****

**EGE ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**2020 – 2021**

**GÜZ DÖNEMİ**

**GÖRÜNTÜ İŞLEME DERSİ**

**PROJE 2**

**DERİN ÖĞRENME İLE GÖRÜNTÜ TANIMA / NESNE TESPİTİ**

Ege Doğan Dursun 05170000006

**1. Image Classification, Object Detection ve Image Segmentation kavramları arasındaki farklar nelerdir? Belirtiniz. (5)**

Image Classification, görselleri çeşitli paternlere ve bağlantılara dayalı olarak sınıflandırmaya verilen isimdir. Makine öğrenmesi söz konusu olduğunda, bu tür sınıflandırmalarda görsellerin arasındaki farkları ve özgünlüklerini belirten çeşitli paternler ve öznitelikler tanımlanarak bu sayede çeşitli görsellerin sınıflandırılması ve etiketlenmesi gerçekleştirilir. Object Detection, çeşitli tanımlara ve sınıflara bağlı olan objelerin görsel bir uzay içerisinde tespit edilmesini ifade eden bir kavramdır. Yine bu kavramın içerisinde de çeşitli objeler öznitelikleri ile ayrışır ve tespit edilirler. Image Segmentation ise görüntünün bölütlenmesini ifade etmek için kullanılmaktadır. Bu fonksiyon, kimi zaman görüntüdeki belli detayların ön plana çıkarılması için, kimi zaman ise görüntüdeki ana temanın daha da güçlendirilmesi için kullanılabilmektedir ve özellikle tıpta çeşitli dokuların tespiti ve ayrıştırılması gibi konularda yoğun olarak kullanılmaktadır. [1][2][3]

Image Classification, Object Detection fonksiyonu için gerekli olan bir kavramdır. Çünkü bir objeyi tespit etmek istiyorsak onu öncelikle tanımlayabilmeliyiz. Buna ek olarak bir görüntüdeki belirli öğelerin diğerlerinden öne çıkması da onun ayrıştırılmasını kolaylaştırmaktadır. Bu durumda Image Segmentation aslında Image Classification ve dolayısıyla Object Detection konusunda da anahtar bir kavram olabilmektedir.

**2. Ön Çalışma: Ders notlarından derin öğrenme (deep learning) ile ilgilli sunum ve ders videosunu inceledikten sonra internetten derin öğrenme ile ilgili görüntü tanıma örnekleri araştırarak yazılanları okuyunuz. İlgili konularda anlatım yapılan videolar izleyiniz. İncelediğiniz örneklerin, izlediğiniz video ve diğer kaynakların linkeri ile, araştırmanızdan kısaca neler öğrendiğinizi rapora ekleyiniz. (5)**

Proje dahilinde öncelikli olarak keras kütüphanesine ait modüllerin dökümentasyonunu inceledim ve kullanılan metotların hangi amaçlar ile entegre edilebildiğini anlamaya çalıştım. [4]

Ardından, keras kullanılarak oluşturulan örnek Evrişimsel Sinir Ağı modellerine yönelik bir araştırma yaptım ve bu örnek modelleri kendim oluşturmak üzerine uğraştım. [5][6] Geliştirdiğim modellerin yine keras, sklearn, matplotlib gibi kütüphaneler kullanılarak nasıl performans ölçümlemesine tabi tutulabileceği ve elde edilen verilerin nasıl görselleştirilebileceği konusunda araştırmalar gerçekleştirdim. Daha etkin görselleştirmeler yapabilmek için Matplotlib kütüphanesine ait dökümentasyon üzerine çalıştım. [7]

Evrişimsel sinir ağlarındaki Evrişim ve Havuz katmanlarının nasıl çalıştığını ve arkalarındaki mantığı öğrenmek için çeşitli sitelerden araştırmalar yaptım. Düzleştirme, Dropout, Adam optimizasyonu, aktivasyon fonksiyonları gibi çeşitli konularda bilgi almak için çeşitli kaynakları araştırdım. [8][9][10]

Öğrendiklerimi devam eden cümlelerle özetleyebilirim. Öncelikli olarak Evrişimsel Sinir Ağları aslında beyinimizin görsel korteksindeki yorumlayıcı hücrelerin ve retinamızdaki ışık algılayıcı hücrelerin birlikte gerçekleştirdiği fonksiyona çok yakın bir işlev sağlıyor. Konvolüsyon ve Havuz katmanları sayesinde görseller tek bir büyük parça yerine küçük parçalara ayrıştırılarak ve çeşitli filtreler uygulanarak farklı açılardan inceleniyor ve bu tıpkı beynimizde olduğu gibi işlem tasarrufu için veri sıkıştırma yöntemlerine tabi tutuluyor. Aktivasyon fonksiyonlarını da beynimizdeki nöronların eylem potansiyeline benzetebiliriz. Elbette beyindekinden farklı olarak yazılımsal ortamda ifade edilen nöronlar kimyasal değil, matematiksel olarak çalışmaktadır. Bu noktada çeşitli aktivasyon fonksiyonları mevcut. Bunlardan günümüzde en çok tercih edileni ve en performanslı olduğu iddia edileni ReLU ismindeki aktivasyon fonksiyonu. Elbette, çeşitli veri bilimi problemleri için belirli başka aktivasyon fonksiyonları özel olarak daha verimli olabiliyor.

Son olarak karşılaştığım ilginç detaylardan biri, örneğin tıbbi görüntüler gibi çok büyük boyutlu görsellerde, kimi zaman yapay sinir ağının çok küçük ayrıntıları tespit etmede başarısız olmasıydı. Bunu engellemek için görüntüyü tek bir büyük görsel yerine küçük patch’lere ayırarak beyindeki piramidal nöronlara benzetilen Residual Sinir Ağları kullanarak işlemek çok daha verimli sonuç veriyormuş. Aynı şekilde, kullanılan görsellerde hangi noktanın problematik olduğunun maskelenme yöntemi ile işaretlenmesi de verimin arttırılmasını kolaylaştıran etkenlerden olarak belirtiliyor.

**3. Derin Öğrenme ile Görüntü Tanıma / Nesne Tespit Yazılımı Geliştirme**

**3.1 Konu Belirleme ve Problemin Tanımı (5)**

Projemde keras kütüphanesinin içerdiği veri setleri arasında yer alan MNIST veri setini kullanarak bir Evrişimsel Sinir Ağı (CNN) tabanlı görüntü tanıma yazılımı geliştirmeye çalıştım. Geliştirme sürecinin ardından da, hiper parametreleri optimize etmek üzerine uğraşarak geliştirdiğim modelin doğruluk oranlarını olabildiğince yükseltmek için çaba sarfettim.

MNIST veri seti el yazısı ile çizilmiş rakamlardan meydana gelmektedir ve dahilinde 60.000 adet eğitim görseli ve 10.000 adet test görseli bulundurmaktadır. Görsellerin boyutu 28x28 ebatlarındadır ve siyah beyazdır. MNIST veri seti dahilindeki görsellerin birçoğu NIST verisetine ait olmakla birlikte, rakamlara ek olarak karakterleri de içeren ve 240.000 eğitim, 40.000 test görselinden oluşan EMNIST adında bir veri seti de bulunmaktadır.

**3.2 Veriseti Hakkında Bilgi (Kullanılan / Oluşturulan) (5)**

Projede kullanılan veri seti keras kütüphanesi dahilinde de hazır olarak kullanılabilen MNIST veri setidir. MNIST veri seti 60.000 adet eğitim, 10.000 adet test görselinden oluşan ve el yazısıyla yazılmış rakam görselleri içeren bir veri setidir. Bu veri setindeki görsellerin ebatları 28x28’dir ve siyah beyaz görsellerdir.

10’luk sayı sisteminde toplam 10 adet rakam bulunduğundan, MNIST veri setindeki toplam sınıf sayısı da 10 adettir. Bu rakamlar 0, 1, 2, 3, 4, 5, 6, 7, 8 ve 9’dur.

MNIST veri setine ait görseller Amerikalı Census Bureau çalışanları ve Amerikalı lise öğrencilerinin el yazılarından meydana gelmektedir. 1999 yılında NYU’dan Yann LeCun tarafından oluşturulmuştur.

Veri seti 1999 yılından beri çeşitli makine öğrenmesi ve derin öğrenme modellerinin test edilmesinde kullanılmaktadır ve neredeyse standart denilebilecek bir ölçek haline gelmiştir. Kimi zaman derin öğrenme üzerine özelleştirilmiş belirli cihazların performansları kıyaslanırken MNIST veri seti kullanılarak örneğin ResNet50 gibi hazır sinir ağları üzerinde ölçümlenme yapılır ve sonuçlar değerlendirilir.

**3.3.1 Kullandığınız ortam, dil, kütüphane adı ve sürümlerini yazınız. (5)**

Projemde kullandığım ortam Python 3 uyumlu bir IDE olan Spyder’dı. Spyder’ın 3.6.6 sürümü üzerinde çalışmalarımı gerçekleştirdim. Kullandığım Python sürümü ise Python 3.7.0’dı. Projem dahilinde derin öğrenme konusunda fayda sağlayan çeşitli kütüphanelere ait fonksiyonları kullandım. Bunlara örnek olarak keras, numpy, matplotlib, seaborn ve sklearn gösterilebilir. Kullandığım bu kütüphanelerin versiyonları aşağıdaki gibidir.

* **Keras** : 2.4.3
* **Numpy** : 1.19.4
* **Matplotlib** : 3.1.2
* **Seaborn** : 0.11.0
* **Sklearn** : 0.24.0

**3.3.2 Derin Öğrenme ile bir tanıma / tespit uygulaması geliştiriniz, modeli oluşturunuz, model adı ve katman sayısı bilgilerini, kodu ve modelin şemasını rapora ekleyiniz. (10)**

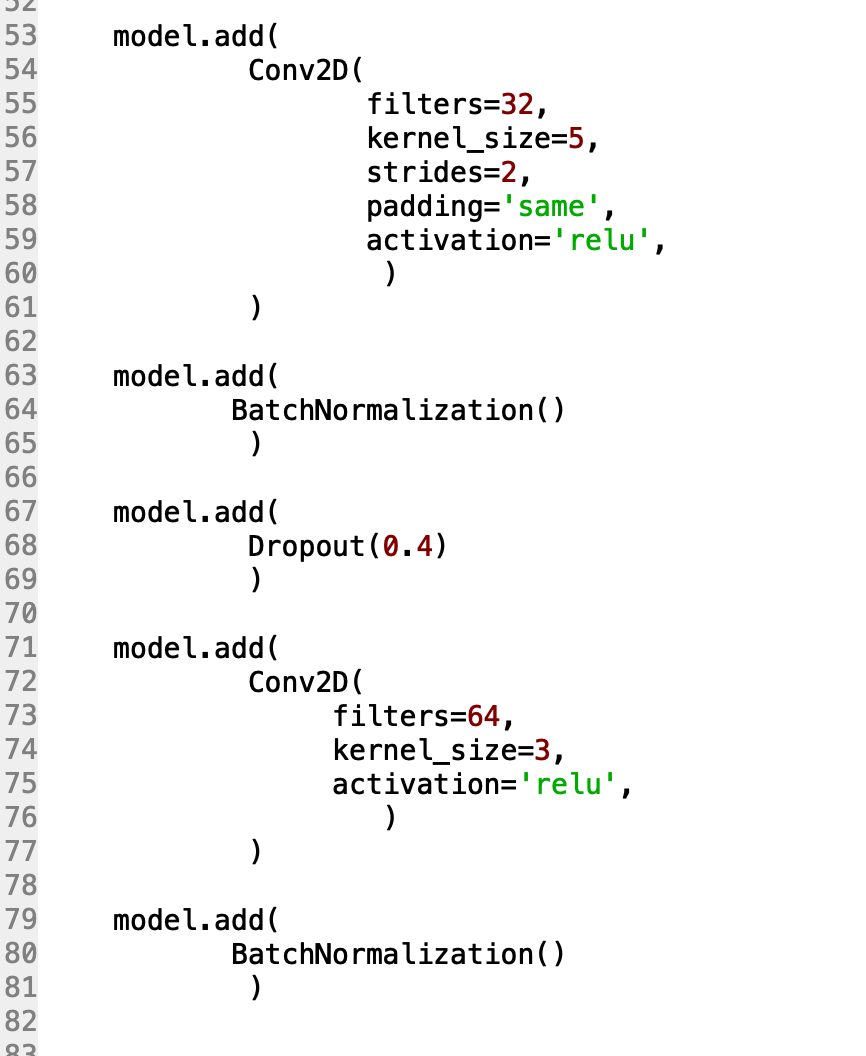
Projemde kullandığım derin öğrenme modeli bir Evrişimsel sinir ağı mimarisidir. Modelin tanımı proje dahilinde “model.py” isminde ayrı bir dökümanda gerçekleştirilmiştir.

Modelde Sequential mimari tercih edilmiştir ve ilgili mimariyi oluşturmak için keras.models modülünden faydalanılmıştır. Modelde Evrişim işlemlerini gerçekleştirmek için keras.layers modülüne ait Conv2D ve MaxPooling2D sınıflarından yararlanılmıştır. BatchNormalization ve Dropout katmanları ile performans optimizasyonu sağlanması hedeflenmiştir. Buna ek olarak evrişim çıktılarını düzleştirmek için Flatten sınıfı ve yoğun katmanlar için de Dense sınıfı kullanılmıştır.

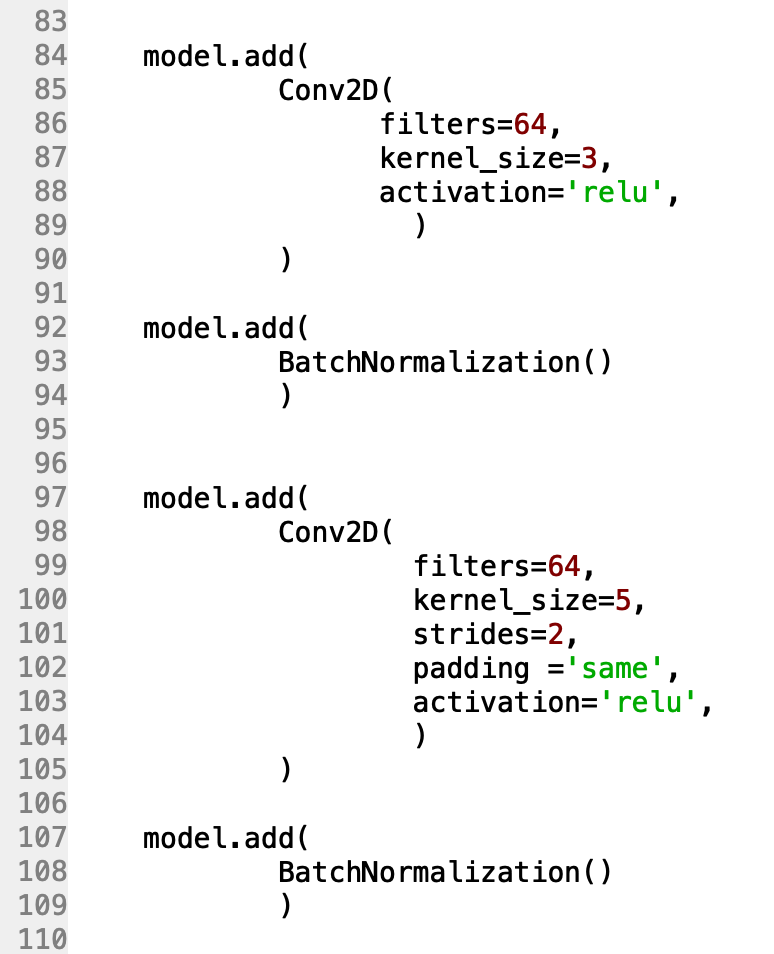
Aşağıdaki görsellerde modele yönelik mimarinin Python kodu şeklindeki ifadesini inceleyebilirsiniz.



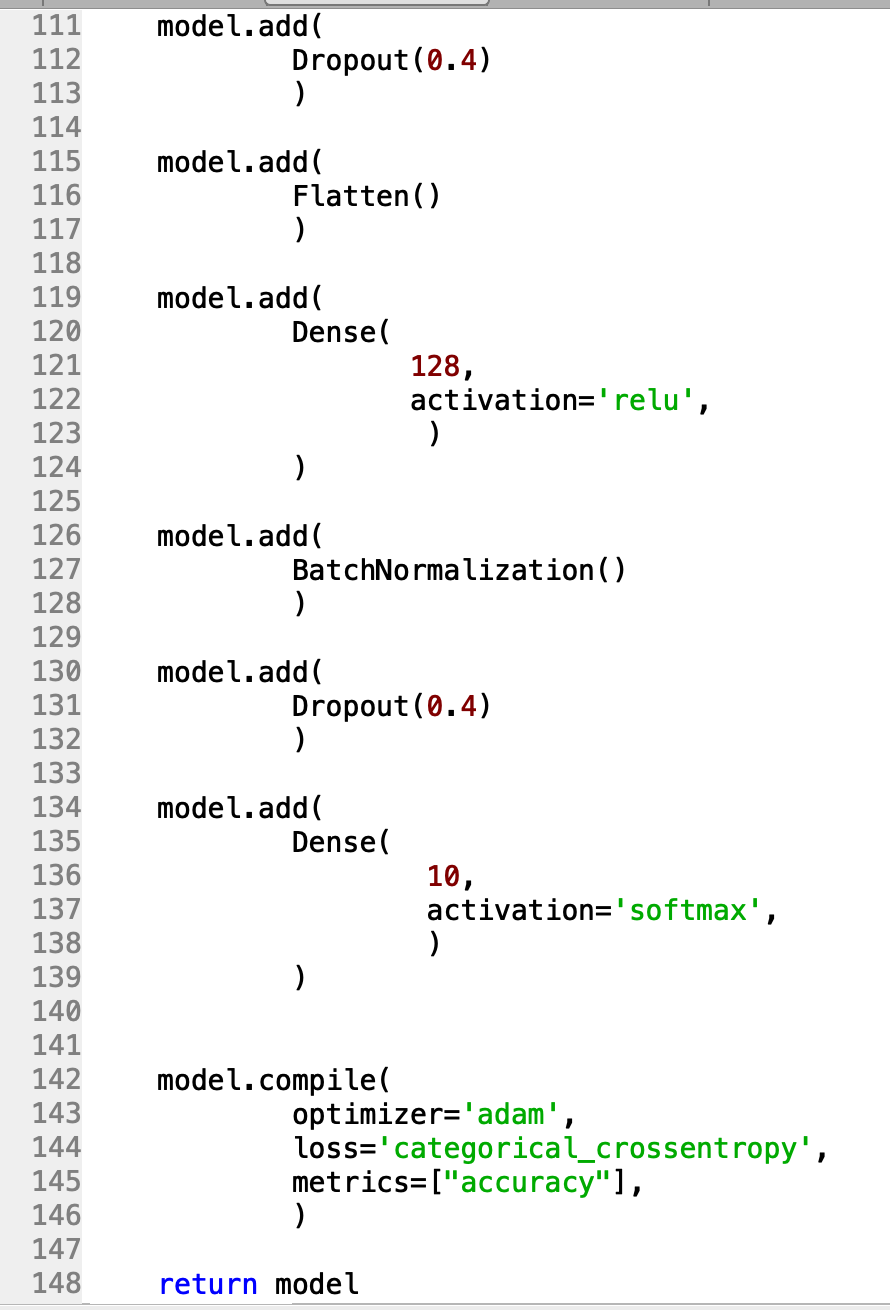
**Görsel 1**: *Modeli tanımlamak ve çağırabilmek için yaratılan model.py sınıfındaki koda ait görüntü.*



**Görsel 2**: *Modeli tanımlamak ve çağırabilmek için yaratılan model.py sınıfındaki koda ait görüntü.*

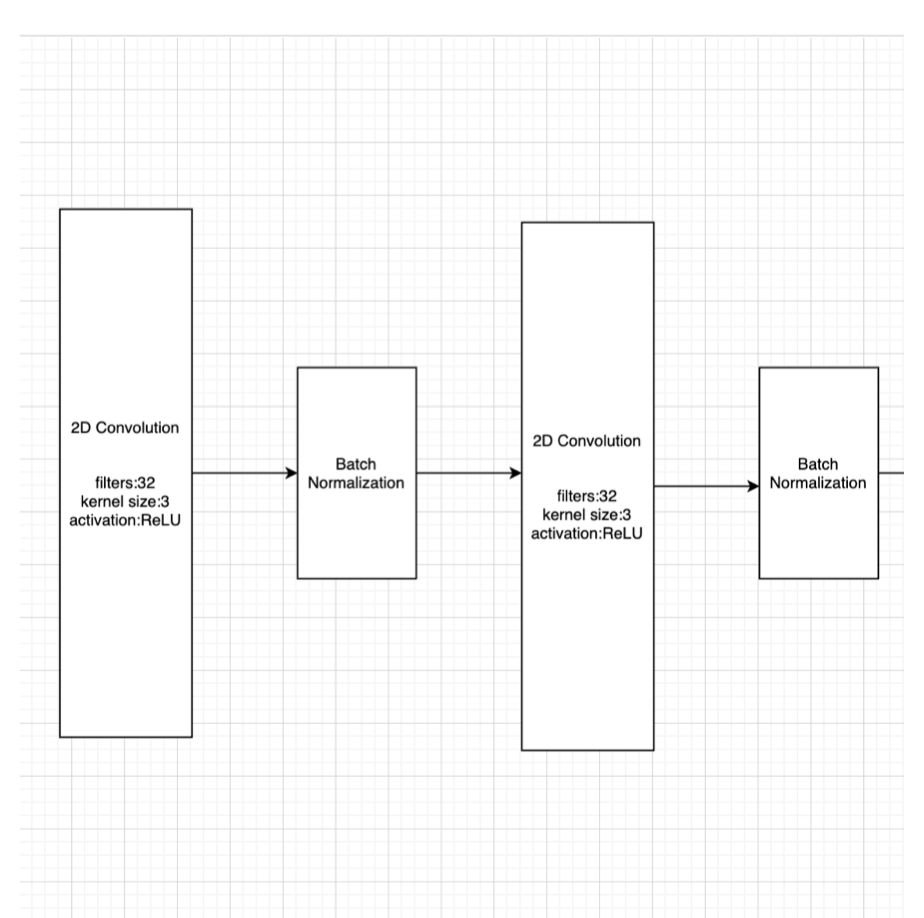


**Görsel 3**: *Modeli tanımlamak ve çağırabilmek için yaratılan model.py sınıfındaki koda ait görüntü.*

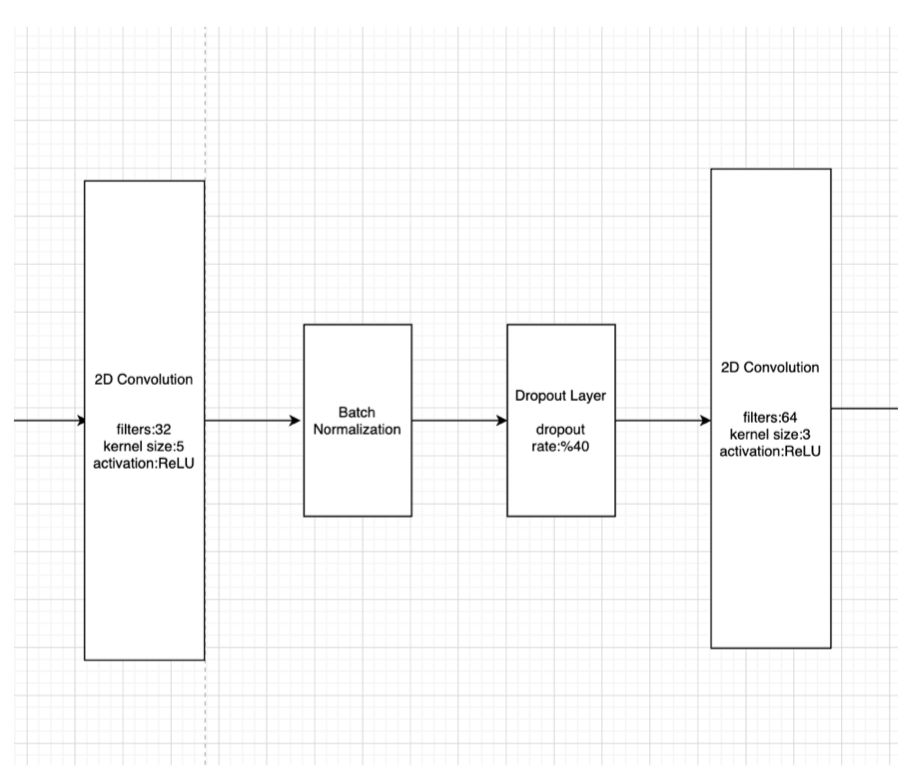


**Görsel 4**: *Modeli tanımlamak ve çağırabilmek için yaratılan model.py sınıfındaki koda ait görüntü.*

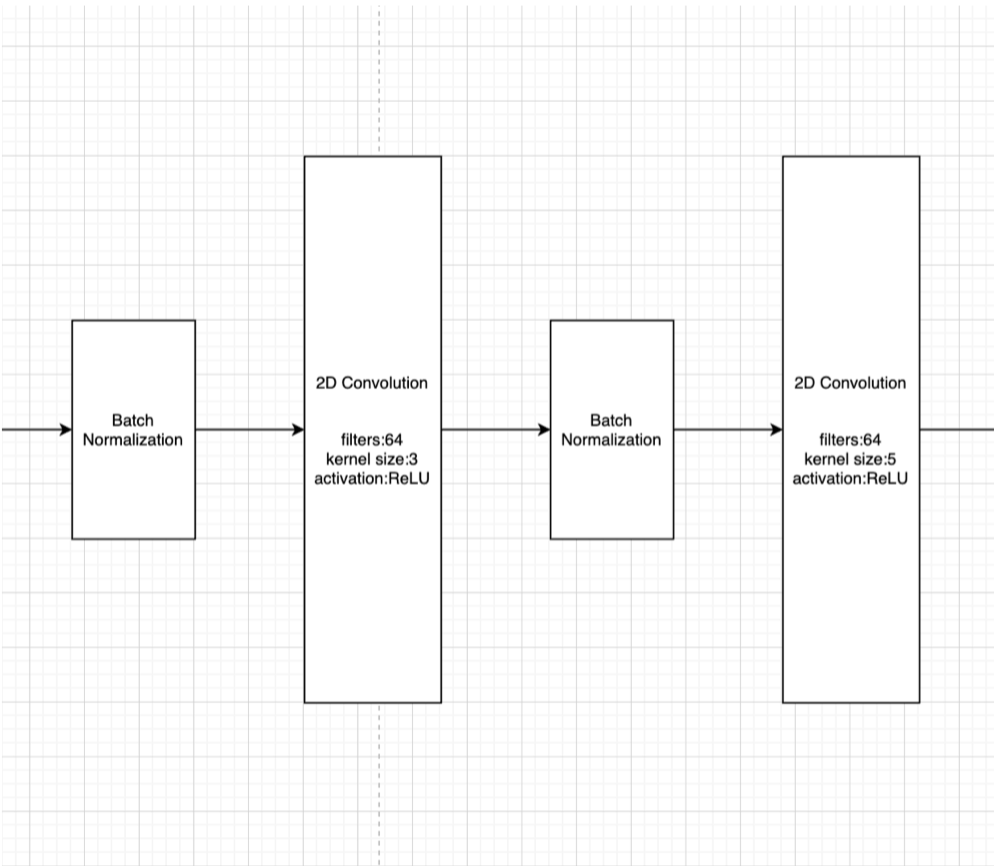
Bu görselere ek olarak modelin mimarisini ifade eden aşağıdaki çizgeleri de inceleyebilirsiniz.



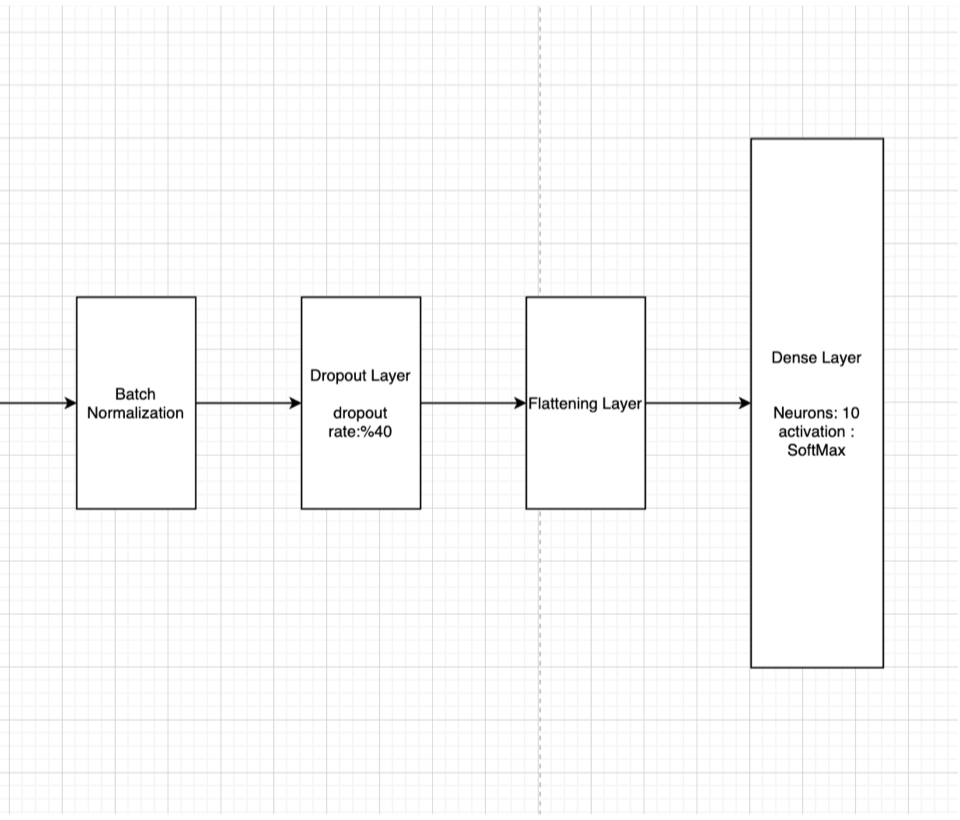
**Görsel 5**: *Model Şeması / Parça 1*



**Görsel 6**: *Model Şeması / Parça 2*



**Görsel 7**: *Model Şeması / Parça 3*



**Görsel 8**: *Model Şeması / Parça 4*

Modelin katmanlarına ve bu katmanların özelliklerine ait bilgileri de aşağıda inceleyebilirsiniz.

1. **Evrişim Katmanı** -> Filtre: 32, Kernel Boyutu:3, Aktivasyon: ReLU
2. **Batch Normalizasyonu Katmanı**
3. **Evrişim Katmanı** -> Filtre: 32, Kernel Boyutu: 3, Aktivasyon: ReLU
4. **Batch Normalizasyonu Katmanı**
5. **Evrişim Katmanı** -> Filtre: 32, Kernel Boyutu: 5, Stride: 2 Aktivasyon: ReLU
6. **Batch Normalizasyonu Katmanı**
7. **Dropout Katmanı** -> Dropout Oranı: %40
8. **Evrişim Katmanı** -> Filtre: 64, Kernel Boyutu: 3, Aktivasyon: ReLU
9. **Batch Normalizasyonu Katmanı**
10. **Evrişim Katmanı** -> Filtre: 64, Kernel Boyutu: 5, Stride: 2, Aktivasyon: ReLU
11. **Batch Normalizasyonu Katmanı**
12. **Evrişim Katmanı** -> Filtre: 64, Kernel Boyutu: 5, Stride: 2, Aktivasyon: ReLU
13. **Batch Normalizasyonu Katmanı**
14. **Dropout Katmanı** -> Dropout Oranı: %40
15. **Düzleştirme Katmanı**
16. **Yoğun Katman** -> Nöron Sayısı: 128, Aktivasyon: ReLU
17. **Batch Normalizasyonu Katmanı**
18. **Dropout Katmanı** -> Dropout Oranı: %40
19. **Yoğun Katman** -> Nöron Sayısı: 10, Aktivasyon: SoftMax

Yukarıda belirtilen katman bilgilerine bağlı olarak geliştirdiğim sinir ağı modelinde:

* 19 adet katman bulunmaktadır.
* 6 adet evrişimsel katman bulunmaktadır.
* İlk 3 evrişimsel katmanda 32 filtre kullanılmıştır.
* Son 3 evrişimsel katmanda 64 filtre kullanılmıştır.
* Optimizasyon yöntemi olarak Adam Optimizasyonu kullanılmıştır.
* Loss fonksiyonu olarak Categorical Crossentropy kullanılmıştır.

Modele ait gerekli tanımları ve gerçekleştirdikten sonra eğitim sürecinin başlatılması için gerekli kodun yazılması gereklilik haline geldi. Bu amaçla modelin eğitimi için kullanılacak olan “training.py” isminde bir Python dökümanı oluşturdum.

Bu doküman dahilinde öncelikli olarak keras, numpy vb. gerekli üçüncü parti kütüphaneleri import ettim ve elbette modele ait mimariyi çağırabilmek için “model.py” modülünü import ettim.

Sonrasında batch boyutu, sınıf sayısı ve eğitimin gerçekleştirileceği epoch sayısı gibi belirli hiperparametreleri tanımladım. MNIST veri setini keras.datasets modülünü kullanarak eğitim ve test setleri ayrı olacak şekilde çağırdım. Theano ve Tensorflow’un görüntü kanallarını ifade ediş formatları birbirinden farklı olduğu için karışıklığı önlemek amacıyla güvence sağlayan ufak bir kod yazdım.

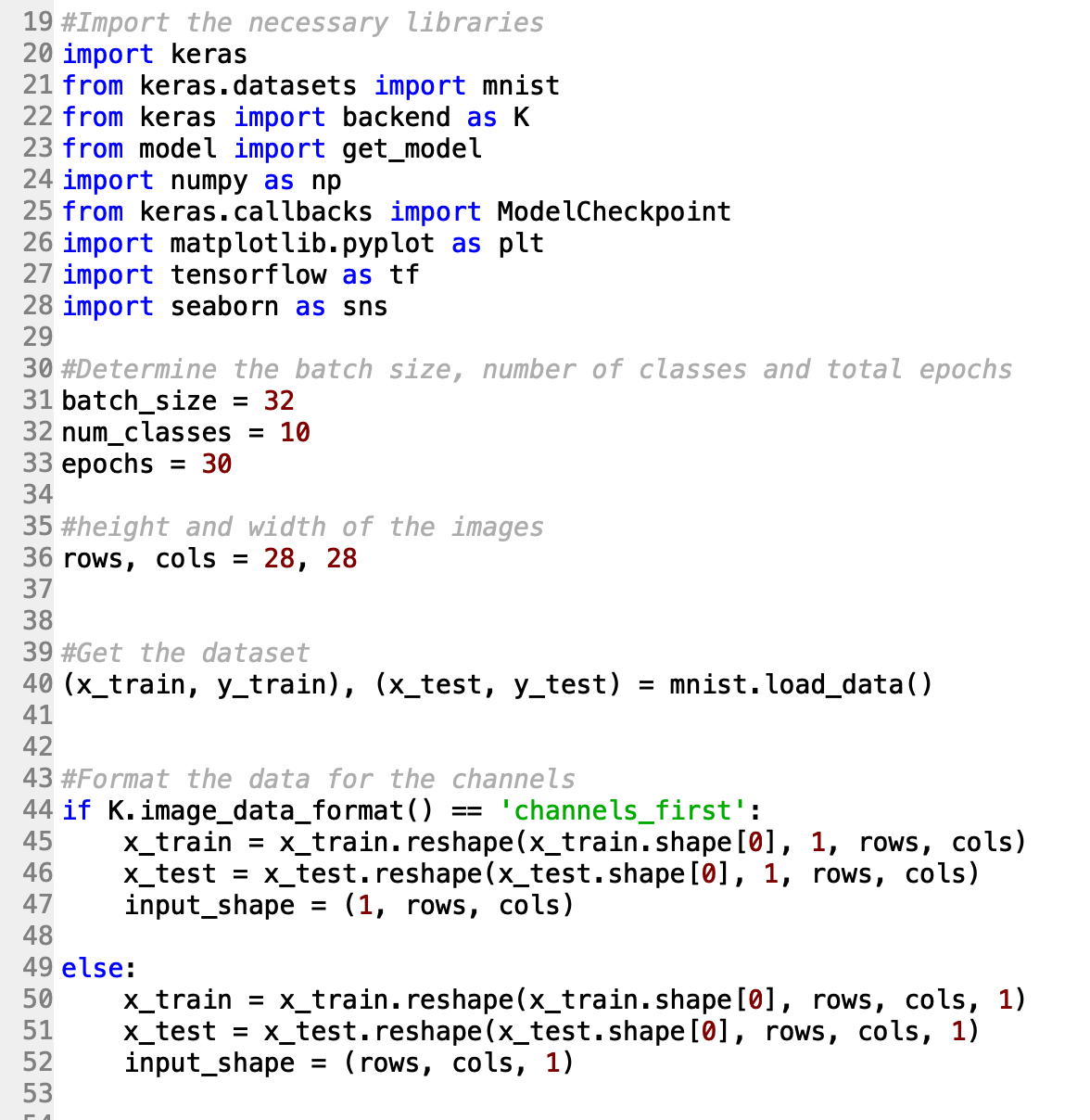
Sonrasında görüntülerin normalizasyonu ve ölçeklendirilmesi için kanal değerlerini 0 ve 1 arasına denk gelecek şekilde ayarladım. Sonrasında etiket değerlerini kategorik değerlere dönüştürdüm.

Önceki görsellerde model.py sınıfında tanımının gerçekleştirildiği modelimi ilgili get\_model metodunu kullanarak çağırdım. Oluşturduğum modelin h5 formatında kaydedilebilmesi ve modelle ilgili gelişmelerin eğitim süresince kaybolmaması için ModelCheckpoint sınıfından faydalandım.

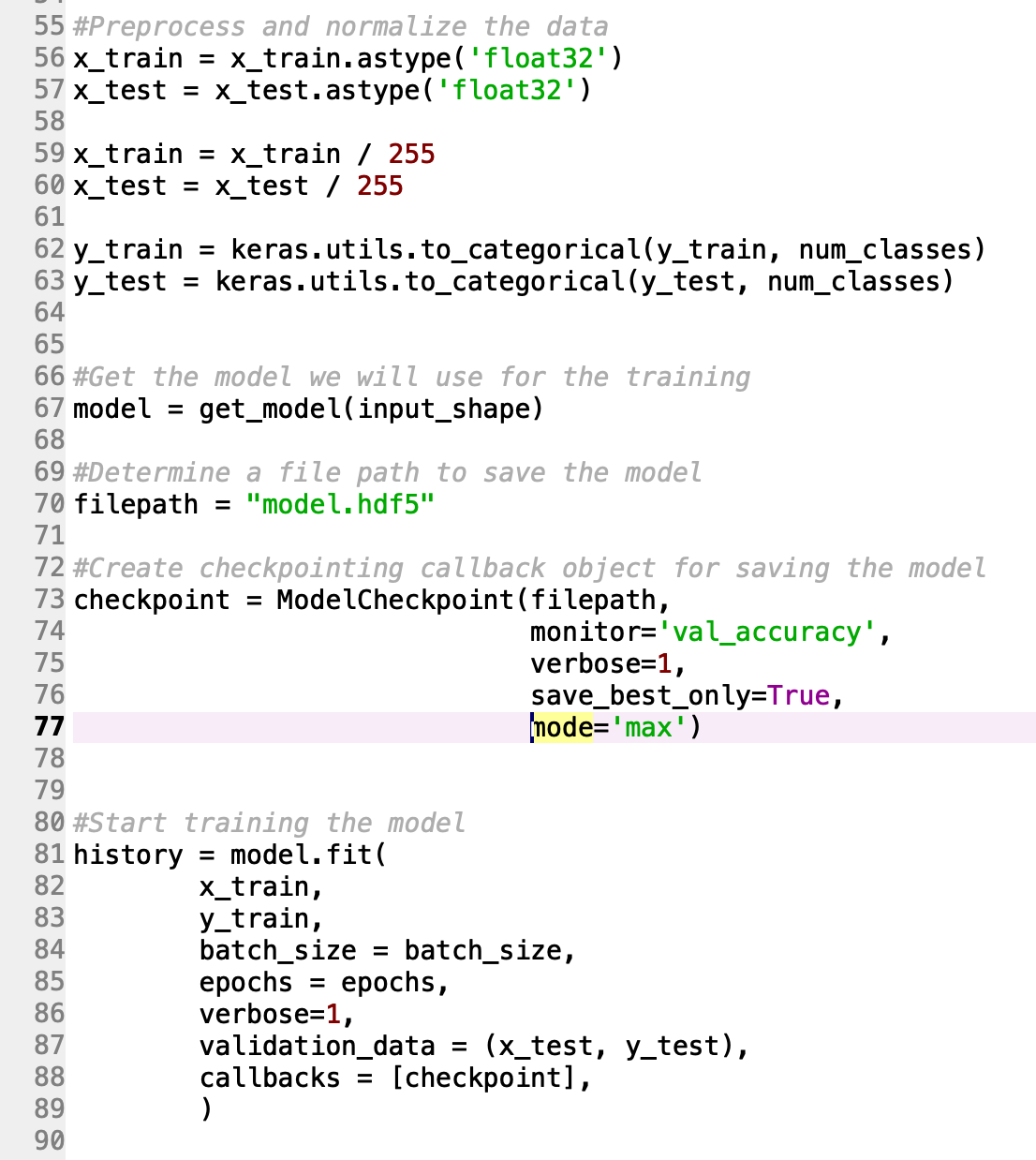
Ardından eğitim verilerini, belirlenen epoch sayısı boyunca eğitilmek üzere modele entegre ettim ve eğitimin tamamlanmasını bekledim.

Modelin oluşturduğu Loss ve Accuracy değerlerini kullanıcıya görüntülemenin ardından örnek tahminlemeler gerçekleştirdim. Sonrasında Modelin eğitim sürecine ait Loss ve Accuracy değerlerinin zaman içerisindeki değişimlerini grafikler halinde gösterdim. Son olarak tahminlenen örneklerin doğruluklarını bir Confusion Matrix üzerinde göstererek kullanıcıya görselleştirdim.

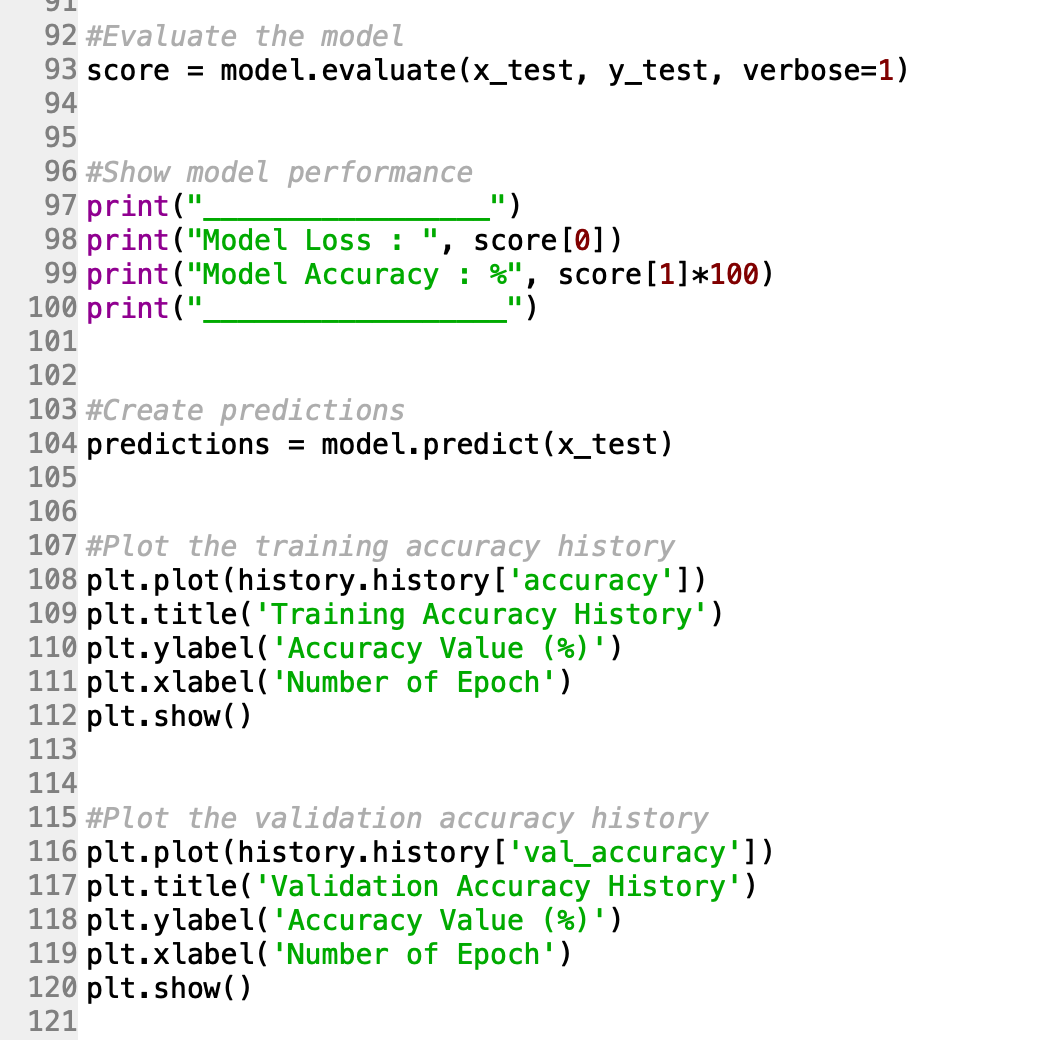
Training.py isimli Python model eğitim dökümanına ait kod görselleri aşağıda belirtilmiştir.



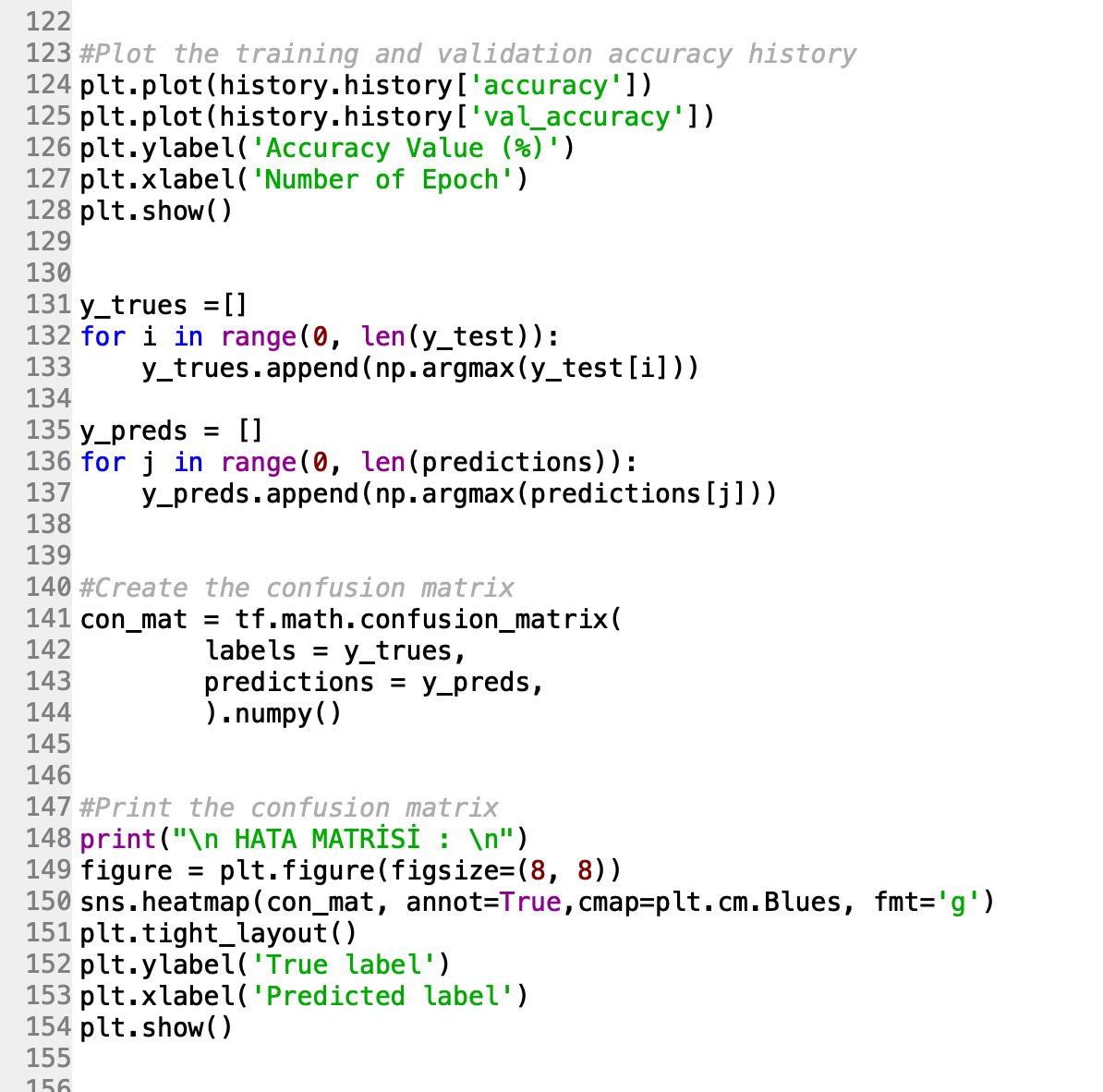
**Görsel 9**: *Training.py dökümanına ait kodların görüntüsü*

**

**Görsel 10**: *Training.py dökümanına ait kodların görüntüsü*

**

**Görsel 11**: *Training.py dökümanına ait kodların görüntüsü*

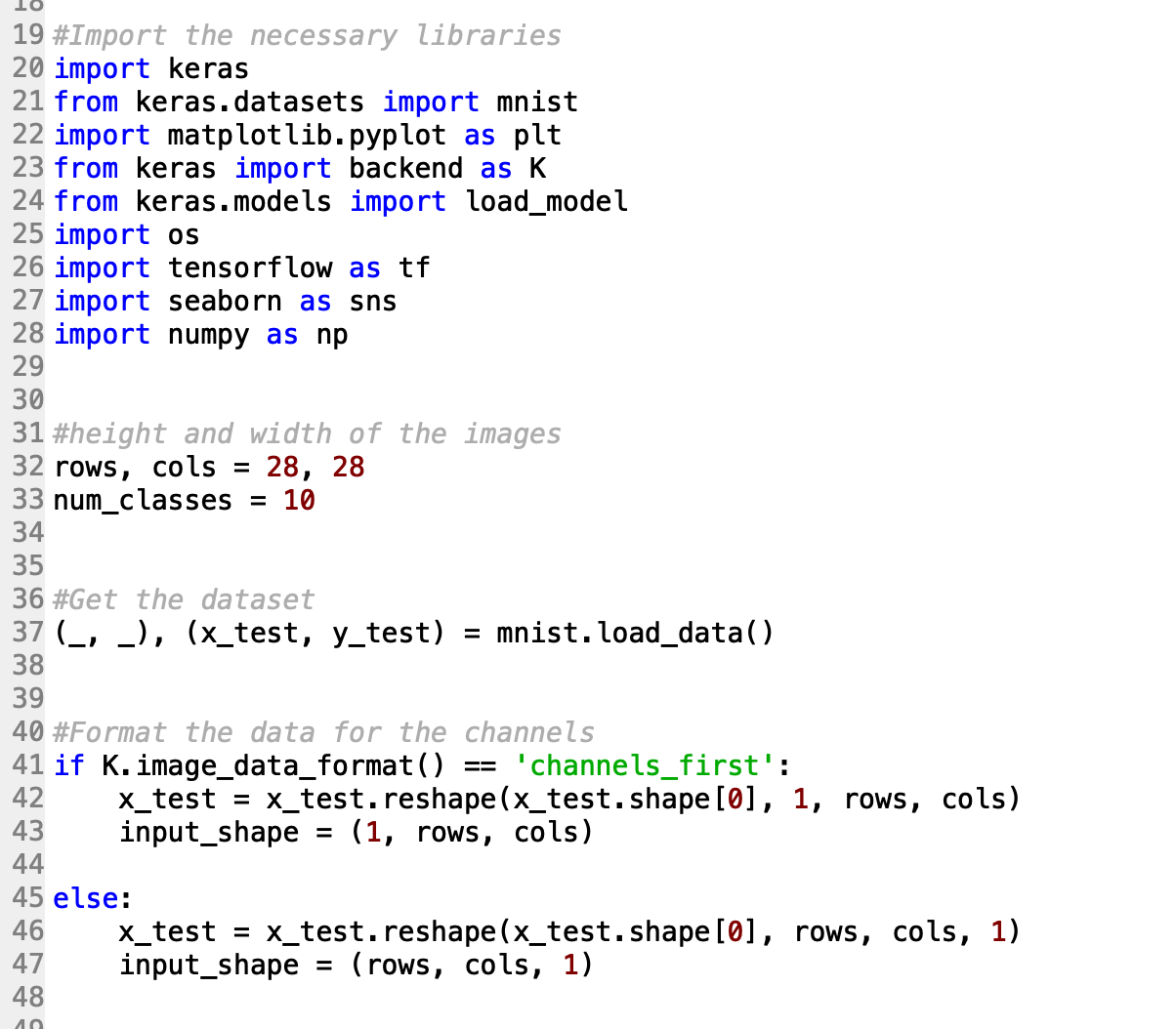
**

**Görsel 12**: *Training.py dökümanına ait kodların görüntüsü*

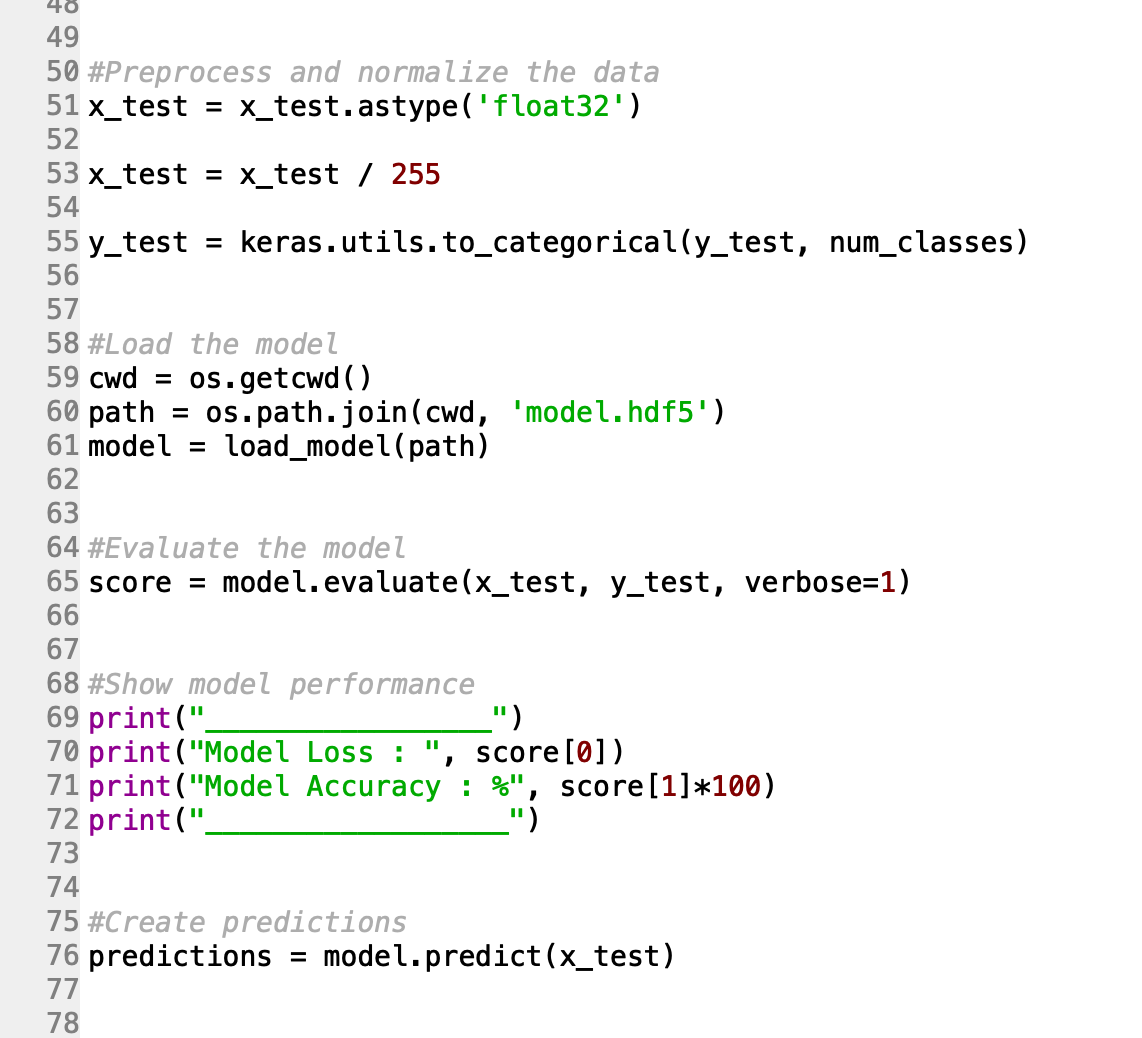
Modelin oluşturulması ve kaydedilmesinden sonra kaydedilen modelin sınanması için “test\_model.py” isminde ayrı bir döküman meydana getirilmiştir.

Bu doküman dahilinde MNIST veri setine ait sadece test verileri alınır ve bu veriler kullanılarak oluşturulan model üzerinde performans değerlemesi gerçekleştirilir. Sonuç olarak elde edilen Loss ve Accuracy değerleri kullanıcıya görüntülenerek bir Confusion Matrix kullanılarak da ifade edilir.

Bahsedilen test\_model.py Python dökümanına ait kod görselleri aşağıda paylaşılmıştır.



**Görsel 13**: *test\_model.py dökümanına ait kodların görüntüsü*

**

**Görsel 14**: *test\_model.py dökümanına ait kodların görüntüsü*



**Görsel 15**: *test\_model.py dökümanına ait kodların görüntüsü*

*(****Modelin optimize edilmesi ve K-Fold Cross Validation*** *kullanılarak iyileştirilmesine yönelik çalışmalar* ***4 numaralı başlık*** *altında anlatılmıştır.)*

**3.3.3 Hangi çalışma / hazır kod / bağlantılardan faydalandınız? Faydalandığınız çalışma ve kaynaklara göre farklılıklarınız nelerdir? Açıklayınız. (10)**

MNIST veri seti oldukça popüler bir veri seti olduğundan internette başlangıç düzeyinden, ileri düzeye kadar birçok eğitim koduna ulaşmak mümkün. Benzer çalışmalara yönelik çeşitli araştırmalar yaptığımda bilinen en iyi MNIST tanımlayıcı modelinin ulaştığı doğruluk değerinin %99,8 olduğunu tespit ettim. Benim geliştirdiğim modelin ise ulaşabildiği maksimum doğruluk oranı yaklaşık %99,62. Kaggle üzerinden gerçekleştirilen çalışmalarda ise ulaşılabilen en yüksek doğruluk oranı %99,75. [11]

Kaggle üzerinden çeşitli yöntemler ile bu çözümlemeyi gerçekleştirmeye çalışmış insanların çalışmalarını inceledim. %100’e ulaştığını iddia eden çeşitli kullanıcılar olsa da, bu tarz kesinlikte bir doğruluk oranının olası aşırı eğitilme durumuyla ilişkili olabileceğini düşünmekteyim. Nispeten basit CNN modelleri kullanılarak %98 civarında doğruluk düzeyine ulaşabilen örnekler mevcut. [12] Gözlemlerime göre, genellikle yeteri kadar etkin ve güçlü dizayn edilmiş bir sinir ağı söz konusuysa doğruluk oranları %99 ve üzerinde oluyor. Yetersiz eğitim süresi, yetersiz nöron sayısı, yetersiz katman sayısı, az sayıda filtre gibi çeşitli etkenler doğruluk oranını negatif yönde etkileyebiliyor.

İncelediğim kaynaklara göre temel farklarımdan biri, projede kullandığım evrişimsel yapay sinir ağı modelini tasarlarken yüksek sayıda evrişim katmanı kullanmam. Bu evrişim katmanlarında da bilgisayarımın işlem gücünün el verdiği kadar yüksek sayıda filtre kullanmaya çalıştım. Bir sonraki bölümde de bahsedeceğim üzere, filtre sayısını düşük tutmak modelin doğruluk düzeyini negatif yönde etkiledi.

**Not:** Yukarıda belirttiğim kaynaklara ek olarak, keras, matplotlib ve sklearn kütüphanesinin dökümentasyonlarından da kaynak olarak faydalandım.

**4. Eğitim işlemini nasıl yaptığınızı (K-Fold CV / Eğitim / Test verisi sayısı, … ) raporda anlatınız. Katman sayısı, epok sayısı, filtre boyutları, seyreltme oranı ve diğer parametre değerlerini değiştirerek sonuçlarınızı iyileştirmeye çalışınız. Hazır bir araç ile hiperparametre optimizasyonu da yapabilirsiniz. Parametre değerlerine karşılık aldığınız Loss, Accuracy gibi başarı sonuçlarını içeren bir tablo ve grafik oluşturunuz. Rapora ekleyiniz. Başarımı arttırmak için hangi yöntemleri kullandığınızı ve ne kadar artış olduğunu belirtiniz. (25)**

**4.4.1 Eğitim İşleminin Anlatımı**

Eğitimde toplamda 2 farklı yöntem kullandım. Birincisi herhangi bir K-Fold çapraz doğrulama yöntemi kullanmadan eğitim gerçekleştirmek; diğeri de 5 katmanlı çapraz doğrulama kullanılarak eğitim gerçekleştirmekti. Esasen, 5 katmanlı çapraz doğrulamayı birincil olarak kullanmak isterdim ancak yüksek epok sayısı ve modelin karmaşıklığı gibi çeşitli etkenlerden dolayı bu durum eğitim süresini aşırı uzatma gibi bir soruna sebep oldu. Elbette daha güçlü bir sistem ile daha ideal bir eğitim süreci gerçekleştirilebilir. 5 katlı çapraz doğrulamayı kabul edilebilir sürelerde gerçekleştirebilmek için epok sayısını 12’ye indirmek zorunda kaldım. Oysa tek tur eğitim gerçekleştirdiğim diğer eğitim metodunda epok sayısı 30 turdu. Bu sayede toplam eğitim süresini normal ve kabul edilebilir bir düzeye indirmeyi başardım.

Veri setinde toplam 60.000 adet eğitim görseli ve 10.000 adet test görseli bulunmaktaydı. Öncelikli olarak bu görsellerin ölçeklenmesi ve normalize edilmesi için gerekli işlemleri gerçekleştirdim. Ardından kategorik etiketleme için gerekli işlemleri gerçekleştirdim. Eğitim sırasında validasyon verisi olarak test verilerini kullandım. 5 katmanlı çapraz doğrulama sırasında da ilk yöntemde olduğu gibi veri bölümlendirmesi yaptım ancak elbette çapraz doğrulamada her tur sırasında veriler çeşitli şekilde karıştırılıp farklı gruplar halinde seçildiler. Bu sayede modele yönelik doğruluk değerleri konusunda herhangi bir Bias olup olmadığını; dolayısıyla sonucun olması gerekenden iyi veya kötü olup olmadığını daha etkin bir şekilde sınamış oldum.

Verilerin eğitimi sırasında kullandığım model çok katmanlı bir evrişimsel sinir ağı modeliydi. Ayrıca modelin eğitimi sırasında gelişmeler oldukça sinir ağındaki ağırlıkların kaydedilebilmesini sağlayacak Callback metotlarını kullandım. Loss fonksiyonu olarak Categorical Crossentropy, Optimizasyon fonksiyonu olarak da Adam Optimizasyonunu tercih ettim.

**4.4.2 Tablo ve Grafik (Parametre Optimizasyonu). Başarımı arttırmak için hangi yöntemleri kullandığınızı ve ne kadar arttığını belirtiniz.**

Başarımı arttırmak için kullandığım çeşitli teknikler ve denediğim yöntemler aşağıdaki gibidir.

* Evrişimsel sinir ağındaki katman sayısını arttırmak
* Evrişimsel katmanlardaki filtre sayılarının yüksek tutulması
* Batch Normalizasyon katmanlarının kullanılması
* Ezberlemeyi engellemek için Dropout katmanları kullanılması
* Görüntü tanımlama amacı için daha etkin olduğu düşünülen ReLU aktivasyon metotunun tercih edilmesi
* Veri setinin eğitime sokulmadan önce ölçeklendirilmesi
* Düzleştirme katmanı kullanılması
* Son katmanda olasılıksal çıkarımda daha etkin olacak Softmax aktivasyon metodu kulanılması.
* Sondan bir önceki Dense Katmanda yeterli olacak sayıda nöron kullanılması
* Performans verimliliği ve doğruluk arasında olabildiğince optimize edilmiş noktada bir batch boyutu kullanmak.

Model, ilk versiyonunda 2 katman evrişimsel katmana ve ardından gelen bir havuz katmana sahipti. Bunun ardından düzleştirme katmanı ve 64 nörondan oluşan bir yoğun katman geliyordu. Bu halindeyken performansı daha düşüktü. (~%88)

Filtre sayıları da 16 olup modelin son haline göre daha düşüktü. Filtre sayılarını arttırmak ve 32’ye çıkarmak modelin performansında küçük miktarda iyileşmeye yol açtı. (~%89) Ardından yoğun katmandaki nöron sayısını arttırmayı denedim. 128, 256, 512 gibi çeşitli değerler, eğitim süresinde uzamaya yol açarken doğruluk değeri üzerindeki değişim anlamlı boyutta sadece 128’de meydana geldi.

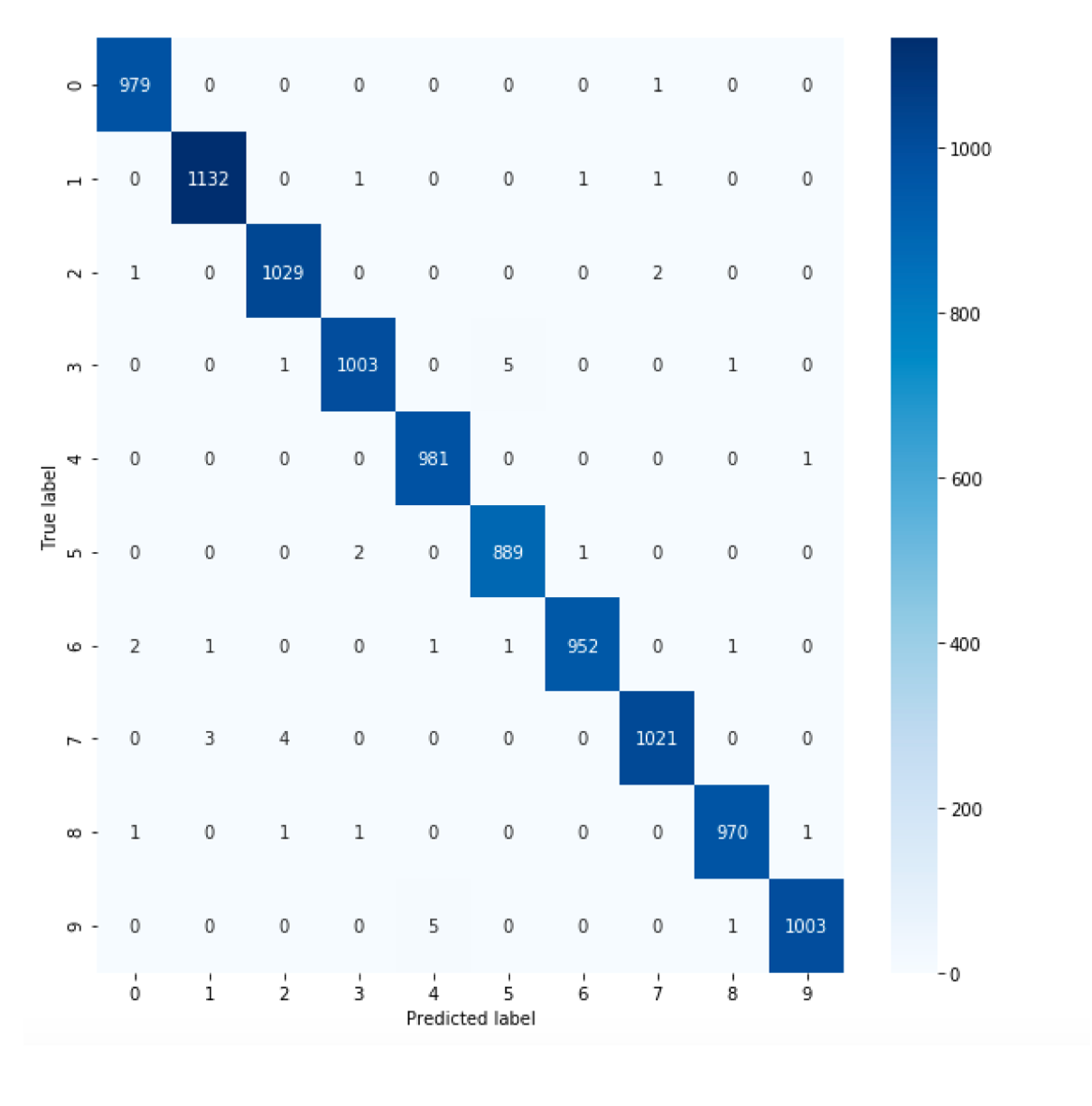
Son katman için 128 nöron sayısının uygun olduğu kanaatine vardım. Ardından evrişimsel katmanların sayısını arttırmanın verimli olabileceği düşüncesine dayanarak uygulamaya koyuldum. Toplamda 6 adet evrişimsel katmanın bulunduğu yeni bir model denedim ve doğruluk oranında ciddi ilerlemeler gördüm. İlk 3 katman için 32, ardındaki 3 katman için 64 filtre kullandım. Doğruluk oranları ulaştığım en yüksek sonuç olan %99.62’ye yaklaşmaya başladı. Batch normalizasyon katmanları ve Dropout katmanları kullanarak modelin aşırı öğrenme, ezberleme gibi durumları minimize ederken performansını maksimize etmek konusunda çaba gösterdim.

Batch sayısını daha düşük tutmayı denediğimde eğitim süresinde ciddi uzamalara tanık oldum. Buna ek olarak çok fazla arttırdığımda ise belli bir noktadan sonra yine eğitimin yavaşlamaya başladığını gördüm. Bu sebepten dolayı 32 değerinin kullanılmaya uygun olduğuna karar kıldım.

Eğitim süresi konusunda ise öncelikli olarak 10 epok boyunca eğitim gerçekleştirilmesini denedim. Fakat 10. Epok’un sonunda veri seti öğrenimini hala gerçekleştirdiğinden sonrasında 20 epok denedim. Ufak ilerlemeler olsa daha hala öğrenme faaliyetinin gerçekleştiğini gördüm ve bu sebepten ötürü 30 epok kullanmak konusunda karar kıldım.

Bütün bu düzenlemeler ve ayarlamalar göz önüne alındığında modelin ilk versiyonu ile son versiyonu arasında **~%11** kadar bir doğruluk oranı artışı sağladığımı söyleyebilirim.

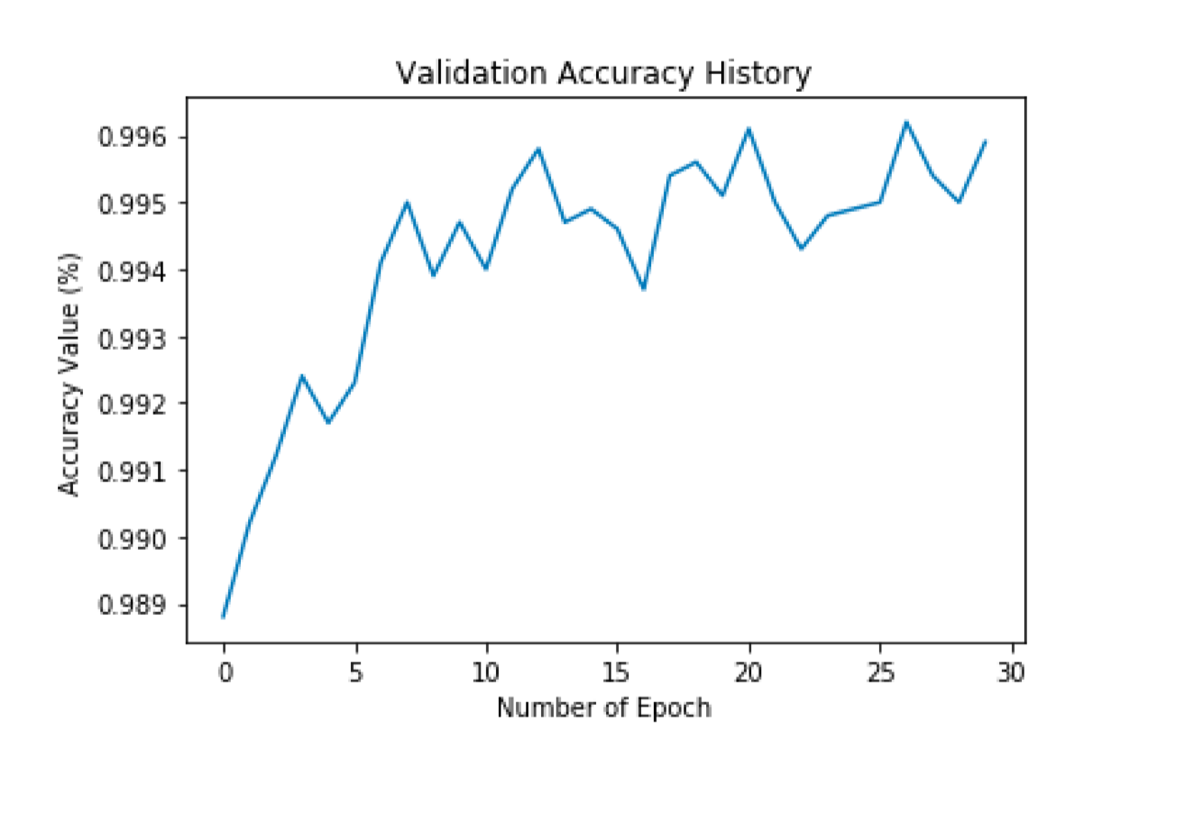
Aşağıda eğitim sürecine ait çeşitli grafiklerin ve görselleştirmelerin yer aldığı şekilleri inceleyebilirsiniz.



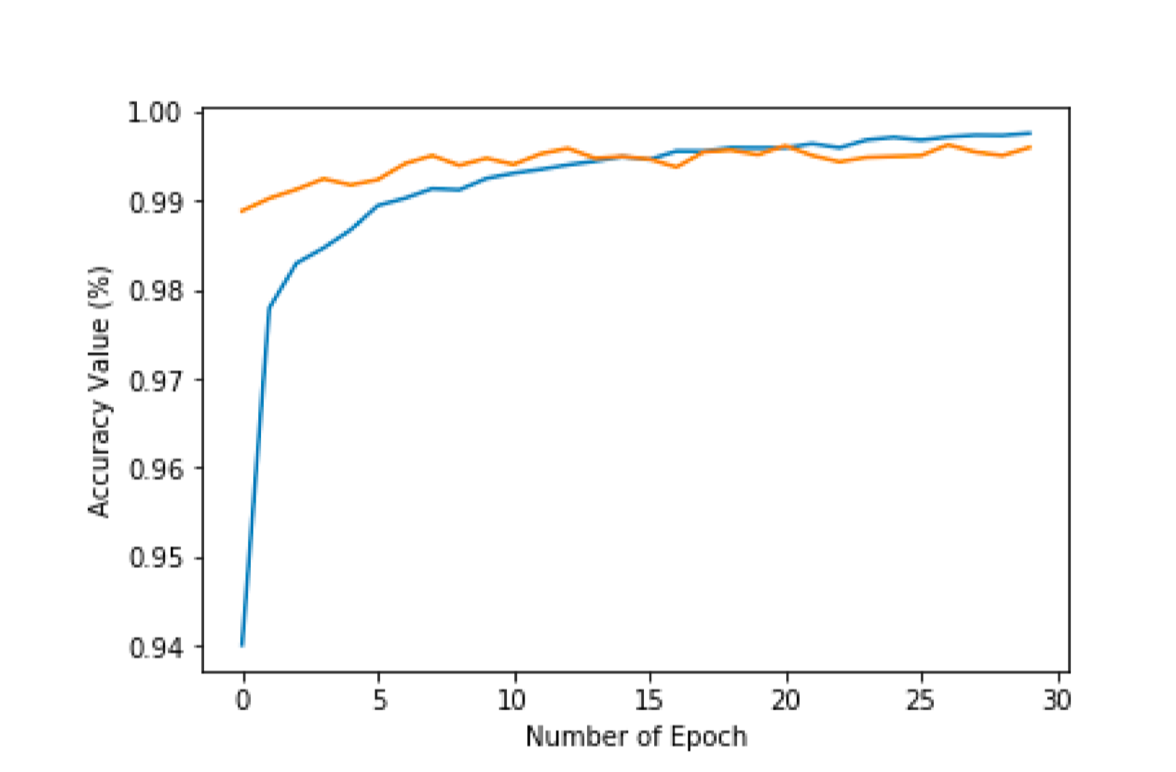
**Görsel 16**: *Eğitim süreci sonrası test seti üzerinde yapılan değerlendirmen elde edilen hata matrisi*

**

**Görsel 17**: *Eğitim setine ait doğruluk değerinin eğitim turları süresince değişiminin grafiği*

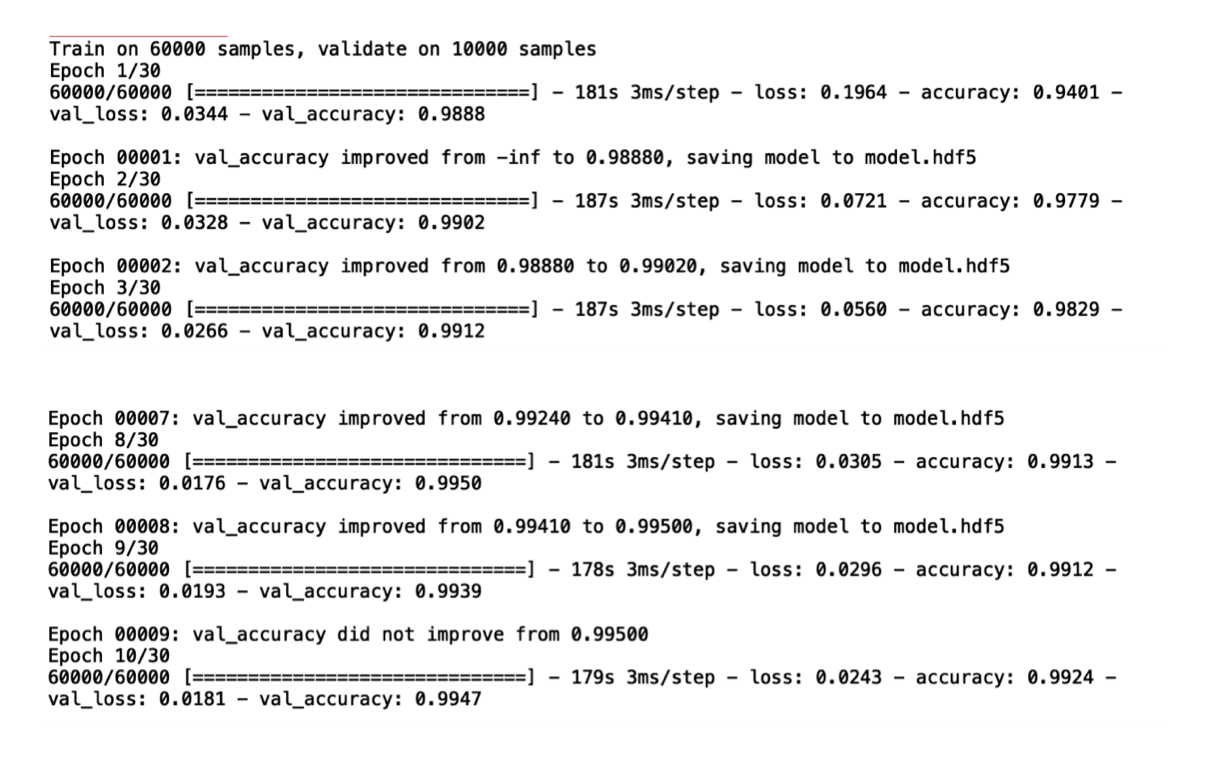
**

**Görsel 18**: *Test setine ait doğruluk değerinin eğitim turları süresince değişiminin grafiği*

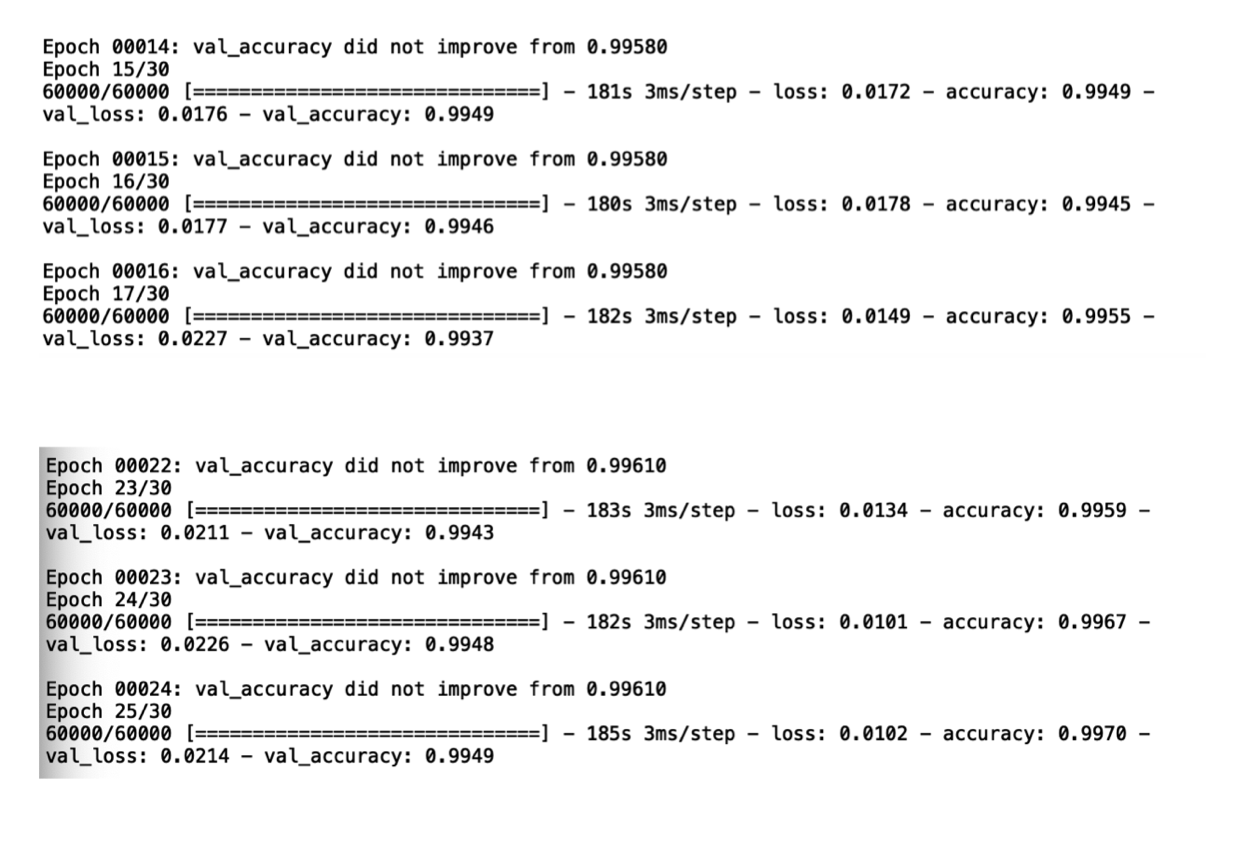
****Görsel 19**: *Eğitim ve Test setine ait doğruluk değerinin görselleştirilmesi*

**

**Görsel 20**: *Eğitim sonucunda ulaşılan Loss ve Accuracy değerlerinin gösterildiği görüntü*



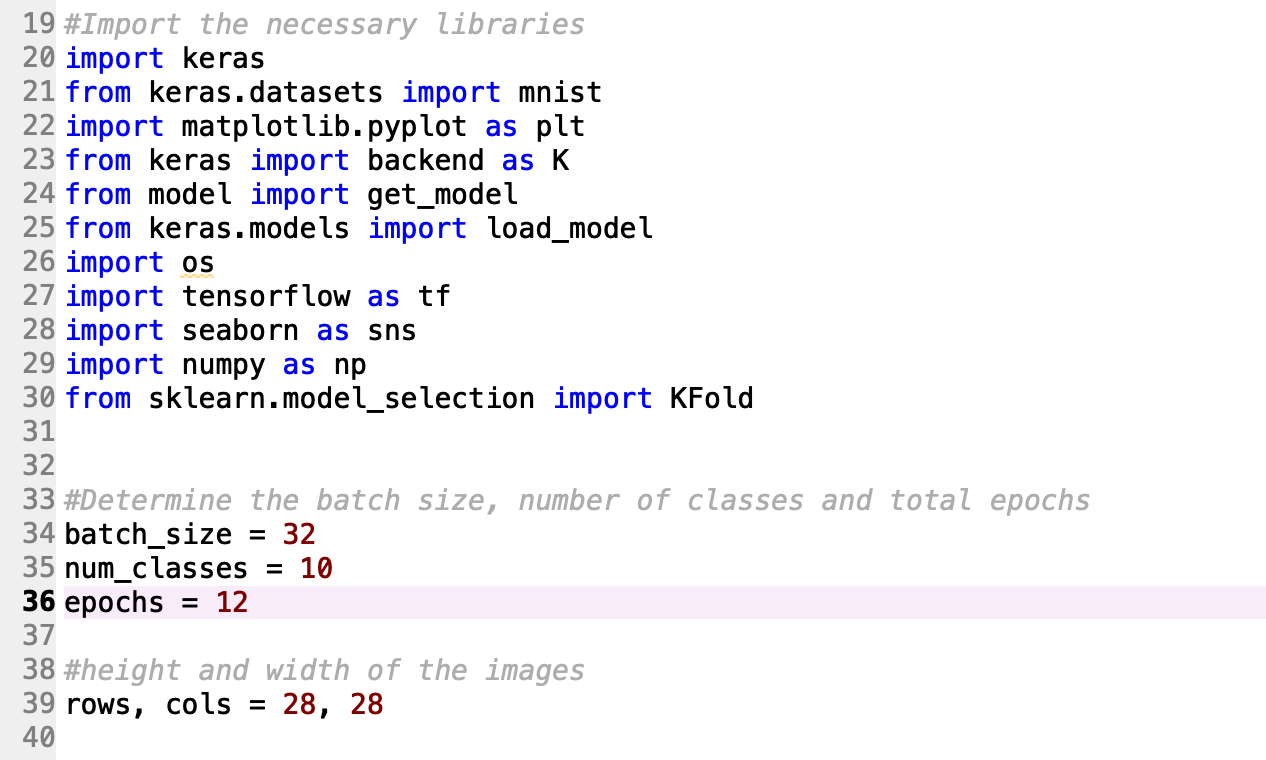
**Görsel 21**: *Eğitim sürecinden bir görüntü*

**

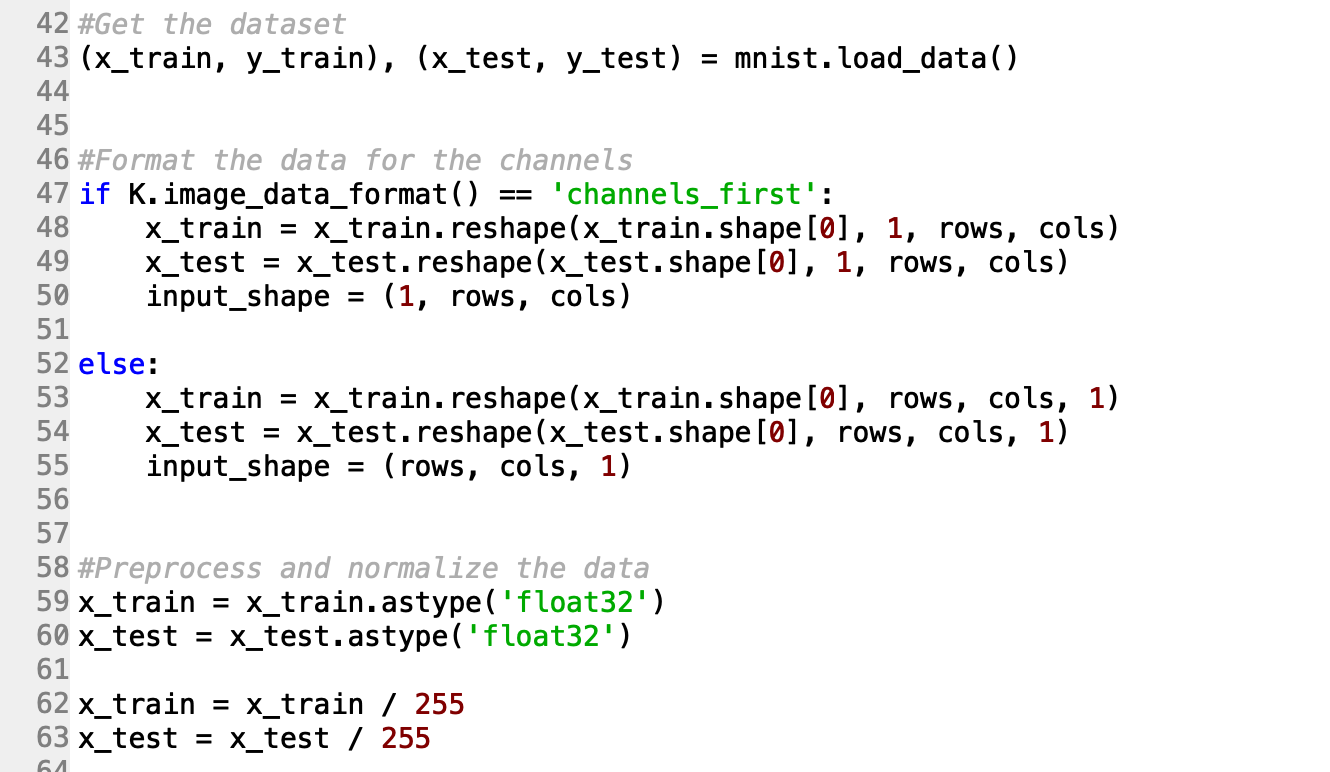
**Görsel 22**: *Eğitim sürecinden bir görüntü*

Yukarıda belirtilenlere ek olarak, bir de daha düşük epok sayısının kullanıldığı, fakat 5 kat çapraz doğrulama gerçekleştirilen bir eğitim de uygulanmıştır. Bu eğitimde kullanılan model aynı olup, sadece epok sayısı üzerinde farkılaştırma uygulanmıştır. Farklılaştırma gerçekleştirilmesinin temel sebebi, önceki gibi 30 epok ile eğitim gerçekleştirmenin eğitim süresinde aşırı artışa sebep olmasıdır. Bu nedenle, 30 yerine 12 epok eğitim gerçekleştirdim.

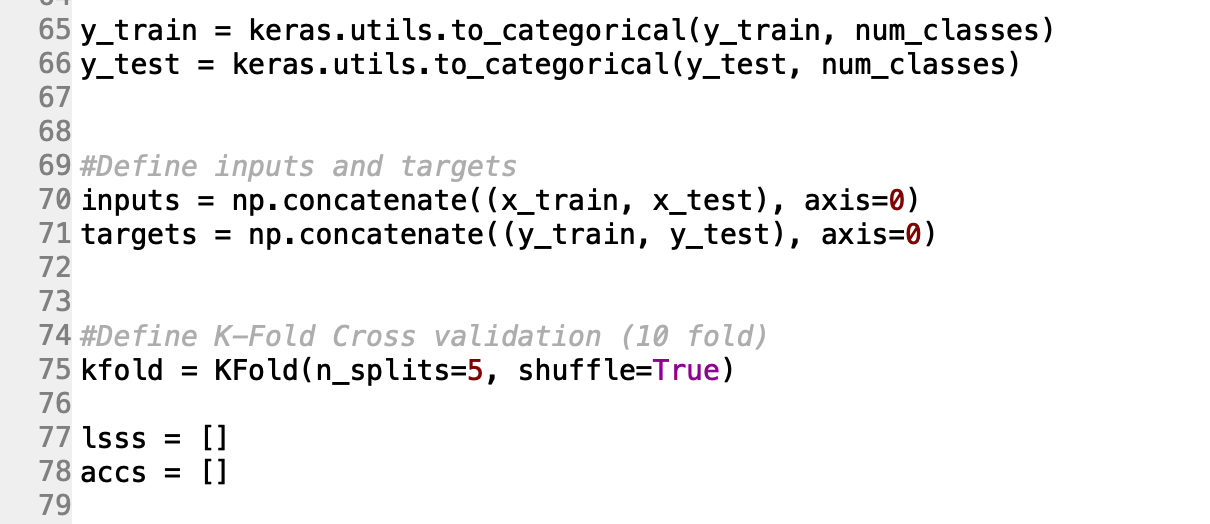
Aşağıda çapraz doğrulama kullanarak eğitim gerçekleştirdiğim Python dökümanına (kfold.py) ait kod görsellerini paylaşıyorum.



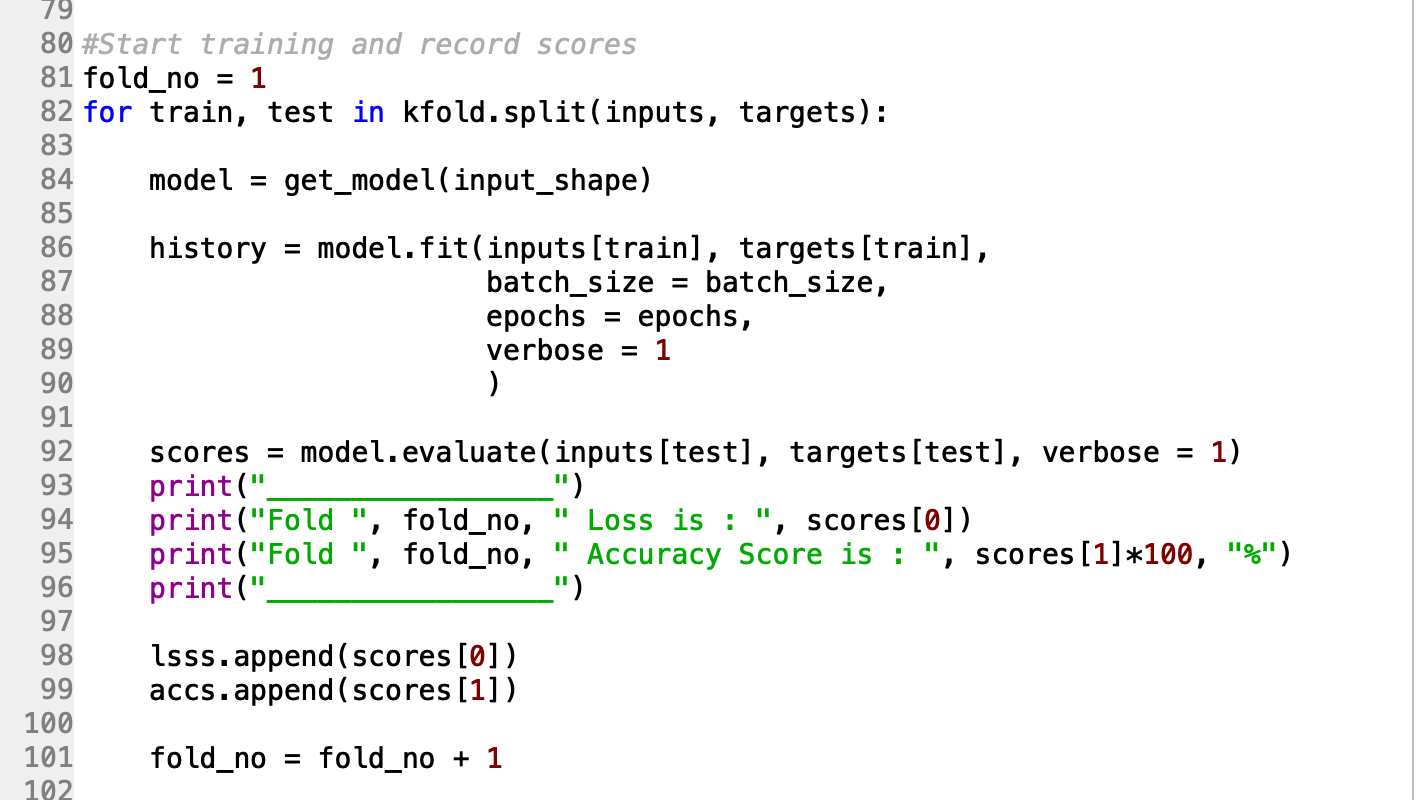
**Görsel 23**: *kfold.py dökümanına ait kodun bir görüntüsü*



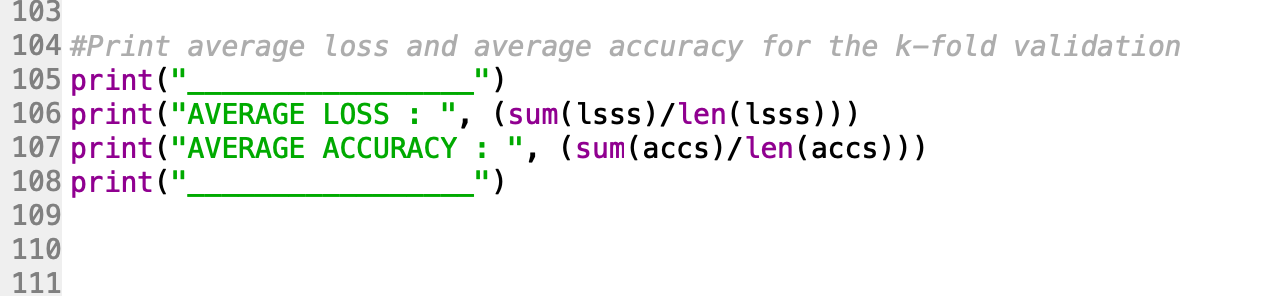
**Görsel 24**: *kfold.py dökümanına ait kodun bir görüntüsü*

**

**Görsel 25**: *kfold.py dökümanına ait kodun bir görüntüsü*

**

**Görsel 26**: *kfold.py dökümanına ait kodun bir görüntüsü*



**Görsel 27**: *kfold.py dökümanına ait kodun bir görüntüsü*

5 katlı çapraz doğrulama uygulandıktan sonra elde edilen doğruluk oranlarına yönelik çıktılar aşağıdaki gibidir.

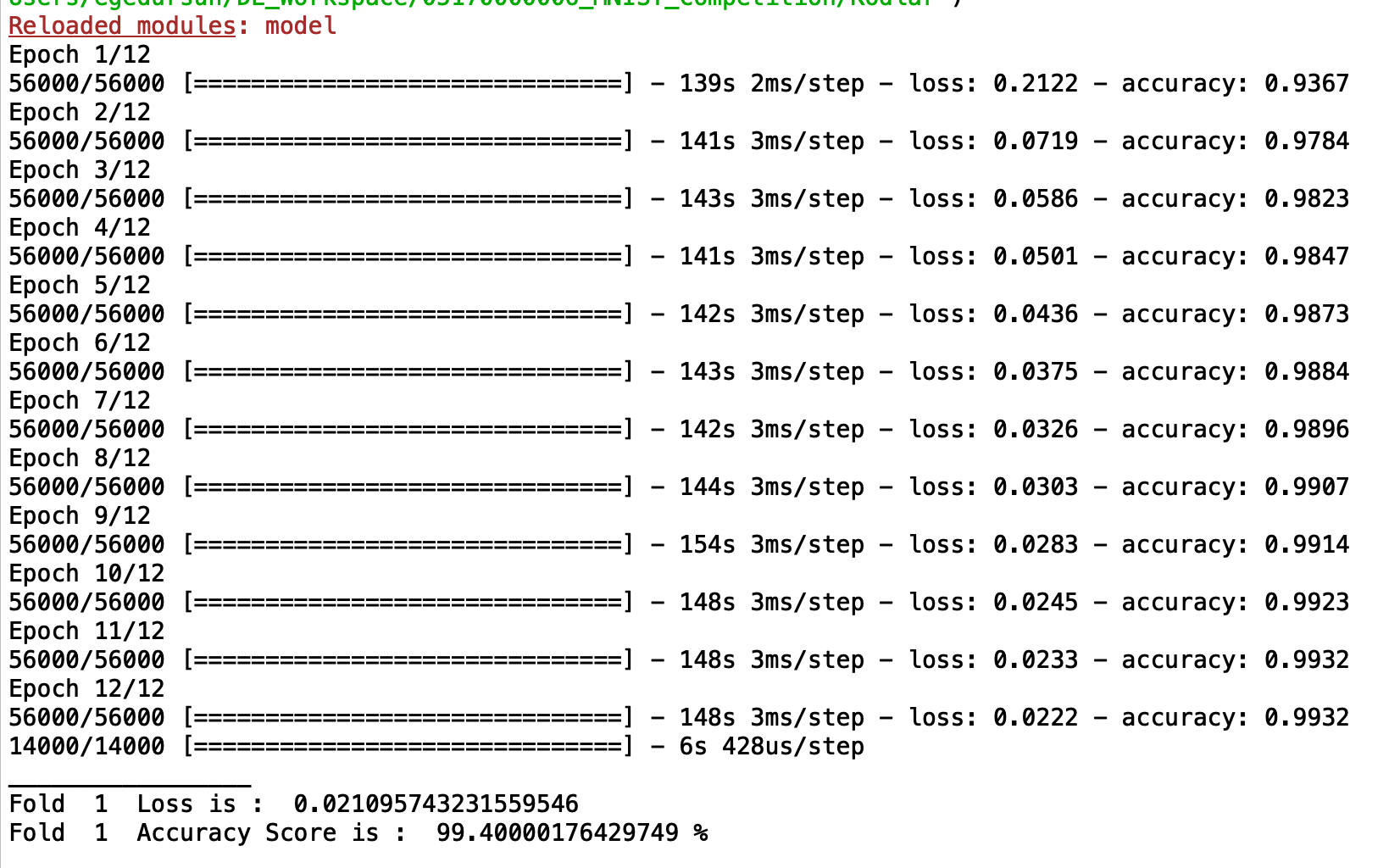
1. ~%99.40
2. ~%99.33
3. ~%99.38
4. ~%99.47
5. ~%99.34

**Ortalama**: **~%99.38**

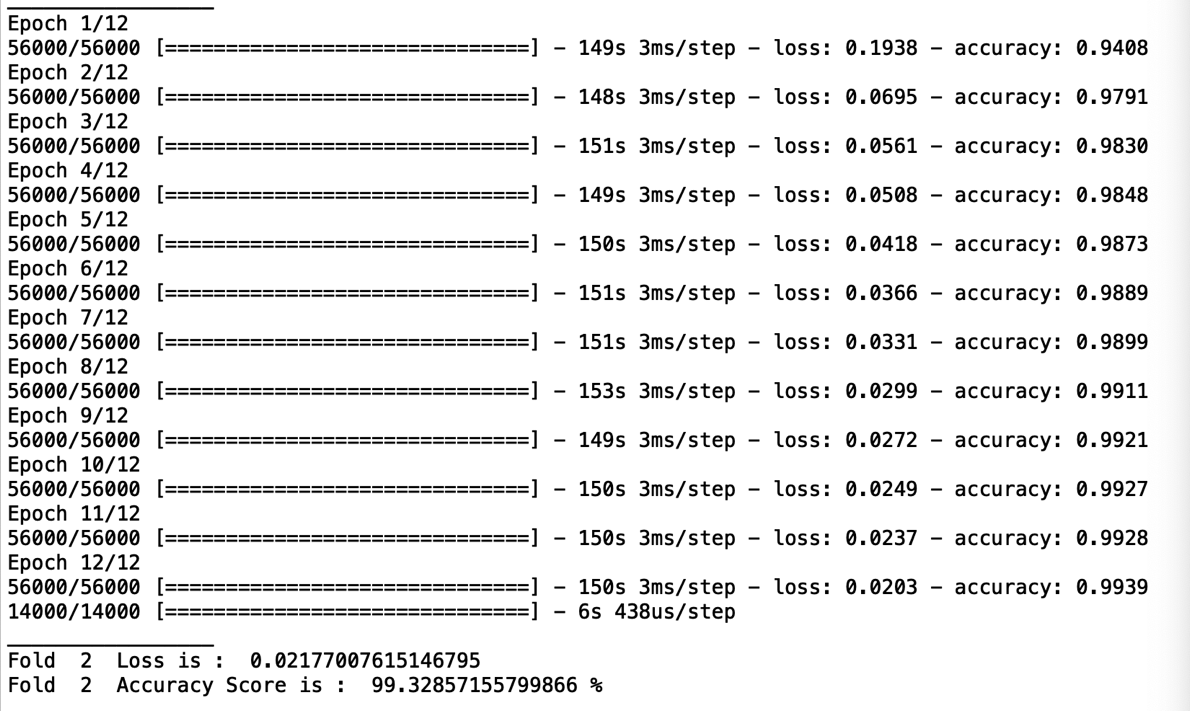
Doğruluk değerleri incelendiğinde, modelin yüksek oranda Bias taşıdığına yönelik herhangi bir kuşkuya sebep olmadığı görülmektedir. Sonuçlar birbirine yeterince yakındır.

K-Fold çapraz doğrulama olmadan gerçekleştirilen önceki eğitimin sonucu olan %99,62 ile arasında yer alan ~%0,24’lük farkın sebebi muhtemelen çapraz doğrulamayı mümkün kılma gayesiyle eksiltilen 18 epok’tur. (Donanım işlem gücü kısıtlayıcı faktör.)

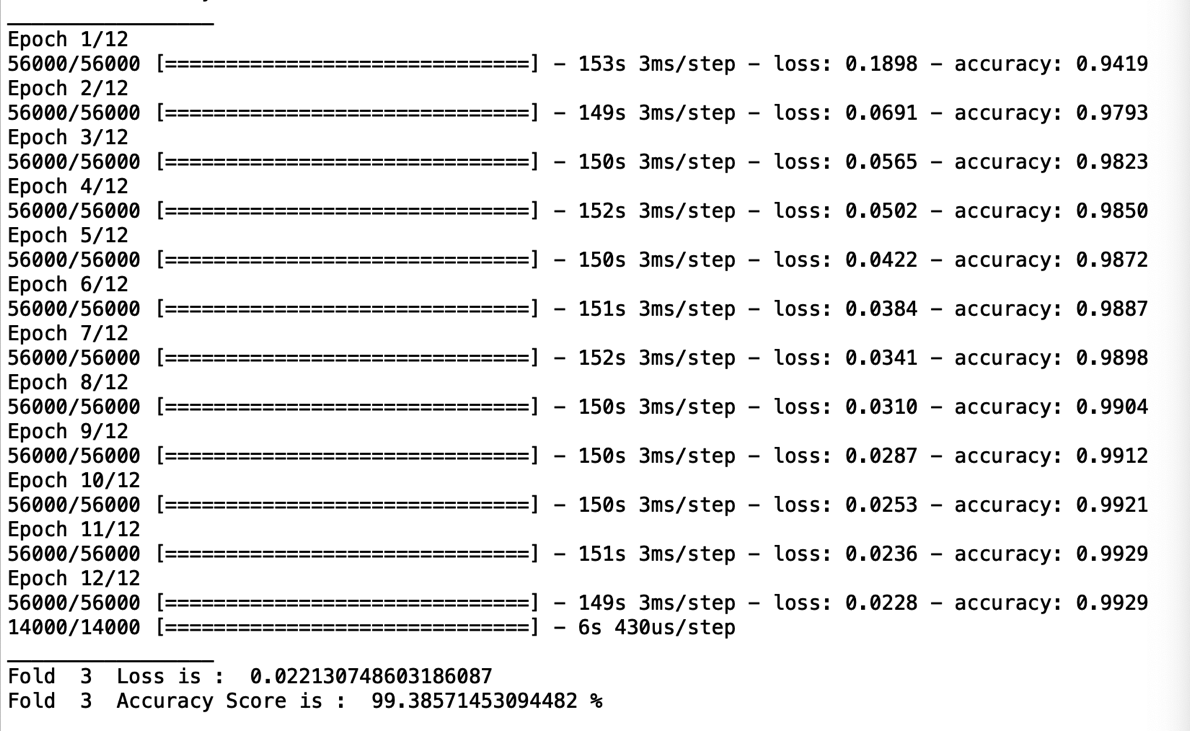
Aşağıda 5 kat çapraz doğrulama kullanılarak gerçekleştirilen eğitimin konsol çıktılarını inceleyebilirsiniz.



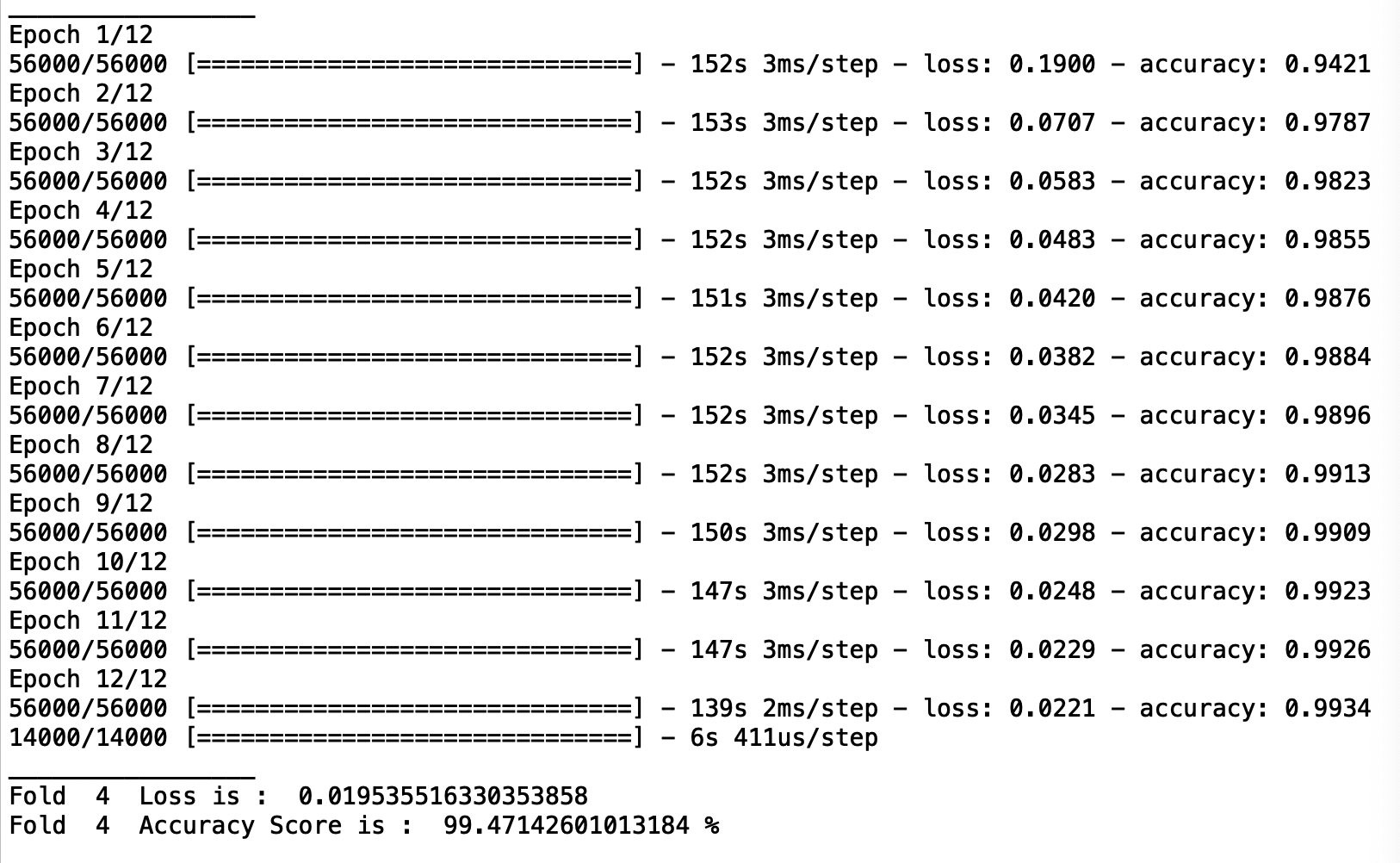
**Görsel 28**: *kfold.py eğitim sürecine ait bir görüntü*

**

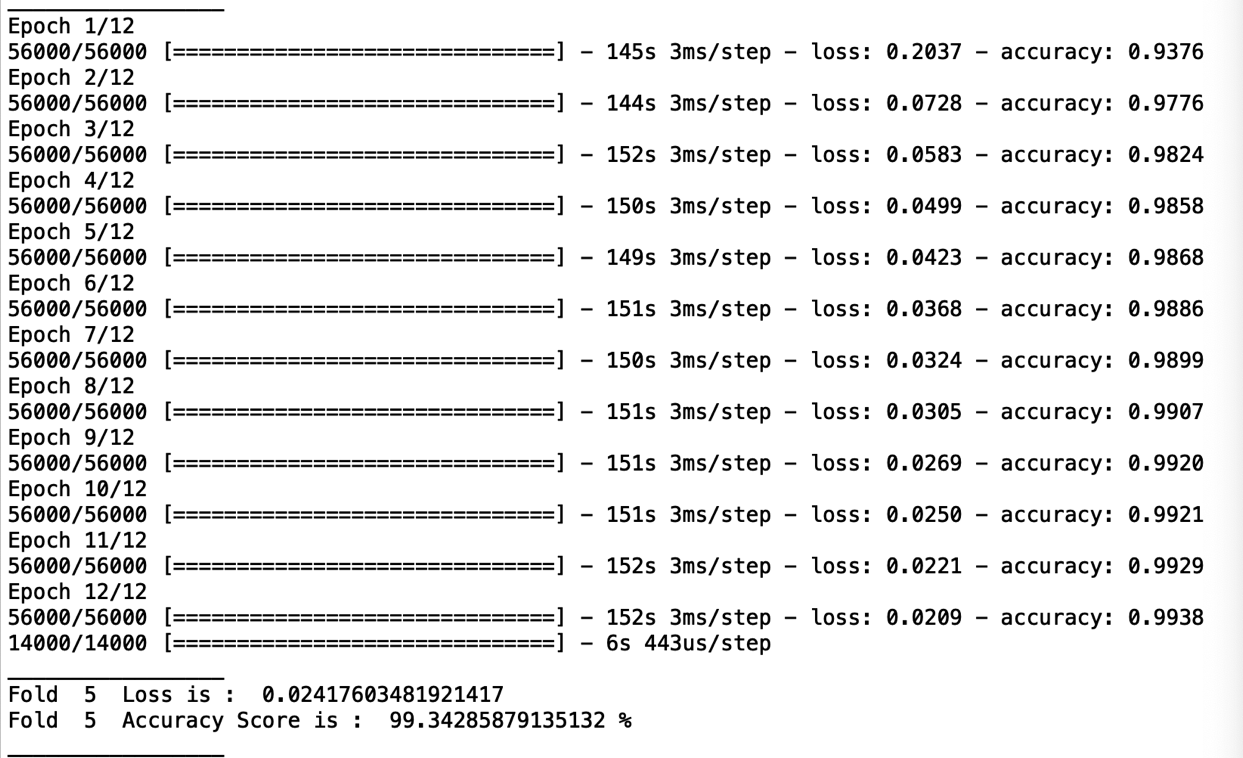
**Görsel 29:** *kfold.py eğitim sürecine ait bir görüntü*

**

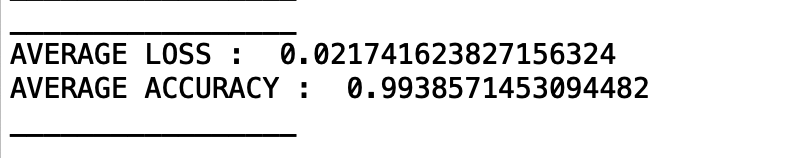
**Görsel 30**: *kfold.py eğitim sürecine ait bir görüntü*

**

**Görsel 31**: *kfold.py eğitim sürecine ait bir görüntü*



**Görsel 32**: *kfold.py eğitim sürecine ait bir görüntü*



**Görsel 33**: *kfold.py eğitim sürecine ait bir görüntü*

**4.4.3 Yorumlar (En başarılı parametre değerleri, nedenleri)**

Batch sayısı açısından en başarılı parametre değerini 32 olarak belirledim. Bunun en temel sebeplerinden biri batch sayısını çok yüksek tutmanın eğitim hızı açısından negatif geri dönüşe sebep olması, çok düşük tutmanın da yine aynı şekilde negatif etkiye yol açmasıydı. 32 ise optimum hızda işlenmesini sağlıyordu. Filtre sayısının daha da arttırılmasının performansı olumlu etkileyebileceğini düşünmekteyim ancak kulandığım donanım buna elverecek kadar güçlü olmadığından 3 adet 32 filtreli, 3 adet de 64 filtreli evrişimsel katman kulandım.

Buna ek olarak, Dropout değeri üzerinde çeşitli oynamalarda da bulundum. 0.2, 0.3 ve 0.4 dropout değerleri denedim ve aralarında kayda değer doğruluk farklarına rastlamadım. Fakat daha düşük seçmenin negatif etkilere sahip olabileceği düşüncesindeyim.

Daha fazla evrişimsel katman kullanmak avantaj sağlayabilir fakat yine donanımla ilgili sebeplerden ötürü katman sayısını fazla sayıda tutamadım. Yoğun katmandaki nöron sayısını daha fazla tutmak, örneğin 256 veya 512 olarak belirlemek doğrulukta herhangi bir pozitif değişime sebep olmadı. Ancak 32 ve 64 gibi daha düşük değerler negatif etkiye sebep oldu. Bu yüzden 128’i uygun bir değer olarak belirledim.

Görüntü işlemeyle ilgili modellerde performansa olumlu etki yaptığı düşünülen ReLU aktivasyon fonksiyonunu kullanmanın avantaj sağlayabileceğini düşündüm. Son olarak epok sayısını önce 10 olarak belirlemiştim. Fakat 10 turun sonunda model hala öğrenmesine devam edebiliyordu bu sebeple önce 20’ye, sonra yine aynı sebeple 30’a çıkardım. Daha kuvvetli bir donanımdan faydalanarak belki 35 veya 40 epok sonucunda doğrulukta değişim olup olmayacağını görmek de faydalı olabilir.

**5. Özdeğerlendirme Raporu (10)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | İstenen Özellik | Puan | | | Var | Açıklama | Tahmini Puan |
| **0** | Kapak Sayfası | 5 |  | İstenen özelliklerde bir kapak sayfası geliştirildi. | | | **5** |
| **1** | Image Classification, Object Detection ve Image Segmentation Karşılaştırması | 5 |  | Bahsedilen kavramlar hakkında bilgi verildi ve aralarındaki farklar açıklandı. | | | **5** |
| **2** | Araştırma (Ön Çalışma) | 5 |  | Belirtilen konular hakkında araştırmalar yapıldı ve videolar izlendi. Öğrenilenler ve ilginç gelen detaylar raporda belirtildi. | | | **5** |
| **3.1** | Derin Öğrenme / Problemin Tanımı | 5 |  | Projede çözüm üretilen derin öğrenme problemine yönelik bilgi verildi ve amaçtan bahsedildi. | | | **5** |
| **3.2** | Derin Öğrenme / Veri Seti Hakkında Bilgi | 5 |  | Projede kullanılan MNIST veri setine yönelik bilgi verildi. | | | **5** |
| **3.3.1** | Model Oluşturma / Kullanılan Ortam, Dil, Kütüphaneler ve Sürümleri | 5 |  | Projede kullanılan programlama dili, programlama ortamı, kullanılan kütüphaneler ve bunların sürümleri hakkında bilgi verildi ve rapora eklendi. | | | **5** |
| **3.3.2** | Model Oluşturma / Derin Öğrenme ile bir tanıma / tespit uygulaması kodu. Modelin adı, katman sayısı, şeması / şekli | 10 |  | Proje dahilinde geliştirilen modele yönelik bilgiler sağlandı. Katman sayısı belirtildi ve katmanların özellikleri hakkında bilgi verildi. Şema, çizge şeklinde ifade edildi. Modeli ifade eden kodların görselleri paylaşıldı. Veri setinin eğitimini ve testini içeren kodlara ait görseller paylaşıldı. | | | **10** |
| **3.3.3** | Model Oluşturma / Hangi çalışma / hazır kod / bağlantıdan faydalandınız? Faydalandığınız kaynaklara göre farklılıklarınız nelerdir? Açıklayınız. | 10 |  | Benzer çalışmalarla olan farklılıklardan bahsedildi. Yararlanılan kaynaklara yönelik bilgi verildi. Kaynak numarası ile işaretlendi ve kaynakçada bu kaynaklara yönelik linkler belirtildi. Başarı oranları hakkında kısa bir tartışmada bulunuldu. | | | **10** |
| **4.4.1** | Eğitim işleminin anlatımı | 5 |  | Eğitim işleminin gerçekleştirildiği iki yöntem hakkında bilgi verildi. Bu yöntemlerin nasıl gerçekleştirildiği ve hangi teknolojilerin kullanıldığı anlatıldı. | | | **5** |
| **4.4.2** | Tablo ve Grafik (parametre optimizasyonu). Başarımı artırmak için hangi yöntemleri kullandığınızı ve artış miktarını belirtiniz. | 15 |  | Projede gerçekleştirilen eğitim sürecinin sonunda ulaşılan Loss ve Accuracy değerlerine yönelik grafikler rapora eklendi. Parametrelerin optimize edilmesi için gerçekleştirilen adımlardan ve yapılanlardan bahsedildi. Parametre optimizasyonu sonucunda başarım oranlarındaki değişimler tartışıldı. | | | **15** |
| **4.4.3** | Yorumlar (En başarılı parametre değerleri, nedenleri vs.) | 5 |  | En son kullanılan parametre değerlerinde neden karar kılındığı konusunda açıklamalar gerçekleştirildi. Başarım üzerindeki etkileri hakkında tartışıldı ve bunların nedenleri konusunda fikir yürütüldü. | | | **5** |
| **5** | Özdeğerlendirme Raporu | 10 |  | Özdeğerlendirme raporu gereken şekilde dolduruldu. | | | **10** |
| **Ek.1** | Kaynakça | 5 |  | Kaynakça düzenlendi ve özellikle 3.3.3. maddede belirtilen ve faydalanılan kaynaklar hakkında burada gerekli referanslar verildi. | | | **5** |
| **Ek.2** | Rapor Düzeni | 10 |  | Rapor açık ve anlaşılır bir düzende olacak şekilde tasarlandı ve kontrol edildi. | | | **10** |
| **Toplam** | | | | | | | **100** |

**Not:** *Raporda bahsedilen projeye ait kod dökümanları, raporun ilişiğinde olduğu* ***klasöre eklenmiştir****.*

**Not 2:** *Kaynakça* ***bir sonraki*** *sayfada belirtilmiştir.*

**6. Kaynakça**

**[1]** What is Image Clasification? <https://desktop.arcgis.com/en/arcmap/latest/extensions/spatial-analyst/image-classification/what-is-image-classification-.htm> Accessed on 17 01 2021

**[2]** Object Detection. <https://en.wikipedia.org/wiki/Object\_detection> Accessed on 17 01 2021

**[3]** Image Segmentation. <https://en.wikipedia.org/wiki/Image\_segmentation> Accessed on 17 01 2021

**[4]** Keras API Documentation. <https://keras.io/api/> Accessed on 17 01 2021

**[5]** Keras CNN Example. <https://www.kaggle.com/tonypoe/keras-cnn-example> Accessed on 17 01 2021

**[6]** Building a Convolutional Neural Network (CNN) in Keras. <https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5> Accessed 17 01 2021

**[7]** Matplotlib Documentation. <https://matplotlib.org/3.3.3/contents.html> Accessed 17 01 2021

**[8]** Intuitively Understanding Convolutions for Deep Learning. <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1> Accessed 17 01 2021

**[9]** Max Pooling. < https://computersciencewiki.org/index.php/Max-pooling\_/\_Pooling> Accessed 17 01 2021

**[10]** Convolutional Neural Networks: An Intro Tutorial. <https://heartbeat.fritz.ai/a-beginners-guide-to-convolutional-neural-networks-cnn-cf26c5ee17ed> Accessed 17 01 2021

**[11]** How to Choose CNN Architecture MNIST. <https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist> Accessed 17 01 2021

**[12]** Digit Recognition – Notebooks. <https://www.kaggle.com/c/digit-recognizer/notebooks> Accessed 18 01 2021