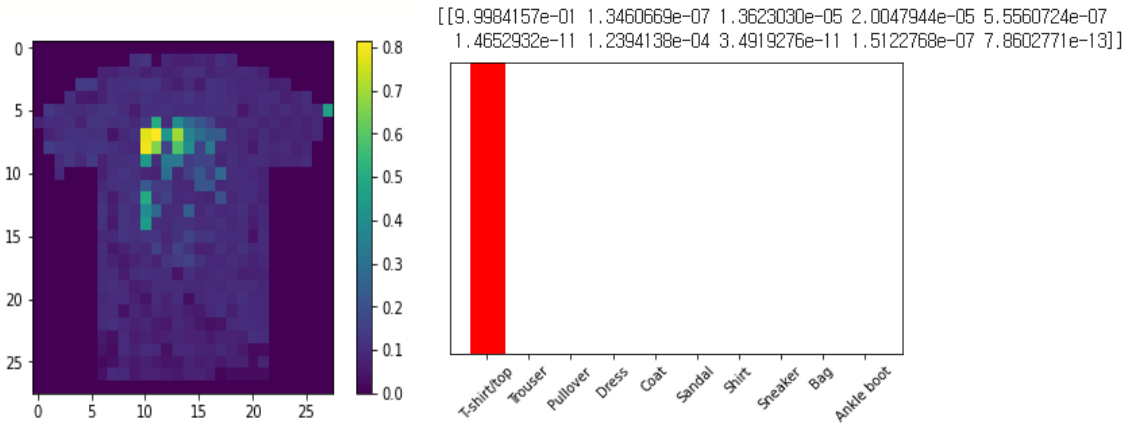
요약하자면 트레인 셋의 개수는 초기 실험에서 진행한 바와 같이, train set의 개수가 늘어남에 따라 accuracy가 증가한다. 그러나 sigmoid에서는 accuracy가 예외적으로 낮아지는데, 이는 sigmoid가 가지고 있는 문제점인 gradient vanishing 문제와 연관이 있다고 추측한다.

3-8. 제작 이미지 삽입 및 테스트

위의 실험 결과를 토대로, 활성함수가 PReLU일 때, train set이 60000개, test set이 5000개, 히든 레이어가 3개, Batch normalization과 Drop out이 존재하는 조건에서 가장 높은 학습 정확도를 보여주었다. 이를 토대로 기존의 코드를 다시 배치하여 Fashion MNIST에서 제공하지 않는 다른 이미지들을 테스트 하여 실제로 이미지를 잘 구분해내는지 알아보도록 한다. 이미지는 클래스에 존재하는 열 가지 종류의 의류에 대하여 각각 하나씩 마련한다.(이미지는 모두 PNG 파일 형식이어야 한다.)

포토샵을 통해 새로 제작한 png 이미지 10장은 다음과 같다.

이미지를 순서대로 테스트를 시도한다. ('T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot' 순서)



matplotlib통해 나타난 그래프를 확인하면 왼쪽 이미지가 t-shirts라는 것을 확신하고 있다. 실제로도 왼쪽의 이미지는 t-shirts이기에 매우 잘 분류된 것을 확인할 수 있다.

[부록-7(b)] 참고- 두 번째 이미지는 trouser이지만 이미지 분류는 traouser가 아닌, 드레스로 구분하는 것을 확인할 수 있다. 작은 확률이지만 티셔트와 수트로도 구분이 되는데, 이는 이미지의 사이즈가 재조정되고 28*28 픽셀화로 변형을 거치면서 이미지를 제대로 인식하지 못한 것으로 추정한다.

[부록-7(c)] 참고- 세 번째 이미지는 pull over이지만, 그래프에 따르면 두 번째 이미지와 같이 드레스로 구분해내는 것을 볼 수 있다. Pull over로 아주 낮은 확률로 추정하고 있지만, dress로 확신하는 것은, 왼쪽 이미지가 기존의 이미지에서 재조정을 거친 후 상하로 길어진 형태로 변형되어 드레스와 같이 기장이 늘어져 이와 같이 분리한 것으로 추정한다.

[부록-7(d)] 참고- 네 번째 이미지는 드레스이다. 전형적인 드레스의 형태를 지닌 만큼, 거의 확실하게 dress로 인식한 것을 알 수 있다.

[부록-7(e)] 참고- 다섯 번째 이미지는 코트이다. 그러나 오른쪽 그래프를 보면, 코트와 셔츠로 애매하게 인식하였고, 그나마 더 높은 확률로 셔츠로 구분해낸 것을 알 수 있다. 이는 코트의 픽셀 형태가 셔츠와 크게 다르지 않은 형태를 가진 것으로 인해 이러한 오류를 발생한 것으로 추정한다.

[부록-7(f)] 참고- 여섯번째 이미지는 sandal이다. 이미지가 픽셀화된 상태지만, sandal 이미지에서 공통적으로 보이는 빈 공간들이 이미지를 거의 완벽하게 sandal로 구분할 수 있는 특징이 되었다고 추정한다.

[부록-7(g)] 참고- 일곱 번째 이미지는 shirt 이다. 그러나 그래프를 확인하면 높은 확률로 coat라고 확신하며 낮은 확률로 shirt를 추정하는 것을 볼 수 있다. 이는 이미지가 다른 이미지들과 같이 28*28 픽셀로 변환되면서 본래 가지고 있던 특징들이 희미해진 이유로 코트의 형태와 유사하게 변형되어 이와 같은 결과가 나타났다고 추측한다.

[부록-7(h)] 참고- 여덟번째 이미지는 sneaker이다. 그러나 이미지가 sandal로 확신하는 결과가 나타나는 것을 볼 수 있다. 이는, sneaker와 sandal이 서로 신발 형태의 유사한 특징이 있기 때문에 분류가 잘못 이루어진 것으로 추정한다.

[부록-7(i)] 참고- 아홉번째 이미지는 bag이다. 오른쪽 그래프를 확인하면 거의 확실하게 이미지를 bag로 구분해낸 것을 알 수 있다. 원본 이미지가 전형적인 가방의 형태를 띄고 있고, 픽셀화된 이미지가 원본과 크게 다르지 않아 이와 같이 높은 확률로 인식되었다는 것을 알 수 있다.

[부록-7(j)] 참고- 열 번째 이미지는 ankle boots이다. 그러나 오른쪽 그래프를 확인하면 거의 확실하게 sandal로 구분해낸 것을 알 수 있다. 실제로 sandal과 같이 굽이 높은 특징들이 픽셀화를 거치면서 sandal로 잘못 구분된 것으로 추정한다.

[부록-7(k)] 참고- 위의 이미지들 중, 제대로 분류되지 못한 coat와 shirts, trouser, ankle boots, pull over의 이미지를 특징이 비교적 뚜렷한 이미지들로 재실험을 해본 결과, 다시 위와 같이 높은 확률로 정확히 구분되는 것을 확인할 수 있었다.
위의 결론들을 종합하였을 때, 직접 이미지를 제작하여 테스트를 하였을 때, 몇가지 이미지들이 제대로 인식되지 못하고 엉뚱한 결과를 도출하는 몇 가지 사례가 있었다. 이는, 이미지가 원래부터 28*28로 제작된 이미지가 아닌, 각양각색의 RGB와 사이즈를 가진 이미지들을 강제로 조정되도록 하는 과정에서, 기존의 이미지가 가진 특징들이 퇴화하고 다른 이미지들과 유사한 형태로 변형되는 문제로 인해, 완전히 다른 종류의 물건이 아닌, 유사한 형태의 이미지가 되는 것이 원인이 된 것으로 추측하였다. 예를 들어 ankle boots와 sandal 등의 종류가 서로 유사한 픽셀 이미지를 가지게 되어, 가장 높은 정확도를 가진 조건에서 테스트를 하였음에도 이러한 잘못된 판단을 하는 것으로 결론을 내리게 되었다.

4. 최종 결과

Fashion MNIST 이미지셋을 이용하여 간단한 CNN 코드 구현을 통해 구해낸 가장 좋은 모델의 조건은, Train set 데이터를 60000개 전부 사용하고, test set이 5000개, EPOCH가 40인 활성함수 PReLU에서, 히든 레이어가 3개일 때 배치 정규화와 드롭 아웃이 모두 적용된 상태이다. 이러한 학습이 과적합이 발생하지 않도록 하기 위해서는 데이터를 늘려 다양한 이미지들을 학습할 수 있게 해야 하는데, 기존의 실험에서 진행하였던 데이터 10000개에서 60000개로 증가시켰을 때, 효과적으로 과적합 발생이 줄어든 것을 확인할 수 있었다. 또한, 최종 결과가 ReLU 함수를 보완하기 위해 최근에 나온 PReLU 활성함수가 실제로도 기존의 활성함수보다 accuracy를 개선하는데 기여한다는 것을 증명한다. 이 외에도 배치 정규화에 드롭 아웃 제약 조건이 추가되었을 때 성능이 개선되는 흥미로운 결과를 관찰할 수 있었는데, 기존에 배치 정규화를 사용할 때, 배치 정규화 자체가 제약 조건의 역할을 맡기 때문에 굳이 드롭 아웃을 추가할 필요가 없다고 배웠던 것에 반에, 실제로 드롭 아웃을 추가하였을 때 학습 개선이 일어난 것을 통해 추가적인 제약 조건이 성능 개선에 어느정도 기여를 할 수 있다는 것을 확인할 수 있었다. 그러나, 기본적으로 Fashion MNIST 이미지 데이터를 이용한 CNN이 정확도가 비교적 높기 때문에 실제로 드롭 아웃을 적용하였을 때 그만큼 늘어나는 시간을 감수할 만큼의 학습 개선인지에 관하여 파악하는 것이 매우 중요하다고 할 수 있다.

이러한 일련의 실험을 통해, 다양한 활성 함수의 개선 정도를 파악하고, 모델에 제약 조건을 부여함으로써 accuracy를 상승시킬 수 있는 다양한 조건 들을 알 수 있었다. 추후 조금 더 accuracy의 상승 정도를 명확히 판단할 만한 유의미한 자료를 얻기 위해서는 조금 더 낮은 정확도의 모델에서 실험할 필요성이 있다고 판단한다.

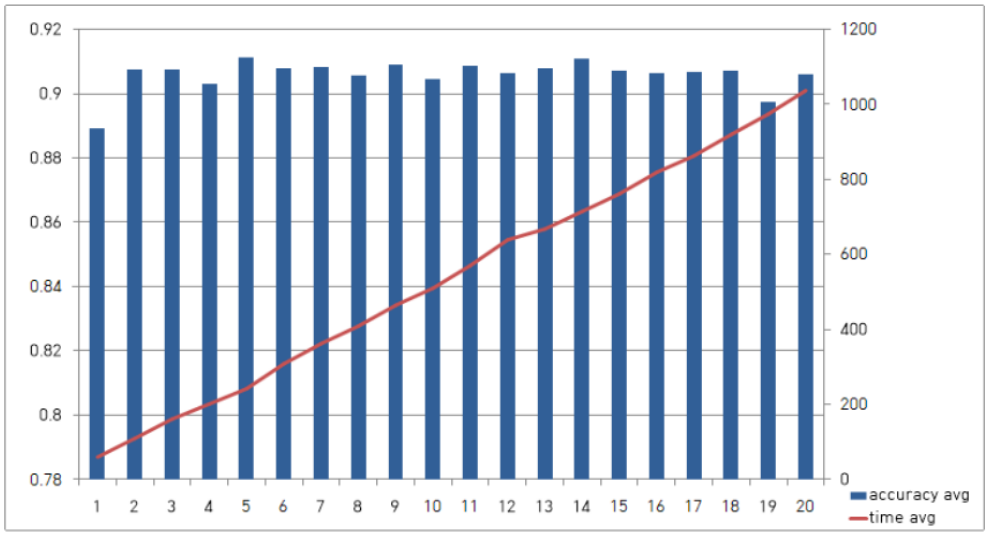
CNN_Fashion_MNIST PReLU Full Code:

https://github.com/Lajancia/CNN_Team_project/blob/master/fashion_mnist_prelu.ipynb

5. 부록

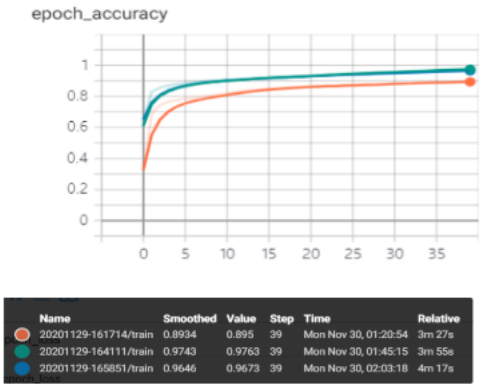
[부록-1(a)]

sigmoid	layer	maxpool	train50000	test6000				
EPOCH	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
10	0.892333329	61.20782375	0.879000008	59.68350625	0.896499991	57.83111572	0.889277776	59.57414858
20	0.909666657	113.7861562	0.907999992	112.2567546	0.904500008	102.3502979	0.907388886	109.4644029
30	0.908333361	166.3065767	0.910833359	163.5652323	0.902999997	153.1558495	0.907388906	161.0092195
40	0.905833304	206.5461817	0.901333332	193.3459761	0.901666641	203.1366601	0.902944426	201.009606
50	0.90716666	239.5398452	0.916333318	230.040158	0.909666657	256.0658813	0.911055545	241.8819615
60	0.912833333	302.5891554	0.905833304	317.7501342	0.904833317	304.5211427	0.907833318	308.2868108
70	0.907000005	348.137006	0.90716666	380.2291691	0.91049999	357.9324167	0.908222218	362.0995306
80	0.907000005	406.2343462	0.90383333	412.400954	0.906166673	408.3345683	0.905666669	408.9899561
90	0.909666657	468.6679296	0.906833351	468.421288	0.91049999	456.254566	0.908999999	464.4479279
100	0.907500029	507.2417336	0.908500016	506.6357734	0.897833347	515.7860708	0.904611131	509.8878593
110	0.909500003	560.3707447	0.907333314	585.3461115	0.908500016	565.0653939	0.908444444	570.2607501
120	0.906499982	648.3249681	0.904166639	646.1189997	0.90866667	622.0476277	0.90644443	638.8305318
130	0.910166681	649.594049	0.909166694	676.7379022	0.904333353	676.0982924	0.907888909	667.4767478
140	0.91233331	725.944581	0.908166647	706.7846985	0.912500024	709.8911984	0.910999993	714.206826
150	0.910666645	745.9428394	0.907500029	757.9677095	0.903333306	779.9836967	0.90716666	761.2980819
160	0.904166639	827.7317629	0.907000005	812.5511642	0.907666683	815.6187325	0.906277776	818.6338865
170	0.904333353	851.67924	0.910666645	867.595221	0.904666662	868.9801774	0.906555553	862.7515461
180	0.907000005	916.3719399	0.90383333	909.9352121	0.911000013	932.4469929	0.907277783	919.584715
190	0.901833355	984.8363471	0.899500012	974.2664955	0.890666664	964.2967818	0.897333344	974.4665414
200	0.90866667	1045.149535	0.905499995	1050.056385	0.903500021	1014.999992	0.905888895	1036.735304



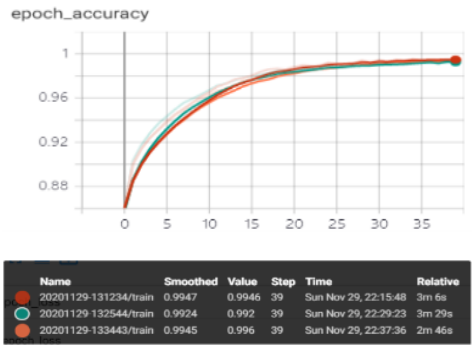
[부록-1(b)]

sigmoid	layer	maxpool	train50000	test6000	batch없음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.870833337	219.9717414	0.882166684	220.2076147	0.883000016	218.8862057	0.878666679	219.6885206
2	0.906833351	252.1462181	0.90533334	246.9339039	0.906666696	243.9699278	0.906277796	247.6833499
3	0.911833346	272.9477696	0.905833304	274.1504169	0.911833346	266.2515218	0.909833332	271.1165694



[부록-1(c)]

sigmoid	layer	maxpool	train50000	test6000	batch있음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.899833322	176.4152796	0.896166682	172.9733632	0.901833355	172.7727463	0.899277786	174.0537964
2	0.911333323	195.5940123	0.913666666	194.1058161	0.909333348	199.302633	0.911444445	196.3341538
3	0.915499985	217.5681834	0.904999971	218.528976	0.917500019	219.574502	0.912666659	218.5572205

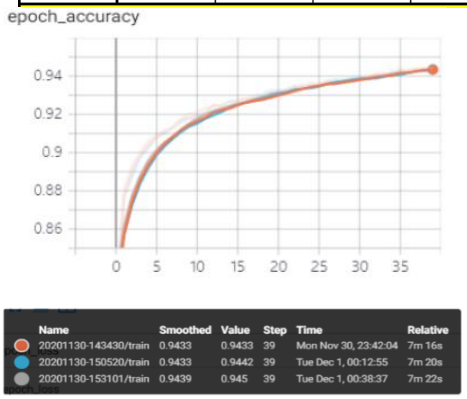


[부록-1(d)]

sigmoid	layer	maxpool	train50000	test6000	batch있음	드롭아웃있음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.905833304	202.5909991	0.906666696	198.2970502	0.904833317	197.1760683	0.905777772	199.3547059
2	0.919833362	234.6903243	0.917999983	234.9220512	0.922999978	237.677536	0.920277774	235.7633038
3	0.925499976	266.8513412	0.922833323	269.9921107	0.926833332	268.3070052	0.925055544	268.3834857

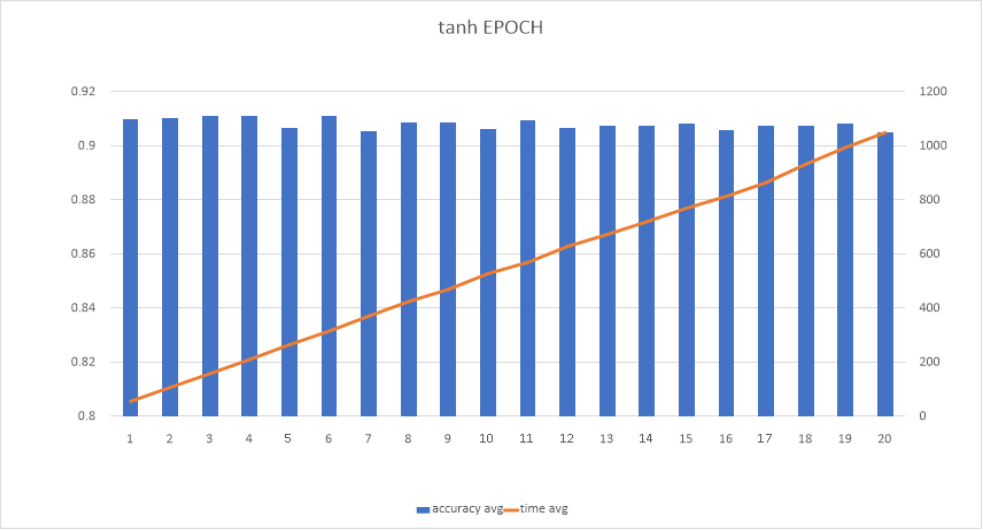
[부록-1(e)]

sigmoid	layer	maxpool	train변화		batch있음	드롭아웃있음		
train	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy av	time avg
20000	0.934	454.5294	0.929	450.3651	0.9205	449.3293	0.927833	451.4079
40000	0.927	453.9536	0.92525	453.1699	0.9295	458.5864	0.92725	455.2366
60000	0.917167	455.5461	0.923	452.3802	0.921833	455.1628	0.920667	454.363



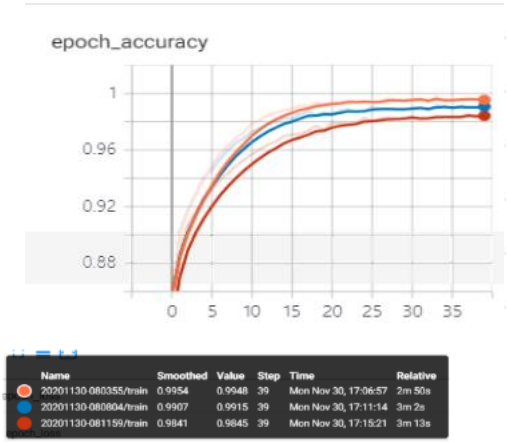
[부록-2(a)]

tanh	layer	maxpool	train50000	test6000				
EPOCH	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
10	0.904999971	56.21707416	0.907999992	55.31940293	0.916333318	54.68173194	0.909777761	55.40606968
20	0.911666691	107.1563909	0.910666645	107.7219594	0.908999979	107.0116003	0.910444438	107.2966502
30	0.910333335	159.4269454	0.912	157.7147505	0.911166668	157.2579565	0.911166668	158.1332175
40	0.911333323	210.4981577	0.906166673	210.6366725	0.91566664	210.6315203	0.911055545	210.5887835
50	0.906000018	263.9886854	0.904666662	264.5478566	0.909666657	266.3033202	0.906777779	264.9466207
60	0.907333314	316.1996982	0.911333323	313.8272488	0.914333344	316.2883005	0.910999993	315.4384158
70	0.9005	370.3485131	0.910166681	371.3707225	0.906000018	371.1009729	0.905555566	370.9400695
80	0.908833325	423.7316396	0.907333314	429.0006189	0.909666657	421.8208048	0.908611099	424.8510211
90	0.90716666	463.4471116	0.910666645	471.2471533	0.907500029	473.643477	0.908444444	469.445914
100	0.905499995	520.5767949	0.903333306	527.4744592	0.910333335	535.8269961	0.906388879	527.9594167
110	0.909166694	565.4088409	0.907833338	574.081737	0.911666691	569.0936222	0.909555574	569.5280667
120	0.906833351	633.5089855	0.904999971	621.1688251	0.907666683	630.8866956	0.906500002	628.5215021
130	0.907666683	668.5269496	0.906833351	666.9464014	0.907999992	682.736047	0.907500009	672.736466
140	0.905666649	722.2728658	0.906166673	700.3166237	0.910000026	738.0583465	0.907277783	720.2159453
150	0.906166673	780.0452251	0.910000026	746.2179198	0.908166647	783.6180081	0.908111115	769.9603844
160	0.904166639	817.0987899	0.907000005	791.9214582	0.906833351	833.1601939	0.905999998	814.0601474
170	0.909333348	830.3371034	0.90716666	869.2282846	0.906000018	894.2962606	0.907500009	864.6205495
180	0.90866667	942.6633692	0.909333348	918.6120539	0.904500008	936.5171244	0.907500009	932.5975158
190	0.908333361	970.9856915	0.902499974	1028.553517	0.913666666	984.1599236	0.908166667	994.5663772
200	0.906666696	1025.192436	0.903166652	1079.536434	0.904666662	1044.447764	0.904833337	1049.725545



[부록-2(b)]

tanh	layer	maxpool	train50000	test6000	batch없음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.900333345	182.5619948	0.900333345	181.8795145	0.898166656	160.2833352	0.899611115	174.9082815
2	0.911166668	197.6525705	0.909500003	190.3834434	0.911666691	179.4127071	0.910777787	189.1495736
3	0.907500029	212.5038002	0.905166686	201.5719576	0.911166668	200.2766631	0.907944461	204.7841403

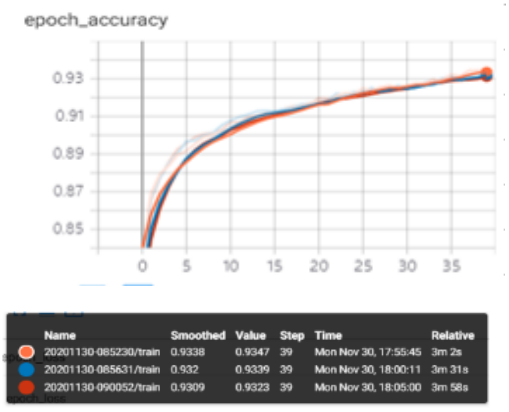


[부록-2(c)]

tanh	layer	maxpool	train50000	test6000	batch있음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.89533335	191.3610418	0.89200002	173.0094173	0.898333311	168.6662595	0.895222227	177.6789062
2	0.90716666	201.0369432	0.914166689	196.2696462	0.916166663	216.1223731	0.912500004	204.4763208
3	0.898833334	227.2148664	0.913999975	216.7166579	0.91383332	241.6424758	0.908888876	228.5246667

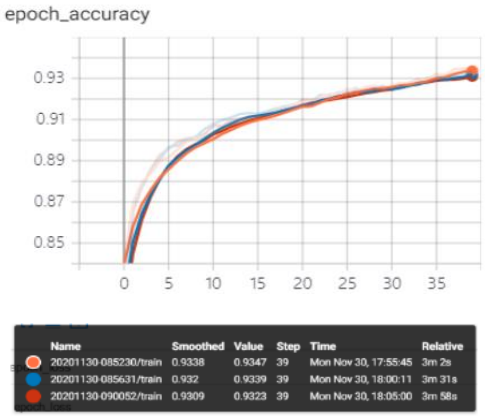
[부록-2(d)]

tanh	layer	maxpool	train50000	test6000	batch있음	드롭아웃있음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.893000007	172.4081604	0.888999999	170.4122474	0.88683331	194.6777074	0.889611105	179.1660384
2	0.912	202.0188944	0.912666678	207.7059431	0.911166668	219.64607	0.911944449	209.7903025
3	0.919166684	237.7698503	0.918833315	230.2510078	0.921000004	247.9076235	0.919666668	238.6428272



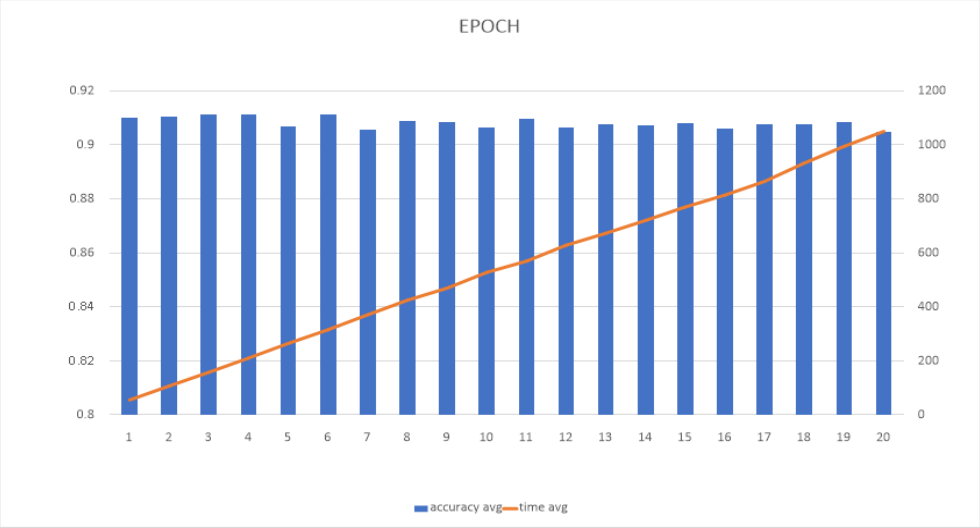
[부록-2(e)]

tanh	layer	maxpool	train 변화	test6000	batch있음	드롭아웃있음		
train	1차accurac	1차 time	2차accurac	2차 time	3차accurac	3차time	accuracy a	time avg
20000	0.894333	88.67242	0.905	87.10181	0.90675	89.38545	0.902028	88.38656
40000	0.909833	168.6541	0.916833	165.8686	0.9115	167.2729	0.912722	167.2652
60000	0.924833	250.7322	0.9065	255.4278	0.912833	259.2027	0.914722	255.1209



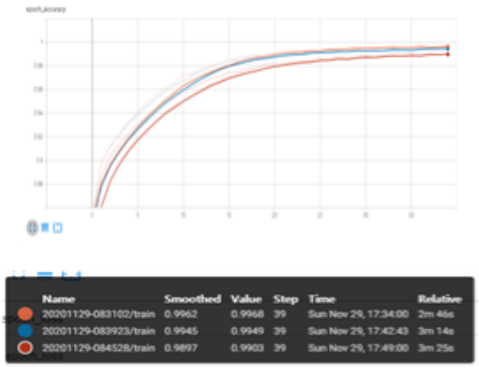
[부록-3(a)]

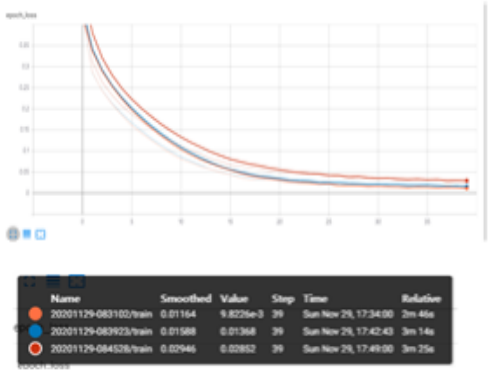
RELU	layer	maxpool	train50000	test6000				
EPOCH	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
10	0.910833359	60.49922276	0.905666649	52.23342967	0.916499972	63.39761615	0.910999993	58.71008952
20	0.913166642	109.2987151	0.915000021	104.8618662	0.916999996	108.7217064	0.915055553	107.6274292
30	0.910833359	164.7613175	0.90866667	159.6220262	0.914666653	159.6667686	0.911388894	161.3500374
40	0.917333305	218.8668537	0.916499972	206.9740913	0.911166668	209.9906454	0.914999982	211.9438635
50	0.909166694	274.6466055	0.904833317	259.1940107	0.911499977	258.602124	0.908499996	264.1475801
60	0.913666666	328.8901062	0.906000018	308.8255119	0.915499985	318.3886504	0.911722223	318.7014229
70	0.916499972	386.7375662	0.911833346	356.9152462	0.914499998	390.7468703	0.914277772	378.1332276
80	0.912	437.1664767	0.914833307	411.0578895	0.910333335	445.4266508	0.912388881	431.2170057
90	0.916166663	498.0675743	0.912500024	463.2885439	0.913999975	510.9585018	0.914222221	490.77154
100	0.91566664	554.694613	0.913166642	513.9473233	0.913166642	566.5737832	0.913999975	545.0719065
110	0.907000005	607.517143	0.912833333	564.5203836	0.915499985	626.7883089	0.911777774	599.6086118
120	0.911499977	663.7683506	0.914666653	610.9739428	0.914499998	673.2660515	0.913555543	649.336115
130	0.912500024	719.4037185	0.916833341	658.1180346	0.911000013	731.7192643	0.913444459	703.0803391
140	0.908833325	777.1659555	0.912166655	737.1940641	0.912833333	779.7675962	0.911277771	764.7092053
150	0.907999992	829.8462043	0.909166694	800.14184	0.908333361	827.9223518	0.908500016	819.3034654
160	0.917999983	888.9265251	0.911833346	837.914149	0.908833325	882.0783303	0.912888885	869.6396681
170	0.914166689	944.8171675	0.914833307	900.7830725	0.914166689	915.9219151	0.914388895	920.507385
180	0.91383332	1002.861308	0.911166668	955.9366231	0.903333306	962.1979835	0.909444431	973.6653049
190	0.907833338	1062.152446	0.913166642	1008.081939	0.910166681	1017.741469	0.910388887	1029.325285
200	0.914333344	1104.691537	0.911833346	1076.213351	0.916000009	1105.846757	0.914055566	1095.583882



[부록-3(b)]

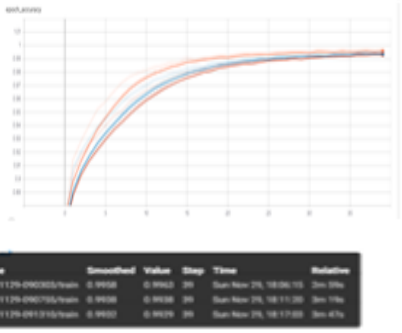
RELU	layer	maxpool	train변화	test6000	batch없음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.909833312	187.2285564	0.911499977	177.3008239	0.913166642	178.036869	0.911499977	180.8554165
2	0.916833341	193.8276999	0.914666653	193.6907754	0.915833354	199.8227186	0.915777783	195.780398
3	0.914833307	205.4265456	0.909333348	210.64301	0.910833359	211.5939782	0.911666671	209.2211791

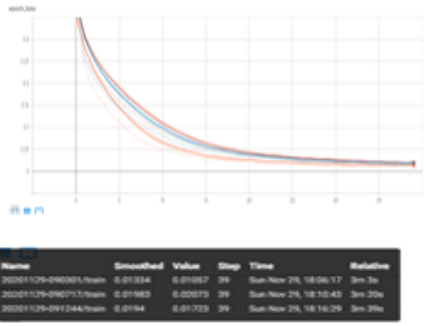




[부록-3(c)]

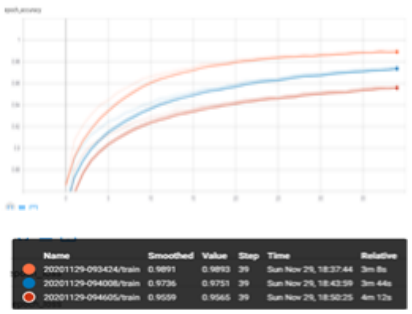
RELU	layer	maxpool	train변화	test6000	batch있음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.905666649	171.7401495	0.901499987	195.2337041	0.903166652	191.8383126	0.903444429	186.2707221
2	0.91383332	188.2385666	0.913666666	206.6266732	0.921166658	204.9440415	0.916222215	199.9364271
3	0.916499972	236.6348417	0.917500019	225.981596	0.917500019	233.5927405	0.91716667	232.0697259

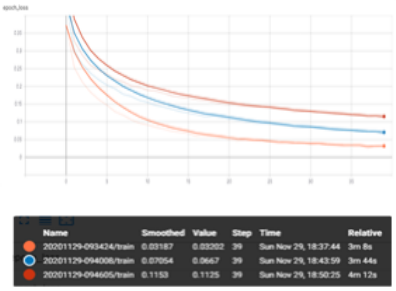




[부록-3(d)]

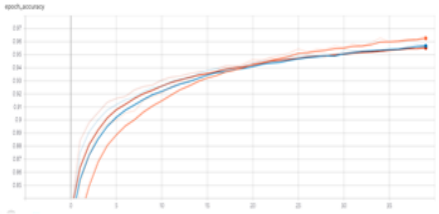
RELU	layer	maxpool	train변화	test6000	batch있음	드롭아웃있음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.905666649	193.1989725	0.90866667	184.2896819	0.913333356	200.3696055	0.909222225	192.61942
2	0.917666674	218.8450191	0.921000004	214.2235618	0.914166689	230.5480232	0.917611122	221.2055347
3	0.932666659	253.6602619	0.926833332	241.4636788	0.927333355	259.9120708	0.928944449	251.6786705

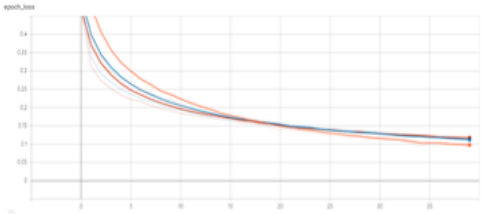


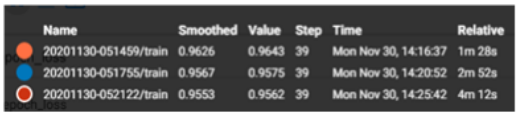


[부록-3(e)]

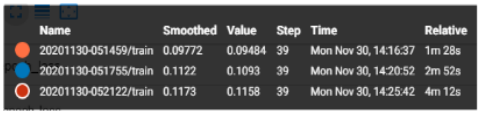
RELU	layer	maxpool	train변화	test6000	batch있음	드롭아웃있음		
train	1차accurac	1차 time	2차accurac	2차 time	3차accurac	3차time	accuracy a	time avg
20000	0.9095	90.19752	0.903833	96.51476	0.917667	98.31813	0.910333	95.01013
40000	0.924667	168.2491	0.925333	177.2211	0.924333	177.4039	0.924778	174.2914
60000	0.922667	252.2187	0.9325	264.7	0.9315	259.4197	0.928889	258.7794







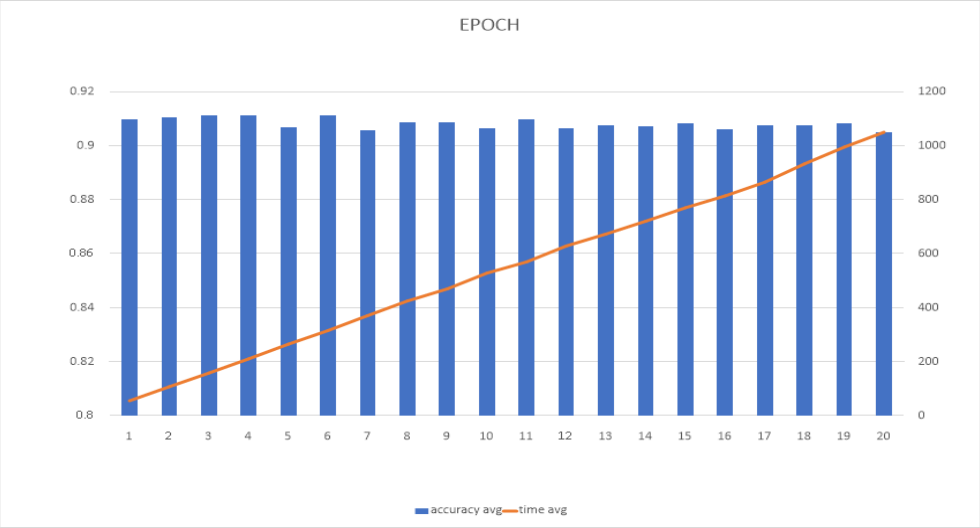
Name	Smoothed	Value	Step	Time	Relative
20201130-051459/train	0.9626	0.9643	39	Mon Nov 30, 14:16:37	1m 28s
20201130-051755/train	0.9567	0.9575	39	Mon Nov 30, 14:20:52	2m 52s
20201130-052122/train	0.9553	0.9562	39	Mon Nov 30, 14:25:42	4m 12s



Name	Smoothed	Value	Step	Time	Relative
20201130-051459/train	0.09772	0.09484	39	Mon Nov 30, 14:16:37	1m 28s
20201130-051755/train	0.1122	0.1093	39	Mon Nov 30, 14:20:52	2m 52s
20201130-052122/train	0.1173	0.1158	39	Mon Nov 30, 14:25:42	4m 12s

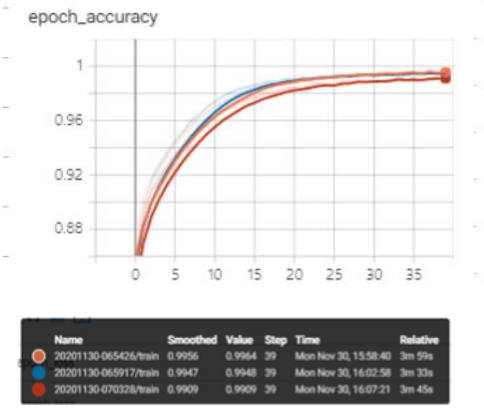
[부록-4(a)]

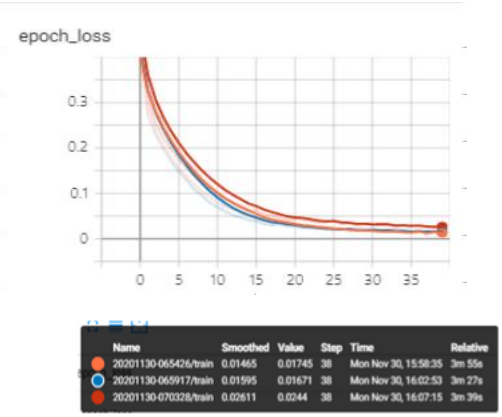
leakyReLU	layer	maxpool	train50000	test6000				
EPOCH	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
10	0.920000017	57.88982606	0.913333356	58.21473098	0.914333344	57.72795844	0.915888906	57.94417183
20	0.906166673	113.6511714	0.912	111.528986	0.916499972	113.0131018	0.911555549	112.7310864
30	0.91383332	164.887923	0.911000013	166.6741319	0.910833359	166.0827911	0.911888897	165.8816153
40	0.913500011	221.1174793	0.915333331	218.1831279	0.91566664	217.4160914	0.914833327	218.9055662
50	0.907500029	270.2617657	0.913166642	270.1429787	0.911333323	267.6081049	0.910666664	269.3376164
60	0.910166681	322.6142504	0.915833354	325.2658098	0.917500019	321.8072135	0.914500018	323.2290912
70	0.912833333	381.7761567	0.914833307	372.708837	0.916833341	374.4294908	0.914833327	376.3048282
80	0.914833307	419.7085102	0.912166655	429.3951874	0.913666666	412.8436484	0.913555543	420.6491153
90	0.911499977	484.7885849	0.911833346	470.9384611	0.909833312	482.4963007	0.911055545	479.4077822
100	0.907500029	521.5303438	0.913999975	544.8698177	0.908500016	522.6815894	0.910000006	529.693917
110	0.916833341	595.9848759	0.910000026	572.2823303	0.909166694	593.1403697	0.91200002	587.1358586
120	0.910166681	630.0370917	0.913333356	668.406621	0.914666653	624.595794	0.91272223	641.0131689
130	0.91383332	714.5256724	0.904999971	675.2788608	0.91233331	711.1064968	0.910388867	700.3036767
140	0.914666653	730.717521	0.909166694	765.5356488	0.911666691	735.2673113	0.911833346	743.8401604
150	0.915000021	879.4207571	0.914499998	814.5958853	0.90383333	778.7926414	0.911111116	824.2697612
160	0.907500029	928.5373144	0.911499977	852.1692598	0.914833307	829.8172565	0.911277771	870.1746102
170	0.913500011	990.1845169	0.911000013	904.9913576	0.90866667	882.2060153	0.911055565	925.7939633
180	0.908333361	1053.357023	0.91233331	975.5696831	0.911333323	934.6500034	0.910666664	987.8589031
190	0.910333335	1031.596457	0.916333318	1011.979819	0.915499985	1106.211356	0.914055546	1049.929211
200	0.910333335	1077.704898	0.91049999	1065.309868	0.910666645	1173.25065	0.91049999	1105.421805



[부록-4(b)]

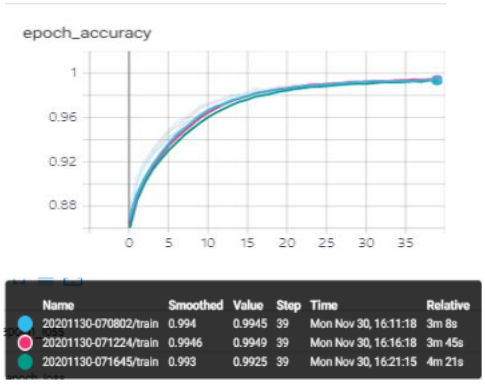
leakyReLU	layer	maxpool	train50000	test6000	batch없음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.899666667	183.7980754	0.906666696	253.1187198	0.907833338	175.172061	0.904722234	204.0296187
2	0.914666653	195.3425901	0.91233331	221.5258574	0.920166671	193.9531608	0.915722211	203.6072028
3	0.910833359	210.7967892	0.912666678	232.8327749	0.912166655	214.4542727	0.911888897	219.3612789





[부록-4(c)]

leakyReLU	layer	maxpool	train50000	test6000	batch있음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.90383333	184.1003809	0.907500029	195.640944	0.9005	182.0899997	0.903944453	187.2771082
2	0.907500029	208.896348	0.912999988	233.7961767	0.90716666	200.2568901	0.909222225	214.3164716
3	0.898000002	230.2937438	0.909833312	269.8235049	0.914833307	237.9177554	0.90755554	246.011668

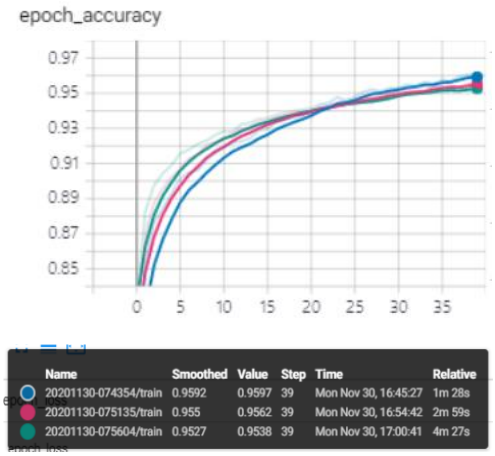


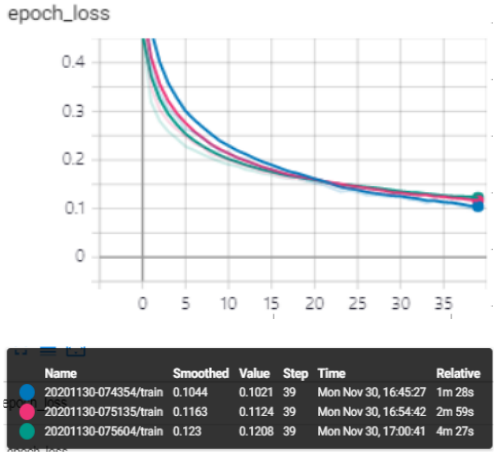
[부록-4(d)]

leakyReLU	layer	maxpool	train50000	test6000	batch있음	드롭아웃있음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.906833351	187.7894483	0.909166694	212.698797	0.906000018	181.3128107	0.907333354	193.9336853
2	0.91049999	222.2392678	0.921166658	266.6902835	0.92233336	217.1466188	0.918000003	235.3587234
3	0.919666648	249.2517245	0.927833319	284.7389338	0.928666651	256.4086061	0.925388873	263.4664214

[부록-4(e)]

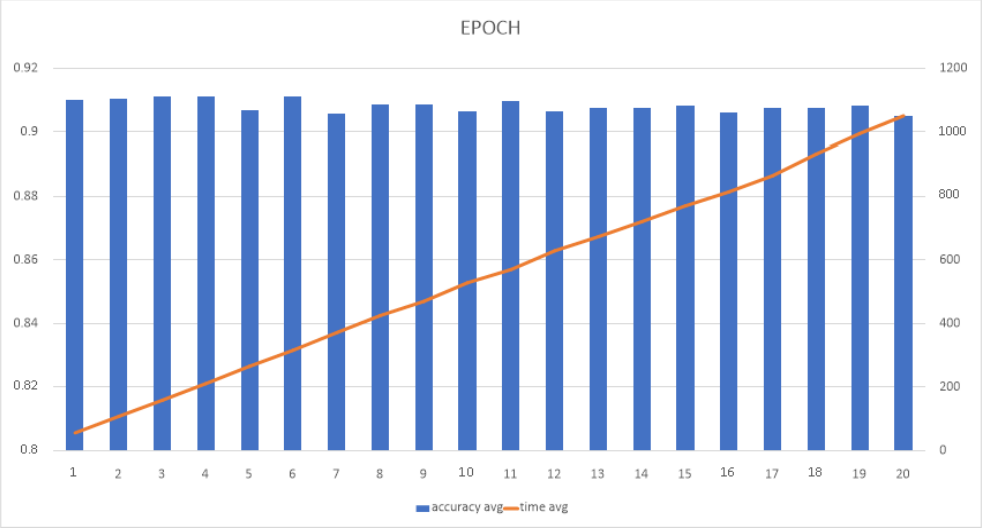
leakyReLU	layer	maxpool	train 변화	test6000	batch있음	드롭아웃있음		
train	1차accurac	1차 time	2차accurac	2차 time	3차accurac	3차time	accuracy a	time avg
20000	0.9115	98.94679	0.908333	93.51427	0.911667	94.27001	0.9105	95.57703
40000	0.920333	186.2007	0.917333	187.3482	0.921	183.6772	0.919556	185.742
60000	0.9255	277.0597	0.924667	273.9588	0.919167	272.8776	0.923111	274.632





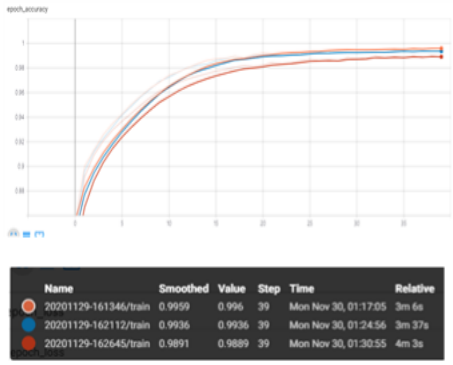
[부록-5(a)]

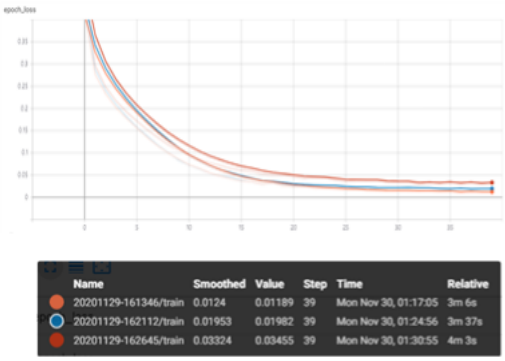
PRELU	layer	maxpool	train50000	test6000				
EPOCH	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
10	0.92250001	63.70675516	0.91983336	77.0286996	0.919333339	71.47853184	0.920555572	70.73799555
20	0.91033334	129.456209	0.91633332	117.780685	0.91250002	141.980946	0.913055559	129.7392801
30	0.9095	179.553075	0.91283333	211.117767	0.91799998	190.421593	0.91344444	193.6974782
40	0.91850001	243.127373	0.91633332	284.020465	0.91533333	247.831999	0.916722218	258.3266125
50	0.91600001	310.841042	0.91533333	352.892424	0.90766668	299.058349	0.913000007	320.930605
60	0.91850001	352.985499	0.90983331	419.065042	0.91616666	306.387405	0.914833327	359.4793152
70	0.91350001	407.075154	0.90866667	485.043545	0.91383332	420.088599	0.912	437.4024325
80	0.91116667	462.188957	0.91466665	558.1229312	0.9173333	490.335111	0.914388875	503.5489996
90	0.91833335	545.74474	0.91016668	638.0340993	0.91500002	537.458394	0.914500018	573.7457445
100	0.91616666	610.025315	0.912666678	707.941708	0.91783333	586.548755	0.915555557	634.8385928
110	0.91549999	683.982599	0.91850001	776.346946	0.90983331	676.552563	0.914611101	712.2940358
120	0.91600001	741.796376	0.91149998	847.612541	0.91666669	750.975645	0.914722224	780.1281875
130	0.91266668	808.919299	0.91600001	915.576398	0.91549999	809.655194	0.914722224	844.7169638
140	0.91883332	874.966582	0.91850001	993.516478	0.91183335	859.601019	0.916388889	909.3613596
150	0.91500002	904.905182	0.9145	1071.60213	0.91333336	875.929382	0.914277792	950.8122312
160	0.91716665	972.741427	0.91333336	1148.32204	0.91250002	942.362094	0.914333344	1021.141853
170	0.91750002	987.375755	0.91266668	1216.53809	0.912	1070.15087	0.914055566	1091.354907
180	0.9095	1057.53384	0.917	1292.86199	0.91383332	1093.64828	0.91344444	1148.014702
190	0.91433334	1093.35299	0.91516668	1359.11723	0.91383332	1150.96933	0.914444447	1201.146518
200	0.91783333	1145.74867	0.91799998	1433.26007	0.91633332	1162.20573	0.917388876	1247.07149



[부록-5(b)]

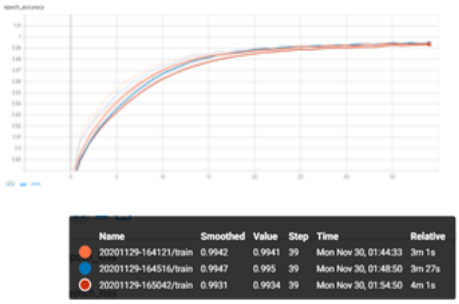
PRELU	layer	maxpool	train50000	test6000	batch없음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.9121667	205.74432	0.9095	209.22909	0.9061667	198.5694218	0.909277777	204.5142767
2	0.9173333	236.55667	0.917666674	237.82662	0.9145	223.99293	0.916499992	232.7920753
3	0.9188333	265.92957	0.906666696	271.80538	0.9183334	250.28852	0.914611121	262.6744893

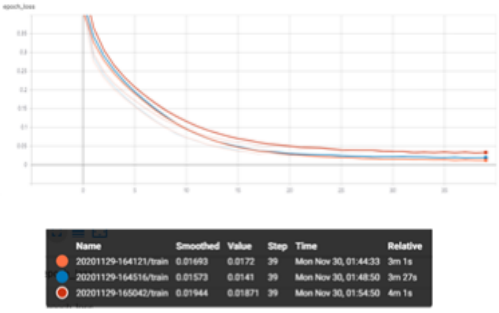




[부록-5(c)]

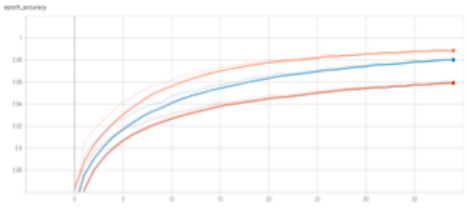
PRELU	layer	maxpool	train50000	test6000	batch있음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.90850002	190.584032	0.91233331	214.163005	0.90499997	192.532322	0.908611099	199.0931197
2	0.91666669	220.894776	0.91416669	258.112709	0.91716665	213.671848	0.916000009	230.8931107
3	0.91799998	263.697109	0.92066669	294.955706	0.91616666	248.310237	0.91827778	268.9876842

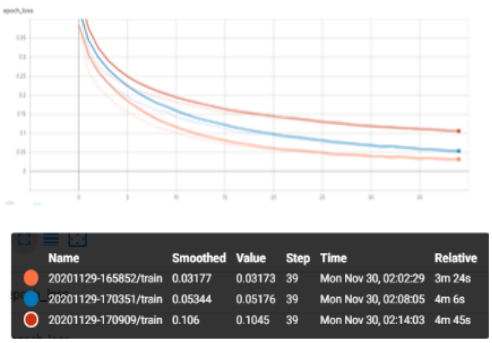




[부록-5(d)]

PRELU	layer	maxpool	train50000	test6000	batch있음	드롭아웃있음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.916499972	194.7200453	0.912666678	230.9742632	0.905833304	217.2824926	0.911666652	214.3256004
2	0.928833306	230.076618	0.926333308	268.6403694	0.92566669	253.8750658	0.926944435	250.8640177
3	0.928166687	270.5085459	0.929499984	314.448694	0.923833311	293.9233253	0.927166661	292.9601884



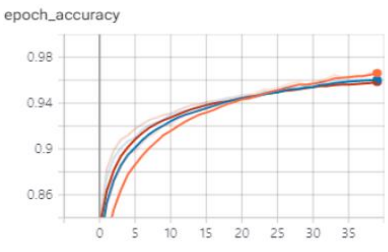


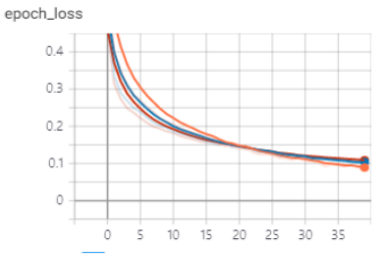
Name	Smoothed	Value	Step	Time	Relative
20201129-165852/train	0.9885	0.9884	39	Mon Nov 30, 02:02:29	3m 24s
20201129-170351/train	0.9801	0.9805	39	Mon Nov 30, 02:08:05	4m 6s
20201129-170909/train	0.9592	0.9597	39	Mon Nov 30, 02:14:03	4m 45s

Name	Smoothed	Value	Step	Time	Relative
20201129-165852/train	0.03177	0.03173	39	Mon Nov 30, 02:02:29	3m 24s
20201129-170351/train	0.05344	0.05176	39	Mon Nov 30, 02:08:05	4m 6s
20201129-170909/train	0.106	0.1045	39	Mon Nov 30, 02:14:03	4m 45s

[부록-5(e)]

PRELU	layer	maxpool	train변화	test6000	batch있음	드롭아웃있음		
train	1차accurac	1차 time	2차accurac	2차 time	3차accurac	3차time	accuracy a	time avg
20000	0.9075	113.67	0.9042	118.35	0.9135	122.1	0.908389	118.0402
40000	0.9142	219.05	0.9258	217.63	0.9245	222.49	0.9215	219.7253
60000	0.935	328.1	0.9317	321.66	0.9308	323.01	0.9325	324.2594



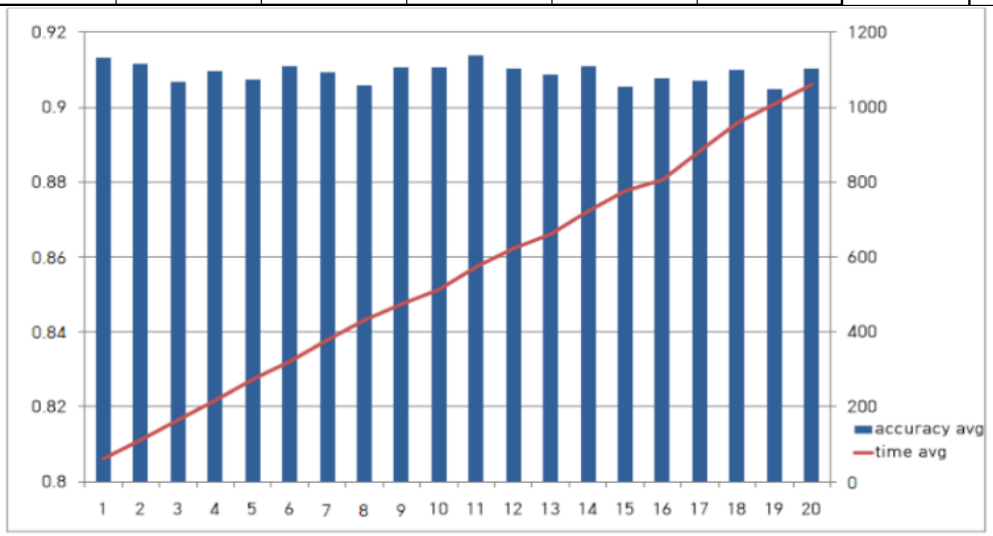


Name	Smoothed	Value	Step	Time	Relative
20201202-182748/train	0.9662	0.9685	39	Thu Dec 3, 03:29:50	1m 51s
20201202-183041/train	0.9602	0.9606	39	Thu Dec 3, 03:34:24	3m 35s
20201202-183456/train	0.9585	0.9601	39	Thu Dec 3, 03:40:19	5m 13s

Name	Smoothed	Value	Step	Time	Relative
20201202-182748/train	0.08916	0.0841	39	Thu Dec 3, 03:29:50	1m 51s
20201202-183041/train	0.1037	0.1034	39	Thu Dec 3, 03:34:24	3m 35s
20201202-183456/train	0.1088	0.1061	39	Thu Dec 3, 03:40:19	5m 13s

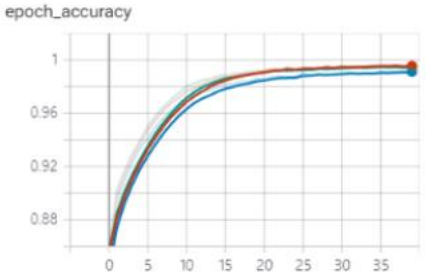
[부록-6(a)]

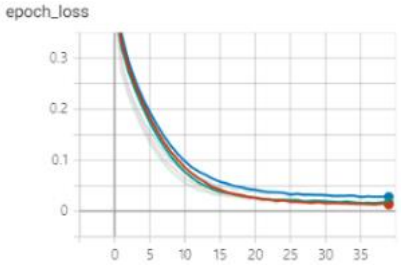
ELU	layer	maxpool	train50000	test6000				
EPOCH	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
10	0.913500011	59.8518827	0.913333356	63.08349204	0.912666678	62.86154056	0.913166682	61.9323051
20	0.91233331	113.1678967	0.90966667	110.548233	0.91233331	112.1281111	0.91144443	111.9480803
30	0.908333344	160.4767525	0.905499977	168.6043584	0.906833351	166.1539657	0.90688889	165.0783589
40	0.90866667	217.7668209	0.911166668	218.4878588	0.908833325	216.5621045	0.909555554	217.6055947
50	0.906499982	279.4729071	0.907000005	277.4612036	0.909166694	263.5168746	0.90755556	273.4836617
60	0.909666657	325.7744675	0.910666649	328.434746	0.912666678	310.3816481	0.910999995	321.5302872
70	0.907999971	385.2205882	0.908999971	381.9389126	0.910833359	368.5422113	0.909277767	378.5672374
80	0.906333327	437.5994766	0.905166686	433.4157128	0.905499995	425.0799959	0.905666669	432.0317284
90	0.909500003	478.8640118	0.912000021	477.9050841	0.910333335	467.8398502	0.91061112	474.8696487
100	0.911000021	518.9108918	0.911000013	513.1063921	0.909500003	508.3097348	0.910500013	513.4423396
110	0.913666691	570.2797806	0.912666657	594.8208387	0.915333331	557.9851687	0.913888893	574.3619293
120	0.907500029	635.6028636	0.912500024	625.2390664	0.911000013	609.4802744	0.910333355	623.4407348
130	0.907666683	668.7595201	0.91049999	657.752629	0.907999992	658.1697993	0.908722222	661.5606495
140	0.910833359	719.6645436	0.909166694	721.1833634	0.912666678	727.3263144	0.91088891	722.7247405
150	0.907333323	787.8124497	0.905500016	789.5335791	0.903666675	754.1757855	0.905500004	777.1739381
160	0.906833351	813.8102827	0.907333306	809.2907264	0.90866667	796.8610344	0.907611109	806.6540145
170	0.908500016	874.107691	0.904166639	927.2508767	0.908166647	849.5070865	0.906944434	883.6218847
180	0.910833359	974.6780443	0.908999979	964.1175048	0.910000026	936.3879809	0.909944455	958.39451
190	0.905833359	1028.356759	0.906833325	1038.182639	0.90200001	961.4645708	0.904888898	1009.334656
200	0.909833317	1097.785554	0.912166655	1085.337994	0.908500016	1000.949508	0.910166663	1061.357685



[부록-6(b)]

ELU	layer	maxpool	train50000	test6000	batch없음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.909666657	176.7816849	0.905499995	166.465796	0.909333348	166.765703	0.908166667	170.0043946
2	0.913166642	183.892657	0.914499998	187.5463305	0.913666666	182.5384066	0.913777769	184.6591314
3	0.904833317	199.9480646	0.906833351	200.7359524	0.911499977	200.6094975	0.907722215	200.4311715



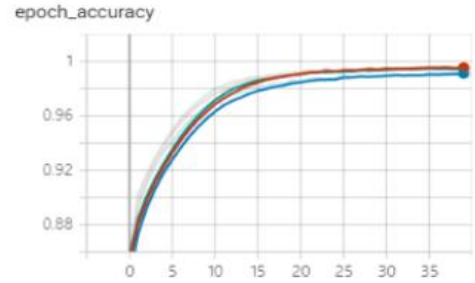


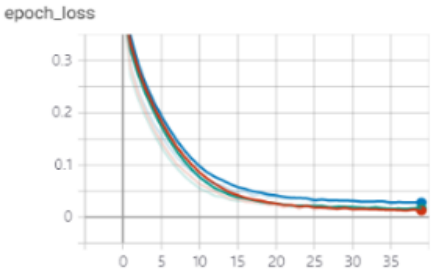
Name	Smoothed	Value	Step	Time	Relative
20201130-144733/train	0.9958	0.9968	39	Mon Nov 30, 23:50:20	2m 40s
20201130-145834/train	0.9945	0.9947	39	Tue Dec 1, 00:01:37	2m 55s
20201130-151222/train	0.991	0.9913	39	Tue Dec 1, 00:15:43	3m 12s

Name	Smoothed	Value	Step	Time	Relative
20201130-144733/train	0.01275	9.8285e-3	39	Mon Nov 30, 23:50:20	2m 40s
20201130-145834/train	0.01736	0.0177	39	Tue Dec 1, 00:01:37	2m 55s
20201130-151222/train	0.02847	0.02829	39	Tue Dec 1, 00:15:43	3m 12s

[부록-6(c)]

ELU	layer	maxpool	train50000	test6000	batch있음	드롭아웃없음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.904333353	226.7840352	0.897499979	221.3625433	0.894500017	219.2581689	0.898777783	222.4682492
2	0.911666691	261.1265352	0.916666687	263.4006147	0.912666678	260.2597296	0.913666685	261.5956265
3	0.919333339	300.8624301	0.904333353	302.3365471	0.902833343	304.1648705	0.908833345	302.4546159





Name	Smoothed	Value	Step	Time	Relative
20201130-144733/train	0.9958	0.9968	39	Mon Nov 30, 23:50:20	2m 40s
20201130-145834/train	0.9945	0.9947	39	Tue Dec 1, 00:01:37	2m 55s
20201130-151222/train	0.991	0.9913	39	Tue Dec 1, 00:15:43	3m 12s

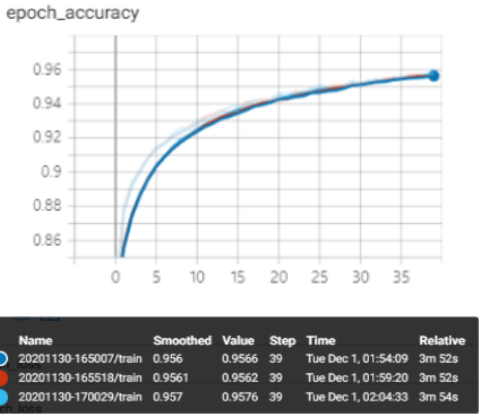
Name	Smoothed	Value	Step	Time	Relative
20201130-144733/train	0.01275	9.8285e-3	39	Mon Nov 30, 23:50:20	2m 40s
20201130-145834/train	0.01736	0.0177	39	Tue Dec 1, 00:01:37	2m 55s
20201130-151222/train	0.02847	0.02829	39	Tue Dec 1, 00:15:43	3m 12s

[부록-6(d)]

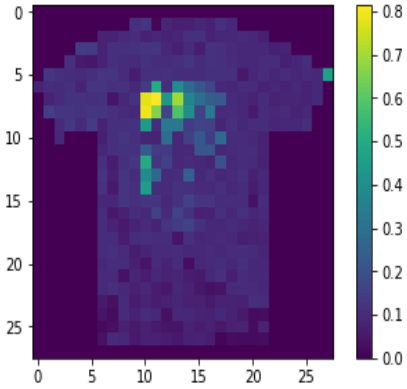
ELU	layer	maxpool	train50000	test6000	batch있음	드롭아웃있음		
layer	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy avg	time avg
1	0.908166647	198.2715209	0.899333358	185.4195232	0.904666662	183.8025653	0.904055556	189.1645365
2	0.919333339	219.9860418	0.924166679	215.4007232	0.926999986	215.6955094	0.923500001	217.0274248
3	0.922166646	240.8465097	0.927666664	244.6136754	0.925000012	248.1791298	0.924944441	244.5464383

[부록-6(e)]

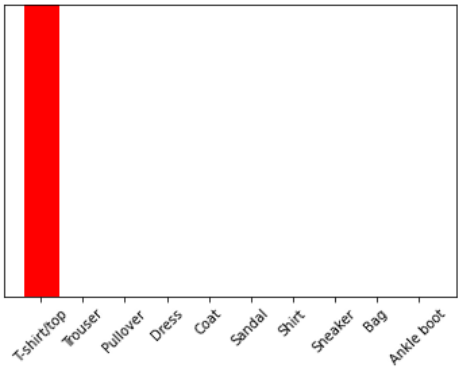
ELU	layer	maxpool	train변화		batch있음	드롭아웃있음		
train	1차accuracy	1차 time	2차accuracy	2차 time	3차accuracy	3차time	accuracy av	time avg
20000	0.922	242.3589	0.921	251.3608	0.9245	253.325	0.9225	249.0149
40000	0.92125	252.8643	0.92575	255.4126	0.92875	254.2758	0.92525	254.1842
60000	0.926667	253.8572	0.9295	255.6888	0.928667	255.8944	0.928278	255.1468



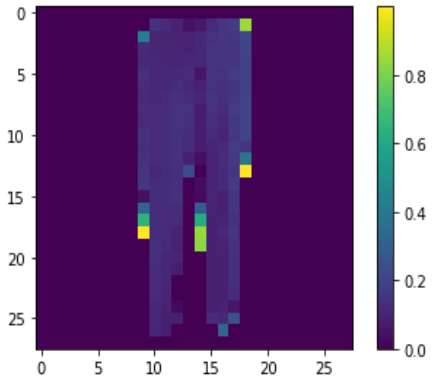
[부록-7(a)]



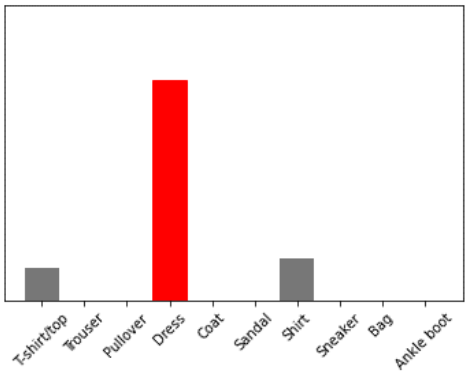
[[9.9984157e-01 1.3460669e-07 1.3623030e-05 2.0047944e-05 5.5560724e-07
1.4652932e-11 1.2394138e-04 3.4919276e-11 1.5122768e-07 7.8602771e-13]]



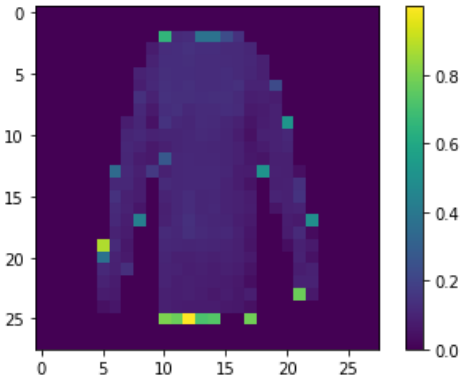
[부록-7(b)]



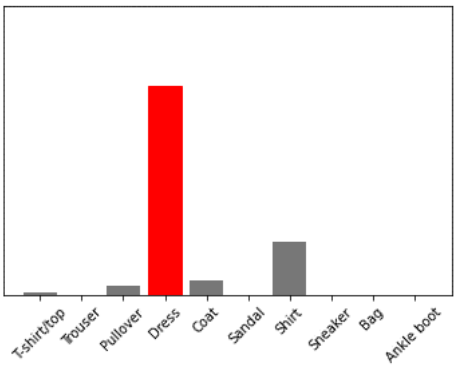
[[1.1029506e-01 1.5263349e-05 1.2644557e-03 7.4368376e-01 1.8295437e-03
7.0041963e-08 1.4291164e-01 5.1493809e-09 2.1071617e-07 2.2741886e-09]]



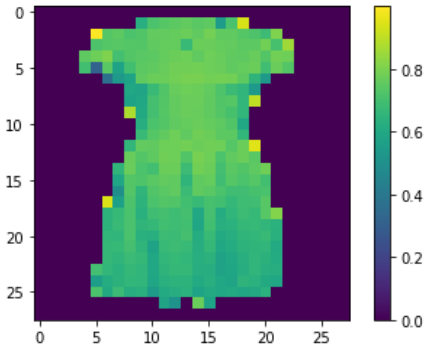
[부록-7(c)]



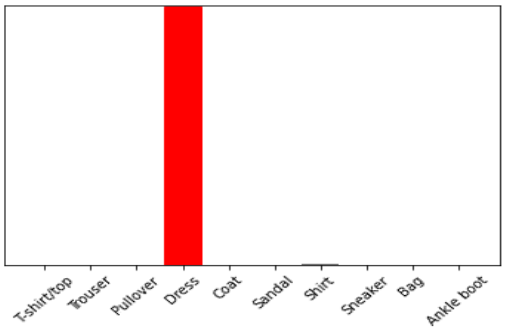

```
[[1.0139654e-02 4.8821444e-06 3.1948134e-02 7.2322524e-01 5.0324984e-02
  1.4155037e-08 1.8435077e-01 2.7456117e-06 3.3546648e-06 1.5061644e-07]]
```



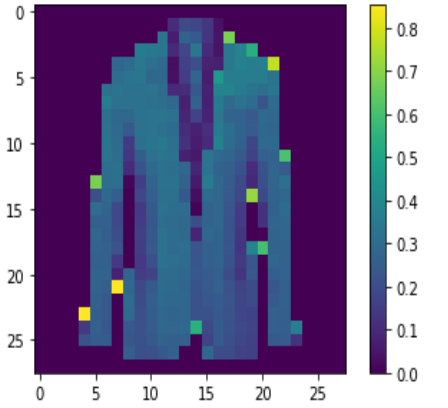
[부록-7(d)]



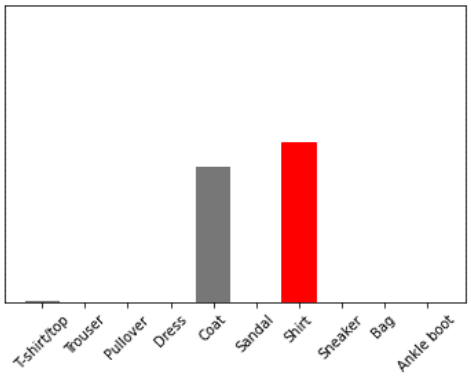

```
[[6.48293004e-04 9.50227331e-06 2.61755488e-08 9.92947459e-01
  8.37188782e-06 5.04101982e-10 6.38494501e-03 1.08496954e-10
  1.46346292e-06 4.16694262e-10]]
```



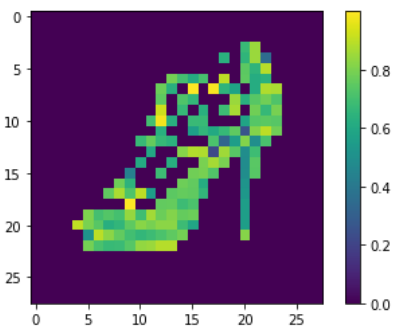
[부록-7(e)]



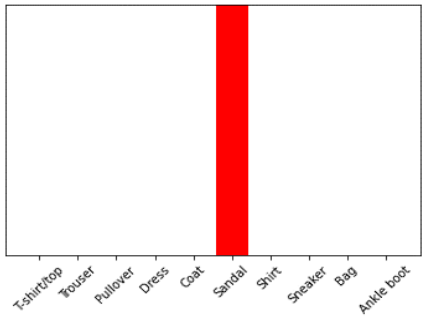

```
[[3.2288076e-03 7.1428605e-09 1.5204267e-03 6.8915127e-05 4.5761117e-01
  7.5003918e-08 5.3756762e-01 2.1811948e-06 7.5576656e-07 7.2814593e-08]]
```



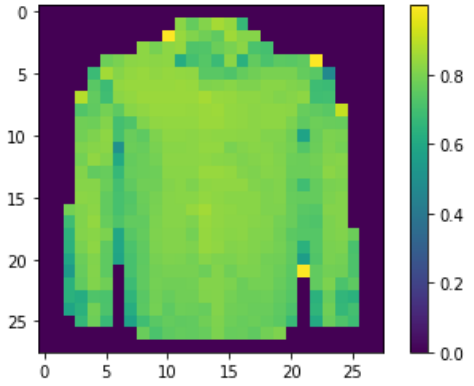
[부록-7(f)]

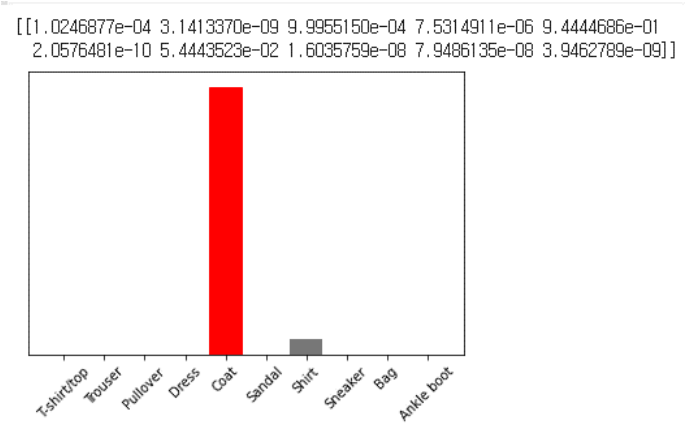



```
[[5.0175481e-07 2.3577045e-11 3.6475354e-11 7.8857871e-09 4.1018633e-11
  9.9999750e-01 1.8559475e-09 7.8132238e-08 3.0639594e-08 1.9303138e-06]]
```

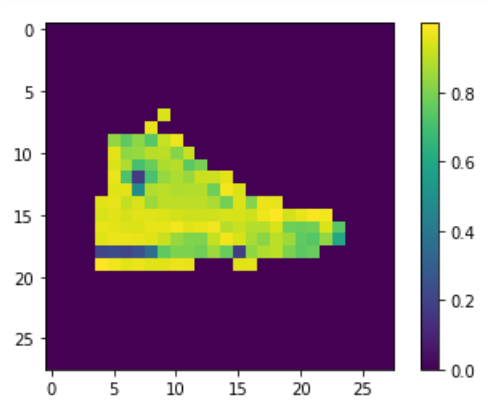


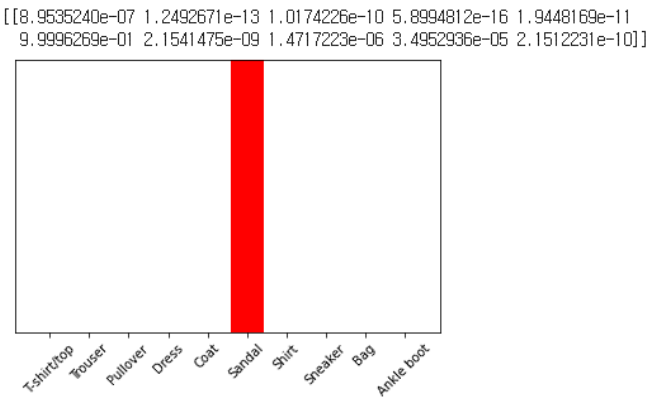
[부록-7(g)]


0
5
10
15
20
25
0.8
0.6
0.4
0.2
0.0

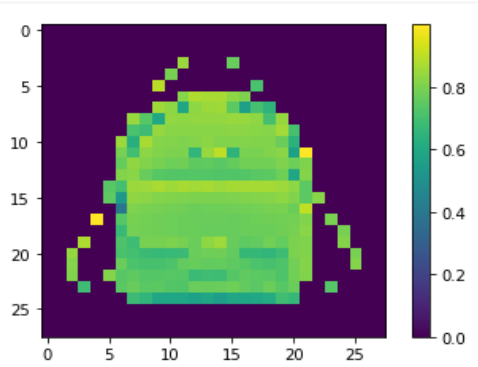

[[1.0246877e-04 3.1413370e-09 9.9955150e-04 7.5314911e-06 9.4444686e-01
2.0576481e-10 5.4443523e-02 1.6035759e-08 7.9486135e-08 3.9462789e-09]]
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot

[부록-7(h)]

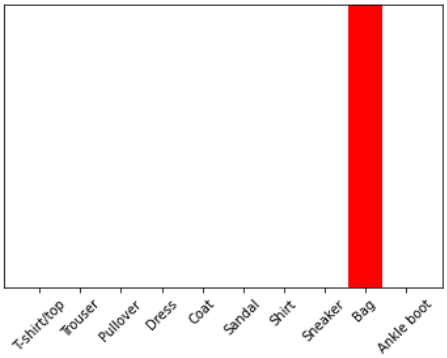

0
5
10
15
20
25
0.8
0.6
0.4
0.2
0.0


[[8.9535240e-07 1.2492671e-13 1.0174226e-10 5.8994812e-16 1.9448169e-11
9.9996269e-01 2.1541475e-09 1.4717223e-06 3.4952936e-05 2.1512231e-10]]
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot

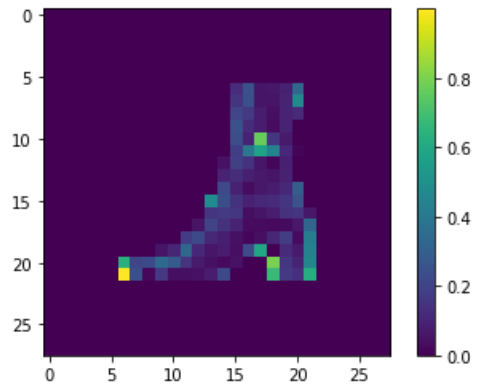
[부록-7(i)]



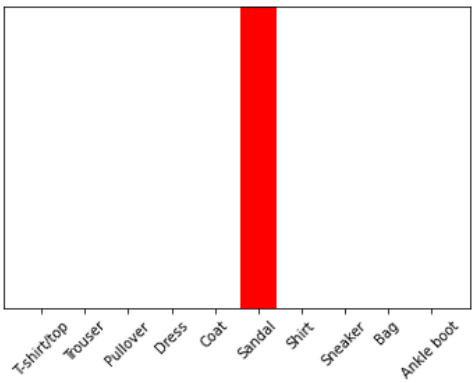

```
[[6.6591369e-08 9.6115734e-15 2.9315912e-09 6.5374223e-10 5.1561357e-05
  4.2770266e-12 2.5493497e-08 1.7010006e-09 9.9994838e-01 1.7345911e-11]]
```



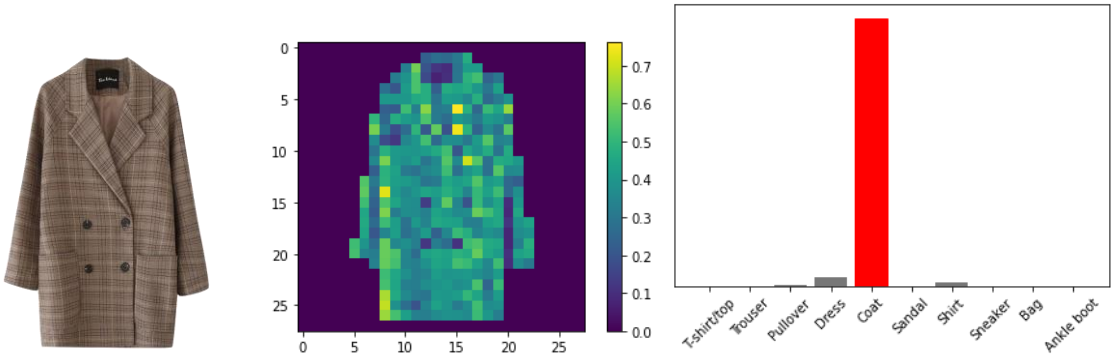
[부록-7(j)]

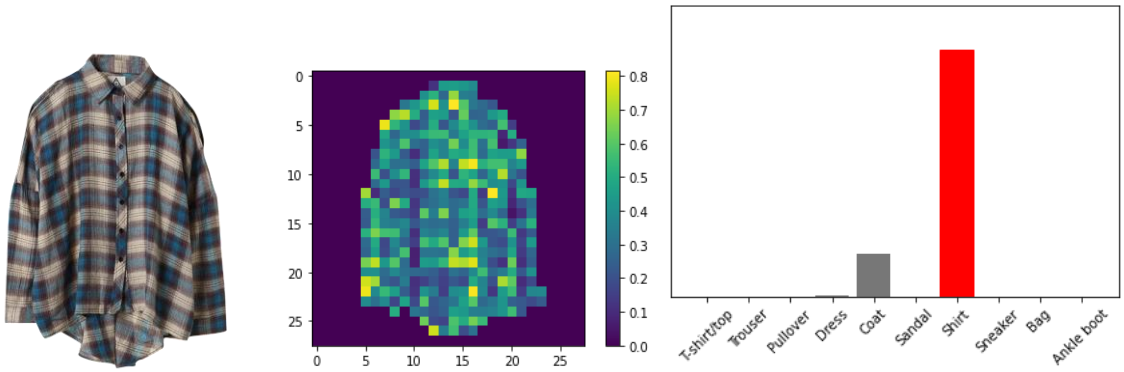


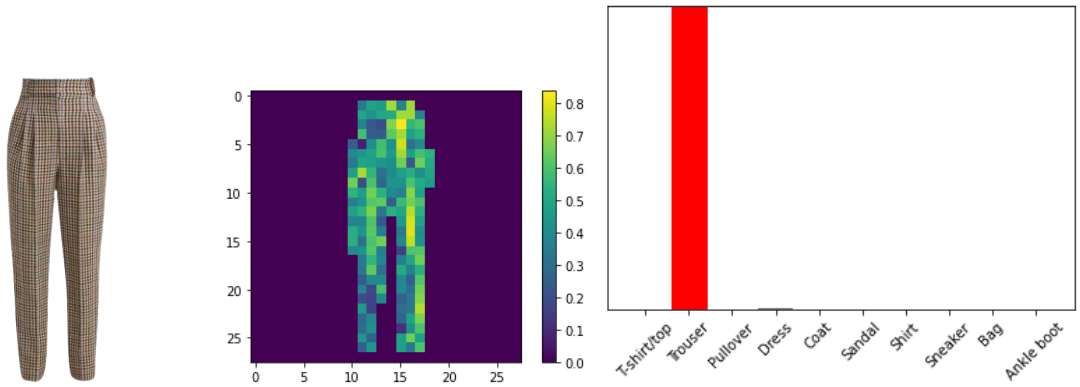

```
[[6.97549316e-04 1.05526254e-07 1.78157370e-05 9.78818548e-07
  2.30685819e-05 9.99106109e-01 7.74130876e-06 1.29518041e-04
  2.31860417e-06 1.47948867e-05]]
```

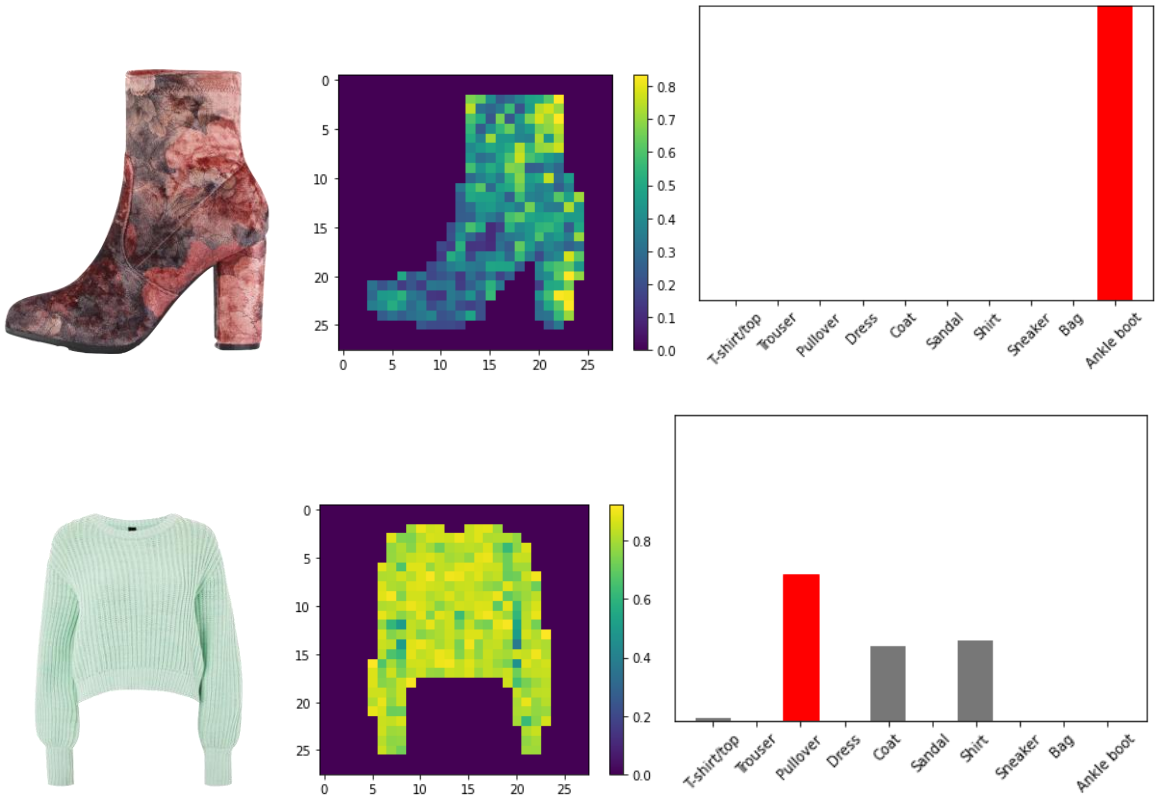


[부록-7(k)]


T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot


T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot


T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot



6. 참고자료

코드 참조:

https://www.tensorflow.org/tutorials/keras/classification?hl=ko

https://blog.naver.com/jeonghj66/222005972393

https://tykimos.github.io/2017/01/27/CNN_Layer_Talk/

https://hackereyes.tistory.com/entry/%EA%B8%B0%EB%B3%B8-keras-%EC%82%AC%EC%9A%A9%EB%B2%95?category=856833

활성함수 참고

https://reniew.github.io/12/

CNN 동작 방식 참고

http://taewan.kim/post/cnn/

https://tykimos.github.io/2017/01/27/CNN_Layer_Talk/

이미지 개수 조절 및 reshape 코드 참고

https://stackoverflow.com/questions/55894691/how-to-change-the-number-of-images-to-use-for-training-and-evaluating-in-tensorf

https://www.tensorflow.org/api_docs/python/tf/reshape

과적합 발생 조건 참고

https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d

이미지 테스트

https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification

https://www.tensorflow.org/tutorials/keras/classification

https://blog.naver.com/jaicoco/221924886395

https://blog.naver.com/idzs_17/222140862694

jpg 이미지

https://www.dior.com/en_us/products/couture-3BO227YTU_H900-ankle-boot-black-polished-calfskin

https://www.today.com/style/why-newscasters-nationwide-wear-same-dress-t113989

https://shopee.ph/Fashion-3way-Mini-Korean-Cute-Sling-Bag-A018-i.17347931.1662000379