2020-2021 Görüntü İşleme Projesi

Ad Soyad: Rahmi Cemre Ünal

Konu : Konvolüsyonel Sinir Ağı Tasarımı Tabanlı Sınıflandırma ve Öğrenme Aktarımı Tabanlı Görüntü Erişimi Uygulaması

Açıklama : Bu ödevde CINIC-10 verisetin üzerinde konvolüsyonel ağ tasarımı yapılacaktır. Bunun yanı sıra daha önce Imagenet verisetinde eğitilmiş VGG-16 ve Resnet-50 modellerinin öğrenme aktarımı(transfer learning) ile ilgili CINIC-10 ile yeniden sınıflandırılması ve bu modelden elde edilen öznitelik vektörü (feature vector) ile bir resme en çok benzeyen 5 adet resmi bulan bir sistem tasarlanacak ve gerçeklenecektir.

YÖNTEM

Veri Okuma

CNN tabanlı bir model eğitimi için öncelikle verilerin uygun formata döüştürülmesi gerekiyor. Bunun için keras kütüphanesinin ImageDataGenerator fonksiyonu kullanıldı. İçerisine verilen rescale parametresi ile alınacak görüntüdeki her pixel değerinin 255'e bölünerek normalize edilmesi sağlandı.

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(rescale=1./255)
```

Verisetini okuyacak generator oluşturulduktan sonra uygun parametrelerle çalıştırıldı. İçerisine fotoğraf sınıflarıınn bulunduğu path, hedeflenen sınıfların tipi (class_mode), alınacak fotoğrafların dönüştürüleceği boyut (target_size), ve verinin karışık olarak okunması için gerekli bilgi (shuffle) girildi.

```
train_it = datagen.flow_from_directory('train/', class_mode='categorica
l', target_size=(224, 224), shuffle=True)
```

Bu şekilde fotoğrafların pixel değerleri 255'e bölünmüş, karışık sırayla okunmuş ve boyutları (224,224)'e getirilmiş oldu.

Yukarıdaki kod parçası ile train için ayrılan veriler okundu. Aynı şekilde validation ve test için ayrılmış veriler de okundu.

Sıfırdan Model Oluşturma ve Eğitme ve Test etme

Model oluşturma ve eğitim işlemleri için keras kütüphanesi kullanıldı. Ödev dökümanındaki tabloda istenilen bir kombinasyon için örnek kod: (2 Katman - Her katman için 32 filtre - 5x5 filtre boyutlu - GlorotNormal - ReLu Aktivasyon - 0.2 Dropout Oranı - Adam Optimizasyon Algoritması)

*10 sınıflı bir tahmin yapılmak istendiği için modelin sonuna Fully connected Layer (Dense) eklendi.

```
from keras.models import Sequential
from keras.layers import Conv2D, Flatten, Dense, Dropout
```

```
model = Sequential()

model.add(Conv2D(32, (5, 5), activation='relu', kernel_initializer='glo
rot_normal', input_shape=(224, 224, 3)))

model.add(Dropout(0.2))

model.add(Conv2D(32, (5, 5), activation='relu', kernel_initializer='glo
rot_normal'))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(10, activation='softmax', kernel_initializer='glorot_no
rmal'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metric
s=['accuracy'])
```

Modelin özeti:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 220, 220, 32)      2432
_________________________________________________________________
dropout (Dropout)            (None, 220, 220, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 216, 216, 32)      25632
_________________________________________________________________
dropout_1 (Dropout)          (None, 216, 216, 32)      0
_________________________________________________________________
flatten (Flatten)            (None, 1492992)           0
_________________________________________________________________
dense (Dense)                (None, 10)                14929930
=================================================================
Total params: 14,957,994
Trainable params: 14,957,994
Non-trainable params: 0
```

Model oluşturulduktan sonra model.fit fonksiyonuyla eğitim başlatıldı.

```
model.fit(
        train_it
        , epochs=10
        , validation_data=val_it
        , batch_size=32
    )
```

Test işlemi için model.evaluate fonksiyonu kullanıldı.

```
results = model.evaluate(test_it, verbose=1)
```

Test işlemi için örnek çıktı: loss: 2.0308 - accuracy: 0.2933

Consufion Matrix için sklearn kütüphanesi kullanıldı.

Öncelikle gerçek etiketler ve tahmin edilen etiketler elde edildi.

```
probabilities = model.predict(test_it)
predicted_labels = [x.argmax() for x in probabilities]
y_true = test_it.classes
```

Daha sonrasında confusion matrix üretildi.

```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(y_true, predicted_labels)
```

UYGULAMA

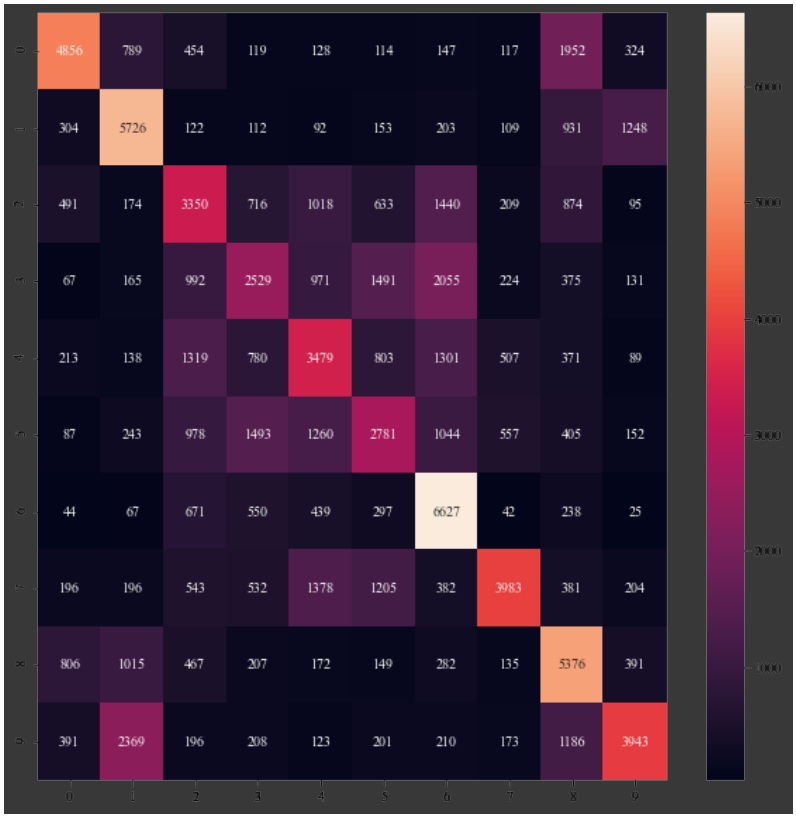
İSTENİLEN KOMBİNASYONLAR

1) 2 Katman Her katman için 32 filtre 3x3 filtre boyutlu GlorotNormal (xavier_normal_) ReLu Aktivasyon 0.2 Dropout Oranı Adam Optimizasyon Algoritması

```
loss: 1.4918 - accuracy: 0.4739
```

```
[[4856  789  454  119  128  114  147  117 1952  324]
 [ 304 5726  122  112   92  153  203  109  931 1248]
 [ 491  174 3350  716 1018  633 1440  209  874   95]
 [  67  165  992 2529  971 1491 2055  224  375  131]
 [ 213  138 1319  780 3479  803 1301  507  371   89]
 [  87  243  978 1493 1260 2781 1044  557  405  152]
 [  44   67  671  550  439  297 6627   42  238   25]
 [ 196  196  543  532 1378 1205  382 3983  381  204]
 [ 806 1015  467  207  172  149  282  135 5376  391]
 [ 391 2369  196  208  123  201  210  173 1186 3943]]
```

```
airplane. acc: 53.96% (ship ile karıştırılmış(1952 airplane, ship
olarak tahmin edilmiş)
automobile. acc: 63.62% (truck ile karıştırılmış)
bird. acc: 37.22% (frog ile karıştırılmış)
cat. acc: 28.10% (frog ile karıştırılmış)
deer. acc: 38.66% (bird ve frog ile karıştırılmış)
dog. acc: 30.90% (cat ile karıştırılmış)
frog. acc: 73.63% (bird ile karıştırılmış)
horse. acc: 44.26% (deer ile karıştırılmış)
ship. acc: 59.73% (automobile ile karıştırılmış)
truck. acc: 43.81% (automobile ile karıştırılmış)
```



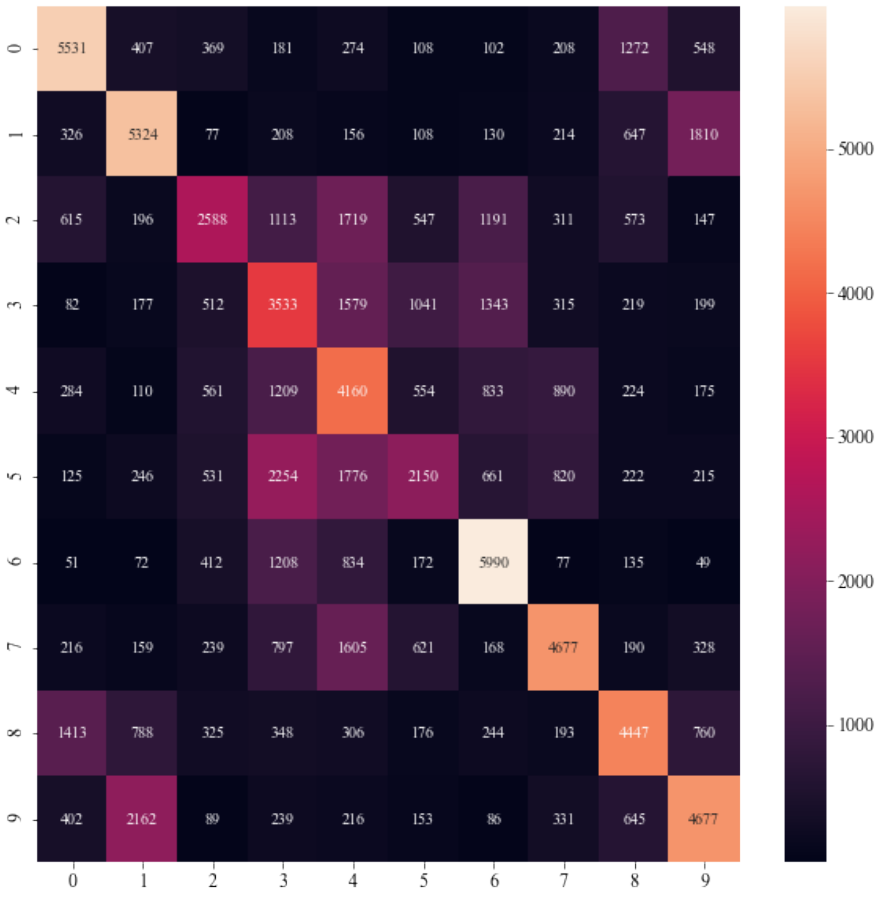
2) 2 Katman Her katman için 32 filtre 5x5 filtre boyutlu GlorotNormal (xavier_normal_) ReLu Aktivasyon 0.2 Dropout Oranı Adam Optimizasyon Algoritması

```
loss: 1.4716 - accuracy: 0.4786
```

```
[[5531  407  369  181  274  108  102  208 1272  548]
 [ 326 5324   77  208  156  108  130  214  647 1810]
 [ 615  196 2588 1113 1719  547 1191  311  573  147]
 [  82  177  512 3533 1579 1041 1343  315  219  199]
 [ 284  110  561 1209 4160  554  833  890  224  175]
 [ 125  246  531 2254 1776 2150  661  820  222  215]
 [  51   72  412 1208  834  172 5990   77  135   49]
 [ 216  159  239  797 1605  621  168 4677  190  328]
 [1413  788  325  348  306  176  244  193 4447  760]
 [ 402 2162   89  239  216  153   86  331  645 4677]]
```

```
airplane. acc: 61.46%
automobile. acc: 59.16%
bird. acc: 28.76%
cat. acc: 39.26%
deer. acc: 46.22%
dog. acc: 23.89%
frog. acc: 66.56%
horse. acc: 51.97%
ship. acc: 49.41%
truck. acc: 51.97%
```

Bir önceki modele göre filtre boyutunun (3,3) den (5,5) e değişimi genel başarıyı etkilememiştir.



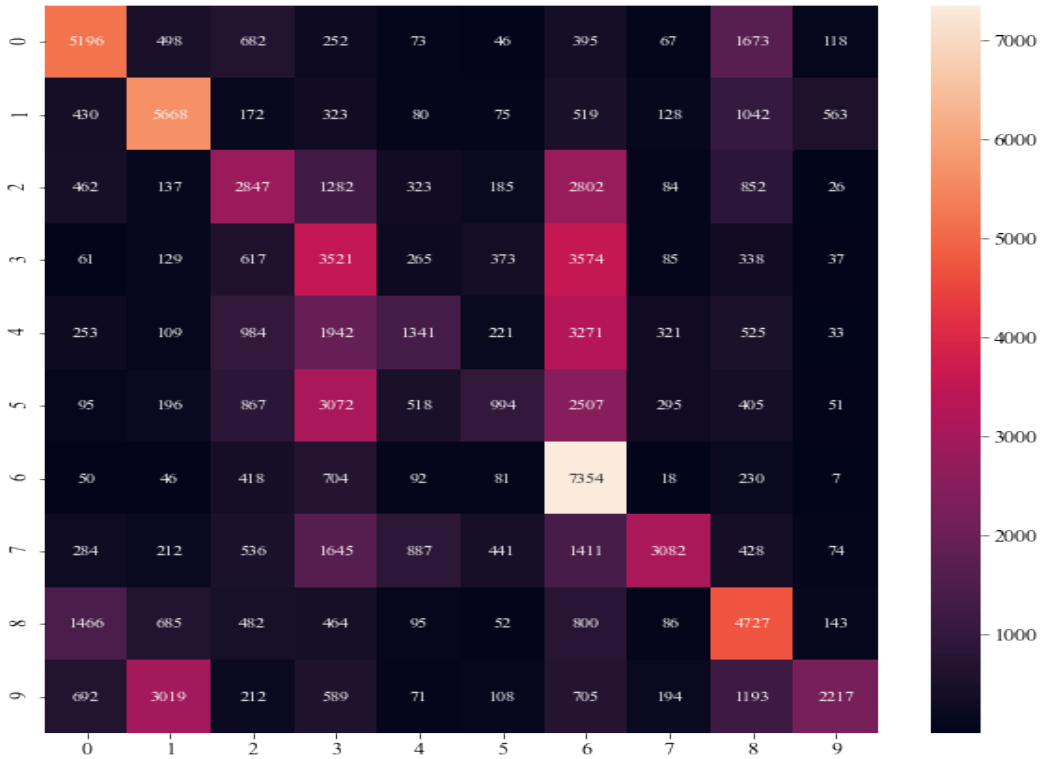
3) 2 Katman Her katman için 32 filtre 3x3 filtre boyutlu GlorotNormal (xavier_normal_) ReLu Aktivasyon 0.7 Dropout Oranı Adam Optimizasyon Algoritması

```
Loss: 1.6329201459884644, Accuracy: 0.4105222225189209]
```

```
[[5196  498  682  252   73   46  395   67 1673  118]
 [ 430 5668  172  323   80   75  519  128 1042  563]
 [ 462  137 2847 1282  323  185 2802   84  852   26]
 [  61  129  617 3521  265  373 3574   85  338   37]
 [ 253  109  984 1942 1341  221 3271  321  525   33]
 [  95  196  867 3072  518  994 2507  295  405   51]
 [  50   46  418  704   92   81 7354   18  230    7]
 [ 284  212  536 1645  887  441 1411 3082  428   74]
 [1466  685  482  464   95   52  800   86 4727  143]
 [ 692 3019  212  589   71  108  705  194 1193 2217]]
```

```
airplane. acc: 57.73%
automobile. acc: 62.98%
bird. acc: 31.63%
cat. acc: 39.12%
deer. acc: 14.90%
dog. acc: 11.04%
frog. acc: 81.71%
horse. acc: 34.24%
ship. acc: 52.52%
truck. acc: 24.63%
```

1. Modele göre Drop Out oranının 0.2 den 0.7 ye çıkarılması başarının düşmesine yol açmıştır. Burdan çıkarılabilecek sonuç, Drop out çok yüksekken öğrenilen bilgilerin çoğu unutulduğu için doğru bir eğitim sürecinin sağlanamadığıdır. Özellikle Deer ve Dog sınıflarının başarısı ciddi oranda düşmüştür. Frog sınıfının başarısı artmış ve 81% gibi yüksek bir oran ölçülmüştür fakat bunun sebebi, modelimizin çoğu tahminine frog demesidir. Diğer sınıflarda yanlış tahmin edilen örneklerin ciddi bir kısmı frog olarak tahmin edildiği için bu sonuç elde edilmiştir.



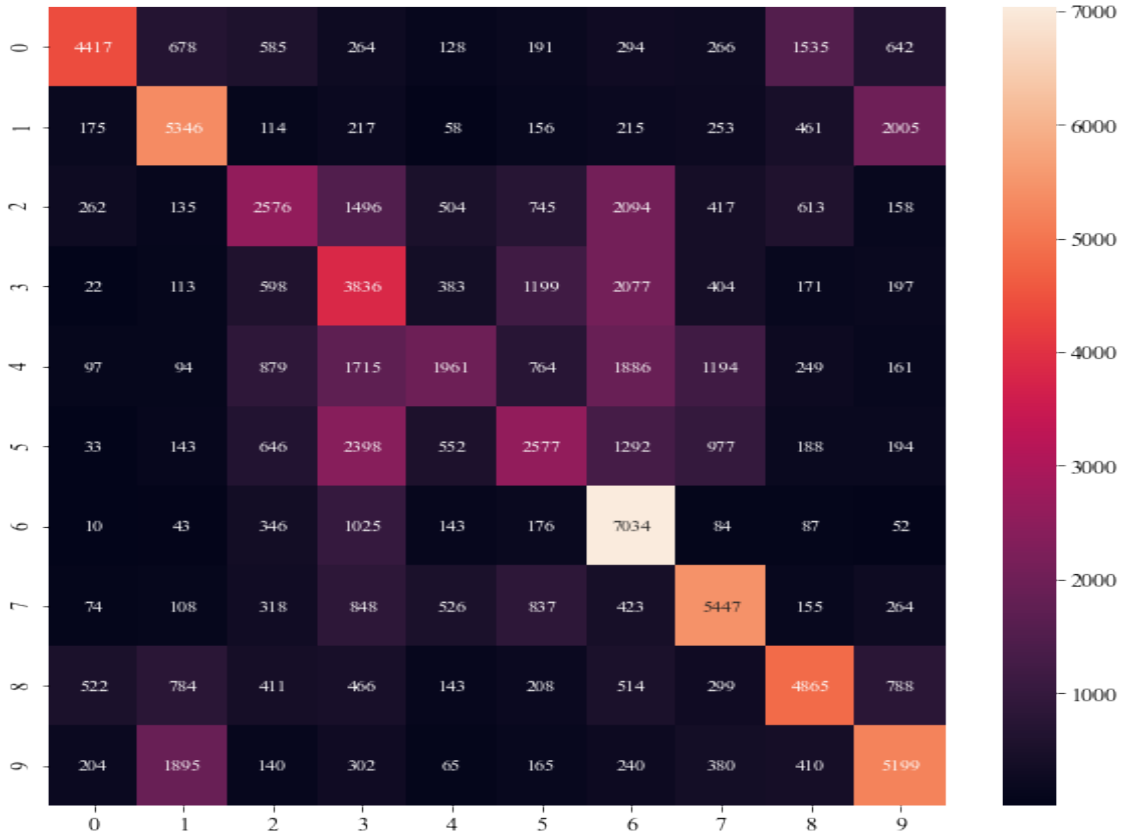
4) Katman Her katman için 32 filtre 3x3 filtre boyutlu GlorotNormal (xavier_normal_) ReLu Aktivasyon 0.2 Dropout Oranı Adam Optimizasyon Algoritması

```
loss: 1.4804 - accuracy: 0.4806
```

```
[[4417  678  585  264  128  191  294  266 1535  642]
 [ 175 5346  114  217   58  156  215  253  461 2005]
 [ 262  135 2576 1496  504  745 2094  417  613  158]
 [  22  113  598 3836  383 1199 2077  404  171  197]
 [  97   94  879 1715 1961  764 1886 1194  249  161]
 [  33  143  646 2398  552 2577 1292  977  188  194]
 [  10   43  346 1025  143  176 7034   84   87   52]
 [  74  108  318  848  526  837  423 5447  155  264]
 [ 522  784  411  466  143  208  514  299 4865  788]
 [ 204 1895  140  302   65  165  240  380  410 5199]]
```

```
airplane. acc: 49.08%
automobile. acc: 59.40%
bird. acc: 28.62%
cat. acc: 42.62%
deer. acc: 21.79%
dog. acc: 28.63%
frog. acc: 78.16%
horse. acc: 60.52%
ship. acc: 54.06%
truck. acc: 57.77%
```

Bu modelin 1. Modele göre tek farkı konvolüsyon katman sayısının 2'den 3'e çıkartılmış olmasıdır. Başarıda ufak bir artış gözlense de birbirine çok yakın sonuçlar elde edilmiştir. İlk model gibi frog sınıfınını başarısı yüksek çıkmıştır fakat yanlış yapılan tahminlerin de içeriğinde fazla sayıda frog geçmiştir.



5) 3 Katman Her katman için 32 filtre 5x5 filtre boyutlu GlorotNormal (xavier_normal_) ReLu Aktivasyon 0.2 Dropout Oranı Adam Optimizasyon Algoritması

```
loss: 2.0308 - accuracy: 0.2933
```

```
[[3099  924 1159  115  311   91  473  497 1472  859]
 [ 324 3818  156  267  289   75  767  414  477 2413]
 [ 388  730 1180  881 1056  982 2302  614  568  299]
 [  94  401  489 1290 1197  927 3364  682  278  278]
 [ 319  271  440  843 1374 1402 2657 1291  201  202]
 [ 133  623  613 1118 1265 1005 2706  879  285  373]
 [  37  213  201  857  880  807 5483  287  141   94]
 [ 256  496  220  698 1422  602 1608 3072  156  470]
 [1140  839  863  288  366  179  946  477 2912  990]
 [ 522 2311  135  265  414   90  748  740  609 3166]]
airplane. acc: 34.43%
automobile. acc: 42.42%
bird. acc: 13.11%
cat. acc: 14.33%
deer. acc: 15.27%
dog. acc: 11.17%
frog. acc: 60.92%
horse. acc: 34.13%
ship. acc: 32.36%
truck. acc: 35.18%
```

En kötü sonuç veren bu model olmuştur. Bu derece bir başarıda düşüşün nedeninin (3x3) filtre boyutundan (5x5)'e geçilmesi olduğu gözlemlenmiştir. Automobile sınıfı çok yüksek oranla Truck sınıfı ile karıştırılmıştır. Model özellikle Bird, Cat, Deer, Dog sınıflarını öğrenememiştir. Ağırlıklı olarak Frog olarak tahmin vermeye yönelmiştir. Modellerin geneline baktığımızda Frog sınıfına ekstra bir yönelme mevcuttur.

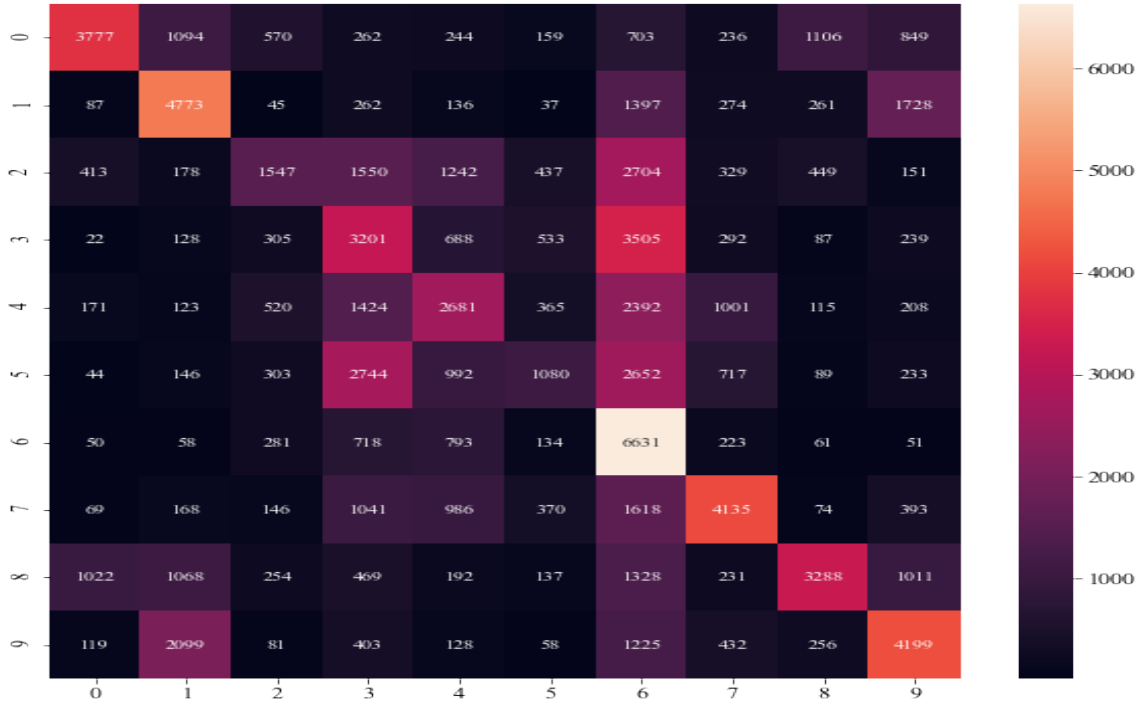
6) 3 Katman Her katman için 32 filtre 3x3 filtre boyutlu GlorotNormal (xavier_normal_) ReLu Aktivasyon 0.7 Dropout Oranı Adam Optimizasyon Algoritması

```
loss: 1.7044 - accuracy: 0.3924
```

```
[[3777 1094  570  262  244  159  703  236 1106  849]
 [  87 4773   45  262  136   37 1397  274  261 1728]
 [ 413  178 1547 1550 1242  437 2704  329  449  151]
 [  22  128  305 3201  688  533 3505  292   87  239]
 [ 171  123  520 1424 2681  365 2392 1001  115  208]
 [  44  146  303 2744  992 1080 2652  717   89  233]
 [  50   58  281  718  793  134 6631  223   61   51]
 [  69  168  146 1041  986  370 1618 4135   74  393]
 [1022 1068  254  469  192  137 1328  231 3288 1011]
 [ 119 2099   81  403  128   58 1225  432  256 4199]]
```

```
airplane. acc: 41.97%
automobile. acc: 53.03%
bird. acc: 17.19%
cat. acc: 35.57%
deer. acc: 29.79%
dog. acc: 12.00%
frog. acc: 73.68%
horse. acc: 45.94%
ship. acc: 36.53%
truck. acc: 46.66%
```

4. model ile kıyasladığımızda sadece drop out oranının 0.2 den 0.7 ye çıkartıldığı ve başarının düştüğü gözlemlenmiştir. Bunun sebebinin daha öncede belirtildiği gibi drop out oranının yüksek olmasının, modelin gereğinden fazla bilgiyi unutması ve eğitim sürecinin sekteye uğraması olduğu düşünülmüştür. Ayrıca modellerin tekli sınıf başarıları incelendiğinde Dog sınıfının her modelde düşük olduğu görülmüştür. Bunun sebebi, yeterli kalitede Dog sınıfına ait veri olmaması olabilir.



Ödev Güncellenmeden Önce Denenen Kombinasyonlar

Ödev güncellemesinden sonraki 6 kombinasyon ve sonuçları yukarıda paylaşılmıştır. Aşağıdaki tabloda, ödev güncellenmeden önce denenen 12 kombinasyon belirtildiği sıralarıyla gösterilmiştr.

Paylaşılan sonuçların formatları:

İlk satır: [Loss, Accuracy]

Diğer kısımlar:

Confusion Matrix(rows= True Labels, columns = Predicted Labels)

10 sınıfa ait başarı oranları

Confusion Matrix görseli

DENENEN KOMBİNASYONLAR

Katman Sayısı	Katman Filtre Sayısı	Filtre Boyutu	Optimizer	Initializer	Activation
1 conv	32	(3,3)	SGD	'he_uniform'	Relu
2 conv	32	(3,3)	SGD	'he_uniform'	Relu
3 conv	32	(3,3)	SGD	'he_uniform'	Relu
3 conv	16	(3,3)	SGD	'he_uniform'	Relu
3 conv	32	(5,5)	SGD	'he_uniform'	Relu
3 conv	64	(5,5)	SGD	'he_uniform'	Relu
3 conv	64	(5,5)	SGD	'glorot_normal'	Relu
3 conv	32	(3,3)	Adam	'he_uniform'	Relu
2 conv	32	(3,3)	Adam	'he_uniform'	Sigmoid
2 conv	32	(3,3)	Adam	'he_uniform'	Relu
2 conv	32	(3,3)	RMSProp	'he_uniform'	Relu
6 conv	32	(3,3)	SGD	'he_uniform'	Relu

1 conv 32 (3,3) SGD 'he_uniform' Relu

```
[1.6839308738708496, 0.37255555391311646]

[[425 130 107   3   8   7   4  33 145  38]
 [ 32 674  12  17   0  10  13  51  51  40]
 [129 103 169  73  11 128  44  67 163  13]
 [  5 125  24 228   8 262 119  82  40   7]
 [ 31  88  36 104  29 231  82 256  33  10]
 [ 15  73  56 144   6 350 105  92  54   5]
 [  3 105  38  75   3 241 328  61  31  15]
 [ 28 134   7  48   5  99  22 519  35   3]
 [143 106  86   9   7  15   5  22 462  45]
 [ 45 535   4  10   5   8   5  30  89 169]]
airplane. acc: 47.22%
automobile. acc: 74.89%
bird. acc: 18.78%
cat. acc: 25.33%
deer. acc: 3.22%
dog. acc: 38.89%
frog. acc: 36.44%
horse. acc: 57.67%
ship. acc: 51.33%
truck. acc: 18.78%
```

2 conv 32 (3,3) SGD 'he_uniform' Relu

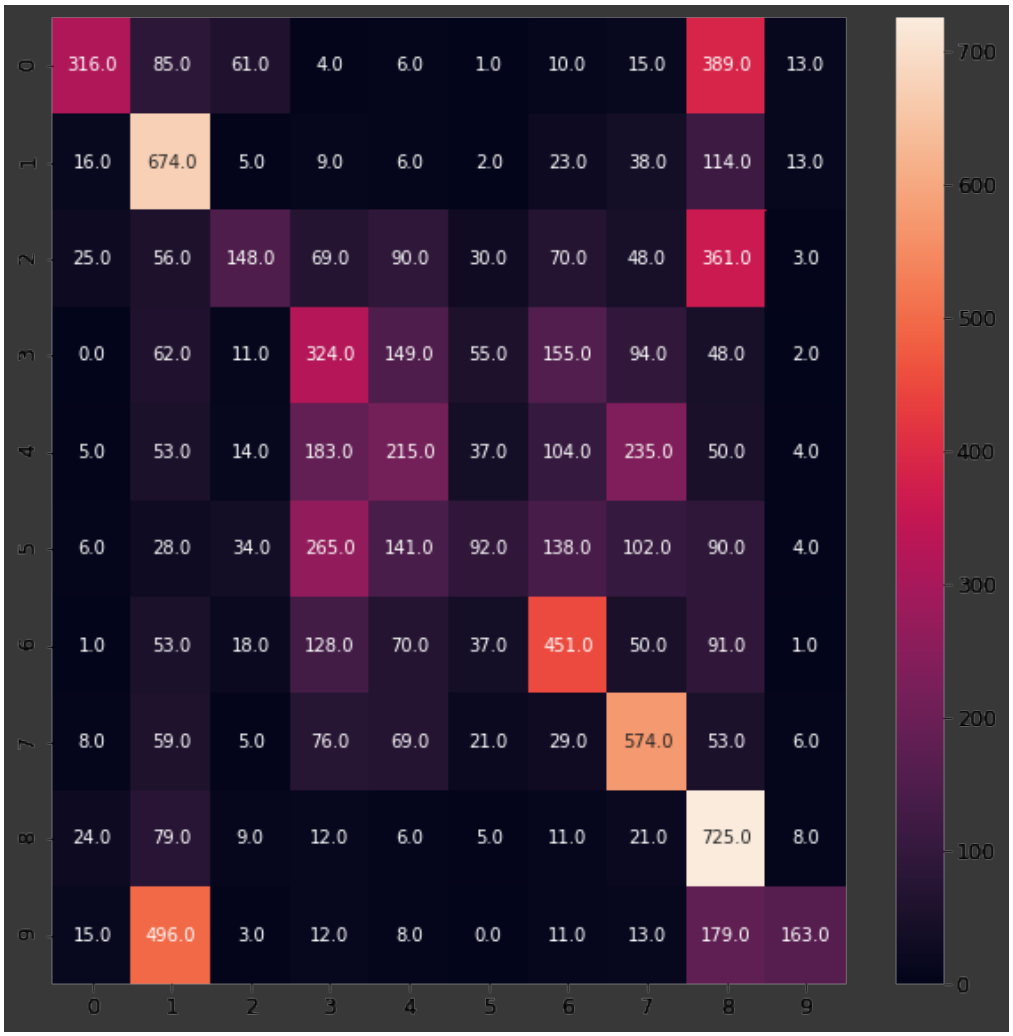
```
[1.537338376045227, 0.441222220659256]
```

```
[[603  37  42   1   8  16   9  16 116  52]
 [ 51 503   9   9  10  26  18  56  54 164]
 [215  36 138  26  51 200  56  38 122  18]
 [  7  28  13 195  65 381 130  31  28  22]
 [ 33  30  17  63 181 303  81 145  31  16]
 [ 17   6  38  77  50 512 105  41  40  14]
 [ 12  34  12  80  23 286 373  20  38  22]
 [ 19  51   4  49  65 148  21 506  19  18]
 [161  61  21  11  11  21   7  32 514  61]
 [ 61 235   6  16   9  12  14  25  76 446]]
airplane. acc: 67.00%
automobile. acc: 55.89%
bird. acc: 15.33%
cat. acc: 21.67%
deer. acc: 20.11%
dog. acc: 56.89%
frog. acc: 41.44%
horse. acc: 56.22%
ship. acc: 57.11%
truck. acc: 49.56%
```

3 conv 32 (3,3) SGD 'he_uniform' Relu

```
[1.6194067001342773, 0.4091111123561859]
```

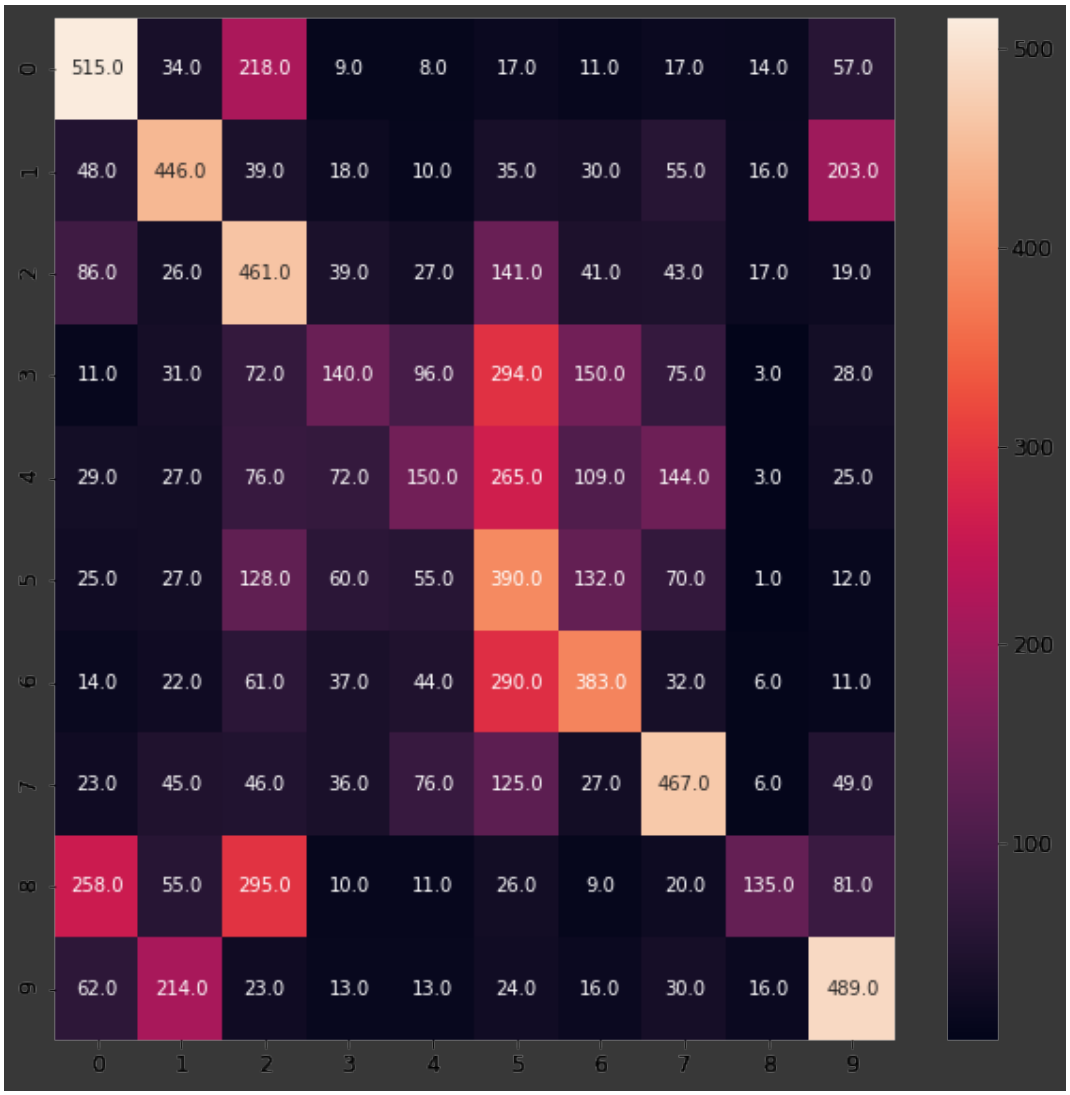
```
[[316  85  61   4   6   1  10  15 389  13]
 [ 16 674   5   9   6   2  23  38 114  13]
 [ 25  56 148  69  90  30  70  48 361   3]
 [  0  62  11 324 149  55 155  94  48   2]
 [  5  53  14 183 215  37 104 235  50   4]
 [  6  28  34 265 141  92 138 102  90   4]
 [  1  53  18 128  70  37 451  50  91   1]
 [  8  59   5  76  69  21  29 574  53   6]
 [ 24  79   9  12   6   5  11  21 725   8]
 [ 15 496   3  12   8   0  11  13 179 163]]
airplane. acc: 35.11%
automobile. acc: 74.89%
bird. acc: 16.44%
cat. acc: 36.00%
deer. acc: 23.89%
dog. acc: 10.22%
frog. acc: 50.11%
horse. acc: 63.78%
ship. acc: 80.56%
truck. acc: 18.11%
```



3 conv 16 (3,3) SGD 'he_uniform' Relu

```
[1.610318660736084, 0.3973333239555359]
```

```
[[515  34 218   9   8  17  11  17  14  57]
 [ 48 446  39  18  10  35  30  55  16 203]
 [ 86  26 461  39  27 141  41  43  17  19]
 [ 11  31  72 140  96 294 150  75   3  28]
 [ 29  27  76  72 150 265 109 144   3  25]
 [ 25  27 128  60  55 390 132  70   1  12]
 [ 14  22  61  37  44 290 383  32   6  11]
 [ 23  45  46  36  76 125  27 467   6  49]
 [258  55 295  10  11  26   9  20 135  81]
 [ 62 214  23  13  13  24  16  30  16 489]]
airplane. acc: 57.22%
automobile. acc: 49.56%
bird. acc: 51.22%
cat. acc: 15.56%
deer. acc: 16.67%
dog. acc: 43.33%
frog. acc: 42.56%
horse. acc: 51.89%
ship. acc: 15.00%
truck. acc: 54.33%
```



3 conv 32 (5,5) SGD 'he_uniform' Relu

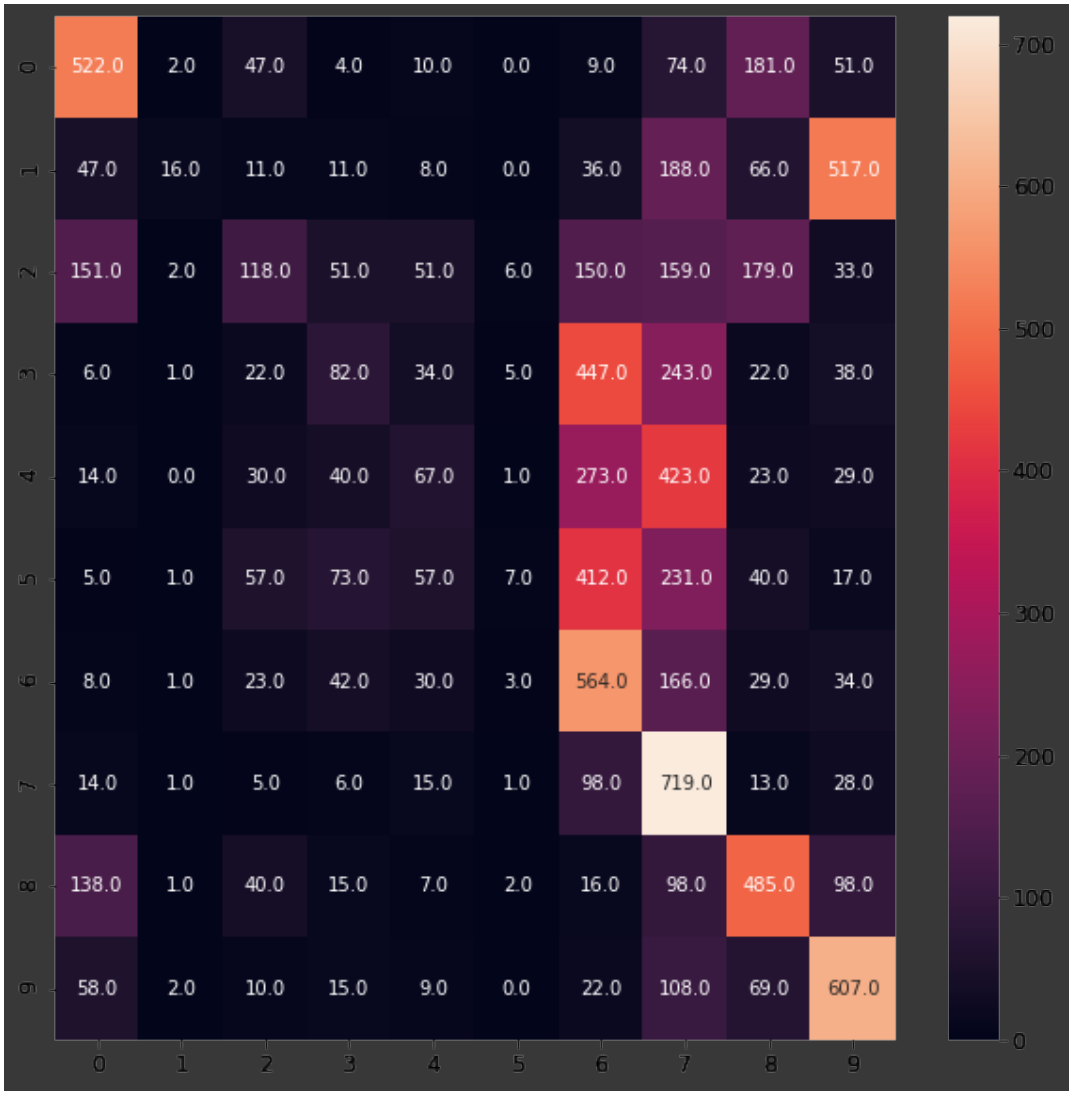
```
[1.908301830291748, 0.308555543422699]

[[610  50  76   9   0   0   2  17  87  49]
 [137 498  33  32   0   0   4  25  44 127]
 [257 116 263  58   2   1  56  26  86  35]
 [ 25 182 176 217   3   0 171  59  27  40]
 [ 84 141 192 141   3   0 141 126  26  46]
 [ 70 103 196 164   3   0 218  72  37  37]
 [ 61 161 135 177   3   1 242  57  37  26]
 [104 149  94  66   6   0  72 325  11  73]
 [339  81  68  14   0   0   1   6 303  88]
 [130 300  28  21   2   0   5   6  92 316]]
airplane. acc: 67.78%
automobile. acc: 55.33%
bird. acc: 29.22%
cat. acc: 24.11%
deer. acc: 0.33%
dog. acc: 0.00%
frog. acc: 26.89%
horse. acc: 36.11%
ship. acc: 33.67%
truck. acc: 35.11%
```

3 conv 64 (5,5) SGD 'he_uniform' Relu

```
[1.6997907161712646, 0.35411110520362854]
```

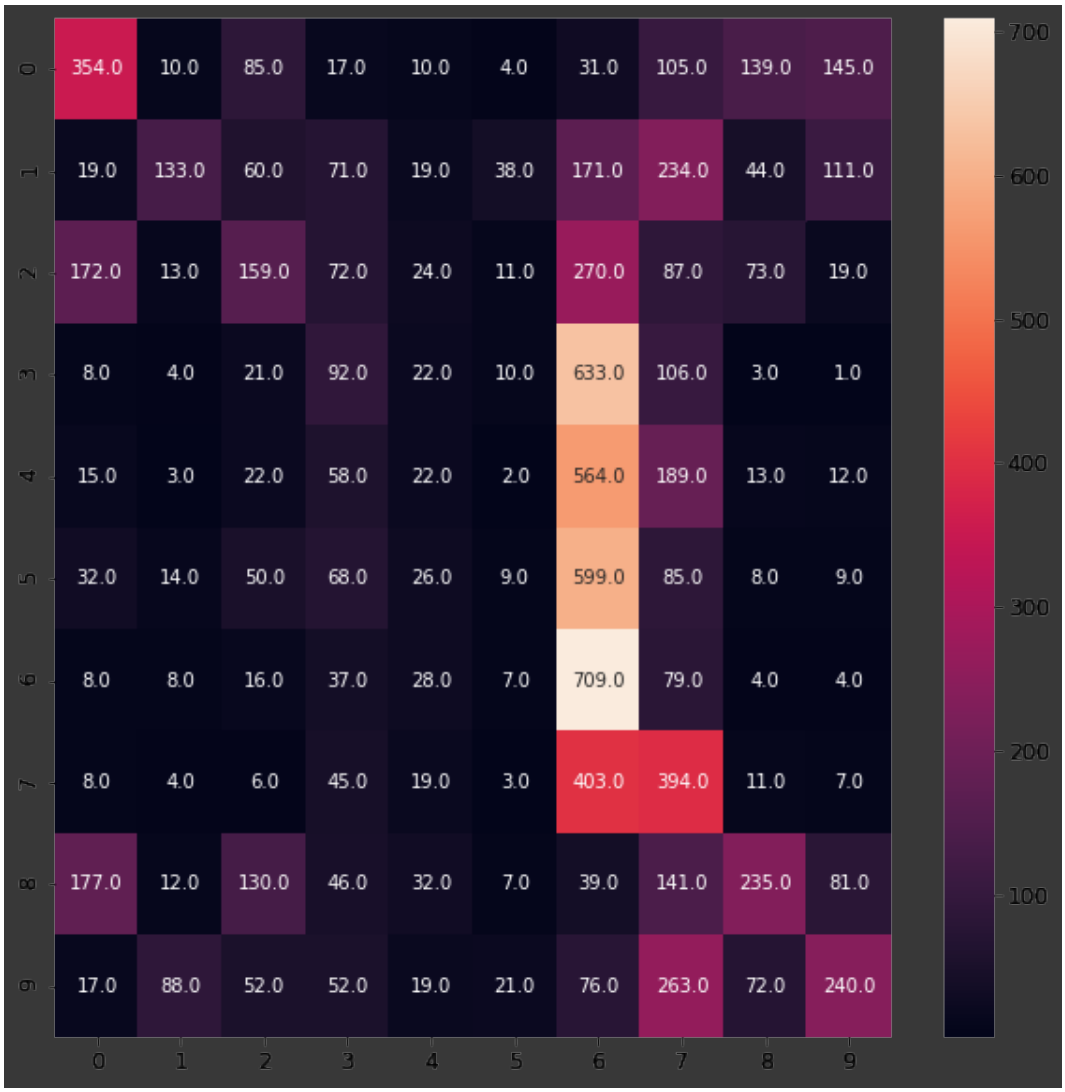
```
[[522   2  47   4  10   0   9  74 181  51]
 [ 47  16  11  11   8   0  36 188  66 517]
 [151   2 118  51  51   6 150 159 179  33]
 [  6   1  22  82  34   5 447 243  22  38]
 [ 14   0  30  40  67   1 273 423  23  29]
 [  5   1  57  73  57   7 412 231  40  17]
 [  8   1  23  42  30   3 564 166  29  34]
 [ 14   1   5   6  15   1  98 719  13  28]
 [138   1  40  15   7   2  16  98 485  98]
 [ 58   2  10  15   9   0  22 108  69 607]]
airplane. acc: 58.00%
automobile. acc: 1.78%
bird. acc: 13.11%
cat. acc: 9.11%
deer. acc: 7.44%
dog. acc: 0.78%
frog. acc: 62.67%
horse. acc: 79.89%
ship. acc: 53.89%
truck. acc: 67.44%
```



3 conv 64 (5,5) SGD 'glorot_normal' Relu

```
[1.981859803199768, 0.2607777714729309]
```

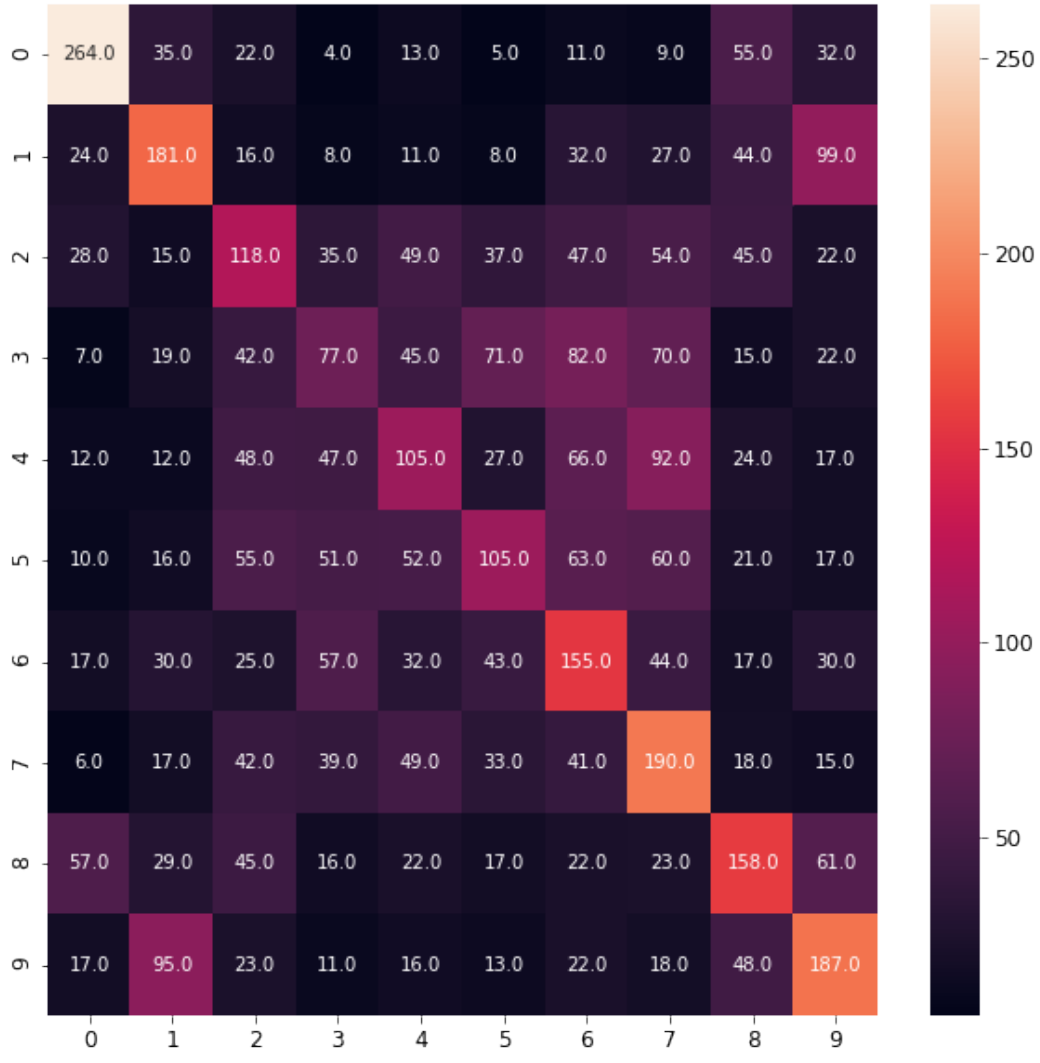
```
[[354  10  85  17  10   4  31 105 139 145]
 [ 19 133  60  71  19  38 171 234  44 111]
 [172  13 159  72  24  11 270  87  73  19]
 [  8   4  21  92  22  10 633 106   3   1]
 [ 15   3  22  58  22   2 564 189  13  12]
 [ 32  14  50  68  26   9 599  85   8   9]
 [  8   8  16  37  28   7 709  79   4   4]
 [  8   4   6  45  19   3 403 394  11   7]
 [177  12 130  46  32   7  39 141 235  81]
 [ 17  88  52  52  19  21  76 263  72 240]]
airplane. acc: 39.33%
automobile. acc: 14.78%
bird. acc: 17.67%
cat. acc: 10.22%
deer. acc: 2.44%
dog. acc: 1.00%
frog. acc: 78.78%
horse. acc: 43.78%
ship. acc: 26.11%
truck. acc: 26.67%
```



3 conv 32 (3,3) Adam 'he_uniform' Relu

```
[4.423496723175049, 0.3422222137451172]
```

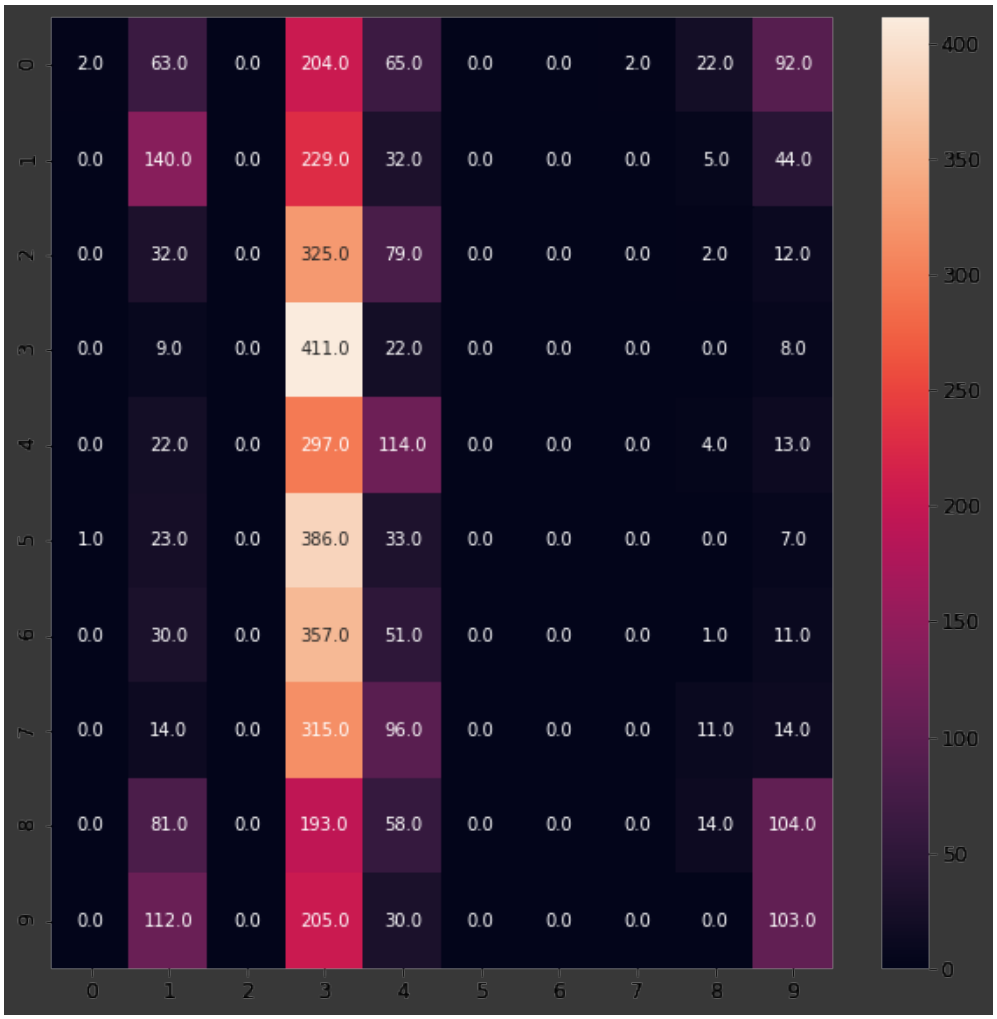
```
[[264  35  22   4  13   5  11   9  55  32]
 [ 24 181  16   8  11   8  32  27  44  99]
 [ 28  15 118  35  49  37  47  54  45  22]
 [  7  19  42  77  45  71  82  70  15  22]
 [ 12  12  48  47 105  27  66  92  24  17]
 [ 10  16  55  51  52 105  63  60  21  17]
 [ 17  30  25  57  32  43 155  44  17  30]
 [  6  17  42  39  49  33  41 190  18  15]
 [ 57  29  45  16  22  17  22  23 158  61]
 [ 17  95  23  11  16  13  22  18  48 187]]
airplane. acc: 58.67%
automobile. acc: 40.22%
bird. acc: 26.22%
cat. acc: 17.11%
deer. acc: 23.33%
dog. acc: 23.33%
frog. acc: 34.44%
horse. acc: 42.22%
ship. acc: 35.11%
truck. acc: 41.56%
```



2 conv 32 (3,3) Adam 'he_uniform' Sigmoid

```
[2.3953604698181152, 0.1742222160100937]
```

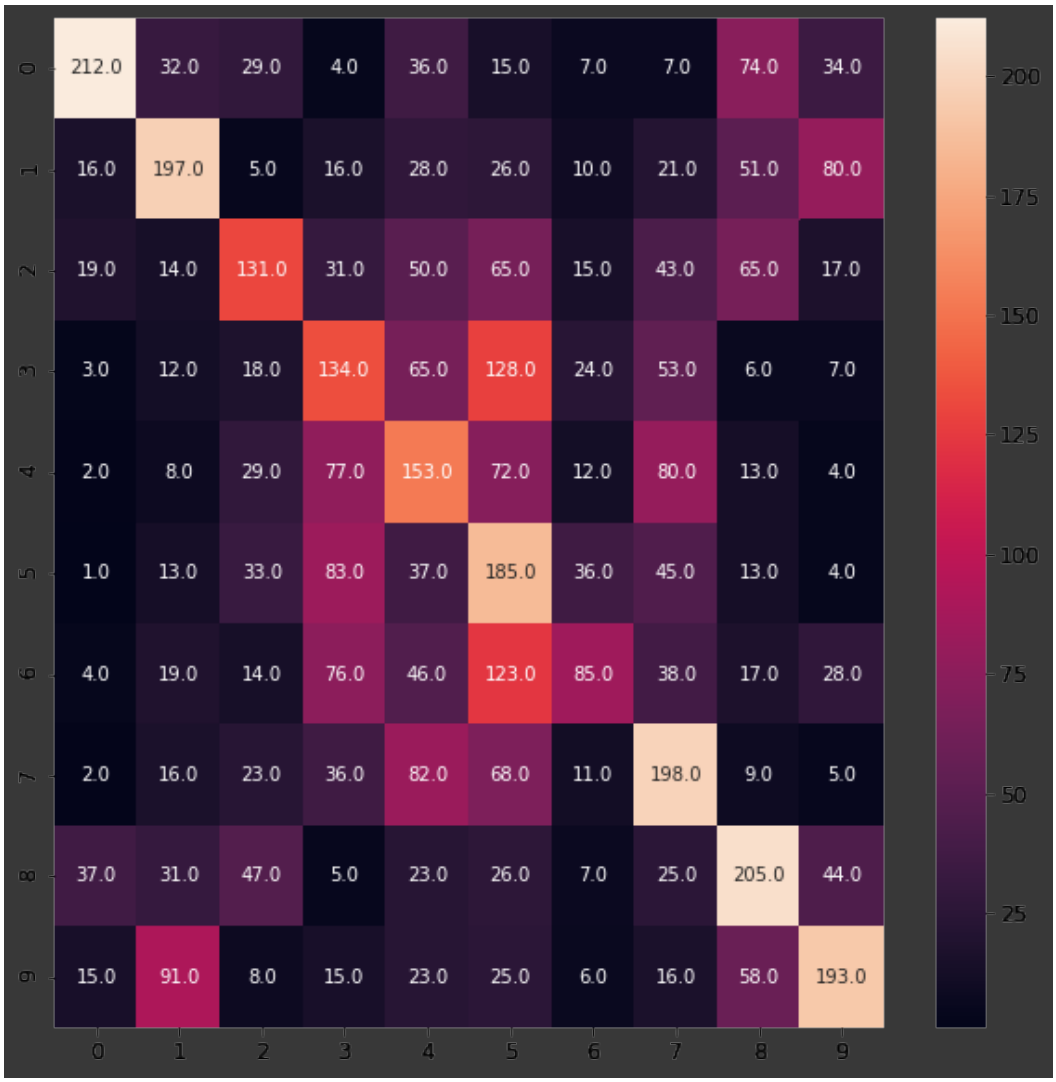
```
[[  2  63   0 204  65   0   0   2  22  92]
 [  0 140   0 229  32   0   0   0   5  44]
 [  0  32   0 325  79   0   0   0   2  12]
 [  0   9   0 411  22   0   0   0   0   8]
 [  0  22   0 297 114   0   0   0   4  13]
 [  1  23   0 386  33   0   0   0   0   7]
 [  0  30   0 357  51   0   0   0   1  11]
 [  0  14   0 315  96   0   0   0  11  14]
 [  0  81   0 193  58   0   0   0  14 104]
 [  0 112   0 205  30   0   0   0   0 103]]
airplane. acc: 0.44%
automobile. acc: 31.11%
bird. acc: 0.00%
cat. acc: 91.33%
deer. acc: 25.33%
dog. acc: 0.00%
frog. acc: 0.00%
horse. acc: 0.00%
ship. acc: 3.11%
truck. acc: 22.89%
```



2 conv 32 (3,3) Adam 'he_uniform' Relu

```
[2.633061170578003, 0.3762222230434418]
```

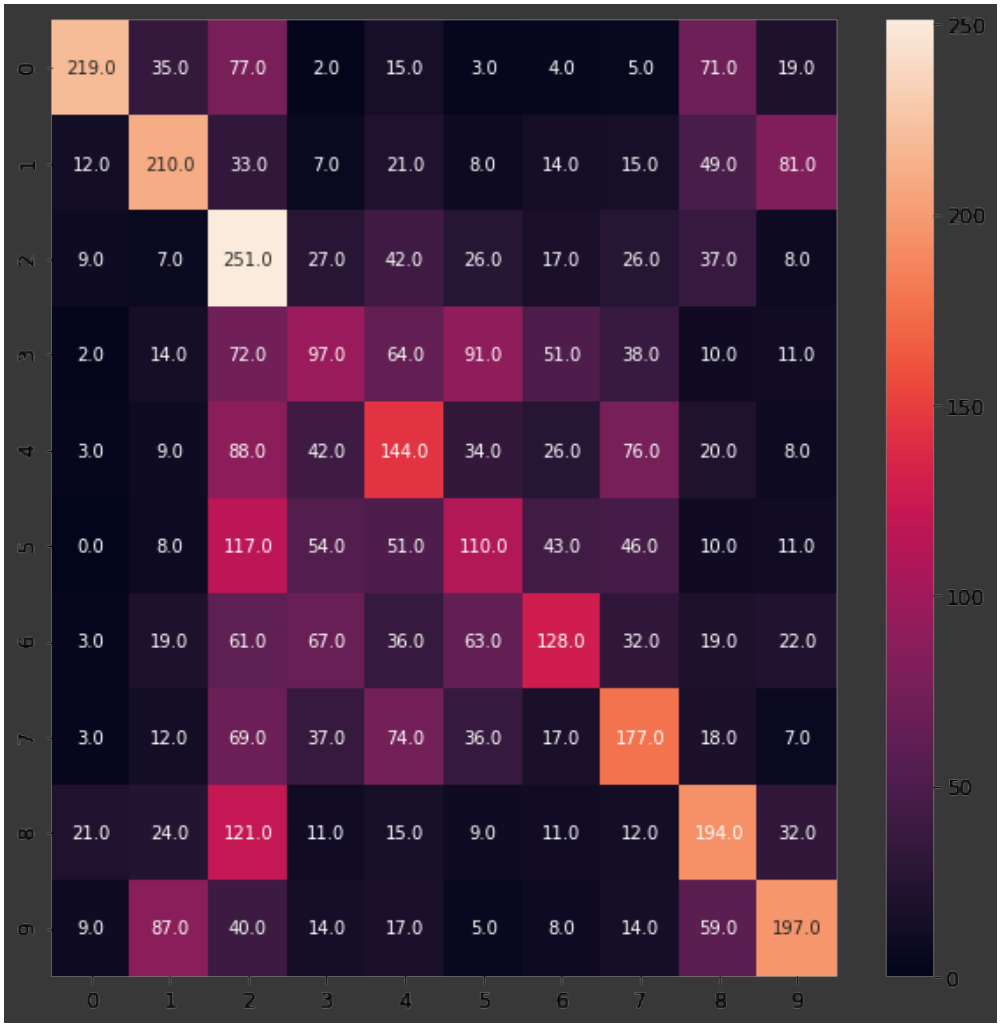
```
[[212  32  29   4  36  15   7   7  74  34]
 [ 16 197   5  16  28  26  10  21  51  80]
 [ 19  14 131  31  50  65  15  43  65  17]
 [  3  12  18 134  65 128  24  53   6   7]
 [  2   8  29  77 153  72  12  80  13   4]
 [  1  13  33  83  37 185  36  45  13   4]
 [  4  19  14  76  46 123  85  38  17  28]
 [  2  16  23  36  82  68  11 198   9   5]
 [ 37  31  47   5  23  26   7  25 205  44]
 [ 15  91   8  15  23  25   6  16  58 193]]
airplane. acc: 47.11%
automobile. acc: 43.78%
bird. acc: 29.11%
cat. acc: 29.78%
deer. acc: 34.00%
dog. acc: 41.11%
frog. acc: 18.89%
horse. acc: 44.00%
ship. acc: 45.56%
truck. acc: 42.89%
```



2 conv 32 (3,3) RMSProp 'he_uniform' Relu

```
[2.633061170578003, 0.3762222230434418]
```

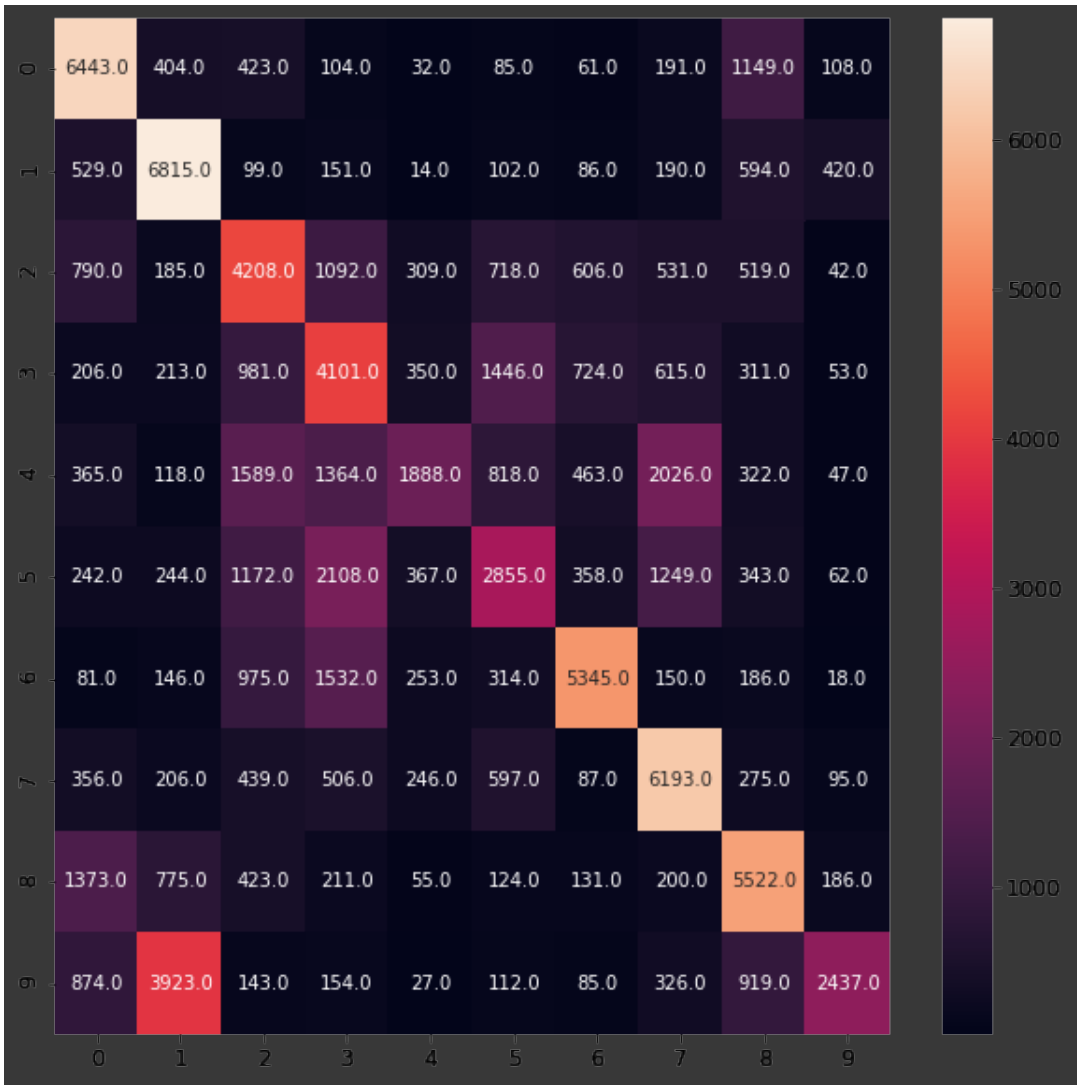
```
[[219  35  77   2  15   3   4   5  71  19]
 [ 12 210  33   7  21   8  14  15  49  81]
 [  9   7 251  27  42  26  17  26  37   8]
 [  2  14  72  97  64  91  51  38  10  11]
 [  3   9  88  42 144  34  26  76  20   8]
 [  0   8 117  54  51 110  43  46  10  11]
 [  3  19  61  67  36  63 128  32  19  22]
 [  3  12  69  37  74  36  17 177  18   7]
 [ 21  24 121  11  15   9  11  12 194  32]
 [  9  87  40  14  17   5   8  14  59 197]]
airplane. acc: 48.67%
automobile. acc: 46.67%
bird. acc: 55.78%
cat. acc: 21.56%
deer. acc: 32.00%
dog. acc: 24.44%
frog. acc: 28.44%
horse. acc: 39.33%
ship. acc: 43.11%
truck. acc: 43.78%
```



Best: 6 conv 32 (3,3) SGD 'he_uniform' Relu

```
[1.3697689771652222, 0.5089666843414307]
```

```
[[6443  404  423  104   32   85   61  191 1149  108]
 [ 529 6815   99  151   14  102   86  190  594  420]
 [ 790  185 4208 1092  309  718  606  531  519   42]
 [ 206  213  981 4101  350 1446  724  615  311   53]
 [ 365  118 1589 1364 1888  818  463 2026  322   47]
 [ 242  244 1172 2108  367 2855  358 1249  343   62]
 [  81  146  975 1532  253  314 5345  150  186   18]
 [ 356  206  439  506  246  597   87 6193  275   95]
 [1373  775  423  211   55  124  131  200 5522  186]
 [ 874 3923  143  154   27  112   85  326  919 2437]]
airplane. acc: 71.59%
automobile. acc: 75.72%
bird. acc: 46.76%
cat. acc: 45.57%
deer. acc: 20.98%
dog. acc: 31.72%
frog. acc: 59.39%
horse. acc: 68.81%
ship. acc: 61.36%
truck. acc: 27.08%
```

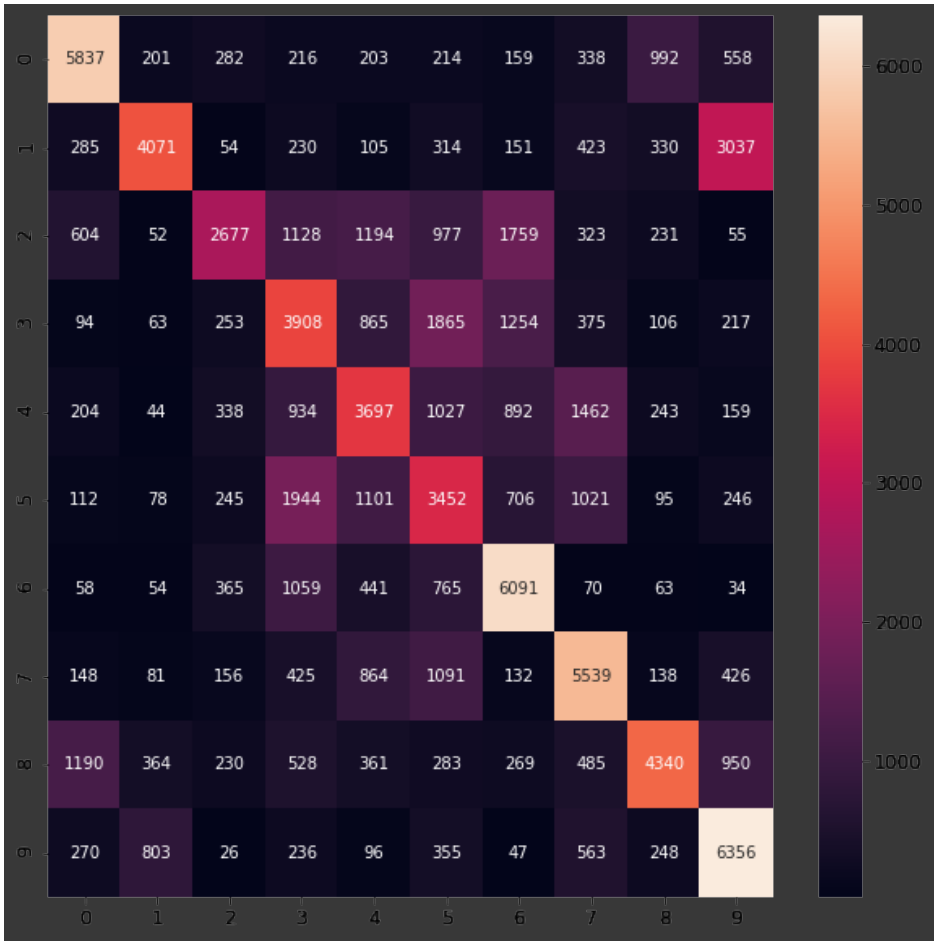


TRANSFER LEARNING

2-a) VGG Son katman eklendi eğitildi

```
[1.3608131408691406, 0.5107555389404297]
```

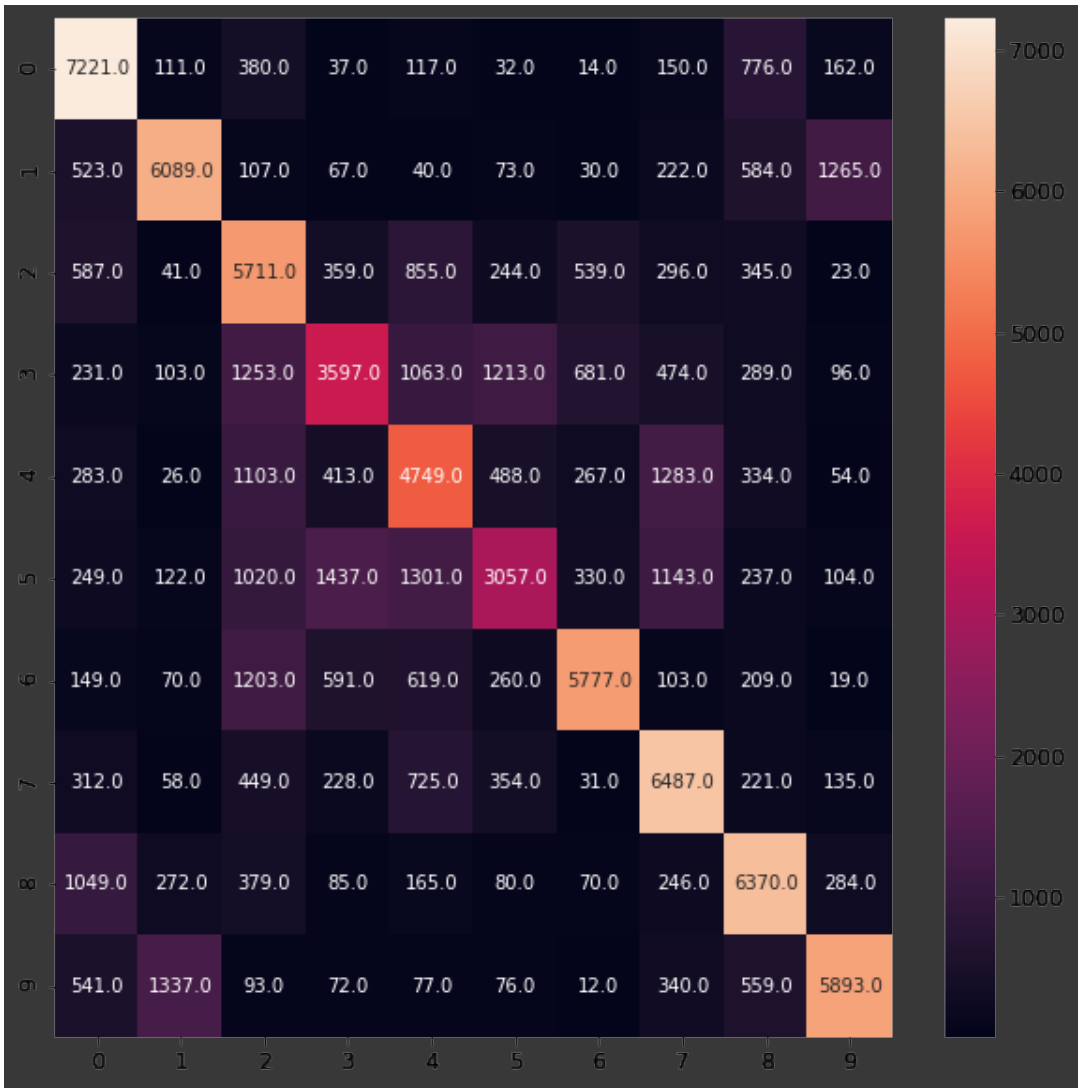
```
[[5837  201  282  216  203  214  159  338  992  558]
 [ 285 4071   54  230  105  314  151  423  330 3037]
 [ 604   52 2677 1128 1194  977 1759  323  231   55]
 [  94   63  253 3908  865 1865 1254  375  106  217]
 [ 204   44  338  934 3697 1027  892 1462  243  159]
 [ 112   78  245 1944 1101 3452  706 1021   95  246]
 [  58   54  365 1059  441  765 6091   70   63   34]
 [ 148   81  156  425  864 1091  132 5539  138  426]
 [1190  364  230  528  361  283  269  485 4340  950]
 [ 270  803   26  236   96  355   47  563  248 6356]]
airplane. acc: 64.86%
automobile. acc: 45.23%
bird. acc: 29.74%
cat. acc: 43.42%
deer. acc: 41.08%
dog. acc: 38.36%
frog. acc: 67.68%
horse. acc: 61.54%
ship. acc: 48.22%
truck. acc: 70.62%
```



2-b) VGG Son katman eklendi 4 katman eğitildi

```
[1.116951584815979, 0.6105666756629944]
```

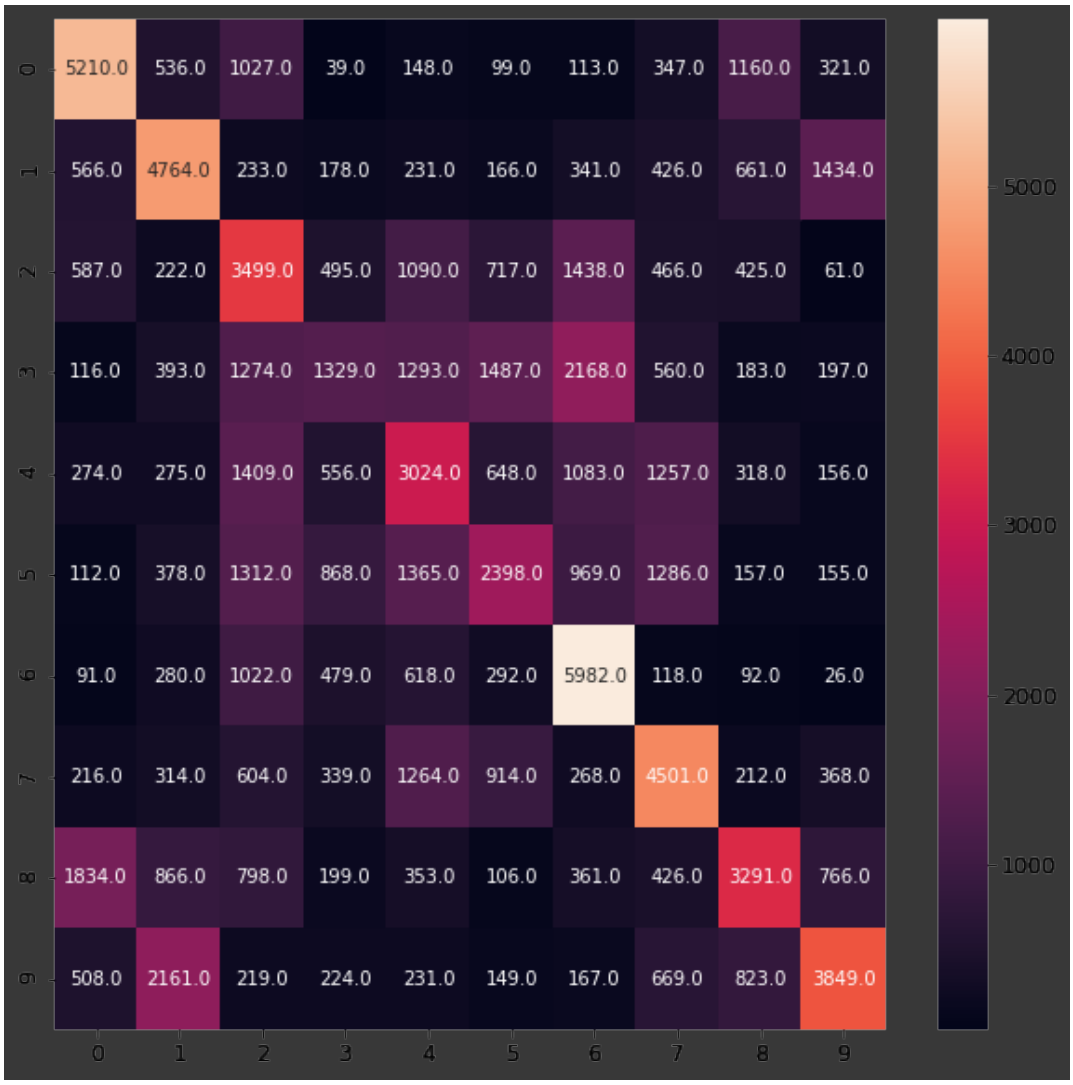
```
[[7221  111  380   37  117   32   14  150  776  162]
 [ 523 6089  107   67   40   73   30  222  584 1265]
 [ 587   41 5711  359  855  244  539  296  345   23]
 [ 231  103 1253 3597 1063 1213  681  474  289   96]
 [ 283   26 1103  413 4749  488  267 1283  334   54]
 [ 249  122 1020 1437 1301 3057  330 1143  237  104]
 [ 149   70 1203  591  619  260 5777  103  209   19]
 [ 312   58  449  228  725  354   31 6487  221  135]
 [1049  272  379   85  165   80   70  246 6370  284]
 [ 541 1337   93   72   77   76   12  340  559 5893]]
airplane. acc: 80.23%
automobile. acc: 67.66%
bird. acc: 63.46%
cat. acc: 39.97%
deer. acc: 52.77%
dog. acc: 33.97%
frog. acc: 64.19%
horse. acc: 72.08%
ship. acc: 70.78%
truck. acc: 65.48%
```



2-a) ResNet son katman eklendi eğitildi

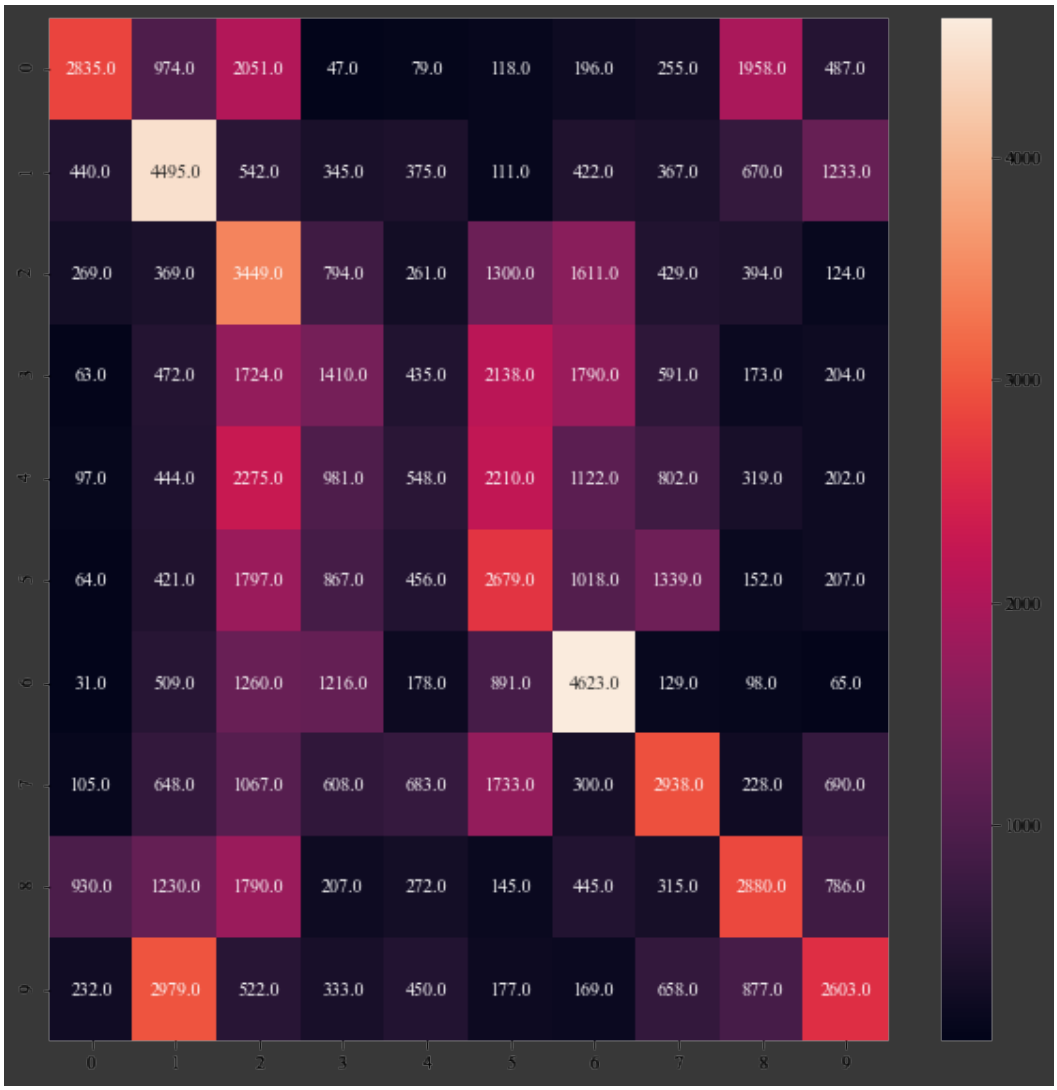
```
[1.5943951606750488, 0.42052221298217773]
```

```
[[5210  536 1027   39  148   99  113  347 1160  321]
 [ 566 4764  233  178  231  166  341  426  661 1434]
 [ 587  222 3499  495 1090  717 1438  466  425   61]
 [ 116  393 1274 1329 1293 1487 2168  560  183  197]
 [ 274  275 1409  556 3024  648 1083 1257  318  156]
 [ 112  378 1312  868 1365 2398  969 1286  157  155]
 [  91  280 1022  479  618  292 5982  118   92   26]
 [ 216  314  604  339 1264  914  268 4501  212  368]
 [1834  866  798  199  353  106  361  426 3291  766]
 [ 508 2161  219  224  231  149  167  669  823 3849]]
airplane. acc: 57.89%
automobile. acc: 52.93%
bird. acc: 38.88%
cat. acc: 14.77%
deer. acc: 33.60%
dog. acc: 26.64%
frog. acc: 66.47%
horse. acc: 50.01%
ship. acc: 36.57%
truck. acc: 42.77%
```



2-b) ResNet son katman eklendi son 4 katman eğitildi

```
loss: 1.8503 - accuracy: 0.3162
[[2835  974 2051   47   79  118  196  255 1958  487]
 [ 440 4495  542  345  375  111  422  367  670 1233]
 [ 269  369 3449  794  261 1300 1611  429  394  124]
 [  63  472 1724 1410  435 2138 1790  591  173  204]
 [  97  444 2275  981  548 2210 1122  802  319  202]
 [  64  421 1797  867  456 2679 1018 1339  152  207]
 [  31  509 1260 1216  178  891 4623  129   98   65]
 [ 105  648 1067  608  683 1733  300 2938  228  690]
 [ 930 1230 1790  207  272  145  445  315 2880  786]
 [ 232 2979  522  333  450  177  169  658  877 2603]]
airplane. acc: 31.50%
automobile. acc: 49.94%
bird. acc: 38.32%
cat. acc: 15.67%
deer. acc: 6.09%
dog. acc: 29.77%
frog. acc: 51.37%
horse. acc: 32.64%
ship. acc: 32.00%
truck. acc: 28.92%
```



Yorum: Sıfırdan eğitilen toplam 18 modelin sonuçları karşılaştırıldığında, sadece tek bir parametre değiştirilip diğer parametreler sabit bırakıldığı durumlarda alınan en iyi sonuç, konvolüsyonel katman sayısının 6 olduğudu durumdadır. Ödev güncellendikten sonraki hali ile 1,2,3 sayılı konvolüsyonel katmana sahip modeller arasından en iyi sonuç veren 3 katmanlı modellerdir.

Yapılan denemeler değerlendirildiğinde en iyi sonuç veren parametre değerleri 3 konvolüsyon katmanı, 32 filtre sayısı, (3,3) filtre boyutu, GlorotNormal ya da he_uniform kernel initializer, Relu Aktivasyon fonksiyonu, 0.2 drop out oranı, Adam ve SGD optimizasyon algoritmalarıdır.

Drop out oranının yüksek seçilmesi, modelin fazla unutmasına yol açtığı için öğrenimdeki verimliliği düşürdüğü gözlemlenmiştir.

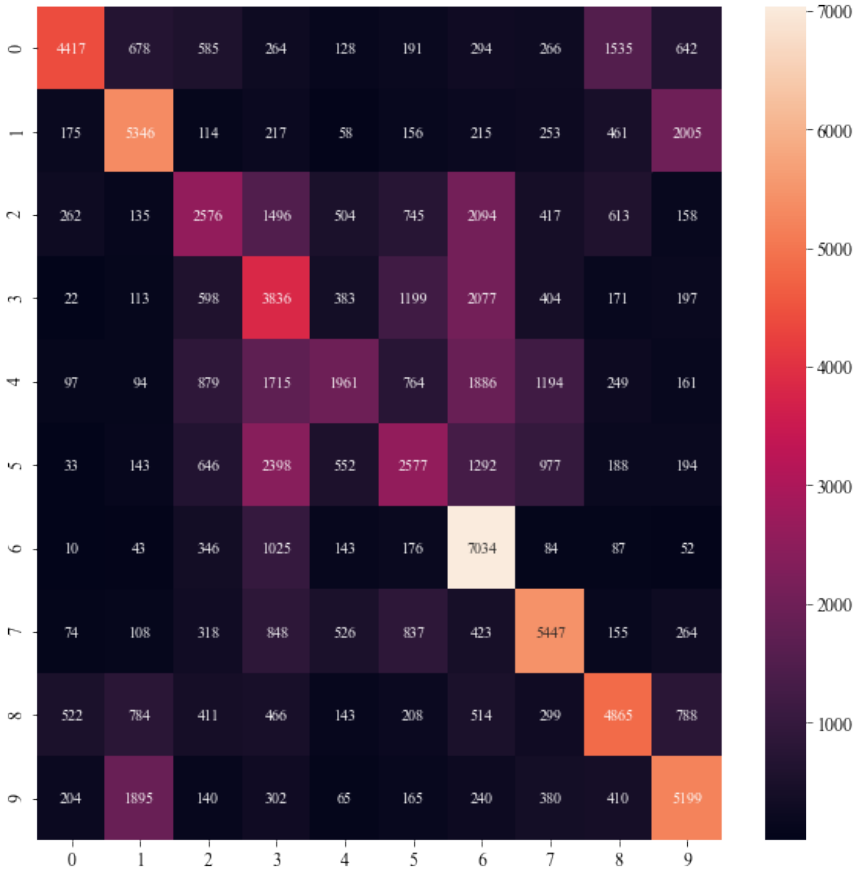
Kernel initializer olarak he uniform ve Glorot Normal, Optimizasyon Algoritması olarak Adam ve SGD birbirine yakın sonuçlar vermişlerdir.

En başarılı seçilen model: Katman Her katman için 32 filtre 3x3 filtre boyutlu GlorotNormal ReLu Aktivasyon 0.2 Dropout Oranı Adam Optimizasyon Algoritması

Confusion Matrix: Sıfırdan eğitilmiş en başarılı sonuç veren modele ait confusion matrix sonucu aşağıdaki şekilde görüldüğü gibidir.

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog
', 'frog', 'horse', 'ship', 'truck']
```

Aşağıda gösterilen sonucu yorumlamak istersek örneğin Airplane en fazla ship ile karıştırılmıştır (1535 adet airplane, ship olarak tahmin edilmiş.) Sınıflara ait en fazla yanlış tahmin edilen sınıflar: (Airplane-Ship, Automobile-Truck, Bird-Frog, Cat-Frog, Deer-Frog, Dog-Cat, Frog-Cat, Horse-Cat, Ship-Automobile, Truck-Automobile)



Bu sonuçlara göre, Araba ve kamyon, kedi ve köpek, birbirlerine yakın görüntüler içeren sınıflar oldukları için tahmin sonuçları da bu yönde etkilenmiştir.Modellerin genelinde gözlenen bu gibi yanlış sınıflandırmaların, sınıfların birbirine benzerliği yüzünden olduğu düşünülmüştür.

Öğrenim aktarımı yöntemiyle eğitilmiş en başarılı model, son 4 katmanı eğitilmiş VGG16 modelidir.Bunun sebebinin VGG16 modelinin çok sayıda eğitilmiş ve ağırlıkları kaydedilmiş parametresinin olmasıdır. VGG16'nın kendi total parametre sayısı 138,357,544 iken ResNet50'nin 25,636,712'dir. Buna göre VGG16 modelinin daha önceden eğitilmiş parametre sayısı olarak bariz bir avantajı vardır. Ayrıca VGG modelleri arasında, son 1 katman yerine son 4 katman eğitime açıkken daha iyi sonuç alınmasının sebebi, VGG16 modelinin son katmanlarındaki fully connected layerların daha iyi öğrenebilmesi olduğu düşünülmüştür. Bu sayede orijinal ağırlıklar ve son 4 katmanın eğitilmesi sonucunda oluşan ağırlıklar kullanılarak en optimal model oluşturulmuş ve test edildiğinde 61% oranında başarı sağlanmıştır.

Öğrenim Aktarımı Modelinin Özniteliklerini Kullanarak Görüntü Erişiminin Gerçeklenmesi

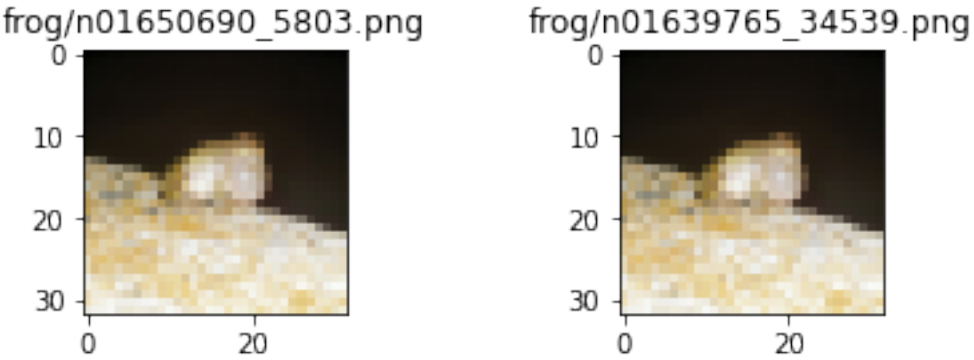
Öğrenim aktarımı yöntemiyle eğitilmiş 4 model için verisetindeki her fotoğrafın ayrı ayrı öznitelikleri çıkartılmıştır. 4 modele ait 4 ayrı öznitelik vektörü, 10 sınıf, her sınıfa ait 3 test, her teste ait en yakın 5 fotoğraf toplam 720 fotoğraf etmektedir. Bu yüzden yapılan testlerde en yüksek başarıyı veren, öğrenim aktarımı ile sadece son katmanı eğitilmiş VGG modelinden çıkartılmış öznitelikler kullanılmıştır.

10 sınıf için her sınıfa ait 3 test fotoğrafı ile en yüksek benzerliğe sahip 5 fotoğraf bulunmuştur.

Toplam 30 test fotoğrafı için en benzer fotoğraflar gösterilmiştir.

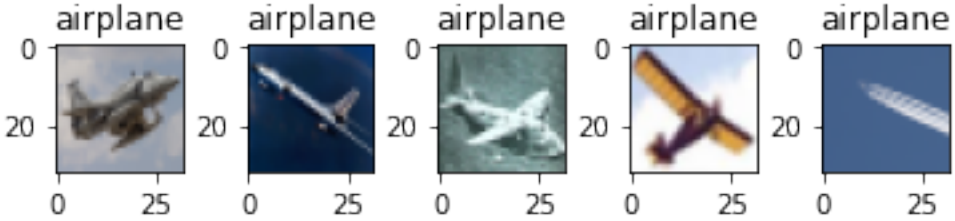
Bazı test fotoğrafları ile en yakın benzerliği bulunan fotoğrafın aynı olduğu gözlemlenmiştir. Bunun sebebi bazı fotoğrafların farklı isimlerle birden fazla kez veri setinde bulunmasıdır.

Örneğin:

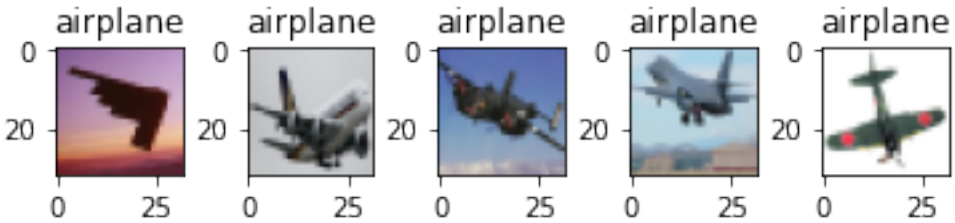


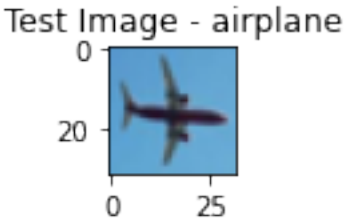
Testler:

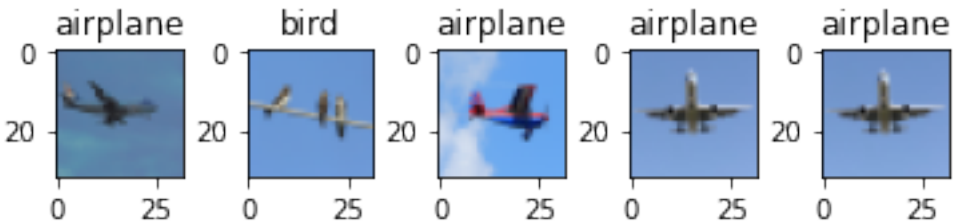

Test Image - airplane
0
20
0
25

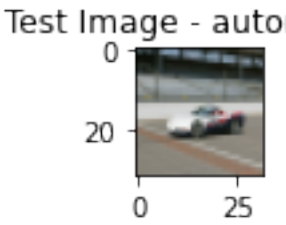

airplane
airplane
airplane
airplane
airplane

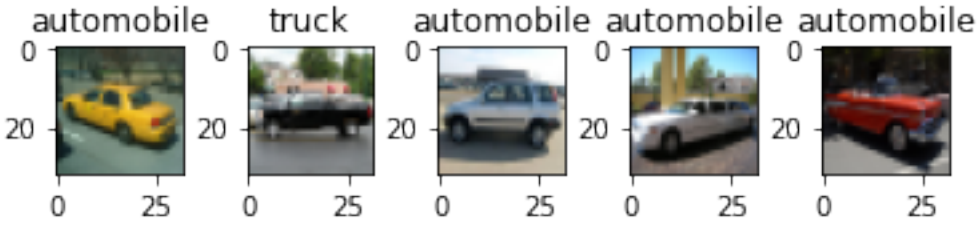

Test Image - airplane
0
20
0
25

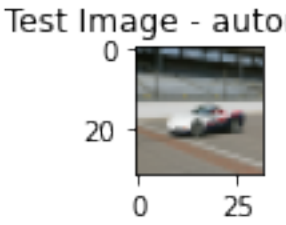

airplane
airplane
airplane
airplane
airplane

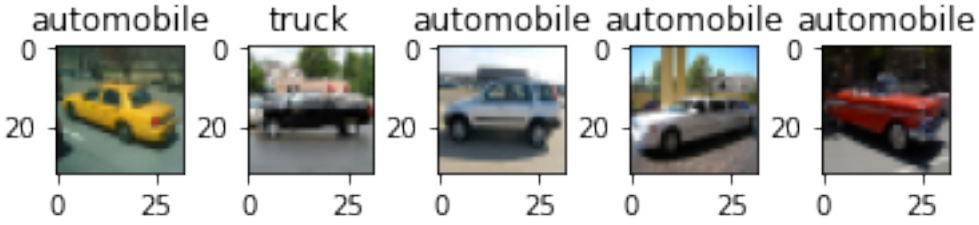

Test Image - airplane
0
20
0
25

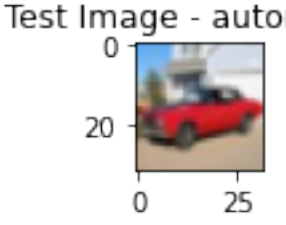

airplane
bird
airplane
airplane
airplane

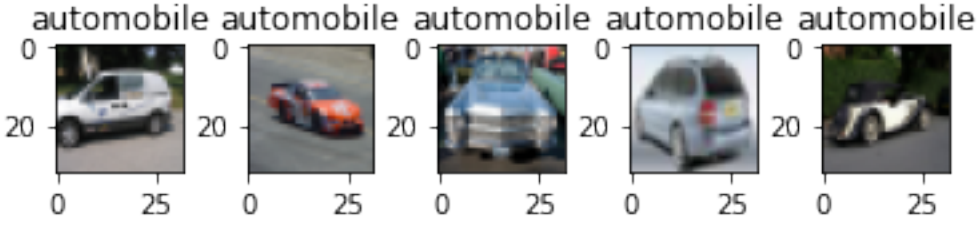

Test Image - automobile
0
20
0
25


automobile
truck
automobile
automobile
automobile
0
20
0
25

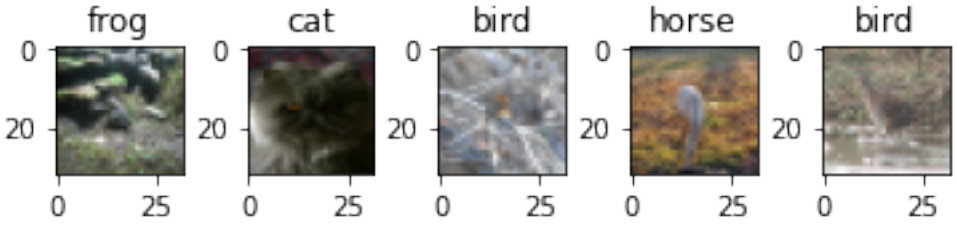

Test Image - automobile
0
20
0
25


automobile
truck
automobile
automobile
automobile
0
20
0
25

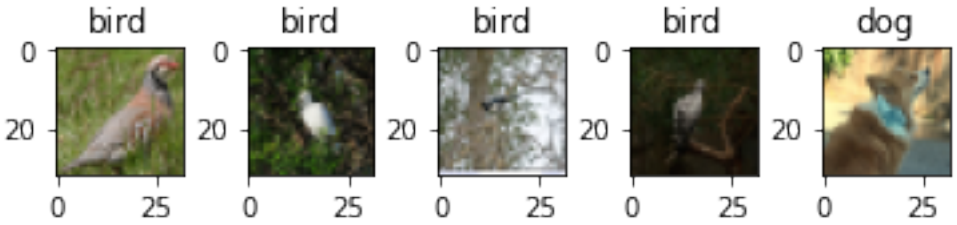

Test Image - automobile
0
20
0
25


automobile
automobile
automobile
automobile
automobile
0
20
0
25


Test Image - bird
0
20
0
25


frog
cat
bird
horse
bird
0
20
0
25


Test Image - bird
0
20
0
25


bird
bird
bird
bird
dog
0
20
0
25

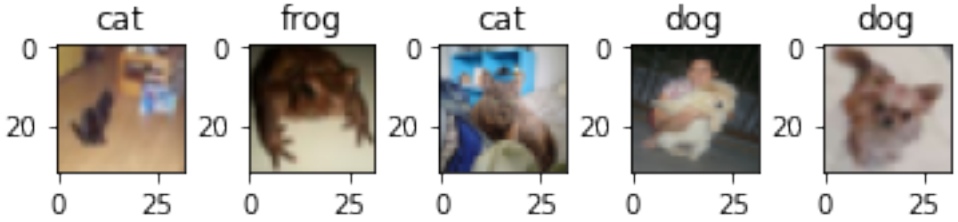

Test Image - bird
0
20
0
25


frog
cat
bird
deer
frog
0
20
0
25

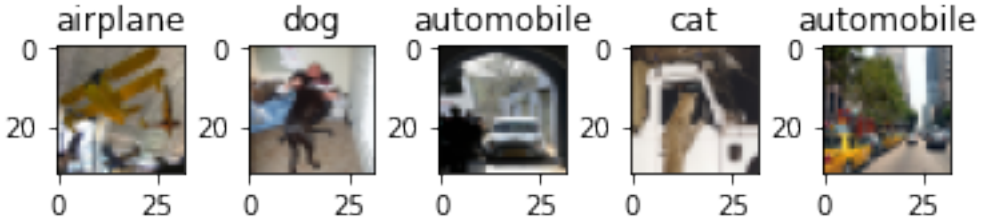

Test Image - cat
0
20
0
25


cat
cat
cat
cat
dog
0
20
0
25


Test Image - cat
0
20
0
25

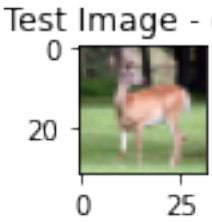

cat
frog
cat
dog
dog
0
20
0
25


Test Image - cat
0
20
0
25


airplane
dog
automobile
cat
automobile
0
20
0
25


Test Image - deer
0
20
0
25


cat
dog
bird
deer
cat
0
20
0
25

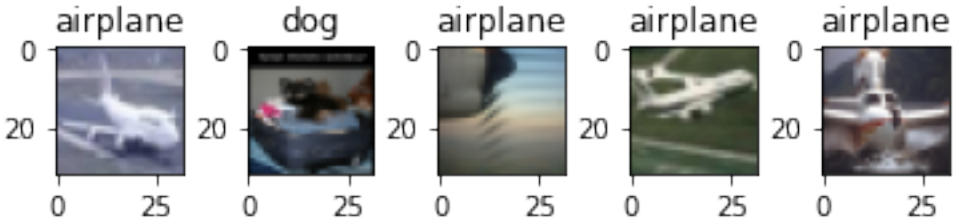

Test Image - deer
0
20
0
25


dog
horse
horse
horse
horse
0
20
0
25

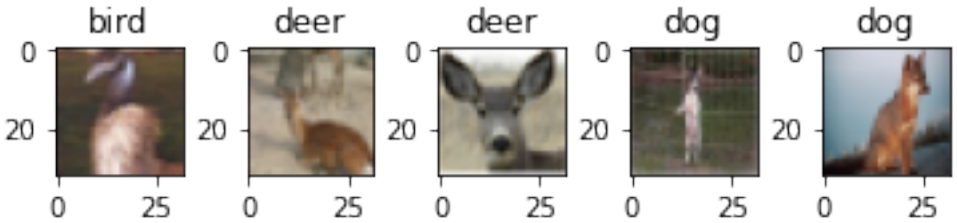

Test Image - deer
0
20
0
25


horse
deer
dog
deer
deer
0
20
0
25


Test Image - dog
0
20
0
25


airplane
dog
airplane
airplane
airplane
0
20
0
25

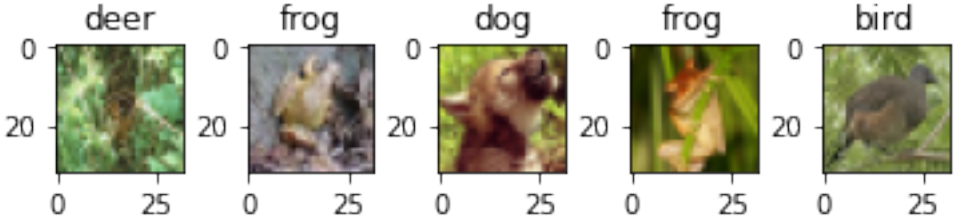

Test Image - dog
0
20
0
25


bird
deer
deer
dog
dog
0
20
0
25


Test Image - dog
0
20
0
25


horse
cat
horse
dog
cat
0
20
0
25

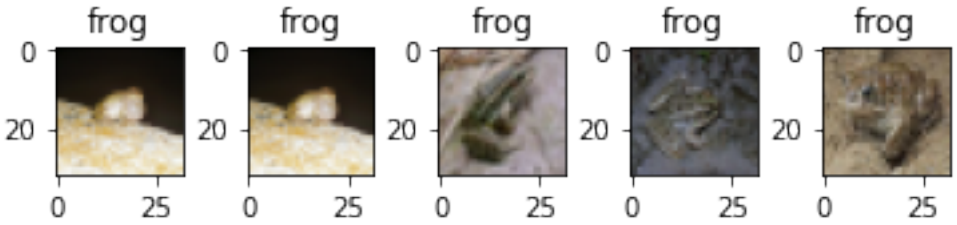

Test Image - frog
0
20
0
25

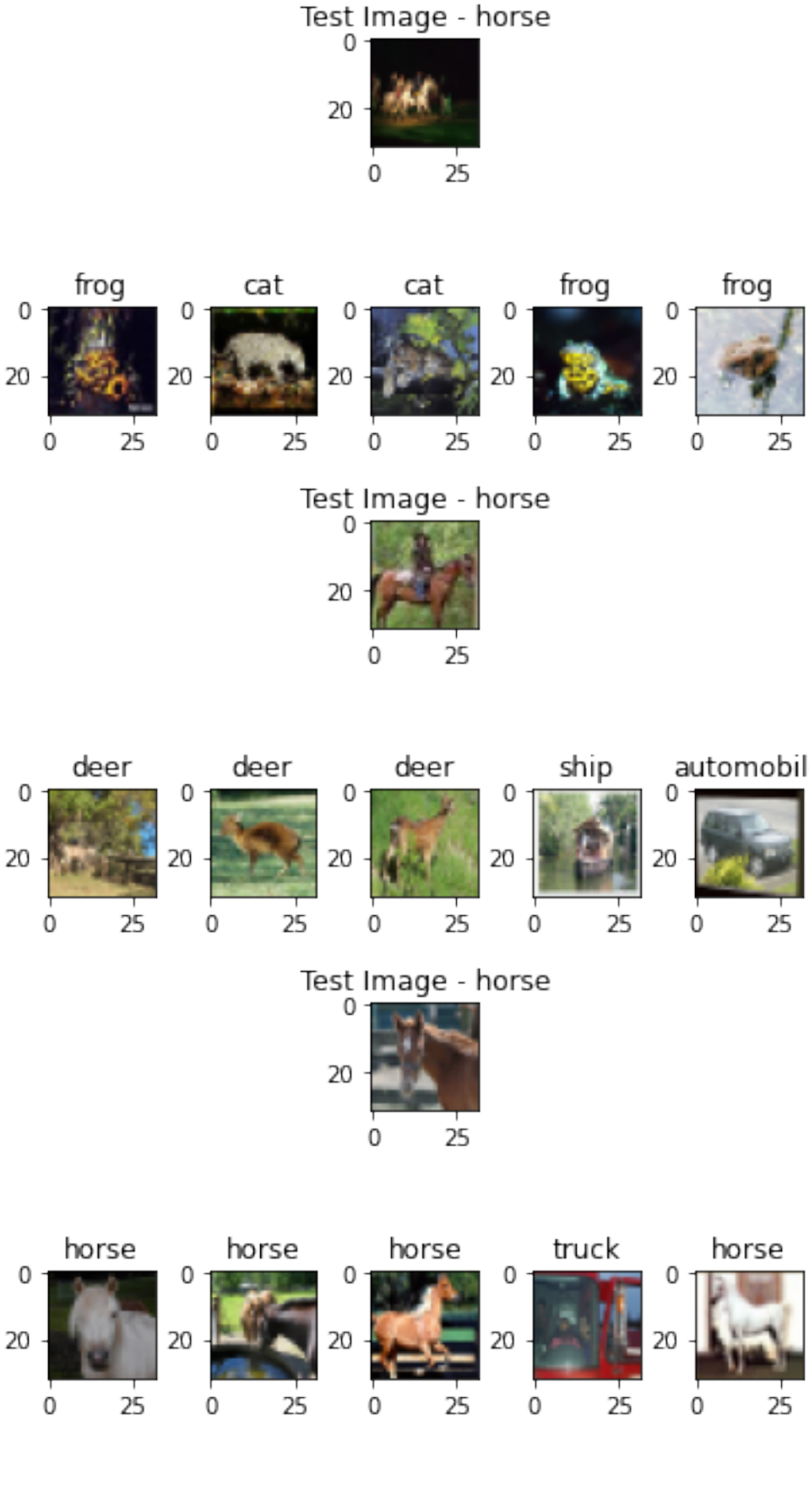

deer
frog
dog
frog
bird
0
20
0
25


Test Image - frog
0
20
0
25

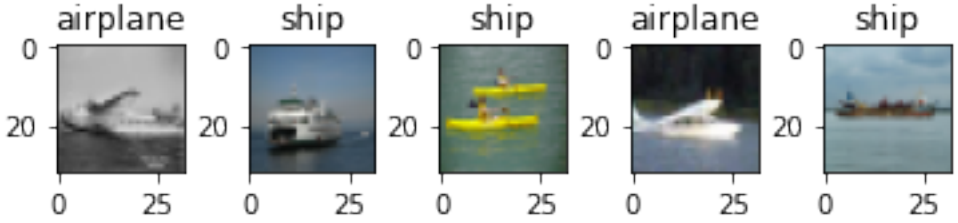

frog
frog
frog
deer
frog
0
20
0
25


Test Image - frog
0
20
0
25

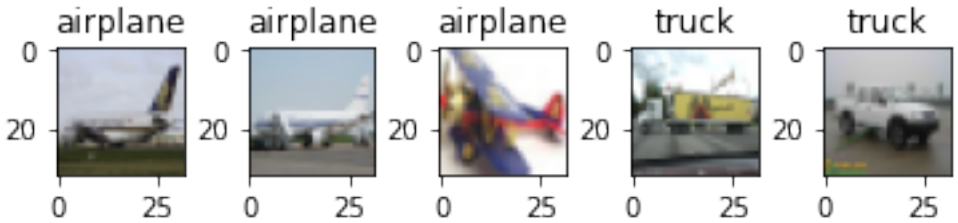

frog
frog
frog
frog
frog
0
20
0
25


Test Image - horse
frog
cat
cat
frog
frog
Test Image - horse
deer
deer
deer
ship
automobile
Test Image - horse
horse
horse
horse
truck
horse

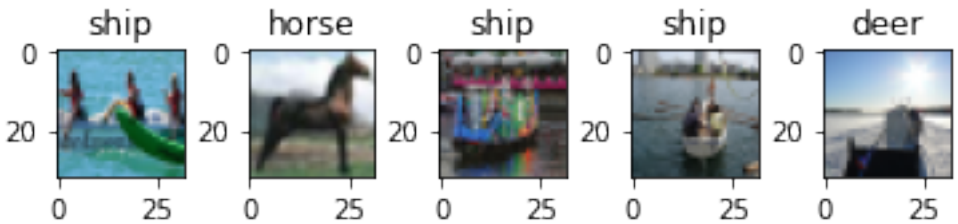

Test Image - ship
0
20
0
25

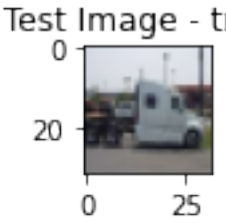

airplane
ship
ship
airplane
ship

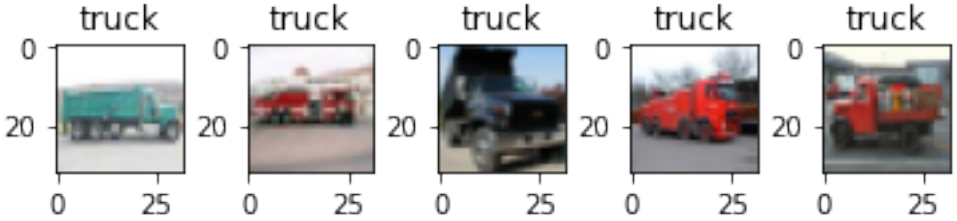

Test Image - ship
0
20
0
25


airplane
airplane
airplane
truck
truck

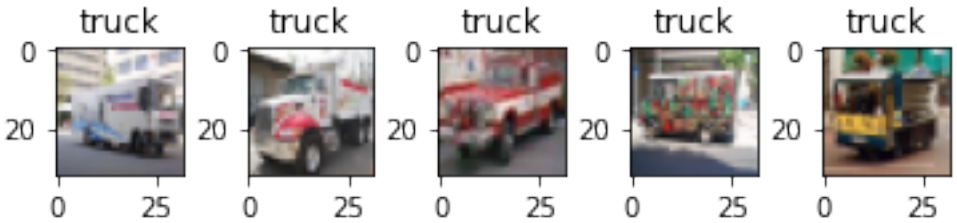

Test Image - ship
0
20
0
25

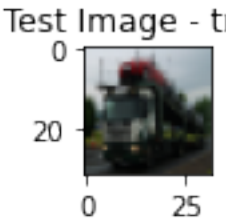

ship
horse
ship
ship
deer


Test Image - truck
0
20
0
25


truck
truck
truck
truck
truck
0
20
0
25


Test Image - truck
0
20
0
25


truck
truck
truck
truck
truck
0
20
0
25


Test Image - truck
0
20
0
25


truck
truck
truck
truck
truck
0
20
0
25

SONUÇ

Bu çalışma kapsamında konvolüsyonel sinir ağı yapısı, katmanların önemi ve kullanım şekilleri, hiper parametre analizi gibi konularda deneyim edindim. Ayrıca Öğrenim aktarımı methoduyla hazır bir modeli tekrar kendi verisetimizde nasıl eğitebileceğimizi öğrenme fırsatım oldu.

Yapılan denemelerde, eğitilen modellerden en kötü sonuç verenler (5x5) filtre boyutuna sahip modellerdi. Ayrıca drop out oranının yapılan bazı denemelerde 0.7 gibi yüksek bir değerde olması öğrenim performansını düşürerek başarısız sonuçlar alınmasına yol açtı. Genel olarak frog olarak yanlış tahmin edilen çok sayıda test gözlemlendi. Dog sınıfının başarısı diğer sınıflara göre aşağıda kaldı. Bunun sebebinin çok farklı şekil, pozisyon, durumda frog fotoğrafının verisetinde bulunması olabileceği düşünüldü. Dog sınıfınıın düşük başarı almasının sebebi ise, çok kötü kalitede dog fotoğraflarının bulunması ve bunun eğitiminde yüksek oranda gürültüye sebep olması olabilir.

Öğrenim aktarımı yönteminin avantajı olarak sayılabilecek hususların başında, halihazırda büyük veri setleriyle eğitilmiş ve yüksek başarı alınmış modelleri, kendi veri setimize uyarlayarak eskiden öğrendiği ağırlıkları kaybetmeden kullabilmemiz geliyor. Bunun en büyük avantajı, belki de bizim ulaşamayacağımız doğruluk oranını sağlayan ağırlıkların hazır olarak kullanılabilmesidir. Örneğin VGG16 modelinin sadece son 4 katmanını eğiterek, sıfırdan eğitilmiş bütün modellerden daha başarılı sonuçlar alınmıştır.

Bir dezavantaj olarak bahsedilmesi gereken kısım ise, modelin baş ve orta kısmına müdahele edemeyişimiz. Önceden eğitilmiş katmanların ağırlıklarını kullanmak istediğimiz için içerisinde çok bir değişiklik yapamıyoruz. Sadece son kısmına müdahele edebiliyoruz. Her model yapısı her iş için yüksek doğruluk oranı sağlama garantisi vermediği için işimize uygun olmayan bir model yapısı ile öğrenim aktarımı yöntemini uygulamak çok mantıklı olmayabilir. Zira ResNet50 modeli ile yapılan denemeler ile sıfırdan eğitilmiş modeller arasındaki başarı birbirine benzer şekilde ölçüldü. İlk katmanlarda ResNet50 ağırlıklarını kullanmamız, ekstra bir avantaj sağlayamadı.