SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ
FAKÜLTESİ

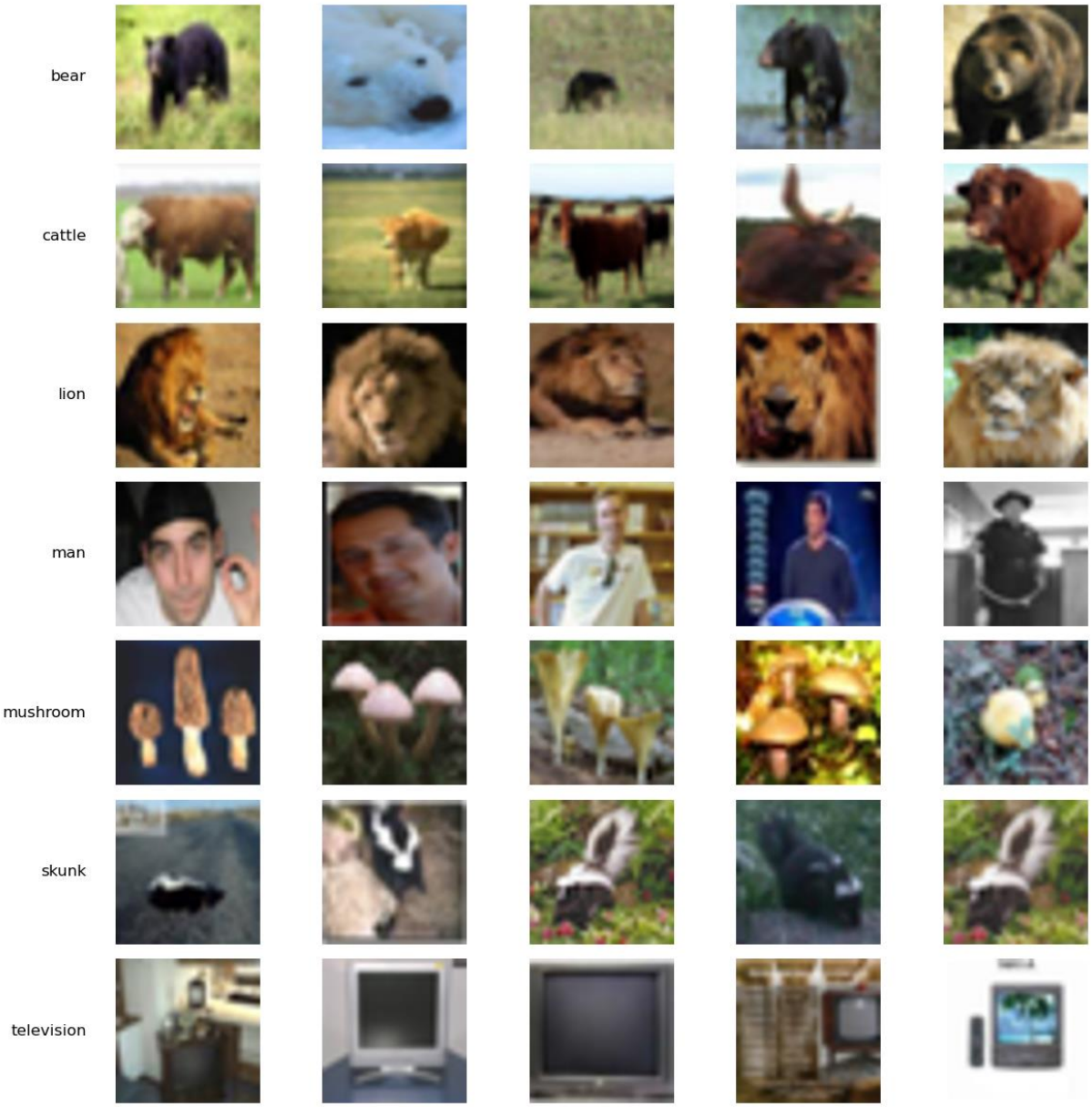
DERİN ÖĞRENME
VE
EVRİŞİMLİ SİNİR AĞLARI

Ad Soyad:Seda Nur EREN
Numara: B201210030
Şube: 1 –A
E-Mail: seda.eren@ogr.sakarya.edu.tr

2023-2024 Bahar Dönemi

1)

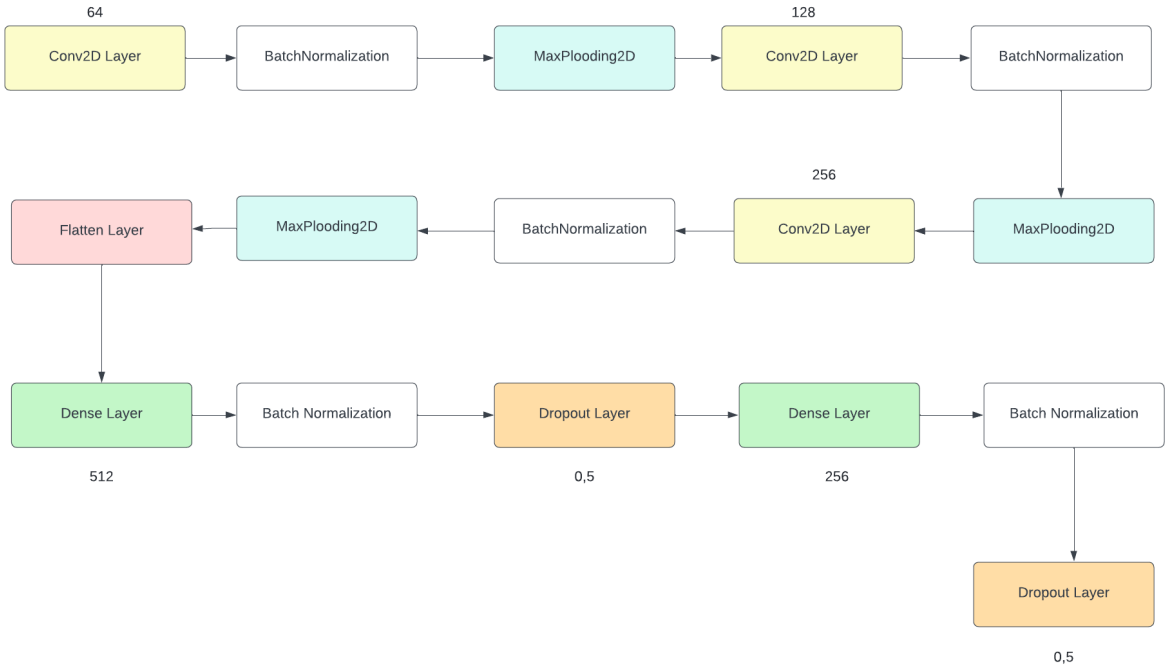
Benim veri setinde üzerinden çalıştığım sınıflar Bear,Cattle,Lion,Man,Mushroom,Skunk,Television şeklindeydi. Bu grafiklerden örnek görüntüleri show_class_images fonksiyonu aracılığıyla seçip gösterdim.



2)

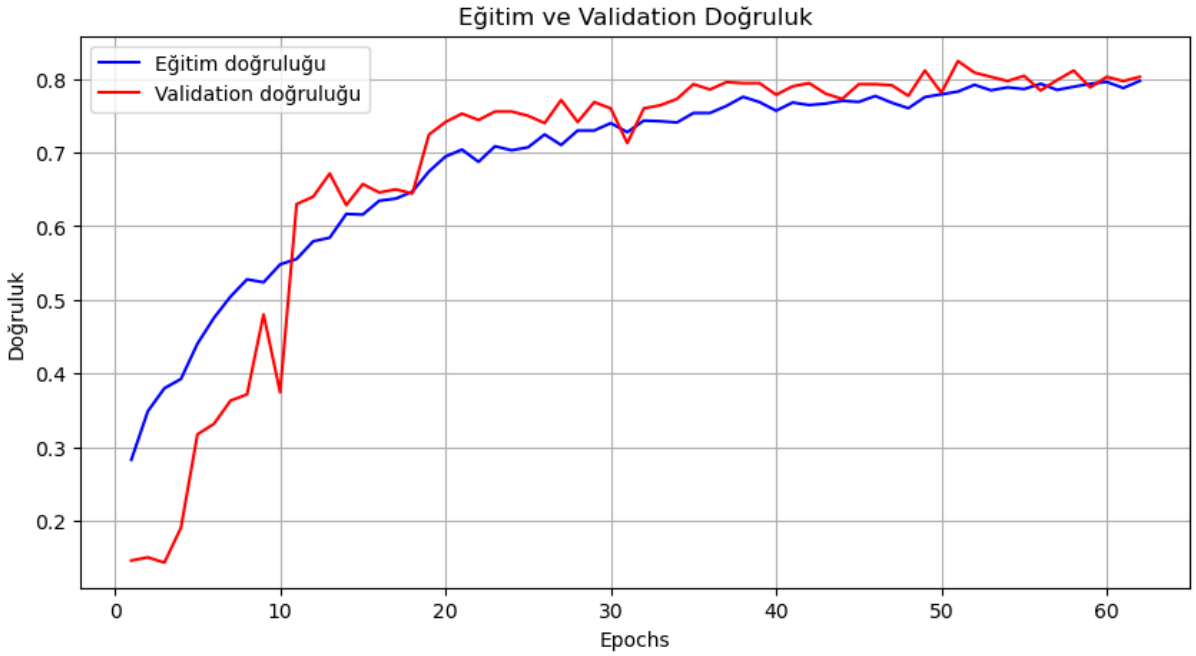
Ben eğittiğim ağda 3 konvolüsyon 3 dense katmanı kullandım. Ayrıca 3 tane maxPlooding ,1 Flatten ve 2 Dropout katmanı kullandım. Kullanılan 3 maksimum havuzlama katmanı, modelin veri boyutunu azaltırken önemli öznitelikleri özetler ve rotasyonel değişmezlik sağlar. Düzleştirme katmanı, modelin tam bağlantılı katmanlarla kullanılmasını ve daha karmaşık ilişkileri öğrenmesini mümkün kılar. Dropout katmanları ise aşırı öğrenmeyi önleyerek modelin genelleme yeteneğini artırır.

Bu katmanların kombinasyonu, daha sağlam, daha verimli ve daha doğru tahminler üreten bir sinir ağı oluşturmamıza yardımcı olur.
Blok şeması aşağıdaki gibidir.

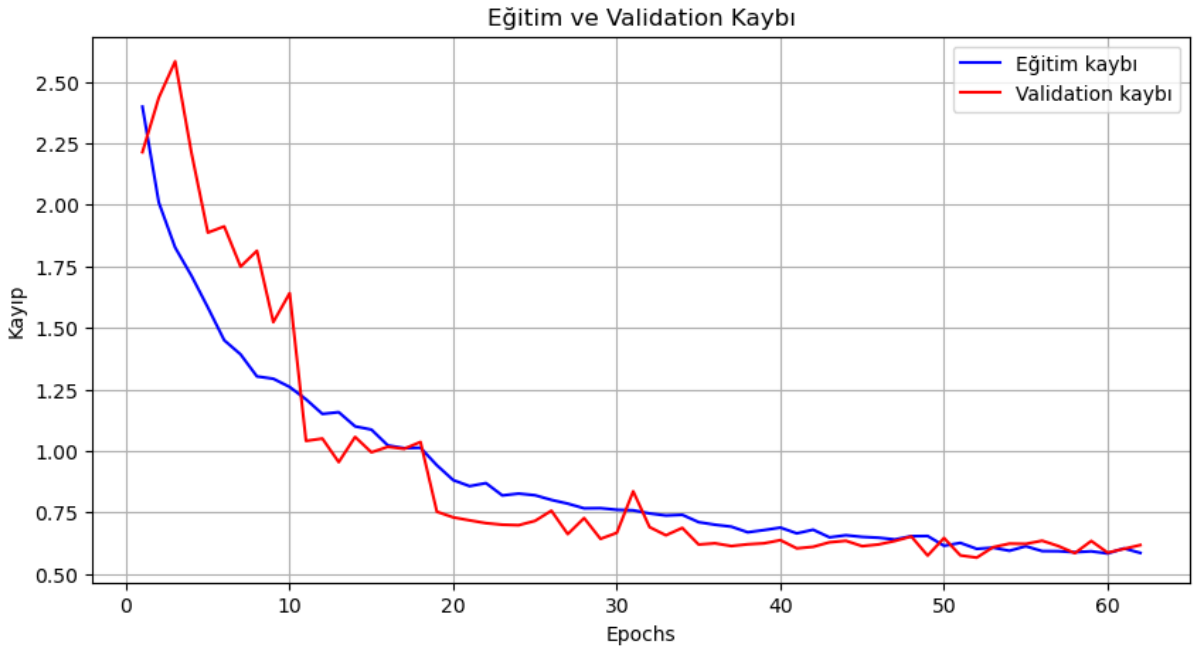


3)

Eğitim doğruluğu başlangıçta hızlı bir şekilde artarak 5. epoch'ta %50'nin üzerine çıkmaktadır. Daha sonra kademeli olarak artmaya devam ederek 50. epoch'ta %80'e ulaşmaktadır. Validation doğruluğu ise eğitim doğruluğuna kıyasla daha dalgalı bir seyir izlemekte ve başlangıçta daha yavaş artmaktadır. 5. epoch'ta %30'a yakın bir değere ulaşırken, 50. epoch'ta %80'e kadar yükselmektedir. Model, eğitim verileri üzerinde hızla öğrenerek yüksek doğruluk seviyesine ulaşmaktadır. Validation doğruluğu da artmaktadır, ancak eğitim doğruluğuna göre daha dalgalı bir şekilde ilerlemektedir. Bu durum, modelin eğitim verilerine aşırı uyum (overfitting) göstermeye başladığına dair bir işaret olabilir.



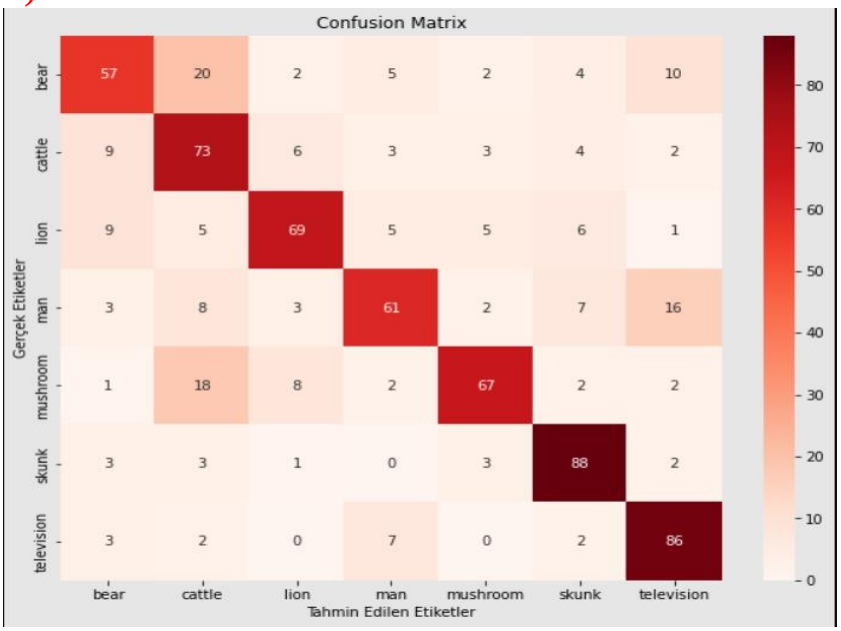
Eğitim kaybı 1. epoch'ta 2.0 seviyesinden başlar ve her epoch'ta kademeli olarak düşerek 50. epoch'ta 0.75'in altına inmektedir. Validation kaybı ise eğitim kaybına kıyasla daha yüksek başlamaktadır ve 1. epoch'ta 2.5 seviyesinden başlar. 50. epoch'ta ise 0.75'e kadar düşmektedir. Model, eğitim verileri üzerinde kaybı önemli ölçüde azaltarak iyi bir performans göstermektedir. Validation kaybı da düşmektedir, ancak eğitim kaybına göre daha dalgalı ve yavaş bir şekilde ilerlemektedir. Bu durum, modelin eğitim verilerine aşırı uyum (overfitting) göstermeye başladığına dair bir işaret olabilir.




```
55/55 ━━━━━━━━━━━━━━━━━━━━ 7s 115ms/step - accuracy: 0.7844 - loss: 0.6055 - val_accuracy: 0.7886 - val_loss: 0.6337 - learni
ng_rate: 1.0000e-04
Epoch 60/100
55/55 ━━━━━━━━━━━━━━━━━━━━ 6s 99ms/step - accuracy: 0.7904 - loss: 0.5991 - val_accuracy: 0.8029 - val_loss: 0.5862 - learnin
g_rate: 1.0000e-04
Epoch 61/100
55/55 ━━━━━━━━━━━━━━━━━━━━ 5s 89ms/step - accuracy: 0.7895 - loss: 0.6044 - val_accuracy: 0.7971 - val_loss: 0.6017 - learnin
g_rate: 1.0000e-04
Epoch 62/100
55/55 ━━━━━━━━━━━━━━━━━━━━ 8s 139ms/step - accuracy: 0.8085 - loss: 0.5600 - val_accuracy: 0.8029 - val_loss: 0.6170 - learni
ng_rate: 1.0000e-04
Epoch 62: early stopping
Restoring model weights from the end of the best epoch: 52.

Model: "sequential_2"
```

4)



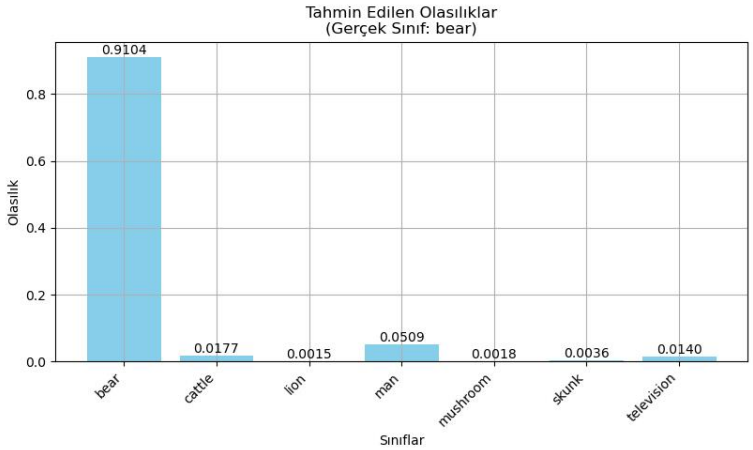
Bu karışıklık matrisi, bir sınıflandırma modelinin performansını değerlendirmek için kullanılır. Matris, modelin doğru ve yanlış sınıflandırma sayısını gösterir. X ekseni tahmin edilen etiketleri, Y ekseni ise gerçek etiketleri gösterir. Renk skalası, doğru ve yanlış sınıflandırmaların yoğunluğunu ifade eder; koyu kırmızı daha yüksek sayıları, açık renkler ise daha düşük sayıları temsil eder.

Bu matrisi yorumlayacak olursak:

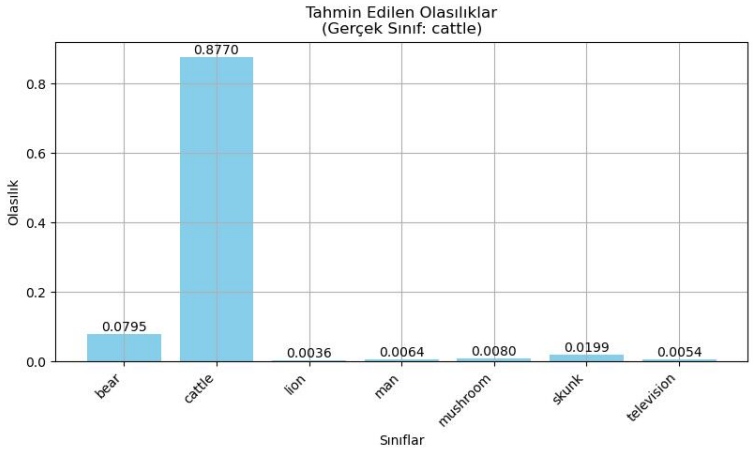
Bear (Ayı): 57 ayı doğru sınıflandırılmış. Ancak 20 tanesi sığır olarak, 5 tanesi mantar olarak ve geri kalanı diğer kategorilere yanlış sınıflandırılmış.
Cattle (Sığır): 73 sığır doğru sınıflandırılmış. Ancak 9 tanesi ayı olarak, 6 tanesi adam olarak ve geri kalanı diğer kategorilere yanlış sınıflandırılmış.
Lion (Aslan): 69 aslan doğru sınıflandırılmış. Ancak 5 tanesi ayı olarak, 3 tanesi sığır olarak ve geri kalanı diğer kategorilere yanlış sınıflandırılmış.
Man (Adam): 61 adam doğru sınıflandırılmış. Ancak 8 tanesi sığır olarak, 7 tanesi televizyon olarak ve geri kalanı diğer kategorilere yanlış sınıflandırılmış.
Mushroom (Mantar): 67 mantar doğru sınıflandırılmış. Ancak 1 tanesi sığır olarak, 3 tanesi skunk olarak ve geri kalanı diğer kategorilere yanlış sınıflandırılmış.
Skunk (Gelincik): 88 gelincik doğru sınıflandırılmış. Ancak 3 tanesi mantar olarak ve geri kalanı diğer kategorilere yanlış sınıflandırılmış.
Television (Televizyon): 86 televizyon doğru sınıflandırılmış. Ancak 7 tanesi mantar olarak ve geri kalanı diğer kategorilere yanlış sınıflandırılmış.
Genel olarak model, en iyi performansı "skunk" (gelincik) ve "television" (televizyon) sınıflarında göstermiş, en kötü performansı ise "man" (adam) sınıfında göstermiş gibi görünüyor. Yanlış sınıflandırmalar, genellikle ayı ve sığır gibi sınıflar arasında daha yaygın. Modelin bazı sınıfları ayırt etmede zorlandığını söyleyebiliriz.

5)

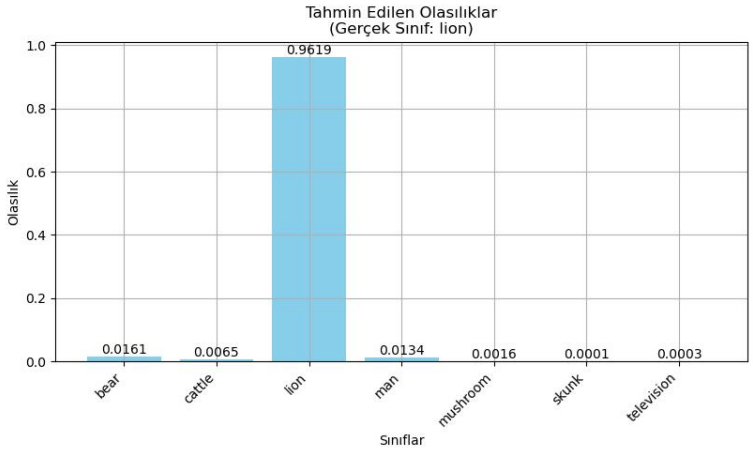
1)**BEAR**

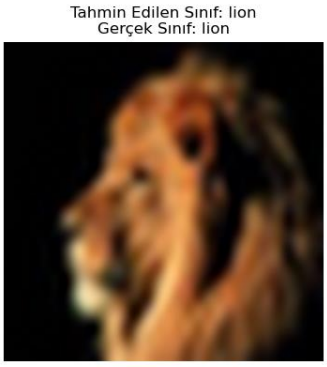


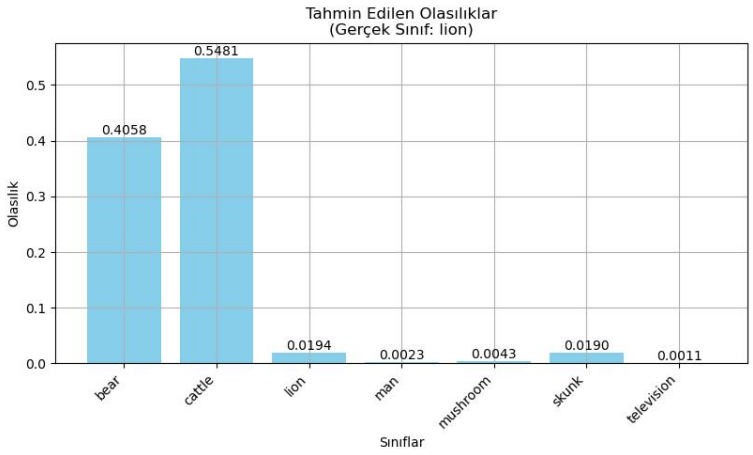
2)CATTLE



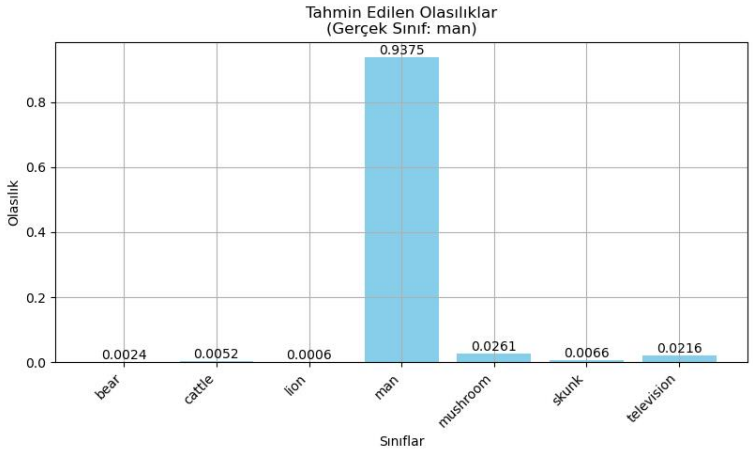
3)LİON



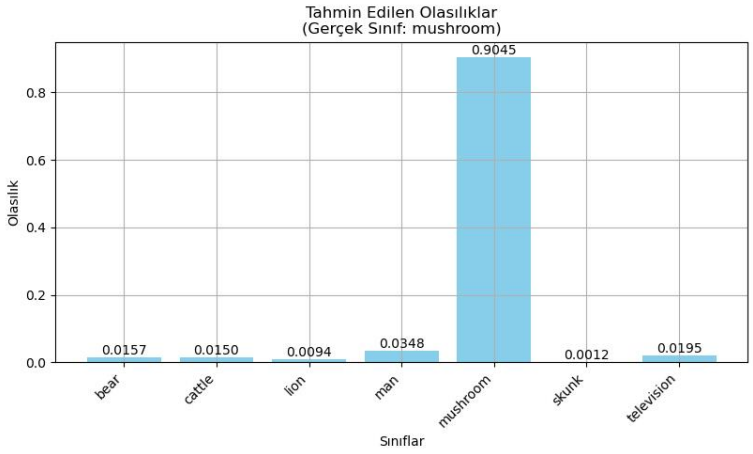




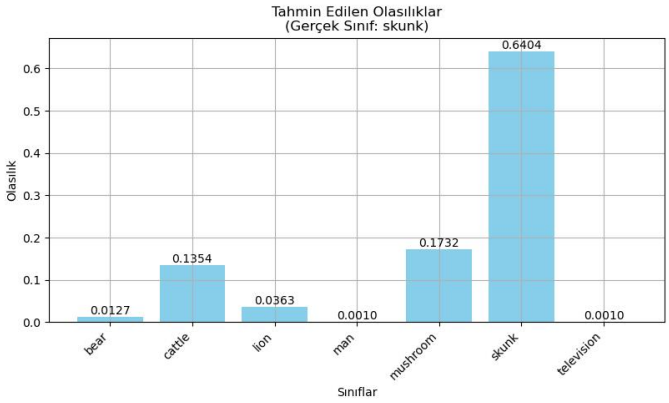
4)MAN



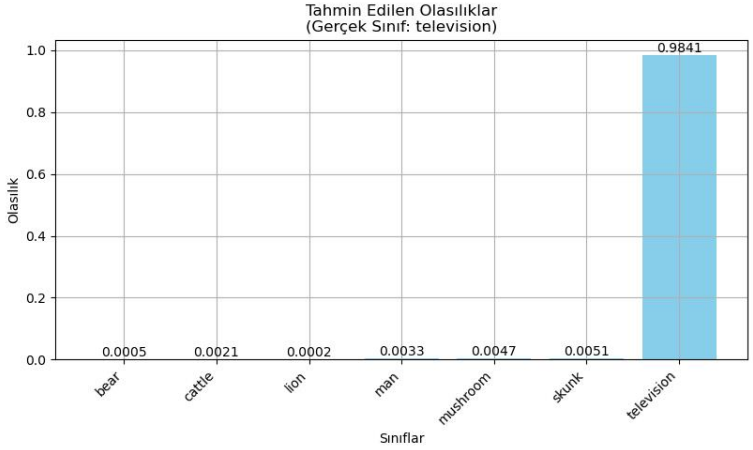
5)MUSHROOM



6)SKUNK



7)TELEVISION



6) ÖDEV KODU

```
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 tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout, BatchNormalization, LeakyReLU
from sklearn.metrics import confusion_matrix
from keras.callbacks import ReduceLROnPlateau, EarlyStopping

import seaborn as sns
from PIL import Image

# 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'
]

def show_class_images(class_indices, num_examples=5):
    (train_images, train_labels), (_, _) =
cifar100.load_data(label_mode='fine')
    fig, axes = plt.subplots(len(class_indices), num_examples, figsize=(12,
12))

    for i, class_index in enumerate(class_indices):
        class_images = train_images[train_labels.flatten() == class_index]
        np.random.shuffle(class_images)

        for j in range(num_examples):
            pil_image = Image.fromarray(class_images[j])
            resized_image = pil_image.resize((pil_image.width * 5,
pil_image.height * 5), Image.BICUBIC)
            axes[i, j].imshow(resized_image)
            axes[i, j].axis('off')

            if j == 0:
                axes[i, j].text(-0.2, 0.5, sinif_isimleri[class_index], fontsize=12,
ha='right', va='center', rotation=0, transform=axes[i, j].transAxes)

    for ax in axes.flat:
        ax.label_outer()
```

```
    plt.tight_layout()
    plt.subplots_adjust(wspace=0.1, hspace=0.1)
    plt.show()

# Veri kümesini yükle
(train_images, train_labels), (test_images, test_labels) =
cifar100.load_data(label_mode='fine')

# Seçilen sınıflar
selected_classes =  [3, 19, 43, 46, 51, 75, 87]

# 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))

# Callbacks
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,
patience=5, min_lr=0.0001)
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
verbose=1, restore_best_weights=True)

# Veri artırma
```

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    zoom_range=0.2,
    shear_range=0.2,
    fill_mode='nearest'
)

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(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(256, 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'])

# 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=[reduce_lr, early_stopping])

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
plt.figure(figsize=(10, 5))
plt.plot(epochs, acc, 'b-', label='Eğitim doğruluğu')
plt.plot(epochs, val_acc, 'r-', label='Validation doğruluğu')
plt.title('Eğitim ve Validation Doğruluk')
plt.xlabel('Epochs')
plt.ylabel('Doğruluk')
plt.legend()
plt.grid(True)
plt.show()

# Kayıp grafiği
plt.figure(figsize=(10, 5))
plt.plot(epochs, loss, 'b-', label='Eğitim kaybı')
plt.plot(epochs, val_loss, 'r-', label='Validation kaybı')
plt.title('Eğitim ve Validation Kaybı')
plt.xlabel('Epochs')
plt.ylabel('Kayıp')
plt.legend()
plt.grid(True)
plt.show()
```

```
# Örnek bir görüntü seçilip modelin test edilmesi
class_indices =  [3, 19, 43, 46, 51, 75, 87]

show_class_images(class_indices)

# Tüm test veri setiyle modeli test edilip çıkış vektörlerinin elde edilmesi
output_vectors = model.predict(test_images_selected)

# Tüm test veri setiyle modeli test et ve tahmin edilen etiketleri elde et
predicted_labels = np.argmax(output_vectors, axis=1)
true_labels = np.argmax(test_labels_selected, axis=1)

# Filtreleme işlemi
filtered_indices = np.where(np.isin(true_labels,
range(len(selected_classes))) & np.isin(predicted_labels,
range(len(selected_classes))))[0]
filtered_true_labels = true_labels[filtered_indices]
filtered_predicted_labels = predicted_labels[filtered_indices]

# 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='Reds', 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 çubuk grafik olarak çiz
    plt.subplot(1, 2, 1)
    plt.bar(range(num_classes), output_vector[0], color='skyblue')
    plt.xticks(range(num_classes), [sinif_isimleri[i] for i in
selected_classes], rotation=45, ha='right')
    plt.xlabel('Sınıflar')
    plt.ylabel('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(i, prob, f'{prob:.4f}', ha='center', va='bottom')

    # 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()
```