T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BLM4540 - Görüntü İşleme

Proje

Tahir Can Özer – 17011061

tcanozerr@gmail.com

Dersin Yürütücüsü

Prof. Dr. Mine Elif Karslıgil

The Oxford-IIT Pet veri kümesinin analizi

Aşağıdaki dataset e ilişkin infonun yer aldığı görselde de görülebileceği gibi ilgili soruların cevapları maddeler halinde verilmiştir.

1. Veri kümesinde kaç farklı cins hayvan bulunduğu, her cinsten kaç örnek görüntü olduğunu eğitim ve test kümeleri için ayrı ayrı olacak şekilde tespit ediniz.

    - 37 farklı hayvan cinsi
    - Her cinsden yaklaşık 200 adet fotoğraf bulunmaktadır.

2. Eğitim ve test kümelerinde toplam örnek sayılarının tespitini yapınız.

    - Eğitim : 3680
    - Test: 3669

```
tfds.core.DatasetInfo(
    name='oxford_iiit_pet',
    version=3.2.0,
    description='The Oxford-IIIT pet dataset is a 37 category pet image dataset with roughly 200
images for each class. The images have large variations in scale, pose and
lighting. All images have an associated ground truth annotation of breed.',
    homepage='http://www.robots.ox.ac.uk/~vgg/data/pets/',
    features=FeaturesDict({
        'file_name': Text(shape=(), dtype=tf.string),
        'image': Image(shape=(None, None, 3), dtype=tf.uint8),
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=37),
        'segmentation_mask': Image(shape=(None, None, 1), dtype=tf.uint8),
        'species': ClassLabel(shape=(), dtype=tf.int64, num_classes=2),
    }),
    total_num_examples=7349,
    splits={
        'test': 3669,
        'train': 3680,
    },
    supervised_keys=('image', 'label'),
    citation="""@InProceedings{parkhi12a,
      author       = "Parkhi, O. M. and Vedaldi, A. and Zisserman, A. and Jawahar, C.~V.",
      title        = "Cats and Dogs",
      booktitle    = "IEEE Conference on Computer Vision and Pattern Recognition",
      year         = "2012",
    }""",
    redistribution_info=,
)
```

Yöntem

Tasarlanan modelin eğitimi için Google Colab ücretsiz üyeliği kullanılmıştır.

Konvülüsyon katman aktivasyon fonksiyonu : ReLu. Hızlı , kolay hesaplanabilir olması sebebiyle tercih edilmiştir.

Output katmanı aktivasyon fonksiyonu : Softmax. Output olarak 2 den fazla sınıf değerimiz olduğu için softmax tercih edilmiştir.

Optimizer : Adam. Farklı optimizerlar denenmiş ancak en iyi sonucu Adam vermiştir.

```
dataset, info = tfds.load('oxford_iiit_pet:3.*.*', with_info=True)
```

Tensorflow yardımıyla The Oxford-IIIT Pet dataseti indirilir ve dataset isimli objeye aktarılır.

```
def normalize(image, mask):
```

Eğitim ve test verisetindeki görselleri 0-1 arasına normalize etmeye yarayan metot.Ayrıca bölütleme maskeleri 1,2,3 olarak etiketlendiği için kolaylık sağlaması için bu değerleri 0,1,2 olarak değiştiriyoruz.

```
def resize(image, mask):
```

Görselleri ve maskeleri 128x128 boyutlarına çevirmeye yarayan metot.

```
train_dataset =dataset["train"].map(trainImage,num_parallel_calls=tf.data.AUTOTUNE)
test_dataset = dataset["test"].map(testImage, num_parallel_calls=tf.data.AUTOTUNE)
def trainImage(datapoint) ,def testImage(datapoint):
```

Her bir eğitim ve test görselini map yardımıyla normalize edip boyutlarını değiştirmeyi sağlayan metot.

```
epochNumber = 25
batchSize = 64
bufferSize = 1000
trainBatches =
train_dataset.cache().shuffle(bufferSize).batch(batchSize).repeat()
trainBatches =
trainBatches.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
validationBatches = test_dataset.take(3000).batch(batchSize)
testBatches = test_dataset.skip(3000).take(669).batch(batchSize)
```

Batch boyutunu , buffer boyutunu ayarlayıp eğitim için gerekli veriyi hazırlamaya yarayan kısım. Validation için 3000 görsel test içinse kalan 669 görsel ayrılmıştır.

```
def dice_coef(y_true, y_pred, smooth=1):
```

Dice Coefficient metriği için gerekli hesaplamaların yapıldığı fonksiyon.Parametre olarak ground truth data yı ve predicted data sonuçlarını alıp dice coefficientı hesaplar.

```
e1 = Conv2D(64, 3, activation='relu', padding='same')(inputs)
EN1 = MaxPooling2D(pool_size=(2, 2))(e1)
```

Görselden özellikleri çıkarmak için kullandığımız, filtre sayısı:64, filtre boyutları 3x3 olan encoder katmanımız.İlk katman olması sebebiyle input olarak dataseti alır. Konvülüsyon işlemi uygulayıp 2 stride için en büyük değerlerden oluşan matrisi EN1 e aktarır.

```
e2 = Conv2D(128, 3, activation='relu', padding='same')(EN1)
EN2 = MaxPooling2D(pool_size=(2, 2))(e2)
```

Bir önceki encoder katmanının sonucunu input olarak alan 128 filtreli 3x3 boyutunda bir konvülüsyon katmanı.

```
d4 = UpSampling2D(size=(2, 2))(EN4)
DC4 = Conv2D(512, 3, activation='relu', padding='same')(d4)
```

EN4 isimli encoder katmanından aldığı sonucun çözünürlüğünü arttırıp konvülüsyon uygulayan katman.

```
result1 = tf.keras.layers.concatenate([EN3, DC4])
```

Bir önceki decoder katmanından ve denk düşen encoder katmanından gelen sonucu toplamaya yarayan fonksiyon

```
d3 = UpSampling2D(size=(2, 2))(result1)
DC3 = Conv2D(256, 3, activation='relu', padding='same')(d3)
```

Input olarak denk düşen encoder katmanından ve bir önceki decoder katmanından sonuçları alan konvülüsyon katmanı.

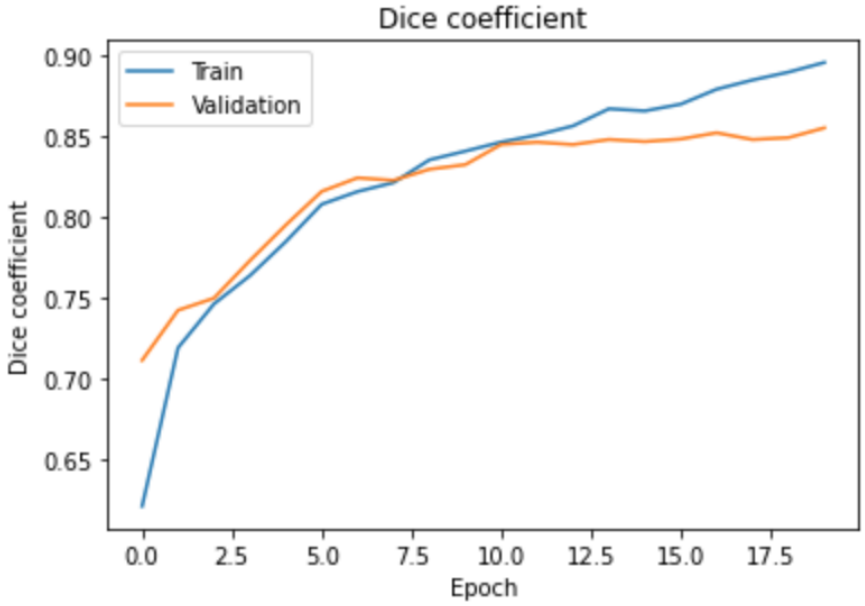
```
output = Conv2D(3, 3, activation='softmax', padding='same')(DC1)
```

Veri setindeki output katmanı. Predicted maskta 3 sınıf olmasından dolayı output katmanında 3 sınıf vardır. Bu sınıflar :
1: Foreground
2: Background
3: Not classified

Yapılan hiper parametre denemeleri sonucu en başarılı model : **def linkNet512()** olmuştur. İlgili model 20 epoch boyunca eğitilmiş olup başarım sonuçları aşağıdaki gibidir.

Epoch 20/20
57/57 [==============================] - 35s 625ms/step - loss: 0.2650 - accuracy: 0.8950 - dice_coef: 0.8952 - val_loss: 0.4097 - val_accuracy: 0.8552 - val_dice_coef: 0.8548

Uygulama



Dice Coefficient metriği modelin performansını ölçmek için kullanılır. Geliştiren modelde ise epoch sayısı arttıkça modelin başarım oranının arttığı gözlenebilmektedir. Ancak 10 epoch sonra modelin giderek overfit olduğu da göze çarpmaktadır.

Sonuç

Derin öğrenme ağları kullanılarak yapılan görüntü bölütleme işlemleri daha doğru sonuçlar vermektedir. Derin öğrenme süresince birden çok katmandan geçirildiği için daha iyi optimize olabilmektedirler. Ek olarak mevcut derin öğrenme modellerinin gerçek zamanlı olarak çalışabilmeleri sayesinde farklı uygulamalarla birlikte kullanılabilmesine olanak sağlamaktadır.

Geliştirmiş olduğum modelde:

Epoch 20/20
57/57 [==============================] - 35s 625ms/step - loss: 0.2650 - accuracy: 0.8950 - dice_coef: 0.8952 - val_loss: 0.4097 - val_accuracy: 0.8552 - val_dice_coef: 0.8548

%85 başarım oranına sahiptir.