گزارش پروژه یادگیری عمیق

نویسنده: رضا کریم‌زاده

شماره دانشجویی: 98206234

استاد درس: دکتر فاطمی زاده

زمستان 98

فهرست

فهرست اشکال

1 بخش اول برچسب زدن تصاویر

برای این بخش در مرحله‌ی اول ابتدا فایل‌های images_info.json، labels.json و categories_info.json خوانده شد و با استفاده از نام و id که در این فایل‌ها قرار دارد، 5000 داده پس از تغییر سایز به 224*224*3 در یک بردار ذخیره گردید. سپس لیبل‌های مربوط به هر تصویر در یک بردار جداگانه قرار داده شد که متناظر با بردار تصاویر است. به عنوان مثال یک تصویر و شماره‌ی لیبل‌های آن به شکل زیر است.

شکل 1-1 تصویری از دیتاست و شماره ی لیبل ها

در مرحله‌ی بعد چون تصـاویر چند لیبلی هستند لازم است آن‌ها باینری سازی مولتی لیبل شوند، نتیجه به صورت زیر برای تصویر بالا حاصل می‌شود.

```
print(multi_onehot[0])
print(mlb.classes_)

[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1
 1 1 0 0 0 0]
[ 1  2  3  4  5  6  7  8  9 10 11 13 14 15 16 17 18 19 20 21 22 23 24 25
 27 28 31 32 33 34 35 36 37 38 39 40 41 42 43 44 46 47 48 49 50 51 52 53
 54 55 56 57 58 59 60 61 62 63 64 65 67 70 72 73 74 75 76 77 78 79 80 81
 82 84 85 86 87 88 89 90]
```

در مرحله‌ی بعد 10 درصد از داده‌ها را به عنوان داده‌ی تست جدا و مابقی را برای آموزش می‌گذاریم.

```
train size:  4500
test size:  500
class Num:  80
```

در نهایت برای آموزش شبکه‌ی برچسب گذار با استفاده از روش transfer learning مدل از پیش آماده شده‌ی MobileNetV2 را بدون قسمت‌های آخر که وظیفه‌ی لیبل گذاری را دارد، لود می‌کنیم. تمامی وزن‌های این قسمت freeze می‌شود. سپس قسمت لیبل گذاری را با توجه به نیاز خود طبق 80 کلاس داده شده از نو طراحی می‌شود. خلاصه‌ی شبکه‌ی طراحی شده به شکل زیر است.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Model) (None, 7, 7, 1280)        2257984
_________________________________________________________________
flatten_1 (Flatten)          (None, 62720)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               16056576
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 80)                10320
=================================================================
Total params: 18,357,776
Trainable params: 16,099,792
Non-trainable params: 2,257,984
```

در مرحله‌ی آخر با استفاده از تابع هزینه‌ی binary_crossentropy و تعداد ایپاک 50 و بهینه ساز Adam با ضریب یادگیری 1e-4 به نتایج زیر دست یافتیم.

```
Epoch 49/50
4500/4500 [==============================] - 8s 2ms/step - loss: 0.0044 - acc: 0.9998 - val_loss: 0.1248 - val_acc: 0.9715
Epoch 50/50
4500/4500 [==============================] - 8s 2ms/step - loss: 0.0043 - acc: 0.9998 - val_loss: 0.1249 - val_acc: 0.9717
Test loss: 0.12492065433883667
Test accuracy: 0.971650201225281
```

یک نمونه از تصویر لیبل گذاری شده:

شکل 2-1 نمونه تصویر لیبل گذاری شده

2 بخش دوم تولید کپشن

در این قسمت هم در ابتدا داده‌ها آموزش که شامل 5000 تصویر که هر تصویر تقریبا دارای 5 کپشن است را بارگذاری می‌کنیم و پس از حذف علایم نگارشی اضافه و تبدیل حروف بزرگ به کوچک کپشن های متناظر هر تصویر را در یک لیست قرار می‌دهیم.

در نهایت تصاویر را به سایز 224*224*3 تبدیل می‌کنیم تا در استخراج ویژگی توسط شبکه‌ی cnn به مشکل برخورد نکنیم.

شکل 2-1 یک نمونه تصویر با کپشن متناظر

در مرحله‌ی بعدی پیش پردازش لازم اســت کپشــن‌ها را هم طول کنیم و در ابتدا و انتهای هرکدام یک عبارت که نشــان‌دهنده‌ی آغاز و پایان جمله اســت اضافه کنیم. اکنون لازم است هر کپشن برای خورانده شــدن به شبکه به صورت عددی تبدیل شود، برای این کار از tokenizer استفاده می‌کنیم تا هر کلمه را به یک عدد متناظر نســبت دهد. در نهایت برای یک نمونه کپشــن عددی شــده‌ی متناظر به صــورت زیر است.

```
print(train_seqs[0])
print(new_captions[0])
```

```
[  4   3  47  10 297   3  26  11   3  24   9  29   5   2   2   2   2   2
   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
   2   2   2   2   2   2   2   2   2   2   2   2   2   2]
<start> a room with chairs a table and a woman in it  <end> <pad> <pad> <pad> <pad> <pad>
```

در گام بعد باید شــبکه‌ی cnn آموزش دیده شــده در قســمت قبل را به مدل وارد کنیم و فقط لایه‌های لیبل گذاری را حذف و فقط از لایه‌ی ویژگی برای ایجاد state اولیه برای شبکه‌ی rnn استفاده کنیم.

لایه‌ی ویژگی در این شبکه 1280 تایی است.

```
feature vec. size:  1280
```

نکته: برای سهولت بیشتر و افزایش سرعت در مرحله‌ی آموزش ابتدا کلیه‌ی تصاویر آموزش را به شبکه‌ی استخراج ویژگی می‌دهیم و تمامی بردارهای ویژگی متناظر هر تصویر را ذخیره می‌کنیم.

برای بخش rnn از functional model برای آزادی عمل بیشتر استفاده می کنیم. مدل rnn از یک لایه‌ی Dense، Embedding و سه واحد GRU برای تولید کپشن تشکیل شده است. که لایه‌ی Dense برای فشرده کردن بردار ویژگی استخراج شده از تصویر و لایه‌ی Embeding برای tokenizer قرار دهده شد.

در نهایت با استفاده از روش teacher forcing مدل آموزش دید.

نکته: در این مدل برای تعریف تابع هزینه جست و جوی زیادی شد در keras مدل‌های binary_crossentropy و sparse_categorical_crossentropy وجود دارد که برای داده‌های وان هات شده اکثرا کاربرد دارند ولی در اینجا داده‌های ما وان هات نیست اما یک لاس خوب در tensorflow1 وجود دارد که عملکرد بهتری نسبت به دو لاس دیگر نشان داد.

این لاس به صورت تابعی تعریف شد و در فرایند آموزش مورد استفاده قرار گرفت.

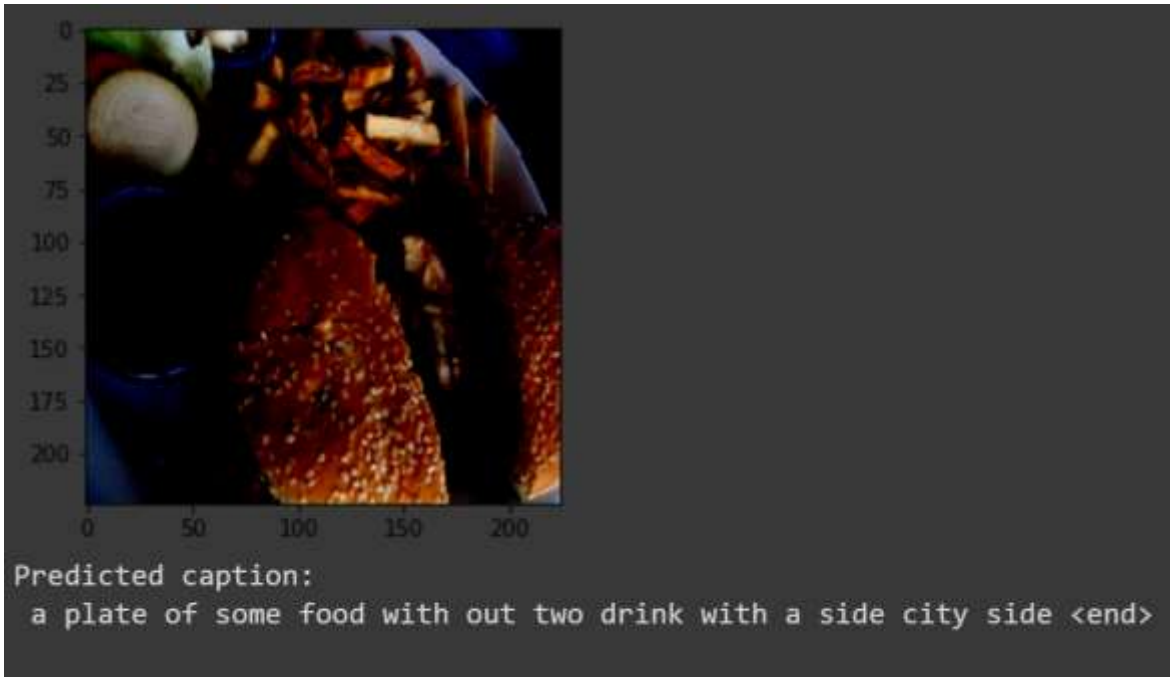
```
# Calculate the loss. This outputs a
# 2-rank tensor of shape [batch_size, sequence_length]
loss = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y_true,
                                                      logits=y_pred)

# Keras may reduce this across the first axis (the batch)
# but the semantics are unclear, so to be sure we use
# the loss across the entire 2-rank tensor, we reduce it
# to a single scalar with the mean function.
loss_mean = tf.reduce_mean(loss)
```

بهینه‌ساز این مدل RMSprop با ضریب یادگیری 1e-3 بود.

```
Epoch 49/50
256/256 [==============================] - 42s 166ms/step - loss: 0.1049
Epoch 50/50
256/256 [==============================] - 41s 162ms/step - loss: 0.1039
```

در نهایت loss پس از 50 ایپاک از 1.27به 0.1 رسید.



شکل 2-2 نمونه کپشن تولید شده