

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

گزارش پروژه پایانی درس تصویر پردازی رقمی

استاد مربوطه:

دکتر مهدی امیری

ارائه کنندگان:

فاطمه صالح نیا

نفیسه جعفری

تابستان ۹۸

سرفصل مطالب

مقدمه.....	۵
۱ فصل اول: سری های زمانی.....	۶
۱-۱ سری های زمانی چیست؟.....	۶
۲-۱ روش های تحلیل برای سری زمانی.....	۶
۳-۱ کاربرد ها.....	۶
۲ فصل دوم: معرفی روش SSIM.....	۷
۱-۲ دید کلی.....	۷
۲-۲ فرمول.....	۷
۳-۲ مثال.....	۸
۴-۲ کد.....	۹
۳ فصل سوم: معرفی روش PSNR.....	۱۰
۳-۱ دید کلی.....	۱۰
۲-۳ تعریف.....	۱۱
۱-۳ مثال.....	۱۴
۴ فصل چهارم: Recurrence plot.....	۱۴
۱-۴ پس زمینه.....	۱۴
۲-۴ تعریف.....	۱۵
۵ فصل پنجم: کار با دیتاست پروژه.....	۱۵
۱-۵ دیتاست و پردازش آن.....	۱۵
۲-۵ بلوک دیاگرام مربوط به پروژه.....	۱۶
۳-۵ ایجاد recurrence plot برای هر پنجره.....	۱۷
۴-۵ کد مربوط به تعیین پنجره و ایجاد تصاویر.....	۱۷

۶ فصل ششم: روش های استفاده شده در تعیین شباهت.....	۲۰
۶-۱ استفاده از شبکه عصبی کانولوشن.....	۲۰
۶-۱-۱ معرفی شبکه های عصبی کانولوشن.....	۲۰
۶-۲ معماری CNN.....	۲۱
۶-۳ استفاده از روش ssim.....	۲۳
۶-۴ استفاده از روش psnr.....	۲۳
۶-۵ ارزیابی نتایج.....	۲۴
۷ فصل هفتم: کارهای جانبی.....	۲۵
۷-۱ استفاده از یک شبکه با استفاده از tensorflow.....	۲۵
۸ فصل هفتم: کارهای آینده.....	۲۵
۹ مراجع.....	۲۵

فهرست شکل ها

- شکل ۱: $MSE = 144$ $SSIM = 0.913$ ۸
- شکل ۲: $MSE = 144$ $SSIM = 0.840$ ۸
- شکل ۳: $MSE = 144$ $SSIM = 0.694$ ۹
- شکل ۴: تصاویر مرتبط با کد ارائه شده به عنوان مثال $ssim$ ۹
- شکل ۵: مثال برای رابطه $PSNR$ ۱۴
- شکل ۶: مثال برای recurrence plot برای یک موج سینوسی و یک موج سینوسی به علاوه یک سیگنال خطی ۱۵
- شکل ۷: ایجاد پنجره روی داده ها ۱۶
- شکل ۸: فایل های ایجاد شده recurrence plot برای هر پنجره به تفکیک برچسب ۱۹
- شکل ۹: نمایش چند پنجره به صورت recurrence plot ۱۹
- شکل ۱۰: معماری CNN اولیه به کار برده شده ۲۱
- شکل ۱۱: نمایش خروجی شبکه اولیه ۲۱
- شکل ۱۲: استفاده از یک شبکه پیشرفته برای بررسی امکان بهبود دقت ۲۲
- شکل ۱۳: نتایج حاصل از استفاده از شبکه پیچیده تر ۲۲
- شکل ۱۴: نتایج بدست آمده مربوط به استفاده از پنجره با اندازه ۱۰ ۲۳
- شکل ۱۵: میزان دقت بدست آمده در روش $ssim$ ۲۳
- شکل ۱۶: نمایش نتیجه خروجی به روش $psnr$ با حد تعیین شده ۱۸ ۲۴

فهرست جداول

- جدول ۱: نتایج مقایسه روش های استفاده شده ۲۴

مقدمه

در این پروژه ما می خواهیم به کمک دیتاست ارائه شده به صورت سری زمانی، حالت بعدی سری های مشابه را پیش بینی کنیم. از آنجایی که تحلیل داده ها به صورت مصور، می تواند نتایج بهتری را ارائه کند، به این صورت که یافتن شباهت تصاویر، از بررسی یک سری زمانی راحت تر است، ابتدا به ساخت تصاویر به روش ایجاد recurrence plot از تصاویر، می پردازیم. در مرحله بعد، با استفاده از چند روش مختلف، میزان شباهت تصاویر آموزشی را با تصاویر تست بررسی می کنیم. در نهایت برچسب تصویری که بیشترین شباهت را با تصویر تست داشته باشد، به عنوان برچسب داده تست تعیین می شود. بر همین مبنا، میزان دقت در یک مجموعه تصویر تست به دست آمده که همان میزان دقت روش است. پس از ارائه چند روش از روش های تعیین شباهت تصاویر، به ارائه نتایج و بررسی آنها می پردازیم.

۱ فصل اول: سری های زمانی

۱-۱ سری های زمانی چیست؟

در علوم مختلف، به یک توالی یا دنباله از متغیرهای تصادفی که در فاصله های زمانی ثابت نمونه برداری شده باشند، اصطلاحاً سری زمانی یا پیشامد تصادفی در مقطع زمان می گویند. به عبارت دیگر منظور از یک سری زمانی مجموعه ای از داده های آماری است که در فواصل زمانی مساوی و منظمی جمع آوری شده باشند. روش های آماری که این گونه داده های آماری را مورد استفاده قرار می دهد مدل های تحلیل سری زمانی نامیده می شود. مانند فروش فصلی یک شرکت طی سه سال گذشته. یک سری زمانی مجموعه مشاهدات تصادفی ای است که بر اساس زمان مرتب شده باشند. مثال های آن در اقتصاد و حتی رشته های مهندسی دیده می شود.

۱-۲ روش های تحلیل برای سری زمانی

روش های تحلیل سری زمانی به دو دسته تقسیم می شوند: روش های دامنه فرکانس و روش های دامنه زمان. دسته اول شامل تحلیل طیفی و تحلیل موجک و دسته دوم شامل تحلیل های خودهمبستگی و همبستگی متقابل است. افزون بر این می توان روش های تحلیل سری زمانی را به دو دسته پارامتری و ناپارامتری تقسیم کرد. در روش های پارامتری چنین انگاشته می شود که فرایند مانای احتمالاتی دارای ساختاری مشخص است که می توان آن را با تعداد اندکی پارامتر (از جمله با استفاده از مدل خود همبسته یا میانگین متحرک) توصیف کرد. در این روش ها هدف تخمین پارامترهای مدلی است که فرایند احتمالاتی را توصیف می کند. در مقابل، روش های ناپارامتری صریحاً کوواریانس یا طیف فرایند را بدون در نظر گرفتن ساختاری مشخص برای آن تخمین می زنند. همچنین می توان روش های تحلیل سری زمانی را به دسته روش های خطی و غیر خطی یا روش های تک متغیره و چندمتغیره تقسیم کرد.

۱-۳ کاربردها

تحلیل سری های زمانی در زمینه آمار، اقتصاد، زلزله شناسی، هواشناسی، و ژئوفیزیک، هدف اصلی تلقی می شود. همچنین در زمینه پردازش سیگنال، مهندسی کنترل و مهندسی ارتباطات، برای تشخیص و برآورد سیگنال استفاده می شود. زمینه دیگر کاربرد تحلیل سری زمانی در داده کاوی، تشخیص الگو و تجزیه و تحلیل روش های یادگیری ماشین برای خوشه بندی، طبقه بندی، پرس و جو با محتوا، تشخیص ناهنجاری و همچنین پیش بینی می باشد.

- سری زمانی در اقتصاد، مانند قیمت سهام در روزهای متوالی، صادرات در ماه های متوالی، متوسط درآمد در ماه های متوالی ...
- سری زمانی فیزیک، بویژه در علوم مربوط به آثار جوی، علوم دریایی، فیزیک زمین (ژئوفیزیک).
- سری های زمانی بازاریابی، تجزیه و تحلیل ارقام فروش در هفته یا ماه ها متوالی یک مسئله مهم در تجارت است.

- سری‌های زمانی جمعیت نگاری، اندازه‌گیری سالانه جمعیت با هدف پیش بینی تغییرات جمعیت در مدت زمان ده تا بیست سال آینده.
- فرایندهای دوتایی، سری‌هایی که مشاهدات یکی از دو مقدار که معمولاً با ۰ و ۱ نشان می‌دهند را اختیار کند، که بخصوص در نظریه ارتباطات اتفاق می‌افتد را فرایند دوتایی می‌نامند.
- فرایندهای نقطه ای، نوعی سری زمانی که پیشامدهای رخ داده به طور تصادفی در زمان رخ داده، زمان‌های رخ دادن تصادفات قطارها.

۲ فصل دوم: معرفی روش SSIM

۲-۱ دید کلی

SSIM یا Structural similarity متدی جهت مقایسه تشابه دو تصویر می باشد. فرض کنید تصویری داریم که روی اون تصویر تغییری رو ایجاد می کنیم؛ مثلاً نویز اضافه می کنیم. حالا با استفاده از این متد محاسبه می کنیم که تصویر چقدر نسبت به تصویر اصلی تفاوت پیدا کرده است.

SSIM روشی برای پیش بینی کیفیت درک شده از تلویزیون های دیجیتال و تصاویر سینمایی و همچنین انواع دیگر تصاویر و فیلم های دیجیتال است. مدل اصلی در آزمایشگاه مهندسی تصویر و فیلم (LIVE) در دانشگاه تگزاس در آستین ساخته شد و بیشتر به طور مشترک با آزمایشگاه تصویر محاسباتی (LCV) در دانشگاه نیویورک تهیه و تولید شد. انواع دیگری از این مدل در آزمایشگاه محاسبات تصویر در دانشگاه واترلو ایجاد شده و به صورت تجاری به بازار عرضه شده است.

SSIM برای اندازه گیری شباهت بین دو تصویر استفاده می شود. شاخص SSIM یک معیار مرجع کامل است. به عبارت دیگر، اندازه گیری یا پیش بینی کیفیت تصویر بر اساس یک تصویر اولیه فشرده نشده یا بدون تغییر به عنوان مرجع است. SSIM به منظور بهبود روشهای سنتی مانند حداکثر نسبت سیگنال به نویز (PSNR) و میانگین خطای مربع (MSE) طراحی شده است.

۲-۲ فرمول

$$SSIM(x, y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

فرمول ۱: رابطه SSIM

μ_x میانگین x

μ_y میانگین y

σ_x^2 واریانس x

σ_y^2 واریانس y

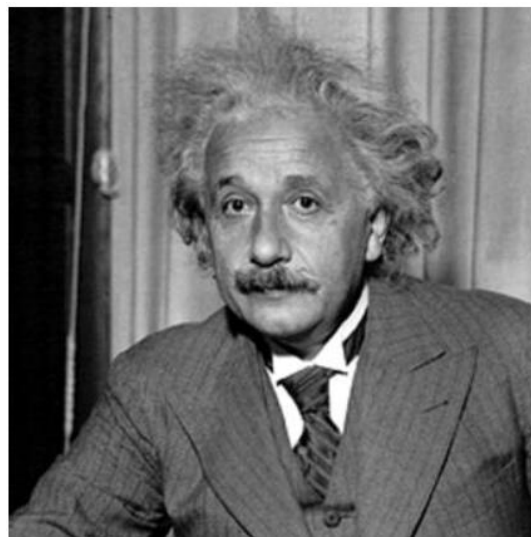
σ_{xy} کواریانس x و y

دو متغیر برای تثبیت تقسیم با مخرج ضعیف (کوچک) $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$

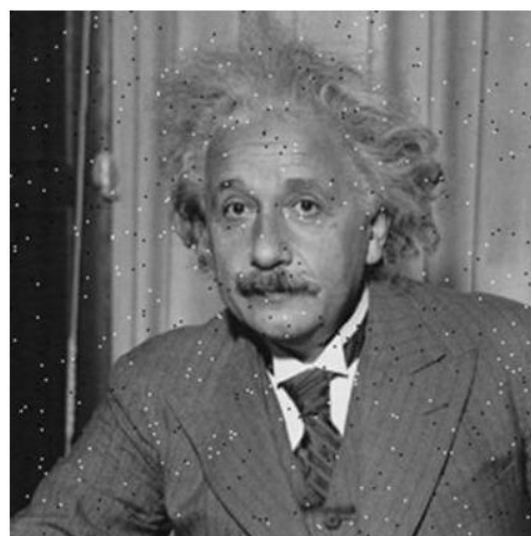
L محدوده پویا برای مقادیر پیکسل

$k_1 = 0.01$ و $k_2 = 0.03$ به صورت پیش فرض قرار دارند.

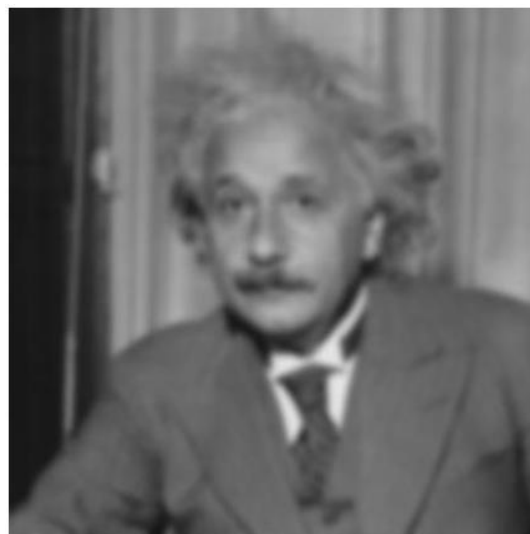
۲-۳ مثال



شکل ۱: $SSIM = 0.913$, $MSE = 144$

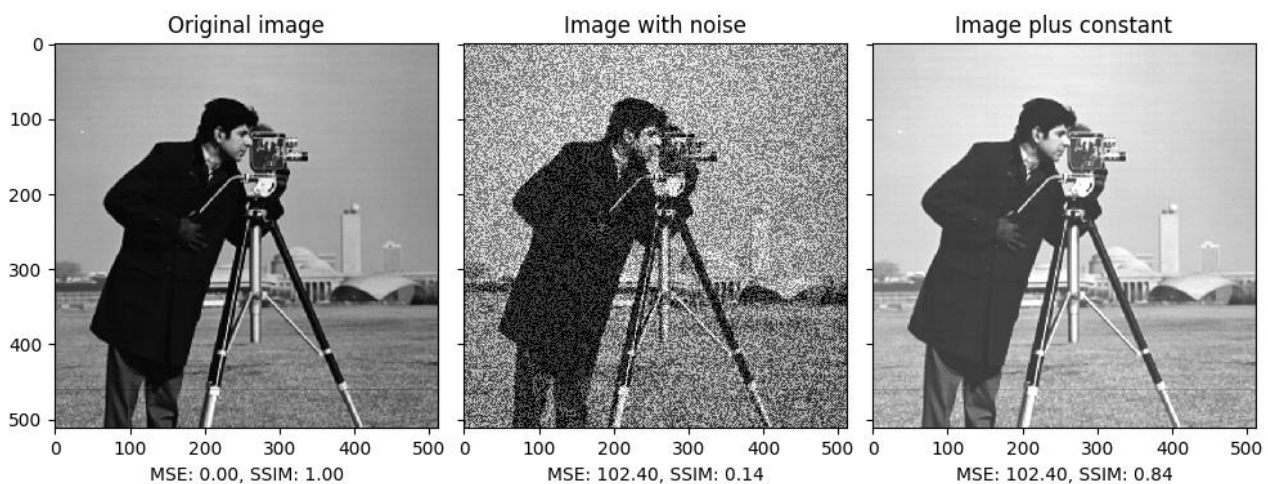


شکل ۲: $SSIM = 0.840$, $MSE = 144$



شکل ۳: $MSE = ۱۴۴$, $SSIM = ۰.۶۹۴$

۲-۴ کد



شکل ۴: تصاویر مرتبط با کد ارائه شده به عنوان مثال *ssim*

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data, img_as_float
from skimage.metrics import structural_similarity as ssim

img = img_as_float(data.camera())
rows, cols = img.shape
noise = np.ones_like(img) * ۰.۲ * (img.max() - img.min())
noise[np.random.random(size=noise.shape) > ۰.۵] *= -۱
def mse(x, y):
    return np.linalg.norm(x - y)
img_noise = img + noise
img_const = img + abs(noise)
```

```
fig, axes = plt.subplots(nrows=۱, ncols=۳, figsize=(۱۰, ۴),
                        sharex=True, sharey=True)
ax = axes.ravel()
mse_none = mse(img, img)
ssim_none = ssim(img, img, data_range=img.max() - img.min())

mse_noise = mse(img, img_noise)
ssim_noise = ssim(img, img_noise,
                  data_range=img_noise.max() - img_noise.min())

mse_const = mse(img, img_const)
ssim_const = ssim(img, img_const,
                  data_range=img_const.max() - img_const.min())
label = 'MSE: {:.۲f}, SSIM: {:.۲f}'

ax[۰].imshow(img, cmap=plt.cm.gray, vmin=۰, vmax=۱)
ax[۰].set_xlabel(label.format(mse_none, ssim_none))
ax[۰].set_title('Original image')
ax[۱].imshow(img_noise, cmap=plt.cm.gray, vmin=۰, vmax=۱)
ax[۱].set_xlabel(label.format(mse_noise, ssim_noise))
ax[۱].set_title('Image with noise')
ax[۲].imshow(img_const, cmap=plt.cm.gray, vmin=۰, vmax=۱)
ax[۲].set_xlabel(label.format(mse_const, ssim_const))
ax[۲].set_title('Image plus constant')

plt.tight_layout()
plt.show()
```

۳ فصل سوم: معرفی روش PSNR

۳-۱ دید کلی

نسبت سیگنال به نویز (SNR - Signal to Noise ratio) معیاری برای نمایش میزان سیگنال مفید در مقابل سیگنال مزاحم (یا نویز) در سیستم‌های الکتریکی است. این عدد، نسبت توان سیگنال به توان نویز است، و آن را بر حسب دسی‌بل (dB) بیان می‌کنند. معمولاً مقدار کمتر از ۱۲ dB نشان‌دهنده مشکل جدی نویز در خطوط انتقال است، مقدار بالاتر از ۲۰ dB رضایت‌بخش، و مقدار بالاتر از ۳۰ dB مناسب است. در واقع این شاخص هرچه بیشتر باشد، وضعیت بهتر بوده و نشان‌دهنده شدت سیگنال مفید بیشتری است.

۲-۳ تعریف

نسبت سیگنال به نویز به عنوان نسبت توان یک سیگنال (اطلاعات معنی دار) به قدرت نویز پس زمینه (سیگنال ناخواسته)

تعریف می شود:

$$SNR = \frac{P_{\text{signal}}}{P_{\text{noise}}},$$

جایی که P توان متوسط است. هر دو توان سیگنال و نویز باید در نقاط یکسان یا معادل آن در یک سیستم و در پهنای باند سیستم یکسان اندازه گیری شوند.

بسته به اینکه سیگنال ثابت باشد (s) یا یک متغیر تصادفی (S)، نسبت سیگنال به نویز برای نویز تصادفی N با مقدار مورد

انتظار صفر می شود:

$$SNR = \frac{s^2}{\sigma_N^2} \quad SNR = \frac{E[S^2]}{\sigma_N^2}$$

جایی که E به مقدار مورد انتظار اشاره دارد، این جا یعنی میانگین S^2

اگر سیگنال و نویز در همان امپدانس اندازه گیری شود، SNR را می توان با محاسبه مربع نسبت دامنه بدست آورد:

$$SNR = \frac{P_{\text{signal}}}{P_{\text{noise}}} = \left(\frac{A_{\text{signal}}}{A_{\text{noise}}} \right)^2$$

که A دامنه میانگین ریشه مربع (RMS) است (برای مثال، ولتاژ RMS).

دسی بل:

از آنجا که بسیاری از سیگنال ها دامنه دینامیکی بسیار گسترده ای دارند، اغلب سیگنال ها با استفاده از مقیاس دسی بل

لگاریتمی بیان می شوند. بر اساس تعریف دسی بل، سیگنال و نویز ممکن است در دسی بل (dB) بیان شود.

$$P_{\text{signal,dB}} = 10 \log_{10}(P_{\text{signal}})$$

$$P_{\text{noise,dB}} = 10 \log_{10}(P_{\text{noise}}).$$

در یک روش مشابه، SNR ممکن است به صورت دسی بل نیز بیان شود.

$$SNR_{\text{dB}} = 10 \log_{10}(SNR).$$

با استفاده از تعریف SNR :

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{noise}}} \right).$$

با استفاده از قانون تعیین کننده برای لگاریتم ها:

$$10 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{noise}}} \right) = 10 \log_{10}(P_{\text{signal}}) - 10 \log_{10}(P_{\text{noise}}).$$

جایگزینی تعاریف SNR، سیگنال و نویز در دسی بل در معادله فوق منجر به فرمول مهمی برای محاسبه نسبت سیگنال به نویز در دسی بل می شود، هنگامی که سیگنال و نویز نیز در دسی بل هستند:

$$\text{SNR}_{\text{dB}} = P_{\text{signal,dB}} - P_{\text{noise,dB}}.$$

در فرمول فوق، P در واحدهای قدرت مانند وات (W) یا میلی وات (mW) اندازه گیری می شود و نسبت سیگنال به نویز عدد خالص است.

اما، هنگامی که سیگنال و نویز در ولت (V) یا آمپر (A) اندازه گیری می شود، اندازه گیری دامنه است، برای بدست آوردن مقدار متناسب با توان، ابتدا باید مربع شوند.

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \left[\left(\frac{A_{\text{signal}}}{A_{\text{noise}}} \right)^2 \right] = 20 \log_{10} \left(\frac{A_{\text{signal}}}{A_{\text{noise}}} \right) = (A_{\text{signal,dB}} - A_{\text{noise,dB}}).$$

محدوده پویا:

مفاهیم نسبت سیگنال به نویز و دامنه دینامیکی از نزدیک مرتبط هستند. محدوده دینامیکی نسبت بین قویترین سیگنال سالم در یک کانال و حداقل سیگنال قابل تشخیص را اندازه گیری می کند، که برای بیشتر اهداف سطح نویز است. SNR نسبت بین سطح سیگنال دلخواه (لزوماً قدرتمندترین سیگنال ممکن) و نویز را اندازه گیری می کند. اندازه گیری نسبت سیگنال به سر و صدا نیاز به انتخاب نماینده یا سیگنال مرجع دارد. در مهندسی صدا، سیگنال مرجع معمولاً موج سینوسی در سطح اسمی یا تراز استاندارد است، مانند ۱ کیلوهرتز در +۴ dBu (۱,۲۲۸ ولت RMS).

SNR معمولاً برای نشان دادن نسبت سیگنال به نویز معمولاً انجام می شود، زیرا ممکن است که (نزدیک) نسبت سیگنال به نویز آنی به طور قابل توجهی متفاوت باشد. این مفهوم را می توان به عنوان عادی کردن سطح نویز به ۱ (دسی بل) و اندازه گیری میزان "ایستادگی" سیگنال درک کرد.

فاصله از توان متقابل:

در فیزیک ، میانگین توان یک سیگنال AC به عنوان میانگین مقدار جریان ولتاژ برابر است. برای مدارهای مقاومتی (غیر واکنشی) ، جایی که ولتاژ و جریان در فاز هستند ، این معادل محصول ولتاژ و جریان rms است :

$$P = V_{\text{rms}} I_{\text{rms}}$$

$$P = \frac{V_{\text{rms}}^2}{R} = I_{\text{rms}}^2 R$$

اما در پردازش و ارتباط سیگنال معمولاً چنین چیزی فرض می شود به این صورت است که معمولاً در هنگام اندازه گیری قدرت یا انرژی یک سیگنال ، آن فاکتور شامل نمی شود. این ممکن است باعث ایجاد برخی سردرگمی در بین خوانندگان شود ، اما ضریب مقاومت برای عملیات معمولی که در پردازش سیگنال انجام می شود یا برای نسبت توان محاسباتی قابل توجه نیست. در بیشتر موارد ، قدرت یک سیگنال به سادگی در نظر گرفته می شود.

$$P = V_{\text{rms}}^2 = \frac{A^2}{2}$$

جایی که "A" دامنه سیگنال AC است.

۱-۳ مثال



شکل ۵: مثال برای رابطه PSNR

۴ فصل چهارم: Recurrence plot

Recurrence plot، یک تکنیک پیشرفته برای تحلیل داده های غیرخطی است. این نمودار به صورت یک تصویر یا گرافی از یک ماتریس مربعی است که در آن عناصر ماتریس وابسته به زمانی هستند که در آن ستون ها و ردیف ها با یک جفت معین مطابقت دارند. به کمک این نمودارها، می توان سری زمانی را به تصویر تبدیل کرد. این تبدیل باعث می شود علاوه بر اینکه الگوی سری زمانی قابل مشاهده است، بتوان از سری زمانی به صورت تصاویر ورودی برای شبکه یا هرمدل دیگری استفاده کرد.

۴-۱ پس زمینه

فرآیندهای طبیعی می توانند یک رفتار متمایز داشته باشند، به عنوان مثال، هر فرآیند می تواند دوره های مختلف (به عنوان چرخه های فصلی یا میلانکوویچ)، حتی چرخه های نامنظم (به عنوان ثال، نوسان جنوبی ال نینو) داشته باشد. علاوه بر این، بازگشت حالتها، به این معنی است که پس از مدتی واگرایی، مجدداً این رفتار به الگوی معینی نزدیک می شود. بازگشت حالات در طبیعت مدتهاست که شناخته شده است و در کارهای اولیه نیز مورد بحث قرار گرفته است (به عنوان مثال هنری پوانکار ۱۸۹۰).

تاریخچه

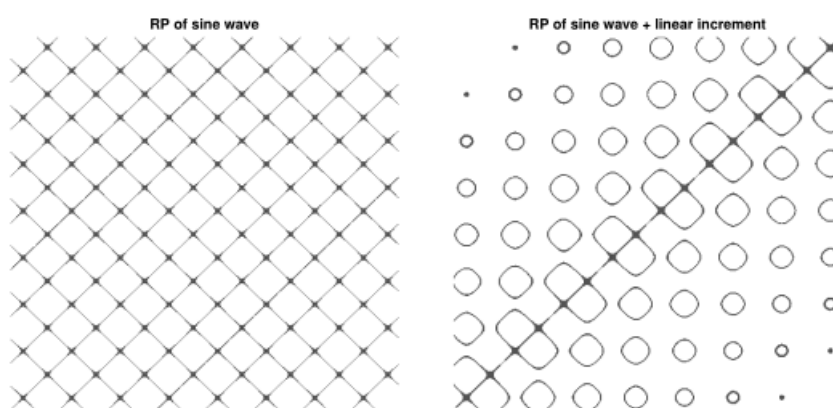
اکمن و همکاران در سال ۱۹۸۷، Recurrence plot را معرفی کردند، که راهی برای تجسم ماهیت دوره ای یک مسیر از طریق یک فضای فازی فراهم می کند. غالباً فضای فاز از ابعاد کافی (دو یا سه) برای تصویربرداری برخوردار نیست، زیرا فضاهای فاز با ابعاد بالاتر فقط با طرح ریزی به زیر فضاهای دو یا سه بعدی قابل مشاهده است. با این حال، ساخت یک Recurrence plot به ما امکان می دهد تا جنبه های مشخصی از مسیر فضای فاز m -بعدی را از طریق یک نمایندگی دو بعدی بررسی کنیم.

۴-۲ تعریف

یک Recurrence، زمانی است که مسیر به مکانی که قبلاً از آن بازدید کرده است، برمی گردد. Recurrence plot مجموعه جفت های زمانی را نشان می دهد که مسیر در همان مکان قرار دارد، یعنی مجموعه (i, j) در حالیکه:

$$\vec{x}(i) = \vec{x}(j)$$

این رابطه می تواند موارد بسیاری از اطلاعات را نشان دهد. را نشان دهد: به عنوان مثال، اگر مسیر کاملاً با دوره T باشد، سپس تمام این جفت های زمانی چندین بار از بقیه فضا جدا می شوند و به عنوان خطوط مورب قابل مشاهده خواهند بود.



شکل ۶: مثال برای recurrence plot برای یک موج سینوسی و یک موج سینوسی به علاوه یک سیگنال خطی

۵ فصل پنجم: کار با دیتاست پروژه

۵-۱ دیتاست و پردازش آن

دیتاست ارائه شده برای انجام این پروژه، به صورت یک سری زمانی در قالب یک فایل csv می باشد که دارای سه ستون است. ستون اول مربوط به شماره سطر، ستون دوم تحت عنوان Thrust و ستون سوم نیز تحت عنوان Duration_nsec ارائه شده است. این دیتاست شامل حدود ۵۳ هزار سطر داده است. هدف، پیش بینی مقدار thrust در دو گام پس از اتمام سری زمانی داده شده می باشد.

در اولین گام بایستی دیتاست را با پنجره های معین تقسیم کرده و به هر پنجره برچسب مربوطه را نسبت داد. بدین منظور از دو اندازه پنجره مختلف ۱۰ و ۱۵ تایی استفاده شده است، برای تعیین برچسب این پنجره ها نیز از ستون مربوط به Thrust، داده دو سطر بعد از اندازه پنجره به عنوان برچسب در نظر گرفته شده است. به عنوان مثال، چنانچه پنجره زیر (به رنگ سبز)، در دیتاست در نظر گرفته شود، مقدار برچسب برابر ۳۱- خواهد بود.

Num	Thrust	Duration_nsec
1	26	456922
2	-38	633052
3	33	524136
4	23	524130
5	77	214501
6	-72	758124
7	83	996325
8	-42	425361
9	30	752013
10	-38	740231
11	23	101061
12	-31	126352

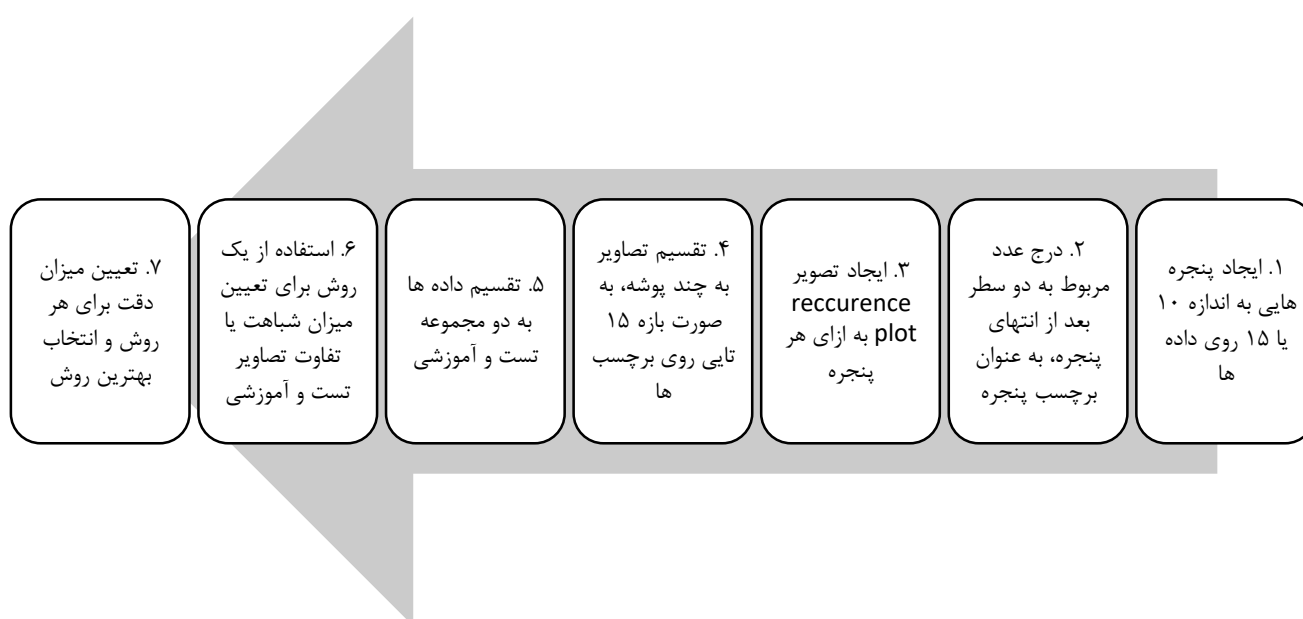
شکل ۷: ایجاد پنجره روی داده ها

با استفاده از روشی که ارائه شد، می توان پنجره هایی با تعداد ۲۰ داده در نظر گرفت. با توجه به اینکه در هر ستون ۱۰ ویژگی در نظر گرفته شود. اما پس از تعیین برچسب های تصاویر، بایستی برچسب ها نیز درون دسته های مشخصی قرار داده شوند، بدین منظور از عدد ابتدایی بازه برچسب ها که مقدار آن ۸۶۱- است تا بزرگترین مقدار برچسب که ۲۰۵۵ است، به صورت ۱۵ تایی در پوشه هایی با نام اول بازه برچسب های هر دسته، تقسیم می کنیم. به همین روش حدوداً ۵۰۰۰۰ Recurrence plot بدست آوردیم؛ که از این ها برای آموزش مدل استفاده خواهیم کرد.

در مرحله بعد، بایستی بر روی هر پنجره یک recurrence plot زده شود، که در بخش ادامه به آن خواهیم پرداخت.

۵-۲ بلوک دیاگرام مربوط به پروژه

اگر بخواهیم روال کلی پروژه را در قالب یک بلوک دیاگرام ارائه دهیم. می توان آن به صورت زیر نمایش داد:



۳-۵ ایجاد recurrence plot برای هر پنجره

همانطور که در بخش قبل بیان شد، تقسیم بندی برای برچسب ها و داده ها، انجام شد، حال در هر دسته باید بنا به مقادیر موجود برای دو ویژگی تعیین شده، یک recurrence plot ایجاد شده و به صورت تصویر با فرمت jpg در پوشه مربوط به برچسب مرتبط با پنجره، ذخیره شود. سپس از این تصاویر به عنوان ورودی در هر روش استفاده خواهد شد.

۴-۵ کد مربوط به تعیین پنجره و ایجاد تصاویر

با توجه به مطالبی که در دو بخش قبل بیان شد، کد زیر به منظور تعیین پنجره ها و همچنین ایجاد تصاویر در نظر گرفته شده است. نکته قابل ذکر این است که داده ها در آرایه های مربوط به هر پنجره، نباید به صورت سطری ذخیره شوند، بلکه برای حفظ ماهیت سری زمانی، باید در هر آرایه مربوط به پنجره، ابتدا مقادیر ۱۰ تایی ستون اول و سپس مقادیر ۱۰ تایی ستون دوم ذخیره شود.

```
import numpy as np
import argparse
import cv2 as cv
import pandas as pd
fileenn="Sp%.csv"

df = pd.read_csv('Sp%.csv', delimiter=',', header=0, usecols=[1,2])
def scaling(series):
    minimum = np.amin(series)
    maximum = np.amax(series)
    new = np.zeros(len(series))
    for i in range(len(series)):
        new[i] = (series[i] - minimum)/(maximum - minimum)
    return new

def Mat2Image(matrix, fileName):
    minimum = np.amin(np.min(matrix))
    maximum = np.amax(np.amax(matrix))
    diff = maximum-minimum
    print("max= %.1f, min= %.1f"%(minimum,maximum))
    for i in range(matrix.shape[0]):
        for j in range(matrix.shape[1]):
            matrix[i,j] = 255*((matrix[i,j]-minimum)/(diff))
    cv.imwrite(fileName, matrix)

# Recurrence (Distance) Plot
def rplot(series, err, bin=0):
    dim = len(series)
    rp = np.zeros((dim,dim))
    for x in range(dim):
        for y in range(dim):
            rp[x,y] = abs(series[x] - series[y])
    return rp

def loadingData_x1(fileName):
    dataframe\ = pd.read_csv('Sp%.csv', delimiter=',', header=0, usecols=[1],
engine='python') #first line is read as header
    return dataframe\
def loadingData_x2(fileName):
```

```

dataframe۲ = pd.read_csv(fileName, usecols=[۲], engine='python') #first line
return dataframe۲

#### Main
if __name__ == "__main__":
    import os
    win_lenth = ۱۰
    half_len_win = int(win_lenth / ۲)
    dis_array = [[], []]
    win = []
    windows = []
    path_array = []
    win_min = ۰
    kj = win_lenth
    print('win_length', win_lenth)
    number = ۰
    for i in range(half_len_win, len(df), half_len_win):
        win = []

        counter = i - half_len_win
        while (counter < kj):
            dataframe\ = loadingData_x\ (filenn)
            win\ = dataframe\.iloc[counter:kj]
            win\ = win\.reset_index().values.ravel()
            win.append(win\ )
            win\ = win\[۱::۲]
            print('=====')
            dataframe۲ = loadingData_x۲ (filenn)
            win۲ = dataframe۲\.iloc[counter:kj]
            win۲ = win۲\.reset_index().values.ravel()
            win.append(win۲)
            win۲ = win۲\[۱::۲]
            windows.extend(win\ )
            windows.extend(win۲)
            counter = counter + ۱
            kj = kj + ۱
            number = number + ۱
            if kj >= ۵۳۶۲۴:
                break;

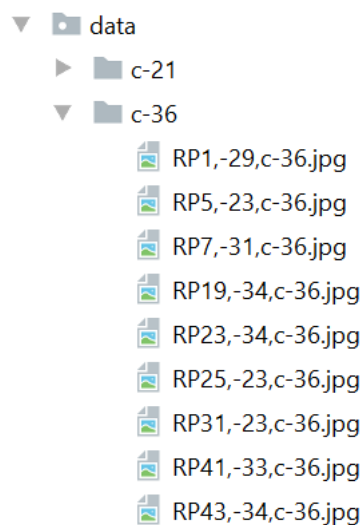
        dataset = windows
        windows = []
        print(dataset)
        y_label = dataframe\.iloc[kj]
        label = y_label.reset_index().values.ravel()
        ylabel = label[۱]

        print("label_y: "+str(ylabel))
        new_dataset = scaling(dataset)
        rp = []
        rp = rplot(new_dataset, err=۰,۰۳, bin=۰)
        first = -۸۶۱
        for index in range(-۸۶۱, ۲۰۵۵, ۱۵):
            # "c" + index = -۸۶۱ + ۱۵
            if index <= ylabel < (index+۱۵):
                yclass = "c"+str(index)
                print("class:" + str(yclass))
                path = "data/"+str(yclass)

```

```
try:
    os.mkdir(path)
except OSError:
    print("Creation of the directory %s failed" % path)
else:
    print("Successfully created the directory %s " % path)
    Mat2Image(rp, "data/"+str(yclass)+"/RP"+str(number)+", "+
str(ylabel)+", "+str(yclass) + ".jpg")
    print('finish win creation')
```

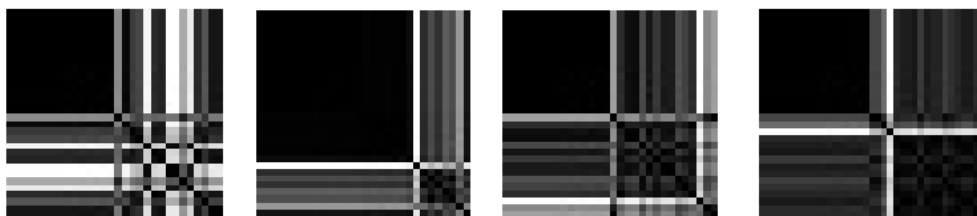
پس از اجرای کد فوق، می توان در پوشه ای با نام data که از قبل ایجاد کرده ایم، پوشه های مربوط به داده ها را به همراه نام پوشه به عنوان برچسب مشاهده کرد. علاوه بر این، در نام گذاری هر فایل، ابتدا شماره هر داده، سپس برچسب خود پنجره و برچسب دسته قابل مشاهده است.



شکل ۸: فایل های ایجاد شده *recurrence plot* برای هر پنجره به تفکیک برچسب

برای نمونه، در شکل فوق، می توان *recurrence plot* های مربوط به چند پنجره، در بازه برچسب ۳۶- تا ۲۱- را مشاهده کرد.

در زیر تصویر چند پنجره که به صورت *recurrence plot* تبدیل و ذخیره شده است، مشاهده می شود:



شکل ۹: نمایش چند پنجره به صورت *recurrence plot*

۶ فصل ششم: روش های استفاده شده در تعیین شباهت

۶-۱ استفاده از شبکه عصبی کانولوشن

در این بخش ، روشی مبتنی بر ماتریس های فاصله، recurrence plot و شبکه عصبی کانولوشنال (CNN) پیشنهاد شده است که نیازی به مهندسی ویژگی ندارد. recurrence plot تجسم حالتهای بازگشت از یک سیستم دینامیکی است. برای پیش بینی برچسب سری زمانی، ابتدا سری زمانی به صورت پنجره های معین به یک تصویر با کمک recurrence plot تبدیل شده و سپس CNN با آن تصاویر آموزش داده می شود. نتایج نشان می دهد که این روش از لحاظ دقت و فراخوانی قادر به دستیابی به نتایج بهتر از یک رویکرد تنها مبتنی بر ویژگی است. . مزیت رویکرد پیشنهادی ما این است که نیازی به استخراج ویژگی های از پیش تعریف شده ندارد و از آنجا که بر اساس ماتریس های فاصله RPs است ، بستگی به انتخاب آستانه ندارد که محاسبه آن دشوار باشد و در معرض خطا باشد.

در سالهای گذشته ، معماری شبکه های عصبی مصنوعی عمیق ، مرزهای عملکرد در کارهای یادگیری ماشینی مانند تشخیص تصویر ، ترجمه زبان و تشخیص صدا را تحت تاثیر قرار داده اند که نام های معدودی دارند. به خصوص ، شبکه های عصبی کانولوشنی (CNNs) نشان داده اند که نتایج برجسته ای را در مشکلات تشخیص تصویر بدست می آورند. CNN ها قادر به استخراج خودکار ویژگی ها در سطوح مختلف انتزاع از داده های شبیه به تصویر هستند و باعث می شود که لزوم تعریف دستی ویژگی ها از بین برود.

پیش از این ایده تبدیل سیگنال های خام به تصاویر برای آموزش مورد استفاده قرار گرفته است. به عنوان مثال ، Bddapati و همکاران سیگنالهای صوتی محیطی را به عنوان سه تصویر (طیف سنج ، ضریب های سفالسی Mel-Frequency و Recurrence plot) برای آموزش CNN ها نشان دادند.

۶-۱-۱ معرفی شبکه های عصبی کانولوشن

یک شبکه عصبی کانولوشن یا پیچشی^۱، نوعی شبکه عصبی مصنوعی است که با موفقیت در کارهای بینایی ماشین استفاده شده است. در اولین اجرای آن توسط LeCun و همکاران، این شبکه، برای تشخیص ارقام دستنویس توسعه داده شد. به طور معمول ، CNN شامل لایه های کانولوشن ، استحکام^۲ و لایه های کاملاً متصل^۳ سنتی است.

لایه کانولوشن یا پیچشی شامل مجموعه ای از فیلترها (یا هسته ها) است که در عرض و ارتفاع تصویر ورودی حلقه می شوند. تفاوت بین یک لایه کاملاً متصل و یک لایه پیچشی در این است که دومی می تواند الگوهای ثابت را از ترجمه یاد بگیرد که یک ویژگی مورد نظر در سیستم های تشخیص تصویر است. همچنین می تواند الگوهای سلسله مراتبی را بیاموزد. یعنی اولین لایه ها، الگوهای محلی کوچک را یاد می گیرند در حالی که لایه های پی در پی الگوهای انتزاعی تر را از لایه های قبلی می آموزند. لایه ادغام تصویر ورودی را به قسمت هایی غیر همپوشانی تقسیم می کند. لایه های pooling به کاهش اندازه ورودی

^۱ Convolutional Neural Network

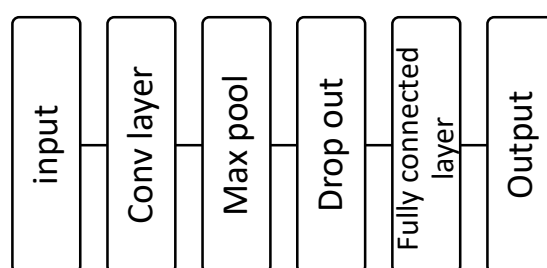
^۲ Pooling

^۳ Fully connected

کمک می کند و بنابراین ، تعداد پارامترها را کاهش می دهد که باعث می شود روند یادگیری کارآمدتر شود. لایه های پیچشی و pooling در کنار هم جمع می شوند و معمولاً آخرین لایه در یک آرایه یک بعدی قرار می گیرد که به عنوان ورودی برای لایه (های) کاملاً متصل نهایی برای بدست آوردن طبقه بندی نهایی استفاده می شود.

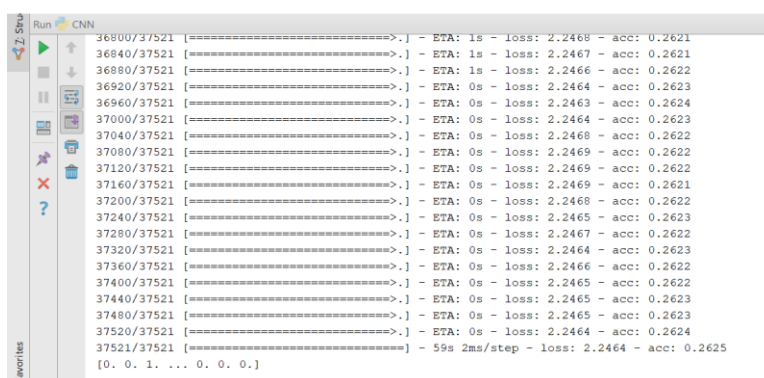
۶-۲ معماری CNN

CNN با ورودی تصاویر با ابعاد $1 \times 30 \times 30$ (عرض ، ارتفاع ، کانال ها) آموزش داده می شود. این در حالتی است که پنجره ها با اندازه ۱۵ در نظر گرفته شوند. همانطور که پیش ازین اعلام شد، ما یک بار از پنجره با اندازه ۱۰ و یک بار با اندازه ۱۵ استفاده کرده ایم. **Error! Reference source not found.** معماری شبکه اولیه را نشان می دهد.



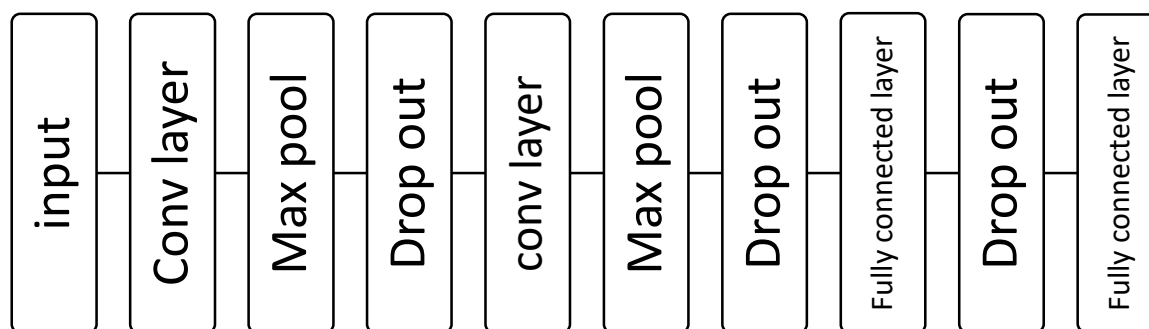
شکل ۱۰: معماری CNN اولیه به کاربرده شده

نتایج بدست آمده به روش فوق، در شکل زیر نمایش داده شده است. همانطور که در تصویر مشاهده می شود، این روش دقت ۲۶ درصد را می دهد. ضمناً برای آزمایشات خود ، از اعتبار سنجی متقابل ۳۰ استفاده کرده ایم. به این صورت که ۳۰ درصد داده ها در هر بار اجرا، مربوط به داده تست و بقیه به عنوان داده آموزشی استفاده خواهد شد. تعداد گام های اجرای مدل نیز ۲۰ گام در نظر گرفته شده است.



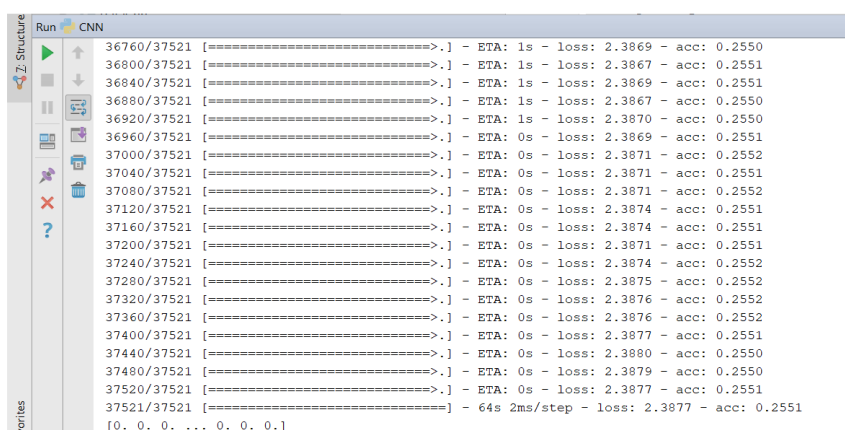
شکل ۱۱: نمایش خروجی شبکه اولیه

پیش بینی می شود با توجه به اینکه با مدل ساده به نتایج خیلی خوبی دست پیدا نکرده ایم، پیچیده تر کردن شبکه نیز به بهبود دقت بدست آمده کمک نکند. اما برای بررسی بیشتر، از یک مدل پیچیده تر نیز استفاده شد، که در شکل زیر معماری شبکه کانولوشنی دوم نشان داده شده است.



شکل ۱۲: استفاده از یک شبکه پیشرفته برای بررسی امکان بهبود دقت

همانطور که پیش بینی شده بود، با پیچیده تر کردن مدل، میزان دقت افزایشی نداشت. خروجی مربوط به شبکه پیچیده، در زیر ارائه شده است.



شکل ۱۳: نتایج حاصل از استفاده از شبکه پیچیده تر

همانطور که در شکل فوق مشاهده می شود، دقت بدست آمده ۲۵ درصد است که نه تنها بهبود نیافته، بلکه پایین تر نیز شده است.

نکته قابل توجه این است که نتایج فوق، بر روی تصاویر مربوط به پنجره های ۱۵ تایی بدست آمده است. برای اینکه اندازه مطلوب بدست بیاید، بر روی شبکه اولیه از تصاویر بدست آمده از پنجره های ۱۰ تایی نیز استفاده شده است، که نتیجه در زیر نمایش داده شده است.

```
Run CNN
34720/37527 [=====] - ETA: 0s - loss: 2.3365 - acc: 0.2454
34880/37527 [=====] - ETA: 0s - loss: 2.3365 - acc: 0.2457
35040/37527 [=====] - ETA: 0s - loss: 2.3367 - acc: 0.2457
35200/37527 [=====] - ETA: 0s - loss: 2.3364 - acc: 0.2459
35360/37527 [=====] - ETA: 0s - loss: 2.3372 - acc: 0.2457
35520/37527 [=====] - ETA: 0s - loss: 2.3375 - acc: 0.2461
35680/37527 [=====] - ETA: 0s - loss: 2.3376 - acc: 0.2461
35840/37527 [=====] - ETA: 0s - loss: 2.3372 - acc: 0.2463
36000/37527 [=====] - ETA: 0s - loss: 2.3371 - acc: 0.2464
36160/37527 [=====] - ETA: 0s - loss: 2.3371 - acc: 0.2465
36320/37527 [=====] - ETA: 0s - loss: 2.3370 - acc: 0.2464
36480/37527 [=====] - ETA: 0s - loss: 2.3376 - acc: 0.2465
36640/37527 [=====] - ETA: 0s - loss: 2.3374 - acc: 0.2467
36800/37527 [=====] - ETA: 0s - loss: 2.3376 - acc: 0.2466
36960/37527 [=====] - ETA: 0s - loss: 2.3377 - acc: 0.2466
37120/37527 [=====] - ETA: 0s - loss: 2.3374 - acc: 0.2468
37280/37527 [=====] - ETA: 0s - loss: 2.3376 - acc: 0.2467
37440/37527 [=====] - ETA: 0s - loss: 2.3370 - acc: 0.2470
37527/37527 [=====] - 13s 350us/step - loss: 2.3370 - acc: 0.2470
[0. 1. 0. ... 1. 0. 0.]
```

شکل ۱۴: نتایج بدست آمده مربوط به استفاده از پنجره با اندازه ۱۰

مطابق نتایج فوق، استفاده از پنجره با اندازه ۱۰، نسبت به پنجره با اندازه ۱۵ نتایج بدتری را ارائه خواهد داد. دقت به دست آمده در این حالت ۲۴ درصد است.

۳-۶ استفاده از روش ssim

با توجه به تعیین اندازه مطلوب، با استفاده از مدل شبکه عصبی کانولوشن، برای روش ssim، از تصاویر مربوط به پنجره با اندازه ۱۵، استفاده شده است. به کمک این روش و با تعیین حد آستانه ۸۰ برای تایید شباهت، نتایج حاصل به صورت زیر است:

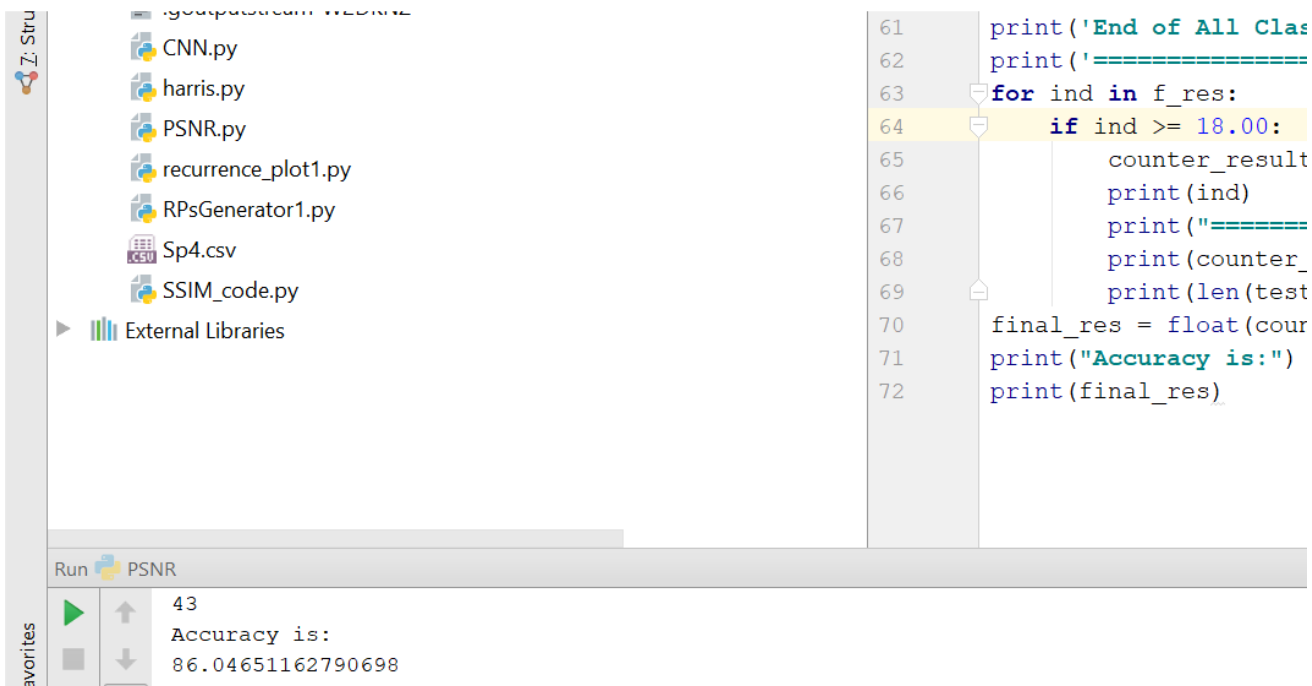
```
Run SSIM_code
0.9184376256883842
=====
39
43
0.8571581198811121
=====
40
43
0.9190066585676099
=====
41
43
Accuracy is:
95.34883720930233
```

شکل ۱۵: میزان دقت بدست آمده در روش ssim

بنا به نتایج بدست آمده، به روش ssim، با حد تایید ۸۰، میزان دقت بدست آمده ۹۵ درصد بوده است، که دقت مطلوبی است.

۴-۶ استفاده از روش psnr

در این بخش به بررسی نتایج حاصل از اعمال روش psnr بر روی داده ها با اندازه پنجره ۱۵ می پردازیم. با اجرای کد مربوط به روش psnr نتایج به صورت زیر بدست می آید:



```

61 print('End of All Clas
62 print('=====
63 for ind in f_res:
64     if ind >= 18.00:
65         counter_result
66         print(ind)
67         print("=====
68         print(counter_
69         print(len(test
70 final_res = float(cour
71 print("Accuracy is:")
72 print(final_res)
    
```

Run PSNR

43
Accuracy is:
86.04651162790698

شکل ۱۶: نمایش نتیجه خروجی به روش *PSNR* با حد تعیین شده ۱۸

همانطور که در نتایج مشاهده می شود، میزان دقت در این حالت با حد آستانه ۱۸، برابر ۸۶،۰۴ درصد بوده است.

۵-۶ ارزیابی نتایج

اگر در حالت کلی بخواهیم روش های فوق را به طور خلاصه تحلیل کنیم، به جدول زیر خواهیم رسید.

جدول ۱: نتایج مقایسه روش های استفاده شده

اندازه پنجره	psnr	ssim	شبکه پیچیده کانولوشن	شبکه ساده کانولوشن	
۱۵	۸۶،۰۴ درصد	۹۵ درصد	۲۵ درصد	۲۶ درصد	میزان دقت
۱۰		-	-	۲۴ درصد	

با استفاده از جدول نتایج، می توان مشاهده کرد که روش *ssim* در مقایسه با سایر روش ها می تواند با اختلاف بالایی، کارآمد باشد. همچنین اندازه پنجره ۱۵ می تواند دقت بهتری نسبت به اندازه پنجره ۱۰ بدست آورد. عدم عملکرد صحیح شبکه در این پروژه را می توان کمبود نمونه های موجود برای هر دسته برچسب در داده های ورودی دانست. بنابراین پیش بینی می شود با استفاده از نمونه های بیشتر برای هر دسته، بتوان با کمک شبکه کانولوشن نیز به نتایج مطلوب دست یافت.

۷ فصل هفتم: کارهای جانبی

۷-۱ استفاده از یک شبکه با استفاده از tensorflow

این شبکه را google در سال ۲۰۱۸ منتشر کرد. برای استفاده از این شبکه فقط کافی است تصاویر train همراه با کلاس هایشان داخل پوشه های متفاوت، در پوشه trainingImages و فایل های تست بدون کلاس داخل پوشه testImages قرار بدهید. سپس کد را اجرا می کنید تا شبکه آموزش ببیند. (myRetrain۲.py)

پس از آموزش شبکه مدل ذخیره خواهد شد. در گام بعدی، کد مربوط به قسمت test را اجرا کنید. (mytest)

هر نمونه در فایل تست با توجه به حداکثر میزان شباهت به یک کلاس نسبت داده می شود؛ سپس سایر کلاس ها با توجه به میزان شباهت مرتب شده و در خروجی نمایش داده می شوند.

تعداد بسیار زیاد داده ها باعث سنگین شدن شبکه شد و عملاً شبکه قبل از ذخیره مدل از بین رفت.

برای حل این مشکل تعداد داده ها را به یک سوم کاهش دادیم.

مدل آموزش داده شده و ذخیره شد؛ اما در زمان تست، دقت بسیار نزدیک به صفر دارد!!!

۸ فصل هفتم: کارهای آینده

یکی از مسائلی که در استفاده از شبکه کانولوشن برای بررسی شباهت مطرح شد، بحث کمبود داده های آموزشی بوده است که در صورت ارتقا دیتاست، می توان امیدوار بود که به نتایج بهتری دست پیدا کنیم.

همچنین با توجه به تنوع زیاد روش های تعیین میزان شباهت، می توان موارد مطرح شده را با سایر روش ها، مقایسه کرده و نتایج آن را مورد تجزیه و تحلیل قرار داد.

علاوه بر این، با توجه به اینکه تعداد نمونه های هر برچسب، با سایر پوشه ها متفاوت است، بایستی به کمک روش هیستوگرام، نسبت به ارزش گذاری داده ها اقدام نمود. به این صورت که پوشه هایی که دارای نمونه های زیادی هستند، در مقایسه با پوشه ای که تنها یک تصویر دارد، از ارزش یکسانی برخوردار نباشند. همین امر می تواند در خروجی به دست آمده موثر واقع شود.

بنابراین پیشنهاد می شود در راستای بهبود روش ها و نتایج، موارد فوق اعمال گردد.

۹ مراجع

- <http://www.mojsozan.com/forum/showthread.php?tid=۲۳۲۸>
- https://en.wikipedia.org/wiki/Structural_similarity
- <https://fa.wikipedia.org/wiki/psnr>
- https://en.wikipedia.org/wiki/Signal-to-noise_ratio
- <https://www.quora.com/What-is-a-signal-to-noise-ratio>
- <http://www.recurrence-plot.tk/>
- <https://github.com/agnaldoae/simpleRP>
- <https://www.kaggle.com/tigurius/recuplots-and-cnns-for-time-series-classification/comments>