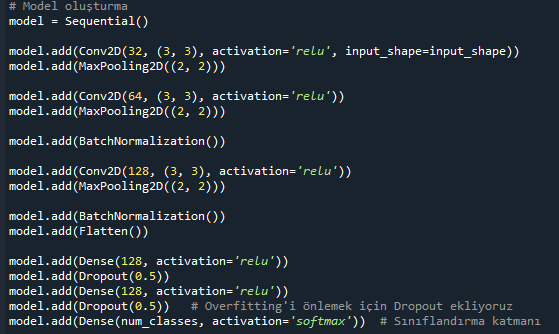
# **BİTİRME PROJESİ VİZE RAPOR**

**CİHAN OZAN / 203908032**

**ImageDataGenerator**, ile görüntü veri setlerini eğittim.  
**validation\_split** parametresini, veri setinin bir bölümünü ( %20’) eğitim ve doğrulama setleri olarak ayırmak için kullandım, ImageDataGenerator parametresidir.

**Model 1 :**

****

**Evrişim Katmanları ve Havuzlama Katmanları:** Modelde üç evrişim katmanı ve üç havuzlama katmanı bulunuyor. Bu, görüntülerdeki özelliklerin hiyerarşik bir şekilde çıkarılmasına ve boyutların azaltılmasına olanak tanır.

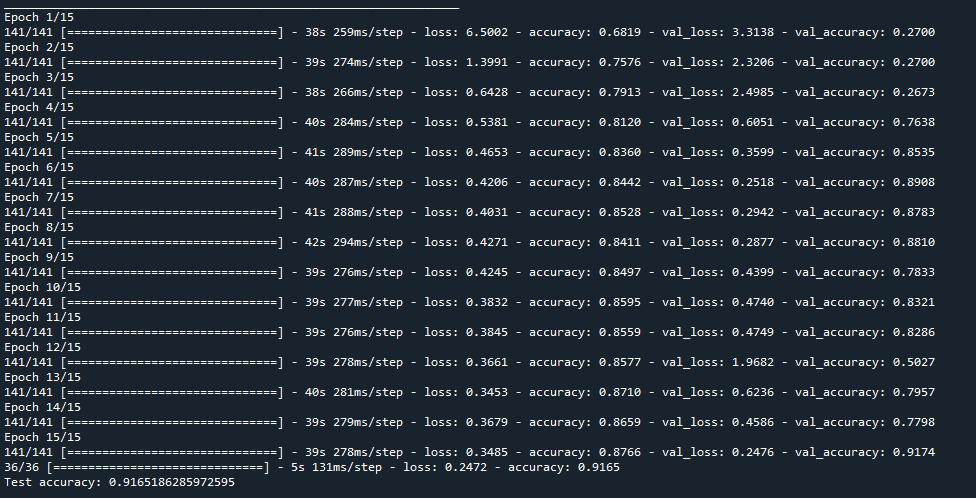
**Normalizasyon:** Batch normalization, her evrişim katmanından sonra kullanılır. Bu, eğitim sürecini hızlandırmaya ve ağın daha istikrarlı öğrenmesini sağlamaya yardımcı olur.

**Yoğun Katmanlar ve Dropout:** Yoğun katmanlar, ağın özellikleri ve özniteliklerini öğrenmesine olanak tanır. Dropout, aşırı uyum (overfitting) riskini azaltmak için kullanılır.

**Çok Katmanlı:** Model oldukça derin, yani katman sayısı fazla. Bu, daha karmaşık özelliklerin öğrenilmesine ve daha yüksek seviyeli görsel özniteliklerin temsil edilmesine olanak tanır.

İki tane Conv2D ve MaxPooling2D katmanları ile başlar, ardından bir BatchNormalization bir sonra yeniden Conv2D ve MaxPooling2D katmanları sonra ise Flatten ve iki Dense Dropout katman eklenir.

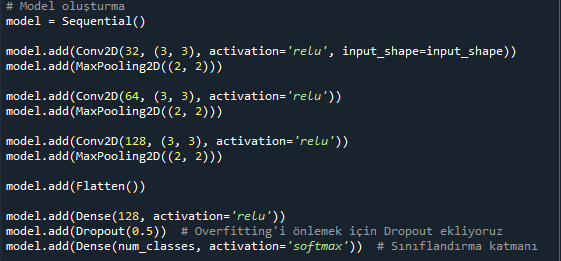
**Model 1 Çıktı:**



Test Doğruluğu: %91

Eğitim süresi: 15 epoch

**Model 2:** Küçük veri setleri için kullanılabilir

****

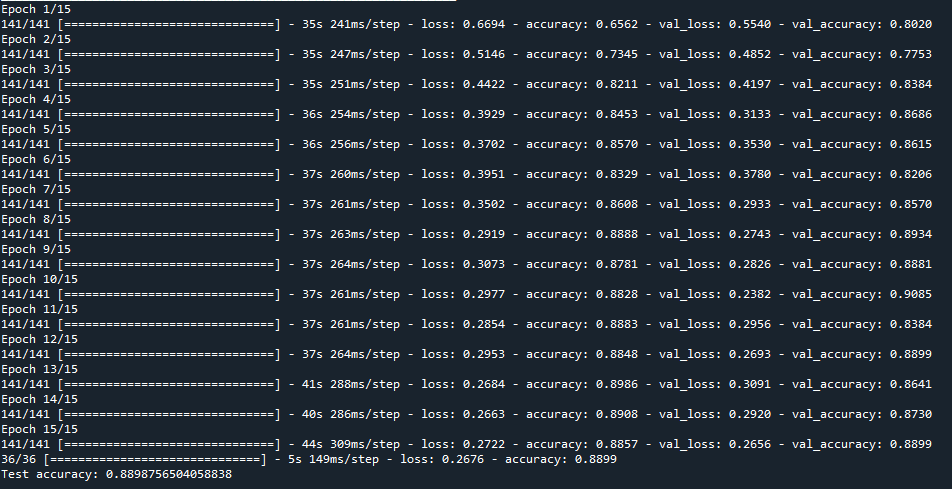
**Daha Az Evrişim ve Havuzlama Katmanları:** Modelde sadece iki evrişim katmanı ve havuzlama katmanı bulunuyor. Bu, daha az sayıda parametre ve daha basit bir yapıya işaret eder.

**Normalizasyon Yok:** Batch normalization katmanları bu modele eklenmemiş. Bu, eğitim sürecinin biraz daha kararsız olabileceği anlamına gelir, ancak daha hızlı eğitim sağlayabilir.

**Daha Az Karmaşıklık:** Modelin daha az katmanı olduğu için daha az karmaşıktır. Daha küçük veri setleri veya daha az hesleme gücü gerektiren senaryolar için daha uygundur.

Üç tane Conv2D ve MaxPooling2D katmanları ile başlar, ardından bir Flatten, bir Dense ve bir Dropout katman sonrada sınıflandırma katmanı eklenir.

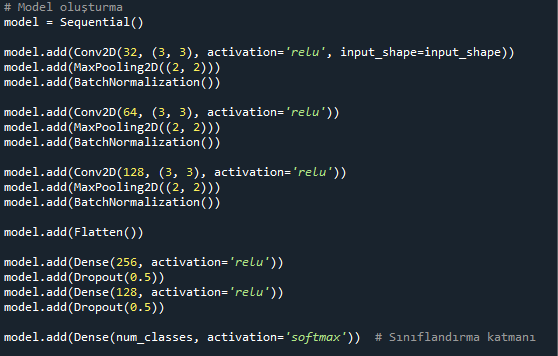
**Model 2 Çıktı:**



Test Doğruluğu: %88

Eğitim süresi: 15 epoch

**Model 3:**

****

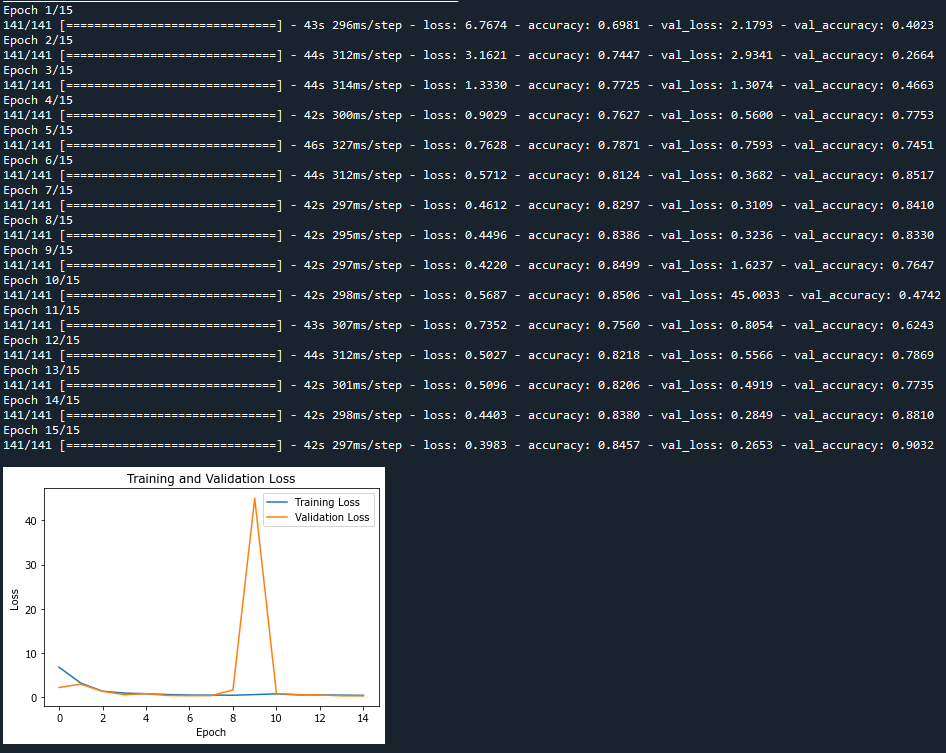
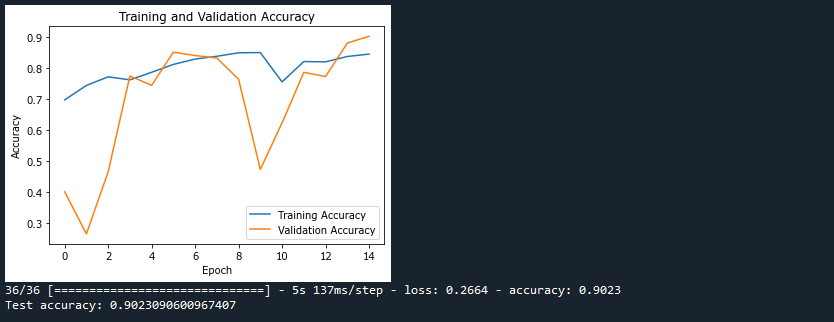
**Normalizasyon Her Katmandan Sonra:** Model, her evrişim ve havuzlama katmanından sonra batch normalization kullanır. Bu, eğitim sırasında daha istikrarlı gradyanlar sağlar ve daha hızlı öğrenmeyi teşvik eder.

**Daha Fazla Evrişim ve Havuzlama Katmanları:** Model, daha fazla evrişim ve havuzlama katmanıyla daha derin bir yapıya sahiptir. Bu, daha fazla özellik çıkarımı sağlar ve daha karmaşık görsel yapıları modellemeye olanak tanır.

**Daha Büyük Yoğun Katmanlar:** Modelin yoğun katmanlarında daha fazla nöron bulunur. Bu, daha yüksek seviyeli özelliklerin temsil edilmesine ve daha karmaşık sınıflandırma kararlarının alınmasına yardımcı olabilir.

Üç kez Conv2D, MaxPooling2D, BatchNormalization katmanları ekledim, ardından bir Flatten, bir Dense ve bir Dropouttan arka arkaya 2 kere ekledim, sonrada sınıflandırma katmanı eklendim.

**Model 3 Çıktı:**

Test Doğruluğu: %90

Eğitim süresi: 15 epoch

**Kodun Açıklaması:**

1. **Veri Artırma ve Ön İşleme:** ImageDataGenerator sınıfı kullanılarak veri artırma ve ön işleme yapılmaktadır. Bu, veri kümesindeki görüntüler üzerinde rastgele dönüşümler yaparak eğitim verisi çeşitliliğini artırır ve modelin genelleme yeteneğini geliştirir.
2. **Eğitim ve Doğrulama Verilerinin Yüklenmesi:** flow\_from\_directory yöntemi kullanılarak eğitim ve doğrulama verileri yüklenmektedir. Bu yöntem, veri kümesinin belirtilen dizin yapısını kullanarak verileri otomatik olarak yükler.
3. **Modelin Oluşturulması:** Sequential modeli kullanılarak bir CNN modeli oluşturulmaktadır. Model, Convolutional, MaxPooling, BatchNormalization ve Dropout katmanları içermektedir.
4. **Modelin Eğitimi:** Oluşturulan model, fit yöntemi kullanılarak eğitilmektedir. Bu, veri artırma ve ön işleme yapılırken, eğitim ve doğrulama verileri üzerinde belirtilen sayıda epoch boyunca eğitilmesini sağlar.
5. **Eğitim ve Doğrulama Kayıpları ve Doğrulukları:** Eğitim sırasında kaydedilen eğitim ve doğrulama kayıp ve doğruluk değerlerini görselleştirmek için matplotlib kullanılmaktadır.
6. **Modelin Değerlendirilmesi:** Eğitim sonrası, modelin test verisi üzerindeki doğruluğu değerlendirilmektedir.
7. **Modelin Kaydedilmesi:** Eğitim sonrası, model .h5 dosya biçiminde kaydedilmektedir.

**Kod Kısmı:**

**Model 1 Kodu:** Diğer çıktılarımda kod aynı modeller farklı

import os

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization

# Veri artırma ve ön işleme

train\_datagen = ImageDataGenerator(

rescale=1./255, # Pixel değerlerini [0, 1] aralığına ölçeklendirme

rotation\_range=20, # Rastgele rotasyon

width\_shift\_range=0.2, # Yatayda rastgele kaydırma

height\_shift\_range=0.2, # Dikeyde rastgele kaydırma

shear\_range=0.2, # Şeğirme dönüşümü

zoom\_range=0.2, # Rastgele yakınlaştırma

horizontal\_flip=True, # Yatayda rastgele çevirme

fill\_mode='nearest', # Doldurma stratejisi

validation\_split=0.2 # Eğitim ve doğrulama verisi ayırımı

)

# Veri seti yolu

dataset\_dir = os.path.expanduser('C:/Users/Cihan/Desktop/archive/data')

# Eğitim verilerinin yüklenmesi

train\_generator = train\_datagen.flow\_from\_directory(

dataset\_dir,

target\_size=(128, 128), # Görüntülerin yeniden boyutlandırılacağı hedef boyut

batch\_size=32, # Batch boyutu

class\_mode='categorical', # Sınıflandırma tipi

subset='training' # Eğitim verisi

)

# Doğrulama verilerinin yüklenmesi

validation\_generator = train\_datagen.flow\_from\_directory(

dataset\_dir,

target\_size=(128, 128),

batch\_size=32,

class\_mode='categorical',

subset='validation' # Doğrulama verisi

)

# Hiperparametreler

input\_shape = (128, 128, 3) # Girdi görüntü boyutu (yükseklik, genişlik, kanal sayısı)

num\_classes = 4 # Sınıf sayısı

# Model oluşturma

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=input\_shape))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(BatchNormalization())

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(BatchNormalization())

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5)) # Overfitting'i önlemek için Dropout ekliyoruz

model.add(Dense(num\_classes, activation='softmax')) # Sınıflandırma katmanı

# Modelin derlenmesi

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Modelin özeti

model.summary()

# Modelin eğitimi

history = model.fit(train\_generator, epochs=15, validation\_data=validation\_generator)

# Modelin değerlendirilmesi

test\_loss, test\_acc = model.evaluate(validation\_generator)

print(f'Test accuracy: {test\_acc}')

# Modelin kaydedilmesi

model.save('satellite\_classification\_model.h5')

Kodum aynı sadece model oluşturma kısmında değişiklik var.