KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ



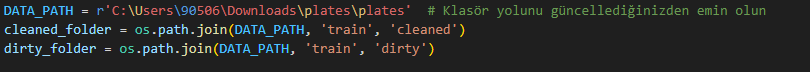
YÜKSEK DÜZEY PROGRAMLAMA PROJE RAPORU

KONU:DIGIT RECOGNIZER

HAZIRLAYAN:202013172001) EMİRHAN SOLAK

DANIŞMAN:DOÇ.DR.HASAN TEMURTAŞ

KODLAR:

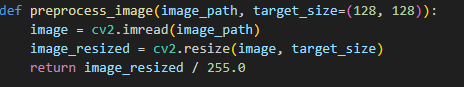


**DATA\_PATH değişkeni, Windows işletim sistemi altında bir dosya yolunu saklar. Bu yol, dosyaların yer aldığı ana klasörün yolunu belirtir.**

**r ifadesi, "raw string" anlamına gelir. Bu, backslash (\) karakterlerinin özel bir anlam taşımaması için kullanılır. Yani bu yazım tarzı, Python'da dosya yollarını doğru bir şekilde belirtmek için önemlidir.**

**os.path.join() fonksiyonu, farklı dizinlerin yollarını platform bağımsız bir şekilde birleştirir. Bu fonksiyon, dizinleri birleştirirken doğru dosya yolu ayırıcılarını (örneğin Windows için \, Unix tabanlı sistemler için /) kullanır.**

**cleaned\_folder ve dirty\_folder değişkenleri, train alt klasöründeki cleaned ve dirty adlı alt klasörlerin tam yolunu oluşturur.**



**cv2.imread(image\_path)**:

Bu fonksiyon, **OpenCV kütüphanesi** ile bir **görüntü dosyasını** okur.

**mage\_path**: Görüntü dosyasının tam yolu.

Bu fonksiyon, görüntüyü bir **NumPy dizisi** olarak okur, yani bir matris şeklinde.

**cv2.resize(image, target\_size)**:

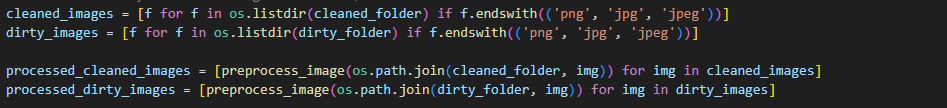
Bu fonksiyon, okunan görüntüyü **belirtilen boyutlara** yeniden boyutlandırır.

**target\_size**: Hedef boyut (width, height) şeklinde belirtilir. Bu durumda varsayılan olarak (128, 128) olarak ayarlanmış. Bu, görüntüyü 128x128 piksel boyutuna yeniden boyutlandırır.

**image\_resized / 255.0**:

Bu satır, **görüntüdeki piksel değerlerini normalize eder**. Görüntüdeki piksel değerleri 0 ile 255 arasında yer alır, ancak genellikle derin öğrenme modelleri için bu değerlerin **0 ile 1 arasında olması tercih edilir**.

Piksel değerleri **255'e bölünerek 0-1 aralığına çekilir**. Bu normalizasyon işlemi, modelin daha hızlı ve daha doğru öğrenmesini sağlar.



**cleaned\_folder** ve **dirty\_folder** klasörlerindeki **png, jpg ve jpeg uzantılı** görüntü dosyalarını okur, her birini **preprocess\_image** fonksiyonu ile **ön işler** ve işlenmiş görüntülerin listesini oluşturur. Bu işlem, genellikle derin öğrenme ve görüntü işleme projelerinde kullanılır, çünkü modelin doğru öğrenmesi için verilerin belirli bir formata getirilmesi gerekir (yeniden boyutlandırma, normalizasyon.



**ön işlenmiş** görüntüleri (hem **cleaned** hem de **dirty** klasörlerinden) **NumPy dizilerine** dönüştürür. NumPy dizileri, **derin öğrenme modelleri** için sıklıkla kullanılan bir veri yapısıdır, çünkü modeller bu formatı daha verimli şekilde işleyebilir. Bu dönüşüm, **görüntüleri model eğitimi** veya **tahminler** gibi işlemler için hazır hale getirir.



**cleaned** ve **dirty** görüntülerine karşılık gelen **etiket dizilerini** oluşturur. **y\_cleaned** dizisi, **temiz** görüntülerin etiketlerini içerirken **y\_dirty** dizisi, **kirli** görüntülerin etiketlerini içerir. Bu etiketler genellikle **denetimli öğrenme** algoritmalarında, örneğin **binary classification (ikili sınıflandırma)** problemlerinde kullanılır.

**y\_cleaned**: Temiz görüntüler için 0 etiketi,

**y\_dirty**: Kirli görüntüler için 1 etiketi içerir.



**X\_cleaned** ve **X\_dirty** görüntü dizilerini birleştirerek **X** adı altında tek bir veri kümesi oluşturur. Aynı şekilde, **y\_cleaned** ve **y\_dirty** etiketlerini birleştirerek **y** adlı tek bir etiket kümesi oluşturur.

**X**: Temiz ve kirli görüntülerden oluşan veri kümesi.

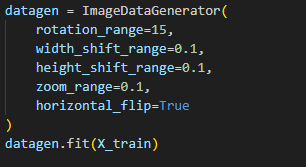
**y**: Bu görüntülerle ilişkili etiketler (temiz görüntüler için 0, kirli görüntüler için 1).



**X** ve **y** verilerini **%80 eğitim** ve **%20 doğrulama** kümelerine böler. Eğitim kümesi (**X\_train, y\_train**) modelin eğitiminde kullanılırken, doğrulama kümesi (**X\_val, y\_val**) modelin doğruluğunu değerlendirmek için kullanılır. Bu işlem, modelin **overfitting** (aşırı uyum sağlama) yapmasını engellemeye yardımcı olur ve modelin gerçek dünya verileriyle ne kadar iyi genelleme yapabileceğini test eder.

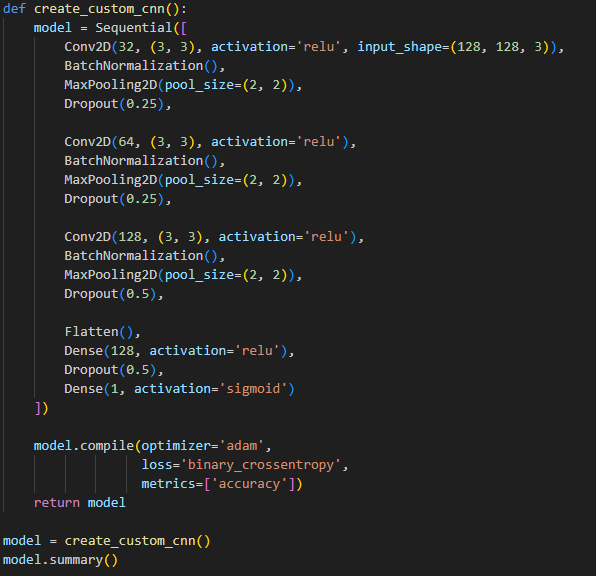


veri kümesindeki sınıf dengesizliklerini dengelemek için **sınıf ağırlıkları hesaplar**.



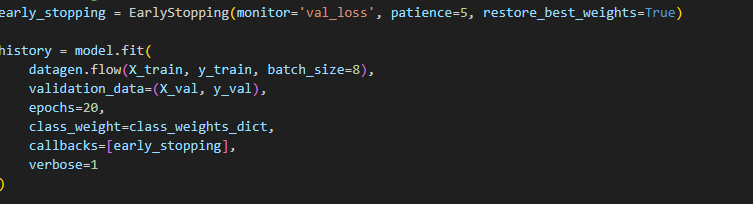
**veri artırma** işlemi için **ImageDataGenerator** sınıfını oluşturur. Parametreler sayesinde, eğitim görüntülerine çeşitli dönüşümler (dönme, kaydırma, yakınlaştırma, yatay çevirme) uygulanır. Bu, modelin daha çeşitli verilerle eğitilmesini ve daha genel bir model oluşturulmasını sağlar.

**datagen.fit(X\_train)** işlemi ile, veri artırma işlemi **eğitim verisi** üzerine yapılabilir. Bu işlemin ardından, modelin eğitiminde bu artırılmış veriler kullanılabilir.

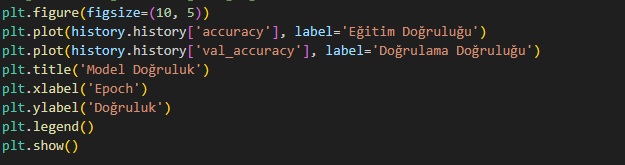


basit bir **Konvolüsyonel Sinir Ağı (CNN)** oluşturur ve bu ağ, **ikili sınıflandırma** görevine uygun şekilde yapılandırılmıştır.

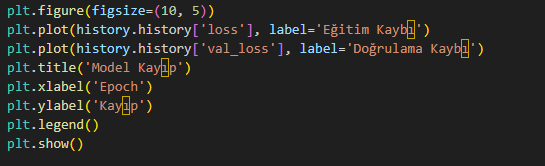
Model, **Adam** optimizasyon algoritmasıyla eğitilir ve **binary\_crossentropy** kayıp fonksiyonu ile ikili sınıflandırma yapılır.



modelin eğitim sürecinde **veri artırma (data augmentation)** kullanarak eğitim yapmasını sağlar. Ayrıca, **erken durdurma** (early stopping) stratejisini kullanarak modelin aşırı öğrenmesini (overfitting) engellemeye çalışır. Eğitim sırasında doğrulama kaybı izlenir ve belirli bir sayıda epoch boyunca kayıpta iyileşme olmazsa eğitim durdurulur.



modelin eğitim ve doğrulama doğruluğunu karşılaştırmak için bir çizgi grafiği oluşturur. Eğitim doğruluğu ve doğrulama doğruluğunun zamanla nasıl değiştiğini görmek, modelin öğrenme sürecini analiz etmede yardımcı olur.



**plt.figure(figsize=(10, 5))**:

Bu satır, grafik boyutlarını belirler. Grafik genişliği 10 birim, yüksekliği 5 birim olarak ayarlanmıştır. Bu boyutlar, grafiklerin okunabilirliğini artırır.

**plt.plot(history.history['loss'], label='Eğitim Kayb')**:

**history.history['loss']**: Eğitim sırasında her epoch'ta modelin kaybı (loss) değerlerini içerir. Bu, modelin eğitim verisi üzerindeki başarısızlık seviyesini gösterir.

**label='Eğitim Kayb'**: Bu etiketi kullanarak eğitim kaybını grafikte tanımlıyoruz. Bu etiket, grafikteki çizgi için gösterilecek adı belirtir.

**plt.plot(history.history['val\_loss'], label='Doğrulama Kayb')**:

**history.history['val\_loss']**: Modelin her epoch'ta doğrulama verisi üzerindeki kaybı (validation loss) değerlerini içerir. Bu, modelin daha önce görmediği doğrulama verisi üzerinde ne kadar başarılı olduğunu gösterir.

**label='Doğrulama Kayb'**: Bu etiket doğrulama kaybını grafikte tanımlar.

**plt.title('Model Kaybı')**:

Grafiğin başlığıdır. **"Model Kaybı"** başlığı, grafikte gösterilen verinin modelin kaybını temsil ettiğini belirtir.

**plt.xlabel('Epoch')**:

X ekseninde **epoch** sayısını gösterir. Her epoch bir eğitim adımını temsil eder.

**plt.ylabel('Kayb')**:

Y ekseninde modelin kaybı (loss) değerleri gösterilir. Bu, modelin eğitim ve doğrulama verileri üzerindeki başarısızlık seviyesini temsil eder.

**plt.legend()**:

**legend()** fonksiyonu, grafikteki çizgilerin hangi kategorilere ait olduğunu belirtmek için kullanılan bir açıklama (legend) ekler. Bu sayede, eğitim kaybı ve doğrulama kaybı arasındaki farkları kolayca anlayabilirsiniz.

**plt.show()**:

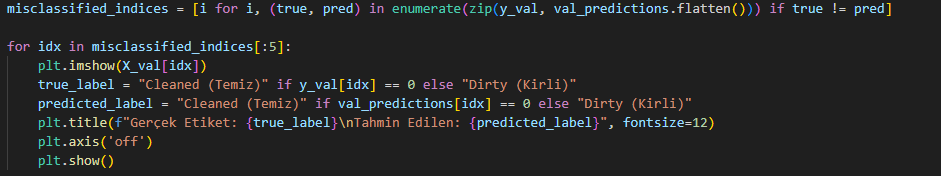
**show()** fonksiyonu, çizilen grafiği ekranda görüntüler.



modelin doğrulama seti üzerinde yaptığı **probabilite** tahminlerini, **0** veya **1** şeklinde **ikili sınıflandırma** sonuçlarına dönüştürür.

**Probabilite > 0.5**: Tahmin edilen sınıf **1** (pozitif).

**Probabilite ≤ 0.5**: Tahmin edilen sınıf **0** (negatif).



modelin yanlış sınıflandırdığı örneklerin ilk 5 tanesini görselleştirir. Görsellerin üzerine, her birinin **gerçek etiketi** ve **tahmin edilen etiketi** yazılır.



.h5 formatında kaydeder ve kaydın başarılı olduğuna dair bir mesaj verir.

ÇIKTILAR:

