
Beykent Üniversitesi

Yazılım Mühendisliği

Veri Madenciliği Projesi

# Görseller Üzerinden Tahmin Üretme & Image Captioning

Boğaç Küçük
Kutay Can Kaynak

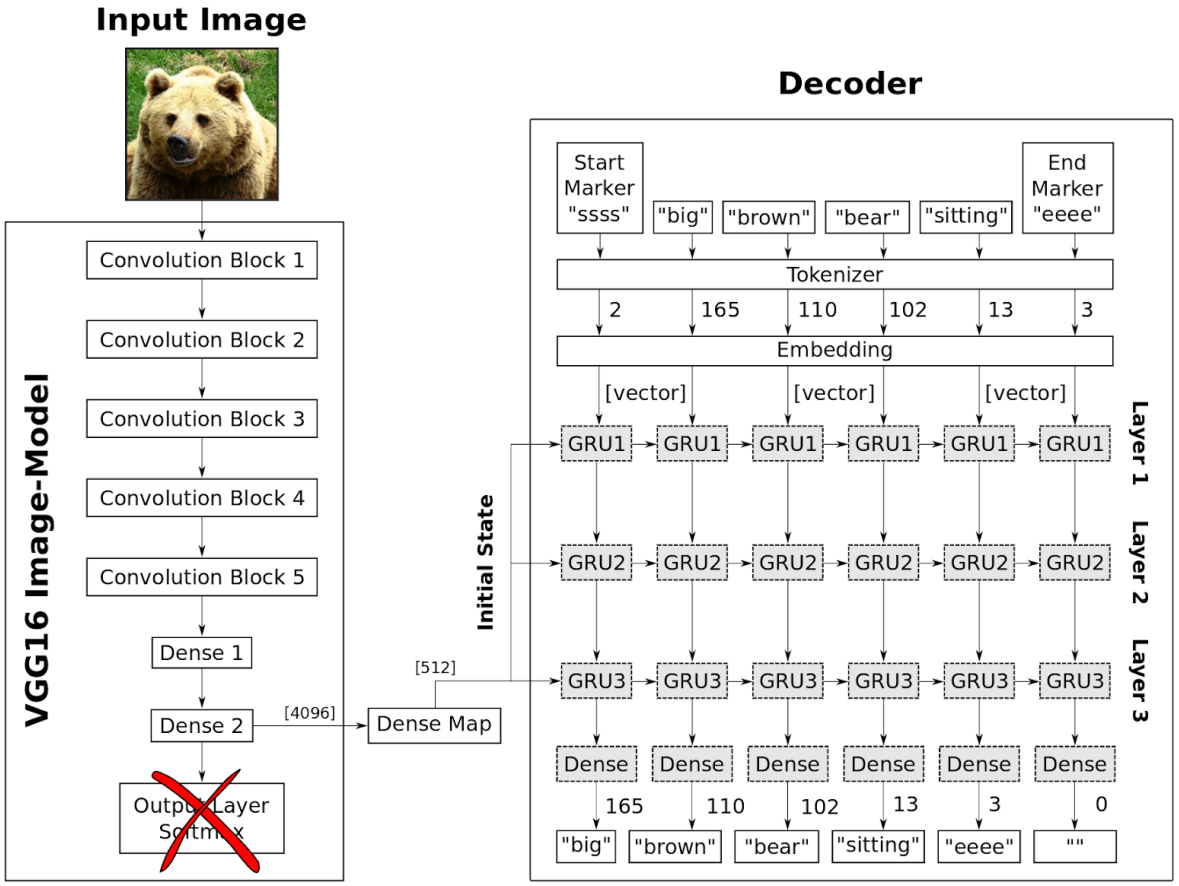
# İçindekiler

# Giriş

Image Captioning bilgisayar görü (computer vision) ve doğal dil işleme (natural language processing) konularının karması bir konudur. Bir yapay sinir ağına girdi olarak bir resim verilerek çıktı olarak bir metin elde edilmesi hedeflenir.

Geliştirilen projede konvolüsyonel sinir ağları kullanılmıştır. Model olarak hazır olarak eğitilmiş olan VGG16 modeli kullanılmıştır. Bunun sebebi görüntü işleme yapacak olan bilgisayarların üst düzey donanımlara sahip olması gerekmektedir. Son kullanıcı bilgisayarları bu eğitim süreci için yetersiz kalmaktadır. Bu yüzden daha önceden eğitimli olan bir model tercih edilmiştir.

VGG16 modelinin temel yapısından farklı olarak hazır modelin son katmanı çıkarıldı. Amaç bir düşünce vektörü üretmek olduğundan son katman ihtiyaca göre değiştirilmiştir. Standart VGG16 modelinin son katmanında normalde sınıflandırma yapılmaktadır. Yani model resmin içerisinde hangi nesneler olduğuna dair bir tahmin yürütüyor. Bu tip bir tahmine ilk aşamada ihtiyaç duyulmamaktadır. Bu sebepten dolayı modelden çıktı katmanını çıkarılmıştır. Resmin içeriği düşünce vektörüne yazılmakta bu sayede resim ondalıklı sayılara indirgenmiş olmaktadır. Bu aşamadan sonra üretilen düşünce vektörü ve resmin içeriğini anlatan cümleyi (caption) decoder'a girdi olarak verilecek. Decoder'a verilecek cümlelerde başlangıç ve bitiş tokenleri bulunmaktadır. Decoder output olarak kelimeler üretecek ve bu kelimeler gerçekte olması gereken kelimelerle karşılaştırılacaktır. Bu karşılaştırma sonucu bir loss değeri üretilip model optimize edilecektir. Eğitim tamamlandıktan sonra verilecek inputlar değişecek. Bir resim girdi olarak verilecek , resim VGG16 ağından geçirilecek ve bir düşünce vektörü üretilecek. Decoder'a bu düşünce vektörünü ve başlangıç tokeni verilecek yani girdi olarak bir caption verilmemektedir. Model aldığı bu inputlarla bir kelime üretecek ve bir sonraki aşamada üretilen bu kelime yeni input olacak ve tekra bir kelime üretecek. Modelin temel yapısı şu şekilde özetlenebilir:




---

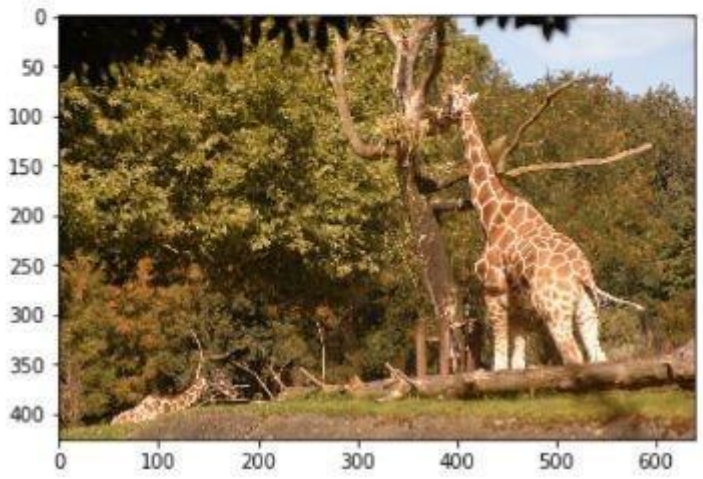
**Bu projede Coco veri seti kullanılmıştır. Kullanılan veri seti boyutu yaklaşık 25 GB kadar olduğundan iletilen dosyanın içinde yer almamaktadır. İletilen dosyanın içinde coco.py adındaki dosyada veri setinin yüklenmesi ve dahil edilebilmesi için gerekli fonksiyonlar yazılmıştır. Kod çalıştırıldığında veri seti yükleme fonksiyonu çalıştırılırsa veri seti otomatik olarak yüklenmeye başlayacaktır.**

---

Örnek olarak bir fonksiyonla veri setinin içinden 1 indeksli resmi ve caption cümleleri görüntüsü şu şekilde olmaktadır:

```
show_image(idx=1)
```

```
A giraffe eating food from the top of the tree.
A giraffe standing up nearby a tree
A giraffe mother with its baby in the forest.
Two giraffes standing in a tree filled area.
A giraffe standing next to a forest filled with trees.
```




# Kullanılan Kütüphaneler

1. Matplotlib: Grafik görüntüleme ve veri görselleştirme
2. Tensorflow: Derin öğrenme fonksiyonlarının bulunduğu açık kaynak kütüphane
3. Numpy: Python için geliştirilmiş aritmetik amaçlı kütüphane
4. Os: Dosya işlemleri için kullanılan kütüphane
5. PIL: Resimler üzerinde işlemler için kullanılan kütüphane
6. Keras: Tensorflow kütüphanesi
7. Coco: Veri setine erişim için gerekli fonksiyonların bulunduğu kütüphane

# VGG16 Modelinin Yüklenmesi ve Oluşturulması

Modelin projeye yüklenmesi için Keras kullanılmaktadır. Gerekli kütüphaneler yüklendiğinde bir değişkene VGG16() fonksiyonu atandığında model yüklenecektir.

```
image_model = VGG16()
```

Modelin temel yapısı diyagram üzerinde verilmişti. Editör üzerinden görüntülemek istenirse image_model.summary() fonksiyonuyla modelin katmanları görüntülenebilir.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | (None, 224, 224, 3) | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1180160 |
| block4_conv2 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| block4_conv3 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| block4_pool (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 14, 14, 512) | 2359808 |

Hazır modelde 13 katman konvolüsyonel katman, 2 adet dense katman ve bir adet çıktı katmanı bulunmaktadır. Birinci yani input katmanı özelliklerine dikkat edilirse 224x224 boyutunda üç kanallı bir resim almaktadır. Kanal sayısı RGB (Red-Green-Blue) özelliğinden gelmektedir. Daha önce belirtildiği gibi son katman sinir ağından çıkarılacaktır. Bunun için projede son katman transfer_layer adında bir değişkene atanmıştır.

Modeli oluşturma aşamasında beklenen veri boyutu belirlenip bir değişkende saklanmalı ve modeli oluştururken parametre olarak verilmelidir. Programlama sürecinde kodun ekran görüntüsü şekildeki gibidir.

```
In [14]: transfer_layer = image_model.get_layer('fc2')

In [15]: image_model_transfer = Model(inputs=image_model.input,
                                       outputs=transfer_layer.output)

In [16]: img_size = K.int_shape(image_model.input)[1:3]
         img_size

Out[16]: (224, 224)

In [17]: transfer_values_size = K.int_shape(transfer_layer.output)[1]
         transfer_values_size

Out[17]: 4096

In [ ]:
```

Buradaki çıktılar anlamı yapay sinir ağı 224x224 boyutunda bir resimi girdi olarak alıp 4096 uzunluğunda bir vektör üretip çıktı olarak vektörü vermekte olduğudur

# Optimizasyon ve Performans

Normal koşullarda görsel işleme yapan bir sinir ağı oluştururken veri setindeki tüm görselleri sinir ağından geçirmek gerekmektedir. Bir görselin sinir ağından geçip üretilen vektörün optimize edilmesine ‘epoch’ denilmektedir. 20 epoch eğitim yapıldığı takdirde veri setindeki tüm görseller 20 kez sinir ağından geçecektir. Bu durum performans ve hızı bir hayli düşürecektir. VGG16 modeli hali hazırda eğitildiği için eğitilebilecek bir model değil. Bir görsel için her zaman aynı çıktıyı üretmektedir. Bu yüzden bir epoch eğitim yapıp çıkan değerler saklandığı takdirde hız oldukça artmaktadır.

Süreci izleyebilmek amaçlı yazılmış ekrana resimlerin yüzde kaçının işlenip işlemden çıktığını gösteren fonksiyon:

```
def print_progress(count, max_count):
    pct_complete = count / max_count

    msg = '\r- Progress: {0:.1%}'.format(pct_complete)

    sys.stdout.write(msg)
    sys.stdout.flush()
```

Resimleri modele yükleyecek olan fonksiyon:

```
def process_images(data_dir, filenames, batch_size=32):,
    num_images = len(filenames),

     shape = (batch_size,) + img_size + (3,),
    image_batch = np.zeros(shape=shape, dtype=np.float16),

     start_index = 0,

    while start_index < num_images:,
        print_progress(count=start_index, max_count=num_images),
        end_index = start_index + batch_size,

        if end_index > num_images:,
            end_index = num_images,

        current_batch_size = end_index - start_index,

        for i, filename in enumerate(filenames[start_index:end_index]):,
            path = os.path.join(data_dir, filename),
            img = load_image(path, size=img_size),
            image_batch[i] = img

        transfer_values_batch = image_model_transfer.predict(image_batch[0:current_batch_size]),
        transfer_values[start_index:end_index] = transfer_values_batch[0:current_batch_size],

        start_index = end_index,

    print(),
     return transfer_values
```

# Tokenleştirme

Tokenleştirme işlemine başlamadan önce başlangıç ve bitiş tokenleri belirlenir Başlangıç tokeni ‘ssss’, bitiş tokeni ‘eeee’ olmaktadır. Caption’lara başlangıç ve bitiş tonkenlereni eklemek için yazılan fonksiyon:

```
In [23]: def mark_captions(captions_listlist):
             captions_marked = [[mark_start + caption + mark_end
                                 for caption in captions_list]
                                 for captions_list in captions_listlist]
             return captions_marked
```

```
In [25]: captions_marked = mark_captions(captions)
```

```
In [26]: captions_marked[1]
```

```
Out[26]: ['ssss A giraffe eating food from the top of the tree. eeee',
          'ssss A giraffe standing up nearby a tree  eeee',
          'ssss A giraffe mother with its baby in the forest. eeee',
          'ssss Two giraffes standing in a tree filled area. eeee',
          'ssss A giraffe standing next to a forest filled with trees. eeee']
```

Sonrasında keras kütüphanesinde tokenleştirme yapabilmek için tüm caption'lar tek bir liste içerisinde bulunmalıdır. Verileri düzleştirmek için yazılan fonksiyon:

```
def flatten(captions_listlist):
    captions_list = [caption
                     for captions_list in captions_listlist
                     for caption in captions_list]
    return captions_list
```

```
captions_flat = flatten(captions_marked)
```

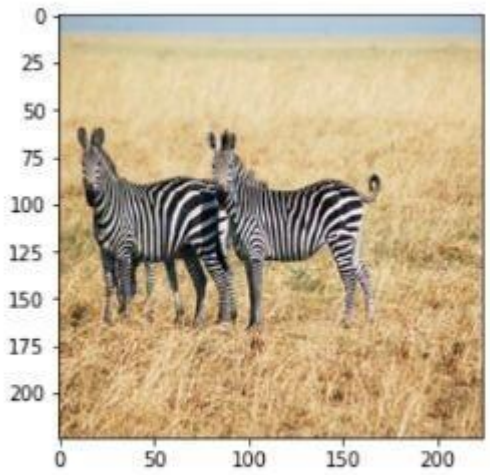
```
num_words = 10000
```

Tokenleştirme işlemi için bir wrapper klası yazdım. init metodu içerisinde tokenleştirme gerçekleştirilmektedir.

```
class TokenizerWrap(Tokenizer): # CP captions_to_tokens'i kopyalama, yazılacak

    def __init__(self, texts, num_words=None):
        Tokenizer.__init__(self, num_words=num_words)

        self.fit_on_texts(texts)

        self.index_to_word = dict(zip(self.word_index.values(),
                                      self.word_index.keys()))

    def token_to_word(self, token):
        word = " " if token == 0 else self.index_to_word[token]
        return word

    def tokens_to_string(self, tokens):
        words = [self.index_to_word[token] for token in tokens if token != 0]

        text = " ".join(words)

        return text

    def captions_to_tokens(self, captions_listlist):
        tokens = [self.texts_to_sequences(captions_list)
                  for captions_list in captions_listlist]
        return tokens
```

# Sonuç ve Ekran Görüntüleri

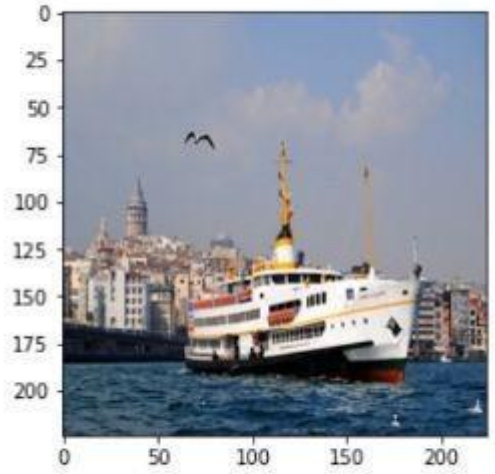
```
generate_caption('images/1.jpg')
```




```
Predicted caption:
 a group of zebras are standing in a field eeee
```

Başarılı bir tahmin gerçekleştirilmiştir.

```
generate_caption('images/2.jpg')
```




```
Predicted caption:
 a large body of water with a clock tower in the background eeee
```

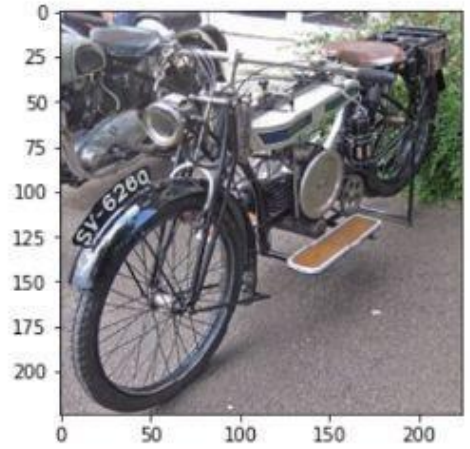
Kısmen doğru bir tahmin gerçekleştirilmiştir.

```
generate_caption('images/3.jpg')
```

```
Predicted caption:
 a man riding a wave on top of a surfboard eeee
```

Hatalı bir tahmin gerçekleştirilmiştir.

```
generate_caption_coco(idx=11)
```

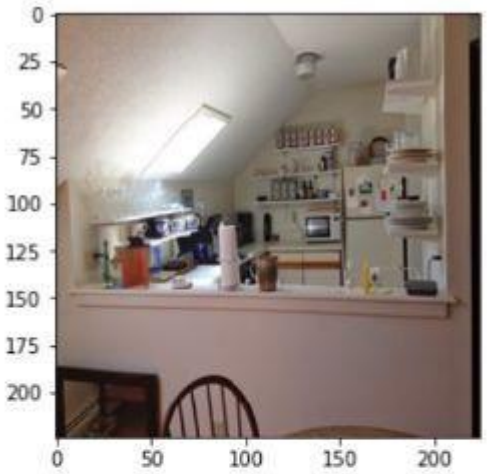


```
Predicted caption:
 a bike parked next to a parking meter eeee

True captions:
A motorcycle parked in a parking space next to another motorcycle.
An old motorcycle parked beside other motorcycles with a brown leather seat.
Motorcycle parked in the parking lot of asphalt.
A close up view of a motorized bicycle, sitting in a rack.
The back tire of an old style motorcycle is resting in a metal stand.
```

Doğru bir tahmin gerçekleştirilmiştir.


```
Predicted caption:
 a group of people are sitting at a table eeee

True captions:
a man and woman cut into a big cake
A man and woman standing in front of a cake.
A women who is cutting into a cake.
two people standing near a table with a cake
A woman cutting into a cake with a man standing behind her
```

Doğru bir tahmin gerçekleştirilmiştir.




```
Predicted caption:
 a kitchen with a stove and a refrigerator eeee

True captions:
A small kitchen with low a ceiling
A small kichen area with a sunlight and and angled ceiling.
an image of a kitchen loft style setting
a small kitchen with a lot of filled up shelves
A kitchen with a slanted ceiling and skylight.
```

Doğru bir tahmin gerçekleştirilmiştir.

# Kaynakça


1. https://allennlp.org/
2. http://nlpprogress.com/
3. https://www.youtube.com/watch?v=uCSTpOLMC48&t=6s
4. http://cocodataset.org/#home
5. https://www.investopedia.com/terms/d/datamining.asp