Speech Enhancement and Denoising

پروژه پایانی درس پردازش گفتار

دانشکده مهندسی کامپیوتر

استاد: دکتر صامتی

بهار ۱۴۰۳

بهار بهزادی‌پور ۴۰۲۲۰۷۱۵۵

مهتا فطرت ۴۰۲۲۱۲۲۲۵

فهرست مطالب

۱ مقدمه **۲**
۱.۱ شرح مسئله ۲
۱.۱.۱ شبکه‌های عصبی چندلایه ۲
۲.۱.۱ کتابخانه‌ها ۳
۲.۱ شرح مسائل ۴
۱.۲.۱ تخمین توابع ۴
۲.۲.۱ دسته‌بندی ۵
۳.۲.۱ رفع نویز ۶

۲ بخش اول **۷**
۱.۲ داده‌ها ۷
۲.۲ مدل ۸
۱.۲.۲ تابع اول ۸
۲.۲.۲ تابع دوم ۹
۳.۲.۲ تابع سوم ۱۰
۳.۲ نتایج ۱۰
۱.۳.۲ تابع اول ۱۰
۲.۳.۲ تابع دوم ۱۰
۳.۳.۲ تابع سوم ۱۰

۳ بخش دوم **۱۱**
۱.۳ داده‌ها ۱۱
۲.۳ مدل ۱۱
۱.۲.۳ تابع اول ـ نویز نزدیک به صفر ۱۱
۲.۲.۳ تابع دوم ـ نویز متوسط ۱۲
۳.۲.۳ تابع سوم ـ نویز بسیار بزرگ ۱۲
۳.۳ نتایج ۱۳
۱.۳.۳ تابع اول ۱۳
۲.۳.۳ تابع دوم ۱۴
۳.۳.۳ تابع سوم ۱۴

۴ بخش سوم **۱۵**
۱.۴ داده‌ها ۱۵
۲.۴ مدل ۱۶
۱.۲.۴ تابع اول ـ ورودی دوبعدی خروجی یک بعدی ۱۶
۲.۲.۴ تابع دوم ـ ورودی دوبعدی خروجی دو بعدی ۱۶
۳.۴ نتایج ۱۶
۱.۳.۴ تابع اول ۱۶
۲.۳.۴ تابع دوم ۱۶

۵ بخش چهارم **۱۷**
۱.۵ داده‌ها ۱۷
۲.۵ مدل ۱۷
۳.۵ نتایج ۱۸

۶ بخش پنجم **۱۹**
۱.۶ داده‌ها ۱۹
۲.۶ مدل ۱۹
۳.۶ نتایج ۲۰

۷ بخش ششم **۲۱**
۱.۷ داده‌ها ۲۱
۲.۷ مدل ۲۱
۳.۷ نتایج ۲۲

فصل ۱

مقدمه

بهبود کیفیت گفتار و رفع نویز آن همواره مورد توجه بوده‌است و جزء معدود مسائل حوزه‌ی پردازش گفتار است که می‌توان گفت هنوز کاملا حل‌شده تلقی نمی‌شو و جای پیشرفت دارد. با این حال، مدل‌ها و ابزارهای بسیاری برای بهبود گفتار زبان انگلیسی وجود دارد که از کیفیت مطلوبی نیز برخوردار هستند. در این میان، زبان فارسی نسبت له تکنولوژی‌های اخیر در این زمینه جای پیشرفت زیادی دارد. بررسی‌های اولیه نشان می‌دهد که مدل بهبود گفتاری که برای زبان فارسی مناسب‌سازی شده باشد در حال حاضر وجود ندارد. در این پروژه ما در تلاش هستیم تا وضعیت ابزارهای حاضر را برای بهبود گفتار فارسی بررسی کنیم و بتوانیم در راه بهبود آن‌ها گام‌های موثری برداریم.

۱.۱ شرح مسئله

بهبود گفتار (Speech Enhancement) حذف نویز (Denoising) به فرآیندهایی اطلاق میشود که کیفیت و وضوح سیگنال گفتار را بهبود میبخشند و نویزهای پس زمینه را کاهش میدهند. این مسئله در کاربردهای مختلفی از جمله سیستم های تشخیص گفتار، ارتباطات تلفنی، و کمک به افراد با مشکلات شنوایی اهمیت دارد. در زبان فارسی، به دلیل تفاوت های صوتی و ساختاری با دیگر زبان ها، نیاز به تحقیق و توسعه ویژه ای در این زمینه وجود دارد.

۱.۱.۱ شبکه‌های عصبی چندلایه

مبحث شبکه‌های عصبی به یادگیری با الگو گرفتن از ساختار سلول‌های عصبی انسان می‌پردازد. هر سلول عصبی بدن انسان، دارای رشته‌های ورودی اطلاعات (در اثر محرک‌ها) به نام دندریت و رشته‌ی(های) خروجی اطلاعات به نام آکسون است. (تصویر **؟؟**[۱])

هر نورون یک آستانه‌ی تحریک دارد. هرگاه محرک‌های ورودی آن از حد مشخصی فراتر بروند باعث تحریک و پاسخگویی آن می‌شوند. این نورون ها تحت شبکه‌های بزرگی از طریق رشته‌های مذکور به هم متصل می‌شوند و یک پیام ورودی را از محل محرک‌های اولیه به محل هدف منتقل می‌کنند و دستور خاصی را در آن محل باعث می‌شوند.
در حوزه‌ی هوش مصنوعی با الگو گرفتن از این ساختار، هر نورون شامل یک یا چندین ورودی، یک تابع

[۱] https://en.wikipedia.org/wiki/Neuron

فعال‌ساز و یک یا چندین خروجی است. (تصویر **؟؟**) همچنین هر نورون در شبکه‌ی عصبی مصنوعی دارای یک وزن (اهمیت) است و ورودی نهایی نورون برابر مجموع وزن‌دار ورودی‌های آن بعلاوه‌ی یک bias است. این وزن‌ها و bias ها درواقع پارامترهای مسئله‌ی یادگیری هستند که باید بهینه‌سازی شوند. مجموع وزن‌دار مذکور، وارد یک تابع فعال‌ساز می‌شود که خروجی نهایی نورون را تشکیل می‌دهد. تابع فعال‌ساز انواعی دارد که در بخش‌های بعد با برخی از آن‌ها آشنا خواهیم شد.

نورون‌ها در ساختار شبکه‌ای به هم متصل می‌شوند (خروجی برخی، ورودی دیگری می‌شود). ساختار این شبکه، از جمله متغیرهای مسئله است و باید مخصوص به هر مسئله مناسب‌سازی شود. لایه‌ی اول نورون‌ها لایه‌ی ورودی و لایه‌ی آخر لایه‌ی خروجی نامیده می‌شود. لایه‌های دیگر به عنوان لایه‌های پنهان شناخته می‌شوند. (تصویر **؟؟**[۲])

مسئله‌ی یادگیری در شبکه‌های عصبی درواقع یادگیری وزن‌ها، مقادیر bias و حتی اتصالات بین نورون‌ها را شامل می‌شود. برای یادگیری این پارامترها، لازم است که ابتدا یک معیار برای ارزیابی شبکه داشته باشیم. این معیار تحت عنوان loss function تعریف می‌شود و می‌تواند توسط توابع مختلفی توصیف شود که در بخش‌های آینده به معرفی انواعی از آن خواهیم پرداخت. در یک دیدگاه کلی، این توابع یک معیار تفاوت بین خروجی مطلوب و خروجی شبکه را به دست می‌دهند که کمینه کردن این تفاوت (تا حدی که موجب بیش‌برازش نشود)، به پاسخ مطلوب مسئله منجر می‌شود.
به این ترتیب شبکه‌ی عصبی برای یادگیری بهترین وزن‌ها و bias ها، از داده‌های آموزشی استفاده می‌کند، خروجی ابتدایی خود را (با پارامترهایی که مثلا به صورت رندوم مقداردهی‌اولیه شده‌اند) با استفاده از loss function ارزیابی می‌کند و سعی می‌کند پارامترهایش را در هر گذر از داده‌ها، در جهت کاهش مقدار خطا بروزرسانی کند. با پیمایش چند باره‌ی داده‌ها، می‌توان به دقت‌های بهتری برای شبکه دست یافت.

۲.۱.۱ کتابخانه‌ها

مدیریت داده‌ها

به منظور خواندن اطلاعات پایگاه‌داده‌ها، تبدیل آن‌ها به فرمت ورودی مناسب برنامه، ساختار داده‌ی مناسب نگه‌داری و انجام محاسبات بر روی داده‌ها، مقیاس‌بندی داده‌ها و تقسیم آن‌ها به مجموعه‌های آموزشی و آزمایشی نیاز به امکانات و toolkit هایی از زبان پیاده‌سازی، پایتون، داریم. این کتابخانه‌ها به شرح زیر انتخاب شده‌اند.

- sklearn.preprocessing: برای انجام نرمال‌سازی، centralizing، scaling و به طور کلی پیش‌پردازش داده‌ها
- sklearn.model_selection: به منظور تقسیم مجموعه‌ی داده‌ها به دو دسته‌ی تست و آموزش به نسبت مطلوب
- numpy: ساختمان‌داده‌ی اصلی مورد استفاده برای مدیریت و manipulation داده‌ها
- pandas: برای خواندن و نمایش فایل‌های با فرمت csv

[۲] https://en.wikipedia.org/wiki/Neural_network

مدیریت شبکه‌ی عصبی

برای طراحی و ایجاد ساختار لایه‌بندی شبکه‌ی عصبی، مقداردهی اولیه متغیرهای شبکه، آموزش و بهینه‌سازی شبکه‌ی عصبی، استفاده از توابع فعال‌ساز و به طور کلی ایجاد، یادگیری و پاسخ‌دهی شبکه‌ی عصبی از کتابخانه‌ی tensorflow.keras استفاده شده است.
این کتابخانه امکاناتی برای مدیریت شبکه‌های عصبی در اختیار می‌گذارد که در ادامه موارد مربوط به پروژه‌ی حاضر را برمی‌شماریم. انواع لایه‌های شبکه‌ی عصبی از جمله

- Dense
- Conv۲D
- Reshape
- Flatten،

انواع توابع فعال‌ساز از جمله

- linear
- relu
- sigmoid
- softmax
- tanh

انواع loss_function از جمله

- Mean Square Error
- Sparce Categorical Cross Entropy

انواع بهینه‌ساز ها از جمله adams که در این پروژه به کار گرفته شده.
انواع initializer ها که در این پروژه از مقدار default آن‌ها استفاده شده‌است.
و در آخر، آموزش و پیش‌بینی‌های شبکه بر روی داده‌های آموزشی و آزمایشی هم توسط این کتابخانه مدیریت می‌شوند.

مصورسازی داده‌ها و نتایج

به منظور مصورسازی و شهود نسبت به داده‌های ورودی، نمایش تصاویر، نمایش توابع هدف و نمایش نتایج پیش‌بینی مدل به همراه نتایج صحیح به صورت تصویری از کتابخانه‌ی matplotlib.pyplot استفاده شده است.

۲.۱ شرح مسائل

۱.۲.۱ تخمین توابع

در این بخش، به کاربرد شبکه‌های عصبی در تخمین توابع پرداخته‌شده.

بخش اول ـ تخمین توابع با ورودی یک بعدی

مطلوب این بخش، حالت خاصی از تخمین توابع به عنوان توابع با ورودی یک‌بعدی است. ابتدا لازم است که توابعی با این ویژگی به عنوان توابع هدف تولید شوند. سپس برای هر بخش لازم است که شبکه‌های عصبی با ابعاد، توابع فعال‌ساز، نوع لایه‌بندی، و تابع خطای مناسب طراحی شوند. در آخر این شبکه‌های عصبی آموزش داده می‌شوند و نتایج آنها مورد بررسی و ارزیابی قرار می‌گیرد. همچنین لازم است که به تاثیر عناصر مختلف مسئله بر روی نتایج پرداخته شود. جزئیات مربوط به پیاده‌سازی و تحلیل نتایج هر یک از بخش‌های مورد توضیح در فصل‌های آینده خواهد آمد.

بخش دوم ـ افزودن نویز به داده‌ها

در این بخش نیز مانند بخش گذشته، به تخمین توابع با ورودی یک‌بعدی پرداخته شده با این تفاوت که داده‌های دنیای واقعی به دلایلی اعم از خطای انسانی، ماشینی، دقت لوازم اندازه‌گیری و یا به علت ماهیت نادقیق داده‌ها، دارای نویز یا خطا باشند. در این صورت رفتار شبکه‌ی عصبی تغییر خواهد کرد. این موضوع در این بخش بر اساس اندازه‌ی نویزهای متفاوت مورد بررسی و تحلیل قرار خواهد گرفت.

بخش سوم ـ تخمین توابع با ابعاد بالاتر

در این بخش به تخمین توابع با ابعاد بالاتر هم برای ورودی و هم برای خروجی پرداخته خواهد شد. اگرچه شبکه‌های عصبی محدودیتی در ابعاد تابع مورد تخمین ندارند، در این بخش به بررسی یک تابع با ورودی دو بعدی و خروجی یک بعدی و یک تابع با ورودی دو بعدی و خروجی دو بعدی پرداخته شده است تا نتایج در سیستم مختصات سه بعدی قابل نمایش باشند. برای نمایش گرافیکی تابع با خروجی دو بعدی نیز این دو خروجی که از هم وابسته هستند، در دو نمودار مجاور هم نمایش داده شده‌اند.

بخش چهارم ـ تخمین توابع نامنظم

در این بخش به بررسی توابعی پرداخته شده که ضابطه‌ی مشخصی ندارند. این توابع می‌توانند با پرش‌های ناگهانی در نقاط خروجی خود همراه باشند. همچنین از الگوی همواری پیروی نمی‌کنند. در این بخش لازم است که ابتدا یک چنین تابعی به صورت دستی تولید شده و سپس رقومی‌سازی (دیجیتایز) شده و به عنوان تابع هدف مد نظر قرار داده شود. سپس این توابع به روشی مشابه روش‌های قبل تخمین زده شده و نتایج متفاوت با قسمت‌های قبل مورد تحلیل قرار گرفته‌اند.

۲.۲.۱ دسته‌بندی

در این بخش به کاربرد شبکه‌های عصبی در دسته‌بندی داده‌ها پرداخته می‌شود که یک نوع یادگیری supervised است. در این مسائل، داده‌های برچسب‌زده‌شده به شبکه ورودی داده می‌شوند. شبکه با توجه به ساختار داخلی خود (پارامترها، لایه‌بندی و ...) به هر داده یک دسته‌بندی نسبت می‌دهد. این دسته‌بندی می تواند به صورت یک احتمال برای هر دسته خروجی داده‌شود. سپس این خروجی با برچسب صحیح داده‌ها مقایسه شده، تحت یک تابع loss ارزیابی شده و در فاز آموزش این تصمیم‌گیری بهبود داده می‌شود.

بخش پنجم ـ دسته‌بندی داده‌های تصویری

در این بخش باید مسئله‌ی دسته‌بندی بر روی یک پایگاه‌داده‌ی تصویری/صوتی/... پیاده‌سازی شود. پایگاه‌داده‌ی انتخابی برای این بخش، پایگاه‌داده‌ی ارقام دست‌نویس MNIST است. در ابتدای این بخش، شهودی نسبت به داده‌ها به دست آورده می‌شود. سپس یک شبکه‌ی عصبی با توابع ارزیابی مخصوص منظور دسته‌بندی، در این بخش، طراحی و آموزش داده می‌شود. سپس این شبکه مورد استفاده‌ی عملی قرار گرفته و مورد ارزیابی قرار می‌گیرد.

۳.۲.۱ رفع نویز

کاربرد دیگری از شبکه‌های عصبی رفع نویز در داده‌های صوتی/تصویری/... است. این کاربرد ذیل مفهوم autoencoder ها مطرح می‌شود.

Autoencoders

معمولا داده‌های صحیح دامنه‌ی مورد بررسی، نسبت به همه‌ی داده‌های موجود، ساخت‌یافته‌تر هستند و در دامنه‌ی محدودتری جای می‌گیرند. یک نمونه از این مسئله یک تابع مارپیچ در یک فضای سه‌بعدی است. چنین تابعی علی‌رغم اینکه با داده‌های سه بعدی قابل بیان است، به علت ساختار خاص و منظمی که دارد، قابل ذخیره‌سازی با تنها یک پارامتر است. (تصویر **؟؟**[۳]).

autoencoder ها از چنین خصوصیتی بهره می‌گیرند. autoencoder ها دو بخش encoder و decoder دارند. بخش encoder داده‌ها را از ابعاد موجود، به ابعاد کوچک‌تری می‌برد و به عبارتی encode می‌کند. این داده‌ها با ابعاد کمتر، تنها دارای ویژگی‌های مهم‌تر و ساختارمندتر داده‌ها هستند. سپس بخش decoder می‌آموزد که چگونه این داده‌ها از ابعاد پایین‌تر را به ابعاد بالاتری ببرد و به عبارتی بازسازی کند.
autoencoder ها کاربردهای متنوعی دارند. از جمله‌ی این کاربردها تشخیص آنومالی در داده‌ها می‌باشد. چراکه داده‌هایی که دارای آنومالی باشند، درواقع از ساختار خاص آن مجموعه داده پیروی نمی‌کنند. بنابراین نمی‌توانند در ابعاد محدود کد شده، ذخیره‌سازی و نتیجتا بازیابی شوند. در نتیجه با داده‌ی خروجی خود اختلاف زیادی دارند. کاربرد دیگر آن‌ها denoising (**؟؟**[۴] تصویر) است. چرا که نویز داده‌ها که جزو ویژگی‌های اصلی و ساخت‌یافته داده‌ها نمی‌باشد، در ساختار کاهشی/افزایشی autoencoder ها از بین رفته و داده‌های خروجی صحیح و بدون نویز یادگرفته می‌شوند.

بخش ششم ـ رفع نویز داده‌های تصویری

در بخش آخر این پروژه، کاربرد شبکه‌های عصبی به صورت عملی در پایگاه‌داده‌ی انتخابی در بخش قبل مورد بررسی قرار می‌گیرد. ابتدا لازم است در داده‌های آموزشی و آزمایشی نویز ایجاد شود. سپس این مجموعه داده‌ی جدید به عنوان ورودی به شبکه‌ی عصبی داده‌شده و خروجی مطلوب شبکه نیز از مجموعه‌ی اولیه گرفته می‌شود. سپس توانایی واقعی مدل آموزش‌دیده در رفع نویز از داده‌های نویزدار تست که تا به حال ندیده‌است بررسی می‌شود.

[۳] https://en.wikipedia.org/wiki/Helix
[۴] https://en.wikipedia.org/wiki/Autoencoder

فصل ۲

بخش اول

۱.۲ داده‌ها

در این بخش ابتدا تعدادی (۳) تابع (تصویر **؟؟**) با ورودی و خروجی یک‌بعدی به شرح زیر به عنوان توابع هدف ایجاد شده اند.

۱. $f(x) = x^2$

۲. $f(x) = xsin(x) + x$

۳. $f(x) = \frac{1}{0.5+e^{e^x(1-x)x^2}}$

این توابع به صورت یک black box از طریق ایجاد مجموعه نقاط آموزشی و آزمایشی به شبکه‌ی عصبی عرضه خواهند شد. به این ترتیب که یک بازه‌ی دلخواه از ورودی در نظر گرفته شده، خروجی‌های صحیح مربوط به آن از ضابطه‌ی تابع استخراج شده و مجموعه‌ی این نقاط، پایگاه‌داده‌ی مسئله را تشکیل می‌دهند.
این داده‌ها سپس برای عملکرد بهتر شبکه scale می‌شوند. به منظور مقیاس‌بندی داده‌ها از MinMaxScaler استفاده شده‌است که داده ها را با رابطه‌ي ۱.۲ به مقیاس بازه‌ی ۰ تا ۱ می‌برند و این امکان را به دست می‌دهند که داده‌ها را با وارون transformation انجام شده ۲.۲، به حالت اولیه بازگرداند.

$$X_std = (X - X.min(axis = 0))/(X.max(axis = 0) - X.min(axis = 0)) \quad (۱.۲)$$

$$X_scaled = X_std * (max - min) + min \quad (۲.۲)$$

این داده‌ها سپس با نسبت ۸۵ به ۱۵ به مجموعه‌های آموزشی و آزمایشی تقسیم می‌شوند. مجموعه‌های آزمایشی برای فاز آموزش و مجموعه‌های آزمایشی برای فاز آزمایشی شبکه‌ی عصبی مورد استفاده خواهند بود.

۲.۲ مدل

۱.۲.۲ تابع اول

تعریف مدل ۱

در این قسمت از یک مدل ۲ لایه با ورودی ۱ بعدی، ۱۰ نورون در لایه‌ی اول و ۱۵ نوورن در لایه‌ی دوم و خروجی یک بعدی استفاده شده‌است. استفاده از یک لایه موجب نیاز به تعداد زیادی نورون در لایه‌ی اول می‌شود و با کمی آزمون و خطا نتیجه می‌شود که ساختاری دولایه با ۱۰ـ۱۵ نورون در هر لایه به نتایج مطلوبی منجر می‌شود. در بخش‌های بعد، خواهیم دید که با افزایش و کاهش حجم شبکه، عملکرد آن به چه صورت تغییر می‌کند.
تابع فعالساز relu انتخاب شده است. (تصویر ؟؟[۱]). این تابع از کارآمدترین توابع فعال‌ساز بوده و با سرعت بیشتری به نتایج بهتری منجر می‌شود. استفاده از توابع دیگری از جمله sigmoid به علت ماهیت مسئله که تخمین مقدار خروجی تابع است کارساز نیستند زیرا مقادیر نزدیک به یک را تنها در ورودی‌های بسیار بزرگ می‌دهند.

مدل با ۱۰۰ iteration و batch_size برابر ۱۰ آموزش داده شده است. batch_size های پایین‌تر، (به عنوان مثال عدد ۵ آزمایش شد) به دقت‌های پایین‌تر منجر می‌شد و چرخه‌های یادگیری پایین‌تر نیز امکان رسیدن به حالت مطلوب را نداشتند و ۱۰۰ یک بازه‌ی اطمینان خوب پس از انجام آزمایش‌های متعدد است.

ارزیابی مدل ۱

ارزیابی این مدل با تابع MSE یک loss برابر $1.5e-04$ نتیجه می‌دهد که نسبت به سایر مدل‌ها عملکرد بهتری دارد. (تصویر ؟؟)

کاهش داده‌های آموزشی

در صورت کاهش ۱۵ درصدی داده‌های آموزشی با همین مدل با همین پارامترها، کیفیت نتایج به شدت افت می‌کند و دقت آن بسیار کاهش می‌یابد. (تصویر ؟؟)

کاهش چرخه‌های آموزشی

در صورت کاهش تعداد چرخه‌های آموزشی به تعداد ۵۰ چرخه، با همین مدل با همین پارامترها، کیفیت نتایج به شدت افت می‌کند و دقت آن بسیار کاهش می‌یابد. (تصویر ؟؟)

تعریف مدل ۲ ـ افزایش حجم شبکه

در این قسمت از یک مدل ۲ لایه با ورودی ۱ بعدی، ۱۰۰ نورون در لایه‌ی اول و ۵۰ نوورن در لایه‌ی دوم و خروجی یک بعدی استفاده شده‌است. علی‌رغم اینکه این تابع به کمک یک شبکه‌ی کوچک‌تر با دقت قابل‌قبولی تخمین زده‌شده است، برای نمایش تاثیر حجم مدل، مدل بزرگ‌تری انتخاب شده و این مدل به نتایج بسیار بهتری منجر می‌شود. سایر پارامترها از جمله تعداد چرخه‌ها،توابع فعال‌ساز، batch_size

[۱] https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

و ... تغییری نکرده‌اند.

ارزیابی مدل ۲

ارزیابی این مدل با تابع MSE یک loss برابر $1.2e-05$ نتیجه می‌دهد که نسبت به سایر مدل‌ها عملکرد بهتری دارد. نتایج مصور این مدل در بخش نتایج خواهد آمد.

تعریف مدل ۳ ـ کاهش حجم شبکه

در این قسمت از یک مدل ۲ لایه با ورودی ۱ بعدی، ۵ نورون در لایه‌ی اول و ۸ نوورن در لایه‌ی دوم و خروجی یک بعدی استفاده شده‌است. علی‌رغم اینکه این تابع به کمک یک شبکه‌ی بزرگ‌تر با دقت قابل‌قبولی تخمین زده‌شده است، برای نمایش تاثیر حجم مدل، مدل کوچک‌تری انتخاب شده و این مدل به نتایج بسیار بدتری منجر می‌شود. سایر پارامترها از جمله تعداد چرخه‌ها،توابع فعال‌ساز، batch_size و ... تغییری نکرده‌اند.

ارزیابی مدل ۳

ارزیابی این مدل با تابع MSE یک loss برابر 0.0015 نتیجه می‌دهد که نسبت به سایر مدل‌ها عملکرد بدتری دارد. (تصویر **؟؟**)

۲.۲.۲ تابع دوم

تعریف مدل ۱

در این قسمت به علت پیچیده‌شدن تابع هدف، از یک مدل ۲ لایه با ورودی ۱ بعدی، ۱۰۰ نورون در لایه‌ی اول و ۱۰۰ نوورن در لایه‌ی دوم و خروجی یک بعدی استفاده شده‌است. این اعداد به انجام آزمایش‌های متعدد به دست‌آمده و استفاده از حتی ۸۰ نورون در هر لایه، باعث نامطلوب شدن خروجی مدل می‌گشت.
تابع فعال‌ساز به دلیل مطرح شده در قسمت قبل، relu بوده و تابع loss هم همچنان MSE است که برای مقاصد تخمین تابع مناسب می‌باشد.
توجه داریم در این قسمت به علت بزرگ‌تر شدن حجم شبکه، به تعداد چرخه‌های آموزشی بیشتری نیاز داریم. این تعداد با آزمون و خطا برابر ۱۵۰۰ در نظر گرفته شده. آزمایشات نشان دادند که تعداد چرخه‌های حتی تا ۱۰۰۰ چرخه، نمی‌توانستند خطای مدل را به زیر مرتبه‌ی یک هزارم کاهش دهند. همچنین batch_size بزرگ‌تر از ۵، منجر به افزایش loss مدل می‌گشت.

ارزیابی مدل ۱

ارزیابی این مدل با تابع MSE یک loss برابر $3.2e-04$ نتیجه می‌دهد که نسبت به سایر مدل‌های آزمایش‌شده عملکرد بهتری دارد. نتایج مصور این مدل در قسمت نتایج آورده شده‌است.

۳.۲.۲ تابع سوم

تعریف مدل ۱

در این قسمت اگرچه ضابطه‌ی تابع هدف نسبت به تابع قبل پیچیدگی بیش‌تری دارد، همچنان از یک مدل ۲ لایه با ورودی ۱ بعدی، ۱۰۰ نورون در لایه‌ی اول و ۱۰۰ نوورن در لایه‌ی دوم و خروجی یک بعدی استفاده شده‌است. مشابه تابع دوم، این اعداد به انجام آزمایش‌های متعدد به دست‌آمده و استفاده از حتی ۸۰ نورون در هر لایه، باعث نامطلوب شدن خروجی مدل می‌گشت.
تابع فعال‌ساز به دلیل مطرح شده در قسمت قبل، relu بوده و تابع loss هم همچنان MSE است که برای مقاصد تخمین تابع مناسب می‌باشد.
توجه داریم در این قسمت به علت بزرگ‌تر شدن حجم شبکه، به تعداد چرخه‌های آموزشی بیشتری نیاز داریم. این تعداد با آزمون و خطا برابر ۵۰۰ در نظر گرفته شده. آزمایشات نشان دادند که تعداد چرخه‌های حتی تا ۴۰۰ چرخه، نمی‌توانستند خطای مدل را به زیر مرتبه‌ی یک هزارم کاهش دهند. همچنین batch_size بزرگ‌تر از ۵، منجر به افزایش loss مدل می‌گشت.

ارزیابی مدل ۱

ارزیابی این مدل با تابع MSE یک loss برابر $3.8e-05$ نتیجه می‌دهد که نسبت به سایر مدل‌های آزمایش‌شده عملکرد بهتری دارد. نتایج مصور این مدل در قسمت نتایج آورده شده‌است.

۳.۲ نتایج

۱.۳.۲ تابع اول

بهترین نتایج به دست آمده برای این تابع ناشی از مدل ۲ (معرفی شده در بخش مدل‌ها) است. (تصویر **؟؟**)

۲.۳.۲ تابع دوم

بهترین نتایج به دست آمده برای این تابع ناشی از مدل ۱ (معرفی شده در بخش مدل‌ها) است. (تصویر **؟؟**)

۳.۳.۲ تابع سوم

بهترین نتایج به دست آمده برای این تابع ناشی از مدل ۱ (معرفی شده در بخش مدل‌ها) است. (تصویر **؟؟**)

فصل ۳

بخش دوم

۱.۳ داده‌ها

در این بخش به بررسی تاثیر نویز در داده‌ها بر کارایی شبکه خواهیم پرداخت. به این منظور یک نویز با توزیع گاوسی به توابع اضافه می‌کنیم. این کار را از یک نویز نزدیک به صفر آغاز می‌کنیم و تا نویز بسیار بزرگی ادامه می‌دهیم. (تصویر **؟؟**)

این توابع به صورت یک black box از طریق ایجاد مجموعه نقاط آموزشی و آزمایشی به شبکه‌ی عصبی عرضه خواهند شد. به این ترتیب که یک بازه‌ی دلخواه از ورودی در نظر گرفته شده، خروجی‌های صحیح (هرچند دارای نویز) مربوط به آن از ضابطه‌ی تابع استخراج شده و مجموعه‌ی این نقاط، پایگاه‌داده‌ی مسئله را تشکیل می‌دهند.

این داده‌ها سپس برای عملکرد بهتر شبکه scale شده‌اند. مشابه بخش قبل، به منظور مقیاس‌بندی داده‌ها از MinMaxScaler استفاده شده‌است.

این داده‌ها سپس با نسبت ۸۵ به ۱۵ به مجموعه‌های آموزشی و آزمایشی تقسیم می‌شوند. مجموعه‌های آزمایشی برای فاز آموزش و مجموعه‌های آزمایشی برای فاز آزمایشی شبکه‌ی عصبی مورد استفاده خواهند بود.

۲.۳ مدل

۱.۲.۳ تابع اول ـ نویز نزدیک به صفر

تعریف مدل ۱

با وجود اعمال یک نویز بسیار کوچک از scale نیم، شبکه به طرز عجیبی از این نویز تاثیر می‌پذیرد و نتایج بسیار بدی را حتی در داده‌های آموزشی ارائه می‌کند. به منظور مشاهده‌ی این تاثیر در این بخش از همان مدلی استفاده شده است که در حالت بدون نویز تابع، نتایج مناسبی ارائه می‌داد. این مدل ۲ لایه با ورودی ۱ بعدی، ۱۰ نورون در لایه‌ی اول و ۱۵ نوورن در لایه‌ی دوم و خروجی یک بعدی است و با همان توابع فعال‌ساز و خطا و همان تعداد چرخه می‌باشد.

ارزیابی مدل ۱

ارزیابی این مدل با تابع MSE یک loss برابر 0.6 نتیجه می‌دهد که عملکرد مناسبی ندارد. (تصویر ؟؟)

تعریف مدل ۲ ـ افزایش حجم شبکه

به منظور افزایش دقت شبکه برای داده‌های دارای نویز، حجم مدل افزایش داده شد. مدل جدید دارای دو لایه و ۱۰۰ نورون در هر لایه می‌باشد. سایر پارامترهای مدل و پارامتر های فاز آموزش مدل بدون تغییر باقی می‌مانند.
در اثر افزایش حجم شبکه، مدل به کارایی مناسبی می‌رسد و تطبیق خوبی با داده‌ها پیدا می‌کند.

ارزیابی مدل ۲

ارزیابی این مدل با تابع MSE یک loss برابر 0.00098 نتیجه می‌دهد که نسبت به مدل اولیه عملکرد بهتری دارد. نتایج این مدل را در بخش نتایج خواهید دید.

۲.۲.۳ تابع دوم ـ نویز متوسط

تعریف مدل ۱

مدل دومی که برای تابع با نویز نزدیک به صفر ساخته شد، اگرچه در نویز متوسط کارایی کمتری نشان می دهد اما همچنان مقاومت خوبی دارد و به نتایج قابل قبولی منجر می‌شود. بنابراین در این قسمت، همچنان مدلی با دولایه و ۱۰۰ نورون در هر لایه معرفی می‌شود. سایر پارامترهای مدل و آموزش مدل بدون تغییر باقی می‌مانند.

ارزیابی مدل ۱

ارزیابی این مدل با تابع MSE یک loss برابر 0.00084 نتیجه می‌دهد که عملکرد خوبی دارد. نتایج این مدل را در بخش نتایج خواهید دید.

۳.۲.۳ تابع سوم ـ نویز بسیار بزرگ

تعریف مدل ۱

این بار با ثابت نگه داشتن مدل قسمت قبل نویز را باز هم افزایش می‌دهیم. پارامترهای مدل و آموزش مدل نیز برای نمایش این تفاوت بدون تغییر باقی مانده‌اند.
در این حالت مدل مجددا دچار افت کارایی می‌شود. شکل نویزها به گونه ای است که در اطراف صفر، به سمت اعداد منفی تجمع یافته‌اند. مدل دقیقا در این محل، به سمت این داده‌های نویز دار متمایل می‌شود و از شکل تابع اصلی دور می‌شود.

ارزیابی مدل ۱

ارزیابی این مدل با تابع MSE یک loss برابر 0.0058 نتیجه می‌دهد که عملکرد مناسبی ندارد. (تصویر ؟؟)

تعریف مدل ۲ ـ افزایش تعداد چرخه

در این بخش سعی می‌شود با ثابت نگه‌داشتن مدل اول، با افزایش تعداد چرخه ها، فرصت بیشتری به مدل برای بهبود کارایی‌اش داده شود. به این منظور تعداد چرخه‌ها به ۱۰۰۰ افزایش می‌یابد. هرچند انطباق مدل در این حالت همچنان ایده‌آل نیست اما باعث بهبود دقت مدل می‌شود.

ارزیابی مدل ۲

ارزیابی این مدل با تابع MSE یک loss برابر 0.0051 نتیجه می‌دهد که نسبت به حالت‌های دیگر عملکرد بهتری دارد. نتایج مصور این مدل را در بخش نتایج مشاهده خواهید کرد.

تعریف مدل ۳ ـ افزایش حجم شبکه

در این حالت با ثابت نگه داشتن سایر پارامترها، حجم شبکه را به تعداد ۲۰۰ نورون در هر لایه می‌رسانیم. برخلاف تاثیری که در تابع اول این بخش مشاهده شد، افزایش حجم شبکه در اینجا تاثیر مثبتی نداشته و انطباق تابع را کاهش می دهد.

ارزیابی مدل ۳

ارزیابی این مدل با تابع MSE یک loss برابر 0.0051 نتیجه می‌دهد که عملکرد مناسبی ندارد. (تصویر **؟؟**)

۳.۳ نتایج

وجود نویز در داده‌ها بر عملکرد شبکه تاثیر می‌گذارد. شبکه ممکن است سعی کند که تمام این نقاط را پوشش دهد و به عبارتی overfit کند. این حالت باعث دقت پایین در داده‌های تست خواهد بود. از طرفی ممکن است این نویز ها به گونه‌ای باشند که از overfitting جلوگیری کنند (اتفاقی که در داده‌های ما نیوفتاد اما گاه برای از بین بردن overfitting به داده‌ها نویز اضافه می‌کنند. مثل لایه‌ی Drop در keras). همچنین وجود نویز باعث وجود تغییرات ناگهانی در خروجی تابع می‌شود. این مسئله باعث می‌شود که شبکه به جزئیات بیشتری برای یادگیری داده‌ها نیاز داشته باشد چراکه با کوچکترین تغییرات در این بخش‌های تابع، خروجی تغییر می‌کند. به این ترتیب گاهی افزایش اندازه‌ی شبکه، به ایجاد یک مدل بهتر برای داده‌های دارای نویز منجر می‌شود. نکته‌ی دیگر در رابطه با تعداد چرخه‌های آموزشی است. گاه افزایش تعداد چرخه‌های آموزشی می‌تواند به یادگیری بهتر و کاهش خطای میانگین منجر شود و تابع را از اولین پاسخ در دسترس که تمایل به سمت داده‌های دارای نویز است، باز می‌دارد.

۱.۳.۳ تابع اول

بهترین نتایج به دست آمده برای این تابع ناشی از مدل ۲ (معرفی شده در بخش مدل‌ها) است.
(تصویر **؟؟**)

۲.۳.۳ تابع دوم

بهترین نتایج به دست آمده برای این تابع ناشی از مدل ۱ (معرفی شده در بخش مدل‌ها) است.
(تصویر ؟؟)

۳.۳.۳ تابع سوم

بهترین نتایج به دست آمده برای این تابع ناشی از مدل ۲ (معرفی شده در بخش مدل‌ها) است.
(تصویر ؟؟)

فصل ۴

بخش سوم

۱.۴ داده‌ها

در این بخش توابع با ابعاد بالاتر توسط مدل یادگیری می‌شوند. به این منظور توابعی با ابعادی بالاتر اما به نحوی که قابل نمایش گرافیکی باشند ایجاد شده‌اند. (تصویر **؟؟**) این توابع به شرح زیر تعریف می‌شوند.

۱. $f(x,y) = sin(x)sin(y)$

۲. $\begin{cases} f(x,y)_1 = sin(x)sin(y) \\ f(x,y)_2 = sin(\sqrt{x^2+y^2}) \end{cases}$

این توابع به صورت یک black box از طریق ایجاد مجموعه نقاط آموزشی و آزمایشی به شبکه‌ی عصبی عرضه خواهند شد. به این ترتیب که یک بازه‌ی دلخواه از ورودی در نظر گرفته شده، خروجی‌های صحیح (هرچند دارای نویز) مربوط به آن از ضابطه‌ی تابع استخراج شده و مجموعه‌ی این نقاط، پایگاه‌داده‌ی مسئله را تشکیل می‌دهند.
این داده‌ها سپس برای عملکرد بهتر شبکه scale شده‌اند. مشابه بخش‌های قبل، به منظور مقیاس‌بندی داده‌ها از MinMaxScaler استفاده شده‌است.
این داده‌ها سپس با نسبت ۸۵ به ۱۵ به مجموعه‌های آموزشی و آزمایشی تقسیم می‌شوند. مجموعه‌های آزمایشی برای فاز آموزش و مجموعه‌های آزمایشی برای فاز آزمایشی شبکه‌ی عصبی مورد استفاده خواهند بود.

اقدام دیگری که در این بخش برای آماده‌سازی توابع باید انجام شود، ایجاد دوتایی‌های ورودی از x و y است. چرا که مدل ورودی دو بعدی می‌پذیرد.
به این منظور ابتدا با استفاده از تابع meshgrid تمام ترکیب‌های ممکن از x و y های بازه‌ی انتخابی ساخته می‌شود، سپس این زوج داده‌ها reshape می شوند تا نهایتا به صورت یک آرایه از زوج مرتب‌ها دربیایند.

۲.۴ مدل

۱.۲.۴ تابع اول - ورودی دوبعدی خروجی یک بعدی

تعریف مدل

یک مدل مناسب که در این بخش با آزمون و خطا و با استفاده از تجربه‌ی بخش‌های قبل انتخاب شده‌است، یک شبکه با دو لایه‌ی پنهان ۱۰۰ نورونی است. همچنین به دلیل ارائه شده در دو بخش قبل، تابع فعال‌ساز relu یک انتخاب مناسب برای این توابع نیز هست و تابع ارزیابی عملکرد شبکه نیز همچنان MSE است که معیار مناسبی برای ارزیابی تخمین توابع است. این تابع با ۱۰۰ چرخه‌ی یادگیری به دقت مناسبی از مرتبه‌ی ده هزارم می‌رسد.

ارزیابی مدل

ارزیابی این مدل با تابع MSE یک loss برابر $1.4e-4$ نتیجه می‌دهد که عملکرد مناسبی دارد. در قسمت نتایج، این مدل به صورت مصور ارزیابی شده است.

۲.۲.۴ تابع دوم - ورودی دوبعدی خروجی دو بعدی

تعریف مدل

در این قسمت به علت عدم تغییر ماهیت تابع و تنها افزایش تعداد خروجی های آن، از مدل مشابهی استفاده می‌کنیم. تنها تفاوت اصلی در ابعاد خروجی شبکه است که از یک به دو افزایش می‌یابد. همچنین در ورودی-خروجی دادن به مدل، باید به این نکته توجه شود که در اینجا علاوه بر ورودی، خروجی نیز باید به روش مشابهی به صورت آرایه‌ای از زوج مرتب‌ها درآید.
سایز پارامترهای مدل و آموزش آن بدون تغییر باقی می‌مانند.

ارزیابی مدل

ارزیابی این مدل با تابع MSE یک loss برابر $5.5e-4$ نتیجه می‌دهد که عملکرد مناسبی دارد. در قسمت نتایج، این مدل به صورت مصور ارزیابی شده است.

۳.۴ نتایج

۱.۳.۴ تابع اول

بهترین نتایج به دست آمده برای این تابع ناشی از مدل معرفی شده در بخش مدل‌ها است.
(تصویر ؟؟)

۲.۳.۴ تابع دوم

بهترین نتایج به دست آمده برای این تابع ناشی از مدل معرفی شده در بخش مدل‌ها است.
(تصویر ؟؟)

فصل ۵

بخش چهارم

۱.۵ داده‌ها

در این بخش خواسته شده که یک تابع نامنظم به صورت دستی رسم شود . (تصویر **؟؟**)

سپس لازم است که یک‌سری نقاط روی این تابع تخمین زده و به عنوان پایگاه‌داده در نظر گرفته‌شود. این کار به کمک یک سرویس آنلاین رقومی‌سازی (digitizer) انجام شده است. (تصویر **؟؟**) این سرویس، تصویر تابع رسم شده را ورودی گرفته و تعداد دلخواهی نقطه روی منحنی قرمز را به فرمت یک پایگاه‌داده‌ی csv خروجی می‌دهد. این پایگاه‌داده توسط کتابخانه‌ی pandas خوانده شده و سپس طبق روال بخش‌های قبلی، مقیاس‌بندی و به مجموعه‌های آموزشی و آزمایشی به نسبت ۸۵ به ۱۵ تقسیم شده است.

۲.۵ مدل

تعریف مدل

به علت شکل دارای پیچیدگی و دامنه‌ی وسیع تابع ایجادشده، مدل در نقاط ابتدایی و انتهایی دامنه، به سختی fit می‌شود. این مسئله باعث شد که شبکه‌ی بزرگی از حجم ۲۰۰۰ نورون در لایه‌ی اول و ۳۰۰۰ نورون در لایه‌ی دوم نیاز باشد. درواقع هرچه تابع پیچیده‌تر و نامنظم‌تر باشد، فاقد الگوی مشخص خواهد بود و یادگیری آن در ابعاد کوچک دشوارتر و همراه با ازدست‌رفت اطلاعات خواهد بود. به عبارتی مدل نیازمند شبکه‌ی بزرگی است تا بتواند بخش‌های مختلف و الگوهای مختلفی را که در تابع وجود دارد بسازد و یادبگیرد.

همچنین به علت حجم بالای شبکه، تعداد چرخه‌ی به نسبت بزرگی از مرتبه‌ی ۵۰۰ چرخه نیاز است تا تمام وزن‌ها و bias ها بهینه‌سازی شوند.

همانطور که در بخش‌های پیشین گفته‌شد، تابع فعال‌ساز و تابع خطای مناسب برای تخمین توابع، به ترتیب relu و MSE می‌باشد که در این بخش نیز از آن‌ها استفده شده‌است.

ارزیابی مدل

ارزیابی این مدل با تابع MSE یک loss برابر 0.0019 نتیجه می‌دهد که عملکرد مناسبی دارد. در قسمت نتایج، این مدل به صورت مصور ارزیابی شده است.

۳.۵ نتایج

برای این تابع، علیرغم حجم بزرگ مدل و تعداد چرخه‌ی بالا، تابع همچنان در دو سر دامنه‌ی خود برای fit شدن با مشکل مواجه است که به نظر می‌رسد به علت عدم توزیع مناسب نقاط آموزشی در آن نقاط باشد.
همچنین وجود نقاط تغییرات ناگهانی باعث می‌شود که به ازای ورودی‌های بسیار مشابه، خروجی‌های متفاوتی داشته‌باشیم. این یعنی مدل نیازمند یادگیری جزئیات بیشتری است و به حجم بزرگ‌تری نیاز دارد. به این ترتیب حجم مورد نیاز شبکه تابعی از میزان پیچیدگی تابع رسم شده (یا به عبارتی خشم رسّام :)) است.

بهترین نتایج به دست آمده برای این تابع ناشی از مدل معرفی شده در بخش مدل‌ها در تصویر **؟؟** آمده‌است.

فصل ۶

بخش پنجم

۱.۶ داده‌ها

پایگاه‌داده‌ی انتخابی برای دسته‌بندی در این بخش، پایگاه‌داده‌ی ارقام دست‌نویس MNIST است. (تصویر **؟؟**) این پایگاه‌داده یکی از پایگاه داده‌های مثالی keras است و از طریق keras.datasets.mnist در دسترس است. با استفاده از تابع load_data این کتابخانه، این پایگاه‌داده به صورت تقسیم‌شده به مجموعه‌های آموزشی و آزمایشی به دست می‌آید.
این پایگاه‌داده شامل ۶۰۰۰۰ تصویر ۲۸*۲۸ پیکسلی از ارقام دست‌نویس انگلیسی در مجموعه‌ی آموزشی خود است. این ارقام (برچسب داده‌ها) از ۰ تا ۹ را شامل می‌شوند.
از جمله‌ی پیش‌پردازش‌هایی که لازم است بر روی این داده‌ها انجام شود، مقیاس‌بندی و reshaping است. به منظور scaling می‌دانیم تصاویر شامل پیکسل‌هایی با مقادیر ۰ تا ۲۵۵ هستند. بنابراین با تقسیم این مقادیر به ۲۵۵، می‌توانیم مقدار این پیکسل‌ها را به بازه‌ی ۰ و ۱ مقیاس کنیم.
به منظور reshaping هم از آنجا که مدل تنها توانایی پذیرش داده‌های یک‌بعدی n تایی را دارد، تصاویر ۲۸*۲۸ را به صورت خطی به آرایه‌های ۷۸۴ تایی تبدیل می‌کنیم. به این ترتیب مجموعه‌ی آموزشی به ۶۰۰۰۰ داده‌ی ۷۸۴ تایی تبدیل می‌شود و می‌توان آن را به مدل ورودی داد.
در مورد برچسب‌ها نیاز به مقیاس‌بندی نیست و در بخش مدل در این باره بیشتر توضیح داده شده است.

۲.۶ مدل

خروجی مدل

اما در مورد برچسب‌ها، یک دیدگاه آن است که یک خروجی عددی از ۰ تا ۹ داریم. اما این دیدگاه مشکلاتی دارد. از جمله خروجی غیر صحیح مدل و مناسب نبودن توابع loss برای ارزیابی تصمیم مدل.
یک دیدگاه بهتر که با ماهیت دسته‌بندی مسئله هم‌خوانی بهتری دارد، استفاده از کدگذاری one-hot است. به این ترتیب که مدل به تعداد کلاس های مختلف خروجی خواهد داشت. به عنوان مثال در مورد مسئله‌ی دسته‌بندی اعداد، مدل دارای ۱۰ خروجی خواهد بود. هر خروجی مقداری می‌گیرد و خروجی با بیشترین مقدار، دسته‌ی داده را تعیین می‌کند. این خروجی‌ها را می‌توان با یک لایه‌ی اضافی در شبکه با تابع فعال‌ساز softmax به یک توزیع احتمالاتی تبدیل کرد. به این ترتیب هر خروجی بیانگر احتمال

تعلق به آن دسته‌بندی خاص است و مجموع این احتمالات برابر یک خواهد بود.

تابع loss function

برای فرمت خاص در نظر گرفته‌شده برای خروجی (one-hot) کتابخانه‌ی keras یک loss function مناسب ارائه می‌کند به نام sparce categorical cross entropy. این تابع یک برچسب ورودی میگیرد و یک خروجیِ در اینجا ۱۰ بعدی. سپس با توجه به شماره‌ی مربوط به دسته بندی، خطای دسته‌بندی را با رابطه‌ی ۱.۶ محاسبه می‌کند.

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i)] \tag{۱.۶}$$

به این ترتیب، به عنوان مثال یک تخمین با خروجی‌های $[0.1, 0.9]$ برای برچسب ۲ loss به نسبت کوچکی خواهد داشت زیرا احتمال زیادی را به آن دسته‌بندی نسبت داده‌است. از رابطه‌ی تابع هم واضح است که یک پیشبینی ۱ برای دسته‌ی صحیح، موجب خطای ۰ می‌شود.

تعریف مدل

با توجه به مطالب دو قسمت قبل و به کمک آزمون و خطا، مدلی که برای این بخش تعریف شده است، یک مدل دو لایه‌ی پنهان با ۱۶ نورون در هر لایه و یک خروجی ۱۰ بعدی است. توابع فعال‌ساز مورد استفاده relu هستند و تابع loss هم همانطور که پیش‌تر گفته شد از نوع sparce categorical cross entropy است.
این مدل تنها با ۲۰ چرخه‌ی یادگیری به دقتی بالا دست می‌یابد.

ارزیابی مدل

ارزیابی این مدل با تابع MSE یک loss برابر 0.15 و دقت بالای ۹۷ درصد نتیجه می‌دهد که عملکرد مناسبی دارد. در قسمت نتایج، یک نمونه پیش‌بینی این مدل آورده شده‌است.

۳.۶ نتایج

دقت مدل مشروح، در داده‌های آزمایشی برابر ۹۷ درصد به دست آمده است. قابلیت دسته‌بندی این مدل تحت یک اینترفیس ساده فراهم شده است و با دریافت یک عدد از داده‌های تست، پیش‌بینی خود را نمایش می‌دهد.
تصویر **؟؟** چند نمونه از این پیش‌بینی‌ها را نشان می‌دهد.

فصل ۷

بخش ششم

۱.۷ داده‌ها

در این بخش به بررسی کاربرد شبکه‌های عصبی پرسپترون چندلایه در رفع نویز از داده‌ها پرداخته می‌شود. برای آماده‌سازی داده‌ها، ابتدا داده‌ها خوانده، مقیاس‌بندی شده و به مجموعه‌های آموزشی و آزمایشی تقسیم شده‌اند. سپس بر روی این داده‌های تصویری، نویز ایجاد شده است. این نویز دارای توزیع نرمال بوده و توسط کتابخانه‌ی numpy.random در بازه‌ی ۰ و ۱ (بازه‌ی مقیاس‌شده‌ی ۰ تا ۲۵۵) ایجاد می‌شود و سپس با یک ضریب به نام noise_factor به داده‌ها اعمال می‌شود. این ضریب معیاری از مقدار نویز اعمال‌شده است.
نویز در سه مقدار کم، متوسط و بسیار زیاد به داده‌ها اعمال شده است و در بخش‌های مختلف کارایی مدل بررسی شده است. این سه مقدار به ترتیب دارای noise_factor 0.2، 0.45 و 0.7 هستند. (تصویر **؟؟**)

۲.۷ مدل

Autoencoders

در این بخش از ساختار Autoencoder ها استفاده شده که شرح آن در فصل اول در بخش ۳.۲.۱ آمد.

تعریف مدل

طبق نکات ذکر شده در رابطه با ساختار Autoencoder ها، مدل تعریف شده در این بخش یک ساختار کاهشی/افزایشی خواهد داشت. به ترتیبی که ورودی ۷۸۴ بعدی خواهد بود (به تعداد پیکسل‌های تصاویر)، لایه‌ی پنهان اول ۴۰۰ نورون، لایه‌ی پنهان دوم ۱۰۰ نورون و لایه‌ی سوم پنهان مجددا ۷۸۴ نورون خواهد داشت.
آنچه ذکر شد، ساختار اصلی شبکه‌ی رفع نویز مورد استفاده در این بخش بود. اما وجود دو لایه‌ی کمکی دیگر ضروری است. اولین لایه‌ای که بعد از لایه‌ی سوم می‌آید و از امکاناتی است که کتابخانه‌ی keras در اختیار می‌گذارد، لایه‌ی Reshape است که خروجی ۷۸۴ بعدی شبکه را مجددا به شکل ۲۸*۲۸ اولیه‌ی تصویر تغییر می‌دهد. به این ترتیب، شبکه یک تصویر reshape شده را دریافت و یک تصویر

در قالب اصلی اولیه که رفع نویز شده برمیگرداند.
دومین لایه‌ای که وجود آن ضروری است، یک لایه‌ی convolutional با یک تک نورون است. لایه‌ی convolutional درواقع به تعداد پارامتر اولش فیلتر بر روی داده‌های عموما از نوع تصویر (دو بعدی) تعریف می‌کند. ابعاد این فیلتر نیز به عنوان پارامتر دوم لایه داده می‌شود که در اینجا از نوع متداول و کارامد آن یعنی فیلتر ۳*۳ استفاده شده است. پارامتر سوم (padding) نشان دهنده‌ی عدم تغییر ابعاد تصویر است که طبعا مطلوب ماست.
درصورت عدم اعمال این لایه، مدل علیرغم رفع نویز ساختگی، نوی دیگری از نویز جزئی را بر روی تصاویر به جا می‌گذاشت. این نویز ثانویه به صورت پیکسل‌های کاملا تیره ظاهر می‌شد (تصویر **؟؟**). به منظور رفع این نویز، یک لایه‌ی convolutional تک فیلتر به شبکه اضافه شد تا با یادگیری از داده‌های اصلی، فیلتر مناسبی بر روی تصاویر خروجی اعمال و این نویز را از بین ببرد.
نکته‌ی قابل توجه آن است که روش اصلی و شناخته‌شده‌ی رفع نویز از تصاویر توسط شبکه‌های عصبی همین استفاده از لایه‌های Conv۲D و Conv۲DTranspose برای بخش decoder است. اما این نوع شبکه‌ها وارد مباحث CNN می‌شوند و در اینجا ترجیح بر آن شد که از روش‌های معمول و با استفاده از شبکه‌های عصبی dense این رفع نویز انجام بشود هرچند که استفاده از Conv۲D ها می‌توانست کارآمد تر باشد.

در نهایت این شبکه، با ساختاری که گفته‌شد و با تابع فعال‌ساز relu و تابع خطای MSE طراحی شد و با تنها ۵۰ چرخه‌ی یادگیری به دقت مطلوبی دست یافت.
این شبکه در هر سه میزان نویز کارآمد بوده و در هر سه بخش از این شبکه برای رفع نویز استفاده شده است.

ارزیابی مدل

همانطور که گفته شد، این مدل برای هر سه مرتبه نویز کارایی مناسبی از خود نشان می‌دهد. با این حال، کاهش کیفیت تصاویر در مرتبه‌های بالاتر نویز کاملا مشهود است و استفاده از مدل‌های با ابعاد متفاوت می‌تواند این کیفیت را بهبود بخشد.
اما در رابطه با تفاوت عملکرد مدل در داده‌های آموزشی و آزمایشی، باز هم این مدل عملکرد خوبی را در داده‌های تست مشابه داده‌های آموزشی نشان می دهد. با این حال در مورد برخی داده‌های آزمایشی شاهد خطا هستیم. به عنوان مثال دقت کنید به تصویر **؟؟** در بخش نتایج. در این تصویر مشخص است که مدل در رفع نویز از عدد ۴ دچار خطا شده و ظاهر آن را تغییر داده و به عدد ۹ متمایل کرده‌است.
این تفاوت عملکرد در داده‌های آموزشی و آزمایشی منطقی است و به علت تماما نادیده بودن داده‌های تست، ممکن است که الگوهای خاصی در کدگذاری از دست رفته باشند و در کد گشایی شاهد نتایجی مشابه مثال اخیر باشیم.
میزان کارایی این شبکه در هریک از مقادیر نویز به صورت مصور در قسمت نتایج آمده‌است.

۳.۷ نتایج

برای نویز کم ،(noise_factor=۰.۲) داده‌های اولیه، نویزدار و denoise شده در تصویر **؟؟** آمده است.
برای نویز متوسط ،(noise_factor=۰.۴۵) داده‌های اولیه، نویزدار و denoise شده در تصویر **؟؟** آمده است.

و برای نویز زیاد ،(noise_factor=۰.۷) داده‌های اولیه، نویزدار و denoise شده در تصویر **؟؟** آمده است.