



SAKARYA
ÜNİVERSİTESİ

T.C. SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
DERİN ÖĞRENME VE ERİŞİMLİ SİNİR AĞLARI DERSİ
PROJE RAPORU

Çok Sınıflı Sınıflandırıcı Ağı Tasarımı ve Eğitimi

Hazırlayan : Mehmet Oğuz Özkan
Öğrenci Numarası : B201210065

Nisan 2024

Giriş

Geliştireceğimiz proje çok sınıflı sınıflandırıcı niteliği taşıyacaktır. Bu yapıyı kurgulamak için cifar100 datasetini kullanacağız. Oluşturacağımız modelde konvolüsyon ve dense katmanları bulunacak. Modelimizi cifar100 datasetindeki örneklerle eğiterek tahmin yeteneğini ölçeceğiz.

Projede Kullanılan Sınıflar

Projemizde daha önceden belirlenmiş 7 sınıfı kullandık. Bunlar: 8=bicycle, 28=cup, 44=lizard, 50=Mouse, 61=plate, 86=telephone, 98=woman. Cifar100 veri setinde 100 farklı sınıfa ait 50000 eğitim 10000 test görüntüsü bulunmaktadır. Biz kullanacağımız 7 sınıfı bu veri setinden ayıkladık ve one hot encoding yaparak eğitime ve teste hazır hale getirdik.

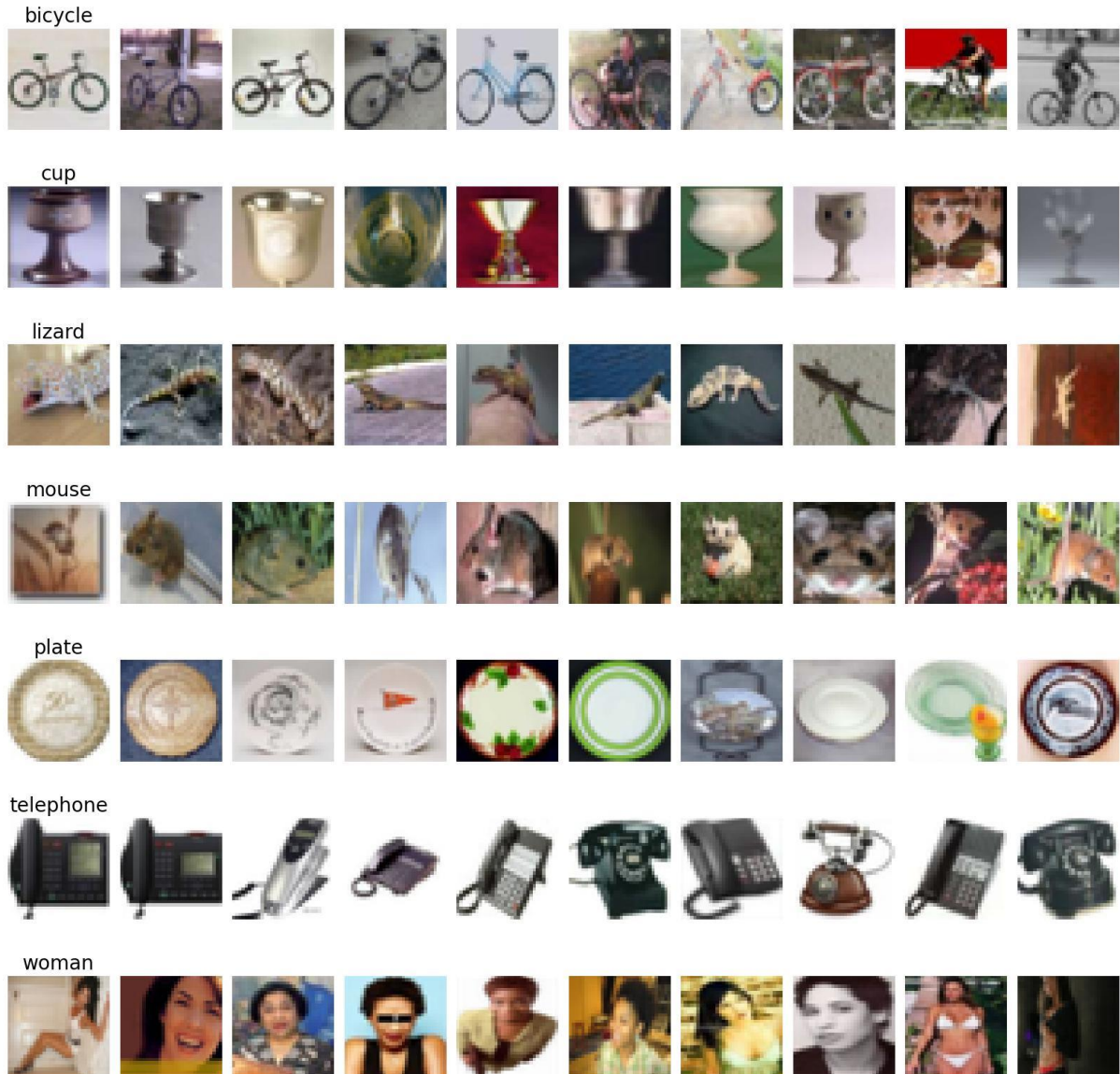


Figure 1: Sınıflara ait örnek görüntüler

Model Tasarımı

Eğitimde kullanacağımız modeli tasarlarken bizden istenen kısıtlara uyarak 3 adet Konvolüsyon katmanı ve 2 adet Dense katmanı kullandım. Konvolüsyon katmanlarında aktivasyon fonksiyonunu Relu ve kullandığım filtrelerin boyutlarını 3x3 olarak ayarladım. Her konvolüsyon katmanından sonra 2x2'lik MaxPooling katmanı ekledim. Özellik çıkarma aşaması bittikten sonra bir Flatten katmanı ile Dense katmanlarına geçtim. 2 Dense katmanı arasına bir Dropout koyarak gereksiz bağlantıları kopardım. Dense2 katmanında Softmax aktivasyon fonksiyonu kullanarak çıkışa ulaştım.

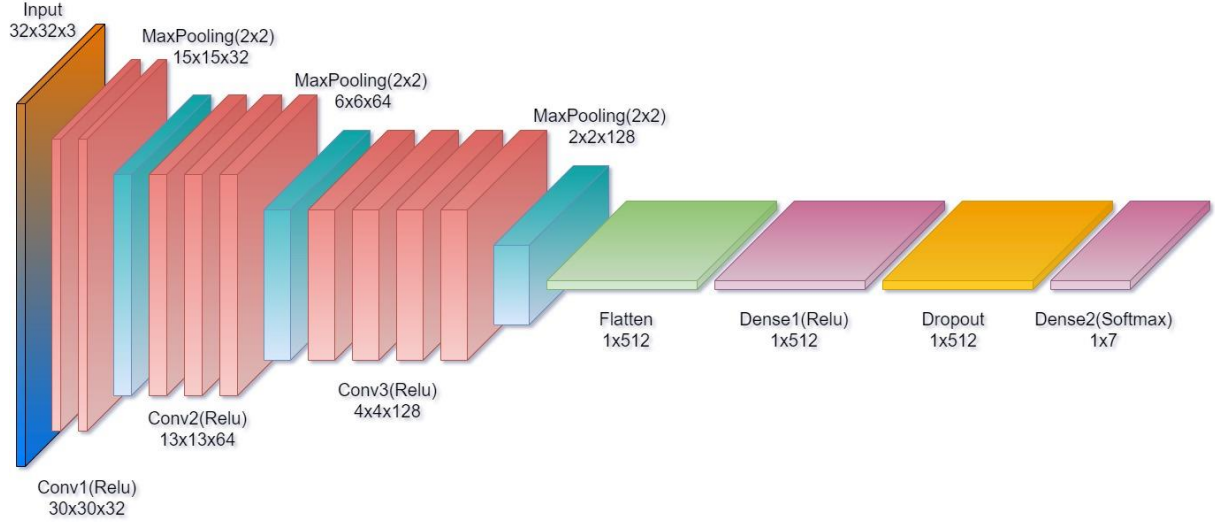


Figure 2: Modelin blok şeması

Eğitim

Tasarladığımız modeli Adam optimizasyonu kullanarak eğittik. Birçok farklı epoch sayısı denememize rağmen modelimizin validation accuracy oranı 0.8'e varamadı ve genelde 0.75 etrafında toparlandı. Epoch sayısını arttırmanın bize fayda sağlamadığı fark ettik. Bu yüzden epoch sayısını ortalama seviyede tutmaya karar verdim. Bu yüzden epoch sayısını 250 olarak kararlaştırdım.

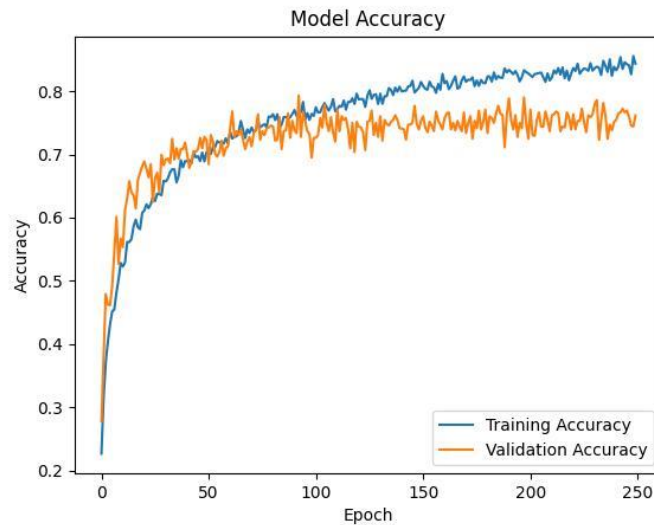


Figure 3: Model doğruluk tablosu

Bu tablo modelimizin eğitim ve doğrulama verileri augment edildikten ve overfit hali ortadan kaldırıldıktan sonraki halidir.

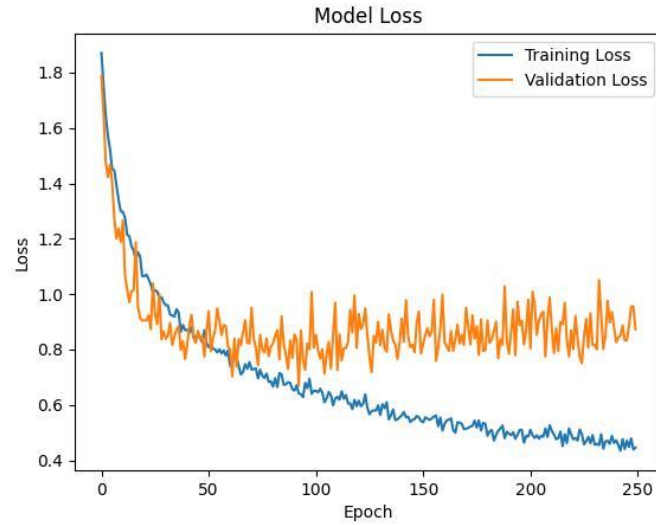


Figure 4: Model kayıp grafikleri

Bu 2 grafiğe baktığımızda modelin eğitim seti için kayıp değeri oldukça düşük (%44.59), bu da modelin eğitim verisine oldukça iyi uyum sağladığını gösterir. Ayrıca, eğitim seti için doğruluk oranı yüksektir (%84.34), yani model eğitim verisinde çoğunlukla doğru tahminler yapmıştır.

Ancak, doğrulama seti için kayıp değeri daha yüksektir (%87.32), bu da modelin doğrulama setinde beklenenden daha fazla hata yaptığını gösterir. Ayrıca, doğrulama seti için doğruluk oranı da eğitim setine göre daha düşüktür (%76.19). Bu da modelin eğitim setine aşırı uyum sağladığını ve genelleştirme yeteneğinin düşük olduğunu gösterebilir.

Genel olarak, model eğitim verisinde iyi performans gösteriyor gibi görünüyor ancak genelleştirme yeteneği düşük olabilir. Modelin daha iyi genelleştirme yapabilmesi için düzenleme teknikleri kullanılabilir veya modelin karmaşıklığı azaltılabilir. Bu, doğrulama setindeki performansı artırarak modelin daha iyi performans göstermesini sağlayabilir.

Confusion (Karmaşıklık) Matrisi

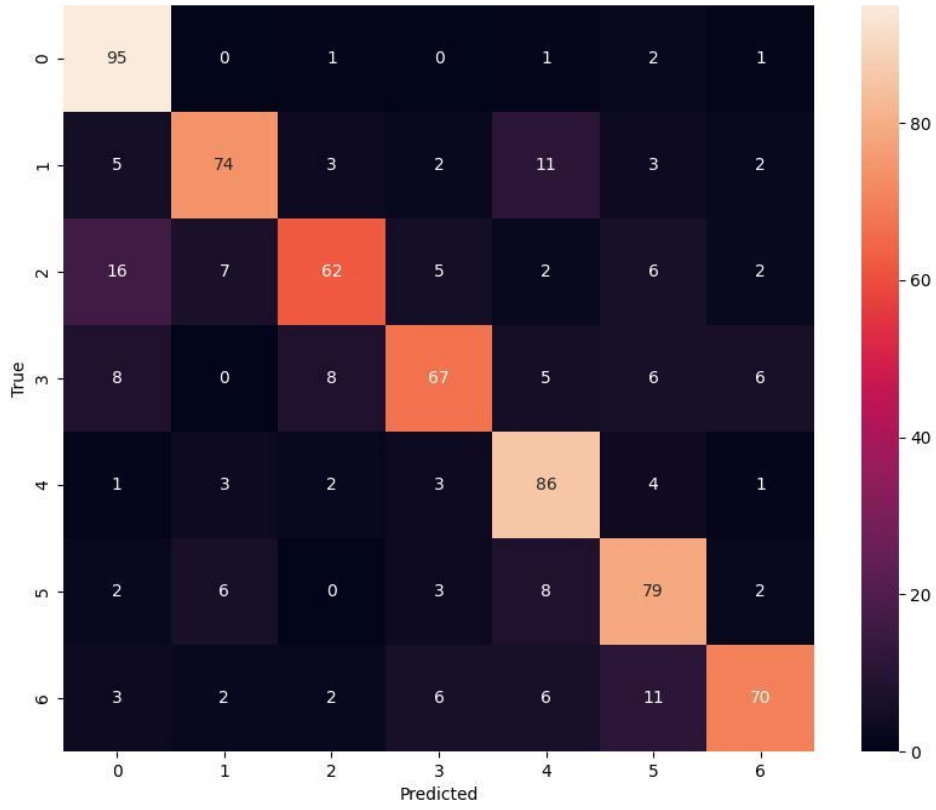
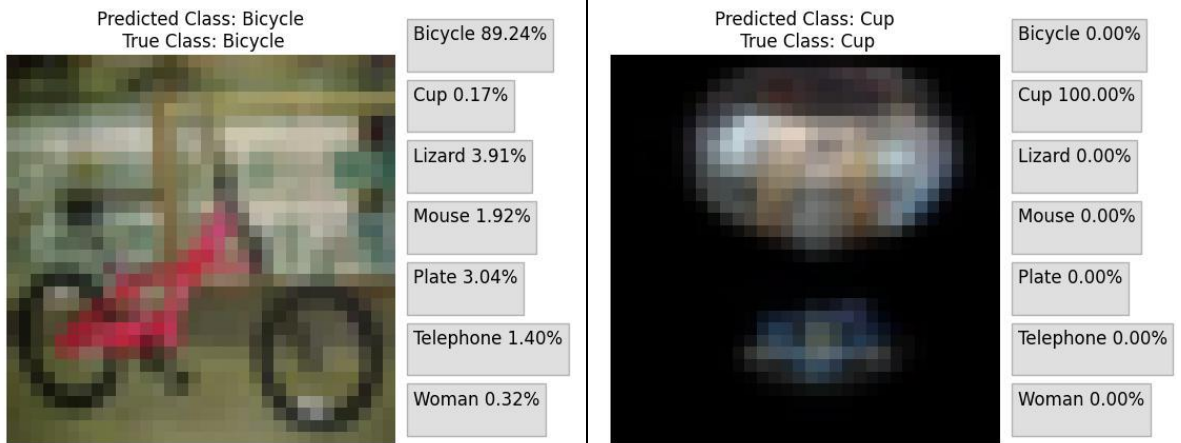


Figure 5: Model test edilirken ortaya çıkan karmaşıklık matrisi

Karmaşıklık matrisi ya da confusion matrix, bir sınıflandırma modelinin performansını değerlendirmek için kullanılan bir araçtır. Confusion matrix, modelin gerçek ve tahmin edilen sınıfları ne kadar doğru veya yanlış tahmin ettiğini gösterir.

Örnek Test Sonuçları



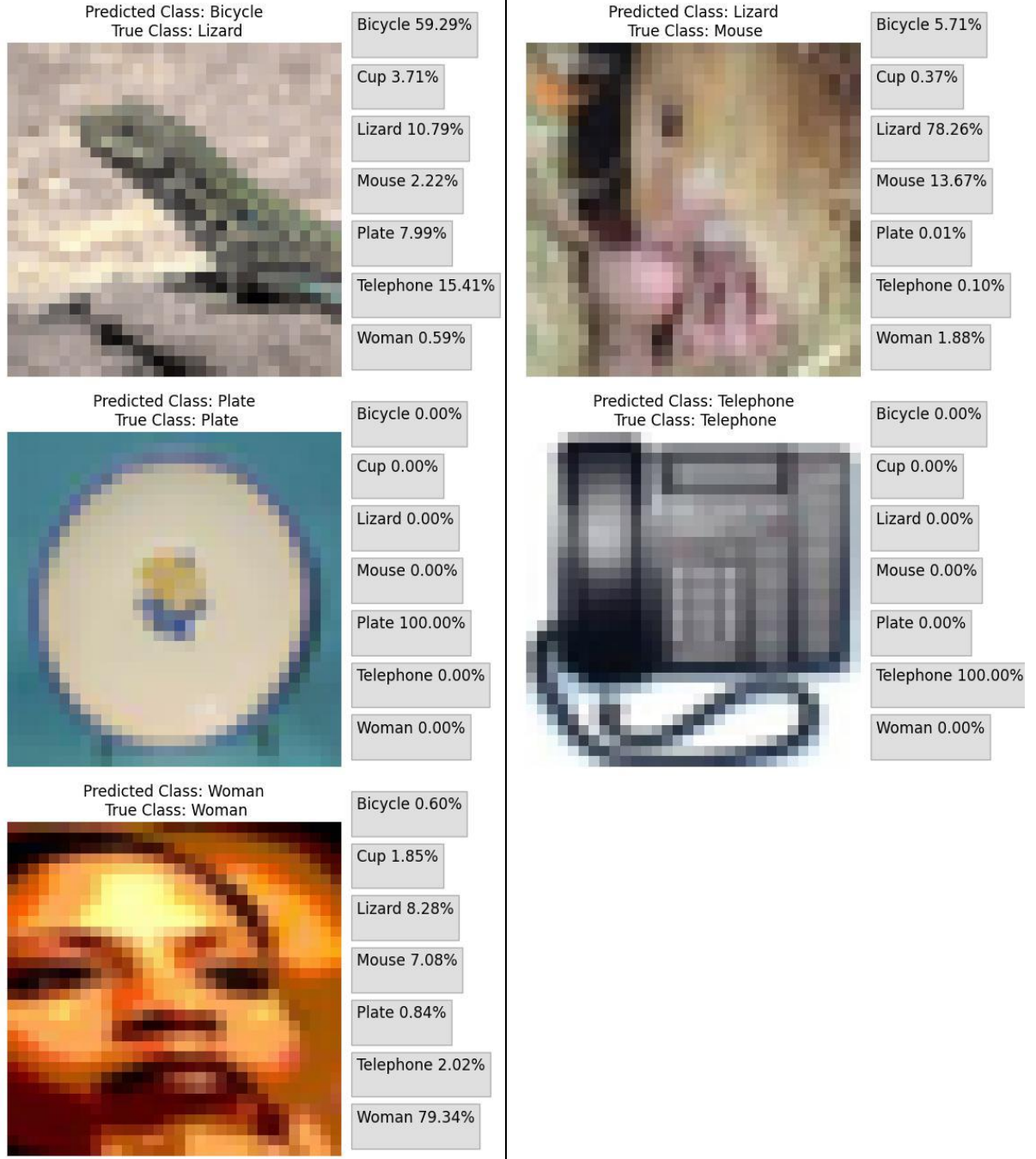


Figure 6: Örnek test sonuçları

Her sınıf için yapılmış örnek birer predict() sonucu doğru sonuçları ile birlikte yukarıda verilmiştir. Modelin daha önce görmediği resimleri tahmin etmesi istenmiştir. Bunun sonucunda modelin hangi resmi hangi sınıfa daha yakın gördüğüne dair ayrıntılı yüzdesel tahminleri yukarıda verilmiştir.

Proje Kodları

Bu projeyi gerçekleştirirken kullandığımız kodların tamamı aşağıda verilmiştir. Bu kodları kullanarak modeli olduğu gibi yeniden tasarlayabilirsiniz.

Gerekli Kütüphaneler

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.metrics import confusion_matrix
4 from keras.datasets import cifar100
5 from keras.models import Sequential
6 from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
7 from keras.utils import to_categorical
8 import seaborn as sns
9 import random
10 import time
11 from keras.preprocessing.image import ImageDataGenerator
12 from datetime import datetime
```

Fonksiyonlar

```
1 def load_dataset(selected_classes):
2     (X_train, y_train), (X_test, y_test) = cifar100.load_data(Label_mode='fine')
3     train_mask = np.isin(y_train, selected_classes).flatten()
4     test_mask = np.isin(y_test, selected_classes).flatten()
5     X_train, y_train = X_train[train_mask], y_train[train_mask]
6     X_test, y_test = X_test[test_mask], y_test[test_mask]
7     return (X_train, y_train), (X_test, y_test)
```

```
1 def display_sample_images(X_train, y_train, selected_classes, class_names, num_images=10):
2     fig, axes = plt.subplots(len(selected_classes), num_images, figsize=(16, 16))
3     for i, class_label in enumerate(selected_classes):
4         class_indices = np.where(y_train.flatten() == class_label)[0]
5         np.random.shuffle(class_indices)
6         for j in range(num_images):
7             axes[i, j].imshow(X_train[class_indices[j]])
8             axes[i, j].axis('off')
9             if j == 0:
10                 axes[i, j].set_title(f"{class_names[class_label]}", fontsize=20)
11     plt.tight_layout()
12     plt.savefig('images/display_sample_images.jpg')
13     plt.show()
```



```
1 def create_model(input_shape, num_classes):
2     model = Sequential()
3     model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
4     model.add(MaxPooling2D(pool_size=(2, 2)))
5     model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
6     model.add(MaxPooling2D(pool_size=(2, 2)))
7     model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
8     model.add(MaxPooling2D(pool_size=(2, 2)))
9     model.add(Flatten())
10    model.add(Dense(512, activation='relu'))
11    model.add(Dropout(0.5))
12    model.add(Dense(num_classes, activation='softmax'))
13    return model
```



```
1 def plot_training_history(history):
2     plt.plot(history.history['accuracy'], Label='Training Accuracy')
3     plt.plot(history.history['val_accuracy'], Label='Validation Accuracy')
4     plt.title('Model Accuracy')
5     plt.xlabel('Epoch')
6     plt.ylabel('Accuracy')
7     plt.legend()
8     plt.savefig('images/model_accuracy.jpg')
9     plt.show()
10
11    plt.plot(history.history['loss'], Label='Training Loss')
12    plt.plot(history.history['val_loss'], Label='Validation Loss')
13    plt.title('Model Loss')
14    plt.xlabel('Epoch')
15    plt.ylabel('Loss')
16    plt.legend()
17    plt.savefig('images/model_loss.jpg')
18    plt.show()
```




```
1 def evaluate_model(model, X_test, y_test, num_classes):
2     y_pred = model.predict(X_test)
3     y_pred_classes = np.argmax(y_pred, axis=1)
4     y_test_classes = np.argmax(y_test, axis=1)
5     cm = confusion_matrix(y_test_classes, y_pred_classes)
6     plt.figure(figsize=(10, 8))
7     sns.heatmap(cm, annot=True, fmt='d')
8     plt.xlabel('Predicted')
9     plt.ylabel('True')
10    plt.savefig('images/confusion_matrix.jpg')
11    plt.show()
```



```
1 def plot_sample(X, y, index, class_names):
2     plt.figure(figsize = (4,4))
3     plt.imshow(X[index])
4     plt.xlabel(class_names[int(y[index][0])])
```



```
1 def transform_labels(y,num):
2     for i in range(len(y)):
3         if y[i]==8:
4             y[i]=0
5         if y[i]==28:
6             y[i]=1
7         if y[i]==44:
8             y[i]=2
9         if y[i]==50:
10            y[i]=3
11        if y[i]==61:
12            y[i]=4
13        if y[i]==86:
14            y[i]=5
15        if y[i]==98:
16            y[i]=6
17
18    y=to_categorical(y,num_classes=num)
19    return y
```



```
1 def evaluate_random_sample(model, X_test, y_test, test_class_names):
2     index = np.random.randint(0, len(X_test))
3     sample_image = X_test[index]
4     sample_label = y_test[index]
5     prediction = model.predict(np.expand_dims(sample_image, axis=0))
6     predicted_class = np.argmax(prediction)
7
8     true_class_name = test_class_names[np.argmax(sample_label)]
9     predicted_class_name = test_class_names[predicted_class]
10
11    plt.imshow(sample_image)
12    plt.title(f"Predicted Class: {predicted_class_name}\n True Class: {true_class_name}")
13    plt.axis('off')
14    i=0
15    for class_name in test_class_names:
16        plt.text(+33, i*5, f"{test_class_names[class_name]} {prediction[0][class_name]*100:.2f}%\n", fontsize=12, bbox=dict(facecolor='gray', alpha=0.25))
17        i=i+1
18    current_time = datetime.now().strftime("%Y%m%d%H%M%S")
19    plt.savefig(f'images/random_test_{current_time}.jpg',bbox_inches='tight')
20    plt.show()
```


Veri Hazırlama

```
1 selected_classes = [8, 28, 44, 50, 61, 86, 98]
2 class_names = {8: "bicycle", 28: "cup", 44: "lizard", 50: "mouse", 61: "plate", 86: "telephone", 98: "woman"}
3
4 (X_train, y_train), (X_test, y_test) = load_dataset(selected_classes)
5 print(X_train.shape)
6 print(y_train.shape)
7 print(X_test.shape)
8 print(y_test.shape)
9
10 display_sample_images(X_train, y_train, selected_classes, class_names)
11
12 input_shape = X_train.shape[1:]
13 num_classes = len(selected_classes)
14 X_train = X_train.astype('float32') / 255
15 X_test = X_test.astype('float32') / 255
16 y_train = transform_labels(y_train, num_classes)
17 y_test = transform_labels(y_test, num_classes)
18
19 train_datagen = ImageDataGenerator(
20     rotation_range=40,
21     width_shift_range=0.2,
22     height_shift_range=0.2,
23     shear_range=0.2,
24     zoom_range=0.2,
25     horizontal_flip=True,
26     fill_mode='nearest'
27 )
28 validation_datagen = ImageDataGenerator()
29
30 train_generator = train_datagen.flow(X_train, y_train, batch_size=32)
31 validation_generator = validation_datagen.flow(X_test, y_test, batch_size=32)
```

Model İnşası


```
1 model = create_model(input_shape, num_classes)
2 model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
3 model.summary()
4
5 history = model.fit(
6     train_generator,
7     steps_per_epoch=len(X_train) // 32,
8     epochs=250,
9     validation_data=validation_generator,
10    validation_steps=len(X_test) // 32
11 )
```

Analiz




```
1 plot_training_history(history)
2
3 print("Confusion Matrix:")
4 evaluate_model(model, X_test, y_test, num_classes)
```

Test



```
1 test_class_names = {0: "Bicycle", 1: "Cup", 2: "Lizard", 3: "Mouse", 4: "Plate", 5: "Telephone", 6: "Woman"}
2 evaluate_random_sample(model, X_test, y_test, test_class_names)
```

Kaydetme



```
1 model.save("models/model.h5")
```