DERİN ÖĞRENME FİNAL PROJESİ

HAZIRLAYAN

ADI SOYADI **:** FURKAN DOĞAN

ÖĞRENCİ NUMARASI : 180303026

DERS ADI **:** DERİN ÖĞRENME

DERS YÜRÜTÜCÜSÜ : DR. SAEİD AGAHİAN

İÇİNDEKİLER

1. GİRİŞ

Problemin Tanımı

Bu projede, "Köpek mi Kedi mi?" sorusunu cevaplamak için bir görüntü sınıflandırma modeli oluşturmayı amaçlıyoruz.

Kaggle platformunda bulunan "Dogs vs. Cats" veri setini kullanacağız.

Bu projede, derin öğrenme tekniklerini kullanarak, bir görüntüde köpek mi yoksa kedi mi olduğunu tahmin etmek için bir sinir ağı modeli geliştireceğiz.

Veri Seti Açıklaması

Veri seti, Kaggle'da "Dogs vs. Cats" yarışması için sağlanan bir veri setidir.

Veri seti, 25.000 adet köpek ve kedi görüntüsünden oluşmaktadır. Fakat ben bu veri setini 7500 köpek 7500 de kedi olmak üzere azalttım.

Her bir görüntü, RGB renk uzayında ortalama boyutlara sahip fakat aynı boyutlarda bulunmayan görüntülere sahiptir. Görüntüler jpeg formatındadır.

Veri seti, eğitim ve test veri setlerine bölünmüştür. Eğitim seti 12.000 görüntüden oluşurken, test seti 3.000 görüntü içermektedir.

Amaç ve Hedefler

Bu projenin amacı, köpek ve kedi görüntülerini sınıflandırmak için bir sinir ağı modeli oluşturmaktır.

Modelin doğruluk oranını artırmak ve yanlış sınıflandırmaları minimize etmek hedeflenmektedir.

Sonuç olarak, gerçek dünyadaki görüntülerde de köpek ve kedi türlerini doğru bir şekilde sınıflandırabilecek, güvenilir bir model elde etmeyi hedefliyoruz.

2. VERİ HAZIRLIĞI

Bu kod parçası, eğitim setindeki örnek görüntüleri görselleştirmek için kullanılır. İlk olarak, fig adında bir Figure nesnesi oluşturulur ve figsize parametresiyle boyutu belirlenir. suptitle metoduyla başlık ayarlanır. Ardından, bir döngü yardımıyla ilk 25 örnek görüntüyü işleyeceğiz. subplot fonksiyonuyla 5x5 bir alt-parça düzeni oluşturulur ve i + 1 indeksiyle her alt-parçaya sırasıyla erişilir. load_img fonksiyonuyla görüntü yolu alınır ve imshow fonksiyonuyla görüntü ekrana çizdirilir. Son olarak, axis("off") ile eksenler gizlenir, tight_layout ile alt-parçalar arasındaki boşluklar ayarlanır ve show metoduyla görüntü gösterilir.

```
fig = plt.figure(1, figsize = (8, 8))
fig.suptitle("Eğitim seti örnek görüntüler")

for i in range(25):

    plt.subplot(5, 5, i + 1)
    image = load_img(FILES + "/train/" + train_df["file"][i])
    plt.imshow(image)
    plt.axis("off")

plt.tight_layout()
plt.show()
```

Aynı işlemi köpek görüntüleri ve kedi görüntüleri için de tekrarlıyoruz.

Bu kod parçasında, yalnızca köpek sınıfına ait örnek görüntüler gösterilmektedir. train_df veri çerçevesinde "label" sütununa göre filtreleme yapılır ve yalnızca "dog" etiketi olan görüntüler seçilir.

```
fig = plt.figure(1, figsize = (8, 8))
fig.suptitle("Eğitim seti örnek köpek görüntüleri")

for i in range(25):

    plt.subplot(5, 5, i + 1)
    image = load_img(FILES + "/train/" + train_df.query("label == 'dog'").file.values[i])
    plt.imshow(image)
    plt.axis("off")

plt.tight_layout()
plt.show()
```

```
fig = plt.figure(1, figsize = (8, 8))
fig.suptitle("Eğitim seti örnek kedi görüntüleri")

for i in range(25):

    plt.subplot(5, 5, i + 1)
    image = load_img(FILES + "/train/" + train_df.query("label == 'cat'").file.values[i])
    plt.imshow(image)
    plt.axis("off")

plt.tight_layout()
plt.show()
```

Bu kod parçasında ise yalnızca kedi sınıfına ait örnek görüntüler gösterilmektedir. Yine train_df veri çerçevesinde "label" sütununa göre filtreleme yapılır ve yalnızca "cat" etiketi olan görüntüler seçilir.

3. VGG-16 UYGULANMASI

VGG-16, 2014 yılında Visual Geometry Group (VGG) tarafından geliştirilen bir evrişimli sinir ağı (CNN) modelidir. VGG-16, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) yarışmasında o dönemdeki birçok görevde en iyi performansı göstererek önemli bir başarı elde etmiştir. Aşağıda, VGG-16'nın detayları, avantajları ve dezavantajları bulunmaktadır:

Mimarisi:

- VGG-16, 16 adet evrişim katmanı ve 3 adet tam bağlı (dense) katmandan oluşur.
- Evrişim katmanlarında 3x3 boyutunda filtreler kullanılır ve ardışık olarak uygulanır.
- Maksimum havuzlama (MaxPooling) katmanları, her evrişim bloğunun ardından yer alır.
- Bu yapı, modelin daha derin ve karmaşık özellikleri öğrenmesini sağlar.

Öğrenme Kapasitesi:

- VGG-16, çok sayıda parametreye sahip bir modeldir. Toplamda yaklaşık 138 milyon parametreye sahiptir.
- Büyük veri setleriyle eğitildiğinde, genel olarak iyi bir performans sergiler.
- Derinlik ve karmaşıklık, daha fazla öğrenme kapasitesi sağlar.

Genellemenin İyi Olması:

- VGG-16, transfer öğrenme yöntemleriyle iyi bir şekilde kullanılabilir. Önceden eğitilmiş ağırlıkları kullanarak, farklı görsel tanıma görevlerinde iyi sonuçlar elde edilebilir.
- Genel olarak, çeşitli görüntü sınıflandırma görevlerinde yüksek doğruluk oranları elde etmiştir.

Basit ve Düzenli Bir Yapı:

- Evrişim ve havuzlama katmanlarının tekrarlanan basit yapısı, modelin anlaşılmasını ve uygulanmasını kolaylaştırır.
- Aynı boyutta filtrelerin tekrarlanması, modelin düzenli bir şekilde çalışmasını sağlar.

Avantajları:

- VGG-16, daha derin mimarilere kıyasla daha basit ve anlaşılır bir yapıya sahiptir.
- Genel olarak, yüksek doğruluk oranları elde edebilir, özellikle büyük veri setleriyle eğitildiğinde.
- Transfer öğrenme için iyi bir başlangıç noktasıdır.

Dezavantajları:

- VGG-16, çok sayıda parametreye sahip olduğu için eğitimi daha uzun sürebilir ve daha fazla hesaplama gücü gerektirebilir.

- Bu modelin derinliği, ağırlıkların saklanması ve uygulanması için daha fazla bellek gerektirebilir.
- Daha hafif ve daha küçük bir model tercih ediliyorsa, VGG-16 modeli yeterli olmayabilir.

Sonuç olarak, VGG-16 modeli, evrişimli sinir ağı mimarileri arasında önemli bir yer tutan ve birçok görüntü sınıflandırma görevinde başarılı sonuçlar elde eden bir modeldir. Ancak, daha hafif ve daha hızlı modeller tercih ediliyorsa, daha yeni ve optimize edilmiş mimariler kullanılabilir.

```python
def VGG_16(input_shape = (224, 224, 3), n_classes = 1000):

    model = Sequential(
        [
            Conv2D(filters = 64, kernel_size = (3, 3), padding = "same", activation = "relu", input_shape = input_shape),
            Conv2D(filters = 64, kernel_size = (3, 3), padding = "same", activation = "relu"),
            MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

            Conv2D(filters = 128, kernel_size = (3, 3), padding = "same", activation = "relu"),
            Conv2D(filters = 128, kernel_size = (3, 3), padding = "same", activation = "relu"),
            MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

            Conv2D(filters = 256, kernel_size = (3, 3), padding = "same", activation = "relu"),
            Conv2D(filters = 256, kernel_size = (3, 3), padding = "same", activation = "relu"),
            Conv2D(filters = 256, kernel_size = (3, 3), padding = "same", activation = "relu"),
            MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

            Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"),
            Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"),
            Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"),
            MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

            Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"),
            Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"),
            Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"),
            MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

            Flatten(),
            Dense(units = 4096, activation = "relu"),
            Dense(units = 4096, activation = "relu"),
            Dense(units = n_classes, activation = "softmax")
        ]
    )

    return model
```

Bu kod parçası, VGG-16 modelinin bir implementasyonunu içerir. Bu model, giriş olarak (224, 224, 3) boyutunda bir görüntü tensörü alır ve n_classes parametresiyle belirtilen sayıda sınıf için sınıflandırma yapar.

Ağın temel yapısı şu şekildedir:

Evrişim Katmanları:

- 64 filtreli 3x3 boyutunda bir evrişim katmanı, 'relu' aktivasyon fonksiyonu kullanılarak uygulanır.

- 64 filtreli bir başka 3x3 boyutunda bir evrişim katmanı, yine 'relu' aktivasyon fonksiyonu ile uygulanır.
- Ardından, 2x2 boyutunda bir maksimum havuzlama (MaxPooling) katmanı uygulanır.
- Bu adımlar, 128 filtreli iki evrişim katmanı için tekrarlanır.
- Ardından, 256 filtreli üç evrişim katmanı için aynı işlemler tekrarlanır.
- Bir sonraki aşamada, 512 filtreli üç evrişim katmanı daha uygulanır.
- Son olarak, 512 filtreli üç evrişim katmanı daha uygulanır.
- Maksimum havuzlama katmanları, her evrişim bloğunun ardından tekrarlanır.
- Düzleştirme katmanı ile evrişim bloklarının çıktıları düzleştirilir.
- Ardından, 4096 birimli iki tam bağlı (dense) katman uygulanır, her biri 'relu' aktivasyon fonksiyonu ile.
- Son katman, n_classes sayısına eşit olan birimlerle tam bağlı bir sınıflandırma katmanıdır. Bu katmanda 'softmax' aktivasyon fonksiyonu kullanılarak sınıf olasılıkları hesaplanır.

Bu şekilde, VGG-16 modeli oluşturulur ve fonksiyon son olarak bu modeli döndürür.

4) VERİ İŞLEME VE ÇOĞALTMA

Bu adımda, kullanılacak olan veri setinin önişleme işlemleri gerçekleştirilmiştir.

```
train_data, val_data = train_test_split(train_df,
                                        test_size = 0.2,
                                        stratify = train_df["label"],
                                        random_state = 666)
```

Bu kod parçası, veri setini eğitim ve doğrulama kümelerine bölmek için kullanılan bir işlemi gerçekleştirir. Aşağıda detaylar verilmiştir

- train_df: Veri seti, bir DataFrame olarak varsayılan olarak temsil edilir.
- train_test_split: Bu, sklearn.model_selection modülünden bir işlevdir. Veri setini eğitim ve test kümelerine ayırmak için kullanılır.
- train_data ve val_data: Ayırma işlemi sonucunda oluşacak olan eğitim ve doğrulama veri kümelerini temsil eden değişkenlerdir.
- test_size = 0.2: Ayırma işleminde, veri setinin yüzde 20'sini doğrulama veri kümesi olarak ayırmak için kullanılır. Bu durumda, veri setinin yüzde 80'i eğitim veri kümesi olarak kalır.
- stratify = train_df["label"]: Ayırma işleminde, sınıf dağılımının korunmasını sağlamak için kullanılır. label sütunu, veri setindeki örneklerin etiketlerini içerir ve bu sütuna göre ayırma işlemi yapılır.

- random_state = 666: Ayırma işleminin rastgele gerçekleşmesi için kullanılan bir değerdir.

Veri Çoğaltma

```
datagen = ImageDataGenerator(
    rotation_range = 30,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    brightness_range = (0.5, 1),
    zoom_range = 0.2,
    horizontal_flip = True,
    rescale = 1./255,
)

sample_df = train_data.sample(1)

sample_generator = datagen.flow_from_dataframe(
    dataframe = sample_df,
    directory = FILES + "/train/",
    x_col = "file",
    y_col = "label",
    class_mode = "categorical",
    target_size = (224, 224),
    seed = 666
)

plt.figure(figsize = (14, 8))

for i in range(50):

    plt.subplot(5, 10, i + 1)

    for X, y in sample_generator:

        plt.imshow(X[0])
        plt.axis("off")
        break

plt.tight_layout()
plt.show()
```

Bu kod parçası, veri artırma işlemlerini kullanarak bir örnek görüntüye çeşitli dönüşümler ve manipülasyonlar uygulayarak yeni örnekler oluşturur. Bu, eğitim veri kümesinin çeşitliliğini artırarak modelin daha genellemesini ve daha iyi performans göstermesini sağlar.

Veri artırma parametreleri:

- rotation_range = 30: Görüntüleri rastgele olarak 0 ile 30 derece arasında döndürmek için kullanılır.
- width_shift_range = 0.1: Görüntüleri yatay yönde rastgele olarak 0.1 oranında kaydırmak için kullanılır.
- height_shift_range = 0.1: Görüntüleri dikey yönde rastgele olarak 0.1 oranında kaydırmak için kullanılır.
- brightness_range = (0.5, 1): Görüntülerin parlaklığını rastgele olarak 0.5 ile 1 arasında değiştirir.
- zoom_range = 0.2: Görüntüleri rastgele olarak yakınlaştırır veya uzaklaştırır.
- horizontal_flip = True: Görüntüleri yatay eksende rastgele olarak çevirir.
- rescale = 1./255: Görüntü piksellerinin değerlerini 0-255 aralığından 0-1 aralığına yeniden ölçeklendirir.
- sample_df: Örnek bir veri çerçevesi (DataFrame) alınır. Bu örnekte, eğitim veri kümesinden rastgele bir örnek seçilir.
- datagen.flow_from_dataframe: Veri artırma işlemlerini uygulamak için bir veri akışı oluşturur.
- dataframe: Veri çerçevesi, bu durumda sample_df, kullanılır.
- directory: Görüntülerin bulunduğu dizin belirtilir.
- x_col: Görüntü dosyalarının bulunduğu sütunun adı belirtilir.
- y_col: Etiketlerin bulunduğu sütunun adı belirtilir.
- class_mode: Sınıf modu, bu durumda "categorical" olarak belirtilir.
- target_size: Görüntülerin hedef boyutu belirtilir.
- seed: Rastgele dönüşümlerde tekrarlanabilirlik sağlamak için kullanılan çekirdek değeri belirtilir.
- plt.figure ve plt.subplot: Bir matplotlib figürü ve alt grafikleri oluşturur.
- sample_generator ile döngü: Her bir örneğin veri artırma uygulanmış hali elde edilir ve görselleştirilir.

- for X, y in sample_generator: sample_generator üzerinde döngü oluşturulur ve her iterasyonda X (görüntü verisi) ve y (etiket) alınır.
- plt.imshow(X[0]): İlk görüntüyü alır ve görselleştirir.
- plt.axis("off"): Eksenleri kapatır.
- break: İlk örneği aldıktan sonra döngüden çıkar.
- plt.tight_layout ve plt.show: Grafikleri düzenler ve gösterir.

```
train_datagen = ImageDataGenerator(
    rotation_range = 15,
    horizontal_flip = True,
    preprocessing_function = preprocess_input
)

val_datagen = ImageDataGenerator(preprocessing_function = preprocess_input)
```

Bu kod parçası, veri artırma işlemleri ve görüntü ön işleme işlemleri için ImageDataGenerator nesnelerini tanımlar. İşlem adımlarını aşağıda açıklamaya çalışacağım:

- train_datagen: Eğitim veri seti için bir ImageDataGenerator nesnesi oluşturulur. Bu nesne, eğitim verileri üzerinde çeşitli veri artırma yöntemlerini uygulamak için kullanılacak.
- rotation_range = 15: Görüntüleri rastgele olarak -15 ile +15 derece arasında döndürmek için kullanılır. Bu, görüntülerin çeşitliliğini artırır ve modele daha fazla genelleme yeteneği kazandırır.
- horizontal_flip = True: Görüntüleri yatay eksende rastgele olarak çevirmek için kullanılır. Bu, görüntülerin simetrisini kullanarak daha fazla veri çeşitliliği elde etmeyi sağlar.
- preprocessing_function = preprocess_input: Görüntü ön işleme işlevi, preprocess_input olarak belirtilir. Bu işlev, görüntü verilerinin önceden işlenmesi için kullanılır. Örneğin, VGG-16 modelinin gerektirdiği şekilde görüntüleri ölçeklendirir ve merkezler.
- val_datagen: Doğrulama veri seti için bir ImageDataGenerator nesnesi oluşturulur. Bu nesne, doğrulama verilerinin önceden işleme işlemlerini yapmak için kullanılacak.
- preprocessing_function = preprocess_input: Görüntü ön işleme işlevi, preprocess_input olarak belirtilir. Bu işlev, doğrulama verilerinin önceden işlenmesi için kullanılır.

```python
train_generator = train_datagen.flow_from_dataframe(
    dataframe = train_data,
    directory = FILES + "/train/",
    x_col = "file",
    y_col = "label",
    class_mode = "categorical",
    target_size = (224, 224),
    batch_size = batch_size,
    seed = 666,
)

val_generator = val_datagen.flow_from_dataframe(
    dataframe = val_data,
    directory = FILES + "/train/",
    x_col = "file",
    y_col = "label",
    class_mode = "categorical",
    target_size = (224, 224),
    batch_size = batch_size,
    seed = 666,
    shuffle = False
```

Bu kod parçası, eğitim ve doğrulama veri setlerini ImageDataGenerator nesneleri aracılığıyla yüklemek için kullanılır. Veriler DataFrame'den alınır ve görüntülerin dosya yolları, etiketleri ve diğer önemli parametreler bu nesnelere sağlanır. Bu veri akışları daha sonra modelin eğitim ve doğrulama aşamalarında kullanılır.

1.)

- train_generator: Eğitim veri akışı için bir ImageDataGenerator nesnesi oluşturulur. Bu nesne, eğitim verilerini yüklemek ve veri artırma işlemlerini uygulamak için kullanılacak.
- flow_from_dataframe: ImageDataGenerator'ın bir yöntemidir ve verileri bir DataFrame'den almak için kullanılır.
- dataframe = train_data: DataFrame, eğitim veri setini temsil eder. Bu DataFrame, görüntülerin dosya yollarını ve etiketlerini içermelidir.
- directory = FILES + "/train/": Görüntülerin bulunduğu dizini belirtir. FILES değişkeni projenin dosya dizinini temsil eder.

- x_col = "file": DataFrame'deki sütun adını belirtir, bu sütun görüntü dosya yollarını içermelidir.
- y_col = "label": DataFrame'deki sütun adını belirtir, bu sütun etiketleri içermelidir.
- class_mode = "categorical": Sınıf modu "categorical" olarak ayarlanır, çünkü çok sınıflı sınıflandırma problemini ele alıyoruz.

- target_size = (224, 224): Görüntülerin hedef boyutunu belirtir, burada 224x224 piksel olarak ayarlanmıştır.
- batch_size = batch_size: Batch boyutunu belirtir, burada batch_size değişkenine atanan değeri kullanır.
- seed = 666: Rastgelelik için kullanılan çekirdek değerini belirtir.

2.)

- val_generator: Doğrulama veri akışı için bir ImageDataGenerator nesnesi oluşturulur. Bu nesne, doğrulama verilerini yüklemek ve görüntü ön işleme işlemlerini uygulamak için kullanılacak.
- flow_from_dataframe: ImageDataGenerator'ın bir yöntemidir ve verileri bir DataFrame'den almak için kullanılır.
- dataframe = val_data: DataFrame, doğrulama veri setini temsil eder. Bu DataFrame, görüntülerin dosya yollarını ve etiketlerini içermelidir.
- directory = FILES + "/train/": Görüntülerin bulunduğu dizini belirtir. FILES değişkeni projenin dosya dizinini temsil eder.
- x_col = "file": DataFrame'deki sütun adını belirtir, bu sütun görüntü dosya yollarını içermelidir.
- y_col = "label": DataFrame'deki sütun adını belirtir, bu sütun etiketleri içermelidir.
- class_mode = "categorical": Sınıf modu "categorical" olarak ayarlanır, çünkü çok sınıflı sınıflandırma problemini ele alıyoruz.
- target_size = (224, 224): Görüntülerin hedef boyutunu belirtir, burada 224x224 piksel olarak ayarlanmıştır.
- batch_size = batch_size: Batch boyutunu belirtir, burada batch_size değişkenine atanan değeri kullanır.
- seed = 666: Rastgelelik için kullanılan çekirdek değerini belirtir.
- shuffle = False: Veri akışının karıştırılmasını devre dışı bırakır. Doğrulama sırasında karıştırma gereksiz olduğu için bu değer False olarak ayarlanır.

```
Found 12000 validated image filenames belonging to 2 classes.
Found 3000 validated image filenames belonging to 2 classes.
```

"Found 12000 validated image filenames belonging to 2 classes.": Bu çıktı, eğitim veri akışının doğrulama işleminden geçirildiğini ve toplam 12000 görüntü dosyasının doğrulandığını göstermektedir.

"Found 3000 validated image filenames belonging to 2 classes.": Bu çıktı, doğrulama veri akışının doğrulama işleminden geçirildiğini ve toplam 3000 görüntü dosyasının doğrulandığını göstermektedir.

5. MODEL OLUŞTURMA

Bu kod parçası, VGG16 modelini önceden eğitilmiş ağırlıklarla yükleyerek, modelin dondurulmasını sağlar ve ardından yeni bir model oluşturur. Bu yeni model, önceden eğitilmiş VGG16 modeline birkaç tam bağlı katman ve dropout katmanları ekleyerek özelleştirilir. Böylece, önceden eğitilmiş VGG16 modelinin özellik çıkarma yetenekleri kullanılarak yeni bir sınıflandırma modeli oluşturulmuş olur.

- base_model = VGG16(weights="imagenet", input_shape=(224, 224, 3), include_top=False): VGG16 modeli, ImageNet veri kümesi üzerinde önceden eğitilmiş ağırlıkları ile birlikte yüklenir. input_shape parametresi, modele giriş olarak beklenen görüntü boyutunu (224x224 piksel) belirtir. include_top=False parametresi, modelin tamamen bağlı (fully connected) katmanlarını içermemesini sağlar.
- for layers in base_model.layers: layers.trainable = False: base_model içindeki tüm katmanların eğitilebilirliği (trainable) devre dışı bırakılır. Bu, VGG16 modelinin önceden eğitilmiş ağırlıklarını dondurarak, bu ağırlıkların eğitim sırasında güncellenmesini engeller.
- def vgg16_pretrained():: vgg16_pretrained adında bir fonksiyon tanımlanır.
- model = Sequential([...]): model isimli bir Sequential modeli oluşturulur. Bu model, VGG16 modelini, yeni eklenen katmanlarla birleştirir.
- base_model: Daha önce oluşturulan base_model VGG16 modelini temsil eder. Bu, önceden eğitilmiş ağırlıklara sahip VGG16 modelini içerir.

- GlobalAveragePooling2D(): Global ortalama havuzlama katmanı, özellik haritalarının boyutunu düşürür ve verileri daha düz bir formata getirir.
- Dense(100, activation="relu"): 100 nöronlu tam bağlı (fully connected) katman, girişleri 100 boyutlu bir vektöre dönüştürür ve ReLU aktivasyon fonksiyonunu uygular.

- Dropout(0.4): %40 dropout oranı ile dropout katmanı, aşırı uyum (overfitting) riskini azaltır.
- Dense(64, activation="relu"): 64 nöronlu bir tam bağlı katman daha eklenir ve ReLU aktivasyon fonksiyonu uygulanır.
- Dense(2, activation="softmax"): 2 sınıflı (binary) çıkış katmanı eklenir ve softmax aktivasyon fonksiyonu kullanılarak sınıflandırma yapılır.
- return model: Oluşturulan model döndürülür.
- tf.keras.backend.clear_session(): Keras arka planını temizler ve önceki tanımlanmış modelleri temizler. Bu, bellek kullanımını optimize etmeye yardımcı olur ve daha fazla hafıza tüketimini önler.

```
model = vgg16_pretrained()

model.compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = "accuracy")

model.summary()
```

Bu kod parçası, VGG16 modeline dayanan bir sinir ağı modeli oluşturur, modeli derler ve son olarak modelin özetini ekrana yazdırır.

- model = vgg16_pretrained() ifadesi, vgg16_pretrained() fonksiyonunu çağırarak önceden tanımlanmış VGG16 modeline dayanan bir model oluşturur. Bu model, VGG16 modelinin özellik çıkarma katmanlarını içerir ve üzerine eklenen tam bağlı katmanlarla özelleştirilir.
- model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"]) ifadesi, modelin derlenmesini yapar. Bu adımda, kayıp fonksiyonu (loss) olarak kategorik çapraz entropi kullanılır, optimizasyon algoritması (optimizer) olarak "adam" seçilir ve doğruluk (accuracy) metriği kullanılır. Derleme işlemi, modelin eğitim aşamasında kullanılacak optimizasyon stratejisini belirler.
- model.summary() ifadesi, oluşturulan modelin özetini yazdırır. Bu özet, modelin katmanlarını, her katmandaki parametre sayısını ve toplam model parametre sayısını gösterir. Ayrıca, her katmandaki çıktı şekillerini ve toplam parametre sayısını da görüntüler.

```python
reduce_lr = ReduceLROnPlateau(
    monitor = "val_accuracy",
    patience = 2,
    verbose = 1,
    factor = 0.5,
    min_lr = 0.000000001
)

early_stopping = EarlyStopping(
    monitor = "val_accuracy",
    patience = 5,
    verbose = 1,
    mode = "max",
)

checkpoint = ModelCheckpoint(
    monitor = "val_accuracy",
    filepath = "catdog_vgg16_.{epoch:02d}-{val_accuracy:.6f}.hdf5",
    verbose = 1,
    save_best_only = True,
    save_weights_only = True
)
```

Bu geri dönüt(callbacks), eğitim sırasında modelin performansını izlemek, öğrenme oranını azaltmak, erken durdurma yapmak ve en iyi modeli kaydetmek için kullanılır.

ReduceLROnPlateau:

- monitor: İzlenen metrik. Burada, doğrulama doğruluğunu (val_accuracy) izler.
- patience: Metriğin iyileşme göstermediği epoch sayısı. Eğer belirtilen sayıda epoch boyunca iyileşme olmazsa, öğrenme oranı azaltılır.
- verbose: Ayrıntıları görüntüler. 0: sessiz, 1: ilerlemeyi görüntüler.
- factor: Öğrenme oranını azaltma faktörü. Öğrenme oranı, factor değeriyle çarpılır.
- min_lr: Minimum öğrenme oranı sınırı.
- Bu geri dönüt, doğrulama doğruluğu iyileşmezse öğrenme oranını azaltır. Bu, modelin daha iyi sonuçlar elde etmek için daha yavaş bir şekilde öğrenmesine yardımcı olur.

EarlyStopping:

- monitor: İzlenen metrik. Burada, doğrulama doğruluğunu (val_accuracy) izler.
- patience: Metriğin iyileşme göstermediği epoch sayısı. Eğer belirtilen sayıda epoch boyunca iyileşme olmazsa, eğitim erken durdurulur.
- verbose: Ayrıntıları görüntüler. 0: sessiz, 1: ilerlemeyi görüntüler.
- mode: İzlenen metriğin maksimize edilmesi (max) veya minimize edilmesi (min) gerektiğini belirtir.
- Bu geri dönüt, doğrulama doğruluğu iyileşmezse eğitimi erken durdurur. Bu, aşırı uyum (overfitting) durumunda aşırı eğitimi önlemek için kullanılır.

ModelCheckpoint:

- monitor: İzlenen metrik. Burada, doğrulama doğruluğunu (val_accuracy) izler.
- filepath: Model ağırlıklarının kaydedileceği dosya yolu ve ismi. Burada, kaydedilen dosyaların adında epoch numarası ve doğruluk değeri yer alır.
- verbose: Ayrıntıları görüntüler. 0: sessiz, 1: ilerlemeyi görüntüler.
- save_best_only: Sadece en iyi performans gösteren model ağırlıklarının kaydedilmesini sağlar.
- save_weights_only: Sadece model ağırlıklarının kaydedilmesini sağlar, modelin kendisi değil.

6. EĞİTİM VE SONUÇLAR

Bu adımda, oluşturulan model eğitilmiş ve performansı değerlendirilmiştir. İlgili adımlar aşağıda detaylı olarak açıklanmıştır.

```
history = model.fit(
    train_generator,
    epochs = 10,
    validation_data = val_generator,
    validation_steps = val_data.shape[0] // batch_size,
    steps_per_epoch = train_data.shape[0] // batch_size,
    callbacks = [reduce_lr, early_stopping, checkpoint]
)
```

Bu kod parçası, modelin belirtilen sayıda epoch için eğitilmesini sağlar. Eğitim ilerledikçe, kayıp fonksiyonu ve doğruluk metriklerini izler ve geri dönüt işlevleri tarafından tanımlanan aksiyonları gerçekleştirir. Eğitim tamamlandığında, eğitim geçmişi (history) değişkeninde kaydedilen metrik değerlerine erişebilirsiniz.

- train_generator: Eğitim veri setini oluşturan veri üreteci (data generator) nesnesi. Bu veri üreteci, eğitim veri setini bölerek her bir adımda modelin bir yığın (batch) veriyi işlemesini sağlar.
- epochs: Eğitim döngüsünün kaç kez tamamlanacağını belirten parametre. Burada, 10 epoch için eğitim yapılacaktır.
- validation_data: Doğrulama veri setini oluşturan veri üreteci nesnesi. Bu veri üreteci, doğrulama veri setini bölerek modelin her epoch sonunda doğrulama performansını değerlendirmesini sağlar.
- validation_steps: Doğrulama veri setindeki adım sayısı. Burada, doğrulama veri setinin boyutuna dayanarak hesaplanmıştır.
- steps_per_epoch: Her bir epoch için adım sayısı. Burada, eğitim veri setinin boyutuna dayanarak hesaplanmıştır.
- callbacks: Geri dönüt (callback) işlevlerinin listesi. Burada, önceden tanımladığımız reduce_lr, early_stopping ve checkpoint geri dönütleri kullanılmıştır. Bu geri dönütler,

eğitim sırasında belirli olaylara (örneğin, öğrenme oranını azaltma, erken durdurma, en iyi modeli kaydetme) tepki vererek ek işlemler gerçekleştirir.

```
Epoch 1/10
93/93 [==============================] - ETA: 0s - loss: 0.2630 - accuracy: 0.9226
Epoch 1: val_accuracy improved from -inf to 0.97520, saving model to catdog_vgg16_.01-0.975204.hdf5
93/93 [==============================] - 249s 2s/step - loss: 0.2630 - accuracy: 0.9226 - val_loss: 0.0577 - val_accuracy: 0.9752 - lr: 0.0010
Epoch 2/10
93/93 [==============================] - ETA: 0s - loss: 0.0891 - accuracy: 0.9675
Epoch 2: val_accuracy improved from 0.97520 to 0.98370, saving model to catdog_vgg16_.02-0.983696.hdf5
93/93 [==============================] - 214s 2s/step - loss: 0.0891 - accuracy: 0.9675 - val_loss: 0.0443 - val_accuracy: 0.9837 - lr: 0.0010
Epoch 3/10
93/93 [==============================] - ETA: 0s - loss: 0.0780 - accuracy: 0.9704
Epoch 3: val_accuracy did not improve from 0.98370
93/93 [==============================] - 219s 2s/step - loss: 0.0780 - accuracy: 0.9704 - val_loss: 0.0484 - val_accuracy: 0.9806 - lr: 0.0010
Epoch 4/10
93/93 [==============================] - ETA: 0s - loss: 0.0592 - accuracy: 0.9768
Epoch 4: val_accuracy improved from 0.98370 to 0.98743, saving model to catdog_vgg16_.04-0.987432.hdf5
93/93 [==============================] - 217s 2s/step - loss: 0.0592 - accuracy: 0.9768 - val_loss: 0.0377 - val_accuracy: 0.9874 - lr: 0.0010
Epoch 5/10
93/93 [==============================] - ETA: 0s - loss: 0.0540 - accuracy: 0.9793
Epoch 5: val_accuracy did not improve from 0.98743
93/93 [==============================] - 223s 2s/step - loss: 0.0540 - accuracy: 0.9793 - val_loss: 0.0392 - val_accuracy: 0.9864 - lr: 0.0010
Epoch 6/10
93/93 [==============================] - ETA: 0s - loss: 0.0472 - accuracy: 0.9819
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.

Epoch 6: val_accuracy did not improve from 0.98743
93/93 [==============================] - 239s 3s/step - loss: 0.0472 - accuracy: 0.9819 - val_loss: 0.0388 - val_accuracy: 0.9864 - lr: 0.0010
Epoch 7/10
93/93 [==============================] - ETA: 0s - loss: 0.0397 - accuracy: 0.9833
Epoch 7: val_accuracy improved from 0.98743 to 0.98811, saving model to catdog_vgg16_.07-0.988111.hdf5
93/93 [==============================] - 215s 2s/step - loss: 0.0397 - accuracy: 0.9833 - val_loss: 0.0363 - val_accuracy: 0.9881 - lr: 5.0000e-04
Epoch 8/10
93/93 [==============================] - ETA: 0s - loss: 0.0388 - accuracy: 0.9864
Epoch 8: val_accuracy did not improve from 0.98811
93/93 [==============================] - 217s 2s/step - loss: 0.0388 - accuracy: 0.9864 - val_loss: 0.0378 - val_accuracy: 0.9878 - lr: 5.0000e-04
Epoch 9/10
93/93 [==============================] - ETA: 0s - loss: 0.0354 - accuracy: 0.9870
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.

Epoch 9: val_accuracy did not improve from 0.98811
93/93 [==============================] - 214s 2s/step - loss: 0.0354 - accuracy: 0.9870 - val_loss: 0.0398 - val_accuracy: 0.9864 - lr: 5.0000e-04
Epoch 10/10
93/93 [==============================] - ETA: 0s - loss: 0.0330 - accuracy: 0.9882
Epoch 10: val_accuracy did not improve from 0.98811
93/93 [==============================] - 235s 3s/step - loss: 0.0330 - accuracy: 0.9882 - val_loss: 0.0368 - val_accuracy: 0.9878 - lr: 2.5000e-04
```

Bu çıktılar, modelin eğitim sürecini gösterir ve her bir epoch için eğitim ve doğrulama performansını raporlar. Çıktıların açıklaması:

- Epoch 1/10: Toplam 10 epoch döngüsü olduğunu belirtir.
- 93/93 [==============================]: Her bir epoch için toplam 93 adımın olduğunu gösterir.
- ETA: 0s: Tahmini tamamlanma süresini belirtir.
- loss: 0.2630: Eğitim kaybının 0.2630 olduğunu gösterir.
- accuracy: 0.9226: Eğitim doğruluğunun 0.9226 olduğunu gösterir.
- val_loss: 0.0577: Doğrulama kaybının 0.0577 olduğunu gösterir.
- val_accuracy: 0.9752: Doğrulama doğruluğunun 0.9752 olduğunu gösterir.
- lr: 0.0010: Öğrenme oranının 0.0010 olduğunu gösterir.

- Bu bilgiler, her bir epoch sonunda eğitim ve doğrulama metriklerini raporlar. Ayrıca, saving model to... ifadesi, en iyi doğrulama doğruluğuna sahip olan modelin kaydedildiğini gösterir. Örneğin, catdog_vgg16_.01-0.975204.hdf5 dosyası, ilk epoch sonunda elde edilen en iyi modeli temsil eder.

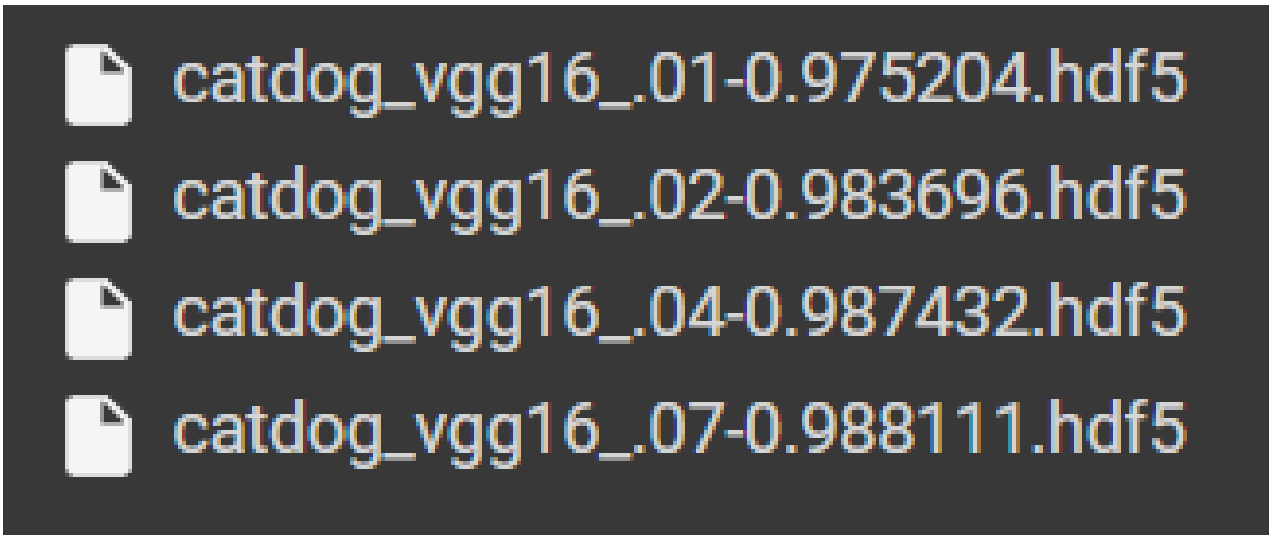
Ayrıca bu çıktılardan, ReduceLROnPlateau ve EarlyStopping , geri dönüt işlevlerinin öğrenme oranını azaltma ve erken durdurma işlemlerini gerçekleştirdiğini gösterir.

```
1 tf.keras.backend.clear_session()
2
3 model = vgg16_pretrained()
4
5 model.load_weights("/content/catdog_vgg16_.07-0.988111.hdf5")
```

Bu kod parçası, temiz bir Keras oturumu oluşturur, VGG16 modelini tanımlar ve daha sonra belirtilen dosya yolundan modelin ağırlıklarını yükler. Böylece, modelin daha önce eğitilmiş ağırlıklarını kullanarak tahminler yapabilirsiniz.

- tf.keras.backend.clear_session(): Önceden tanımlanmış tüm Keras modellerini, grafikleri ve değişkenleri temizler. Bu, bir TensorFlow oturumunun durumunu sıfırlar ve bellekteki gereksiz öğeleri temizler.
- model = vgg16_pretrained(): Önceden tanımlanmış bir vgg16_pretrained fonksiyonunu çağırarak VGG16 modelini oluşturur. Bu fonksiyon, VGG16 modelini ve ardışık katmanları içeren bir Keras Model nesnesini döndürür.
- model.load_weights("/content/catdog_vgg16_.07-0.988111.hdf5"): Belirtilen dosya yolundan (örneğin "/content/catdog_vgg16_.07-0.988111.hdf5") modelin ağırlıklarını yükler. Bu, daha önce kaydedilmiş bir modelin ağırlıklarını geri yüklemek için kullanılır. Bu örnekte, "catdog_vgg16_.07-0.988111.hdf5" dosyasından modelin ağırlıkları yüklenir.

Aşağıdaki görselde ise tüm epoch adımlarında en yüksek doğruluk oranına ulaşan epochları gösterip modelin ağırlıklarını kaydeder.



7. SONUÇLAR VE GRAFİKLER

Bu adımda, oluşturulan model eğitilmiş ve performansı değerlendirilmiştir. İlgili adımlar aşağıda detaylı olarak açıklanmıştır.

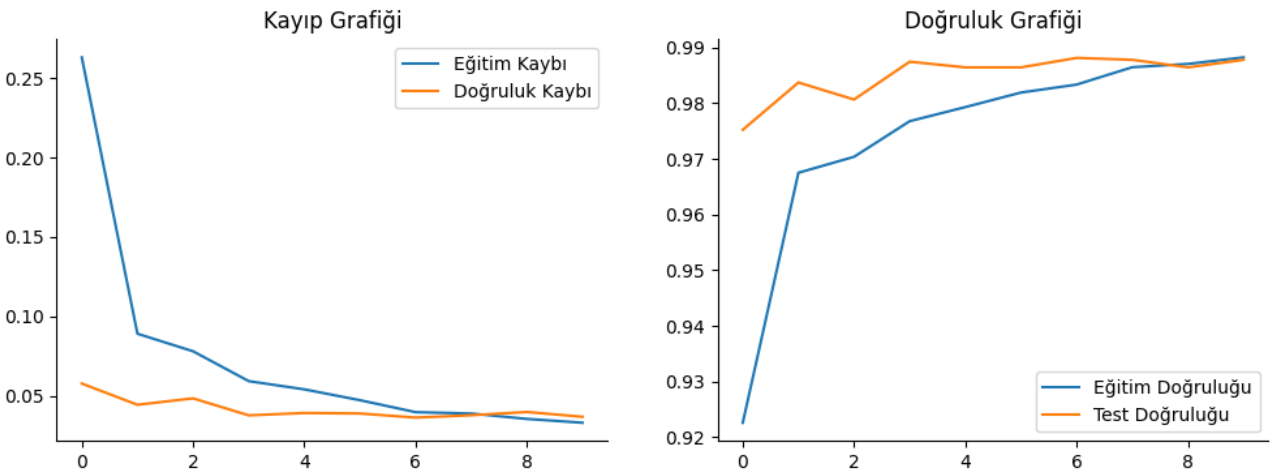
```
fig, axes = plt.subplots(1, 2, figsize = (12, 4))

sns.lineplot(x = range(len(history.history["loss"])), y = history.history["loss"], ax = axes[0], label = "Eğitim Kaybı")
sns.lineplot(x = range(len(history.history["loss"])), y = history.history["val_loss"], ax = axes[0], label = "Doğruluk Kaybı")

sns.lineplot(x = range(len(history.history["accuracy"])), y = history.history["accuracy"], ax = axes[1], label = "Eğitim Doğruluğu")
sns.lineplot(x = range(len(history.history["accuracy"])), y = history.history["val_accuracy"], ax = axes[1], label = "Test Doğruluğu")
axes[0].set_title("Kayıp Grafiği"); axes[1].set_title("Doğruluk Grafiği")

sns.despine()
plt.show()
```

Bu kod parçası, eğitim geçmişinin kayıp ve doğruluk değerlerini gösteren bir çizim oluşturur. Sol tarafta eğitim kaybı ve doğruluk kaybı grafiği yer alırken, sağ tarafta eğitim doğruluğu ve test doğruluğu grafiği bulunur.



Bu grafiğe bakarak, eğitim sürecinin ilerlemesiyle kayıp değerlerinin azaldığını ve doğruluk değerlerinin arttığını gözlemleyebilirsiniz. Bu grafikler, modelin eğitim ve doğrulama veri setleri üzerindeki performansını görsel olarak değerlendirmemizi sağlar.

```
val_pred = model.predict(val_generator, steps = np.ceil(val_data.shape[0] / batch_size))
val_data.loc[:, "val_pred"] = np.argmax(val_pred, axis = 1)

labels = dict((v, k) for k, v in val_generator.class_indices.items())

val_data.loc[:, "val_pred"] = val_data.loc[:, "val_pred"].map(labels)
```

```
24/24 [==============================] - 25s 1s/step
```

Bu kod parçası, eğitilmiş modelin doğrulama veri kümesi üzerindeki tahminlerini gerçekleştirir ve tahmin sonuçlarını "val_pred" sütunu olarak val_data DataFrame'ine ekler. Böylece, her bir görüntünün tahmin edilen sınıf etiketi veri setinde tutulur.

- val_pred = model.predict(val_generator, steps=np.ceil(val_data.shape[0] / batch_size)): val_generator üzerinde modelin tahmin yapmasını sağlar. model.predict() işlevi, doğrulama veri kümesi üzerindeki görüntülerin sınıflarını tahmin eder. Tahmin sonuçları val_pred değişkenine atanır. steps parametresi, toplam adım sayısını belirtir ve np.ceil(val_data.shape[0] / batch_size) ifadesi, doğrulama veri kümesindeki görüntülerin kaç adımda tamamlanacağını hesaplar.

- val_data.loc[:, "val_pred"] = np.argmax(val_pred, axis=1): val_pred dizisindeki tahmin olasılıklarının en yüksek olan sınıf indekslerini seçer ve val_data DataFrame'ine "val_pred" sütunu olarak ekler. np.argmax() işlevi, tahmin olasılıkları dizisindeki en yüksek değeri bulur ve buna karşılık gelen sınıf indeksini döndürür.
- labels = dict((v, k) for k, v in val_generator.class_indices.items()): val_generator nesnesinin sınıf indekslerini ve etiketlerini içeren class_indices özniteliğini kullanarak, etiketlerin sınıf indekslerine karşılık geldiği bir sözlük oluşturur. Bu, tahminlerin sayısal değerler yerine etiketlerle ilişkilendirilmesini sağlar.
- val_data.loc[:, "val_pred"] = val_data.loc[:, "val_pred"].map(labels): "val_pred" sütunundaki tahminlerin sayısal değerlerini, labels sözlüğündeki etiketlere dönüştürür. map() işlevi, her bir tahmin değerini labels sözlüğüne göre eşleştirir.

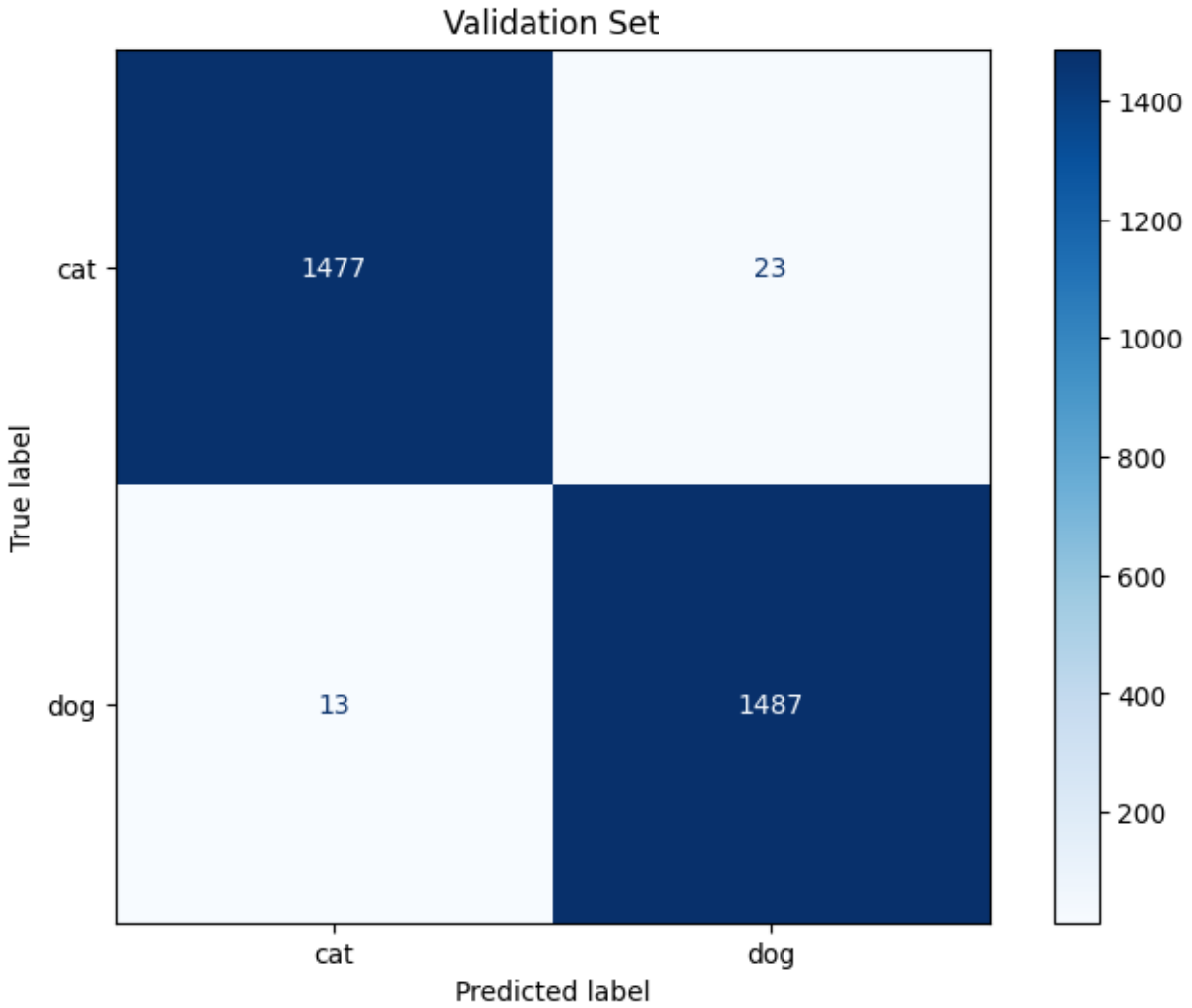
```python
fig, ax = plt.subplots(figsize = (9, 6))

cm = confusion_matrix(val_data["label"], val_data["val_pred"])

disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = ["cat", "dog"])
disp.plot(cmap = plt.cm.Blues, ax = ax)

ax.set_title("Validation Set")
plt.show()
```

Bu adımda modelin eğitimi artık gerçekleştirilmiş, eğitim ve doğrulama kayıp/accuracy değerlerinin grafiklerle görselleştirilmesi sağlanmış ve modelin performansı karmaşıklık matrisi ile değerlendirilmiştir.



Sonuç olarak, değerlerin açıklaması şu şekildedir:

- Sol üst (1477): "cat" olarak gerçek etikete sahip ve doğru bir şekilde "cat" olarak tahmin edilen örneklerin sayısı.
- Sol alt (13): "cat" olarak gerçek etikete sahip ancak yanlış bir şekilde "dog" olarak tahmin edilen örneklerin sayısı.
- Sağ üst (23): "dog" olarak gerçek etikete sahip ancak yanlış bir şekilde "cat" olarak tahmin edilen örneklerin sayısı.
- Sağ alt (1487): "dog" olarak gerçek etikete sahip ve doğru bir şekilde "dog" olarak tahmin edilen örneklerin sayısı.

8. HATA ANALİZİ

Bu bölümde hatalı tahmin edilen görüntüleri tespit edip çıktı halinde göreceğiz.

```
val_errors = val_data[(val_data.label) != (val_data.val_pred)].reset_index(drop = True)
val_errors
```

val_errors veri çerçevesi, modelinizin hatalı tahminler yaptığı örnekleri içerir ve bu örnekleri daha ayrıntılı bir şekilde analiz etmenize olanak sağlar. Bunun sonucunda tam 36 tane hatalı örnek çıkar Dosyaların isimleriyle birlikte gerçek değerler ve hatalı tahmin değerleri yazdırılır.

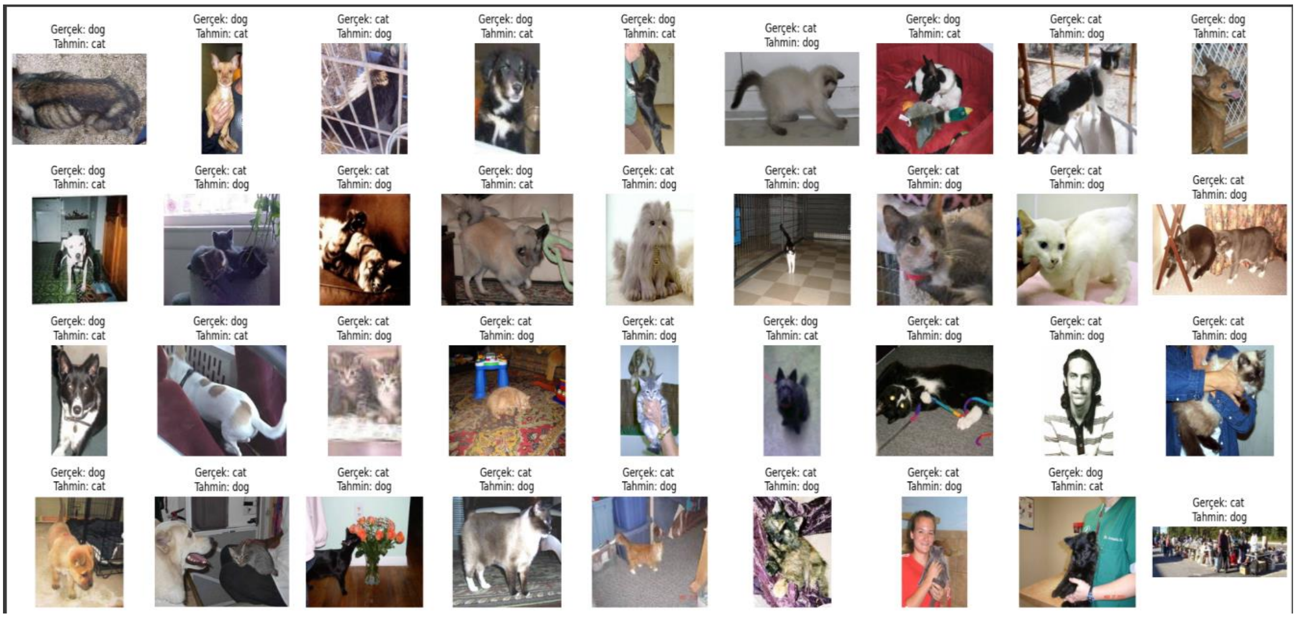
```
fig = plt.figure(1, figsize = (24, 20))

for i in range(36):

    plt.subplot(9, 9, i + 1)
    image = load_img("/content/drive/MyDrive/kedi_kopek_unzip/train/" + val_errors.file[i])
    plt.imshow(image)
    plt.axis("off")
    plt.title(f"Gerçek: {val_errors['label'][i]} \nTahmin: {val_errors['val_pred'][i]}")

plt.tight_layout()
plt.show()
```

Bu kod parçasında hatalı tahminlerin hangi görüntüler olduğu ekrana çıktı olarak bir tablo içerisinde verilir. Bu şekilde, hatalı tahminlerin görsel örnekleri bir altlık tablosu olarak görselleştirilir ve her bir hücrede gerçek ve tahmin edilen sınıf etiketleri görüntülenir. Bu, modelinizin hangi örneklerde hatalı tahminler yaptığını daha ayrıntılı bir şekilde analiz etmenize yardımcı olur.



Son olarak test veri seti üzerinde yeni tahminler yapıp işlemi bitiriyoruz.

```python
test_datagen = ImageDataGenerator(preprocessing_function = preprocess_input)

test_generator = test_datagen.flow_from_dataframe(
    dataframe = test_df,
    directory = FILES + "/test1/",
    x_col = "file",
    y_col = None,
    class_mode = None,
    target_size = (224, 224),
    batch_size = batch_size,
    seed = 666,
    shuffle = False
)
```

```
Found 7500 validated image filenames.
```

```python
test_preds = model.predict(test_generator, steps = np.ceil(test_df.shape[0] / batch_size))

test_df["test_preds"] = np.argmax(test_preds, axis = 1)
labels = dict((v,k) for k,v in train_generator.class_indices.items())

test_df['test_preds'] = test_df['test_preds'].map(labels)
```

```
59/59 [==============================] - 56s 943ms/step
```

```python
sample_test = test_df.sample(16).reset_index(drop = True)

fig = plt.figure(1, figsize = (24, 20))
fig.suptitle("Örnek Tahminler")

for i in range(len(sample_test)):

    plt.subplot(8, 8, i + 1)
    image = load_img("/content/drive/MyDrive/kedi_kopek_unzip/test1/" + sample_test.file[i])
    plt.imshow(image)
    plt.axis("off")
    plt.title(f"Tahmin :  {sample_test['test_preds'][i]}")

plt.tight_layout()
plt.show()
```