

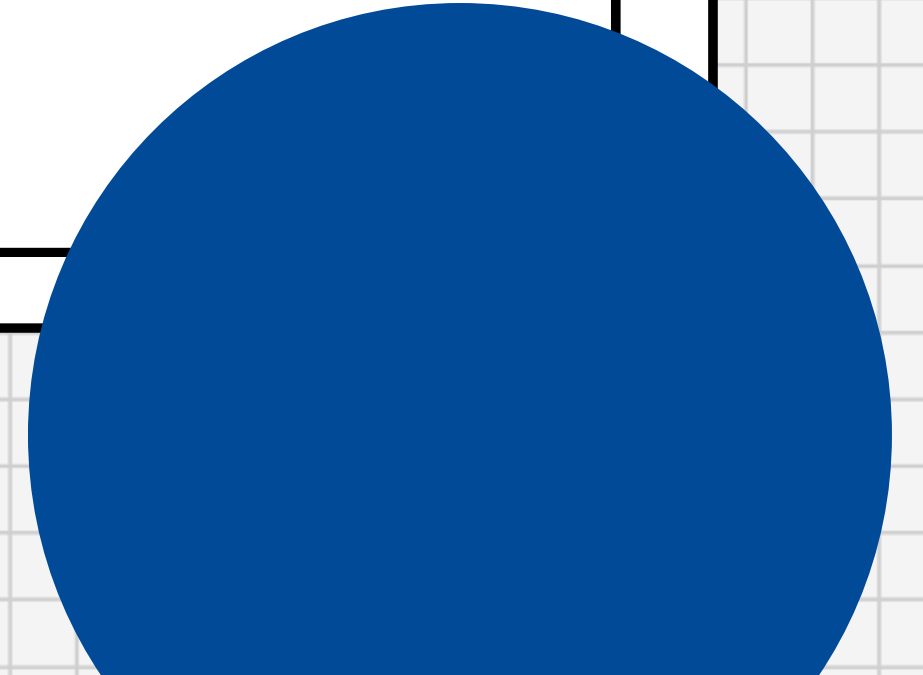
-Image Captioning-

Arama



Bilgisayar Mühendisliği Uygulama Tasarımı II

Nihal Demir



Kodlayıcı - Kod çözücü Modeli

Kodlayıcılar görsel girdiyi okuyan ve dahili bir temsil kullanarak içeriği sabit uzunlukta bir vektöre kodlayan bir ağ modeli, kod çözücüler ise kodlanmış görseli okuyan ve metinsel açıklama çıktısını üreten bir ağ modelidir.

Kodlayıcı: CNN

CNN, minimum ön işleme ile doğrudan piksel görüntülerden görsel kalıpları tanımak için tasarlanmış özel bir çok katmanlı sinir ağıdır. Özellikleri otomatik olarak oluşturmak ve sınıflandırıcı ile birleştirmek için kullanılmaktadır. Derin öğrenme için sıklıkla kullanılan bir mimaridir ve bilgisayarlı görü alanında yaygın olarak kullanılmaktadır. Genellikle bilgisayarlı görü, ses tanıma gibi alanlarda kullanılmakla birlikte tavsiye sistemleri, doğal dil işleme, video analizi gibi birçok alanda başarı göstermektedir.

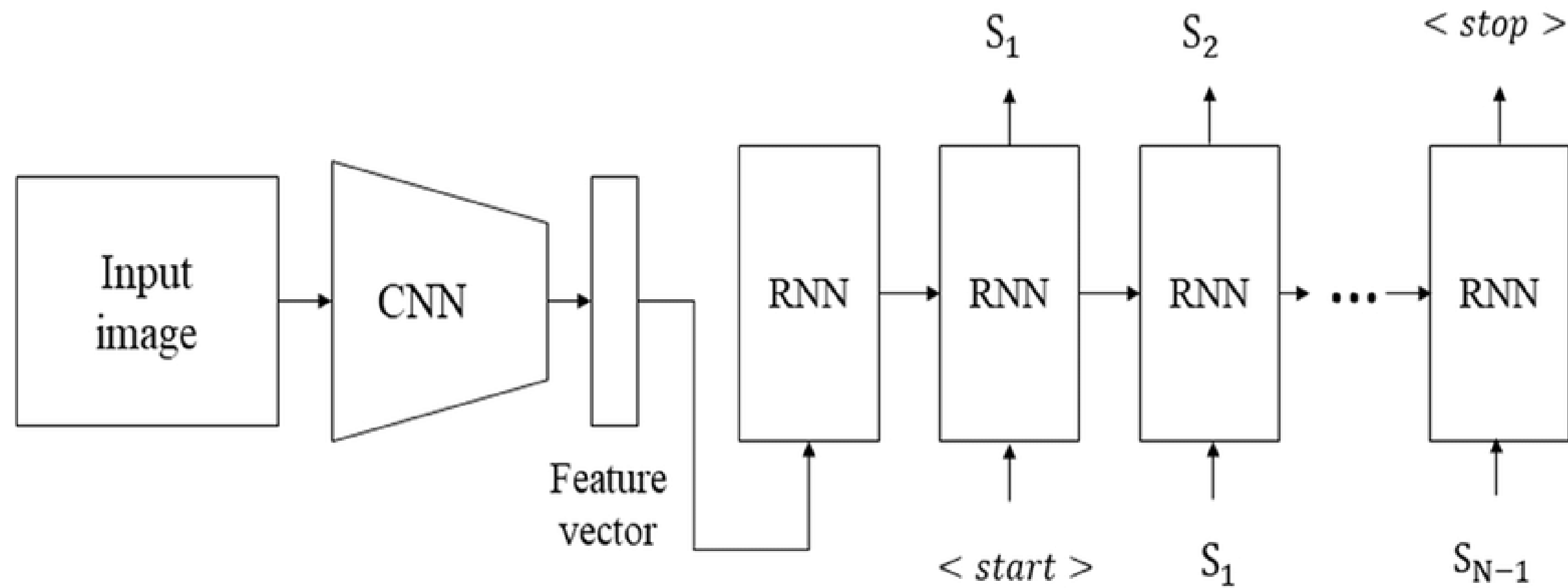
Kod Çözücü: RNN

RNN'ler olarak da bilinen tekrarlayan sinir ağları, gizli durumlara sahipken önceki çıktıların girdi olarak kullanılmasına izin veren bir sinir ağları sınıfıdır. RNN'ler genelde bir sonraki adımı tahmin etmek için kullanılır. Diğer derin öğrenme yapılarından en büyük farkları ise hatırlamalarıdır. Normal sinir ağlarında çıktı sadece o anki girdiye bağlıyken, RNN çıktısı sistemin genel durumuna da bağlıdır.

RNN'in vektörden diziye, diziden vektöre, diziden diziye gibi farklı işlevler için farklı türleri mevcuttur.

Kodlayıcı - Kod Çözücü Modeli

Arama



Kaydet

İptal

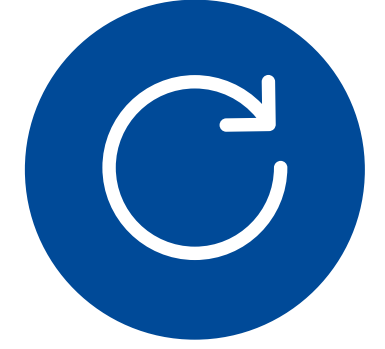
GRADYAN YOK OLMASI ve LSTM

Geriye yayılım, derin sinir ağlarında öğrenmeyi mümkün kılan ana algoritmadır. Geriye yayılımda geri bildirim sinyali çıktı kaybindan alınarak önceki katmanlara yayılır. Geri bildirim sinyalinin üst üste katmanlardan aşağıya yayılırken çok küçülmesi, hatta yok olması ağı eğitilemez hale getirmesi, *gradyan yok olması* olarak adlandırılır.

Bu sorun genellikle yinelemeli ağlarda çok fazla katman olduğu zaman ve geri bildirim sinyalinin tüm bu katmanlara yayılması gerektiğinde oluşur. LSTM katmanı, ana akışa paralel olarak doğrusal bilgi taşıma ile gradyanın derin katmanlar arasında taşınmasını sağlamaktadır. LSTM yapısı içerisindeki kapılar (gate) neyin hatırlanacağını, neyin unutulacağını belirler. Yani gelen girdi önemsizse unutulur, önemliyse bir sonraki aşamaya aktarılır.

Özellik Çıkarımı

VGG16



Özellik, bir görüntüdeki bir nesnenin onu tanımlamaya yardımcı olan parçalarıdır.

Özellikler köşeler, kenarlar, çıkıntılar gibi özellikleri barındırır.

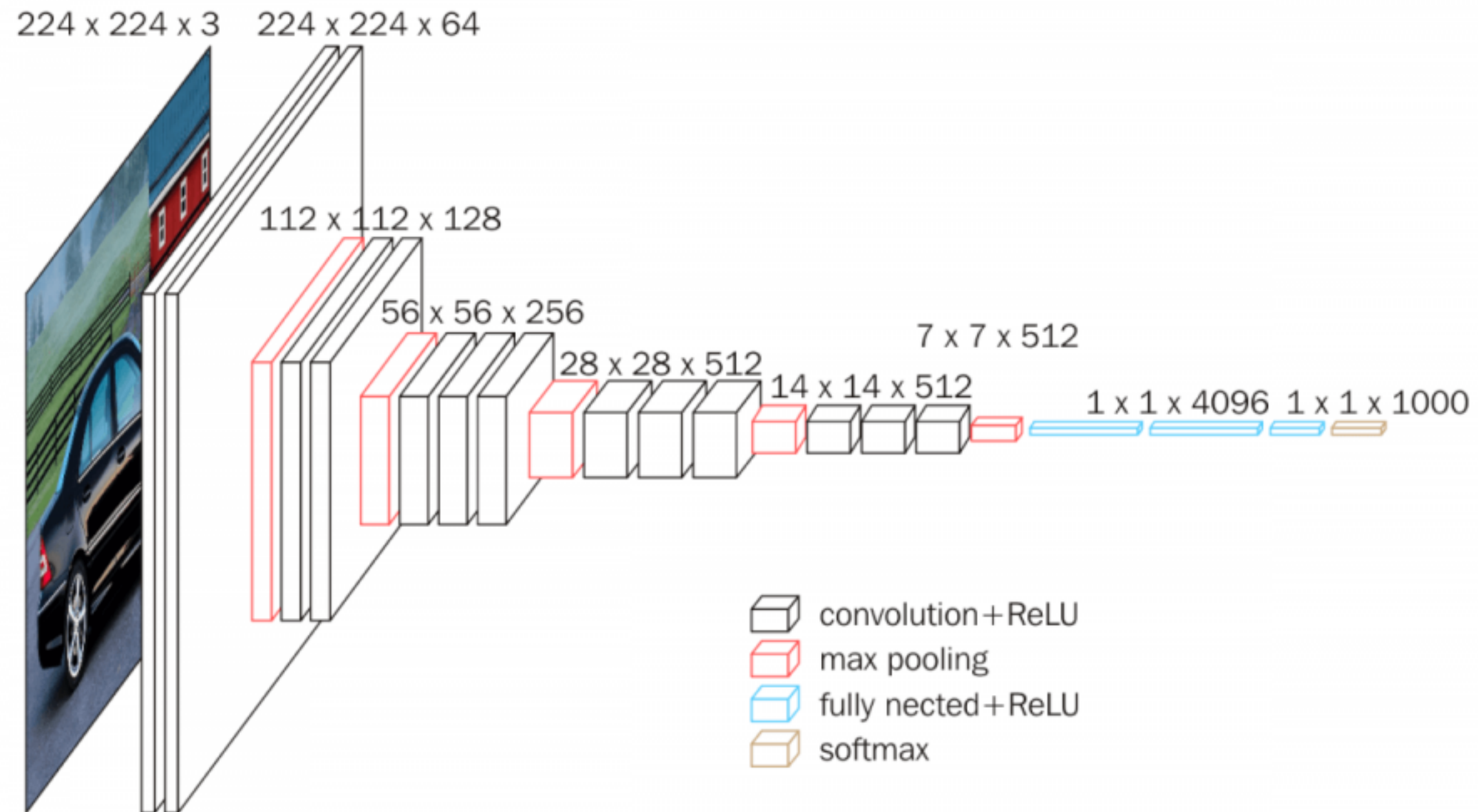
Özellik çıkarımı, daha önce ağ tarafından öğrenilen gösterimler ile yeni örnekler için niteliklerin çıkarılmasıdır.

VGG16, 16 katmanlı ve toplamda 138 milyon parametreye sahip kapsamlı bir derin sinir ağıdır.

VGG16, 3 tam bağlantılı (fully connected) katmana ve 13 evirişim (convolution) katmanına sahiptir.

VGG16 Modeli

Arama



Kaydet

İptal

Girdi (Input): VGG16, 3 (RGB) kanallı giriş tensörünü sabit olarak 224x224 boyutunda alır.

Evrişimli Katmanlar (Convolutional Layers): Evrişimli filtreler, mümkün olan en küçük alıcı alanı olan 3×3'ü kullanır.

ReLU aktivasyonu: Aktivasyon fonksiyonu, düğümden gelen toplam ağırlıklı girdiyi, o girdi için düğümün veya çıktının aktivasyonuna dönüştürmekten sorumludur.

Pooling Katmanları: Bir pooling katmanı birkaç evrişim katmanını takip eder; bu, her bir evrişim adımında oluşturulan özellik haritalarının boyutsallığını ve parametre sayısını azaltmaya yardımcı olur.

Gizli Katmanlar (Hidden Layers): VGG'nin tüm gizli katmanları ReLU'yu kullanır. ReLU kullanımı eğitim süresini kısaltmaktadır.

Tam Bağlantılı Katmanlar (Fully Connected Layers): VGG'nin birbiriyle tamamen bağlantılı üç katmanı vardır: ilk ikisinin her biri 4096 kanala sahiptir ve üçüncüsü, her sınıf için 1 tane olmak üzere 1000 kanala sahiptir.

ReLU Aktivasyon Fonksiyonu



Düzeltilmiş lineer aktivasyon fonksiyonu , kısaca ReLU , pozitif ise doğrudan girişi verecek olan parçalı bir lineer fonksiyondur, aksi takdirde sıfır çıkışı verir.



ReLU, kullanılan modele eğitim kolaylığı ve genellikle daha iyi performans sağladığından birçok sinir ağında varsayılan aktivasyon işlemi olarak kabul edilmektedir.



Hesaplama işlemi daha ucuz ve basittir: aktivasyonlarda üstel fonksiyonu hesaplamaya gerek yoktur.



Bu aktivasyon fonksiyonu ile eğitilen ağlar, gradyanlar düğüm aktivasyonlarıyla orantılı kaldığından, yok olan gradyan probleminden neredeyse tamamen kaçınır.



Kodlama Aşaması

```

import os
import pickle
import numpy as np
from tqdm.notebook import tqdm

from keras.applications.vgg16 import VGG16 , preprocess_input
from keras.utils import load_img, img_to_array
from keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
from keras.models import Model
from keras.utils import to_categorical, plot_model
from keras.layers import Input,Dense,LSTM, Embedding, Dropout, add

```

```

model = VGG16()
model = Model(inputs = model.inputs , outputs = model.layers[-2].output)
print(model.summary())

```

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
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312

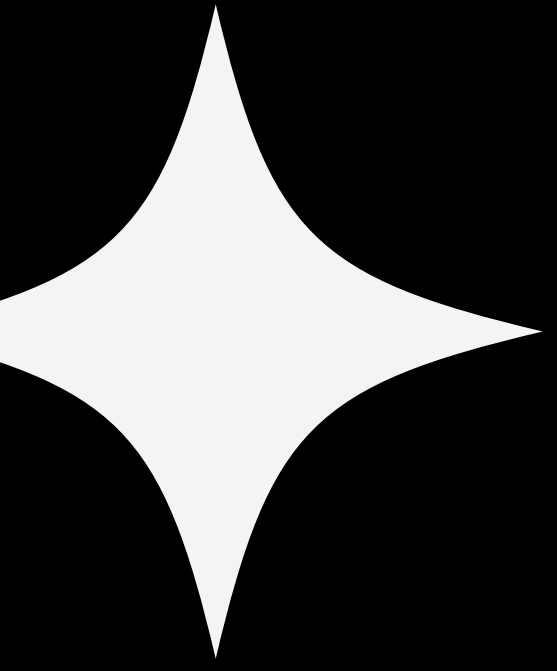
```
features = {}  
directory = '/content/drive/MyDrive/Project/Images'
```

```
for img_name in tqdm(os.listdir(directory)):  
    img_path = directory + '/' + img_name  
    image = load_img(img_path, target_size=(224, 224))  
    image = img_to_array(image)  
    image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))  
    image = preprocess_input(image)  
    feature = model.predict(image, verbose=0)  
    image_id = img_name.split('.')[0]  
    features[image_id] = feature
```

- MyDrive
 - Colab Notebooks
 - Project
 - Images
 - best_model.h5
 - best_model2.h5
 - best_model3.h5
 - best_model4.h5
 - best_model5.h5
 - captions.txt
 - features.pkl

```
WORKING_DIR = '/content/drive/MyDrive/Project'  
pickle.dump(features, open(os.path.join(WORKING_DIR, 'features.pkl'), 'wb'))
```

```
WORKING_DIR = '/content/drive/MyDrive/Project'  
with open(os.path.join(WORKING_DIR, 'features.pkl'), 'rb') as f:  
    features = pickle.load(f)
```

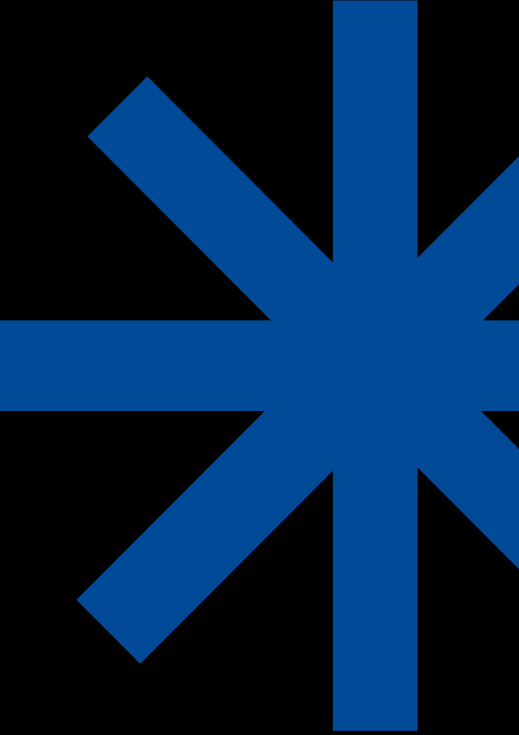


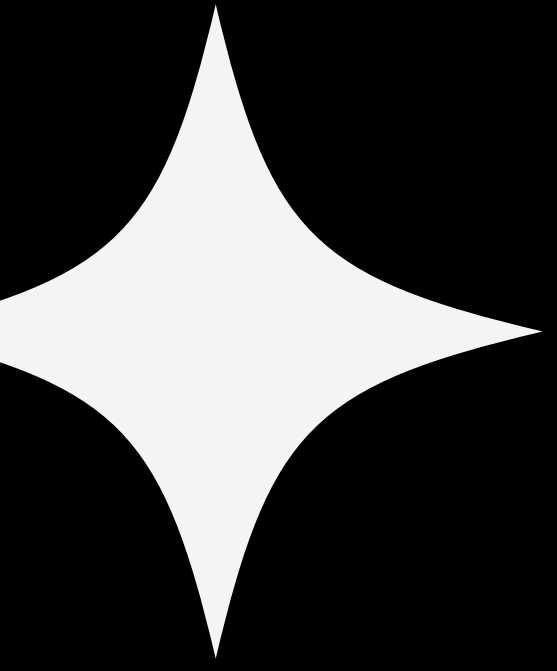
```
mapping = {}
for line in tqdm(captions_doc.split('\n')):
    tokens = line.split(',')
    if len(line) < 2:
        continue
    image_id, caption = tokens[0], tokens[1:]
    image_id = image_id.split('.')[0]
    caption = " ".join(caption)
    if image_id not in mapping:
        mapping[image_id] = []
    mapping[image_id].append(caption)
```

100%  40456/40456 [00:00<00:00, 360399.03it/s]

```
mapping['1000268201_693b08cb0e']
```

```
['A child in a pink dress is climbing up a set of stairs in an entry way .',
 'A girl going into a wooden building .',
 'A little girl climbing into a wooden playhouse .',
 'A little girl climbing the stairs to her playhouse .',
 'A little girl in a pink dress going into a wooden cabin .']
```

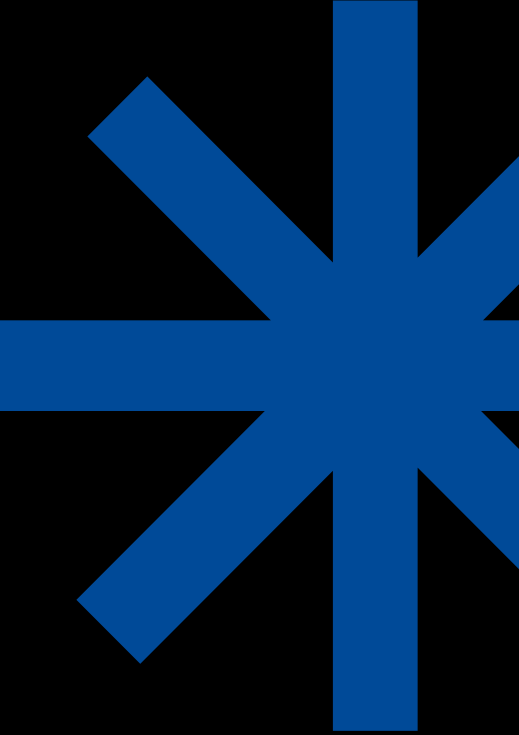




```
def clean(mapping):
    for key, captions in mapping.items():
        for i in range(len(captions)):
            caption = captions[i]
            caption = caption.lower()
            caption = caption.replace('[^A-Za-z]', '')
            caption = caption.replace('\s+', ' ')
            caption = 'startseq ' + " ".join([word for word in caption.split() if len(word)>1]) + ' endseq'
            captions[i] = caption
```

```
mapping['1000268201_693b08cb0e']
```

```
['startseq child in pink dress is climbing up set of stairs in an entry way endseq',
 'startseq girl going into wooden building endseq',
 'startseq little girl climbing into wooden playhouse endseq',
 'startseq little girl climbing the stairs to her playhouse endseq',
 'startseq little girl in pink dress going into wooden cabin endseq']
```





Tokenizer, girdi metinlerinin sözcük belirteçlerinin boşluklarla sınırlandırıldığını varsayar.

`fit_on_texts` ile her kelime belirtecinin ve onun benzersiz tamsayı indeksinin bir eşleşmesini içeren, tüm metin için bir sözlük oluşturulur.

`Tokenizer.fit_on_texts()` işleminden sonra derlem sözlüğüne, yani kelimelerin eşleştirilmesine ve benzersiz tamsayı indekslerine bakılır.



```
tokenizer = Tokenizer()  
tokenizer.fit_on_texts(all_captions)  
vocab_size = len(tokenizer.word_index) + 1
```

`vocab_size`

8485

```
max_length = max(len(caption.split()) for caption in all_captions)  
max_length
```

35

```
image_ids = list(mapping.keys())  
split = int(len(image_ids) * 0.80)  
train = image_ids[:split]  
test = image_ids[split:]
```




Dropout: Eğitim sırasında aşırı öğrenmeyi(overfitting) engellemek için bazı nöronları unutmak için kullanılır.

Dense: Dense, çoğu durumda çalışan standart bir katman türüdür. Yoğun bir katmanda, önceki katmandaki tüm düğümler mevcut katmandaki düğümlere bağlanır.

Embedding: Bir tür gizli katmandır. Bu katman, girdi bilgisini yüksek boyutlu uzaydan düşük boyutlu uzaya eşler ve ağırlar arasındaki ilişki hakkında daha fazla şey öğrenmesine ve verileri daha verimli bir şekilde işlemesine olanak tanır.



```
# encoder model

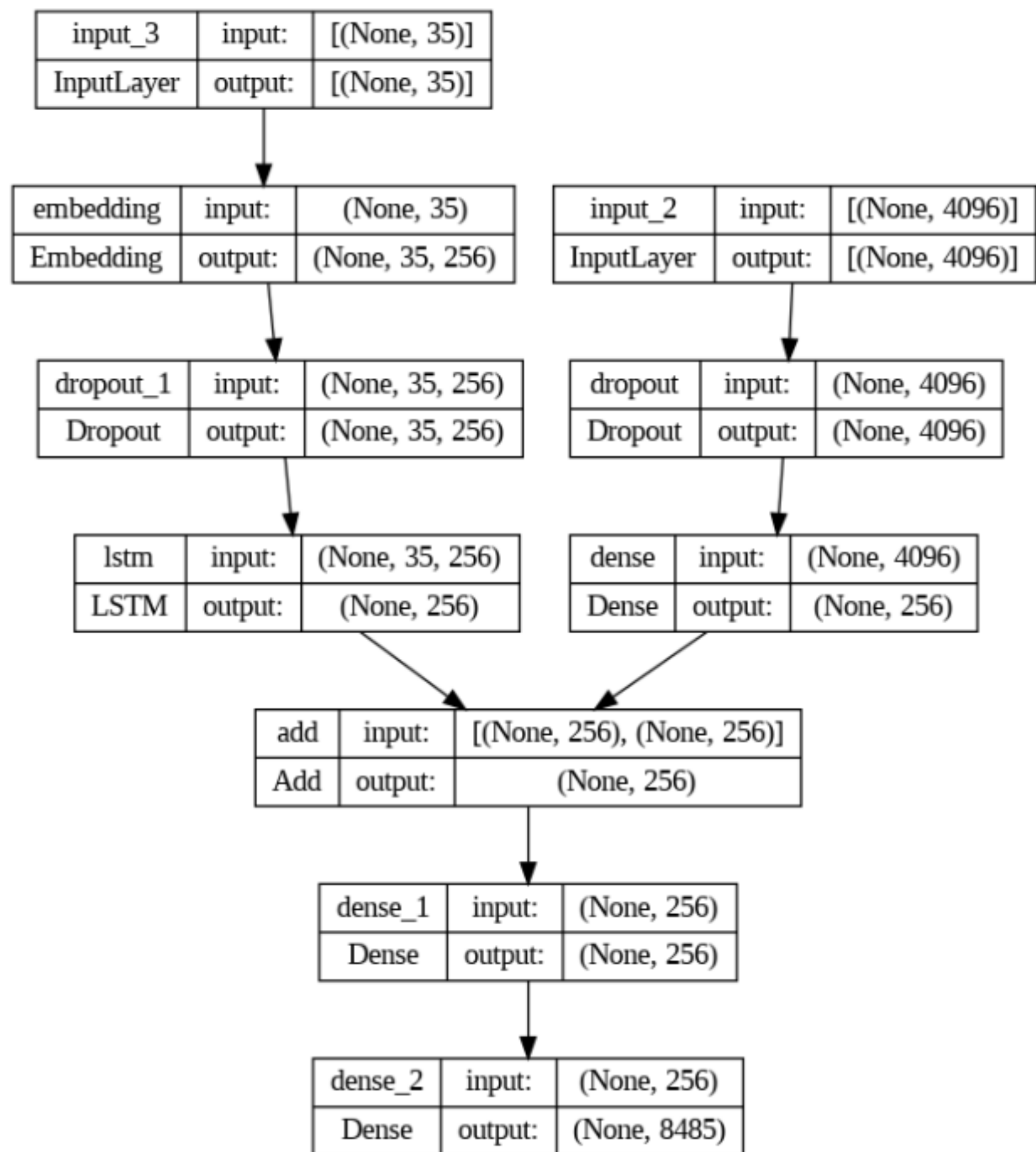
# image
inputs1 = Input(shape=(4096,))
fe1 = Dropout(0.4)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)

# sequence
inputs2 = Input(shape=(max_length,))
se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
se2 = Dropout(0.4)(se1)
se3 = LSTM(256)(se2)

# decoder model
decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)

model = Model(inputs=[inputs1, inputs2], outputs=outputs)
model.compile(loss='categorical_crossentropy', optimizer='adam')

plot_model(model, show_shapes=True)
```





```
▶ epochs = 15
  batch_size = 8
  steps = len(train) // batch_size

  for i in range(epochs):
    generator = data_generator(train, mapping, features, tokenizer, max_length, vocab_size, batch_size)
    model.fit(generator, epochs=1, steps_per_epoch=steps, verbose=1)
```

```
☞ 809/809 [=====] - 1332s 2s/step - loss: 4.7991
809/809 [=====] - 1336s 2s/step - loss: 3.8172
809/809 [=====] - 1327s 2s/step - loss: 3.4606
809/809 [=====] - 1344s 2s/step - loss: 3.2203
809/809 [=====] - 1327s 2s/step - loss: 3.0339
809/809 [=====] - 1324s 2s/step - loss: 2.8799
809/809 [=====] - 1322s 2s/step - loss: 2.7604
809/809 [=====] - 1329s 2s/step - loss: 2.6570
809/809 [=====] - 1331s 2s/step - loss: 2.5762
809/809 [=====] - 1351s 2s/step - loss: 2.5100
809/809 [=====] - 1334s 2s/step - loss: 2.4465
809/809 [=====] - 1323s 2s/step - loss: 2.3944
809/809 [=====] - 1304s 2s/step - loss: 2.3442
809/809 [=====] - 1294s 2s/step - loss: 2.3005
809/809 [=====] - 1313s 2s/step - loss: 2.2608
```

EPOCH: Model eğitilirken verilerin tamamı aynı anda eğitime katılmaz. Belli sayıda parçalar halinde eğitimde yer alırlar. İlk parça eğitilir, modelin başarımı test edilir, başarıma göre geriye yayılım (“backpropagation”) ile ağırlıklar güncellenir. Daha sonra yeni eğitim kümesi ile model tekrar eğitilip ağırlıklar tekrar güncellenir. Bu işlem her bir eğitim adımında tekrarlanarak model için en uygun ağırlık değerleri hesaplanmaya çalışılır. Bu eğitim adımlarının her birine “epoch” denilmektedir.

BATCH SIZE: Batch size(boyutu), parametre güncellemesinin gerçekleştiği ağıya verilen alt örneklerin sayısıdır.

```
def predict_caption(model, image, tokenizer, max_length):
    in_text = 'startseq'
    # iterasyonumuzu max uzunluk kadar döndürüyoruz
    for i in range(max_length):
        # girdi dizisi
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        # pad_sequences sabit uzunluk sağlamak için
        sequence = pad_sequences([sequence], max_length)
        # sonraki kelimenin tahmin edilmesi aşaması
        yhat = model.predict([image, sequence], verbose=0)
        # olasılığı yüksek olan indeksi alırız
        yhat = np.argmax(yhat)
        #index - word dönüşümü
        word = idx_to_word(yhat, tokenizer)
        if word is None:
            break
        # bir sonraki kelimeyi oluşturmak için kelimeyi girdi olarak ekliyoruz.
        in_text += " " + word
        if word == 'endseq':
            break
```

```
from nltk.translate.bleu_score import corpus_bleu
# test verileri ile doğrulama işlemi gerçekleştiriyoruz.
actual, predicted = list(), list()

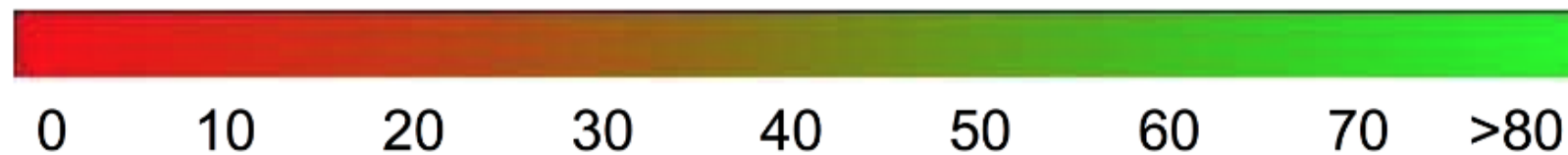
for key in tqdm(test):
    # gerçek başlık
    captions = mapping[key]
    # tahmin edilen başlık
    y_pred = predict_caption(model, features[key], tokenizer, max_length)
    # split işlemi ile kelimelere bölüyoruz
    actual_captions = [caption.split() for caption in captions]
    y_pred = y_pred.split()
    # gerçek ve tahmin edilen başlıkları ayrı ayrı listeliyoruz.
    actual.append(actual_captions)
    predicted.append(y_pred)
# BLEU hesaplanması
print("BLEU-1: %f" % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
print("BLEU-2: %f" % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
```

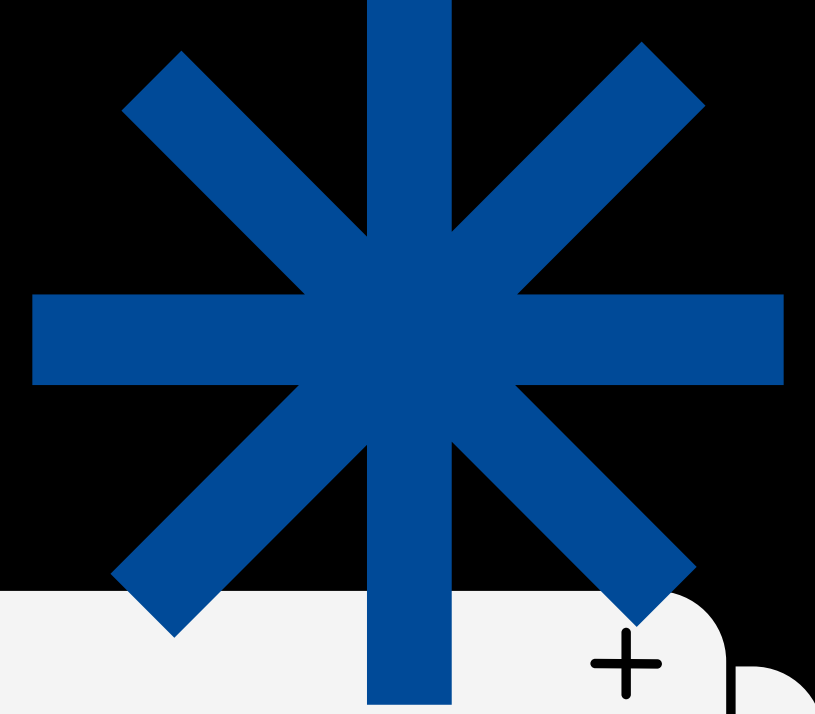
100% 1619/1619 [21:06<00:00, 1.33it/s]

BLEU-1: 0.526225

BLEU-2: 0.296451

BLEU ya da İki Dilli Değerlendirme Alt Çalışması, bir metnin aday çevirisini bir veya daha fazla referans çevirisiyle karşılaştırmak için kullanılan bir puan, kalite ölçüsüdür.





Sonuçlar

BLEU Skorları

Arama



epoch	batch size	bleu-1	bleu-2
5	8	0.545	0.320
5	16	0.554	0.323
10	8	0.523	0.298
10	16	0.529	0.306
15	8	0.526	0.296

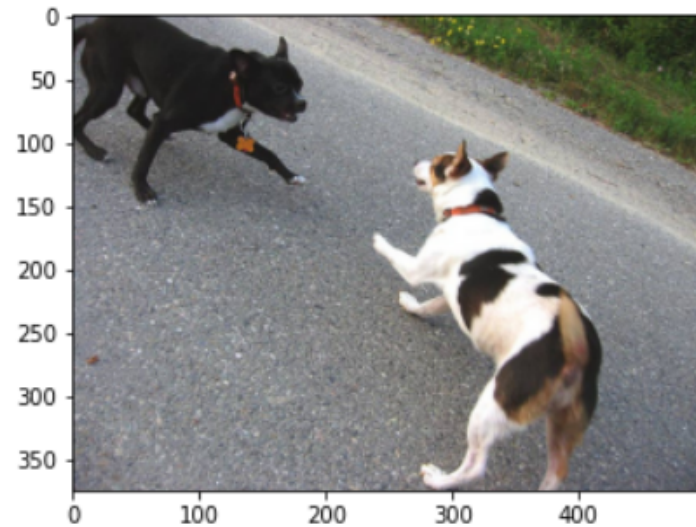
Kaydet

İptal

EPOCH : 5
BATCH SIZE : 8

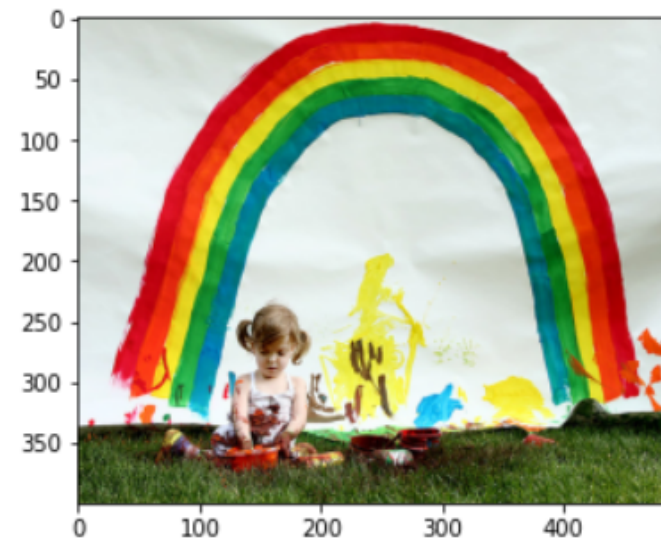
▶ generate_caption("1001773457_577c3a7d70.jpg")

↗ -----Actual-----
startseq black dog and spotted dog are fighting endseq
startseq black dog and tri-colored dog playing with each other on the road endseq
startseq black dog and white dog with brown spots are staring at each other in the street endseq
startseq two dogs of different breeds looking at each other on the road endseq
startseq two dogs on pavement moving toward each other endseq
-----Predicted-----
startseq two dogs are playing with toy in the background endseq



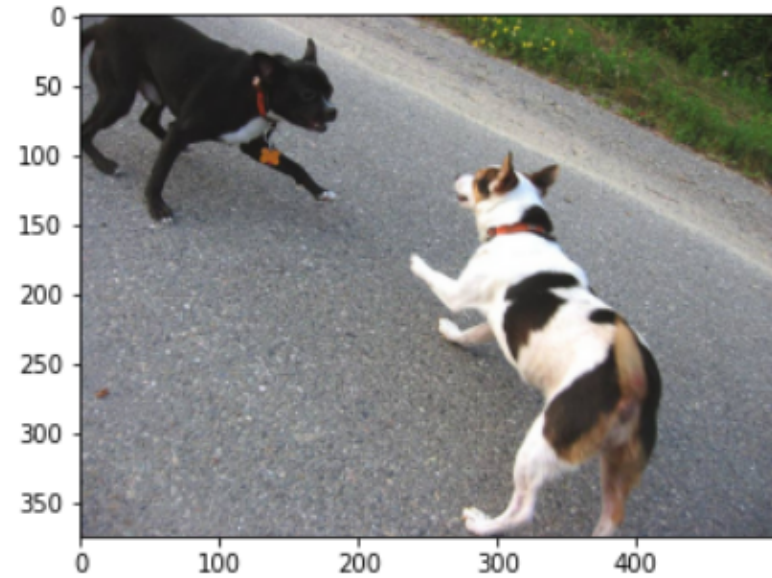
▶ generate_caption("1002674143_1b742ab4b8.jpg")

↗ -----Actual-----
startseq little girl covered in paint sits in front of painted rainbow with her hands in bowl endseq
startseq little girl is sitting in front of large painted rainbow endseq
startseq small girl in the grass plays with fingerpaints in front of white canvas with rainbow on it endseq
startseq there is girl with pigtails sitting in front of rainbow painting endseq
startseq young girl with pigtails painting outside in the grass endseq
-----Predicted-----
startseq two children are sitting in the snow endseq



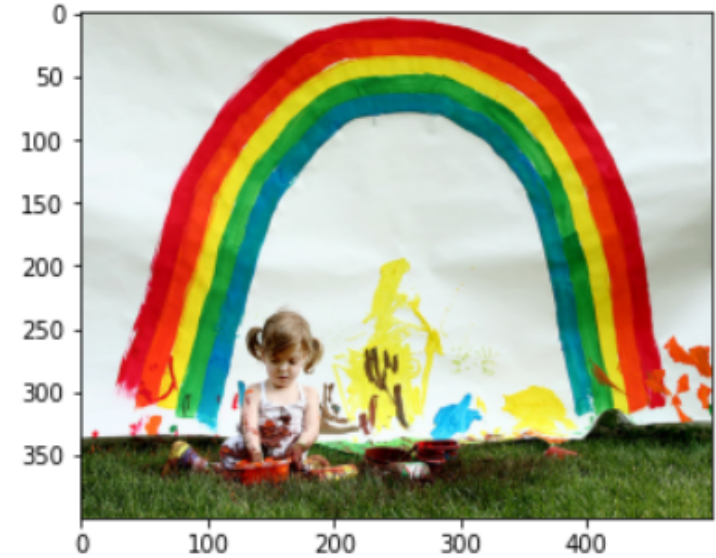
EPOCH : 5
BATCH SIZE : 16

-----Actual-----
startseq black dog and spotted dog are fighting endseq
startseq black dog and tri-colored dog playing with each other on the road endseq
startseq black dog and white dog with brown spots are staring at each other in the street endseq
startseq two dogs of different breeds looking at each other on the road endseq
startseq two dogs on pavement moving toward each other endseq
-----Predicted-----
startseq two dogs are playing tug of war with red toy endseq



generate_caption("1002674143_1b742ab4b8.jpg")

-----Actual-----
startseq little girl covered in paint sits in front of painted rainbow with her hands in bowl
startseq little girl is sitting in front of large painted rainbow endseq
startseq small girl in the grass plays with fingerpaints in front of white canvas with rainbow
startseq there is girl with pigtails sitting in front of rainbow painting endseq
startseq young girl with pigtails painting outside in the grass endseq
-----Predicted-----
startseq two children are playing in the snow endseq



EPOCH : 10
BATCH SIZE : 8

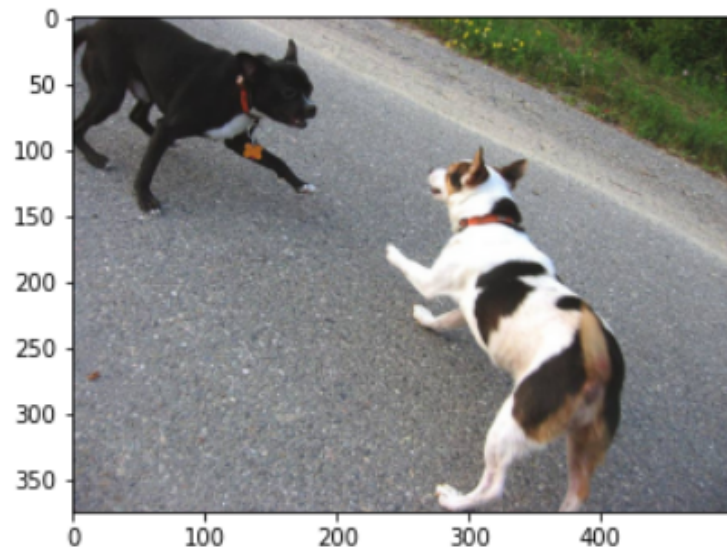
```
generate_caption("1001773457_577c3a7d70.jpg")
```

-----Actual-----

```
startseq black dog and spotted dog are fighting endseq  
startseq black dog and tri-colored dog playing with each other on the road endseq  
startseq black dog and white dog with brown spots are staring at each other in the street endseq  
startseq two dogs of different breeds looking at each other on the road endseq  
startseq two dogs on pavement moving toward each other endseq
```

-----Predicted-----

```
startseq two dogs playing tug of war with red toy endseq
```



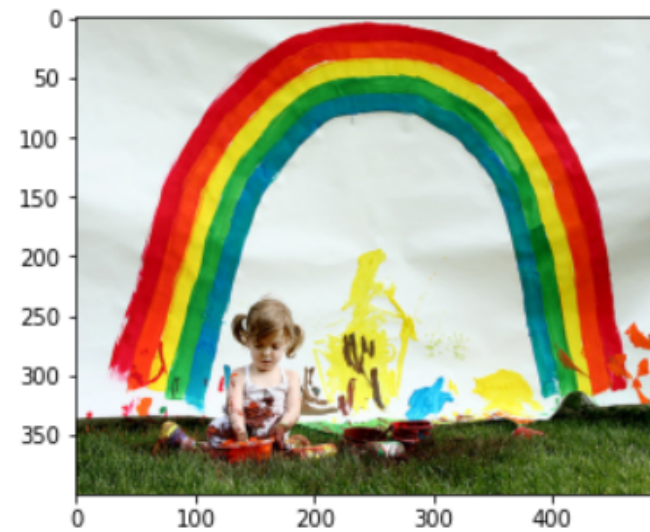
```
generate_caption("1002674143_1b742ab4b8.jpg")
```

-----Actual-----

```
startseq little girl covered in paint sits in front of painted rainbow with her hands in bowl endseq  
startseq little girl is sitting in front of large painted rainbow endseq  
startseq small girl in the grass plays with fingerpaints in front of white canvas with rainbow on it e  
startseq there is girl with pigtails sitting in front of rainbow painting endseq  
startseq young girl with pigtails painting outside in the grass endseq
```

-----Predicted-----

```
startseq two women in colorful outfits are sitting on bench in front of wooden barn endseq
```

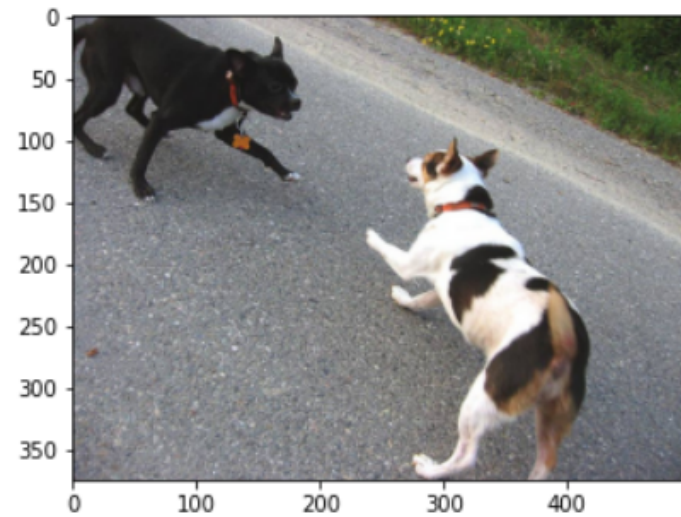


EPOCH : 10
BATCH SIZE : 16

```
[36] generate_caption("1001773457_577c3a7d70.jpg")
```

```
-----Actual-----  
startseq black dog and spotted dog are fighting endseq  
startseq black dog and tri-colored dog playing with each other on the road endseq  
startseq black dog and white dog with brown spots are staring at each other in the street endseq  
startseq two dogs of different breeds looking at each other on the road endseq  
startseq two dogs on pavement moving toward each other endseq
```

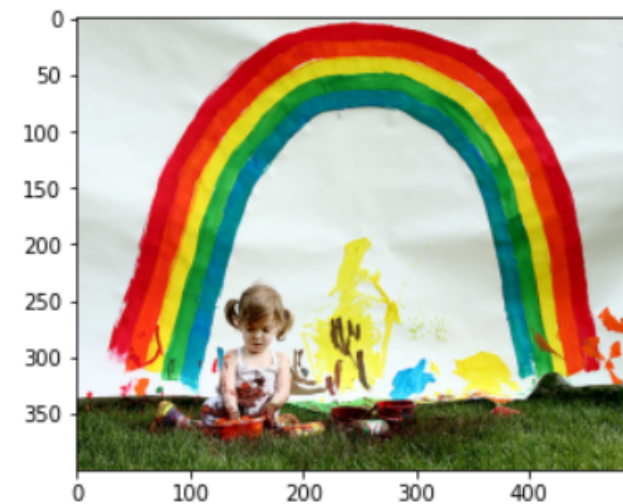
```
-----Predicted-----  
startseq two dogs are playing on the ground endseq
```



```
▶ generate_caption("1002674143_1b742ab4b8.jpg")
```

```
↗ -----Actual-----  
startseq little girl covered in paint sits in front of painted rainbow with her hands in bowl endseq  
startseq little girl is sitting in front of large painted rainbow endseq  
startseq small girl in the grass plays with fingerpaints in front of white canvas with rainbow on it endseq  
startseq there is girl with pigtails sitting in front of rainbow painting endseq  
startseq young girl with pigtails painting outside in the grass endseq
```

```
-----Predicted-----  
startseq two girls are sitting in car in front of green car endseq
```



EPOCH : 15
BATCH SIZE : 8

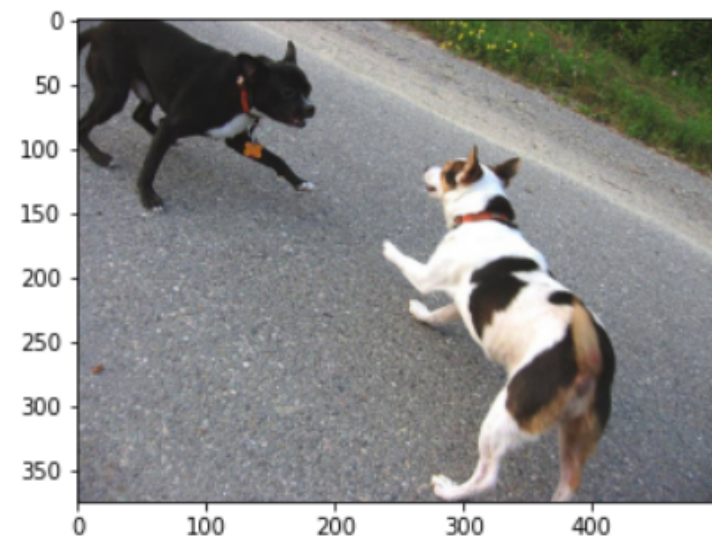
▶ generate_caption("1001773457_577c3a7d70.jpg")

-----Actual-----

startseq black dog and spotted dog are fighting endseq
startseq black dog and tri-colored dog playing with each other on the road endseq
startseq black dog and white dog with brown spots are staring at each other in the street endseq
startseq two dogs of different breeds looking at each other on the road endseq
startseq two dogs on pavement moving toward each other endseq

-----Predicted-----

startseq two dogs playing with toy in the grass endseq



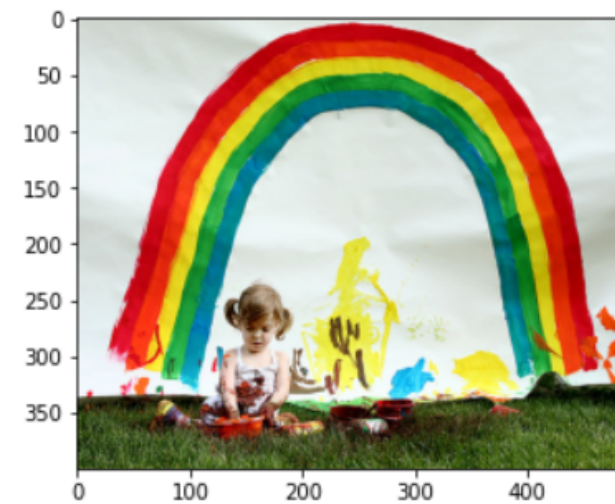
▶ generate_caption("1002674143_1b742ab4b8.jpg")

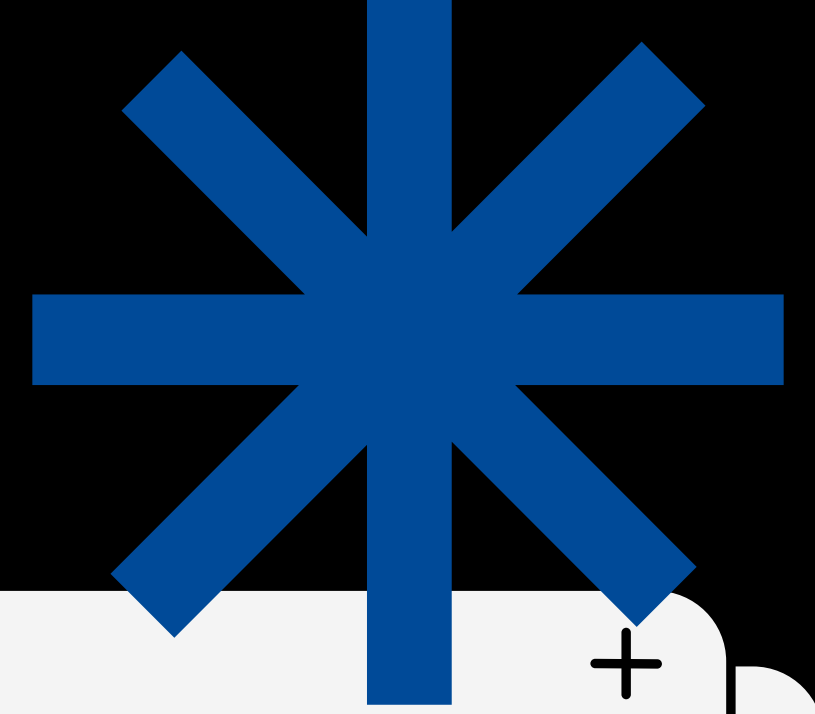
-----Actual-----

startseq little girl covered in paint sits in front of painted rainbow with her hands in bowl endseq
startseq little girl is sitting in front of large painted rainbow endseq
startseq small girl in the grass plays with fingerpaints in front of white canvas with rainbow on it endseq
startseq there is girl with pigtails sitting in front of rainbow painting endseq
startseq young girl with pigtails painting outside in the grass endseq

-----Predicted-----

startseq three children are standing in front of large flower bars endseq





Teşekkürler