دانشگاه بوعلی سینا

باسمه تعالی

تشخیص دست خط با استفاده از شبکه های کانولوشنی

استاد راهنما: دکتر ختن لو

دانشجویان:

امیر فانی ۹۶۱۲۳۵۸۰۳۲

متینه بهرامی شکیب ۹۵۱۲۳۵۸۰۰۶

تابستان ۰۰

امیر فانی و متینه بهرامی شکیب رشته مهندسی کامپیوتر دانشگاه بوعلی سینا از تاریخ ۱۴۰۰/۰۱/۰۱ الی ۱۴۰۰/۰۴/۱۹ روی پروژه با موضوع سیستم تشخیص دست خط زیر نظر استاد راهنما جناب دکتر ختن لو تحقیق و پژوهش نموده و بدین وسیله تعهد می کنیم که مطالب این پایان نامه همگی نتیجه فعالیت و تحقیقات خودمان می باشد.

بدین وسیله گواهی می گردد که آقا/خانم..... دانشجوی رشته..... از تاریخ ..... تا .....زیر نظر اینجانب تحقیق و فعالیـــت نمـــوده و بدین وسیله ایشـــان را واجد شرایـــط برای دریافـــت مدرک کارشـــناسی می دانـــم و با نمره ی ..... کار ایشان را تایید می نمایم.

فهرست

چکیده

امروز با گسترده تر شدن انواع جرایم از جمله جعل دست نوشته ها ایجاد یک سیستم برای تشخیص دست خط افراد امری ضروری به نظر می رسد. هدف از طراحی و پیاده سازی این سیستم این است که سرعت رسیدگی به پرونده های موجود در دادگاه ها را سریع تر کند. و باعث افزایش دقت تشخیص شود. سیستمی که براساس شبکه عصبی کانولوشنی[۱] کار خواهد کرد. معماری استفاده شده از نوع ResNets می باشد، که به علت داشتن مسیر های میانبر و جایگزین از سرعت و دقت بالاتری نسبت به معماری های دیگر برخوردار است. معماری ResNets شامل دو زیر مجموعه است که از هر دوی آن ها استفاده شده است. در این تحقیق از ۶۵۷ نویسنده استفاده شده است. و دقت[۲] تست این سیستم ۸۳/۶۴ درصد و نرخ ضرر[۳] آن ۱ درصد است. در این پروژه برای حصول نتیجه دقیق تر شبکه ۱۰ بار اجرا شده و میانگین آن ارایه شده است.

[۱] CNN

[۲] Accuracy

[۳] Loss

۱ فصل اول

مقدمه

۱/۱ مقدمه

سامانه تشخیص دست خط سیستمی است که با استفاده از یک پایگاه داده جامع نسبت به تشخیص هویت مالک یک دست نوشته اقدام می کند. در حال حاضر کار تشخیص دست خط در دادگاه ها برای اموری مثل وصیت نامه ها، کتب، قولنامه ها توسط کارشناس مربوط انجام می شود. که فرایندی زمان بر است. و از دقت بالایی هم برخوردار نمی باشد. با استفاده از این سیستم می توان علاوه بر دقت، سرعت را هم بهبود بخشید. و فرایند کاری دادگاه ها را تسریع نمود.

تشخیص دست‌خط توانایی رایانه برای دریافت و تشخیص دست نوشته های قابل فهم از منابعی مانند اسناد کاغذی و عکس ها به کمک شبکه های عصبی است. یک سیستم تشخیص دستخط، قالب بندی را انجام و تقسیم بندی صحیح را در کاراکترها انجام می دهد و محتمل ترین کلمات را پیدا می کند. این سیستم قابلیت این را دارد که با تغییر پایگاه داده های خود براساس زبان مورد نظر به کار خود به درستی ادامه دهد.

یکی از موارد قدرتمند تشخیص الگو، تشخیص دستخط می باشد. سیستم های تشخیص دستخط به دو نوع آنلاین و آفلاین تقسیم می شوند. روش های مختلفی برای تشخیص دستخط وجود دارد، شامل روش های آماری از جمله مدل های مارکوف پنهان[۴] و روش های ساختاری مانند؛ شبکه های عصبی. دلایلی برای مورد توجه بودن تشخیص دستخط وجود دارد ازجمله اینکه؛ نوشتن چیزی از تایپ کردن آن ساده تر است، عدم امکان تایپ کردن در برخی از مکانها، و عدم وجود یک صفحه کلید کامل روی برخی از رایانه های کوچک.

روش های آفلاین

تشخیص دست خط آفلاین عبارت است از تبدیل خودکار متن در یک تصویر به کد های حروف، که این کد ها در پردازش متن قابل استفاده است. داده های بدست آمده به این صورت در واقع یک نمایش ایستا از دست خط حساب می شوند. تشخیص دست خط آفلاین یک روش دشوار

[۴] Hidden Markov Model

است چرا که افراد متفاوت سبک های دست خط متفاوتی دارند. امروزه، موتور های OCR بر روی متن های چاپ شده در ماشین و ICR بر روی متن های نوشته شده با دست متمرکز هستند.

در ادامه ویژگی های دو تکنیک سنتی و مدرن برای تشخیص دست خط به شکل آفلاین بررسی می شوند:

- **تکنیک های سنتی**
  - استخراج کاراکتر

تشخیص دست خط به شیوه آفلاین معمولا شامل اسکن کردن فرم ها یا اسناد می باشد. سپس پس از اسکن شدن تصاویر تک تک حروف از هم جدا می شوند. ابزار هایی برای انجام این کار وجود دارند. با این وجود مشکلات متعددی در این راستا وجود دارد که یکی از رایج ترین آن ها این است که؛ رایانه توانایی جدا سازی حروف دستنویسی که به هم چسبیده اند را ندارد و آن دو حرف را به عنوان یک حرف تشخیص می دهد. هرچند الگوریتم هایی وجود دارند که این مشکل را مرتفع می سازند.

  - تشخیص حرف

بعد از این که استخراج حروف رخ داد. از یک موتور تشخیص استفاده می شود تا نظیر آن حروف در رایانه را تعیین کند. امروزه تکنیک های متعددی برای این کار وجود دارد.

  - استخراج ویژگی

استخراج ویژگی مشابه تشخیص ویژگی ها در شبکه عصبی است. با این تفاوت که برنامه نویسان باید به شکل دستی ویژگی هایی که به نظرشان مهم است را باید استخراج کنند. این روش به تشخیص دهندگان کنترل بیش تری می دهد. اما این رویکرد به علت این که ویژگی ها را به شکل خودکار یاد نمی گیرد به زمان بیش تری برای توسعه نیاز دارد.

- **تکنیک های مدرن**

در حالی که در روش های سنتی تمرکز بر جداسازی حروف برای تشخیص است. روش های مدرن تمرکز خود را بر جدا سازی خطوط و تشخیص همه حروف آن خط جدا شده گذاشته است. و به شکل ویژه تمرکز آن بر یادگیری ماشین است تکنیک هایی که توانایی استخراج ویژگی ها را دارند. یکی از مدرن ترین تکنیک ها در این روش استفاده از شبکه های کانولوشنی است.

تشخیص آنلاین

تشخیص دست خط به روش آنلاین شامل تبدیل اتوماتیک متن های نوشته شده به کمک یک Digitizer یا PDA است. این نوع داده ها به عنوان جوهر دیجیتال شناخته می شوند و می توان آن را به عنوان یک نمایش دیجیتالی از دست نویس ها در نظر گرفت. سیگنال های به دست آمده به کد های حروف تبدیل می شوند که در پردازش متن قابل استفاده هستند.

عناصر تشخیص آنلاین دست خط شامل:

- یک قلم یا قلم برای کاربر که میتواند با آن بنویسد.
- یک سطح حساس به لمس، که ممکن است با یک صفحه نمایش خروجی با آن یا در مجاورت آن باشد.
- یک نرم افزار که حرکات قلم را در سطح نوشتار تفسیر می کند و سیگنال های حاصل را به متن دیجیتالی ترجمه می کند.

فرایند تشخیص دست خط آنلاین را می توان به چند مرحله کلی تقسیم کرد:

- پیش پردازش
- استخراج ویژگی
- طبقه بندی

کاربرد های سیستم طراحی شده عبارتند از:

۱. جلوگیری از سو استفاده های مالی و هویتی افراد

۲. جلوگیری از جعل اسناد
۳. افزایش سرعت خدمات رسانی در دادگاه ها با حذف نیروی انسانی
۴. کاهش خطای نیروی انسانی در تشخیص؛ به علت حذف شدن آن
۵. حذف روش های سنتی تشخیص دست خط
۶. راه اندازی سیستم های تشخیص تقلب در مدارس
۷. اعتبار سنجی سریع تر دست نوشته های قدیمی و کتب قیمتی

۱/۲ اهداف و فرضیات

هدف از پیاده سازی این سیستم تشخیص دست خط نویسندگانی است که سیستم پیش تر آن ها را آموزش دیده، این کار با بهره گیری از یادگیری عمیق انجام شده است.

هدف استفاده از این سیستم بالا بردن سرعت ، دقت و همچنین پیشگیری از مخاطرات ناشی از خطای انسانی است.

از دیگر اهداف این پژوهش می توان به بررسی میزان ارزش معماری ResNet در سیستم تشخیص دست خط و به اشتراک گذاری نتایج این تحقیقات به منظور بهبود این روش توسط دیگر پژوهشگران اشاره نمود.

در طراحی سیستم فوق فرض بر این است که با به کارگیری معماری قدرتمند ResNet و همچنین پیش پردازش هایی که بر روی داده ها انجام شده، اهدافی که برای سیستم در نظر گرفته شده است محقق خواهند شد و سیستم نهایی نتایج دقیق تری با سرعت بیشتر و بدون از دست رفتگی داده ها نسبت به روش های دستی ارائه خواهد داد. همچنین فرض شده است که معماری ResNet به علت داشتن میانبر هایی از سرعت بیش تری نسبت به سایر معماری ها برخوردار باشد.

۱/۳ چالش های مسئله

در حال حاضر سیستم هایی که وجود دارند از دقت و سرعت کمی بر خوردارند. این سیستم ها خطای انسانی زیادی دارند. و محدود به پایگاه داده ای است که به وسیله آن سیستم آموزش دیده است.

به علت عدم مکانیزه بودن فرایند امکان از بین رفتن اطلاعات، جعل نتایج، مخدوش کردن اسناد و ... وجود دارد.

۱/۴ چالش های پیش رو

اولین چالشی که برای ایجاد این سیستم وجود داشت؛ نبود تعداد کافی عکس از هر نویسنده بود. در این زمینه دیتاست کم نقصی وجود نداشت. به همین جهت به کمک روشی مثل Data Augmentation سعی شد تا این مشکل حل شود.

چالش بعدی Over Fit شدن شبکه عصبی بود. که با حرس کردن شبکه و کم کردن تعداد پارامتر هایی که باید تمرین داده شوند. و استفاده از روش هایی مثل Data Augmentation، Regularization، Drop Out سعی شد تا این مشکل تا حد زیادی مرتفع شود.

یکی از چالش های موجود این بود که تصاویر در پوشه های مختلف وجود داشتند. اما به کمک قطعه کدی همه این تصاویر در یک پوشه جمع آوری شدند.

چالش دیگر نبود سیستم سخت افزاری مناسب برای پردازش این حجم از داده بود. حتی سامانه Colab هم توانایی پردازش این پایگاه داده را نداشت.

۱/۵ ساختار پایان نامه

در این پایان نامه ابتدا چکیده ای از روند پروژه عنوان شده، در ادامه در فصل اول تحت عنوان مقدمه به بررسی تفصیلی سیستم فعلی تشخیص دست خط و چالش های آن اشاره شده است و در نهایت به اهداف و فرضیاتی که منجر به ماشینی شدن سیستم فعلی خواهد شد، پرداخته شده است.

در فصل دوم تحت عنوان مبانی پژوهش به مفاهیم و ساختار کلی روش های تشخیص دست خط اشاره شده و شبکه های عصبی عمیق که زیر مجموعه ای از این ساختار کلی هستند به تفصیل مورد بررسی قرار گرفته شده اند.

در فصل سوم تحت عنوان روش های پیشین، به بررسی و مطالعه چکیده ای از پژوهش های دیگر محققان در این زمینه پرداخته شده است.

در فصل چهارم تحت عنوان روش های پیشنهادی، بلوک دیاگرام هایی که به بررسی روند شبکه های عصبی و روش پیشنهادی برای پیاده سازی این سیستم پرداخته اند آورده شده و جزئیات عملکرد هر بخش توضیح داده شده است .

در فصل پنجم تحت عنوان پیاده سازی و نتایج، به معرفی پایگاه داده، زبان استفاده شده و جزئیات پیاده سازی در سیستم اشاره می شود، همچنین نتایج نهایی که خروجی این سیستم است گزارش شده و این نتایج با نتایج دیگر پژوهشگرانی که در این زمینه فعالیت کرده اند مقایسه شده است.

در فصل ششم تحت عنوان نتیجه گیری خلاصه ای از اقداماتی که در این پژوهش صورت گرفته شرح داده شده و نتیجه گیری کلی، در رابطه با بازدهی این سیستم صورت گرفته است. و در آخر هم پیشنهاداتی برای بهبود سیستم ارائه شده است.

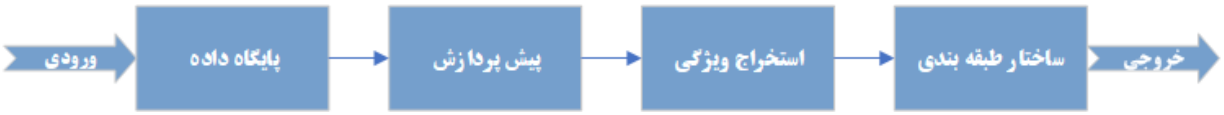
۲ فصل دوم

مبانی پژوهش

۲/۱ ساختار کلی روش های تشخیص دست خط

به منظور پیاده سازی این سیستم ابتدا می بایست پایگاه داده ای متشکل از تعداد زیادی دست خط از نویسنده های مشخص جمع آوری گردد، به طوری که از هر نویسنده تعداد قابل قبولی دست خط موجود باشد. از جمله پایگاه داده های آماده ای که در این زمینه موجود هستند می توان به پایگاه داده های IAM، BFL و CVL اشاره نمود.

در ادامه تصاویر به دسته آمده از این پایگاه داده ها پیش پردازش شده و برای استخراج ویژگی ها، تصاویر به یک مجموعه داده می شوند. این ویژگی های استخراج شده به عنوان ورودی به یک ساختار طبقه بندی وارد شده و در آنجا آموزش داده می شوند در نهایت نتایج به دست آمده تست شده و طبقه بندی می شوند و به عنوان خروجی ارائه داده می شوند.(شکل ۱-۲)



شکل ۱-۲

در گذشته به منظور استخراج ویژگی های تصاویر از روش های دستی استفاده می کردند، اما امروزه با پیشرفت تکنولوژی بدین منظور از شبکه های عصبی کانولوشنی استفاده می شود.

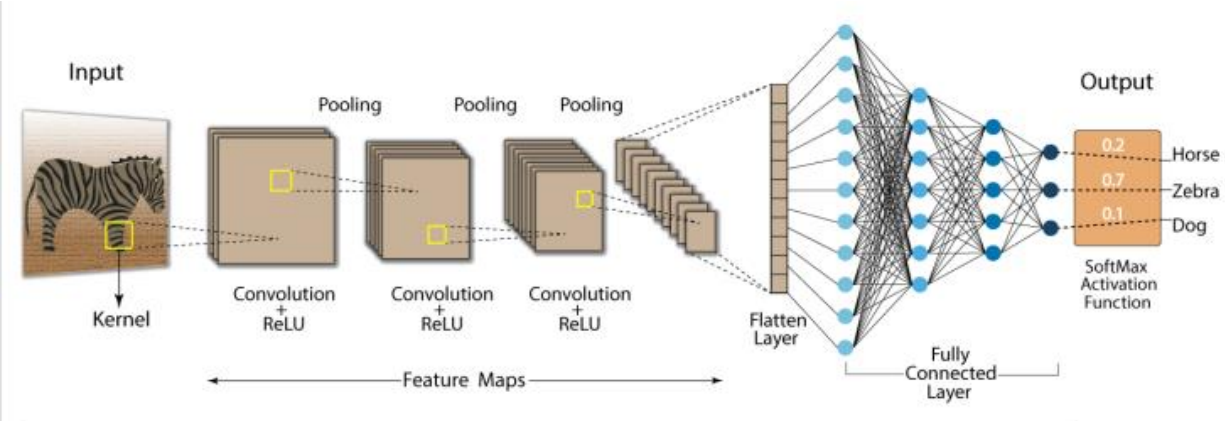
ساختار های متفاوتی در جهت طبقه بندی ویژگی های تصاویر وجود دارند، تعدادی از این ساختار ها عبارتند از[1]:

- Support Vector Machine(SVM)
- Convolutional Neural Network(CNN)
- Decision Tree
- Random Forest

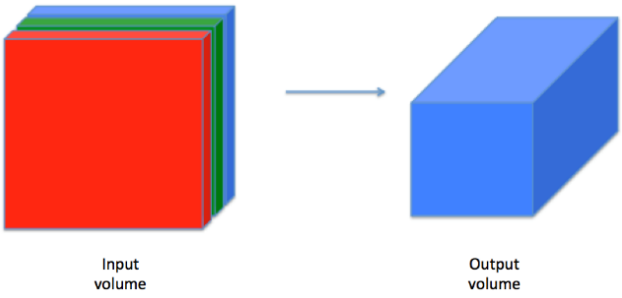
با توجه به اینکه استخراج و طبقه بندی ویژگی ها در این سیستم مبتنی بر شبکه های عصبی عمیق، که شبکه های عصبی کانولوشنی(CNN) زیر مجموعه ای از آن است، می باشد در ادامه به بررسی تفصیلی این دسته می پردازیم.

۲/۲ شبکه های عصبی کانولوشنی

شبکه های عصبی کانولوشنی (Convolutional Neural Networks) یا بطور مخفف CNN به عنوان پایه و اساس سیستم های بینایی ماشین که امروزه مورد استفاده هستند، شناخته می شود (شکل۱-۲). استفاده از یک لایه کانولوشنی به عنوان تبدیل در این نوع شبکه ها باعث می شود که ورودی که یک سایز مشخص دارد به خروجی با سایز متفاوت تبدیل شود. در شکل ۲-۲ این موضوع قابل مشاهده است.



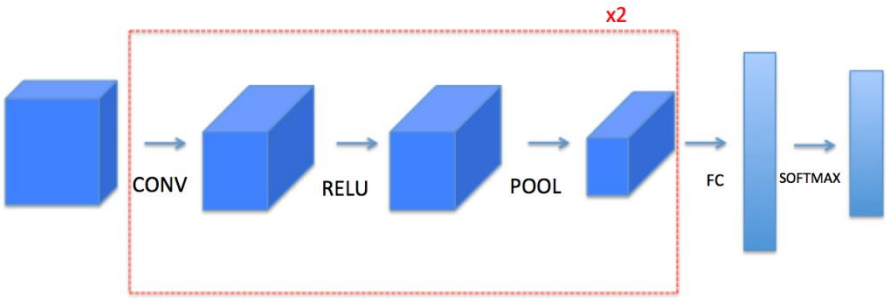
شکل ۲-۲ شبکه های عصبی کانولوشنی



شکل ۳-۲ تغییر ابعاد در تبدیل ورودی به خروجی

یک کامپیوتر برای درک و تشخیص تصویرهای پیچیده مثل تصویر یک سگ، ابتدا ویژگی های ساده تر آن تصویر مانند لبه ها و خم ها را تشخیص می دهد. در یک شبکه عصبی، لایه های متعددی وجود دارند (شکل ۲-۳)؛ در هر یک از این لایه ها، ویژگی های خاصی تشخیص داده می

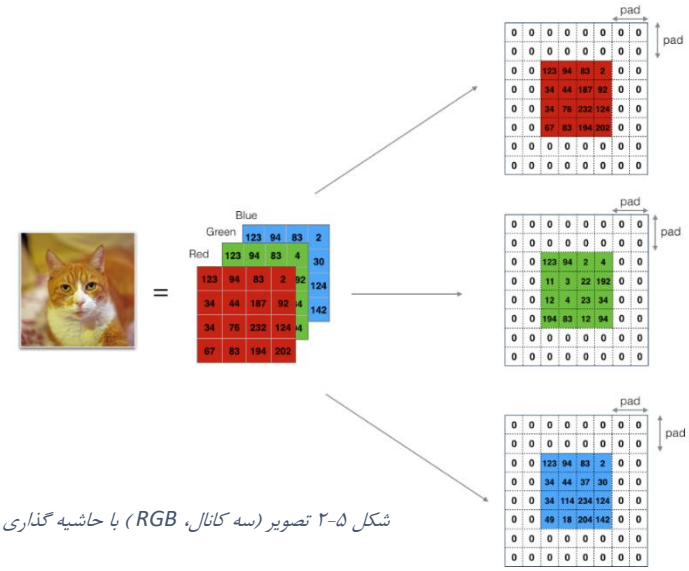
شوند و در نهایت، در لایه آخر، تصویر به طور کامل شناسایی می شود. روندی که توضیح داده شد فرایند کلی نحوه کار یک شبکه عصبی کانولوشن است.



شکل ۴-۲ ترتیب مراحل در شبکه های عصبی کانولوشنی

۲,۲,۱ حاشیه گذاری

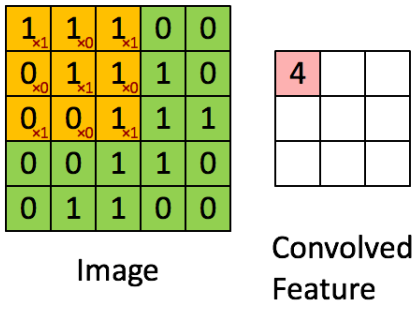
یک از اجزای شبکه های کانولوشنی zero padding می باشد. در این حالت به اطراف آرایه عکس اصلی صفر اضافه می شود تا از دست رفتن اطلاعات در حاشیه ها به حداقل برسد (تصویر۴-۲). در ساخت شبکه های عمیق این موضوع خیلی کاربردی خواهد بود.



شکل ۵-۲ تصویر (سه کانال، RGB) با حاشیه گذاری دو

۲,۲,۲ **کانولوشن**

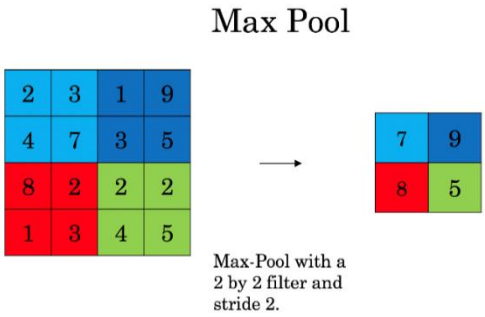
کانولوشن به طور کلی به این شکل عمل می کند؛ که یک ورودی می گیرد. و یک فیلتر در موقعیت های مختلف در آن اعمال می کند(شکل ۵-۲). و یک خروجی با سایز متفاوت می دهد. در شبکه های عصبی بیش از یک فیلتر استفاده می شود. پس در نتیجه خروجی یک ماتریس چند بعدی خواهد بود. که تعداد ابعاد آن همان تعداد فیلتر ها می باشد. در واقع کانولوشن ویژگی هایی را از تصویر به کمک ضرب داخلی خارج می کند.



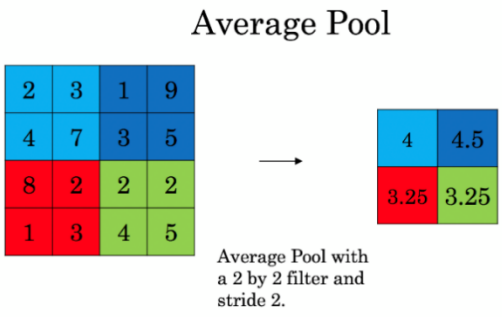
شکل ۶-۲ عملیات کانولوشن

۲,۲,۳ **لایه** pooling

از این لایه برای کاهش ارتفاع و عرض(سایز) تصویر ورودی استفاده می شود. کمک می کند تا هزینه محاسبات کم تر شود. این ابزار به دو شکل بیشینه یابی(شکل۶-۲) و میانگین گیری(شکل ۷-۲) کار می کند.



شکل ۷-۲ کاهش سایز با روش Max Pool

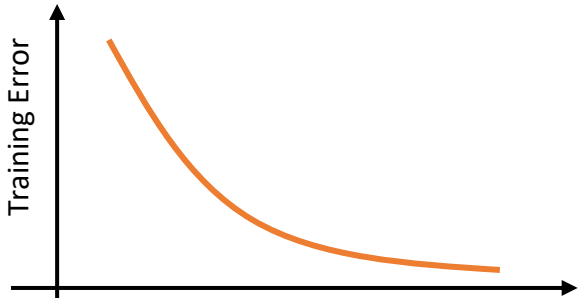


شکل ۸-۲ کاهش سایز با روش Average Pool

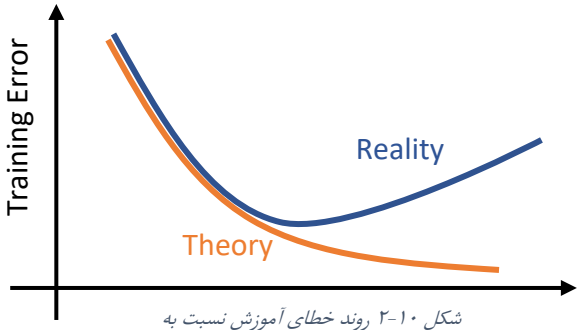
۲/۲/۴ معماری ResNets

شبکه های عصبی متفاوتی وجود دارند. که برای مثال می توان به LeNet-5، AlexNet، VGG، ResNets اشاره کرد. همه این شبکه ها از نوع کانولوشنی بوده و مبنای عملکرد همه آن ها یکسان است. در تئوری هرچه شبکه ای عمیق تر شود عملکرد آن بهتر می شود و قدرت تشخیص و هوشمندی آن بهبود پیدا می کند. اما در عمل خلاف این موضوع رخ می دهد. چرا که تمرین دادن این شبکه و یادگیری پارامتر های آن بسیار دشوار می شود. اما شبکه کانولوشنی ResNet یا همان Residual Networks به علت داشتن مسیر های میانبر این اجازه را می دهد که شبکه های خیلی عمیق را ساخت.

همانطور که ذکر شد عمیق تر شدن یک شبکه باعث افزایش پیچیدگی های آن شبکه می شود. و باعث Over Fit شدن آن شبکه و افزایش Training Error می شود(شکل ۸-۲). اما معماری رزنت به علت داشتن Skip Connection ها از این پیچدگی ها جلو گیری می کند. و خطای آموزش را تا حد زیادی نسبت به معماری های مشابه کاهش می دهد(شکل ۹-۲).



شکل ۹-۲ روند خطای آموزش نسبت به افزایش تعداد لایه ها در معماری ResNet

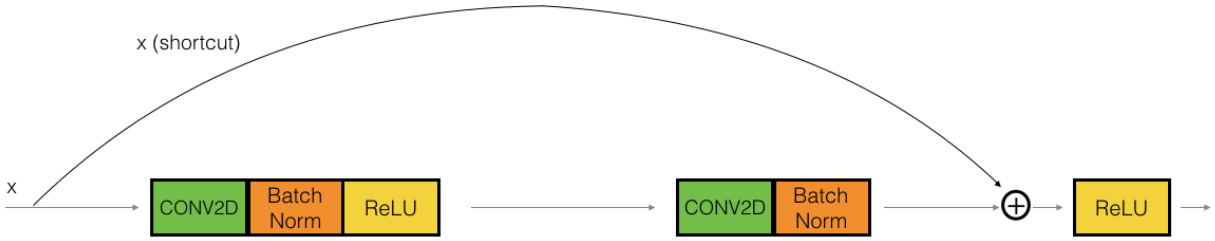


شکل ۱۰-۲ روند خطای آموزش نسبت به افزایش تعداد لایه ها در معماری ساده

در معماری ResNets از دو نوع بلاک متفاوت که به یکسان یا متفاوت بودن ابعاد ورودی و خروجی بستگی دارد استفاده می شود.

۲/۲/۵ بلاک Identity

یک بلاک استاندارد است. و زمانی که ابعاد ورودی و خروجی تابع activation یکسان است استفاده می شود. شکل ۲-۱۰ به خوبی مراحل کار را در این بلاک نمایش می دهد.

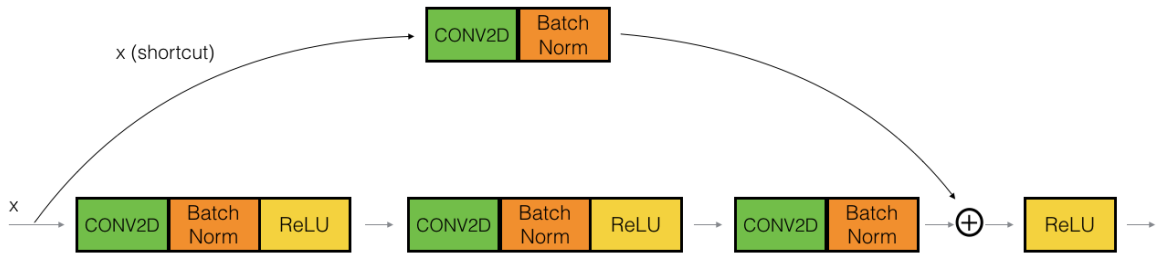


شکل ۲-۱۱ بلاک Identity مسیر میانبر با گذر از دو لایه

مسیر بالایی مسیر میانبر و مسیر پایینی مسیر اصلی می باشد.

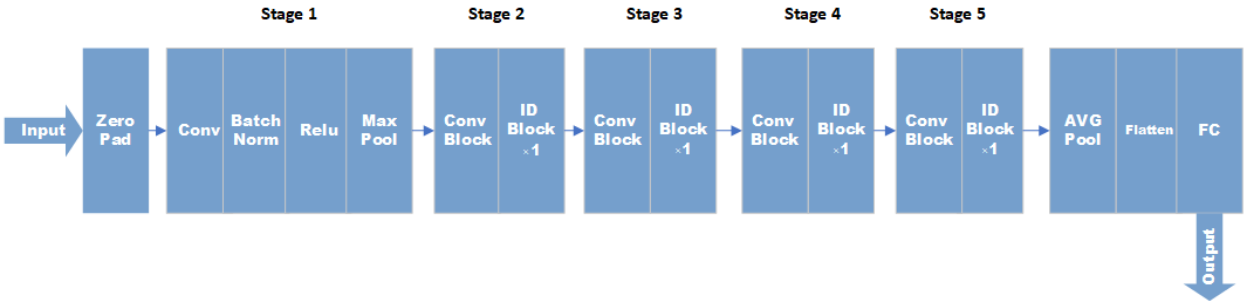
۲/۲/۶ بلاک Convolutional

نوع دوم بلاک است. و زمانی از آن استفاده می شود که سایز ورودی و خروجی یکسان نیست. تفاوت آن با بلاک identity این است که در مسیر میانبر یک کانولوشن وجود دارد. علت وجود این کانولوشن این است که سایز ورودی را با خروجی یکی کند تا بتواند عمل جمع را انجام دهد. شکل ۲-۱۱ نحوه کار این بلاک و مسیر میانبر را نمایش می دهد.



شکل ۲-۱۲ بلاک Convolutional

در آخر برای ساخت یک مدل رزنت می توان از دو بلاک بالا استفاده کرد. و آن ها را باید یک دیگر ترکیب کرد. تا مدلی شبیه شکل ۱۲-۲ ساخته شود.



شکل ۱۳-۲

۲٫۲٫۷ لایه FC

همانطوری که از نام این لایه پیداست تمامی عصب های این لایه به عصب های لایه قبل متصل هستند. وظیفه اصلی لایه fc ترکیب ویژگی محلی در لایه پایین به ویژگی محلی در لایه ها بالاست. در طبقه بندی آخرین fc در شبکه، کل ویژگی ها را ترکیب می کند تا تصویر را دسته بندی کند. به همین خاطر اندازه خروجی این لایه برابر با تعداد کلاس های قابل شناسایی توسط شبکه است.

۲٫۳ کتابخانه Keras

کتابخانه Keras یک کتابخانه open-source شبکه عصبی است که در پایتون نوشته شده است. این کتابخانه روی کتابخانه های TensorFlow, Microsoft Cognitive Toolkit, R Theano قابل اجرا است و روی توسعه ی ماژولار و کاربر پسند تمرکز دارد.

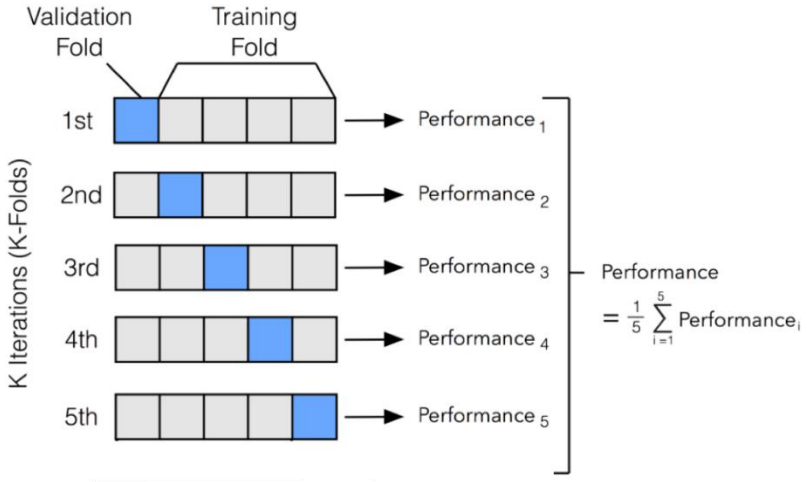
کتابخانه توسط یکی از محققین هوش مصنوعی Google ساخته شد. به صورت پیش فرض برمبنای TensorFlow است. کتابخانه کراس توسط CERN, NASA, NIH و بسیاری از سازمان های علمی دنیا مورد استفاده قرار می گیرد. کراس دارای انعطاف پذیری سطح پایین برای اجرا و ویژگی سادگی سطح بالا برای سرعت بخشیدن به اجرای کد است.

مزایای کتابخانه keras در پایتون:

- مدل های آماده
- پشتیبانی توسط شرکت های بزرگ نظیر؛ Google, Microsoft, Amazon, Apple Nvidia, Uber
- منعطف و قابل تغییر
- قابلیت اجرا در پلتفرم های iOS, Android, web API
- سریع بودن
- به وضوح بیان کردن خطاها

۲٫۴ Cross Validation[2]

یک روش ارزیابی مدل است که تعیین می‌نماید نتایج یک تحلیل آماری بر روی یک مجموعه داده تا چه اندازه قابل تعمیم و مستقل از داده‌های آموزشی است. این روش به‌طور ویژه در کاربردهای پیش‌بینی مورد استفاده قرار می‌گیرد تا مشخص شود مدل مورد نظر تا چه اندازه در عمل مفید خواهد بود. به‌طور کلی یک دور از Cross Validation شامل افراز کردن داده‌ها به دو زیر مجموعه مکمل و سپس انجام تحلیل بر روی یکی از آن زیر مجموعه‌ها (داده های آموزشی) و اعتبار سنجی تحلیل با استفاده از داده‌های مجموعه دیگر است (داده‌های اعتبارسنجی یا آزمایش). برای کاهش پراکندگی، عمل اعتبارسنجی چندین بار با افرازهای مختلف انجام و از نتایج اعتبارسنجی‌ها میانگین گرفته می‌شود. در Cross Validation تعداد k لایه، داده‌ها به K زیرمجموعه افراز می‌شوند. از این K زیرمجموعه، هر بار یکی برای اعتبارسنجی و K-1 تای دیگر برای آموزش بکار می‌روند. این روال K بار تکرار می‌شود و همه داده‌ها دقیقاً یک بار برای آموزش و یک بار برای اعتبارسنجی بکار می‌روند. در نهایت میانگین نتیجه این K بار اعتبارسنجی به‌عنوان یک تخمین نهایی برگزیده می‌شود. عملکرد کلی آن در شکل ۱۳-۲ نشان داده شده است.



شکل ۱۴-۲ نحوه محاسبه Cross Validation

می بایست از Cross Validation[3] به منظور دستیابی به مقاصد زیر استفاده نمود:

۱. استفاده از همه ی داده های دیتاست؛

زمانی که تعداد داده ها کم هستند و آنها به دو بخش آموزش و تست تبدیل می شود، مجموعه داده های تست کم خواهد شد، به عبارت دیگر میزان بازدهی در این مجموعه وابسته به شانس خواهد بود! با استفاده از Cross Validation در واقع K تا مدل متفاوت وجود خواهد داشت در نتیجه می توان پیش بینی را بر روی کل داده ها انجام داد.

۲. داشتن معیار های بیشتر؛

زمانی که الگوریتم یادگیری بر روی چندین مجموعه متفاوت تست می شود نتیجه بازدهی الگوریتم قابل اعتماد تر خواهد بود. به عبارت دیگر با یکبار ارزیابی مجموعه تست یک نتیجه دریافت خواهد شد که ممکن است این نتیجه بر مبنای شانس باشد و به اندازه کافی قابل اعتماد نباشد. پس با معیار های بیشتر می توان نتیجه گیری های مطمئن تری در رابطه با الگوریتم و داده ها انجام داد.

۳. استفاده از مدل پشته سازی؛

در شبکه های عصبی هر لایه خروجی لایه قبل را به عنوان ورودی دریافت می کند. سپس با استفاده از Back Propagation هر لایه خطای خود را محاسبه کرده و به لایه قبلی خود منتقل می کند. چنانچه خواسته شود از Resnet و یا مدل های دیگر استفاده شود خطای مشخص و واضحی وجود نخواهد داشت که به عقب منتقل شود. برای مثال زمانی که از دو مدل استفاده می شود، مدل دوم باید بر روی پیش بینی های مدل اول یادگیری خود را انجام دهد. در اینجا بهترین راه حل استفاده از دو مجموعه داده مختلف برای هر مدل است (یک مجموعه برای آموزش و دیگری برای تست). در واقع با استفاده از مدل اول پیش بینی می شود و آن ها به مدل دوم منتقل خواهد شد. در نهایت آن با Ground Truth[4] مقایسه می شود. در واقع زمانی که داده ها محدود هستند نمی توان هر دو مدل را بر روی یک مجموعه داده آموزش داد. زیرا مدل دوم یادگیری را بر روی داده هایی انجام می دهد که مدل اول قبلا آنها را دیده و پیش بینی کرده است. این بدان معنی است که برای هر مدل یادگیری و تست بر روی یک مجموعه داده انجام نمی شود.

۴. تنظیم دقیق پارامتر ها؛

الگوریتم های یادگیری به تنظیم برخی از پارامترها نیاز دارند. برای یافتن بهترین پارامتر ها می بایست مقادیر مختلف را آزمایش کرد تا به بهترین پارامترها دست یافت. برای این منظور از یک مجموعه سوم به نام مجموعه اعتبار سنجی استفاده می شود. با استفاده از Cross Validation می توان همه مراحل تقسیم داده را با استفاده از یک مجموعه واحد انجام داد.

۳ فصل سوم

روش های پیشین

روش های متفاوتی در راستای پیاده سازی سیستم تشخیص دست خط وجود دارد از جمله روش های کلاسیک و یا روش های مبتنی بر یادگیری عمیق با توجه به اینکه در پیاده سازی این سیستم از یادگیری عمیق بهره برده شده، فلذا در ادامه به اختصار به پژوهش های پیشین در این زمینه که توسط دیگر محققان صورت گرفته است پرداخته می شود.

پژوهشگری به نام Hafemann[5] با استفاده از دو پایگاه داده متفاوت BFL با ۳۱۵ نویسنده و IAM[6] با ۶۵۰ نویسنده سیستم تشخیص دست خط را پیاده سازی کرده است. در این پیاده سازی از روشی که Bertolini[7] در مقاله خود آورده استفاده شده است. برای انجام این آزمایش در پایگاه داده BFL ۱۱۵ نویسنده به صورت تصادفی از بین ۳۱۵ نویسنده اصلی که در پایگاه داده موجود هستند انتخاب شده است، سپس تصاویر به نواحی ۲۵۶*۲۵۶ پیکسل تقسیم بندی شده اند، در ادامه از پایگاه داده IAM ۲۴۰ نویسنده به صورت تصادفی از بین ۶۵۰ نویسنده اصلی که در پایگاه داده موجود هستند، انتخاب شده و سپس تصاویر به نواحی ۲۵۶*۱۲۸ پیکسل تقسیم بندی شده است. در ادامه داده ها برای هر دو پایگاه داده به سه بخش آموزش، تست و ارزیابی تقسیم و از فیلتری با ابعاد ۷*۷ استفاده شده و در نهایت نرخ شناسایی برای پایگاه داده IAM ۹۱.۶۷ درصد و برای پایگاه داده BFL ۹۵.۶۵ درصد گزارش شده است.

پژوهشگر دیگری به نام Tang[8] با روش آفلاین و مستقل از متن این پژوهش را انجام داده است. در این پژوهش ویژگی های تعداد زیادی تصویر بر مبنای CNN استخراج شده، سپس از یک طبقه بند مبتنی بر روش Bayesian استفاده شده است. در این روش از دو پایگاه داده ICDAR2013 با ۳۵۰ نویسنده و CVL با ۳۱۰ نویسنده استفاده شده است.

از پایگاه داده ICDAR2013 چهار نمونه دست خط از هر نویسنده آورده شده است، ۱۰۰ نویسنده به عنوان داده آموزش و ۲۵۰ نویسنده به عنوان داده تست و از پایگاه داده CVL، ۲۷ نویسنده با تعداد ۷ دست خط از هر نفر برای داده های آموزش و ۲۸۳ نویسنده با تعداد ۵ دست خط از هر نفر برای داده های تست استفاده شده است. نرخ شناسایی این روش برای پایگاه داده ICDAR2013 ، ۹۹ درصد و برای CVL ۹۹.۷ درصد می باشد.

Xing[9] پژوهشگر دیگری است که مبتنی بر روش مستقل از متن این آزمایش را پیاده سازی کرده است. این نویسنده روش DeepWriter را بر مبنای Multi-Stream شبکه های CNN پیشنهاد داده است. به این صورت که CNN دو تصویر را به عنوان ورودی دریافت می کند، که هر کدام در یک شبکه مجزا به صورت موازی با یک دیگر آموزش داده می شوند. پس از پایان آموزش هر دو نتیجه با یکدیگر ادغام شده و منجر به یک تصمیم خواهد شد. در آزمایشی که هر مجموعه آموزش شامل ۳۰۱ و ۶۵۷ نویسنده از پایگاه داده IAM بوده است؛ نرخ شناسایی به ترتیب برابر با ۹۹.۱ و ۹۷.۳ درصد شده است. در یک آزمایش دیگر با تعداد ۳۰۰ نویسنده موجود در پایگاه داده HWDB1 نرخ شناسایی ۹۳.۸۵ درصد گزارش شده است.

پژوهشگر بعدی Chen[10] از روش یادگیری ویژگی Semi-Supervised برای شناسایی آفلاین استفاده کرده و با بهره گیری از تکنیک Weighted Label Smoothing Regularization (WLSR) به داده ها برچسب زده شده است، علاوه بر این تکنیک، با استفاده از CNN ویژگی های تصاویر استخراج شده است. در این آزمایش از پایگاه داده های ICDAR2013 با مجموع ۳۵۰ نویسنده که از این تعداد ۱۰۰ نویسنده برای آموزش و ۲۵۰ نویسنده برای تست، CVL با مجموع ۳۱۰ نویسنده که از این تعداد ۲۷ نمونه برای آموزش و ۲۸۳ نمونه برای تست و IAM با حدود ۴۰۰ نویسنده استفاده شده است. دست خط های موجود در این پایگاه داده ها به دو زبان انگلیسی و یونانی می باشد. در این آزمایش پایگاه داده CVL به منظور افزایش تعداد نمونه ها استفاده شده است. نتایج نشان داده که روش یادگیری Semi-Supervised در پایگاه داده ICDAR2013 ۹۶.۶ درصد نرخ شناسایی داشته است. و در پایگاه داده CVL نرخ شناسایی ۹۹.۲ درصد داشته است.

پژوهشگر دیگری به نام Nguyen[11] با استفاده از رویکرد مستقل از متن این آزمایش را انجام داده است. برای طبقه بندی و آموزش داده های پایگاه داده JEITA-HP که شامل ۱۰۰ دست نوشته به زبان ژاپنی است و پایگاه داده های IAM و Firemaker که با یکدیگر ادغام شده و مجموعه واحدی شامل ۹۰۰ دست نوشته به زبان انگلیسی را تشکیل داده اند از روش CNN استفاده شده است. نرخ شناسایی در پایگاه داده JEITA-HP ۹۹.۹۷ درصد، Firemaker با تعداد ۲۵۰ نویسنده

۹۲.۳۸ درصد، IAM با تعداد ۶۵۰ نویسنده ۹۰.۱۲ درصد و در نهایت نیز با ادغام دو پایگاه داده مذکور نرخ شناسایی ۹۱.۸۱ درصد گزارش شده است.

۴ فصل چهارم

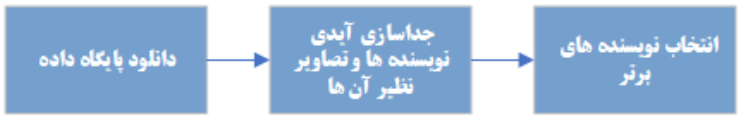
روش پیشنهادی

روش پیشنهادی به این صورت است که در ابتدا تصاویر پایگاه داده بارگذاری می شوند. سپس اقداماتی به عنوان پیش پردازش روی آن ها اعمال می شود تا تصاویر آماده ورود به شبکه شوند. پس از ورود به شبکه پارامتر های شبکه آموزش داده می شوند. در آخر هم مدل ساخته شده ارزیابی می شود و نتایج آن به نمایش در می آید. نمودار کلی روش پیشنهادی در شکل ۱-۴ آمده است.

شکل ۱-۴

۴٫۱ بارگذاری

پایگاه داده مورد نظر دانلود می شود. سپس براساس فایل راهنمای اسامی پایگاه داده، آیدی نویسنده ها و تصاویر نظیر آن ها جدا می شود. سپس برای افزایش سرعت تعدادی از نویسندگانی که دست نوشته بیش تری دارند انتخاب می شوند (نویسنده های برتر). برای گزارش نتایج کلی در آخر از تمامی نویسنده ها استفاده شد. شکل ۲-۴ نمودار آن نشان داده شده است.



شکل ۲-۴

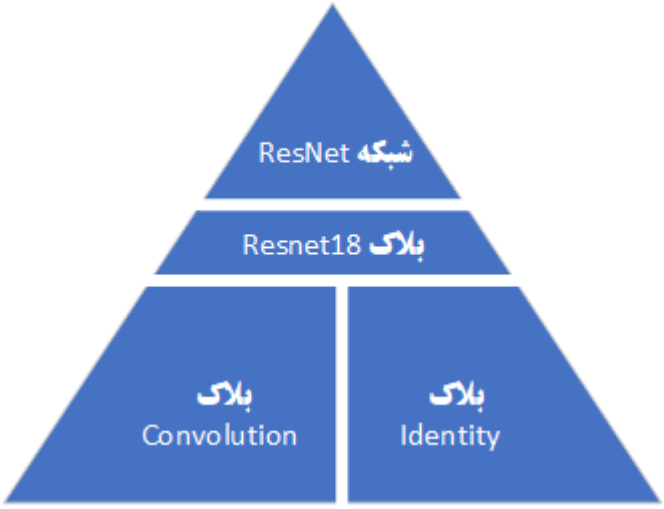
۴٫۲ پیش پردازش

در این مرحله تصاویر در اندازه ۶۴*۶۴ در قالب Gray Scale باز می شوند. سپس مقادیر آرایه های تصاویر نرمال سازی شده و در بازه ۱-۰ قرار می گیرند. سپس آیدی نویسنده ها کد گذاری می شوند(شکل ۳-۴).

شکل ۳-۴

۴/۳ ساخت شبکه

شبکه ساخته شده براساس معماری ResNets می باشد. از سه بلاک به نام های Identity و Convolution و ResNet18 تشکیل شده است. که هر کدام در فصل مبانی پژوهش به تفکیک تشریح شدند. در شکل ۴-۴ نمودار کلی شبکه نشان داده شده است.



شکل ۴-۴

- بلاک Identity

از ۴ گام تشکیل شده است. که در گام اول کانولوشنی بر روی تصاویر ورودی اعمال می شود و سپس نرمال سازی رخ می دهد و تابع Relu به عنوان Activation در آن اعمال می شود. برای کاهش احتمال Overfit شدن از یک Dropout هم استفاده می شود. همین اتفاق در گام های بعدی نیز رخ می دهد. در گام آخر خروجی مسیر میانبر با مسیر اصلی جمع می شود. در شکل ۱۰-۲ می توان نمودار آن را دید.

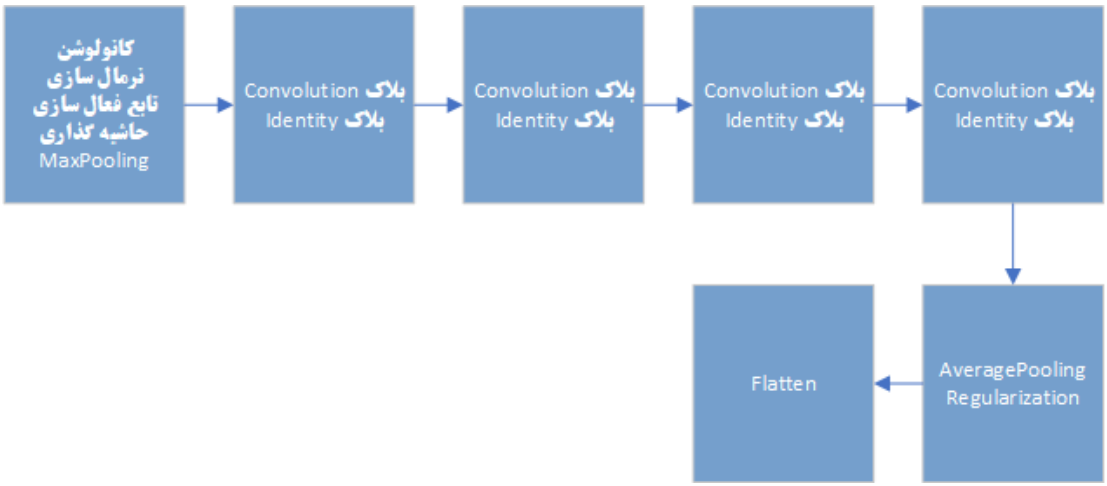
- بلاک Convolution

در این بلاک ابتدا فیلتر ها از ورودی دریافت می شوند. سپس با فیلتر های متفاوت بر روی تصاویر عمل کانولوشن را انجام می دهد. بعد از آن نرمال سازی انجام می شود و تابع فعال سازی Relu در آن اعمال می شود. در ادامه هم برای جلوگیری از Overfit شدن از Dropout استفاده می

شود. همین سلسله فرایند ها در گام های بعدی انجام می شود. و در آخر مسیر اصلی با مسیر میانبر جمع می شود. شکل ۱۱-۲ به خوبی این فرایند ها را نمایش می دهد.

- تابع ResNet18

در این تابع از دو بلاک بالا استفاده می شود تا شبکه کانولوشنی با معماری ResNets ساخته شود. از پنج مرحله تشکیل شده است. در مرحله اول بر روی ورودی حاشیه گذاری شده، عمل کانولوشن رخ می دهد و سپس نرمال سازی می شود. در ادامه تابع Relu استفاده شده و به ترتیب از حاشیه گذاری و MaxPooling استفاده می شود. در گام های دو و سه و چهار و پنج به ترتیب از بلاک Convolution و Identity استفاده می شود. در آخر هم یک AveragePooling استفاده شده و به کمک Regularization سعی می شود تا احتمال Overfit کاهش پیدا کند. لایه Flatten آخرین لایه این مرحله است و مدل پس از آن ساخته می شود. به شکل مختصر نمودار کلی آن در شکل ۵-۴ آورده شده است.



شکل ۵-۴

۵ فصل پنجم

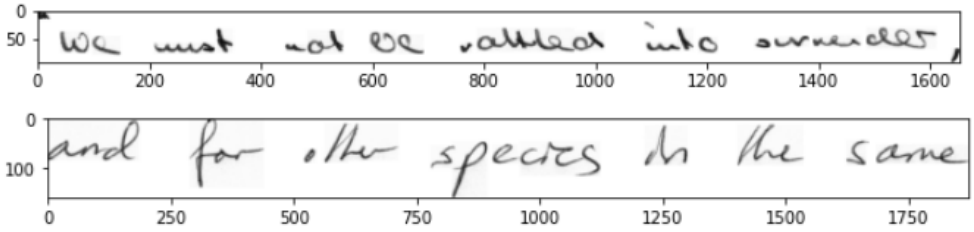
پیاده سازی و نتایج

۵/۱ پایگاه داده

پایگاه داده استفاده شده به نام IAM می باشد و دست خط استفاده شده در این پایگاه داده انگلیسی است که دارای مشخصات زیر است:

- ۶۵۷ نویسنده نمونه هایی از دست خط خود را ارائه دادند.
- ۱۵۳۹ صفحه متن اسکن شده است.
- ۵۶۸۵ جملات جدا شده و برچسب زده شده اند.
- ۱۳۳۵۳ خط متن جدا شده و برچسب زده شده اند.
- ۱۱۵۳۲۰ کلمات جدا شده و برچسب زده شده اند.

نمونه هایی از داده ورودی در شکل ۱-۵ نشان داده شده است:



شکل ۱-۵ نمونه دست خط

پایگاه داده استفاده شده دارای یک فایل راهنما با پسوند txt می باشد. که الگو ذخیره سازی نام عکس ها را مشخص می کند. جدول ۱-۵ خلاصه ای از آن روش را نمایش می دهد.

Form Id	Writer Id
a01-000u	000
a01-000x	001

جدول ۱-۵

سپس با توجه به جدول ۱-۵ عنوان هر نوشته به نویسنده آن نگاشت می شود. شکل ۲-۵ قطعه کد مربوط را نمایش می دهد.

```
form_writer = {}
forms_file_info = "/content/drive/MyDrive/forms_BA.txt"
with open(forms_file_info) as f:
    for line in islice(f, 0, None):
        line_list = line.split(' ')
        form_id = line_list[0]
        writer = line_list[1]
        form_writer[form_id] = writer
```

```
list(form_writer.items())[0:5]
```

```
[('a01-000u', '000'),
 ('a01-000x', '001'),
 ('a01-003', '002'),
 ('a01-003u', '000'),
 ('a01-003x', '003')]
```

شکل ۲-۵خواندن اطلاعات از فایل

تعداد عکس ها برای ۶۵۷ نویسنده ۱۶۷۵۲ می باشد. سپس عکس دست خط ها تبدیل به آرایه هایی در فرمت grayscale خواهند شد. عکس ها به آرایه تبدیل و مقادیر آن ها نرمال سازی می شود. برای یکسان سازی همه عکس ها آن ها در ابعاد ۶۴*۶۴ ذخیره می شوند.(شکل۳-۵)

```
img_files = np.zeros([16752,64,64,1], dtype=np.float64)

img_targets = np.zeros((0), dtype=np.str) #Include the writer_id
img_files_path = np.zeros((0), dtype=np.str)
path_to_files = os.path.join(temp_sentences_path, '*')

for i, file_path in enumerate(glob.glob(path_to_files)):

  #print(len(glob.glob(path_to_files)))

  img = image.load_img(file_path, color_mode="grayscale", target_size=(64, 64))

  #Saving the location of images.
  img_files_path = np.append(img_files_path, file_path)

  #Making the images ready include normalization.
  img_arr = image.img_to_array(img)
  img_arr = np.expand_dims(img_arr, axis=0) # a new dimension added to img_arr
  img_arr = img_arr/255.0

  img_files[i] = img_arr
```

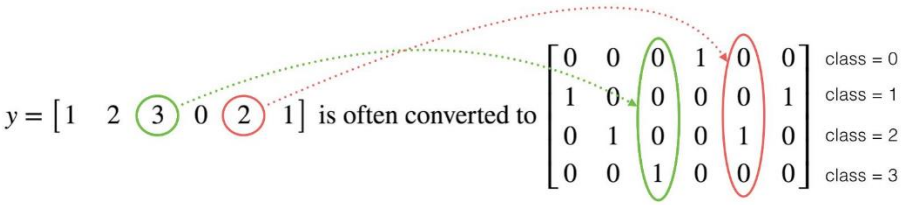
شکل ۳-۵ تبدیل عکس به آرایه

سپس جهت سهولت در امر پردازش لیست img_targets که شامل آیدی نویسنده ها است، توسط کدی که در شکل ۴-۵ آمده Encode و در لیست encoded_img_targets ذخیره می شود. از این لیست برای One Hot Encoding (شکل ۵-۵ و ۶-۵) استفاده خواهد شد.

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
encoder.fit(img_targets)
encoded_img_targets = encoder.transform(img_targets)

print("Writer ID        : ", img_targets[:2])
print("Encoded writer ID: ", encoded_img_targets[:2])
```

```
Writer ID        :  ['344' '013']
Encoded writer ID:  [338  13]
```

شکل ۴-۵ عملیات رمز گشایی



شکل ۵-۵ عملیات One-Hot-Encoding

```
y_train = convert_to_one_hot(y_train, 657).T
y_test = convert_to_one_hot(y_test, 657).T
y_val = convert_to_one_hot(y_val, 657).T
```

شکل ۶-۵ تبدیل به One-Hot-Encoding

۵/۲ پیاده سازی شبکه

در ادامه چهار گام اصلی keras را بررسی خواهد شد:

- Define Keras Model
- Compile Keras Model
- Fit Keras Model
- Evaluate Keras Model

۵/۲/۱ Define Keras Model

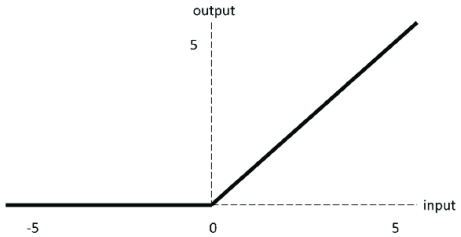
برای پیاده سازی این مدل از معماری ResNets 18 استفاده شده است. مزیت استفاده از معماری ResNets نسبت به معماری های دیگر این است که وقتی تعداد لایه ها زیاد می شود. و یا به عبارت دیگر مدل پیچیده تر و عمیق تر می شود Vanishing Gradient رخ می دهد. و روند آموزش را متوقف می کند. اما با استفاده از معماری ResNets سرعت یادگیری با کاهش Vanishing Gradient افزایش می یابد.

دلیل انتخاب تعداد لایه های ResNets آزمایشات مکرر و ارزیابی نتایج بوده است. با بررسی های مکرر این نتیجه حاصل شده است؛ که ۱۸ لایه پیچیدگی کمتر و تعداد پارامتر های مناسب تری دارد. علاوه بر اینکه از overfitting جلوگیری می کند منجر به خروجی بهتری (نسبت به تعداد لایه های بیشتر) شده است. با بهره گیری از تکنیک هایی مانند drop out، data augmentation و regularization از overfitting جلوگیری شده است.

```
def convolutional_block(X, f, filters, s = 2, training=True, initializer=glorot_uniform):

    # Retrieve Filters
    F1, F2, F3 = filters
    # Save the input value
    X_shortcut = X
    # First component of main path glorot_uniform(seed=0)
    X = Conv2D(filters = F1, kernel_size = 1, strides = (s, s), padding='valid',
                        kernel_initializer = initializer(seed=0))(X)
    X = BatchNormalization(axis = 3)(X, training=training)
    X = Activation('relu')(X)

    # Second component of main path
    X = Conv2D(filters =F2,  kernel_size = f, strides = (1,1), padding = 'same',
                        kernel_initializer = initializer(seed=0))(X)
    X = BatchNormalization(axis = 3)(X , training=training)
    X = Activation('relu')(X)
    X = Dropout(0.2)(X)

    # Third component of main path
    X = Conv2D(filters =F3, kernel_size = 1, strides = (1,1), padding = 'valid',
                        kernel_initializer = initializer(seed=0))(X)
    X = BatchNormalization(axis = 3)(X , training=training)
    X = Dropout(0.2)(X)

    # SHORTCUT PATH
    X_shortcut = Conv2D(filters =F3, kernel_size = 1, strides = (s,s), padding = 'valid',
                              kernel_initializer = initializer(seed=0))(X_shortcut)
    X_shortcut = BatchNormalization(axis = 3)(X_shortcut , training=training)

    # Final step: Add shortcut value to main path (Use this order [X, X_shortcut]), and pas
    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    return X
```

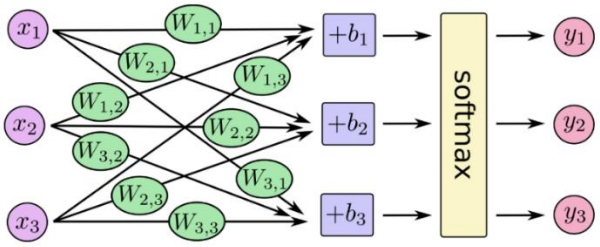
شکل ۵-۷ Convolutional Block شامل سه مسیر اصلی و یک مسیر میانبر

همانطور که در شکل ۵-۷ مشاهده می شود به منظور جلوگیری از کوچک شدن تصاویر و از دست رفتن اطلاعات در بعضی از مراحل padding=same در نظر گرفته شده است. همچنین از تابع فعال ساز Relu استفاده شده است. که عملکرد آن را بر روی خروجی در شکل ۵-۸ مشاهده خواهید کرد.



شکل ۵-۸ نحوه عملکرد تابع فعال ساز Relu

همانطور که در شکل ۱۰-۵ مشاهده می شود، ابتدا padding با سایز ۳ در نظر گرفته شده است، در کل ۵ مرحله وجود دارد. سپس از Average pooling استفاده شده و با این کار تعداد کانال ها افزایش داده شده است. در نهایت از تابع فعال ساز SoftMax استفاده شده است(شکل۹-۵)، زیرا طبقه بندی از نوع multi-class است.



شکل ۹-۵ تابع فعال ساز SoftMax

```
def ResNet18(input_shape , classes ):
  # Define the input as a tensor with shape input_shape
  X_input = Input(input_shape)
  # Zero-Padding
  X = ZeroPadding2D((3, 3))(X_input)
  # Stage 1
  X = Conv2D(64, (7, 7), strides = (2, 2),
             kernel_initializer = glorot_uniform(seed=0))(X)
  X = BatchNormalization(axis = 3)(X)
  X = Activation('relu')(X)
  X = layers.ZeroPadding2D((1, 1))(X)
  X = MaxPooling2D((3, 3), strides=(2, 2))(X)
  X = Dropout(0.2)(X)
  # Stage 2
  X = convolutional_block(X, f = 3, filters = [64, 64, 256], s = 1)
  X = identity_block(X, 3, [64, 64, 256])
  # Stage 3
  X = convolutional_block(X, f=3, filters=[128, 128, 512], s=2)
  X = identity_block(X, 3, [128, 128, 512])
  # Stage 4
  X = convolutional_block(X, f=3, filters=[256, 256, 1024], s=2)
  X = identity_block(X, 3, [256, 256, 1024])
  # Stage 5
  X = convolutional_block(X, f=3, filters=[512, 512, 2048], s=2)
  X = identity_block(X, 3, [512, 512, 2048])
  # AVGPOOL
  X = AveragePooling2D((2, 2), name='avg_pool')(X)
  # Include dropout with probability of 0.2 to avoid overfitting
  X = Dropout(0.2)(X)
  # Output layer
  regularizers.l2(l2=0.02)
  X = Flatten()(X)
  X = Dense(classes, activation='softmax' , kernel_regularizer='l2',
            kernel_initializer = glorot_uniform(seed=0))(X)
```

شکل ۱۰-۵ معماری ResNet18 پنج مرحله ای

با استفاده از کد موجود در شکل ۵-۱۱ ، مدل ساخته خواهد شد. در این بخش X عبارت است از عکس هایی که در فرمت grayscale به مدل داده شده و y عبارت است از نویسندگان که با کد هایی منحصر به فرد قابل تمایز هستند. خط دوم کد یک تصویر کلی از ساختار مدل ارائه می دهد.

```
model = ResNet18(input_shape = (64, 64, 1), classes = 657)
print(model.summary())
```

شکل ۵-۱۱ ساخت مدل

تعداد پارامتر های مدل ساخته شده به شرح زیر در جدول ۵-۲ آورده شده است:

Total Params	15,332,369
Trainable Params	15,301,521
Non-Trainable Params	30,848

جدول ۵-۲

۵٫۲٫۲ Compile Keras Model

برای محاسبه دقت از بهینه ساز Adam[12] که یکی از Adaptive optimizers است، استفاده شده است. این بهینه ساز به دلایل زیر انتخاب شده است:

- در بین بهینه ساز های سازگار Adam بهترین آن ها است.
- برای داده های پراکنده به خوبی عمل می کند.
- نیازی به تنظیم کردن مقدار نرخ یادگیری نیست.

علاوه بر موارد بالا با آزمایشات مکرر و امتحان کردن بهینه ساز های SGD و Adam در نهایت این نتیجه حاصل شد که Adam در این پروژه عملکرد مطلوب تری دارد. همچنین از Categorical Cross Entropy به منظور محاسبه ضرر استفاده شده است. (شکل۵-۱۲)

```
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
↪metrics=['accuracy'])
```

شکل ۵-۱۲ Compile مدل توسط بهینه ساز Adam

Categorical Cross Entropy[13] یک تابع ضرر است که در مدل هایی که خروجی آن ها به صورت multi-class است استفاده می شوند. عملکرد آن بدین صورت است که یک نمونه از بین دسته بندی های ممکن فقط می تواند به یک دسته بندی متعلق باشد و در واقع مدل تصمیم می گیرد که نمونه مورد نظر متعلق به کدام یک از دسته بندی ها است. فرمول محاسبه آن در شکل ۱۳-۵ آورده شده است:

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

شکل ۱۳-۵ فرمول محاسبه ضرر

۵,۲,۳ Fit Keras Model

Augmentation[14] یا داده افزایی به منظور افزایش داده ها با اعمال تغییراتی بر روی تصاویر می باشد تغییرات اعمال شده عبارتند از: (شکل ۱۴-۵ تا ۱۹-۵)

شکل ۱۴-۵ استفاده از تکنیک Shear Intensity

شکل ۱۵-۵ استفاده از تکنیک Horizontal Flip

شکل ۱۶-۵ استفاده از تکنیک Zoom

شکل ۱۷-۵ استفاده از تکنیک Vertical Flip

شکل ۱۸-۵ استفاده از تکنیک Rotation

شکل ۱۹-۵ استفاده از تکنیک Nearest

در شکل ۲۰-۵ کد این بخش مشاهده می شود:

```
train_datagen = ImageDataGenerator( zoom_range = 0.2 , horizontal_flip=True,␣
 ↪fill_mode="nearest", vertical_flip = True,rotation_range=5, shear_range=0.2)
```

شکل ۲۰-۵ اعمال داده افزایی

داده ها می بایست به دو دسته آموزش و تست (اعتبارسنجی) تقسیم شوند. در این پروژه نسبت داده های آموزش به تست به ترتیب ۸۰ به ۲۰ است.

۵،۲،٤ Cross Validation[15]

طبق دلایل گفته شده در فصل مبانی پژوهش برای بهبود نتایج طبق شکل ۲۱-۵ از روش Cross استفاده شده است.

```
for k, (train_index, test_index) in enumerate(kf.split(img_files)):

  X_train, X_test = img_files[train_index], img_files[test_index]
  y_train, y_test = encoded_img_targets[train_index],␣
 ↪encoded_img_targets[test_index]

  X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,␣
 ↪test_size=0.2, shuffle = True)

  y_train = convert_to_one_hot(y_train, 657).T
  y_test = convert_to_one_hot(y_test, 657).T
  y_val = convert_to_one_hot(y_val, 657).T

  train_datagen.fit(X_train)
  train_datagen.fit(X_val)
␣
 ↪print('----------------------------------------------------------------------
  print(f'Training for fold {k} ...')
  history = model.fit( train_datagen.flow(X_train, y_train, batch_size=128),␣
 ↪epochs = 70 , validation_data=(train_datagen.flow(X_val,y_val)) )
  history_list.append(history)
```

شکل ۲۱-۵ اعمال Cross Validation

با آزمایشات مکرر این نتیجه حاصل شد که بهترین مقادیر برای تعداد Fold، Epoch و Batch size از قرار زیر می باشند: (جدول ۳-۵)

Epoch= 70	Batch size = 128	Fold= 10

جدول ۳-۵

۵،۲،۵ Evaluate Keras Model

در بخش ارزیابی مدل داده های تست به عنوان ورودی به آن داده می شود(شکل ۲۲-۵)، چنانچه یادگیری مدل به درستی صورت گرفته باشد و به جای یادگیری حفظ کردن داده ها صورت نگرفته باشد، می بایست نتیجه پیش بینی داده های تست نزدیک به داده های آموزش باشد.

```
scores = model.evaluate(X_test, y_test, verbose=0)
```

شکل ۲۲-۵ ارزیابی مدل

در نهایت بعد از به پایان رسیدن اجرا نتایج از قرار زیر است: (۴-۵)

Folds	ضرر	دقت
Fold1	۳.۳۶	۳۶.۵۱
Fold2	۲.۷۸	۴۸.۸۷
Fold3	۵.۵۷	۵۴.۹۲
Fold4	۲.۱۷	۶۱.۹۷
Fold5	۱.۹۹	۶۵.۱۹
Fold6	۱.۴۵	۷۲.۹۵
Fold7	۱.۳۷	۷۵.۷۰
Fold8	۱.۱۹	۸۱.۰۱
Fold9	۱.۰۷	۸۲.۶۲
Fold10	۰.۹۹	۸۳.۶۴
میانگین:	۱.۵۶	۷۲.۷۷

جدول ۴-۵ ارزیابی مدل با استفاده از داده های تست

در تصویر فوق میزان دقت و نرخ ضرر هر زیرمجموعه به تفکیک نشان داده شده است. در نهایت نیز میانگین وزن دار از نتیجه پیش بینی داده های تست گرفته شده است. بیشترین دقت تست ۸۳/۶۴ درصد و کمترین نرخ ضرر تست برابر است با ۱ درصد. میانگین دقت تست ۷۲/۷۷ درصد و میانگین نرخ ضرر تست برابر است با ۱/۵۶.

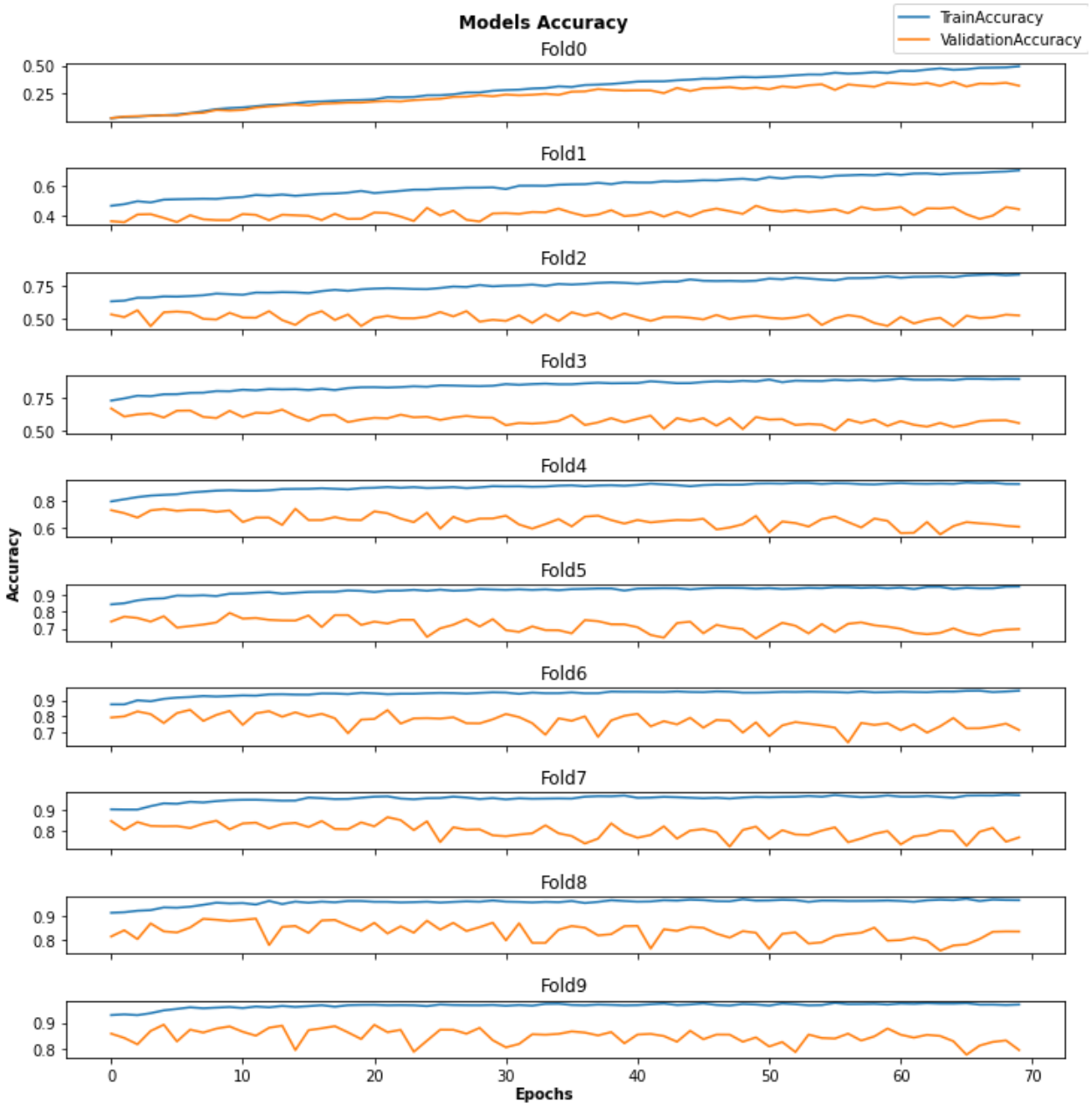
با توجه به جدول۵-۵[۱۶]، طبق بررسی های انجام شده در نتایج پژوهش های دیگر محققان این نتیجه حاصل شد که آن ها از ادغام دو روش متفاوت برای حصول نتیجه بهتر استفاده کردند. برای مثال ویژگی های تصویر را به کمک CNN استخراج کرده و سپس به کمک الگوریتمی دیگر مانند SVM نویسنده را تشخیص داده اند. در حالی که در این پروژه تنها از CNN و معماری ResNets استفاده شده و هر دو عمل استخراج ویژگی ها و تصمیم گیری ها توسط این معماری انجام شده است. و همچنین پایگاه داده های آن ها بیش از پایگاه داده های استفاده شده در این پروژه می باشد. که این خود به افزایش دقت کمک چشم گیری می کند.

نویسنده	پایگاه داده	دقت اندازه گیری شده
Hafemann	IAM	۹۱/۶۷
Hafemann	BFL	۹۵/۶۵
Tang	ICDAR2013	۹۹
Tang	CVL	۹۹/۷
Xing	IAM	۹۸/۲
Xing	HWDB1	۹۳/۸۵
Chen	ICDAR2013	۹۶/۶
Nguyen	IAM	۹۰/۱۲

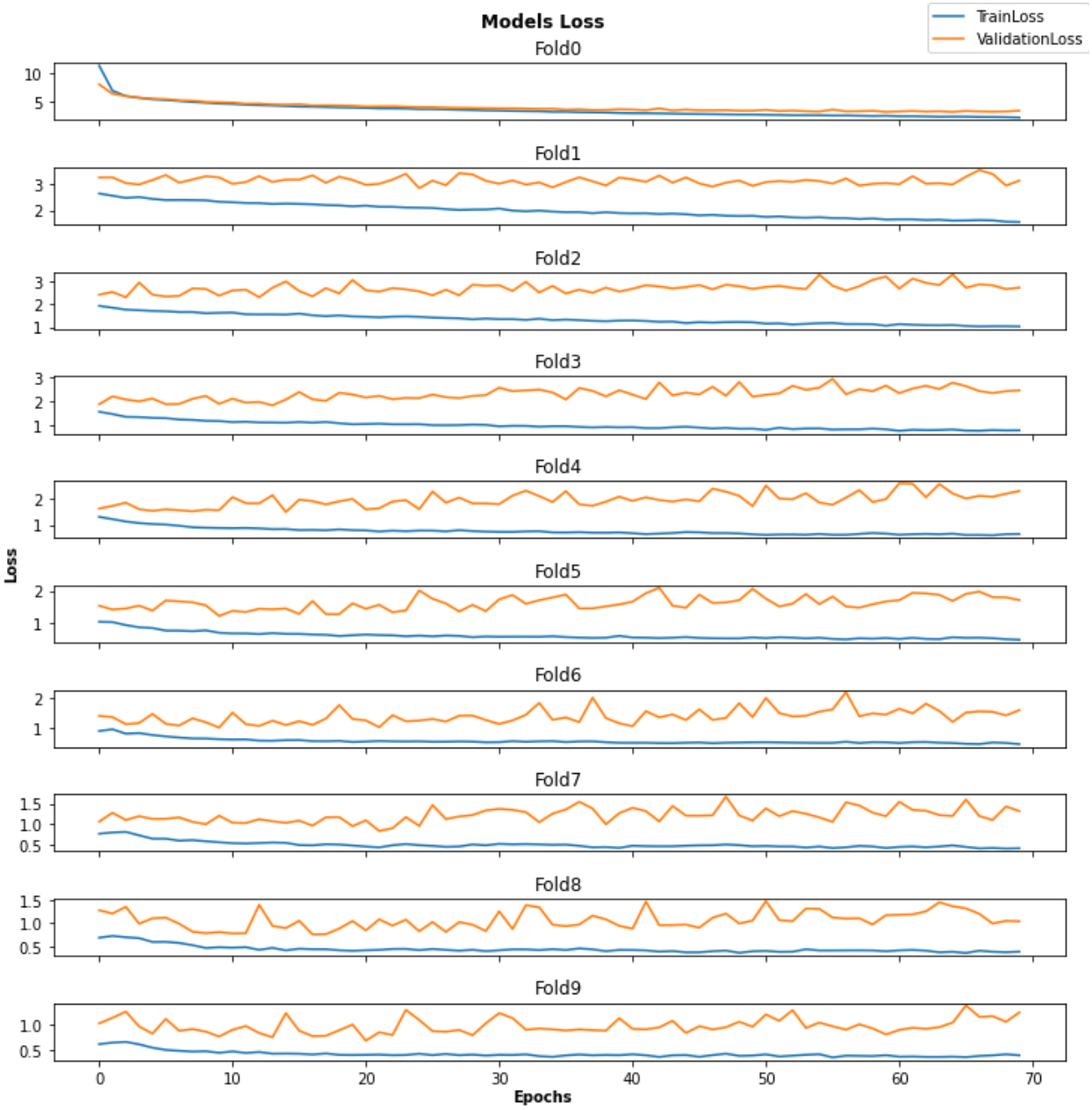
جدول ۵-۵

با توجه به دلایل ذکر شده در بالا رسیدن به دقت ۸۳/۶۴ درصد در این معماری در مقابل دقت ۹۶ درصد که میانگین دقت روش های پیشین می باشد. عددی قابل قبول است.

در مجموعه نمودار های ۱-۵ و ۲-۵ روند داده ها به تفکیک هر زیرمجموعه نمایش داده شده است:



نمودار ۱-۵ مقایسه دقت به تفکیک هر Fold



نمودار ۲-۵ مقایسه ضرر به تفکیک هر Fold

۶ فصل ششم

نتیجه گیری

۶/۱ نتیجه گیری

در این پژوهش از پایگاه داده IAM که دارای ۶۵۷ نویسنده می باشد، استفاده شده است. برای پیش پردازش تصاویر هم آن ها در اندازه ۶۴*۶۴ و در قالب Gray scale استفاده شده اند. برای استخراج ویژگی و ساختار طبقه بندی از شبکه های عصبی عمیق استفاده شده است. ResNets یک معماری عصبی به منظور کاهش پیچیدگی، حل تخریب و حفظ عملکرد خوب است. با کاهش پیچیدگی ، تعداد کمتری از پارامترها نیاز به آموزش دارند و همچنین زمان کمتری را صرف آموزش می کنند. معماری ResNets به علت داشتن مسیر های میانبر باعث افزایش دقت می شود. افزایش سرعت یکی دیگر از ویژگی های این معماری است. به همین دلیل در پیاده سازی این سیستم از این معماری استفاده شده است. زبان برنامه نویسی استفاده شده Python می باشد که به منظور سهولت از کتابخانه Keras استفاده شده است.

در این پژوهش برای جلوگیری از Overfit شدن از Augmentation، Dropout، Regularization استفاده شده است.

در پایان نتایج زیر حاصل شد:

Accuracy Test = 83.64	Loss Test = 1
Accuracy Train = 0.97	Loss Train = 0.39

جدول ۱-۶

برای بهبود عملکرد این سیستم پیشنهادات زیر ارائه می شود:

- استفاده از معماری های ترکیبی، مانند ترکیب شبکه های عصبی و SVM
- استفاده از Transfer Learning
- افزایش تعداد پایگاه داده ها و استفاده همزمان از آن ها

۷ مراجع

[1]Chaitali Dhaware, Mrs. K. H. Wanjale, “Survey On Image Classification Methods In Image Processing”, Department of Computer Engineering, Vishwakarma Institute of Information Technology Pune-India, International Journal of Computer Science Trends and Technology (IJCS T) – Volume 4 Issue 3, May - Jun 2016

[2]https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[3]https://towardsdatascience.com/5-reasons-why-you-should-use-cross-validation-in-your-data-science-project-8163311a1e79

[4] https://en.wikipedia.org/wiki/Ground_truth

[5] L. G. Hafemann, “An analysis of deep neural networks for texture classification,” 2014, 2014.89 f. Dissertagáo (Mestrado em Ciéncia da Computagáo) – Universidade Federal do Paraná, Curitiba, PR, BR.

[6] https://fki.tic.heia-fr.ch/databases/iam-handwriting-database

[7] D. Bertolini, L. S. Oliveira, E. Justino, and R. Sabourin, ‘Texture-based descriptors for writer identification and verification,” Expert Systems with Applications, vol. 40, no. 6, pp. 2069-2080, 2013.

[8]Y. Tang and X. Wu, “Text-independent writer identification via cnn features and joint bayesian,” Frontiers in Handwriting Recognition, 15th International Conference on. IEEE, pp. 566-571, 2016.

[9]L. Xing and Y. Qiao, “Deepwriter: A multi-stream deep cnn for textindependent writer identification,” Frontiers in Handwriting Recognition, 15th International Conference on. IEEE, pp. 584-589,2016.

[10] S. Chen, Y. Wang, C.-T. Lin, and Z. Cao, “Semi-supervised feature learning for off-line writer identifications,” arXiv, 2018.

[11] H. T. Nguyen, C. T. Nguyen, T. Ino, B. Indurkhya, and M. Nakagawa,‘Text-independent writer identification using convolutional neural network,” Pattern Recognition Letters, 2018.

[12] https://towardsdatascience.com/7-tips-to-choose-the-best-optimizer-47bb9c1219e

[13]https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy

[14]https://towardsdatascience.com/exploring-image-data-augmentation-with-keras-and-tensorflow-a8162d89b844

[15] https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[16] Lucas G. Helal, Diego Bertolini, Yandre M. G. Costa, George D. C. Cavalcanti, Alceu S. Britto Jr., Luiz E. S. Oliveira