به نام خدا

دانشکده مهندسی کامپیوتر

پروژه پایانی درس مبانی بینایی کامپیوتر

موضوع: خواندن شماره کارت بانکی و شماره ملی

اعضا:

محمد درمانلو

شماره دانشجویی: ۹۹۵۲۱۲۶۲

سید محمد علی فخاری

شماره دانشجویی: ۹۹۵۲۱۴۹۶

استاد درس: دکتر محمدی

تابستان ۱۴۰۲

برش کارت:

برای اینکه بتوانیم اطلاعات روی کارت بانکی یا ملی را بخوانیم، ابتدا باید محدوده مشخص به کارت را درون تصویر مشخص کرده و با استفاده از تبدیلات هندسی آن را به صورت واضحی دربیاوریم.

برای اینکار چندین راه وجود دارد که در این پروژه روش YOLO, semantic segmentation و روش های پردازش تصویر ساده پیاده سازی شده است.

برای پیاده سازی روش یولو از دیتاستی که خودمان آماده کرده بودیم و دیتاستی که در سایت roboflow پیدا کردیم، استفاده کردیم. برای لیبل زدن برای داده های خودمان هم از سایت roboflow کمک گرفتیم. نتیجه این روش به طور کلی خوب بود اما به دو دلیل از آن در ادامه پروژه استفاده نکردیم: ۱- محدوده ای که برای کارت در نظر میگرفت برای این مساله مناسب نبود و نمیتوانستم که کارت را به خوبی برش دهم. ۲- برای پیاده سازی و استفاده از نتایج آموزش نیاز داشتم که چندین کتابخانه خارج از کتابخانه های داده شده، به پروژه اضافه کنم.

نمونه خروجی پیاده سازی YOLOv7:

در ادامه سعی به پیاده سازی روش پردازش تصویر پرداختم. در این روش که عمدتا از توابع موجود در کتابخانه open-cv استفاده شده است ابتدا تصویر ورودی را خوانده، نویز آن را برطرف کرده و به حالت gray تبدیل میکنیم. برای رفع نویز در این پروژه عمدتا از تابع cv2.fastNlMeanDenoisin استفاده شده است که عملکرد بهتری نسبت به رفع نویز گاوسی داشت. پس از اینکه نویز تصویر را از بین بردیم، آن را با استفاده از روش آستانه گذاری وفقی، دو سطحی کرده و پیکسل های تصویر را وارون میکنیم(پیکسل صفر به یک و پیکسل یک را به صفر تبدیل میکنیم) پس از این کار برای اینکه بتوانیم کارت را تشخیص دهیم باید تمام مساحت آن را به هم متصل کنیم که این کار با استفاده از عملگر مورفولوژی بسته و عنصر ساختاری دایروی با شعاع ۱۷ انجام میدهیم.

پس از اینکه کارت را تشخیص دادیم با استفاده از تابع canny لبه های کارت را پیدا میکنیم. پس از اینکار به راحتی با استفاده از کانتور های شکل چهار نقطه گوشه کارت را پیدا کرده و در آخر با استفاده از تابع warpPerspective، کارت را برش میدهیم تا اطلاعات آن بهتر مشخص باشد.

مشکلی که در این روش وجود داشت این بود که اگر عکس ورودی زمینه شلوغی داشته باشد نمیتوان به راحتی کارت را تشخیص داد و آن را برش زد. به همین دلیل به سراغ روش semantic-segmentation رفتیم. دیتاستی که برای اینکار استفاده شد، دیتاست MIDV500 هست که مجموعه ای از تصاویر کارت شناسایی در حالات مختلف و از کشور های متفاوت است.

مدل پیاده سازی شده در این قسمت بر پایه معماری U-Net و اسکلت اصلی آن مدل MobileNet می‌باشد.

در ادامه دو خروجی آورده شده که یکی توسط پردازش تصویر ساده و دیگری توسط مدل آموزش داده شده است:

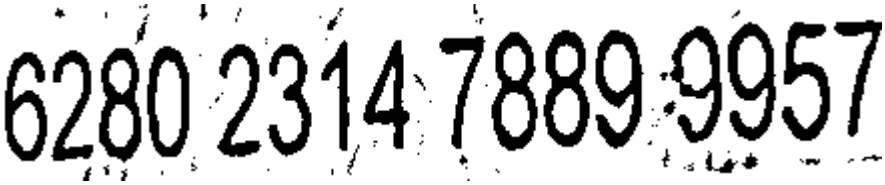
همانطور که در تصویر سمت چپ مشخص است، زمینه این تصویر شلوغ بوده و نمیتوان به راحتی توسط روش های ساده پردازش تصویر آن را برش داد به همین دلیل آن را با استفاده از روش semantic-segmentation برش دادیم.

مشخص کردن اعداد کارت بانکی:

پس از برش کارت باید شماره کارت ۱۶ رقمی آن را تشخیص و با استفاده از مدل آموزش داده شده، بخوانیم. برای اینکار تصویر را خوانده و آن را به سایز ثابت (600, 600) تغییر میدهیم. پس از رفع نویز از تصویر به کمک عملگر های مورفولوژی باز و بسته، آنقدر این کار را تکرار میکنیم تا بتوان تمام ۱۶ رقم شماره کارت را به هم متصل و با استفاده از کانتور ها، مکان آن در کارت را تشخیص دهیم.

پس از شناسایی ۱۶ رقم شماره کارت باید هریک را جداگانه خوانده و به مدل داده تا مقدار پیش بینی آن به دست آید.

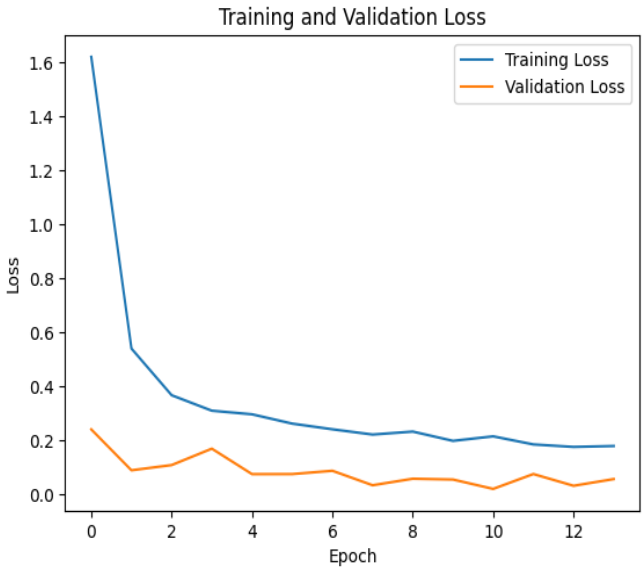
به عنوان مثال با دادن کارت زیر به تابع بالا خروجی به صورت زیر خواهد شد:

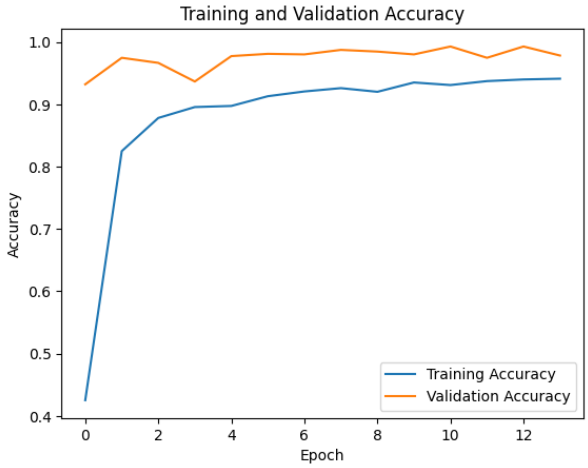


خلاصه مدل آموزش دیده برای تشخیص ارقام کارت بانکی هم به صورت زیر است:

```
Found 9034 images belonging to 10 classes.
Found 1110 images belonging to 10 classes.
Model: "sequential_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_41 (Conv2D)          (None, 32, 32, 32)        2432

 activation_96 (Activation)  (None, 32, 32, 32)        0

 max_pooling2d_33 (MaxPoolin  (None, 16, 16, 32)       0
 g2D)

 conv2d_42 (Conv2D)          (None, 16, 16, 64)        51264

 activation_97 (Activation)  (None, 16, 16, 64)        0

 flatten_18 (Flatten)        (None, 16384)             0

 dense_55 (Dense)            (None, 500)               8192500

 activation_98 (Activation)  (None, 500)               0

 dense_56 (Dense)            (None, 10)                5010

 activation_99 (Activation)  (None, 10)                0

=================================================================
Total params: 8,251,206
Trainable params: 8,251,206
Non-trainable params: 0
_________________________________________________________________
```

نتایج حاصل از آموزش این مدل در طی epoch 20 به صورت زیر است:





در گام بعد باید تابعی بنویسیم تا با گرفتن عکس کارت تشخیص دهد که کارت بانکی است یا کارت اعتباری. با توجه به اینکه در این مساله، تنها دو حالت داریم پس اگر بتوانیم کارتی را تشخیص دهیم که ملی است یا نه تا حدودی نیازمان برطرف می‌شود. برای این مساله، سه لوگوی مهم از کارت ملی را برش داده و به صورت template از آنها برای شناسایی درون یک کارت استفاده میکنیم. در صورتی که مقادیر برگردانده شده توسط تابع template-matching بزرگتر از ۰.۵ بود، کارت ملی و در غیر اینصورت، کارت اعتباری است.

مدل تشخیص اعداد فارسی

در ابتدا تصاویر دیتاست را میخوانیم و ۹۰۰ تصویر به عنوان داده ی ترین و ۱۰۰ تصویر به عنوان داده ی validationدر x_train و x_val ذخیره میکنیم. سپس لیبل های مورد نیاز را هم میسازیم و در y_train و y_valذخیره میکنیم.

در مرحله بعد مدل خود را میسازیم. در این مدل از آنجا که دیتاست کوچکی داریم از انتقال یادگیری استفاده کرده و لایه های ابتدایی resnet50 که بر روی دیتاست imagenet آموزش داده شده است را به عنوان بخش ابتدایی مدل استفاده میکنیم و بعد از آن با یک لایه پولینگ و دو لایه کاملا متصل به ۱۰ نورون پایانی میرسیم. برای آموزش مدل از تابع ضرر categorical_crossentropy و adam optimizer استفاده کردیم و ۳۰ epoch مدل را ترین کردیم.

به دلیل شباهت بالای داده های دیتاست، شبکه به سرعت به دقت های بالا رسید و در epoch های پایانی به دقت ۱۰۰ رسید، که البته برای رفع مشکل دیتاست از data_augmentation هم استفاده کردیم.

```
conv3_block4_2_conv (Conv2D)   (None, 8, 8, 128)    147584      ['conv3_block4_1_relu[0][0]']

conv3_block4_2_bn (BatchNormal  (None, 8, 8, 128)   512         ['conv3_block4_2_conv[0][0]']
ization)

conv3_block4_2_relu (Activatio  (None, 8, 8, 128)   0           ['conv3_block4_2_bn[0][0]']
n)

conv3_block4_3_conv (Conv2D)   (None, 8, 8, 512)    66048       ['conv3_block4_2_relu[0][0]']

conv3_block4_3_bn (BatchNormal  (None, 8, 8, 512)   2048        ['conv3_block4_3_conv[0][0]']
ization)

conv3_block4_add (Add)         (None, 8, 8, 512)    0           ['conv3_block3_out[0][0]',
                                                                 'conv3_block4_3_bn[0][0]']

conv3_block4_out (Activation)  (None, 8, 8, 512)    0           ['conv3_block4_add[0][0]']

sequential_2 (Sequential)      (None, 8, 8, 512)    0           ['conv3_block4_out[0][0]']

global_average_pooling2d_1 (Gl  (None, 512)         0           ['sequential_2[0][0]']
obalAveragePooling2D)

dense_4 (Dense)                (None, 512)          262656      ['global_average_pooling2d_1[0][0
                                                                ]']

dense_5 (Dense)                (None, 10)           5130        ['dense_4[0][0]']

==================================================================================================
Total params: 1,727,882
Trainable params: 267,786
Non-trainable params: 1,460,096
```

دیتاست های استفاده شده:

اعداد انگلیسی https://www.kaggle.com/datasets/yaswanthgali/english-fontnumber-recognition

اعداد فارسی https://www.kaggle.com/datasets/mehdisahraei/persian-alpha

با توجه به حجیم بودن پوشه پروژه در google drive، لینک آن را به اشتراک میگذارم.

https://drive.google.com/drive/folders/1g_cGSMvIHBOb9GEFWTcwXRhgPH-vS6u9?usp=drive_link