علی آراسته ۹۶۱۰۱۱۶۵

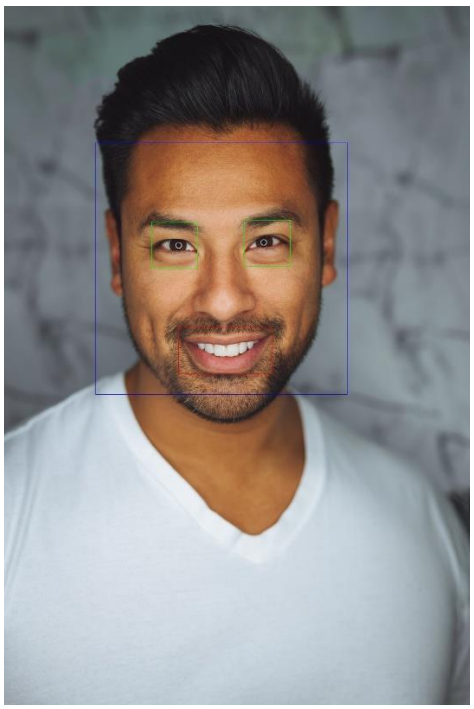
دید کامپیوتری – تمرین سوم

الف) تمارین کامپیوتری:

۱) ابتدا با استفاده از دستور cv2.CascadeClassifier، Classifier های جداگانه‌ای برای صورت، چشم و لبخند ایجاد می‌کنیم. سپس با استفاده از دستور detectMultiScale، برای هر یک از Calssifier های ساخته شده، مختصات مربع‌های در برگیرنده اجزای دلخواه را بدست آورده، سپس آن را در تصویر اصلی رسم می‌کنیم. (برای تشخیص دقیق، برای هر تصویر دو پارامتر scaleFactor و minNeighbors به طور مناسب تنظیم می‌کنیم)

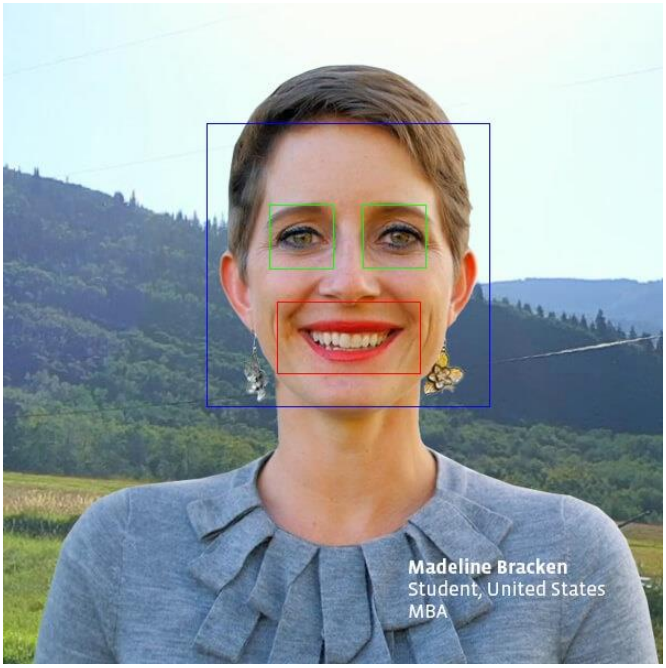
تصویر ۱:

تصویر ۲:

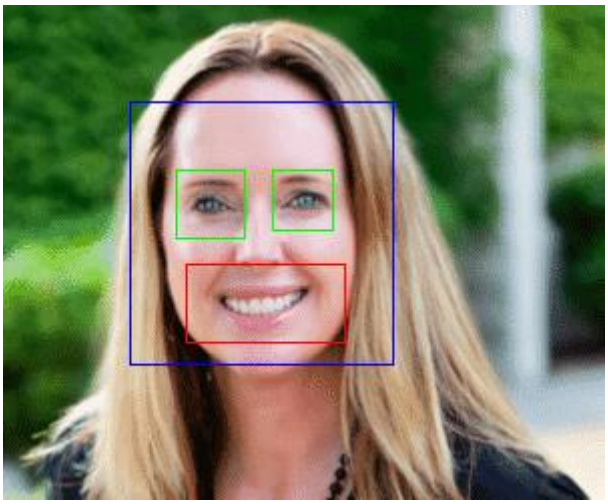


تصویر ۳:

تصویر ۴:

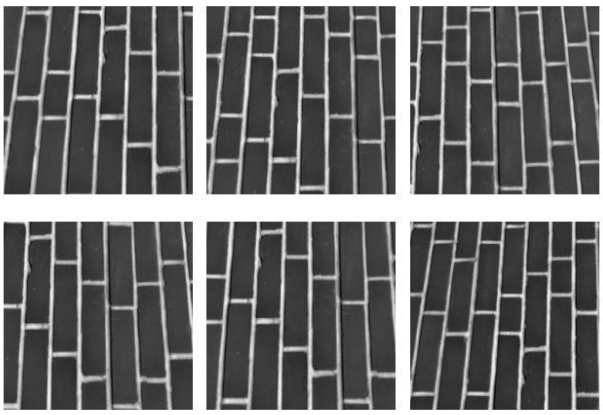


تصویر ۵:

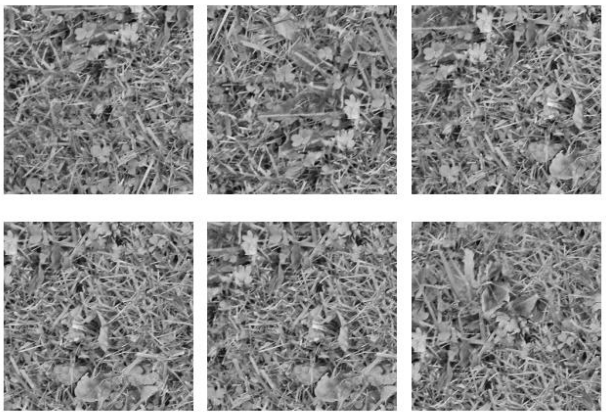


۲) ابتدا با استفاده از کد تمرین قبل، از بخش data از skimage برای هر یک از آجر، علف و شن، ۶ تصویر استخراج کرده، در تصویری نمایش می‌دهیم.

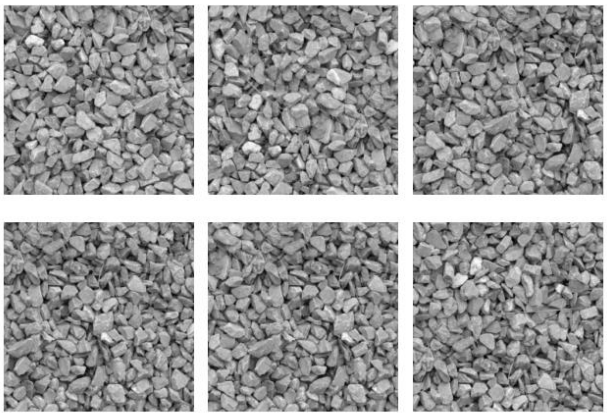
آجر:



علف:



شن:



سپس با استفاده از تحلیل هیستوگرام و با توصیفگر LBP (Local Binary Pattern)، برای هر یک توصیفگری می‌سازیم. نحوه عملکرد توصیفگر LBP به این صورت است که مقدار هر پیکسل را با مقدار پیکسل‌های اطراف آن، که با آن مقدار مشخصی فاصله دارند، مقایسه می‌کند، به هر کدام از این پیکسل‌ها بر اساس کوچکتر یا بزرگتر بودن مقدار آن نسبت به پیکسل اصلی، یکی از دو مقدار صفر یا یک را نسبت می‌دهد و در آخر برای کنار هم قرار دادن این مقادیر به پیکسل مرکزی عددی باینری نسبت می‌دهد. سپس هیستوگرامی با تعداد بین مشخص از مقادیر نسبت داده شده به همه پیکسل‌ها می‌سازد که از آن به عنوان توصیفگر بافت استفاده می‌شود.

تابع local_binary_pattern سه پارامتر کنترلی دارد. پارامتر اول تعداد نقاطی است که برای ساختن ویژگی هر پیکسل استفاده می‌شوند. این نقاط روی یک دایره انتخاب می‌شوند. پارامتر دوم تابع شعاع این دایره است. پارامتر سوم نیز نحوه توزیع نقاط روی این دایره را تعیین می‌کند.

با توجه به متفاوت بودن بافت این سه تصویر، هیستوگرام های آن‌ها نیز متفاوت خواهد بود و در نتیجه می‌توان با استفاده از این هیستوگرام‌ها، این سه نوع تصویر را به خوبی طبقه‌بندی کرد.

در این سوال از روش kullback_score برای تعیین میزان شبیه بودن هیستوگرام تصویر ورودی به هیستوگرام سه بافت آجر، علف و شن استفاده می‌کنیم. سپس بر اساس این مقدار طبقه‌بندی را انجام می‌دهیم.

۳)

الف) ابتدا با استفاده از دستور loadmat از scipy.io، داده‌ها را بارگذاری می‌کنیم. سپس تغییراتی روی فرمت این تصاویر انجام می‌دهیم تا برای پردازش‌های بعدی آماده شوند. در خروجی این مرحله برای هر یک از داده‌های آموزش و تست دو ماتریس ایجاد می‌شود که یکی حاوی تصاویر و دیگری حاوی label تصاویر است.

ب) در این قسمت ایتدا بدون استفاده از dropout، ۵ شبکه کانولوشنی مختلف از ۲ تا ۶ لایه طراحی می‌کنیم و هر یک را با ۲۰ Epoch آموزش می‌دهیم تا بهترین شبکه مشخص شود.

شبکه اول: این شبکه دارای دو لایه است. لایه اول دارای ۳۲ فیلتر ۵*۵ و لایه دوم دارای ۶۴ فیلتر ۳*۳ است. بعد از هر لایه کانولوشنی، از یک BatchNormalization و در ادامه از یک MaxPool با گام حرکت (۲و۲) استفاده می‌کنیم تا نتایج بهتری حاصل شود. در آخر با استفاده از یک Flatten خروجی را به یک بردار تبدیل می‌کنیم سپس با استفاده از دو لایه FullyConnected خروجی را به ۱۰ مقدار تبدیل می‌کنیم.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        2432
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0
_________________________________________________________________
batch_normalization_3 (Batch (None, 16384)             65536
_________________________________________________________________
dense_1 (Dense)              (None, 100)               1638500
_________________________________________________________________
batch_normalization_4 (Batch (None, 100)               400
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1010
=================================================================
Total params: 1,726,502
Trainable params: 1,693,470
Non-trainable params: 33,032
```

بعد از آموزش نتایج زیر بدست می‌آید:

```
loss: 0.0150 - accuracy: 0.9952 - val_loss: 0.7239 - val_accuracy: 0.8845
```

شبکه دوم: این شبکه دارای سه لایه است. لایه اول دارای ۳۲ فیلتر ۵*۵، لایه دوم دارای ۶۴ فیلتر ۵*۵ و لایه سوم دارای ۱۲۸ فیلتر ۳*۳ است. بعد از هر لایه کانولوشنی، از یک BatchNormalization و در ادامه از یک MaxPool با گام حرکت (۲و۲) استفاده می‌کنیم تا نتایج بهتری حاصل شود. در آخر با استفاده از یک Flatten خروجی را به یک بردار تبدیل می‌کنیم سپس با استفاده از دو لایه FullyConnected خروجی را به ۱۰ مقدار تبدیل می‌کنیم.

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 32)        2432
_________________________________________________________________
batch_normalization_5 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        51264
_________________________________________________________________
batch_normalization_6 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_7 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)              0
_________________________________________________________________
batch_normalization_8 (Batch (None, 2048)              8192
_________________________________________________________________
dense_3 (Dense)              (None, 100)               204900
_________________________________________________________________
batch_normalization_9 (Batch (None, 100)               400
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1010
=================================================================
Total params: 342,950
Trainable params: 338,206
Non-trainable params: 4,744
```

بعد از آموزش نتایج زیر بدست می‌آید:

```
loss: 0.0291 - accuracy: 0.9901 - val_loss: 0.4848 - val_accuracy: 0.9017
```

شبکه سوم: این شبکه دارای چهار لایه است. لایه اول و دوم دارای ۳۲ فیلتر ۳*۳، لایه سوم دارای ۳۲ فیلتر ۳*۳ و لایه چهارم دارای ۱۲۸ فیلتر ۳*۳ است. بعد از هر لایه کانولوشنی، از یک BatchNormalization و در ادامه در بعضی از لایه‌ها، از یک MaxPool با گام حرکت (۲و۲) استفاده می‌کنیم تا نتایج بهتری حاصل شود. در آخر با استفاده از یک Flatten خروجی را به یک بردار تبدیل می‌کنیم سپس با استفاده از دو لایه FullyConnected خروجی را به ۱۰ مقدار تبدیل می‌کنیم.

بعد از آموزش نتایج زیر بدست می‌آید:

```
loss: 0.0122 - accuracy: 0.9959 - val_loss: 0.5171 - val_accuracy: 0.9032
```

شبکه چهارم: این شبکه دارای پنج لایه است. لایه اول و دوم دارای ۳۲ فیلتر ۳*۳، لایه سوم وچهارم دارای ۶۴ فیلتر ۳*۳ و لایه پنجم دارای ۱۲۸ فیلتر ۳*۳ است. بعد از هر لایه کانولوشنی، از یک BatchNormalization و در ادامه در بعضی از لایه‌ها، از یک MaxPool با گام حرکت (۲و۲) استفاده می‌کنیم تا نتایج بهتری حاصل شود. در آخر با استفاده از یک Flatten خروجی را به یک بردار تبدیل می‌کنیم سپس با استفاده از دو لایه FullyConnected خروجی را به ۱۰ مقدار تبدیل می‌کنیم.

بعد از آموزش نتایج زیر بدست می‌آید:

```
loss: 0.0230 - accuracy: 0.9919 - val_loss: 0.4422 - val_accuracy: 0.9223
```

شبکه پنجم: این شبکه دارای شش لایه است. لایه اول و دوم دارای ۳۲ فیلتر ۳*۳، لایه سوم وچهارم دارای ۶۴ فیلتر ۳*۳ و لایه پنجم و ششم دارای ۱۲۸ فیلتر ۳*۳ است. بعد از هر لایه کانولوشنی، از یک BatchNormalization و در ادامه در بعضی از لایه‌ها، از یک MaxPool با گام حرکت (۲و۲) استفاده می‌کنیم تا نتایج بهتری حاصل شود. در آخر با استفاده از یک Flatten خروجی را به یک بردار تبدیل می‌کنیم سپس با استفاده از دو لایه FullyConnected خروجی را به ۱۰ مقدار تبدیل می‌کنیم.

بعد از آموزش نتایج زیر بدست می‌آید:

```
loss: 0.0345 - accuracy: 0.9878 - val_loss: 0.4465 - val_accuracy: 0.9185
```

پس آخرین مدل بهترین مدل است. در نتیجه می‌توان نتیجه گرفت با افزایش لایه‌های کانولوشنی، دقت شبکه نیز افزایش می‌یابد.

حال به این شبکه dropout اضافه می‌کنیم. این کار باعث می‌شود در هر دور بعضی از نورون خاموش شوند و در نتیجه آموزش نسبت به تغییرات مقاوم‌تر باشد. این کار با مشخص کردن یک احتمال انجام می‌شود. یعنی برای هر نورون عددی تصادفی بین صفر و یک تعیین می‌شود و اگر مقدار این عدد از احتمال مشخص شده کمتر باشد آن نورون در آن دور از آموزش خاموش می‌شود.

به طور تجربی مقدار احتمال dropout را ۰/۲۵ تعیین می‌کنیم.

بعد از آموزش نتایج زیر بدست می‌آید:

```
loss: 0.1137 - accuracy: 0.9637
```

به طور مشخص dropout، دقت شبکه را افزایش می‌دهد.

پ) در این قسمت از شبکه FullyConnected استفاده می‌کنیم. این نوع شبکه را برای ۵ حالت مختلف از ۲ تا ۶ لایه طراحی کرده، سپس آموزش می‌دهیم.

بعد از آموزش نتایج زیر بدست می‌آید:

```
model 1 =  0.06730177998542786
model 2 =  0.7222265005111694
model 3 =  0.718192994594574
model 4 =  0.6498924493789673
model 5 =  0.6626459956169128
```

نتایج بدست آمده مشخص می‌کند شبکه‌ای که دارای ۳ لایه است بهترین نتیجه را داشته است.

دقت شبکه‌های کانولوشنی به طور معناداری از شبکه‌های FullyConnected بیشتر است. علت اصلی این است که با تبدیل کردن تصویر ۲ بعدی به بردار ۱ بعدی مقدار زیادی از اطلاعات از بین می‌رود.

همچنین در شبکه‌های FullyConnected تعداد پارامترها به شدت بیشتر است در نتیجه از جایی به بعد افزایش لایه‌های شبکه، دقت آموزش را کاهش داده است. در نتیجه استفاده از شبکه‌های کانولوشنی معقول‌تر است.

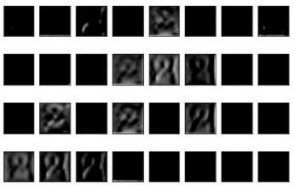
ت) تنها فیلترهای لایه اول قابل نمایش دادن هستند زیرا در لایه‌های بعدی، عمق فیلتر بزرگتر از ۳ است.

فیلترهای لایه اول بهترین شبکه کانولوشنی:

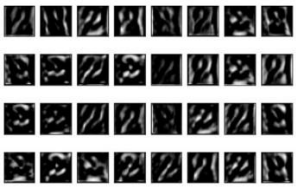
همچنین می‌توان feature map های مربوط به یک تصویر در هر لایه شبکه کانولوشنی را نمایش داد.

به عنوان مثال تصویر زیر را انتخاب می‌کنیم:

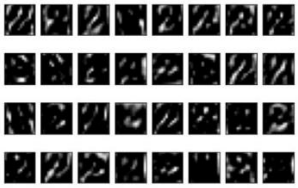
feature map لایه اول بهترین شبکه کانولوشنی:



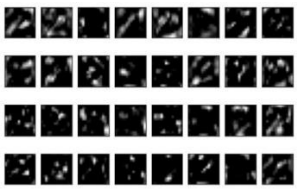
feature map لایه دوم بهترین شبکه کانولوشنی:



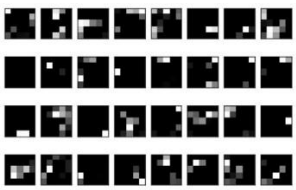
feature map لایه سوم بهترین شبکه کانولوشنی:



feature map لایه چهارم بهترین شبکه کانولوشنی:



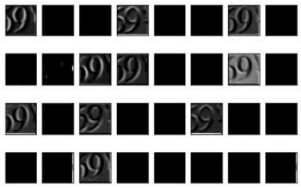
feature map لایه پنجم بهترین شبکه کانولوشنی:



همچنین برای دو تصویر دیگر feature map های لایه اول شبکه کانولوشنی را رسم می‌کنیم.

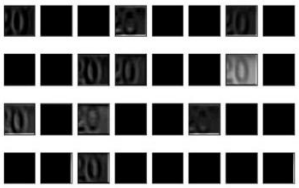
نمونه اول:

feature map لایه اول بهترین شبکه کانولوشنی:



نمونه دوم:

feature map لایه اول بهترین شبکه کانولوشنی:

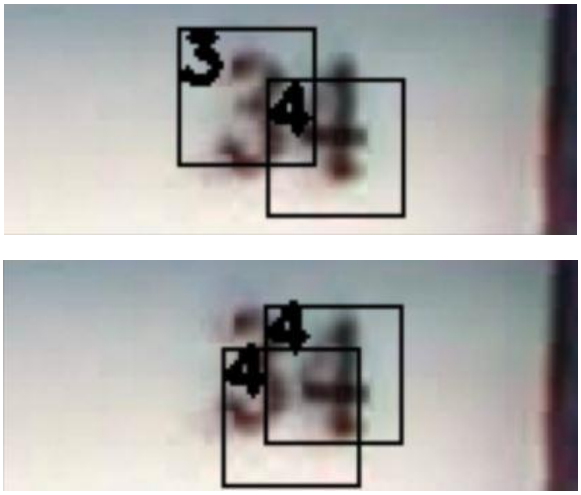


ث) در این قسمت با استفاده از تشکیل هرم و پنجره لغزان و قرار دادن یک threshold، پنجره‌هایی را که به احتمال زیاد حاوی عدد هستند، در اسکیل‌های متفاوت شناسایی می‌کنیم. سپس با استفاده از Non.Maximum.Supperssion، پنجره‌های نزدیک به هم را به یک پنجره تبدیل می‌کنیم. در آخر پنجره‌های باقی‌مانده را روی تصویر اصلی رسم کرده، برای هر یک label مناسب قرار می‌دهیم.

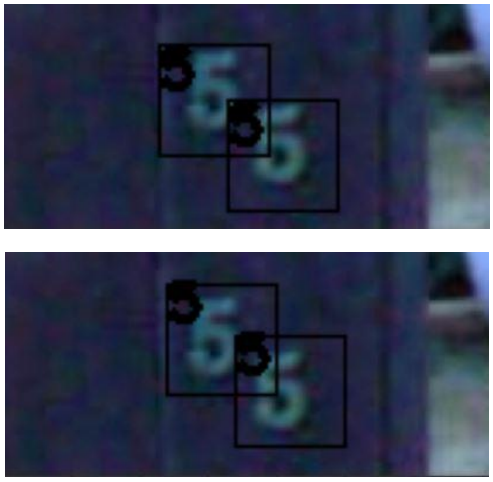
تصاویر اول مربوط به شبکه کانولوشنی و تصاویر دوم مربوط به شبکه FullyConnected است.

تصویر اول:

تصویر دوم:

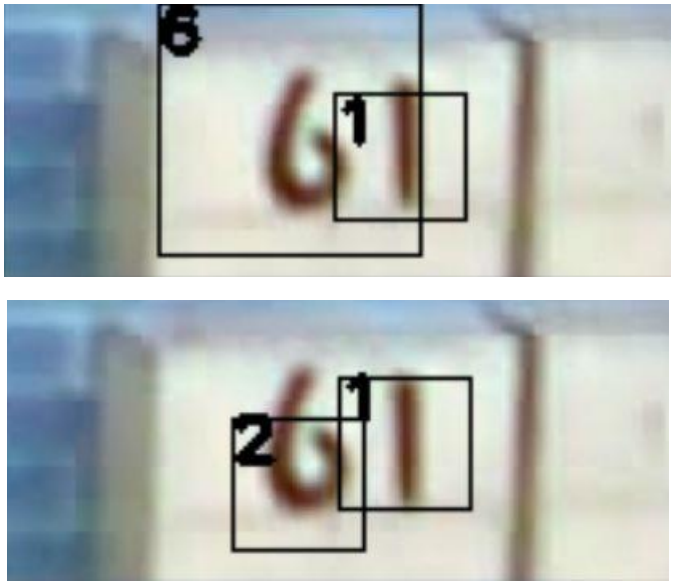


تصویر سوم:



تصویر چهارم:

تصویر پنجم:



به وضوح شبکه کانولوشنی دقت بیشتری دارد.

تمارین پژوهشی:

Video inpainting aims to fill spatio-temporal holes with plausible content in a video. Despite tremendous progress of deep neural networks for image inpainting, it is chal- lenging to extend these methods to the video domain due to the additional time dimension. In this work, we propose a novel deep network architecture for fast video inpainting. Built upon an image-based encoder-decoder model, our framework is designed to collect and refine information from neighbor frames and synthesize still-unknown regions. At the same time, the output is enforced to be temporally consistent by a recurrent feedback and a temporal memory module.Video inpainting can help numerous video editing and restoration tasks such as undesired object removal, scratch or damage restoration, and retargeting.

A straightforward way to perform video inpainting is to apply image inpainting on each frame individually. However, this ignores motion regularities coming from the video dynamics, and is thus incapable of estimating non-trivial appearance changes in image-space over time. Moreover, this scheme inevitably brings temporal inconsistencies and causes severe flickering artifacts.

This article method:

1. Cast video inpainting as a sequential multi-tosingle frame inpainting task and present a novel deep 3D-2D encoder-decoder network. This method effectively gathers features from neighbor frames and synthesizes missing content based on them.

2. Use a recurrent feedback and a memory layer forthe temporal stability. Along with the effective network design, enforce strong temporal consistency via two losses: flow loss and warping loss.

3. Provide a single, unified deep network for the general video inpainting task. conduct extensive subjective and objective evaluations and show its efficacy. Moreover, apply this method to video retargeting and superresolution tasks, demonstrating favorable results.

Most recent methods found in the literature address these issues using either object-based or patch-based approaches.

In object-based methods, a pre-processing is required to split a video into foreground objects and background, and it is followed by an independent reconstruction and merging step at the end of algorithms.

In patch-based methods, the patches from known regions are used to fill in a mask region.

Another article: Copy-and-Paste Networks for Deep Video Inpainting

present a novel deep learning-based algorithm for video inpainting.

This article method:

The system takes a video (X) annotated with the missing pixels (M) in each frame and outputs (Y) the completed video. The video is processed frame-by-frame in the temporal order. Call the frame to be filled as the target frame and the other frames as the reference frames. For each target frame, our network completes the missing region by copying-andpasting contents from the reference frames. To complete a target frame, each reference frame is first aligned to the target frame through the alignment network. Then in the copy network, pixels to be copied from the aligned reference frames are determined by the context matching module. Finally, the outputs from the copy networks are decoded to produce inpainted target frame in the paste network. The input video in the memory is updated with the completed frame, which will subsequently be used as a reference frame, providing more information for the following frames.