**인공지능 HW4** -이수원 교수님-

# 20150439 소혜빈

제출 날짜 : 2019.6.2

## **문제**

1. **Activation Function별 분류 예측 정확도 비교표 (소수 둘째 자리까지, 학습모델은 보조자료 기준으로)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Activation  Score  Class | tanh | | | sigmoid | | | relu | | |
| Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 |
| airplane | 0.45 | 0.42 | 0.43 | 0.40 | 0.45 | 0.42 | 0.56 | 0.47 | 0.51 |
| automobile | 0.60 | 0.58 | 0.59 | 0.42 | 0.24 | 0.30 | 0.55 | 0.52 | 0.53 |
| bird | 0.40 | 0.37 | 0.39 | 0.30 | 0.08 | 0.13 | 0.40 | 0.44 | 0.42 |
| cat | 0.25 | 0.39 | 0.30 | 0.00 | 0.00 | 0.00 | 0.33 | 0.29 | 0.31 |
| deer | 0.34 | 0.26 | 0.29 | 0.19 | 0.23 | 0.21 | 0.37 | 0.28 | 0.32 |
| dog | 0.29 | 0.22 | 0.25 | 0.35 | 0.21 | 0.26 | 0.35 | 0.21 | 0.26 |
| frog | 0.55 | 0.53 | 0.54 | 0.25 | 0.74 | 0.38 | 0.53 | 0.65 | 0.59 |
| horse | 0.56 | 0.47 | 0.51 | 0.31 | 0.25 | 0.27 | 0.54 | 0.60 | 0.58 |
| ship | 0.49 | 0.67 | 0.57 | 0.44 | 0.46 | 0.45 | 0.53 | 0.74 | 0.62 |
| truck | 0.54 | 0.47 | 0.50 | 0.41 | 0.46 | 0.43 | 0.55 | 0.58 | 0.57 |
| 평균 | 0.45 | 0.44 | 0.44 | 0.31 | 0.31 | 0.29 | 0.47 | 0.48 | 0.47 |

1. **①의 결과를 해석한다면 ?**

sigmoid 함수의 결과값은 그 중점이 0이 아니며, 모두 양수 입니다. 이 경우 모수를 추정하는 학습이 어렵다는 단점이 있습니다. 또한, sigmoid함수는 뉴런의 활성화 값이 0 또는 1에 매우 가깝다면 (saturate), 해당 편 미분 값이 0에 매우 가까워지는 특성이 있습니다. back propagation의 경우 chain rule을 이용하는데, 이 과정에서 0에 매우 작은 값이 계속 곱해진다면 그 값은 0으로 점점 수렴하여, back propagation 과정에서 학습의 결과가 전달되지 못하고, 이에 따라 weight 값의 조정이 되지 않습니다. 이런 식으로 weight 값은 초기 값에서 크게 변하지않고, 학습이 되지 않을 것입니다. 이러한 vanishing gradient problem 때문에 sigmoid 함수는 tanh 함수나 relu 함수에 비해 성능이 떨어지는 것으로 알려져 있고, 표에서 나타나는 실제 성능 측정값도 가장 낮은 것을 확인 할 수 있습니다. 특히 cat의 경우에는 학습이 완전히 되지 않은 것으로 보입니다.

tanh 함수는 sigmoid 처럼 비선형 함수이지만, 결과 값의 범위가 -1부터 1이기 때문에 sigmoid와 달리 중심 값이 0입니다. 따라서 sigmoid 보다 optimization이 빠르다는 장점이 있지만, vanishing gradient problem이 발생합니다.

relu (Rectified Linear Unit) 은 선형그래프를 한번 꺾은 형태로 위 두 함수가 갖고있던 vanishing gradient problem 을 해결했고, 위 두함수에 비해 converge 되는 속도가 빠릅니다. 하지만 dying relu problem이 발생합니다. 만약 학습 과정에서 weight가 특정 뉴런이 activate되지 않도록 바뀐다면, 해당 뉴런을 지나는 gradient도 0이 됩니다. 따라서 training 과정에서 해당 뉴런이 한번도 activate되지 않을 수 있고, 심한 경우 network 전에 뉴런의 40%가 죽어 있는 경우도 발생한다고 합니다. (출처 : <http://cs231n.github.io/neural-networks-1/>) 이것을 막기 위해서는 learning rate를 크지 않게 조절하는 것이 중요합니다. 또 다른 해결방안으로는 leaky relu와 같은 activation function을 사용할 수도 있습니다.

1. **각 Actication functrion 마다 분류가 가장 잘 되는 클래스와 가장 안되는 클래스는 ?**

Precision과 Recall의 조화 평균인 F1 Score를 기준으로 결정하면 , tanh 함수의 경우 분류가 가장 잘 되는 class는 automobile 이며 가장 안 되는 class 는 dog 입니다. sigmoid 함수의 경우, 분류가 가장 잘 되는 class는 ship이며, 가장 안 되는 class 는 cat 입니다. relu 함수의 경우 분류가 가장 잘 되는 class는 ship 이며, 가장 안 되는 class 는 dog 입니다.

1. **①에서 평균 F1 Score가 가장 좋은 Activation Function을 기준으로 하여, 아래의 범위에 있는 Epoch 및 데이터 개수를 조정하여 실험. 이 때 평균 F1 Score가 가장 높은 변경 조합(Activation Function, Epoch, Data 수) 및 그때의 평균 F1 Score를 제출하고 결과를 해석 하시오**

* **Epoch : 10, 20, 30**
* **(train/valid/test) 개수 : (5,000/1,000/1,000), (45,000/1,000/10,000)**

ReLU 함수를 기준으로, 실험결과, 평균 F1 Score가 가장 높은 변경 조합은 ReLU-30-45000 이며, 이 때 편균 F1 Score 는 0.65입니다. Activation 함수에 대해서는 relu 함수가 tanh 함수나 sigmoid 함수보다 좋은 성능을 보이고, data 수가 많으면 시간이 오래 걸리지만 성능은 확실히 좋아지는 것으로 보입니다. 또, epoch 값을 늘릴수록 좀 더 성능이 좋은 모델을 얻을 수 있는 것 같습니다.

1. **과제3 데이터(생필품 분류)**에 CNN모델을 적용해서 학습해보고 F1 정확도를 비교하시오

(소수 둘째 짜리까지; 학습data의 차원을 (-1,1,7,1)로 reshape, train:test=8:2, train:validation=3:1로 나눌 것)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| parameters  Class | Conv2D kernel\_size  (1, 2)  MaxPooling2D size  (1, 2) | | Conv2D kernel\_size  (1, 3)  MaxPooling2D size  (1, 2) | | Conv2D kernel\_size  (1, 4)  MaxPooling2D size  (1, 2) | |
| dense output  16 | dense output  64 | dense output  16 | dense output  64 | dense output  16 | dense output  64 |
| Horeca | 0.87 | 0.79 | 0.79 | 0.40 | 0.84 | 0.84 |
| Retail | 0.53 | 0.00 | 0.00 | 0.21 | 0.00 | 0.00 |
| 평균 | 0.70 | 0.40 | 0.40 | 0.31 | 0.42 | 0.42 |

kernel size = (1,2) 이고 dense output = 16, 가장 높은 F1 정확도 값을 얻을 수 있었습니다.

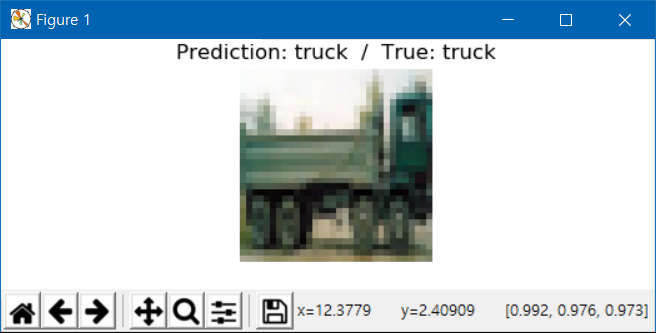
1. **⑤의 결과를 DecisionTree 및 KNN으로 수행했던 과제3의 결과와 비교하여 분석하시오**

과제3의 결과가 Decision Tree 도, KNN도, 아무리 F1 Score 값이 작게 나와봤자, 0.79정도 였던 것에 비해, parameter를 여러 조합으로 시도해봐도 이번 과제4의 결과가 확실히 값이 더 작은 것을 확인할 수 있습니다. 무조건 CNN 이나 Deep learning 방법이 좋은 성능을 보이는 것이 아니라, 문제에 따라 필요한 학습이나 추론 방법을 선택하는 것이 중요한 것으로 보이고, 특히, CNN은 이미지 학습에 있어서 좀 더 나은 성능을 보이는 것 같습니다.

## Run

>>> (epoch 20, 30 의 경우는 생략)

**activation = ‘tanh’, epoch=10**



Using TensorFlow backend.

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\python\framework\op\_def\_library.py:263: colocate\_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.

Instructions for updating:

Colocations handled automatically by placer.

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\keras\backend\tensorflow\_backend.py:3445: calling dropout (from tensorflow.python.ops.nn\_ops) with keep\_prob is deprecated and will be removed in a future version.

Instructions for updating:

Please use `rate` instead of `keep\_prob`. Rate should be set to `rate = 1 - keep\_prob`.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 28, 28, 16) 1216

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 14, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 14, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 3136) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 32) 100384

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 10) 330

=================================================================

Total params: 101,930

Trainable params: 101,930

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\python\ops\math\_ops.py:3066: to\_int32 (from tensorflow.python.ops.math\_ops) is deprecated and will be removed in a future version.

Instructions for updating:

Use tf.cast instead.

Train on 5000 samples, validate on 1000 samples

Epoch 1/10

2019-06-01 21:54:24.354673: I tensorflow/core/platform/cpu\_feature\_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2

5000/5000 [==============================] - 4s 735us/step - loss: 2.0544 - acc: 0.2580 - val\_loss: 1.7817 - val\_acc: 0.3970

Epoch 2/10

5000/5000 [==============================] - 3s 643us/step - loss: 1.7874 - acc: 0.3730 - val\_loss: 1.6515 - val\_acc: 0.4290

Epoch 3/10

5000/5000 [==============================] - 3s 642us/step - loss: 1.6763 - acc: 0.4220 - val\_loss: 1.5772 - val\_acc: 0.4370

Epoch 4/10

5000/5000 [==============================] - 3s 641us/step - loss: 1.6133 - acc: 0.4338 - val\_loss: 1.5361 - val\_acc: 0.4690

Epoch 5/10

5000/5000 [==============================] - 3s 652us/step - loss: 1.5495 - acc: 0.4684 - val\_loss: 1.5433 - val\_acc: 0.4620

Epoch 6/10

5000/5000 [==============================] - 3s 646us/step - loss: 1.5671 - acc: 0.4624 - val\_loss: 1.4582 - val\_acc: 0.4900

Epoch 7/10

5000/5000 [==============================] - 3s 654us/step - loss: 1.4849 - acc: 0.4860 - val\_loss: 1.3919 - val\_acc: 0.5140

Epoch 8/10

5000/5000 [==============================] - 3s 675us/step - loss: 1.4486 - acc: 0.4980 - val\_loss: 1.3914 - val\_acc: 0.5000

Epoch 9/10

5000/5000 [==============================] - 3s 646us/step - loss: 1.4260 - acc: 0.5112 - val\_loss: 1.3984 - val\_acc: 0.5010

Epoch 10/10

5000/5000 [==============================] - 3s 645us/step - loss: 1.3921 - acc: 0.5192 - val\_loss: 1.3104 - val\_acc: 0.5340

Evaluation

1000/1000 [==============================] - 0s 325us/step

Prediction

1000/1000 [==============================] - 0s 349us/step

CNN on CIFAR10

airplane 0.452632

automobile 0.597701

bird 0.402174

cat 0.246914

deer 0.338235

dog 0.287879

frog 0.551402

horse 0.564706

ship 0.493056

truck 0.542553

Name: Precision, dtype: float64

airplane 0.417476

automobile 0.584270

bird 0.370000

cat 0.392157

deer 0.255556

dog 0.218391

frog 0.526786

horse 0.470588

ship 0.669811

truck 0.467890

Name: Recall, dtype: float64

airplane 0.434343

automobile 0.590909

bird 0.385417

cat 0.303030

deer 0.291139

dog 0.248366

frog 0.538813

horse 0.513369

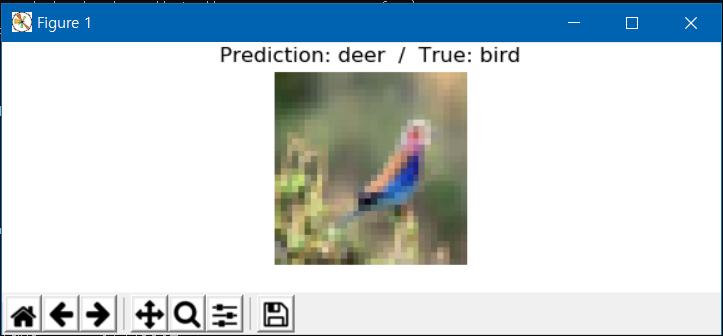
ship 0.568000

truck 0.502463

Name: F1, dtype: float64

Total Accuracy: 44.30

**activation = ‘sigmoid’, epoch = 10**



Using TensorFlow backend.

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\python\framework\op\_def\_library.py:263: colocate\_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.

Instructions for updating:

Colocations handled automatically by placer.

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\keras\backend\tensorflow\_backend.py:3445: calling dropout (from tensorflow.python.ops.nn\_ops) with keep\_prob is deprecated and will be removed in a future version.

Instructions for updating:

Please use `rate` instead of `keep\_prob`. Rate should be set to `rate = 1 - keep\_prob`.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 28, 28, 16) 1216

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 14, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 14, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 3136) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 32) 100384

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 10) 330

=================================================================

Total params: 101,930

Trainable params: 101,930

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\python\ops\math\_ops.py:3066: to\_int32 (from tensorflow.python.ops.math\_ops) is deprecated and will be removed in a future version.

Instructions for updating:

Use tf.cast instead.

Train on 5000 samples, validate on 1000 samples

Epoch 1/10

2019-06-01 21:58:54.405062: I tensorflow/core/platform/cpu\_feature\_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2

5000/5000 [==============================] - 4s 724us/step - loss: 2.3262 - acc: 0.1106 - val\_loss: 2.2874 - val\_acc: 0.1240

Epoch 2/10

5000/5000 [==============================] - 3s 647us/step - loss: 2.2594 - acc: 0.1504 - val\_loss: 2.2062 - val\_acc: 0.2050

Epoch 3/10

5000/5000 [==============================] - 3s 645us/step - loss: 2.1890 - acc: 0.2054 - val\_loss: 2.1291 - val\_acc: 0.2550

Epoch 4/10

5000/5000 [==============================] - 3s 642us/step - loss: 2.1182 - acc: 0.2376 - val\_loss: 2.0696 - val\_acc: 0.2540

Epoch 5/10

5000/5000 [==============================] - 3s 643us/step - loss: 2.0660 - acc: 0.2554 - val\_loss: 2.0201 - val\_acc: 0.2690

Epoch 6/10

5000/5000 [==============================] - 3s 642us/step - loss: 2.0331 - acc: 0.2640 - val\_loss: 1.9855 - val\_acc: 0.2880

Epoch 7/10

5000/5000 [==============================] - 3s 653us/step - loss: 1.9996 - acc: 0.2864 - val\_loss: 1.9561 - val\_acc: 0.3000

Epoch 8/10

5000/5000 [==============================] - 3s 649us/step - loss: 1.9773 - acc: 0.2984 - val\_loss: 1.9377 - val\_acc: 0.3050

Epoch 9/10

5000/5000 [==============================] - 3s 662us/step - loss: 1.9540 - acc: 0.3074 - val\_loss: 1.9063 - val\_acc: 0.3250

Epoch 10/10

5000/5000 [==============================] - 3s 653us/step - loss: 1.9332 - acc: 0.3078 - val\_loss: 1.8905 - val\_acc: 0.3250

Evaluation

1000/1000 [==============================] - 0s 328us/step

Prediction

1000/1000 [==============================] - 0s 352us/step

CNN on CIFAR10

C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples.

'precision', 'predicted', average, warn\_for)

C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in labels with no predicted samples.

'precision', 'predicted', average, warn\_for)

airplane 0.396552

automobile 0.420000

bird 0.296296

cat 0.000000

deer 0.185841

dog 0.346154

frog 0.253049

horse 0.308642

ship 0.441441

truck 0.409836

Name: Precision, dtype: float64

airplane 0.446602

automobile 0.235955

bird 0.080000

cat 0.000000

deer 0.233333

dog 0.206897

frog 0.741071

horse 0.245098

ship 0.462264

truck 0.458716

Name: Recall, dtype: float64

airplane 0.420091

automobile 0.302158

bird 0.125984

cat 0.000000

deer 0.206897

dog 0.258993

frog 0.377273

horse 0.273224

ship 0.451613

truck 0.432900

Name: F1, dtype: float64

Total Accuracy: 32.10

**activation = ‘relu’, epoch = 10**



Using TensorFlow backend.

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\python\framework\op\_def\_library.py:263: colocate\_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.

Instructions for updating:

Colocations handled automatically by placer.

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\keras\backend\tensorflow\_backend.py:3445: calling dropout (from tensorflow.python.ops.nn\_ops) with keep\_prob is deprecated and will be removed in a future version.

Instructions for updating:

Please use `rate` instead of `keep\_prob`. Rate should be set to `rate = 1 - keep\_prob`.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 28, 28, 16) 1216

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 14, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 14, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 3136) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 32) 100384

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 10) 330

=================================================================

Total params: 101,930

Trainable params: 101,930

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

WARNING:tensorflow:From C:\Users\hyebeen\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\python\ops\math\_ops.py:3066: to\_int32 (from tensorflow.python.ops.math\_ops) is deprecated and will be removed in a future version.

Instructions for updating:

Use tf.cast instead.

Train on 5000 samples, validate on 1000 samples

Epoch 1/10

2019-06-01 22:08:13.729860: I tensorflow/core/platform/cpu\_feature\_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2

5000/5000 [==============================] - 4s 743us/step - loss: 2.1368 - acc: 0.1994 - val\_loss: 1.9829 - val\_acc: 0.2590

Epoch 2/10

5000/5000 [==============================] - 3s 639us/step - loss: 1.8920 - acc: 0.3194 - val\_loss: 1.8142 - val\_acc: 0.3410

Epoch 3/10

5000/5000 [==============================] - 3s 631us/step - loss: 1.7438 - acc: 0.3748 - val\_loss: 1.6423 - val\_acc: 0.4210

Epoch 4/10

5000/5000 [==============================] - 3s 632us/step - loss: 1.6468 - acc: 0.4138 - val\_loss: 1.5646 - val\_acc: 0.4500

Epoch 5/10

5000/5000 [==============================] - 3s 630us/step - loss: 1.5976 - acc: 0.4370 - val\_loss: 1.5145 - val\_acc: 0.4440

Epoch 6/10

5000/5000 [==============================] - 3s 629us/step - loss: 1.5359 - acc: 0.4546 - val\_loss: 1.4580 - val\_acc: 0.4810

Epoch 7/10

5000/5000 [==============================] - 3s 639us/step - loss: 1.4811 - acc: 0.4774 - val\_loss: 1.3879 - val\_acc: 0.5110

Epoch 8/10

5000/5000 [==============================] - 3s 641us/step - loss: 1.4274 - acc: 0.5002 - val\_loss: 1.4093 - val\_acc: 0.5160

Epoch 9/10

5000/5000 [==============================] - 3s 670us/step - loss: 1.4129 - acc: 0.5082 - val\_loss: 1.3630 - val\_acc: 0.5090

Epoch 10/10

5000/5000 [==============================] - 3s 642us/step - loss: 1.3637 - acc: 0.5198 - val\_loss: 1.2667 - val\_acc: 0.5560

Evaluation

1000/1000 [==============================] - 0s 317us/step

Prediction

1000/1000 [==============================] - 0s 346us/step

CNN on CIFAR10

airplane 0.558140

automobile 0.547619

bird 0.403670

cat 0.326087

deer 0.367647

dog 0.352941

frog 0.532847

horse 0.544643

ship 0.530612

truck 0.552632

Name: Precision, dtype: float64

airplane 0.466019

automobile 0.516854

bird 0.440000

cat 0.294118

deer 0.277778

dog 0.206897

frog 0.651786

horse 0.598039

ship 0.735849

truck 0.577982

Name: Recall, dtype: float64

airplane 0.507937

automobile 0.531792

bird 0.421053

cat 0.309278

deer 0.316456

dog 0.260870

frog 0.586345

horse 0.570093

ship 0.616601

truck 0.565022

Name: F1, dtype: float64

Total Accuracy: 48.60