تشخیص چهره به همراه خود رمزنگارها و شبکه های عمیق

Face recognition via Deep Stacked Denoising Autoencoders

فرشاد برجعلی زاده، ۶۱۰۳۹۹۰۱۵

پروژه یادگیری ماشین، اسفند ماه ۹۹

۱. چکیده

شاید بتوان گفت در حال حاضر در دنیا یکی از جذاب ترین بخش های پردازش تصویر، کاربرد تشخص چهره (Face recognition) است به دلیل کاربردهای بسیار مهم و جذابی که این سیستم ها دارند از جمله کاربردهای امنیتی و تشخیص حالات چهره، پیاده سازی و بهینه سازی چنین سیستم هایی همواره جزء داغ ترین مباحث پژوهشی و تحقیقاتی در دانشگاه ها و مراکز تحقیقاتی دنیا است دراین پروژه سعی می کنیم که مقدمه ای ازآشنایی با تشخیص چهره و پیاده سازی الگوریتم هایی با استفاده از خود رمزنگارها (Autoencoders) و شبکه های کانولوشنال (Convolutional Network) ارائه دهیم.

۲. مقدمه

تشخیص چهره به مجموعه علوم و تکنولوژی هایی گفته می شود که هدف آن تشخیص هر چه دقیق تر و با سرعت بیشتر چهره انسان است. اگر بخواهیم به صورت علمی تر و یا فنی تر صحبت کنیم، سیستم تشخیص یک کاربرد بیومتریک مبتنی بر هوش مصنوعی است که میتواند به صورت منحصر به فردی افراد را از طریق مقایسه الگوهای مبتنی بر بافت ها و هندسه صورت تشخیص دهد. تکنیک

هـای زیـادی بـرای پیـاده سـازی ایـن تکنولـوژی مـورد اسـتفاده قـرار مـی گیـرد امـا روش کلـی بـدین شـکل اسـت کـه مشخصـه هـای خاصـی از چهـره افـراد بـا یـک دیتـابیس یـا مجموعـه اطلاعـاتی از پـیش ذخیـره شـده (کـه میتوانـد حاصـل نمونـه گیـری از چهـره افـراد باشـد) مقایسـه مـی شـود. تشـخیص چهـره در دهـه هـای اخیـر بـه دلیـل گسـتردگی کـاربرد در زمینـه هـای مختلـف، مـورد توجـه بسـیاری از محققـان قـرار گرفتـه اسـت یکـی از کاربردهـای عامـه و روزمـره ای کـه از آن اسـتفاده مـی شـود قفـل صـفحه گوشـی هـای هوشـمند و ورود و خـروج کارکنـان اداره هـا مـی باشـد. در ایـن پـروژه بـرخلاف مقالـه بـه علـت موجـود نبـود برخـی از دیتاسـت هـای کـار شـده در مقالـه اصـلی مـا بـا اسـتفاده از مجموعـه داده هـای معـروف *ORL Face* کـه دارای ۴۰۰ تصـویر از چهـره ۱۰ نفـر در حـالات مختلـف مـی باشـد بـه انضـمام ۳۰ تصـویر از ۳ فـرد دیگـر کـه بـرای ارزیـابی مـدل اضـافه شـده انـد اسـتفاده مـی کنـیم. بـرای پیـاده سـازی مـدل تشخصـیص چهـره از خـود رمزنگارهـای عمیـق نـویزی (Deep Stacked Denoising Autoencoders) بـه همـراه ۲۸۳۵۶۸ پـارامتر و شـبکه هـای عمیـق کانولوشـنی بـه همـراه ۳۵۹۳۰۰۰ پـارامتر کمـک گرفتـه ایـم و عـلاوه بـر خـود رمزنگارهـای عمیـق نـویزی از خـود رمزنگارهـای عمیـق کانولوشـنی بـه همـراه ۱۵۴۲۸۹ پـارامتر بـه صـورت جداگانـه ای بـرای ارزیـابی و پـیش پـردازش تصـاویر اسـتفاده کـرده ایـم کـه در بخـش هـای ۳.۱، ۳.۲ و ۳.۳ و ۴ به آنها می پردازیم.

۳. مشخصات و پیش پردازش داده ها

۳.۱ مشخصات داده ها

در داده های تصـویری که داریم و از این لینک هم قابل دانلود هسـتند در پوشـه *dataset/orl_faces* ۴۰ پوشـه داریم که هر کدام دارای ۱۰ چهره در حالت های مختلف از هر فرد هسـتند و ۳ پوشـه به نام های *sorin, alex, andreea* را به داده های دیتاست اصلی اضافه کردیم تا به عنوان ارزیابی نهایی مدل

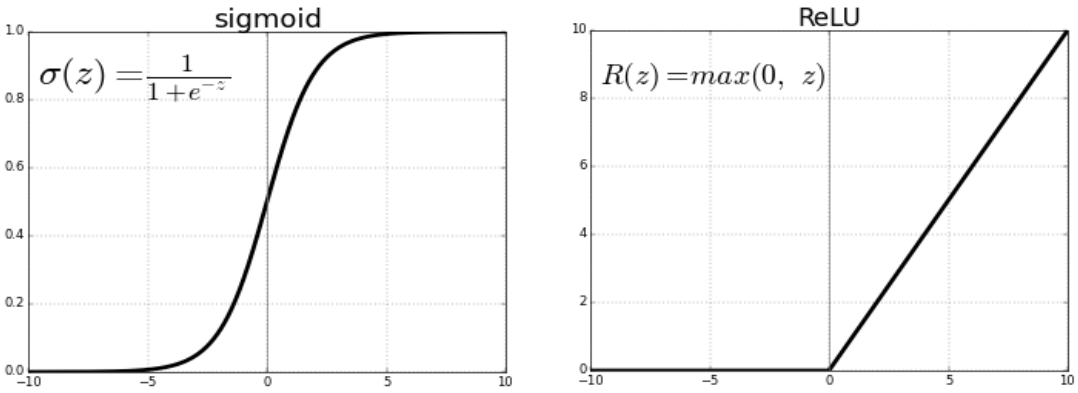
از آنها استفاده کنیم، عکس های هر پوشه به فرمت pmg. **هستند و از ۱ تا ۱۰ شماره گذاری شده اند. سایز تمامی عکس ها به ۳۲*۳۲ پیکسل تغییر داده شده است و داده های** *train* **و** *validataion* **را به ترتیب با تعداد ۳۴۴ و ۸۴ جدا سازی میکنیم، با توجه به مقاله برای یادگیری هرچه بهتر خود رمزنگارها مقداری نویز به داده های وارد میکنم مزیتی که خود رمزنگار با داده های نویزی** (Denoising Autoencoder) **نسبت به خود رمزنگار با داده های غیر نویزی دارد در این است که معمولا خود رمزنگار نویزی تعمیم پذیری** (generalization) **بیشتری نسبت به خود رمزنگار غیر نویزی دارند مقدار نویزی که در داده ها ایجاد میکنیم ۰.۰۱ است، با اعمال نویز داده ها به صورت زیر تبدیل می شوند.**

تصویر ۱. داده ها با اعمال نویز گاووسی ۰.۰۱

برای استفاده از خود رمزنگارها در اینجا ما دو مدل را در نظر گرفته ایم:

۳.۲ خود رمزنگار عمیق (Deep Stacked Denoising Autoencoders)

خود رمزنگارهای عمیق بدین صورت کار می کنند که داده ها را به عنوان ورودی دریافت می کنند و با استفاده از چندین لایه متصل کامل (Fully Connected) **ابتدا داده ها را به فضای کوچک تر برده می شوند** (Encode) **و سپس سعی می کنند که از ابتدا داده ها را بازسازی کنند** (Decode) **در این جا هم دقیقا ما همین کار را انجام داده ایم ابتدا داده ها را با اندازه ۱۰۲۴ گرفته ایم و سپس آنها را به وسلیه خود رمزنگار عمیق به فضایی با اندازه ۱۶ برده ایم و سعی در دوباره ساختن داده ابتدایی از این فضا را داریم. از توابع فعال ساز** (activation function) *relu* **و** *sigmoid* **برای مدل خود رمزنگار عمیق بهره برده ایم.**



تصویر ۲. تابع های فعال ساز relu و sigmoid

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	(None, 1024)	0
dense_31 (Dense)	(None, 128)	131200
dense_32 (Dense)	(None, 64)	8256
dense_33 (Dense)	(None, 16)	1040
dense_34 (Dense)	(None, 32)	544
dense_35 (Dense)	(None, 64)	2112
dense_36 (Dense)	(None, 128)	8320
dense_37 (Dense)	(None, 1024)	132096

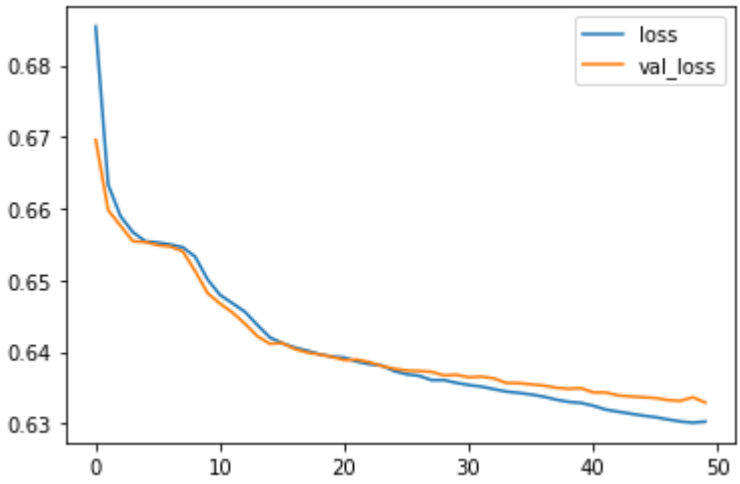
تصویر ۳. نمای کلی از خود رمزنگار عمیق

برای کامپایل کردن خود رمزنگار عمیق هم از بهینه ساز (optimizer) *adam* **و تابعی که به عنوان تابع خطا در نظر گرفته ایم:**

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

تصویر ۴. تابع خطا (loss function)

وزن های شبکه خود رمزنگار عمیق را با ۵۰ بار اجرا روی داده مقدار دهی می کنیم و به مقدار خطای ۰.۶۳ می رسیم، همه این موارد در فایل *face_recognition_deep_autoencoder.ipynb* موجود می باشد.



تصویر۵. نمودار تابع خطا بعد از ۵۰ بار اجرا

۳.۳ خود رمزنگار کانولوشنی (Convolutional autoencoder)

خود رمزنگار های کانولوشنی اساس کار خود را بر شبکه کانولوشنی و پولینگ (Pooling) و لایه های متصل کامل (Fully Conectet) قرار می دهند، اندازه فایل ورودی همان عکس های ۳۲*۳۲ پیکسلی هستند و در ابتدا با استفاده از کانولوشن و پولینگ داده ها را به فضای کوچک تری می بریم (decoe) و سپس عکس این کار را برای دوباره ساختن داده ها انجام می دهیم در این خود رمزنگار علاوه بر کارهایی که در خود رمزنگار عمیق انجام داده ایم تنظیم کننده ای برای لایه های خروجی هم قرار می دهیم (Layer weight regularizers) تا نتیجه آن را در دقت مدل مشاهده کنیم.

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{v_t}} m_t \text{ (outline)}$$
$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\widehat{v_t}+\epsilon}} \widehat{m_t} \text{ (complete)}$$
$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t},\ \widehat{v_t} = \frac{v_t}{1-\beta_2^t}$$

تصویر۶. نحوه محاسبات بهینه ساز adam

با ۳۰ مرتبه اجرای الگوریتم به مقدار ۰.۶۸ خطا می رسیم، تمامی مراحل گفته شده این قسمت در فایل *face_recognition_conve_autoencoder.ipynb* **موجود می باشد.**

۴. مدل تشخیص چهره

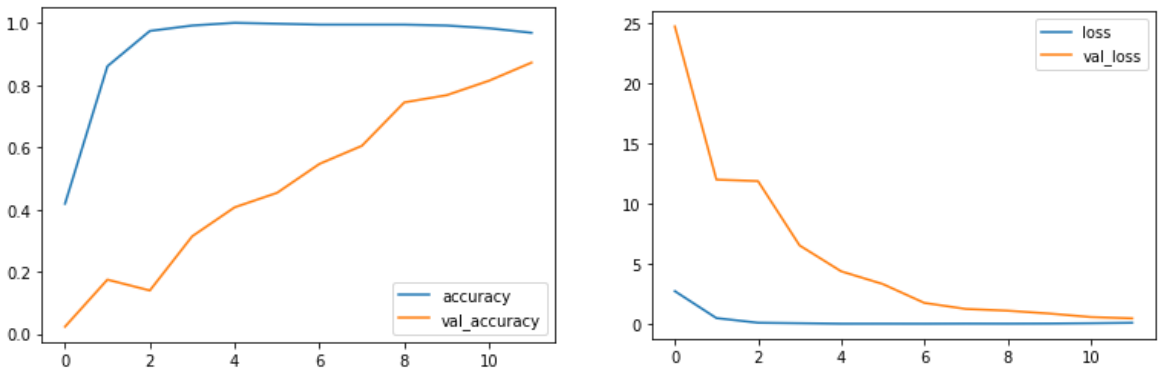
برا ی تشخیص چهره و پیاده سازی آن راه ها و مدل های مختلفی وجود دارد، با توجه به محتوای درس و میزان آشنایی که با شبکه های کانولوشن داشتم مدل را مبتی بر لایه های کانولوشنی می سازم.

این مدل بر پایه دو بلاک اساسی کانولوشنال و لایه های متصل کامل استوار است و در لایه های کانولوشنال از ۱۶ فیلتر با *kernel_size=(3, 3)* **در بلاک اول و از ۳۲ فیلتر با** *kernel_size=(3, 3)* **در بلاک دوم استفاده می کنیم در هر دو بلاک از** *MaxPooling2D* **با سایز (۲*۲) استفاده می کنیم و تابع فعال ساز را هم** *relu* **در نظر گرفته ایم، برای جلوگیری از بیش برازش (***overfiting***) از** *Dropout*(0.25) **بهره می گیریم و خروجی بلاک دوم را به یک بردار تبدیل می کنیم و سپس به لایه های متصل کامل با ۳۰۰۰ و ۴۳ واحد وصل می کنیم، در لایه آخر که در واقع لایه پیش بینی مدل ماست از تابع فعال ساز** *sofmax* **بهره می گیریم.**

با کم و زیاد کردن مقادیر *epochs* **و** *batch_size* **بهترین دقت اجرا را زمانی که** *epochs*=11 **و** *batch_size*=20 **باشد، دریافت می کنیم.**

Layer (type)	Output Shape	Param #
conv2d_29 (Conv2D)	(None, 30, 30, 16)	160
max_pooling2d_15 (MaxPooling	(None, 15, 15, 16)	0
batch_normalization_3 (Batch	(None, 15, 15, 16)	64
dropout_4 (Dropout)	(None, 15, 15, 16)	0
conv2d_30 (Conv2D)	(None, 13, 13, 32)	4640
max_pooling2d_16 (MaxPooling	(None, 6, 6, 32)	0
batch_normalization_4 (Batch	(None, 6, 6, 32)	128
dropout_5 (Dropout)	(None, 6, 6, 32)	0
flatten_2 (Flatten)	(None, 1152)	0
dense_3 (Dense)	(None, 3000)	3459000
dropout_6 (Dropout)	(None, 3000)	0
dense_4 (Dense)	(None, 43)	129043

تصویر ۷. جزئیات مدل *face_recognition_model*



تصویر۸. نمودار های خطا و دقت بعد از ۱۲ بار اجرا

۵. بررسی نهایی (Discussion)

در این پروژه برای استفاده از مدل های پیاده سازی شده و ارزیابی بر روی داده های تست دو مدل را با نام های *deep_model.h5* و *conv_model.h5* ذخیره میکنیم و دقتی که بر روی داده تست فقط با استفاده از دیتاست *Face ORL* با مدل *deep_autoencoder* گرفتیم ٪۹۵ و با مدل *conve_autoencoder* به ٪۸۷ می رسد که با دقت ٪۹۷ مدل فاصله چندانی ندارد و با توجه به این پیاده سازی مقاله را هنوز کسی به انتشار نگذاشته است به نظر قابل قبول می رسد. هدف از این پروژه به چالش کشیدن خود و آشنایی با شاخه ای جذاب از یادگیری ماشین بود، در این پروژه سعی کردم که تمام سعی و تلاشم را انجام دادم که نسبت به مفاهیم سطح بالای مقاله مورد نظر در ابتدا دید پیدا کنم و تا حد خوبی با مسئله پردازش تصاویر آشنا شوم و سپس بتوانم اولین تجربه خود را در پیاده سازی و یادگیری پردازش تصویر به نمایش قرار دهم، تمامی فایل های کار شده در این پروژه در صفحه گیت هاب موجود می باشد و از آن طریق هم می توان به فایل ها دسترسی داشت.

منابع

1. https://www.sciencedirect.com/science/article/abs/pii/S009630031930181X
2. https://keras.io/api/layers/regularizers/
3. https://blog.keras.io/building-autoencoders-in-keras.html
4. https://github.com/curiousily/Deep-Learning-For-Hackers
5. https://github.com/Alireza-Akhavan/deep-face-recognition
6. https://developpaper.com/adam-optimizer/
7. https://realpython.com/face-recognition-with-python/
8. https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/