İSTANBUL TİCARET ÜNİVERSİTESİ

T.C. İSTANBUL TİCARET ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BIL458 BİYOİNFORTİK

DERİN ÖĞRENME AĞLARI İLE COVID-19 Sınıflandırma

Can Okan TAŞKIRAN, 100042773

Proje Sorumlusu: Dr. Arzu Kakışım (İstanbul Ticaret Üniversitesi)

İSTANBUL, 2020

Açıklama

Gerçekleştirmiş olduğumuz çalışma 3 ayrı programdan oluşmaktadır. Bunun sebebi ise bilgisayarım sahip olduğu donanımın derin yapay sinir ağını eğitecek yeterlilikte olmamasıdır. Bu programlar şöyledir:

1. **Preprocessing.py**: Bu programda dateset klasöründe bulunan verilerimiz okunarak belirli ön işleme (preprocess) maruz bırakılır. Burada gerçekleştirilen ön işlem aşağıda daha ayrıntılı olarak anlatılmıştır. Daha sonra bu ön işlemden geçmiş veriler ve bu verilerin sahip oldukları etiketler(labels) NPZ dosya formatına dönüştürülerek daha sonra Google drive yüklenmek üzere bulunduğu dizine kaydedilir.

2. **NameofModel_Num.Class.ipynb:** Bu programın adı sahip olduğu mimariye ve gerçekleştirdiği sınıflandırma sayışa göre şekillenmektedir"Örnek:AlexNet_3class_..". Bu dosyalar Google Colab'ın ücretsiz GPU servisinde kullanmak için tasarlanmıştır. Bu programlarda önce NPZ dosya formatında bulunan veriler ve etiketleri okunur daha sonra train ve test olmak üzere farklı kümeye bölünür. Daha sonra bu veriler tasarlanan model verilerek modelin öğrenmesi gerçekleştirilir. Model öğrenimi sırasında en yüksek validation accuracy'nin elde edildiği ağırlıklar ve eğitim sonunda da model HDF5/H5 formatında kayıt edilir. Bu kaydedilen model isterse bir uygulamamanın ara planında isterse de bizim gibi eğitim sonucunu değerlendirmek için yüklenip kullanılabilir.
   (Eğitim sonucunda kaydedilen modeller ve ağırlıklar test edebilmeniz için ipynb dosyaları ile birlikte verilmiştir.)

3. **performance_model.ipynb**: Bu programda eğitilmiş olan modele test verisi uygulanarak modelin performansı ölçülür. Önce modele test verisi verilir ve tahminler elde edilir. Elde edilen bu tahminler ışığında Roc curveler çizdirilir ve confusion matrixleri oluşturulur.

Devam eden kısımda gerçekleştirilen işlemeler ve Tasarlan Derin sinir ağlarında kullanılan yapılara ilişkin açıklamalar yer verilmiştir.

Ön İşleme

Modeli eğitirken kullanmış olduğumuz veri kümesindeki mr görüntüleri çoğu farklı boyutta sahip olup farklı açılardan elde edilmiştir. Görüntülerin farklı açılarda olması bizim için bir problem doğurmazken, farklı resim boyutlarına sahip olmamız bizim için problem oluşturmaktadır. Veri kümemizdeki her resmi aynı sinir ağına uygulayacağımız için belirlenmiş bir boyuta resimleri kullanmamız gerekmektedir. Görsellerin boyutunu çift yönlü doğrusal interpolasyon yönteminde yararlanarak 350x350 boyutuna indirgeriz. Çift yönlü doğrusal interpolasyon yeniden örnekleme işlemi olup resmin boyunun ve enin farklı

oranlarda olduğu durumlarda bize resimlerimizde herhangi bir deformasyon oluşmadan boyutunda indirgeme yapmamıza olanak sağlamaktadır. Bazı durumlarda ise resim gri resme dönüştürülüp eğitim verilmiştir. Resmimiz 3 renk kanalından oluşuyordu bu renk kanalarından kurtularak resmimiz sadece 1 kanaldan oluşan pikselleri 0-255 arasında değerler alabilen bir gri resim halin getiriyoruz. Bunla da resmi ışığın vurma açası, parlaklık veya imzanın atıldığı kalemin rengi gibi parametrelerden bağımsız hale getiriyoruz. Ayrıca buna ek olarak 3 kanal üzerinde çalışmak yerine tek kanal üzerinde çalışarak hesaplama maliyetimizi de düşürmüş oluyoruz.

En sonunda ise görselleri eğitime vermeden normalizasyon uygularız. Böylelikle değerlerimiz [0-1] aralığına çekilmiş olur. Böylelikle maliyet fonksiyonun (cost fuction) yükü azalmış olur ve eğitim daha kolay hale gelir.

Evrişimsel Sinir Ağları (Convolutional Neural Networks)

Evrişimsel Sininr Ağları (Convolutional Neural Networks-CNNs) ya da diğer adıyla Konvolüsyonel Sinir Ağları görüntü tanıma ve nesne sınıflandırma gibi problemler kullanılan bir derin öğrenme ağ mimarisidir. Doğrudan girişine verilen resim, el yazısı, ses gibi bilgilerden öğrenen, bu bilgilerden özellik çıkarımı yapabilen çok katmanlı bir ağdır. Bu mimari geleneksel derin öğrenme mimarilerinde olduğu gibi giriş ve çıkış katmanlarına sahiptir, bunun dışında ağ üç önemli katmanda meydana gelmektedir. Bunlar konvolüsyon katmanı, havuzalama katmanı ve tam bağlı katmandır. Bu katmanlara çeşitli ara katmanlar ve ara işlemler eklenebilir fakat en temel anlamda ağın bu üç katmandan meydana gelmesi gerekmektedir.

Böyle bir mimari kullanmadan bu işlemi gerçekleştirdiğimiz düşünecek olursak ağın girişlerin verdiğimiz 155x200x3 boyutlarında biri imza resmi için yapay sinir ağı resmin her pikseline bir nöron atayacak ve ağın girişi 93.000 adet ağırlıktan oluşacaktı. Bu miktar ağın eğitime de göz önünde bulundurulduğunda sadece bir katman için bile ağa korkunç bir işlem yükü getirecektir. Oysa Evrişimsel sinir ağlarında resimlerden çıkarılan özellikler üzerinden eğitim gerçekleştirilmektedir.

1. **Konvolüsyon Katmanı (Convolutional Layer)**
   Bu katman verilen resimdeki kenar, köşe gibi resmin hakkında önemli özelliklerin çıkarımının yapıldığı katmandır. Bu katmanda belirli bir filtre seçilerek bütün görüntü üzerinde dolaştırılır yeni bir özellik matrisi elde edilmektedir. Bu katmanda kullanılan filtrenin türünün, boyutunun, kaydırılma miktarın gibi parametreleri katman çıkışında oluşacak özellik matrisinin yapısına ve boyutuna doğrudan etkisi vardır.

2. **Düzleştirme Doğrusal Birim Katmanı (ReLU layer)**
   Gerçekleştirilen konvolüsyon işleminden sonra elde edilen çıktı matris bu katmanda ReLU aktivasyon fonksiyonuna maruz kalır. ReLu aktivasyon fonksiyonun $f(x) = maks(0, x)$ formülüne sahiptir ve konvolüsyon işlemi sonucunda elde edilen negatif çıkıtları sıfır olarak değerlendirmektedir. ReLU burada ağımızın eğitimin

hızlandırmakta ve doğrusal olmayan yapısı sayesinde modelimizde doğrusal olmayan yapı kazandırır.

3. **Havuzlama Katmanı (Pooling Layer)**
   Konvolüsyon ve ReLU katmanından sonra elde edilen çıkış matrisine havuzlama uygulanır. Havuzlama uygulamadaki amaç alt örnekleme (down sampling) ve boyutta azalımı gerçekleştirebilmektir. Bu işlemde yine işlem yükümüzü azaltmakta ve özellik çıkarımıza katkı yapmaktadır. Birçok havuzlama yöntemi vardır bunlardan en yaygın olarak kullanılanı maksimum havuzlama ve ortalama havuzlamadır.

4. **Tam Bağlı Katman (Fully-Connected Layer)**
   Havuzlama katmanında sonra elde edilen çıktı bir vektör haline getirilerek 'Tam Bağlı Katmana' verilir. Bu katman birbirlerine tam bağlı yapay sinirlerden oluşmaktadır ve çalışması da yapay sinir ağlarından farklılık göstermemektedir. Tam bağlı katman genellikle evrşimsel ağların son katmanındır ve sınıf sonuçlar gibi hedefleri iyileştirmek için kullanılmaktadır.

5. **Seyreltme Katmanı (Dropout Layer)**
   Evrişimsel ağlarda eğitimin fazlalığından dolayı bazen ezberleme gerçekleşebilmektedir. Ağın verilen girdileri ezberlemesi demek daha önce verilmeyen bir girdi verildiğinde yanlış çıkış vereceği anlamına gelir. Ağın ezberleme kaynaklı yanlış tahminlerinin önen geçmek için ezberleme yapmasına engel olmak isteriz. Bu katmanda amaçlanan şeyde budur. Ağın belirli bir anında bazı düğümler ağdan iptal edilir ve ağ o düğümler olmadan çalışır. Başka bir anında bu düğümler aktif hale getirilir ve böylelikle ağdaki düğümlerin ezberlemesinin önen geçilmiş olur.

6. **Batch Normalization**
   Öğrenmeyi hızlandırmak için ön işleme sırasında veri setimiz genellikle normalizasyon uygulanır. Bu normlizasyondan sonra ağ içindeki işlemler soncunda bu dağılım değişebilir, normalizasyon kaybolabilir. Bunun öne geçmek için Batch Normalization katmanında yararlanırız.

Data Augmentation (Veri artırma)

Derin sinir ağlarında genellikle nispeten daha büyük ve çeşitli veri kümeleri üzerinde eğitilmiş sistemlerin diğerlerine nazaran daha sağlam (robust) olduğu bilinmektedir. Yeterli büyüklüğü ve veri çeşitliğin olmadığı sistemlerde sağlıklı çalışan tanıma sistemi oluşturmanın mümkün olmayacaktır. Buna çözüm olarak örnek sayısını artırma yoluna gideriz, mevcut veri kümemize döndürme(rotation), öteleme (translation) vb. işlemler uygulayarak sentetik veriler üretiriz.

ModelCheckpoint

Model 30 periyot (epoch) buyunca eğitilir fakat ModelCheckpoint kullanarak eğitimi sırasında en iyi başarımın elde edildiği (epoch) periyottaki modelin ağırlık değerleri saklanır. Bu sayede eğitim sırasında global ya da iyi lokale erişildikten sonra gerçekleşebilecek uzaklaşmaların sonucunda kaybolacak optimum ağırlıkların kaybolmaması sağlanmış olunur. Burada en iyi başarımı tanımlama için validation accuracy değerini kullandık. Bu anlamı eğitimin testi sırasında gerçekleşen en yüksek doğruluğun olduğu periyottaki ağırlıkları saklayacağız.

Early Stopping

Genellikle eğitim sırasında öğrenmenin gerçekleşmediği ya da overfitting oluştuğu durumlarda modelin eğitimin devam etmesini istemeyiz. Bu durumlarada erken durdurmadan yararlanabiliriz. Erkan durdurma için parametreleri her modele göre değiştirmiş olsam da bütün modellere uygulamış bulunmaktayım.

Gerçekleştirilen Çalışmalar

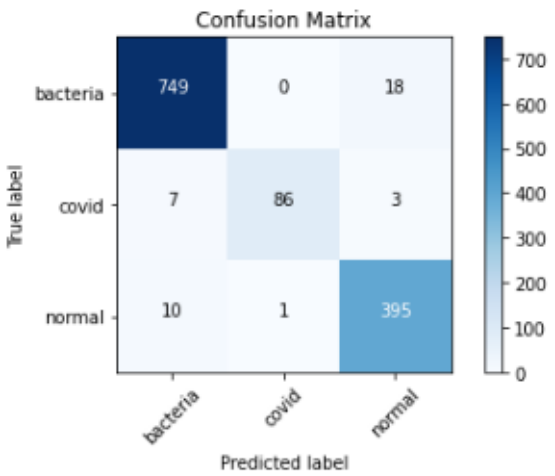
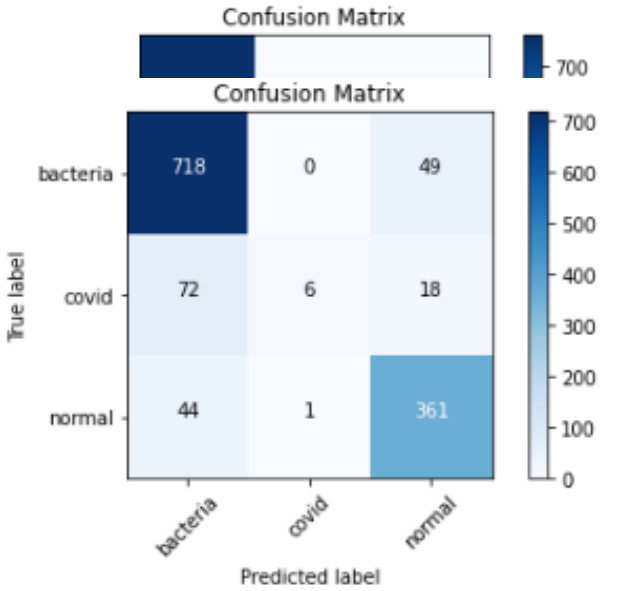
3 sınıf (Covid-Normal-Bakteri)

Öncelikle hem veri setini anlamak hem de en uygun optimizasyon algoritmalarını belirleyebilmek için 3 sınıf üzerinde Data Augmentation olmadan eğitim gerçekleştirmeye çalıştım. Elde ettiğim sonuçlar ve yaptırım çıkarımlara aşağıda verilmiştir.

Bundan sonra verilen ortalama değerleri için macro avg. Baz alınmıştır çünkü Bir macro avg. metriği her sınıf için bağımsız olarak hesaplar yapılır ve ardından ortalamayı alır (Her sınıfa eşit davranır).

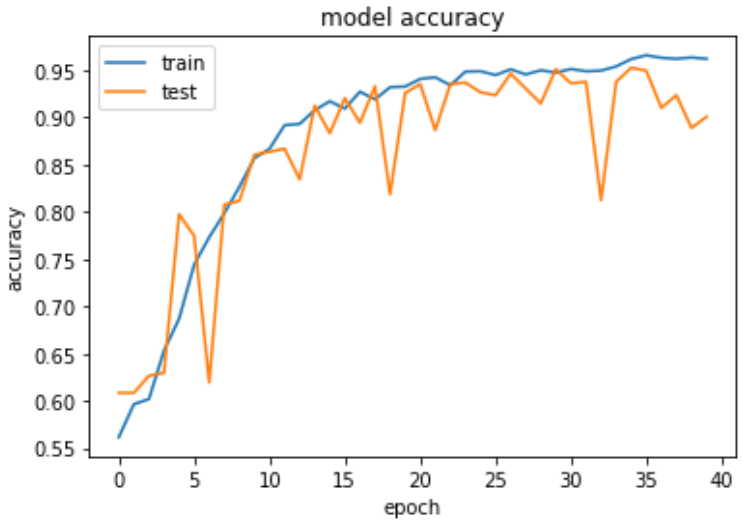
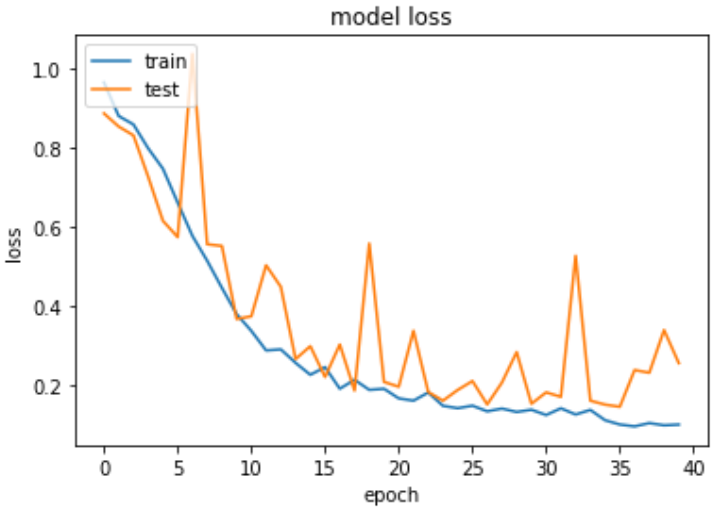
		Accracuy	Precision	Recall	F-Measure
ALEXNet	**SGD**	**0.97**	0.97	0.95	0.96
	Adadelta	**0.60**	0.20	0.33	0.25
	RMSProp	**0.60**	0.20	0.33	0.25
	Adam	**0.86**	0.85	0.63	0.63
	Adagrad	**0.60**	0.20	0.33	0.25
	Adamax	**0.60**	0.20	0.33	0.25

Yukarıda verilen optimizasyon algoritmalarında sadece SGD ve Adam başarılı olduğu için bunlara ilişkin Confusion Matrixler aşağıda verilmiştir.

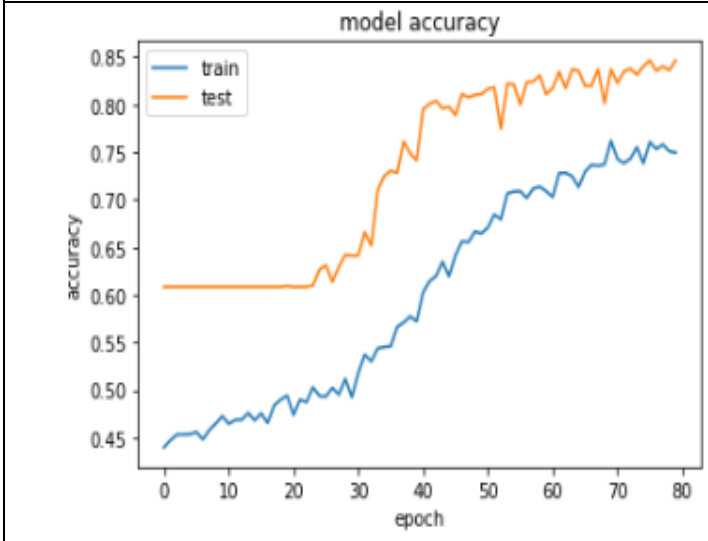
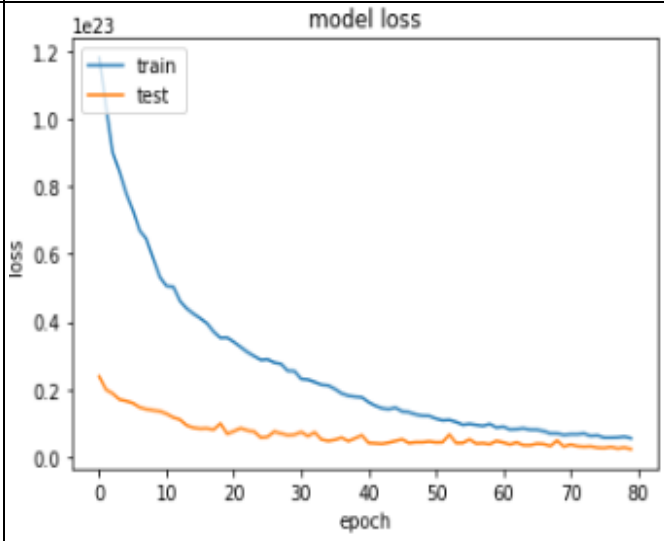
SGD	
	

Diğer yöntemlerde lokal minimum noktasına takınılmış ve belirli bir accuracy'den sonra ilerleme gerçekleşmemiştir. Yan tarafta RMSProp ilişkin Confusion matrix verilmiştir.

Aşağıda sırasıyla SGD ve Adam için öğrenme eğrisi ve kayıp değerinin değişimi grafiklerine yer verilmiştir.

Accuracy	Loss
	

* Grafikler x ekseni modelin eğitiminde geçen periyotlarını(epoch) ve y ekseni ise bu periyotta elde edilen doğruluk değerini ifade etmektedir. Dikkat edilecek olunursa her grafikte ikişer çizgi bulunmaktadır. Bu çizgilerden biri eğitim sırasında elde edilen doğruluk değerini diğeri ise her bir eğitim periyodunun sonunda gerçekleştirilen test sırasında elde edilen doğruluk değerini ifade etmek içindir. Bu iki yöntemin eğitimi için 40 periyot tahsis edilmiştir.

Accuracy	Loss
	

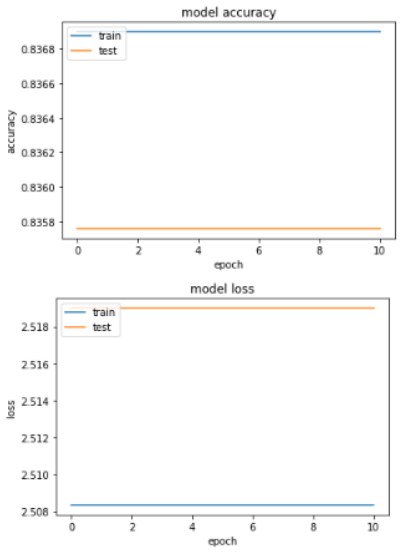
2 Sınıf (Covid-Normal)

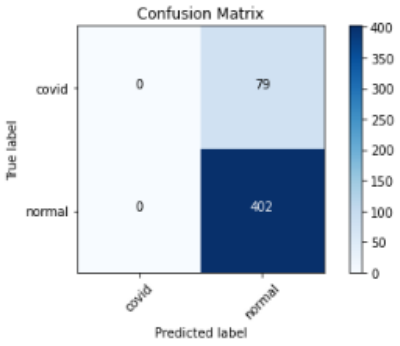
2 sınıf için çalıştığımız durumlarda hem resmin boyutlarını artırmamın hem de 3 kanal (RGB) ile çalışmanın modelin işlem maliyetini artırsa da veri azlığından dolay öğrenmeyi güç hale getirmeyeceğinizi düşünerek (500x500) boyutlarındaki renkli resimler ile çalışırım.

Aşağıda önce AlexNet modelin eğitilmesi sonucunda elde edilen sonuçlar daha sonrada gerçekleşen overfitting probleminde kurtulmak için gerçekleştirilen çabaya yer verilmiştir.

		Accracuy	Precision	Recall	F-Measure
AlexNet	**SGD**	0.99	0.98	0.98	0.98
	Adadelta	0.84	0.42	0.50	0.46
	RMSProp	0.84	0.42	0.50	0.46
	Adam	0.84	0.42	0.50	0.46
	Adagrad	0.84	0.42	0.50	0.46
	Adamax	0.99	0.99	0.99	0.99

SGD





Görüleceği üzere covid için öğrenme gerçekleşmemiştir.

Overfitting'de kurtulma çabalarım:

RMSPROP için öğrenme oranı 1 ve öğrenme hız düşü 1e-6 gibi düşük bir değer verilerek öğrenmeye teşvik edilmeye çalışmıştır fakat yine de başarılı olunmamıştır. Aşağıdaki grafiklerde bu durum çok açık bir şekilde belli olmaktadır.

#1e-6 =0.000.001 (micro)

```
opt=keras.optimizers.rmsprop(lr=1,decay=1e-6)
```

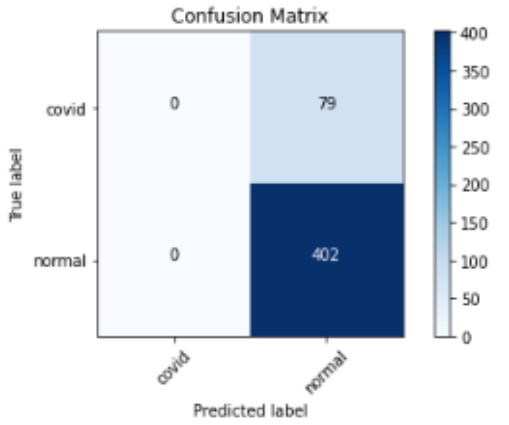
RMSProp(lr=1, decay=1e-6)

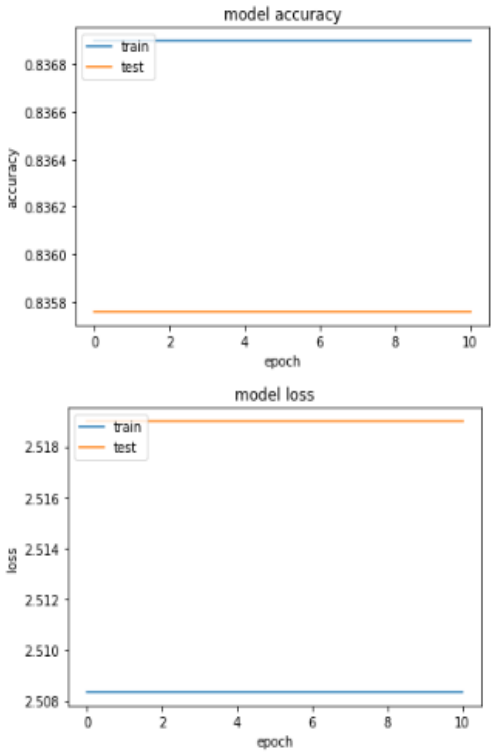
```
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        79
           1       0.84      1.00      0.91       402

    accuracy                           0.84       481
   macro avg       0.42      0.50      0.46       481
weighted avg       0.70      0.84      0.76       481

/usr/local/lib/python3.6/dist-packages/sklearn/metrics/
  _warn_prf(average, modifier, msg_start, len(result))
```





Buna ek olarak modeli eğitirken weightlere rastgele (random) değer atamanın modelin eğitimin katkı yapacağı düşünülere fully-connected layer'ımızda bulunan sinirlerimize rastgele değerler atanmış (#kernel_initializer='random_normal') ve aynı model bu sefer Adam için farklı öğrenme parametreleri (#Adam(lr= 0.001,decay=1e-6)) eğitime sunulmuştur. Fakat ne yazık ki üstedeki öğrenme grafiğinden çokta farklı bir grafik elde edilememiştir.

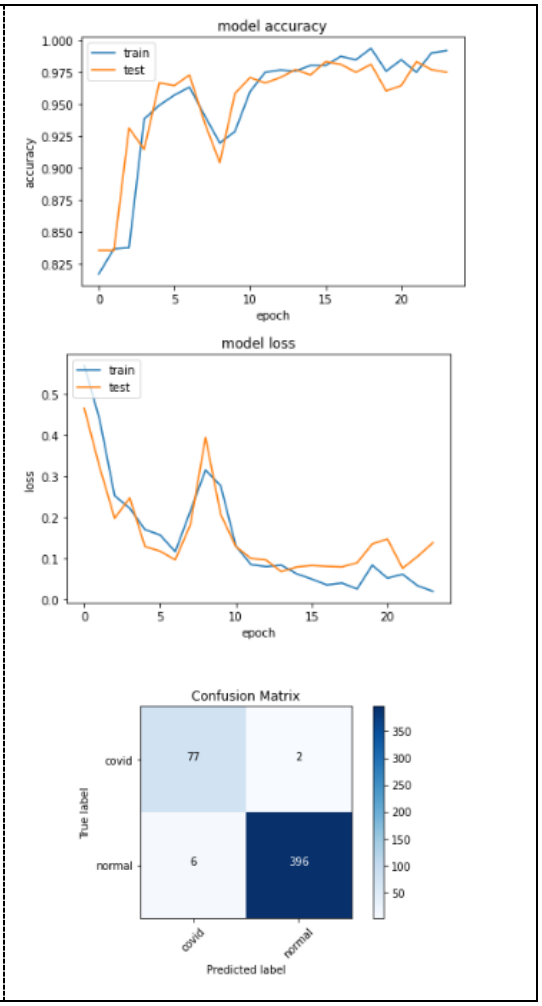
Yapılan araştırmalar doğrultusunda modelin karmaşıklığından (Model complexity) kaynaklı olarak overfittig ile karşılaşıldığı saptanmıştır. Alex-Net'in sahip olduğu konvolüsyon katmanlarından bazıları silinmiş konvolüsyonda kullanılan filtre boyutları küçültülmüş ve tam bağlı katmandaki nöron sayısı azaltılmıştır.
(https://datascience.stackexchange.com/questions/43191/validation-loss-is-not-decreasing)

Aşağıda bu zayıflatılmış Alex-net'in yapısı ve elde edile eğitime ilişkin görsel verilmiştir.

Light AlexNet

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 110, 110, 32)	6176
max_pooling2d_13 (MaxPooling	(None, 55, 55, 32)	0
conv2d_21 (Conv2D)	(None, 45, 45, 128)	495744
max_pooling2d_14 (MaxPooling	(None, 22, 22, 128)	0
conv2d_22 (Conv2D)	(None, 20, 20, 384)	442752
conv2d_23 (Conv2D)	(None, 18, 18, 256)	884992
max_pooling2d_15 (MaxPooling	(None, 9, 9, 256)	0
flatten_5 (Flatten)	(None, 20736)	0
dense_17 (Dense)	(None, 256)	5308672
activation_17 (Activation)	(None, 256)	0
dropout_13 (Dropout)	(None, 256)	0
dense_18 (Dense)	(None, 512)	131584
activation_18 (Activation)	(None, 512)	0
dropout_14 (Dropout)	(None, 512)	0
dense_19 (Dense)	(None, 1000)	513000
activation_19 (Activation)	(None, 1000)	0
dropout_15 (Dropout)	(None, 1000)	0
dense_20 (Dense)	(None, 2)	2002
activation_20 (Activation)	(None, 2)	0

Total params: 7,784,922
Trainable params: 7,784,922



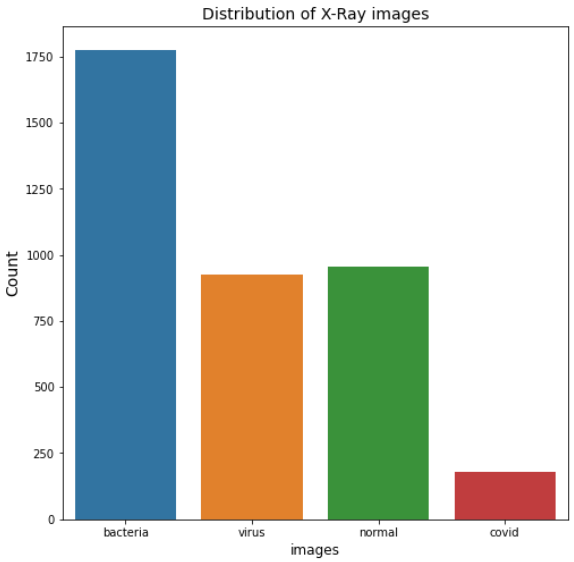
Parametre sayısında 28,039,482'den 7,784,922 gibi bir düşüş gerçekleştirilmiştir. Yaklaşık 3 katı daha hafif bir ağ tasarlanmıştır.

“If no mistake you have made,Yet losing you are. A different game you should play”

Yoda

Benim için gerçekleştirmiş olduğum çalışma burada yeninden başlıyor. Uzun uğraşlar sonucunda eğitime engel olan şeyi ön işleme sırasında gerçekleştirdiği normalizasyon olduğun farkına vardım. Ön işlemeden sonrada yine data augmentation sırasında normalizasyona maruz kalan verilerim eğitime elverişsiz hale gelmekte olduğunu fark ettim. Fakat buraya kadar olan kısmı benim için çok daha geliştirici olduğu için paylaşmaya karar verdim.

4 sınıf (Bakteri-Virüs-Covid-Normal)



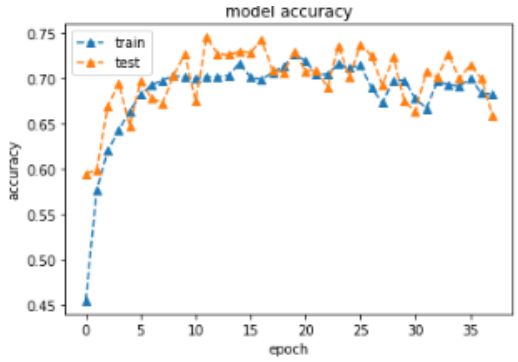
Yanda 4 sınıfa ilişkin veri setindeki örneklerimizin dağılımları verilmiştir. Ağırlıklı olarak veri setimizin bakteriden oluşmakta olduğu görülmektedir.

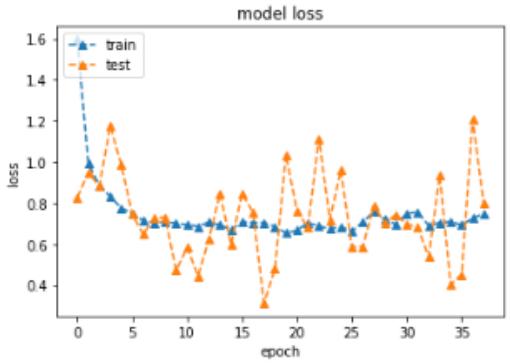
Veri setimiz toplamda 5478 görselden oluşmaktadır. Bu verinin 0.7 kısmı 3834 eğitim de kullanmak üzere ayrılmaktadır. Bu modelin eğitimi gerçekleştirilirken ağa bütün veri kümemiz 40 kere sunulmakta yani 40 epoch gerçekleştirilmektedir, bu periyotların her birinde 64 parçadan (batch) oluşan görüntüler ağa bir epoch için toplam 60 iterasyon gerçekleştirmektedir.($3834/64 \cong 60$)

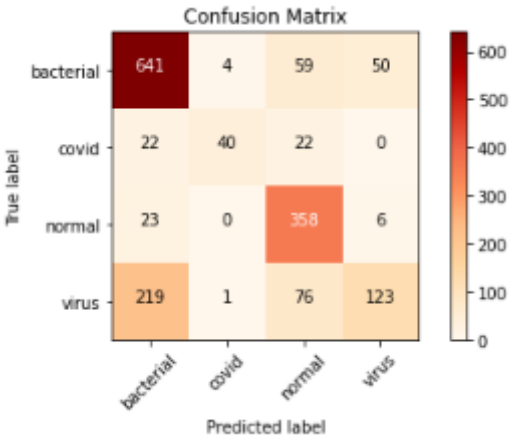
		Accracuy	Precision	Recall	F-Measure
LeNet-5	SGD	0.77	0.80	0.76	0.77
	Adadelta	0.75	0.79	0.72	0.72
	RMSProp	0.75	0.79	0.72	0.72
	Adam	0.71	0.74	0.64	0.65
	Adagrad	0.46	0.11	0.25	0.16
	Adamax	0.46	0.11	0.25	0.16

Yukardaki performans metrikleri incelediğinde Adagrad ve Adamax yönteminin başarısız olduğu açıkça görülmektedir. Aşağıda bu yöntemler içerisinde en başarılı sonuç veren SGD ve başarısız sonuç veren Adagrad için roc eğrileri ile birlikte karmaşıklık matrisleri verilmiştir.

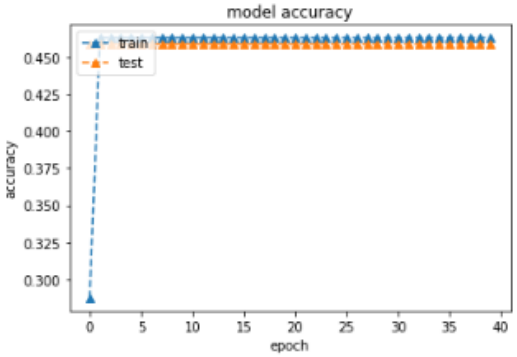
ADAM

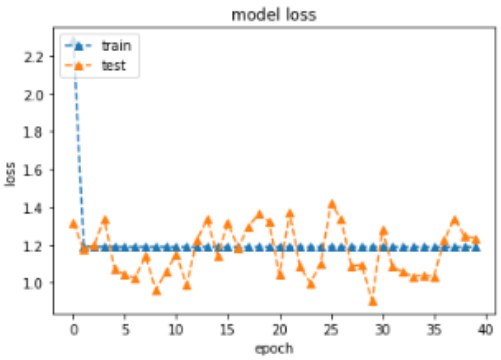


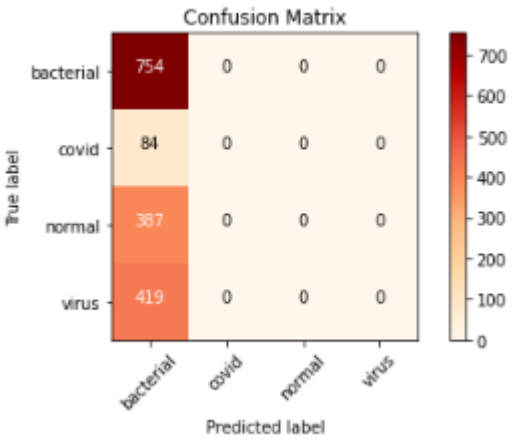




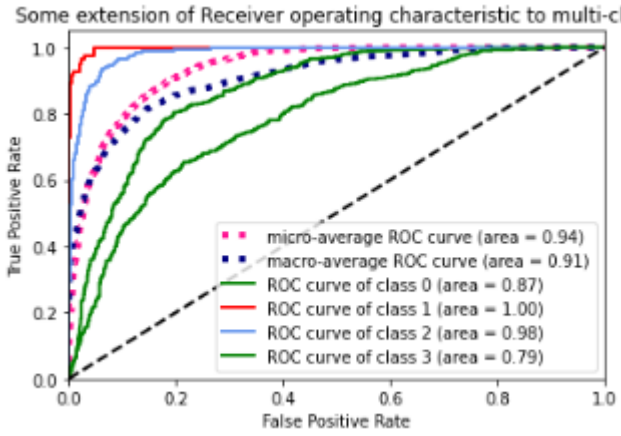
Adagrad



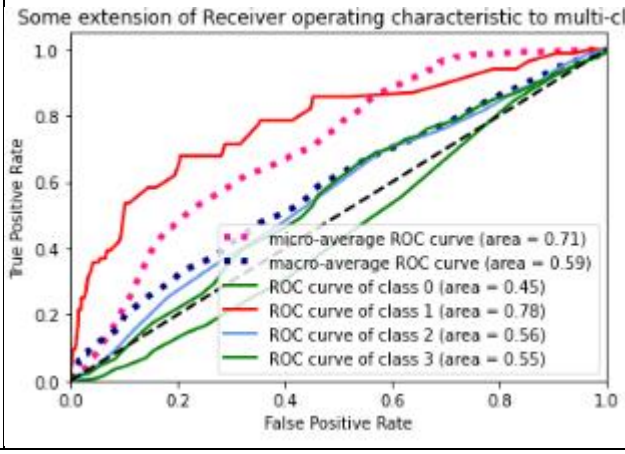




Adam'a ilişkin Roc eğrisi



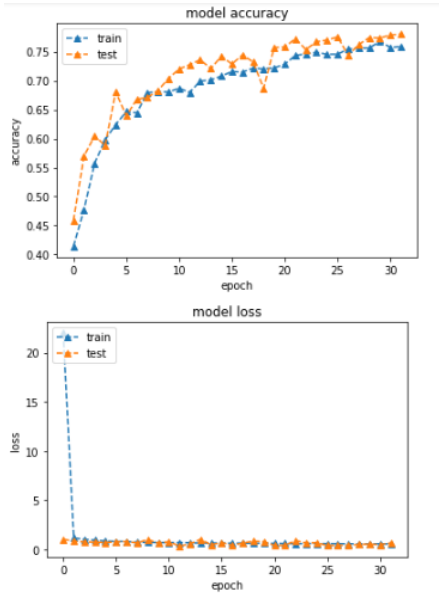
Adagrad'a ilişkin Roc eğrisi

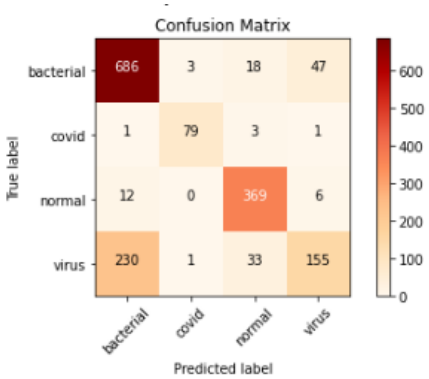


		Accracuy	Precision	Recall	F-Measure
AlexNet	SGD	0.46	0.11	0.25	0.16
	Adadelta	0.78	0.82	0.77	0.78
	RMSProp	0.62	0.50	0.44	0.43
	Adam	0.73	0.74	0.68	0.69
	Adagrad	0.62	0.50	0.44	0.43
	Adamax	0.78	0.83	0.79	0.79

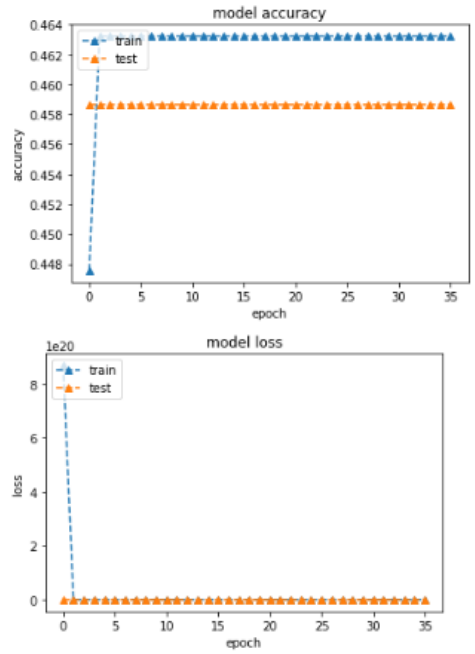
Aşağıda AlexNet'in eğitimi sırasında başarım gösteren Adamax ve başarısız olan SGD'ye ilişkin karmaşıklık matrisleri ve öğrenme eğrisi verilmiştir.

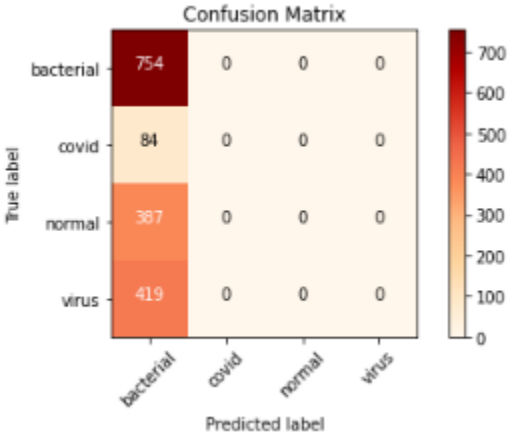
Adamax



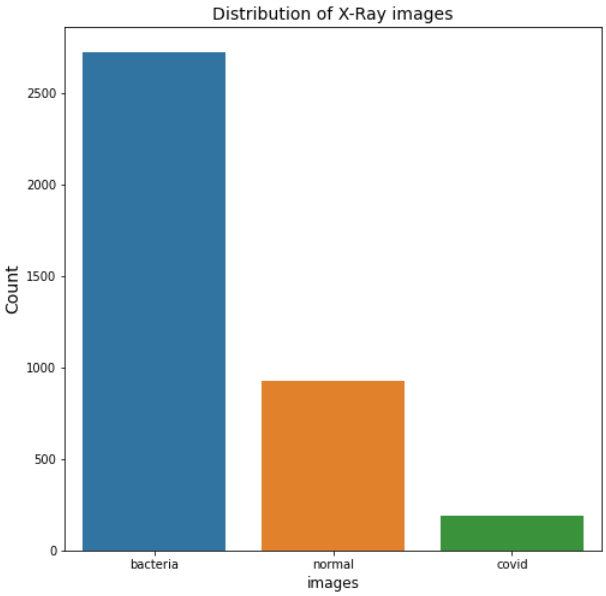


SGD





3 sınıf (Covid-Normal-Bakteri)



Yanda 3 sınıfa ilişkin veri setindeki örneklerimizin dağılımları verilmiştir. Ağırlıklı olarak veri setimizin virüs-bakteriden oluşmakta olduğu görülmektedir.

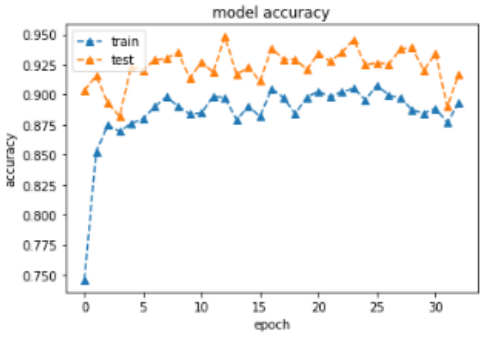
Veri setimiz toplamda 5478 görselden oluşmaktadır. Bu verinin 0.7 kısmı 3834 eğitim de kullanmak üzere ayrılmaktadır. Bu modelin eğitimi gerçekleştirilirken ağa bütün veri kümemiz 40 kere sunulmakta yani 40 epoch gerçekleştirilmektedir, bu periyotların her birinde 64 parçadan (batch) oluşan görüntüler ağa bir epoch için toplam 60 iterasyon gerçekleştirmektedir .

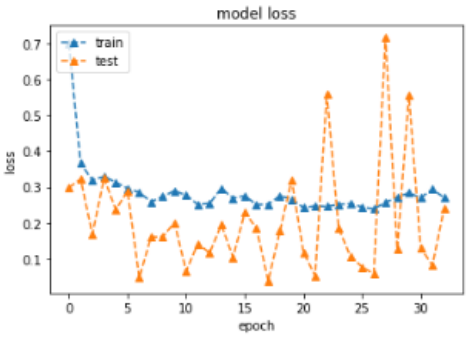
$(3834/64 \cong 60)$

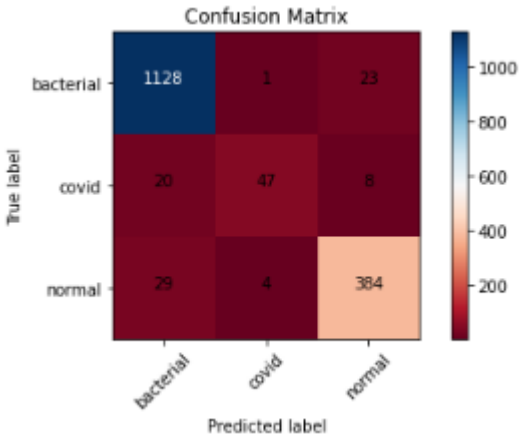
LeNet-5

	Accracuy	Precision	Recall	F-Measure
SGD	0.94	0.91	0.85	0.87
Adadelta	0.95	0.91	0.90	0.90
RMSProp	0.95	0.91	0.90	0.90
Adam	0.95	0.93	0.84	0.88
Adagrad	0.70	0.23	0.33	0.27
Adamax	0.95	0.93	0.92	0.92

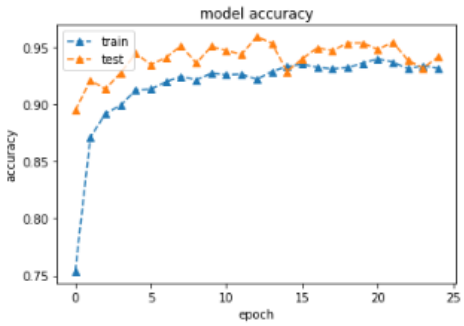
Adam

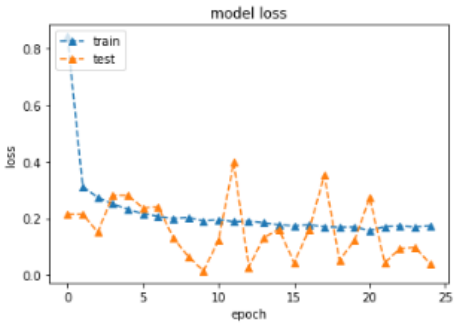


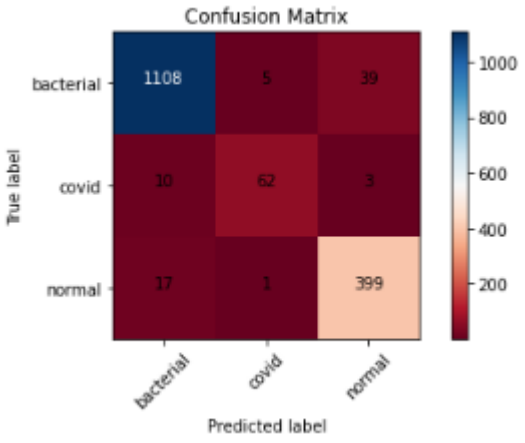


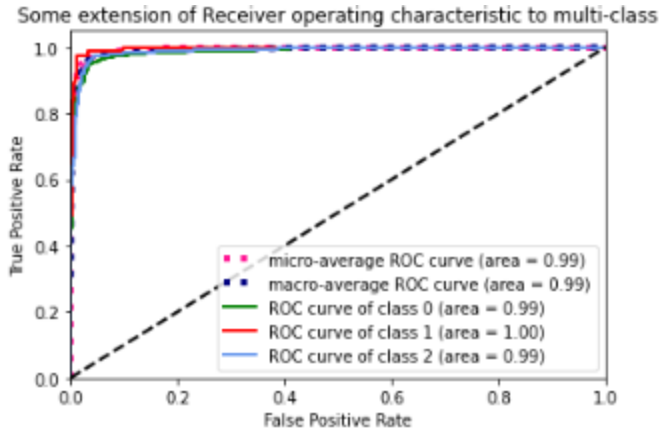


Adamax









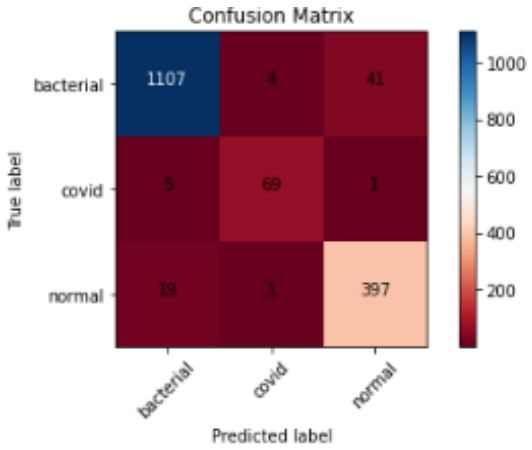
Yan tarafta Adamax'ın oluşturduğu Roc eğrisi görülmektedir.

		Accracuy	Precision	Recall	F-Measure
AlexNet	**SGD**	0.93	0.88	0.91	0.89
	Adadelta	0.96	0.94	0.94	0.94
	RMSProp	0.70	0.23	0.33	0.27
	Adam	0.86	0.82	0.67	0.71
	Adagrad	0.70	0.23	0.33	0.27
	Adamax	0.96	0.95	0.95	0.95

Sınıflandırma Raporları Üzerinden inceleme yaptığımızda Adamax ve Adadelta'nın covid-19 verisi daha doğru sınıflandırdığı görülmektedir. Bu yüzden aşağıda bu ikisine ilişkin karmaşıklık matrisleri yer verilmiştir.

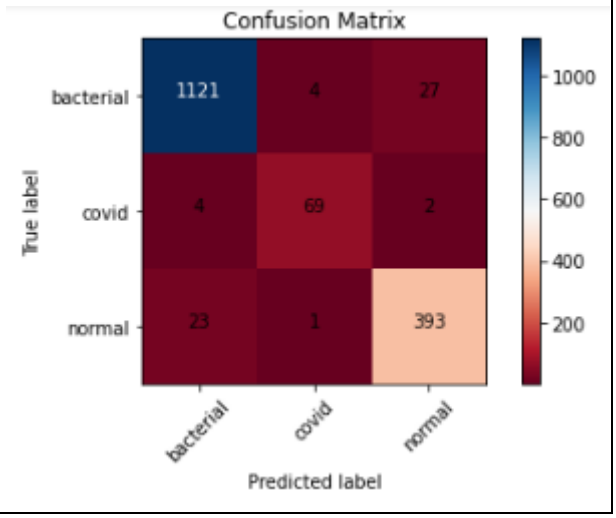
Adadelta'nın Sınıflandırma raporu ve karmaşıklık matrisi

	precision	recall	f1-score	support
bacterial	0.98	0.96	0.97	1152
covid	0.93	0.92	0.93	75
normal	0.90	0.95	0.93	417
accuracy			0.96	1644
macro avg	0.94	0.94	0.94	1644
weighted avg	0.96	0.96	0.96	1644

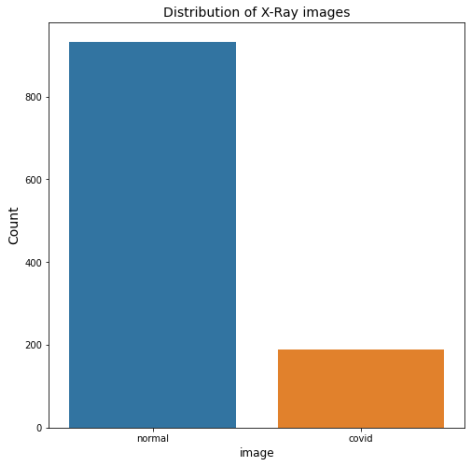


Adamax'ın Sınıflandırma raporu ve karmaşıklık matrisi

	precision	recall	f1-score	support
bacterial	0.98	0.96	0.97	1152
covid	0.93	0.92	0.93	75
normal	0.90	0.95	0.93	417
accuracy			0.96	1644
macro avg	0.94	0.94	0.94	1644
weighted avg	0.96	0.96	0.96	1644



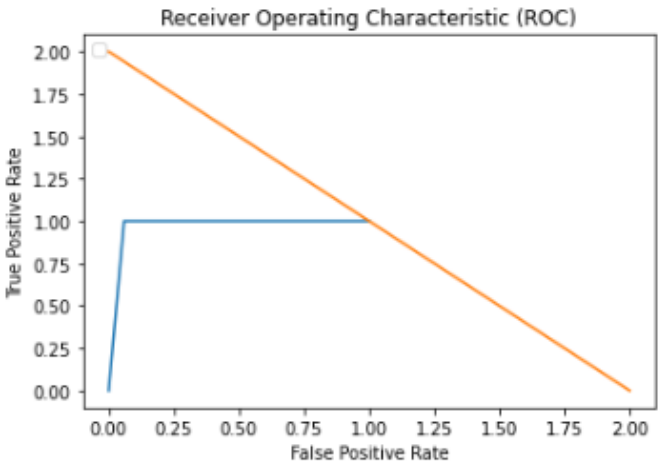
2 Sınıf (Covid-Normal)

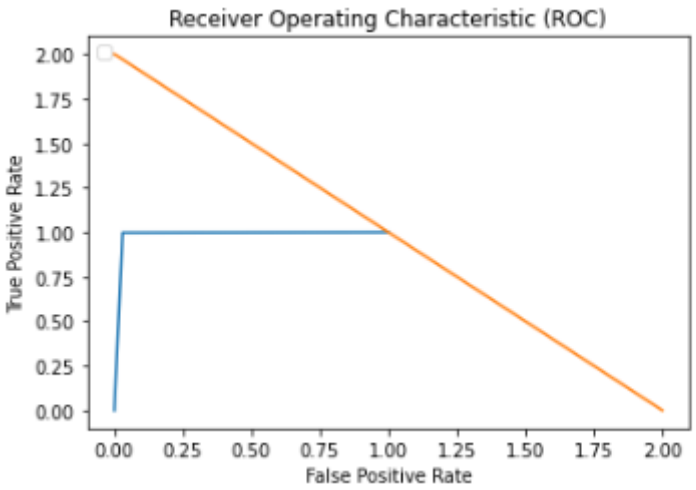


İki sınıf birine olan dağılımı yanda verildiği gibidir. AlexNet için eğitim ilk kısımda gerçekleştirildiği için burada bir daha AlexNet için eğitim gerçekleştirilmemiştir.

		Accracuy	Precision	Recall	F-Measure
LeNet-5	SGD	0.98	0.95	0.98	0.96
	Adadelta	0.99	0.99	0.98	0.99
	RMSProp	0.99	0.98	0.96	0.97
	Adam	0.99	0.99	1.00	0.99
	Adagrad	0.99	1.00	0.97	0.99
	Adamax	0.99	0.99	0.99	0.99

Amacımın Covid-19 belirleyebilmekti bu bağlamda en başarılı gördüğüm Adagrad (Hassasiyeti değerinden dolayı) ve Adamax için aşağıda sırasıyla ROC eğrileri verilmiştir.





CanNET

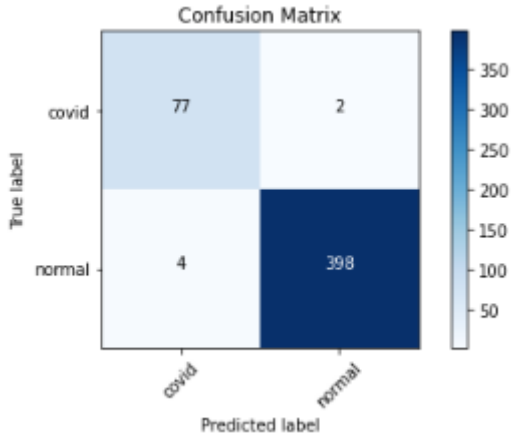
Tasarlana ağ ile ne AlexNet kadar karmaşık düzeyi yüksek ne de LeNet kadar basit bir ağ tasarlamak istenmiştir. AlexNet karmaşıklığına sahip olmayarak ağda başarılı öğrenme gerçekleştirilecek ve LeNet daha başarılı sınıflandırma elde edilecektir. Bu iki önemli mimarinin arasında yer alacak ve daha hızlı tepki veren, daha iyi öğrenen bir ağ tasarlanmak istenmiştir. Bu bağlamda ağ tasarlanırken AlexNet'den feyz alınarak 3lü Conv katmanına (Triple Conv) yer verilmiş ve kullanılan filtre boyutlarında küçülmeye gidilmiştir. Ağın parametreleri ve mimarisi aşağıdaki tabloda verilmiştir.

Katman	Boyut	Parametre
Konvolüsyon Katmanı 1	32x4x4	Stride=2
Havuzlama Katmanı 1	32x2x2	Stride=0 Padding='valid'

Konvolüsyon Katmanı 2	16x4x4	Stride=2
Havuzlama Katmanı 2	16x2x2	Stride=2 Padding=’valid’
Konvlüsyon Katmanı 3_a	32x1x1	Stride=1
Konvlüsyon Katmanı 3_b	32x1x1	Stride=1
Konvlüsyon Katmanı 3_c	32x1x1	Stride=1
Havuzlama Katmanı 3	32x2x2	Stride=2 Padding=’valid’
Tam Bağlı Katman 1 +Dropout 1	1024	Rate=0.4
Tam Bağlı Katman 2+Dropout 2	1024	Rate=0.4

2 Sınıf (Covid-Normal)

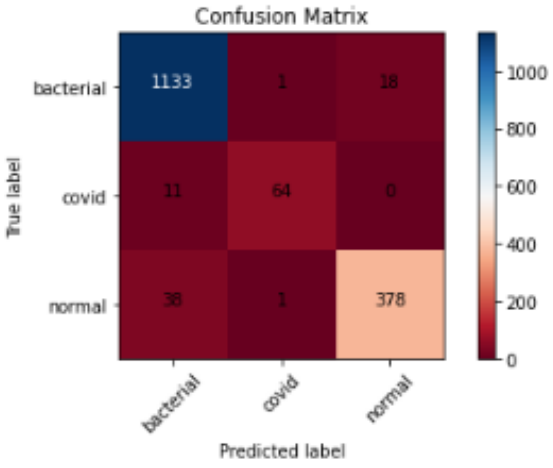
		Accracuy	Precision	Recall	F-Measure
CanNet	SGD	0.93	0.88	0.91	0.89
	Adadelta	0.96	0.94	0.94	0.94
	RMSProp	0.70	0.23	0.33	0.27
	Adam	0.98	0.99	0.95	0.97
	Adagrad	0.70	0.23	0.33	0.27
	Adamax	0.96	0.95	0.95	0.95



Görüldüğü üzere model covid verisini düşük bir kayıpla başarı bir şekilde öğrenmiştir.

3 sınıf (Covid-Normal-Bakteri)

		Accracuy	Precision	Recall	F-Measure
CanNet	**SGD**	**0.96**	**0.96**	**0.91**	**0.94**
	Adadelta	**0.87**	**0.86**	**0.61**	**0.63**
	RMSProp	**0.71**	**0.50**	**0.40**	**0.39**
	Adam	**0.76**	**0.54**	**0.42**	**0.42**
	Adagrad	**0.70**	**0.23**	**0.33**	**0.27**
	Adamax	**0.96**	**0.95**	**0.95**	**0.95**

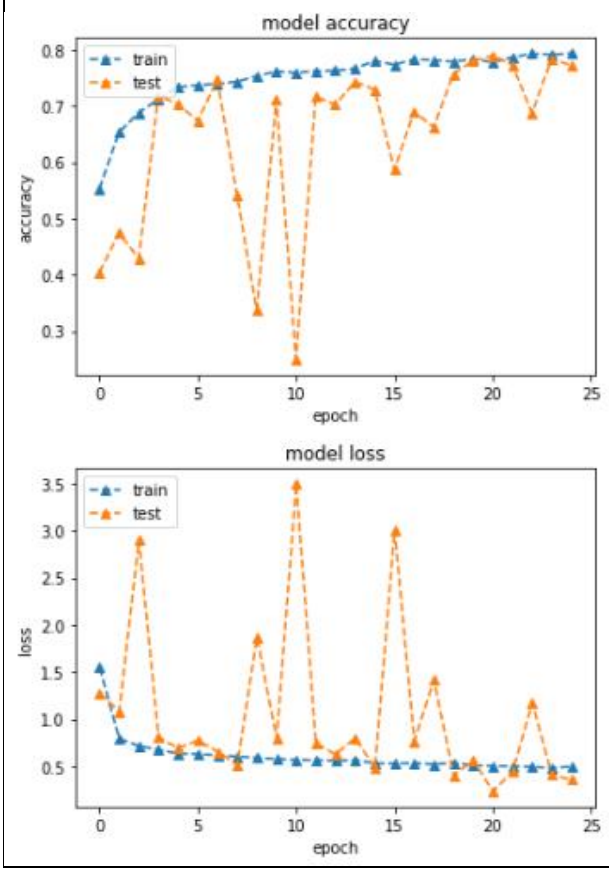


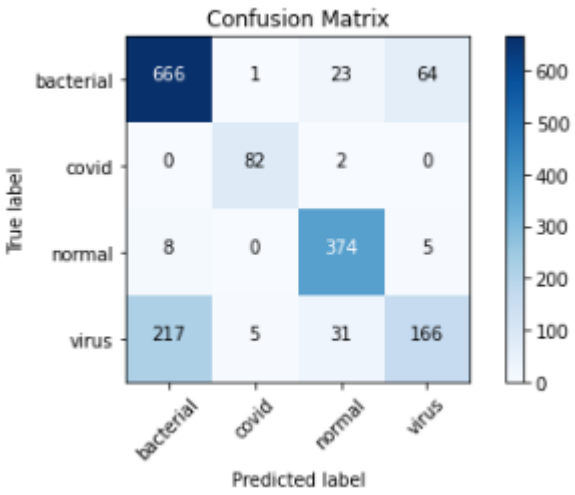
Yanda SGD optimizasyon algoritması kullanıldığında ne kadar başarılı bir sonuç alındığı görülmektedir.

4 sınıf (Bakteri-Virüs-Covid-Normal)

		Accracuy	Precision	Recall	F-Measure
CanNet	SGD	0.80	0.83	0.82	0.82
	Adadelta	0.78	0.81	0.81	0.80
	RMSProp	0.71	0.50	0.40	0.39
	Adam	0.81	0.84	0.82	0.82
	Adagrad	0.70	0.23	0.33	0.27
	Adamax	0.81	0.84	0.82	0.82

Adadelta





Yanda adadelta'nın her epoch göstermiş olduğu eğilim görülmektedir. Yukarda eğitim tamamlanması sonucunda elde edilen karmaşıklık matrisi yer almaktadır.

Sonuç

Yukarda da belirtiğim gibi ağ karmaşıklığı olmayan öğrenmenin optimzasyon algoritmaları ile gerçekleşebildiği hafif bir model tasarlanmak istenmiştir. Modelin eğitimi AlexNete göre kısa sürmekte fakat LeNete göre daha uzun zaman almaktadır. Diğerleri gibi sık olmasa da global optimuma ulaşamadığı durumlar olmaktadır. Modelimiz lightweight bir model olup 2.672.850 parametreden oluşmaktadır. Bunu parametrelerin 1049600 tanesi tam bağlı katmanımızda ,2050 tanesi çıkış katmanımızda bulunmaktadır.
İlk konvolüsyon katmanımızda 1568, ardından gelen batch normalizasyon katmanımızda 128 parametre bulunmaktadır.
İkinci konvolüsyon katmanımızda batch normalizasyon katmanımız ile birlikte toplamda 8272 parametre bulunmaktadır.
Arkasından gelen üçlü konvolüsyon katmanımızda ise toplamda 4576 parametreden oluşmaktadır.
Nöron sayılarına ilişkili olarak ilk konvolüsyon katmanda toplamda 32 nöron, ikinci konvolüsyon 16 nöron üçlü konvolüsyon katmanımız totalde 96 nörondan oluşmaktadır. Tam bağlı katmanızın 1024 yine aynı şekilde ikinci tam bağlı katmanımız 1024 ve çıkış katmanımız sınıf sayısı kadar nörondan oluşmaktadır.
Batch Normalizasyon uygulamam şüphesiz ki ağımın eğitim performansını artırmış durumda ve daha basit bir ağ oluşturmamı sağlamıştır.

Kaynakça

Amidi , A., & Amidi, S. (2018, Kasım 26). *Convolutional Neural Networks cheatsheet*. https://stanford.edu/: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks adresinden alındı

Bui, H. M., Lech, M., Cheng, E., Neville, K., & Burnett, I. S. (2016). Using grayscale images for object recognition with convolutional-recursive neural network. *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE).* Ha Long, Vietnam: IEEE.

Çalik, N., Kurban, O. C., Yilmaz, A. R., Ata, L. D., & Yildirim, T. (29 June 2017). Signature recognition application based on deep learning. *2017 25th Signal Processing and Communications Applications Conference (SIU).* Antalya, Turkey: IEEE.

Daniş, Z. (tarih yok). *Veri Artırımı / Data Augmentation*. Medium: https://medium.com/novaresearchlab/veri-art%C4%B1r%C4%B1m%C4%B1-data-augmentation-fcc839da556b adresinden alındı

Dey, S., Dutta, A., Toledo, J. I., Ghosh, S. K., Llados, J., & Pal, U. (2017). SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification. *Computer Vision and Pattern Recognition*.

KURT, F. (2018). *EVRİŞİMLİ SİNİR AĞLARINDA HİPER PARAMETRELERİN ETKİSİNİN İNCELENMESİ (Yüksek Lisans Tezi).* Ankara, Türkiye.: Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü.

Li, F.-F., Krishna, R., & Xu, D. (2020). *CS231n: Convolutional Neural Networks for Visual Recognition.* http://cs231n.stanford.edu/: https://cs231n.github.io/convolutional-networks/ adresinden alındı

N. Otsu. (1975). A threshold selection method from gray-level histograms,. *Automatica, vol. 11*, 285-296.

Ng, A., Katanforoosh, K., & Mourri, Y. B. (tarih yok). *Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization.* Coursera. adresinden alındı

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 1930-1958.

ŞEKER, A., DİRİ, B., & BALIK, H. H. (2017). Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme. *Gazi Mühendislik Bilimleri Dergisi* , 47-64.

(Yeni covid-19 görsellerine ilişilin link: https://github.com/gabrielpierobon/cnnshapes/blob/master/README.md)