SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

DERİN ÖĞRENME VE EVRİŞİMLİ SİNİR AĞLARI
PROJE ÖDEVİ

Ad Soyad: Ayşe Esra AŞCI

Numara: G201210036

Şube: 1-A

2023-2024 BAHAR DÖNEMİ

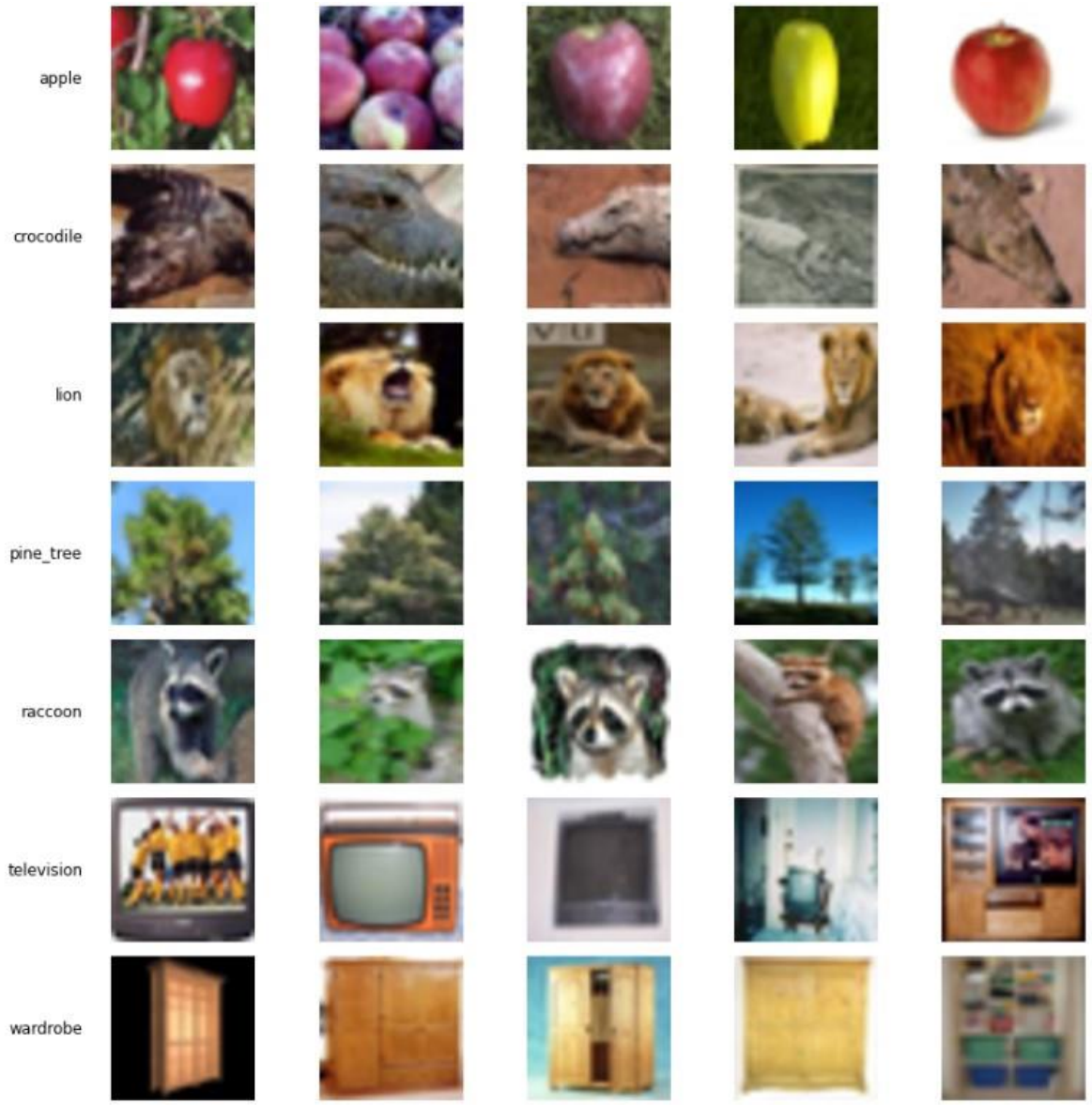
Bu projede, CIFAR-100 veri setinden öğrenci numarasına göre belirlenen 7 sınıf için çok-sınıflı sınıflandırıcı ağı eğitilmesi ve test edilmesi amaçlanmaktadır. CIFAR-100 veri seti, 100 farklı sınıfa ait renkli (RGB) görüntülerden oluşan bir veri setidir. Her sınıf, 32x32 piksel boyutundaki 600 görüntüden oluşmaktadır.

Eğitimde kullanacağım 7 sınıf öğrenci numarama göre

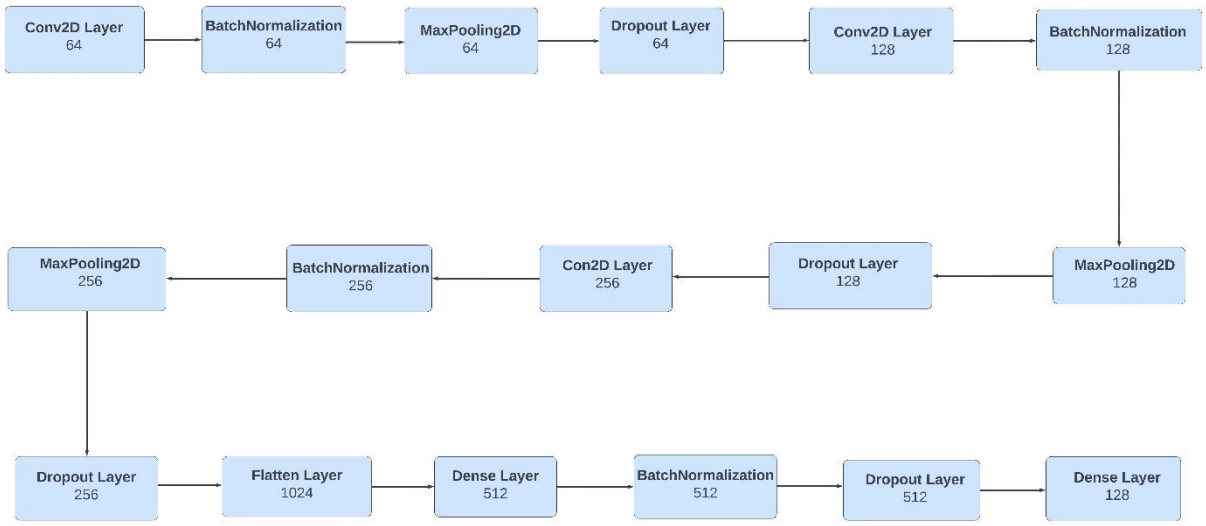
0=apple 27=crocodile 43=lion 59=pine_tree 66=raccoon 87=television 94=wardrobe

bu şekildedir.

Eğitim öncesi, her bir sınıfa ait örnek görüntüler seçilerek görsel olarak incelenmiştir. Aşağıda, her sınıfa ait örnek görüntüler sunulmaktadır. Bu örnek görüntüler, her bir sınıfın temsilcisi olarak seçilmiştir. Eğitim sürecinde, bu sınıflara ait görüntüler kullanılarak modelin eğitimi gerçekleştirilmiştir.



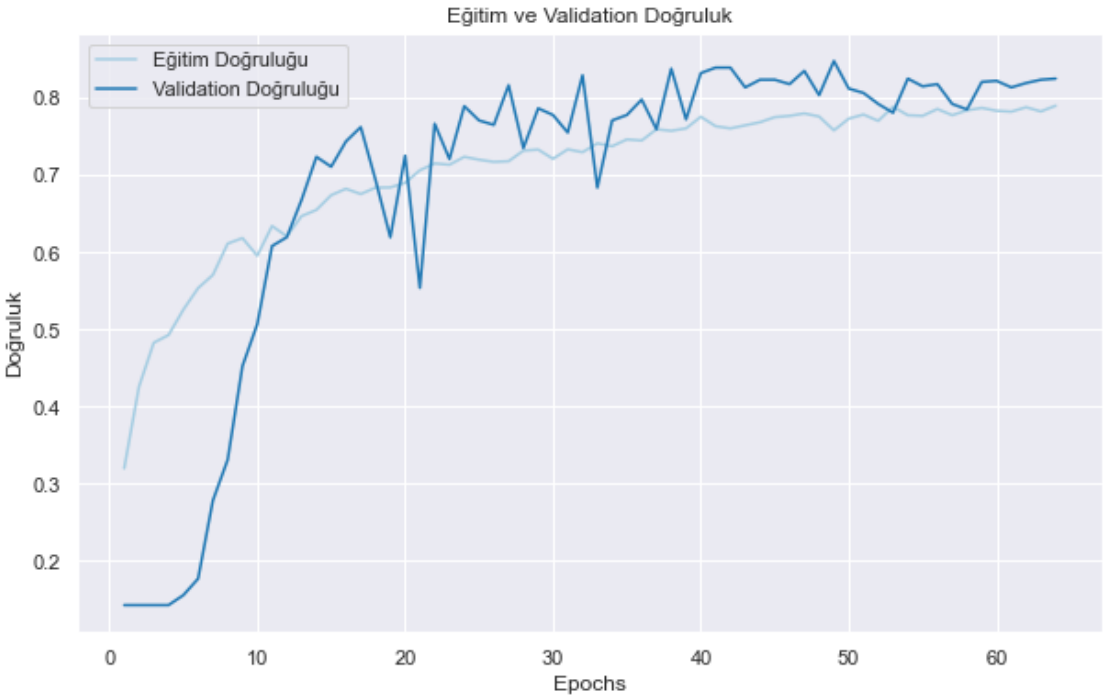
BLOK ŞEMASI



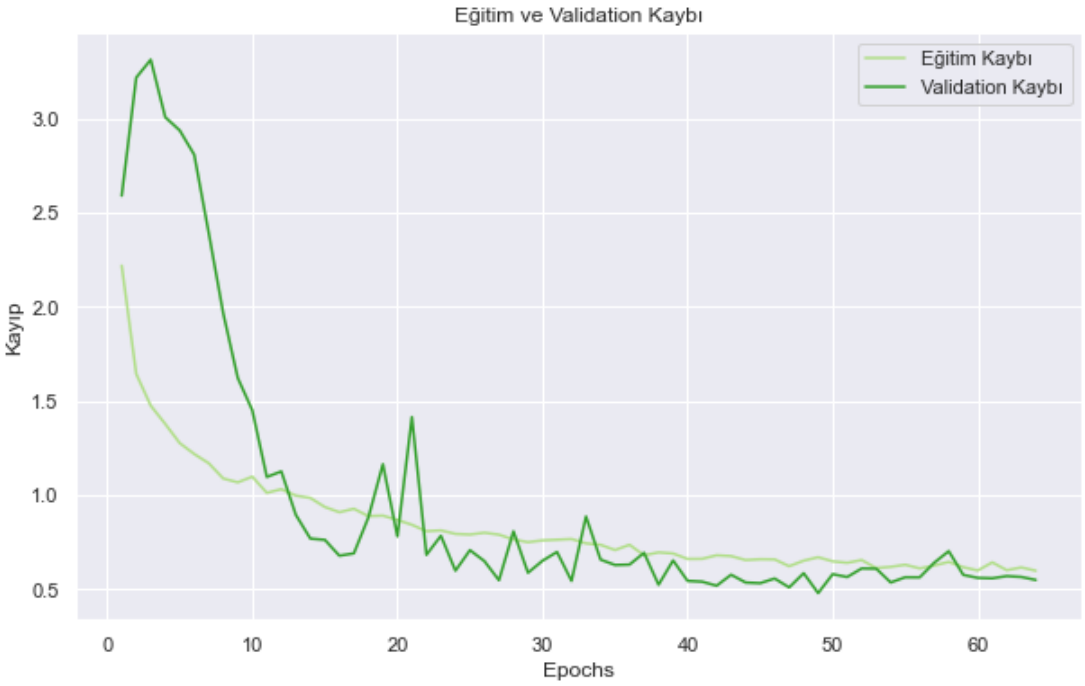
Yukarıda modele ait blok şeması gösterilmiştir.

Bu model, 3 adet konvolüsyon katmanı içermektedir. Her konvolüsyon katmanı, 3x3 boyutunda filtreler ve Relu aktivasyon fonksiyonu kullanmaktadır. İlk katman 64 filtreye sahipken, sonraki iki katman sırasıyla 128 ve 256 filtreye sahiptir. Konvolüsyon katmanlarının ardından modelin karmaşıklığını azaltmaya yardımcı olan üç adet MaxPooling2D katmanı bulunmaktadır. İlk katman 64 filtreye sahipken, sonraki iki katman sırasıyla 128 ve 256 filtreye sahiptir. Dropout katmanları, aşırı uyumu önlemek ve modelin genelleme yeteneğini artırmak için rastgele nöronları devre dışı bırakır. Sonuç olarak, ağ modeli 3 adet konvolüsyon katmanı, 3 adet MaxPooling katmanı, 1 adet Flatten katmanı, 2 adet Dense katmanı, 4 adet BatchNormalization katmanı ve 4 adet Dropout katmanı içeren bir sinir ağıdır. Bu model, verilen giriş verilerini işleyerek, 100 sınıfın olasılık dağılımlarına dönüştürmeyi amaçlamaktadır.

DOĞRULUK (ACCURACY) VE KAYIP (LOSS) GRAFİKLERİ

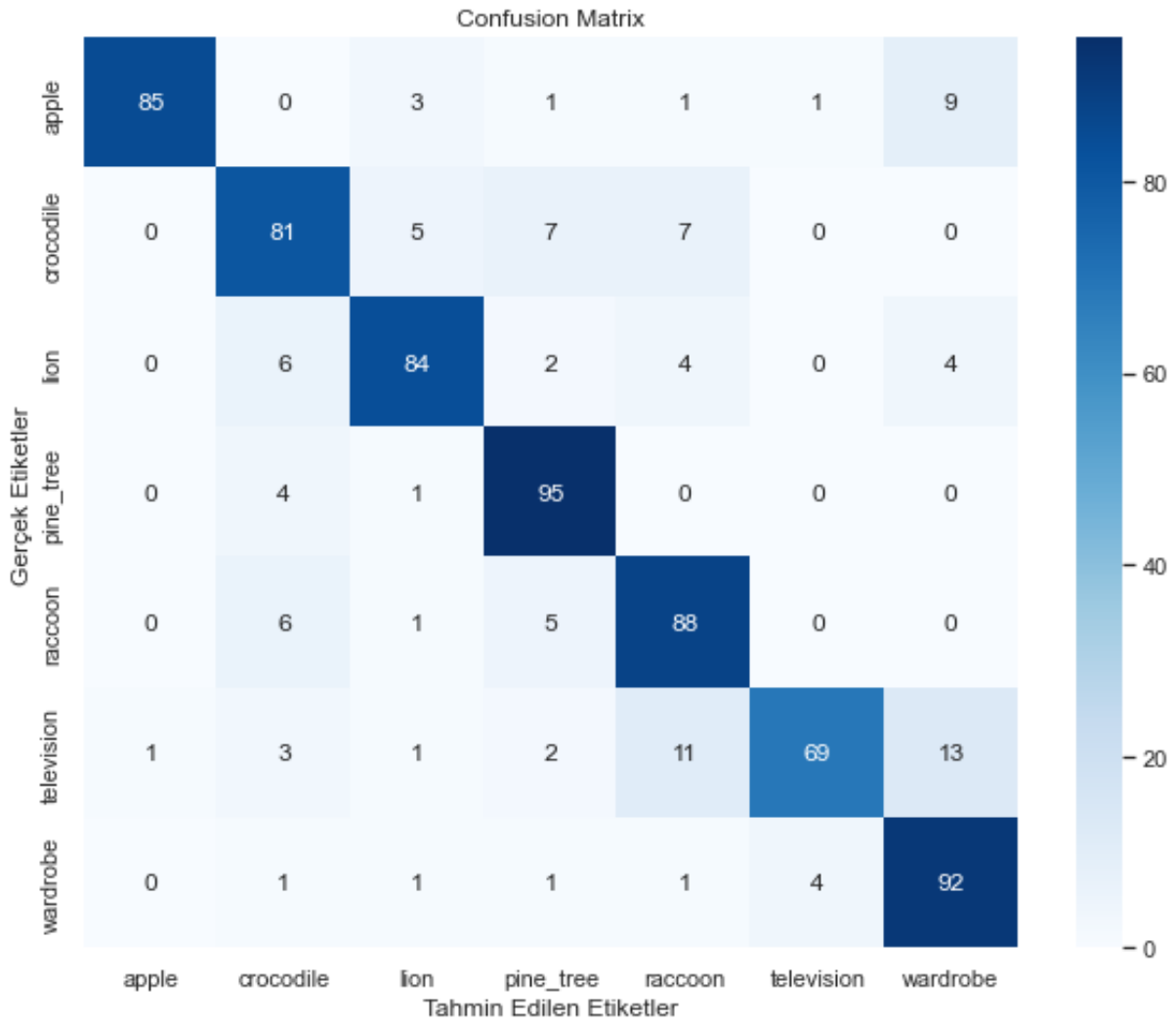


Grafik incelendiğinde, modelin eğitim doğruluğunun başlangıçta hızla artarak kısa sürede %60 seviyelerine ulaştığı ve daha sonra daha yavaş bir artışla %70-%80 aralığında sabitlendiği görülmektedir. Validation doğruluğu ise başlangıçta artış göstermiş ancak eğitim doğruluğuna kıyasla daha dalgalı bir seyir izlemiştir ve %80 civarında daha stabil hale gelmeye başlamıştır. Bu durum, modelin eğitim verisi üzerinde yüksek performans gösterdiğini ancak validation verisi üzerindeki performansının daha tutarsız olduğunu göstermektedir.

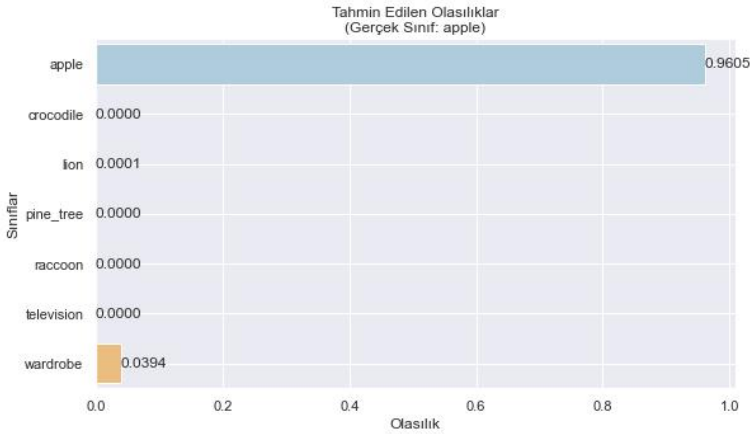


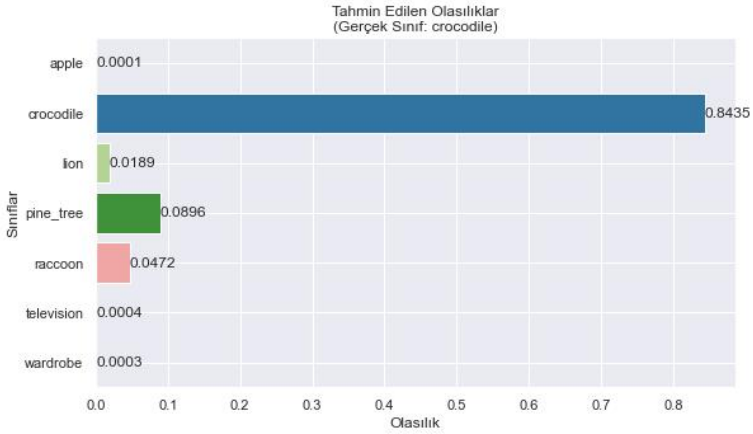
Grafik, modelin iyi öğrenildiğini ve veri verileri üzerinde iyi performans gösterdiğini göstermektedir. Aralıklarındaki yüksek kayıplar, giderek azalarak 1,5 ve 2'ye kadar düşmektedir. Validation kaybı ise başlangıçta eğitim kaybından daha yüksek bir değerden başlayarak daha sonra eğitim kaybına yaklaşmaktadır. Bu, modelin daha önce görmediği verilerde iyi performans gösterdiğini göstermektedir.

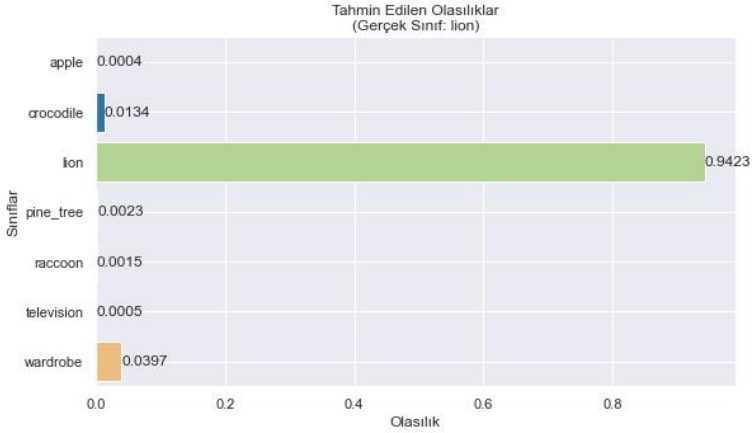
KARMAŞIKLIK(CONFUSİON) MATRİSİ

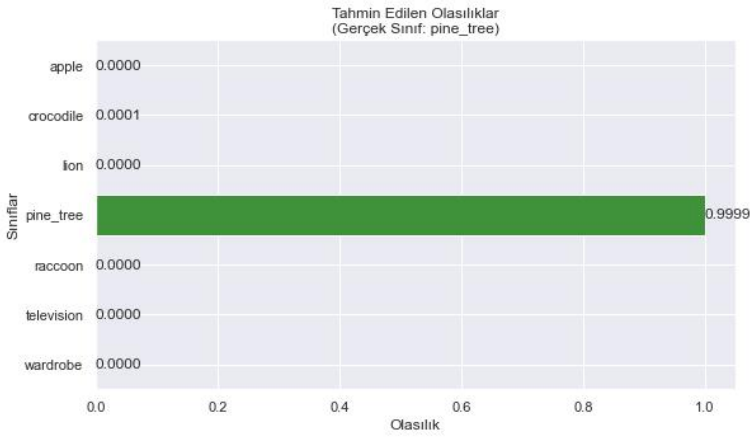


ÖRNEK GÖRÜNTÜLER

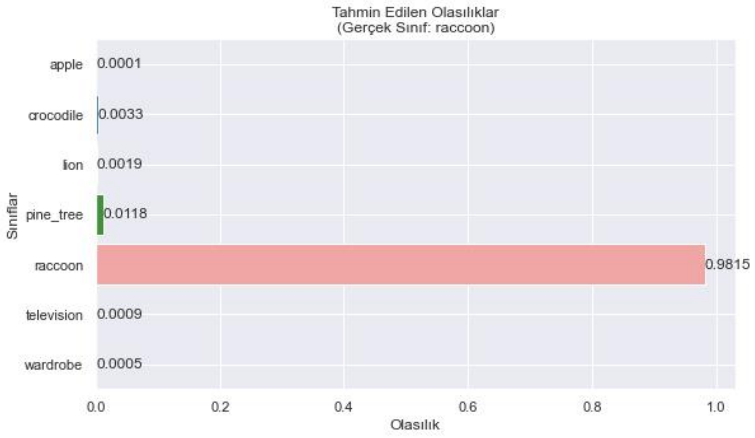




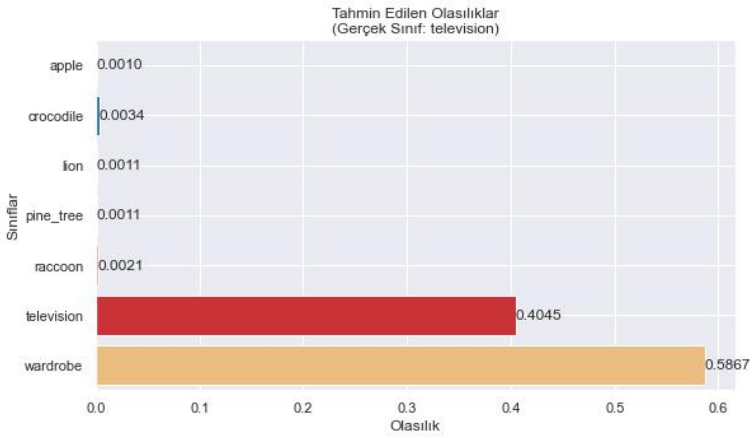



Tahmin Edilen Olasılıklar
(Gerçek Sınıf: pine_tree)
apple 0.0000
crocodile 0.0001
lion 0.0000
pine_tree 0.9999
raccoon 0.0000
television 0.0000
wardrobe 0.0000
Sınıflar
0.0 0.2 0.4 0.6 0.8 1.0
Olasılık

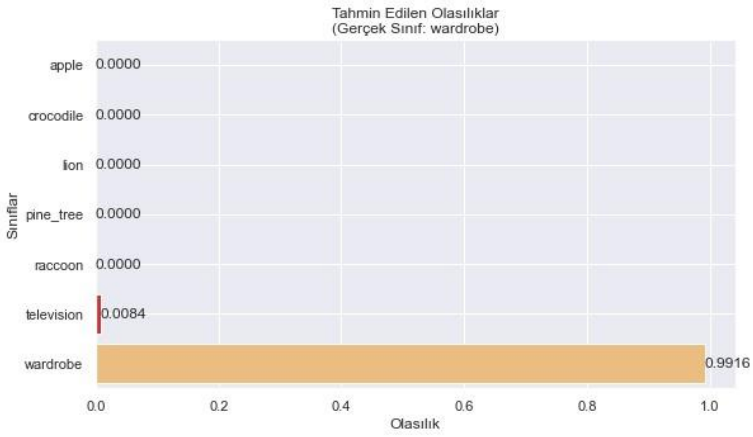

Tahmin Edilen Sınıf: pine_tree
Gerçek Sınıf: pine_tree


Tahmin Edilen Olasılıklar
(Gerçek Sınıf: raccoon)
apple 0.0001
crocodile 0.0033
lion 0.0019
pine_tree 0.0118
raccoon 0.9815
television 0.0009
wardrobe 0.0005
Sınıflar
0.0 0.2 0.4 0.6 0.8 1.0
Olasılık


Tahmin Edilen Sınıf: raccoon
Gerçek Sınıf: raccoon


Tahmin Edilen Olasılıklar
(Gerçek Sınıf: television)
apple 0.0010
crocodile 0.0034
lion 0.0011
pine_tree 0.0011
raccoon 0.0021
television 0.4045
wardrobe 0.5867
Sınıflar
0.0 0.1 0.2 0.3 0.4 0.5 0.6
Olasılık


Tahmin Edilen Sınıf: wardrobe
Gerçek Sınıf: television



KOD

```
import numpy as np

import matplotlib.pyplot as plt

from keras.datasets import cifar100

from keras.utils import to_categorical

from keras.models import Sequential

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

from keras.preprocessing.image import ImageDataGenerator

from keras.callbacks import EarlyStopping, ReduceLROnPlateau

from sklearn.metrics import confusion_matrix

import seaborn as sns


# Stil ve renk paleti ayarları

sns.set(style='darkgrid')

custom_palette = sns.color_palette("Paired", 10)


# Sınıf isimleri

sinif_isimleri = [

    'apple', 'aquarium_fish', 'baby', 'bear', 'beaver', 'bed', 'bee', 'beetle', 'bicycle', 'bottle', 'bowl', 'boy', 'bridge', 'bus', 'butterfly',
'camel',
```

```
    'can', 'castle', 'caterpillar', 'cattle', 'chair', 'chimpanzee', 'clock', 'cloud', 'cockroach', 'couch', 'crab', 'crocodile', 'cup', 'dinosaur',

    'dolphin', 'elephant', 'flatfish', 'forest', 'fox', 'girl', 'hamster', 'house', 'kangaroo', 'keyboard', 'lamp', 'lawn_mower', 'leopard', 'lion',

    'lizard', 'lobster', 'man', 'maple_tree', 'motorcycle', 'mountain', 'mouse', 'mushroom', 'oak_tree', 'orange', 'orchid', 'otter', 'palm_tree', 'pear',

    'pickup_truck', 'pine_tree', 'plain', 'plate', 'poppy', 'porcupine', 'possum', 'rabbit', 'raccoon', 'ray', 'road', 'rocket', 'rose', 'sea', 'seal', 'shark',

    'shrew', 'skunk', 'skyscraper', 'snail', 'snake', 'spider', 'squirrel', 'streetcar', 'sunflower', 'sweet_pepper', 'table', 'tank', 'telephone', 'television', 'tiger',

    'tractor', 'train', 'trout', 'tulip', 'turtle', 'wardrobe', 'whale', 'willow_tree', 'wolf', 'woman', 'worm'

]



# Veri kümesini yükle

(train_images, train_labels), (test_images, test_labels) = cifar100.load_data(label_mode='fine')



# Seçilen sınıflar

selected_classes =  [0, 27, 43, 59, 66, 87, 94]



# Seçilen sınıflara ait veri örneklerini filtrele

train_indices = [i for i, label in enumerate(train_labels) if label[0] in selected_classes]

test_indices = [i for i, label in enumerate(test_labels) if label[0] in selected_classes]



train_images_selected = train_images[train_indices]

train_labels_selected = train_labels[train_indices]

test_images_selected = test_images[test_indices]

test_labels_selected = test_labels[test_indices]



# Verileri normalize et

train_images_selected = train_images_selected.astype('float32') / 255

test_images_selected = test_images_selected.astype('float32') / 255
```

```
# Etiketleri kategorik hale getir

train_labels_selected = to_categorical([selected_classes.index(label) for label in train_labels_selected.flatten()], num_classes=len(selected_classes))

test_labels_selected = to_categorical([selected_classes.index(label) for label in test_labels_selected.flatten()], num_classes=len(selected_classes))



# Veri artırma (data augmentation) işlemi

datagen = ImageDataGenerator(

    rotation_range=20,

    width_shift_range=0.2,

    height_shift_range=0.2,

    horizontal_flip=True,

    shear_range=0.2,

    zoom_range=0.2

)

datagen.fit(train_images_selected)



# Model oluştur

model = Sequential()



model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(32, 32, 3)))

model.add(BatchNormalization())

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

model.add(Dropout(0.3))

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

model.add(BatchNormalization())

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

model.add(Dropout(0.4))



model.add(Conv2D(256, (3, 3), activation='relu'))
```

```
model.add(BatchNormalization())

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

model.add(Dropout(0.4))


model.add(Flatten())


model.add(Dense(512, activation='relu'))

model.add(BatchNormalization())

model.add(Dropout(0.5))


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

model.add(BatchNormalization())

model.add(Dropout(0.5))


model.add(Dense(len(selected_classes), activation='softmax'))


model.compile(optimizer='adam',

              loss='categorical_crossentropy',

              metrics=['accuracy'])


# Öğrenme oranını azaltma callback'i

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=0.00001)


# Erken durdurma callback'i

early_stopping = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)


# Modeli eğit

history = model.fit(datagen.flow(train_images_selected, train_labels_selected, batch_size=64),

                    epochs=100,
```

```
                validation_data=(test_images_selected, test_labels_selected),
                callbacks=[early_stopping, reduce_lr])

model.summary()

# Doğruluk ve kayıp grafiğini çizdir
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
# Doğruluk grafiği çizimi
plt.figure(figsize=(10, 6))
sns.lineplot(x=epochs, y=acc, label='Eğitim Doğruluğu', color=custom_palette[0])
sns.lineplot(x=epochs, y=val_acc, label='Validation Doğruluğu', color=custom_palette[1])
plt.title('Eğitim ve Validation Doğruluk')
plt.xlabel('Epochs')
plt.ylabel('Doğruluk')
plt.legend()
plt.grid(True)
plt.show()
# Kayıp grafiği çizimi
plt.figure(figsize=(10, 6))
sns.lineplot(x=epochs, y=loss, label='Eğitim Kaybı', color=custom_palette[2])
sns.lineplot(x=epochs, y=val_loss, label='Validation Kaybı', color=custom_palette[3])
plt.title('Eğitim ve Validation Kaybı')
plt.xlabel('Epochs')
plt.ylabel('Kayıp')
plt.legend()
```

```
plt.grid(True)

plt.show()

# Tüm test veri setiyle modeli test et ve tahmin edilen etiketleri elde et

predicted_labels = np.argmax(model.predict(test_images_selected), axis=1)

true_labels = np.argmax(test_labels_selected, axis=1)


# Confusion matrix'i oluştur ve göster

conf_matrix = confusion_matrix(true_labels, predicted_labels)

# Sınıf isimleri

filtered_sinif_isimleri = [sinif_isimleri[selected_classes[i]] for i in range(len(selected_classes))]


plt.figure(figsize=(10, 8))

sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',

            xticklabels=filtered_sinif_isimleri, yticklabels=filtered_sinif_isimleri)

plt.xlabel('Tahmin Edilen Etiketler')

plt.ylabel('Gerçek Etiketler')

plt.title('Confusion Matrix')

plt.show()


# Örnek görüntüler ve tahminler

num_classes = len(selected_classes)

num_examples = 1  # Tek bir örnek göstermek için


for sinif_indeks in range(num_classes):

    class_image_indices = np.where(true_labels == sinif_indeks)[0]

    if len(class_image_indices) == 0:

        continue

    random_index = np.random.choice(class_image_indices)

    sample_image = test_images_selected[random_index]
```

```
output_vector = model.predict(sample_image.reshape(1, 32, 32, 3))

predicted_class = np.argmax(output_vector)

true_class = sinif_indeks

predicted_class_name = sinif_isimleri[selected_classes[predicted_class]]

true_class_name = sinif_isimleri[selected_classes[true_class]]


# Çubuk grafik

plt.figure(figsize=(15, 5))

# Tahmin edilen olasılıkları yatay çubuk grafik olarak çiz

plt.subplot(1, 2, 1)

sns.barplot(y=np.arange(num_classes), x=output_vector[0], palette=custom_palette, orient="h")

plt.yticks(range(num_classes), [sinif_isimleri[i] for i in selected_classes])

plt.ylabel('Sınıflar')

plt.xlabel('Olasılık')

plt.title(f'Tahmin Edilen Olasılıklar\n(Gerçek Sınıf: {true_class_name})')

plt.grid(True)


# Tahmin edilen olasılıkları çubukların üzerine yaz

for i, prob in enumerate(output_vector[0]):

    plt.text(prob, i, f'{prob:.4f}', ha='left', va='center')


# Görüntüyü ve tahmin edilen sınıfı göster

plt.subplot(1, 2, 2)

plt.imshow(sample_image, interpolation='lanczos')

plt.axis('off')

plt.title(f"Tahmin Edilen Sınıf: {predicted_class_name}\nGerçek Sınıf: {true_class_name}")


plt.tight_layout()

plt.show()
```