تمرین اول: شبکه‌های Multi-Layer perceptron نسخه دوم*

توسط: مهدی مهدیانی

ایمیل: m.mahdiani@stu.usc.ac.ir , mahdimahdiani@ymail.com

استاد: آقای دکتر شهسوار حقیقی

دستیار آموزش: آقای مهندس صالح

۱ معرفی مجموعه‌داده[1]

MNIST مخفف Modified National Institute of Standards and Technology، یک دیتاست مشهور و پرکاربرد در حوزه یادگیری ماشین و بینایی کامپیوتر است. این مجموعه‌داده شامل تصاویر دست‌نویس ارقام ۰ تا ۹ انگلیسی است که توسط یان لوکان[2] جمع‌آوری شده و به طور گسترده برای آموزش و سنجش الگوریتم‌های مختلف به کار می‌رود. عکس‌ها در این مجموعه‌داده سیاه‌وسفید[3] هستند و دارای ابعاد ۲۸*۲۸ هستند. این مجموعه‌داده از دو بخش آموزش و آزمون تشکیل شده است. در مجموعه آموزش ۶۰۰۰۰ تصویر و در مجموعه‌آزمون ۱۰۰۰۰ تصویر وجود دارد که هر تصویر لیبل (برچسب) ۰ تا ۹ را دارد.

۲ هدف

در این تمرین قصد داریم با این مجموعه‌داده آشنا شویم و سپس با شبکه‌های عصبی MLP دسته‌بندی این مجموعه‌داده را انجام دهیم.

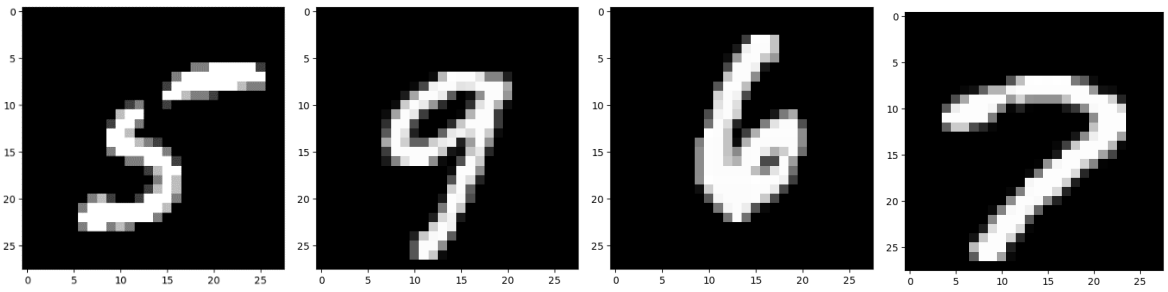
۳ پیش‌پردازش

۳-۱ مجموعه‌داده

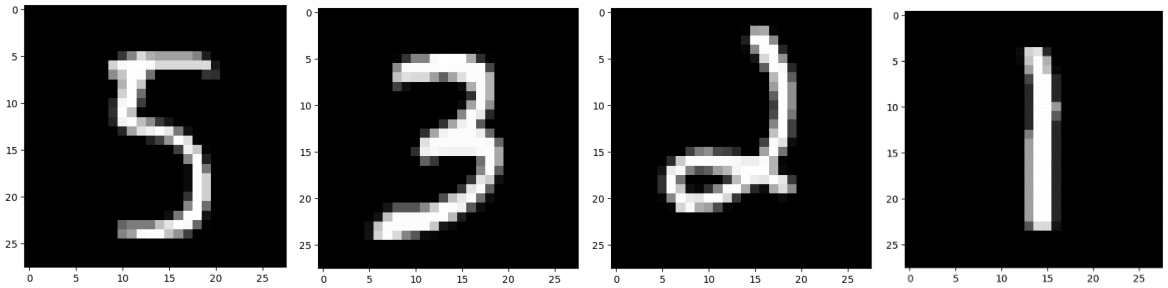
*در انتهای این گزارش فعالیت هایی جهت بهبود دقت آورده شده است

[1] Dataset
[2] Yann leCun
[3] Grayscale

برای واردکردن مجموعه‌داده به نوت‌بوک یا برنامهٔ پایتونی کافی است MNIST را از کتابخانه کراس فراخوانی کنیم. سپس با فراخوانی تابع load_data داده‌های این مجموعه را فراخوانی می‌کنیم. باید توجه شود که این تابع یک tuple از جنس آرایه‌های Numpy را برمی‌گرداند. سپس با استفاده از تابع shape ابعاد هر یک از آموزش و آزمون را به دست می‌آوریم. مجموعه‌داده‌های آموزش این مجموعه دارای ۶۰۰۰۰ داده عکس با ابعاد ۲۸*۲۸ است (60000,28,28) . مجموعه‌داده‌های آزمون این مجموعه دارای ۱۰۰۰۰ داده عکس با ابعاد ۲۸*۲۸ است (10000,28,28). چندین نمونه از این داده‌ها را در زیر مشاهده می‌کنیم.



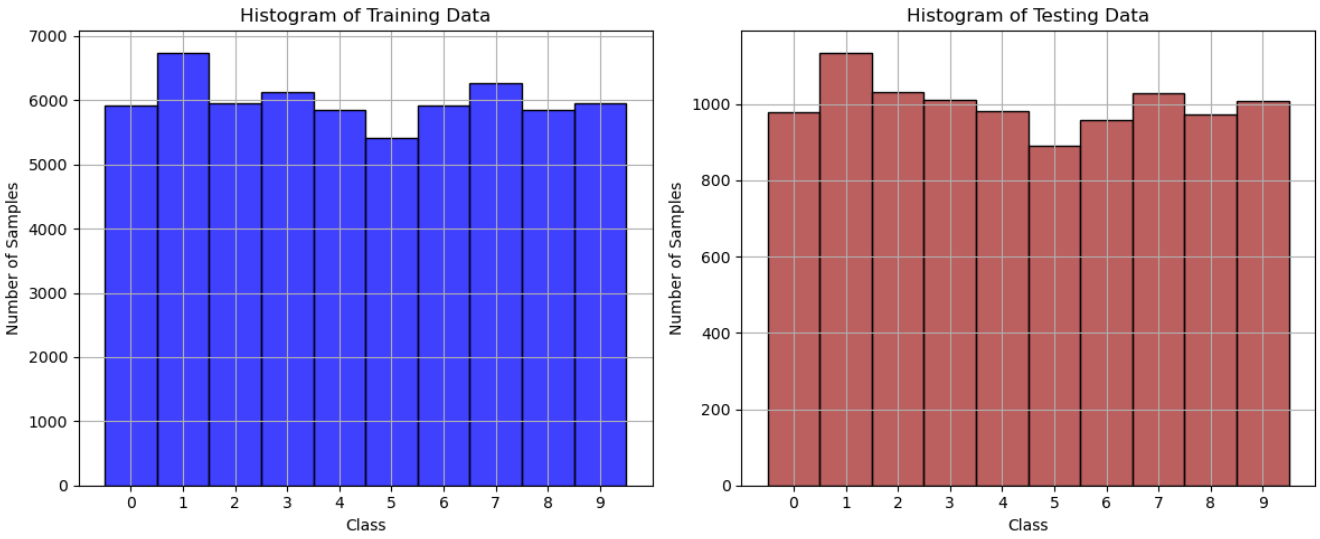
شکل1 نمونه های داده آموزش



شکل 2 نمونه های داده ی آزمون

همچنین لیبل هر نمونه به‌صورت عدد صحیح (۰،۱،۲،۳،۴،۵،۶،۷،۸،۹) است.

۲-۳ هیستوگرام داده‌های آموزش و آزمون



شکل 3 هیستوگرام داده های آموزش و آزمون

با بررسی هیستوگرام داده‌های آموزش متوجه می‌شویم که بیشتر کلاس‌های این مجموعه حدوداً بین ۶۰۰۰ تا ۷۰۰۰ نمونه دارند و این به معنای این است که این دیتاست نسبتاً متعادل است و برای هر رقم تقریباً تعداد تصاویر مشابهی وجود دارد. در وهله بعدی تعداد کمی از کلاس‌های تعداد کم‌تر از ۶۰۰۰ تصویر دارند که به معنای این است که از آن کلاس‌ها نمونه‌های کم‌تری در دیتاست وجود دارد. به‌علاوه هیچ کلاسی بیش از ۷۰۰۰ تصویر ندارد که در نتیجه هیچ کلاسی بیش از حد نشان داده نشده است[۴]. در نتیجه مجموعه‌دادهٔ آموزش ما تقریباً متعادل[۵] است و تعداد تصویر مشابهی برای هر عدد وجود دارد پس دیتاست برای تمرین تشخیص ارقام دست‌نویس با یادگیری ماشین و یادگیری عمیق مناسب است.
هیستوگرام داده‌های آزمونی نیز تعداد نمونه‌ها را در هر کلاس نشان می‌دهد. در این مورد، توزیع نمونه‌ها کمی نامتعادل‌تر است، به‌طوری‌که رقم 1 دارای 1000 نمونه و رقم 9 دارای 800 نمونه است. با این حال، این عدم تعادل قابل توجه نیست و انتظار می رود که الگوریتم های تشخیص دست خط آموزش دیده بر روی داده های آموزشی با این توزیع داده های آزمونی به خوبی کار کنند.

۳-۳ نرمال‌سازی

در این دیتاست عکس‌های سیاه‌سفید وجود دارند که ابعاد ۲۸*۲۸ پیکسل را داراست. هر پیکسل عددی بین ۰ تا ۲۵۵ را به خود نسبت داده است. در نتیجه برای نرمال‌سازی به روش min-max کافی است هر پیکسل را بر ۲۵۵ تقسیم کنیم تا اعداد هر پیکسل بین ۰ و ۱ قرار بگیرد.

۳-۴ کدگذاری one-hot

مجموعه‌دادهٔ ما چند کلاسه است در نتیجه هر نمونه متعلق به یکی از چندین کلاس ممکن خواهد بود. کدگذاری one hot یک بردار دودویی تولید می‌کند که برای هر نمونه، کلاس آن نمونه ۱ و مابقی ۰ می‌شود. برای مثال زمانی که لیبل ما ۴ است بعد از کدگذاری به آرایه زیر خواهیم رسید.

array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])

که درایه مربوط به کلاس ۴ برابر یک است و سایرین صفر هستند.

۴- مدل

۴-۱ ساخت مدل

مدل sequential یک رابط ساده برای پیاده‌سازی شبکه‌های عصبی با طبقه‌بندی لایه به لایه در تنسور فلو است. این مدل معمولاً برای ساختن مدل‌هایی استفاده می‌شود که شامل لایه‌های متوالی هستند، مانند مدل‌هایی که از ساختارهای ساده مثل CNN یا MLP استفاده می‌کنند. با استفاده از Sequential می‌توانیم لایه‌های مختلفی را به ترتیب به مدل اضافه کنیم. این لایه‌ها می‌توانند شامل لایه‌های کاملاً متصل، لایه‌های اساسی، لایه‌های تکرار باشند. ابتدا یک شی از کلاس Sequential می‌سازیم سپس با استفاده از متد add لایه‌های مختلف را به مدل اضافه می‌کنیم. سپس با استفاده از متد compile و تعیین توابع هزینه، بهینه‌ساز و معیارهای ارزیابی مدل را برای آموزش آماده می‌کنیم.

[4] Over re-presented
[5] balance

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 784)	0
dense_5 (Dense)	(None, 120)	94,200
dense_6 (Dense)	(None, 84)	10,164
dense_7 (Dense)	(None, 10)	850

شکل 4 مدل نمونه MLP

باتوجه‌به صورت سؤال، این مدل دارای یک ورودی ۲۸*۲۸ است که بعد از تغییر شکل دوبعدی به یک‌بعدی به ۷۸۴ پیکسل تبدیل می‌شود. سپس یک‌لایه پنهان[6] با ۱۲۰ نورون با تابع فعال‌ساز ReLu است. در لایه پنهان بعدی ۸۴ نورون وجود دارد که تابع فعال‌ساز این لایه هم ReLu است. تمامی نورون‌ها به هم متصل هستند[7] و در نهایت یک‌لایه خروجی با ۱۰ نورون با تابع فعال‌ساز soft max قرار دادیم تا خروجی‌ها را مشخص کنیم، در ادامه به بررسی این لایه‌ها می‌پردازیم. در ستون سوم شکل ۴ تعداد پارامترهای قابل و غیرقابل‌آموزش را مشاهده می‌کنیم. پارامترهای قابل‌آموزش مثل وزن‌ها و بایاس هستند که در طول آموزش به‌روزرسانی می‌شوند و پارامترهای غیرقابل‌آموزش در طول آموزش به‌روزرسانی نمی‌شوند.

لایه اول – لایه ورودی:

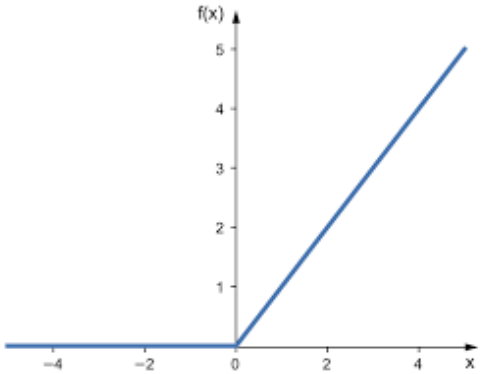
این لایه عکس‌های ورودی را از فرمت دوبعدی (۲۸*۲۸ پیکسل) به فرمت یک‌بعدی (۷۸۴ پیکسل) قبل از ارسال به لایه بعدی تبدیل می‌کند.

لایه دوم – لایه پنهان اول:

این لایه دارای ۱۲۰ نورون با تابع فعال‌ساز ReLU است.

لایه سوم – لایه پنهان دوم:

این لایه دارای ۸۴ نورون با تابع فعال‌ساز ReLU است.



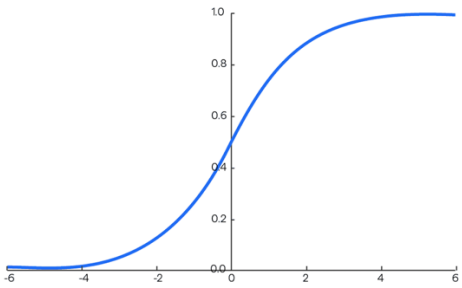
شکل 5 نمودار تابع فعال‌ساز ReLU

[6] Hidden Layer
[7] Fully Connected

لایه چهارم – لایه خروجی:

این لایه باتوجه‌به اینکه ۱۰ کلاس برای طبقه‌بندی وجود دارد دارای ۱۰ نورون است. باتوجه‌به این که این مسئله یک‌طبقه بندی چند کلاسه است، از تابع فعال‌ساز softmax استفاده می‌شود. این تابع یک بردار ورودی را به یک توزیع احتمال تبدیل می‌کند، به‌طوری‌که هر عنصر در بردار خروجی نشان‌دهنده احتمال تعلق آن ورودی به یک کلاس خاص است. در ادامه فرمول و نمودار این تابع را مشاهده می‌کنیم.

$$(۱) \quad softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$



شکل 6 نمودار تابع Softmax

ویژگی‌های تابع Softmax :

- مجموع تمام احتمالات خارجی را برابر یک می‌کند
- الگوهای پیچیده‌تر را قابل‌یادگیری می‌کند
- به طور گسترده در تشخیص چهره، تشخیص اشیا و پردازش زبان طبیعی کاربرد دارد

مزایای استفاده از Softmax :

- استفاده ساده و قابل‌فهم
- بسیار مؤثر برای طبقه‌بندی چند کلاسه
- پیاده‌سازی بسیار گسترده در کتابخانه‌های یادگیری ماشین و یادگیری عمیق

معایب استفاده از Softmax :

- حساس به ورودی‌های بسیار بزرگ
- احتمال اشباع شدن دارد

۲-۴ کامپایل

با استفاده از متد compile ، مدل شبکه عصبی را با پارامترهای مختلف از جمله بهینه‌ساز، تابع هزینه و معیارهای ارزیابی تنظیم می‌شود. در ادامه به بررسی این پارامترها می‌پردازیم.

۴-۲-۱ بهینه‌ساز

وظیفه بهینه‌سازها این است که وزن‌های شبکه را در هر مرحله آموزش به‌روزرسانی کنند. انواع مختلفی از بهینه‌سازها را می‌توان استفاده کرد که در زیر به آن‌ها می‌پردازیم.

- SGD (Stochastic Gradient Descent)
- RMSprop (Root Mean Square Propagation)
- Adam (Adaptive Moment Estimation)
- Adagrad (Adaptive Gradient Algorithm)
- Adadelta

الف) SGD :

یکی از ساده‌ترین بهینه‌سازهاست که در هر مرتبه با گرادیان هزینه نسبت به وزن‌ها به‌روزرسانی را انجام می‌دهد. معمولاً برای مسائل ساده و کوچک با داده‌های مقیاس کوچک مناسب است. این بهینه‌ساز به دلیل سادگی و سرعت اجرای بالا خود مورداستفاده قرار می‌گیرد. اما برای مسائل با سطح پیچیدگی بالا ممکن است بهینه‌سازهای دیگر بهتر عمل کنند.

ب) RMSprop :

RMSprop اصلاح شده‌ای از Gradient Descent است که برای مقابله با نوسانات گرادیان در مسائل غیر محدود استفاده می‌شود. این بهینه‌ساز با نگاه‌داشتن به میانگین مربعات گرادیان‌ها، نرخ یادگیری را تطبیق می‌دهد. معمولاً برای مسائل با تغییرات نوسانی در سرعت یادگیری و یا مقیاس داده‌های متفاوت مناسب است. این بهینه‌ساز برای مسائل با داده‌های نادر و غیر متوازن نیز کارآمد است.

ج) Adam :

Adam یک بهینه‌ساز ترکیبی است که ترکیبی از دو روش Momentum و RMSprop است. این بهینه‌ساز نرخ یادگیری را برای هر وزن به طور جداگانه تطبیق می‌دهد و برای بسیاری از مسائل به‌عنوان یک بهینه‌ساز پیشنهادی استفاده می‌شود. معمولاً به‌عنوان یک بهینه‌ساز عمومی برای بسیاری از مسائل مورداستفاده قرار می‌گیرد. این بهینه‌ساز پایدار است و معمولاً در بسیاری از مسائل خوب عمل می‌کند، از جمله مسائل با داده‌های بزرگ و پیچیده.
هر سه بهینه‌ساز برای مسئله فعلی می‌تواند مناسب باشد، باید آزمایش صورت گیرد و سپس بهترین را انتخاب کنیم.

۴-۲-۲ تابع هزینه[8]

توابع هزینه (cost functions) در شبکه‌های عصبی برای اندازه‌گیری تفاوت بین مقادیر پیش‌بینی شده توسط مدل و مقادیر واقعی (برچسب‌ها یا داده‌های مشاهده شده) استفاده می‌شوند. هر تابع هزینه معمولاً بر اساس نوع مسئله مورداستفاده (به‌عنوان‌مثال، دسته‌بندی یا رگرسیون) و ویژگی‌های داده‌ها انتخاب می‌شود.

- Cross-Entropy Loss (Categorical Cross-Entropy)
- Binary Cross-Entropy Loss
- Mean Squared Error (MSE)
- Huber Loss
- و...

[8] Cost Function

Cross-Entropy Loss (Categorical Cross-Entropy):

در مسائل دسته‌بندی چند کلاسه استفاده می‌شود، به‌ویژه وقتی که برچسب‌ها به‌صورت one-hot encoded باشند. این تابع هزینه میزان اختلاف بین توزیع احتمال پیش‌بینی شده و توزیع احتمال واقعی را اندازه‌گیری می‌کند. درصورتی‌که برچسب‌ها one-hot کدگذاری نشده باشند و به‌صورت عدد صحیح باشند از Sparse cross entropy استفاده می‌شود.

Binary Cross-Entropy Loss:

در مسائل دسته‌بندی دو کلاسه (باینری) استفاده می‌شود. مانند Cross-Entropy Loss ، این تابع هزینه نیز میزان اختلاف بین توزیع احتمال پیش‌بینی شده و توزیع احتمال واقعی را اندازه‌گیری می‌کند، اما برای مسائل دو کلاسه مناسب است.

Mean Squared Error (MSE):

در مسائل رگرسیون استفاده می‌شود. این تابع هزینه میزان میانگین مربعات اختلاف بین پیش‌بینی مدل و مقادیر واقعی را اندازه‌گیری می‌کند.

Huber Loss:

نوع دیگری از تابع هزینه برای مسائل رگرسیون است که در مقابل داده‌های نویزی مقاوم‌تر است. این تابع هزینه تفاوت بین پیش‌بینی مدل و مقادیر واقعی را با درنظرگرفتن یک مقدار ثابت (delta) محدود می‌کند.

۳-۲-۴ معیار اندازه‌گیری (متریک) [۹] :

متریک‌ها ابزاری برای اندازه‌گیری دقت مدل هستند. این معیارها معمولاً به‌صورت عددی ارائه می‌شوند و نشان می‌دهند که مدل چقدر در توانایی پیش‌بینی یا توصیف داده‌ها موفق بوده است . در ادامه تعدادی از متریک‌ها را معرفی کرده و مختصری توضیح می‌دهیم.

دقت (Accuracy):

این متریک معمولی‌ترین متریک در مسائل دسته‌بندی است. دقت مشخص می‌کند که چند درصد از نمونه‌ها به‌درستی دسته‌بندی شده‌اند. برای مسائل با تعداد کلاس‌های متوازن، دقت می‌تواند یک معیار خوب برای ارزیابی عملکرد مدل باشد، اما در مسائل با تعداد کلاس‌های نامتوازن، دقت ممکن است توصیف کاملی از عملکرد مدل نباشد.

دقت (Precision):

Precision نسبت تعداد نمونه‌های درست دسته‌بندی شده به تعداد کل نمونه‌هایی است که مدل آن‌ها را به‌عنوان مثبت دسته‌بندی کرده است. این معیار مفید است زمانی که تعداد نمونه‌های منفی (به‌عنوان‌مثال، داده‌های غیر اسپم در یک مسئله تشخیص اسپم ایمیل) نسبت به نمونه‌های مثبت زیاد است.

Recall:

Recall نسبت تعداد نمونه‌های درست دسته‌بندی شده به تعداد کل نمونه‌های مثبت است. این معیار نشان می‌دهد که مدل چه درصد از تمام نمونه‌های مثبت را به‌درستی شناسایی کرده است .این معیار در مسائلی مثل تشخیص بیماری‌هایی مثل سرطان بسیار مهم است.

[۹] Metric

F1-Score:

F1-Score میانگین هارمونیک دقت و بازیابی است. این معیار یک معیار جامع برای ارزیابی دقت و بازیابی مدل است. برای مسائلی که دقت و بازیابی هر دو اهمیت دارند، F1-Score می‌تواند یک معیار مناسب باشد.

Mean Absolute Error (MAE):

MAE میانگین مطلق اختلاف بین پیش‌بینی مدل و مقادیر واقعی است. این معیار برای مسائل رگرسیون استفاده می‌شود و میزان خطا را به‌صورت مطلق نشان می‌دهد.

Mean Squared Error (MSE):

MSE میانگین مربعات اختلاف بین پیش‌بینی مدل و مقادیر واقعی است. مانند MAE ، MSE نیز برای مسائل رگرسیون استفاده می‌شود و میزان خطا را نشان می‌دهد، اما اختلافات بزرگ‌تر به‌شدت وزن‌دهی می‌شوند.

برای دیتاست MNSIT که یک مسئله طبقه‌بندی چند کلاسه است، استفاده از معیارهای Accuracy ، Precision ، Recall و F1-Score مناسب است. معیار Accuracy به‌صورت کلی برای این دیتاست عملکرد خوبی دارد؛ اما معیارهای دیگر نیز می‌تواند اطلاعات مفیدی را ارائه کند.

۵- آموزش

این مرحله یکی از مهم‌ترین مراحل شبکه‌های عصبی است که با دستور fit صورت می‌گیرد. این دستور با تعیین تعداد ایپک، اندازه دسته‌ها و داده‌های ارزیابی آموزش را انجام می‌دهد. آموزش مدل عموماً از طریق الگوریتم‌های بهینه‌سازی و backpropagation انجام می‌شود. در مرحله آموزش، ابتدا داده‌های آموزش به مدل داده می‌شوند و سپس مقدار خروجی مورد انتظار برای هر داده (دسته از داده‌ها) نیز مشخص می‌شود. سپس مدل بر اساس داده‌ها و مقادیر خروجی مورد انتظار، پیش‌بینی می‌کند و خطای پیش‌بینی خود را با مقدار واقعی خطای مشاهده شده مقایسه می‌کند. سپس با استفاده از روش backpropagation و الگوریتم بهینه‌سازی مشخص شده در مرحله compile، گرادیان خطا نسبت به وزن‌ها محاسبه می‌شود. این گرادیان نشان‌دهنده تأثیر هر وزن در خطا است. سپس با استفاده از این گرادیان‌ها، وزن‌ها به سمت کاهش خطا به‌روزرسانی می‌شوند. این به‌روزرسانی‌ها معمولاً با استفاده از یک الگوریتم بهینه‌سازی مشخص شده مانند(Stochastic Gradient Descent (SGD ، Adam، RMSprop و غیره انجام می‌شود. عمل backpropagationاین فرایند را برعهده دارد که به طور خلاصه به محاسبه گرادیان خطا نسبت به هر وزن و بایاس در شبکه عصبی می‌پردازد. این گرادیان‌ها سپس برای به‌روزرسانی وزن‌ها استفاده می‌شوند تا خطای شبکه کاهش یابد و عملکرد مدل بهبود یابد. این فرایند تکرار می‌شود تا مدل به‌دقت مطلوبی در پیش‌بینی داده‌های آموزش برسد. در ادامه پارامترهای متد fit را بررسی می‌کنیم.

الف) تعداد ایپاک: تعداد دوره‌های آموزش که مدل در طول آن تمامی داده‌ها را مشاهده می‌کند.
ب) اندازه دسته‌ها: تعداد نمونه‌های هر دسته که برای آموزش مدل استفاده می‌شود، در واقع داده‌های ما را به چندین دسته تقسیم می‌کند. این کار باعث می‌شود آموزش مدل با سرعت بیشتری انجام پذیرد. البته باید دقت شود هنگام استفاده از GPU ، اندازه دسته‌ها باید از اندازه قابل‌پذیرش GPU کمتر یا مساوی باشد.
ج) داده‌های ارزیابی: این داده‌ها قسمتی از داده‌های آموزش ما هستند که مدل آن‌ها را کنار گذاشته و مشاهده نمی‌کند و در هر ایپاک برای بررسی عملکرد از آن‌ها استفاده می‌کند.

جهت بررسی عملکرد مدل و رسم نمودارهای مربوطه در هر ایپاک، مقادیر acc , loss , acc_val , loss_val را در یک شی ذخیره کرده تا بعداً از آن‌ها استفاده کنیم.
برای مثال اگر مجموعه‌داده ما ۱۲۰۰۰ نمونه داشته باشد و اندازه دسته‌ها را ۲۰۰ در نظر بگیریم تعداد تکرار در هر ایپاک با روش زیر محاسبه می‌شود.

ابتدا تعداد کل نمونه‌ها را به تعداد دسته‌ها تقسیم می‌کنیم تا تعداد تکرار را به دست آوریم: $\frac{۱۲۰۰۰}{۲۰۰} = ۶۰$

سپس باتوجه‌به اینکه تعداد تکرار برابر با تعداد به‌روزرسانی وزن‌ها است، پس کافی است عدد به‌دست‌آمده برای تعداد به‌روزرسانی وزن‌ها را در تعداد کل ایپاک‌ها ضرب کنیم: ۶۰ * ۵ = ۳۰۰
در حالت کلی فرمول زیر را می‌توان برای تعداد به‌روزرسانی وزن‌ها استفاده کرد.

$$\text{تعداد ایپاک ها} * \frac{\text{تعداد نمونه ها}}{\text{تعداد دسته ها}} \quad (۲)$$

۶ بررسی عملکرد مدل‌های مختلف

۶-۱ مدل شماره یک

با توجه به‌صورت سؤال یک مدل که دارای دولایه پنهان به ترتیب ۱۲۰ و ۸۴ نورون هست را می‌سازیم.

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 120)	94,200
dense_1 (Dense)	(None, 84)	10,164
dense_2 (Dense)	(None, 10)	850

شکل 7 مدل اول

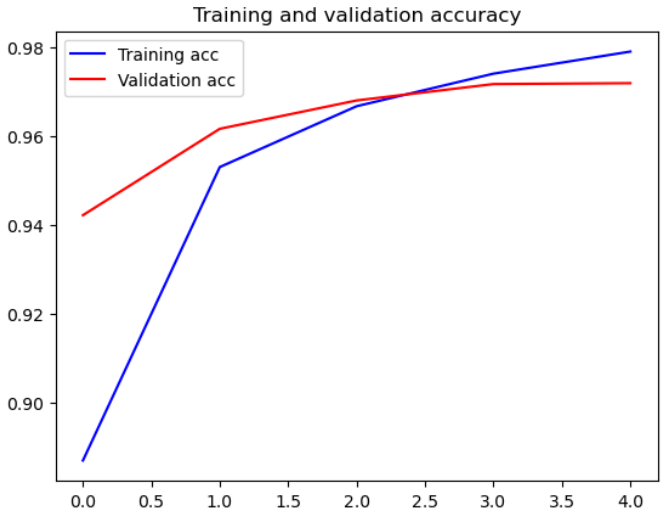
توضیحات مربوط به این مدل قبلاً آورده شده است و از توضیح مجدد آن خودداری می‌کنیم.
ابتدا مدل را با بهینه‌ساز Adam و تابع هزینه categorical_crossentropy و معیار دقت (accuracy) تنظیم کرده سپس عمل آموزش را روی داده‌های آموزش و لیبل‌های کدگذاری شده با تعداد ایپاک ۵ و اندازه دسته‌های ۲۰۰ را آغاز می‌کنیم. در این مرحله داده‌های ارزیابی را همان داده‌های آزمون در نظر می‌گیریم. در جدول شماره ۲ مقدار دقت و ضرر در هر ایپاک و در پایان را بررسی می‌کنیم.

ایپاک	دقت (Accuracy)	ضرر (Loss)
۱	۰.۸۸۶۸	۰.۳۹۴۶
۲	۰.۹۵۳۱	۰.۱۵۹۷
۳	۰.۹۶۶۸	۰.۱۱۳۴
۴	۰.۹۷۴۲	۰.۰۸۶۳
۵	۰.۹۷۹۱	۰.۰۶۸۷
مقدار نهایی	۰.۹۷۲۰	۰.۰۹۱۳

جدول 1 عملکرد مدل در هر ایپاک

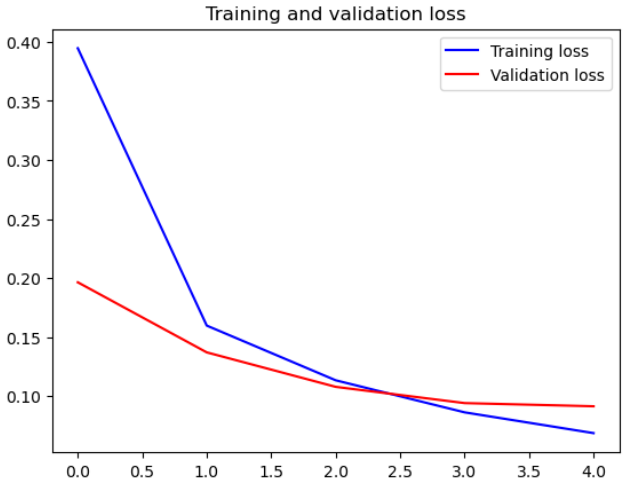
با بررسی این جدول درمی‌یابیم که مدل در هر مرحله ایپاک دقت بیشتری دارد و از مقدار ضرر آن کم می‌شود. در ادامه نمودار دقت و ضرر را بررسی می‌کنیم.

نمودار دقت:



شکل 8 نمودار دقت

باتوجه‌به نمودار دقت شکل ۸ درمی‌یابیم که دقت تمرین از حدود ۹۰ درصد شروع می‌شود و با افزایش ایپاک‌ها افزایش پیدا کرده و به حدود ۹۸ درصد می‌رسد. همچنین دقت اعتبارسنجی با افزایش تعداد ایپاک‌ها بیشتر می‌شود و به نظر می‌رسد به‌خوبی در حال همگرا شدن است. هر دو خط دقت آموزش و اعتبارسنجی همگرا هستند که نشان‌دهنده تناسب خوب است. این بدان معنی است که مدل به‌خوبی به داده‌های جدید و نادیده تعمیم می‌یابد.



شکل9 نمودار ضرر

با تحلیل نمودار ضرر شکل ۹ در میابیم که ضرر آموزش از مقدار زیادی شروع می‌شود و با افزایش ایپاک‌ها مقدار آن به‌شدت کاهش می‌یابد. این نشان می‌دهد که مدل در طول زمان به طور مؤثر از داده‌های آموزشی یاد می‌گیرد.
ضرر اعتبارسنجی نیز با افزایش تعداد دوره‌ها کاهش می‌یابد، اما این کار را با سرعت متری نسبت به ضرر تمرین انجام می‌دهد. ضرر اعتبارسنجی عموماً بالاتر از میزان ضرر آموزش است، این طبیعی است؛ زیرا انتظار می‌رود مدل در داده‌هایی که قبلاً ندیده است کمی بدتر عمل کند (داده‌های اعتبارسنجی).
در این نمودار هیچ نشانه‌ای از بیش برازش[10] وجود ندارد؛ زیرا ضرر اعتبارسنجی به همراه ضرر آموزش در حال کاهش است و با ادامه دوره‌ها هیچ افزایشی در افت اعتبارسنجی وجود ندارد، در ایپاک ۳ به بعد این مقدار کمی شیب آن به صفر میل می‌کند که به نظر می‌رسد با افزایش ایپاک‌ها احتمال بیش برازش وجود خواهد داشت. به‌صورت کلی بیش برازش با کاهش خطای آموزش و افزایش خطای اعتبارسنجی همراه است.

معیار دقت (Accuracy) برای این مجموعه‌داده به نظر می‌رسد معیار مناسبی باشد؛ ولی یک‌بار در این مدل معیارهای recall و f1-score را نیز تنظیم می‌کنیم و خروجی آن‌ها را مشاهده می‌کنیم (سایر پارامترها را تغییر نمی‌دهیم).

معیار دقت	دقت	ضرر (Loss)
recall	۰.۹۷۸۲	۰.۰۶۹
F1-score	۰.۹۷۷۷	۰.۰۸۲۷

جدول 2 عملگرد با معیار های مختلف

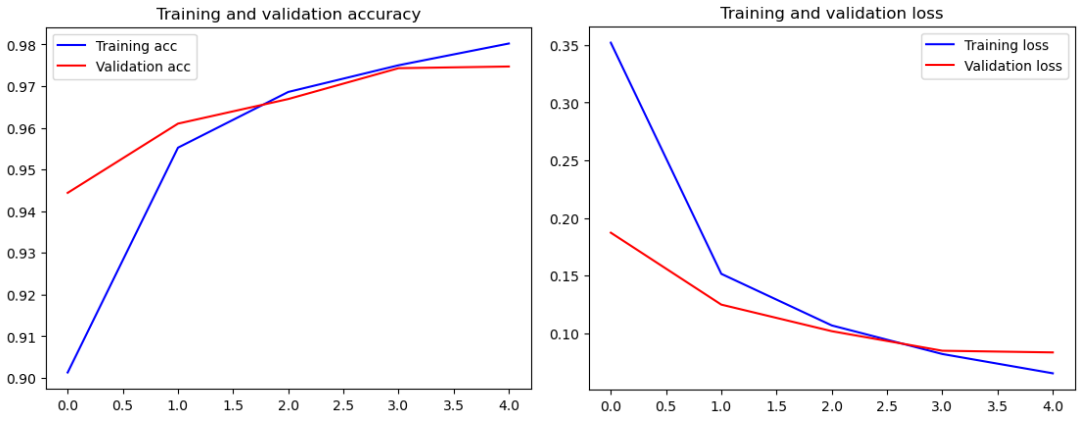
حال بهینه‌ساز را از adam به RMSprop تغییر می‌دهیم و عملکرد مدل را با معیار دقت (Accuracy) بررسی می‌کنیم (سایر پارامترها را تغییر نمی‌دهیم).

ایپاک	دقت (Accuracy)	ضرر (Loss)
۱	۰.۹۰۱۳	۰.۳۵۲۰
۲	۰.۹۵۵۲	۰.۱۵۱۶
۳	۰.۹۶۸۶	۰.۱۰۸۶
۴	۰.۹۷۵۰	۰.۰۸۲۰
۵	۰.۹۸۰۲	۰.۰۶۵۳
مقدار نهایی	۰.۹۷۴۶	۰.۰۸۳۲

جدول3 عملکرد با بهینه‌ساز RMSprop

از جدول فوق می‌توان نتیجه گرفت که این بهینه‌ساز مقداری از بهینه‌ساز Adam بهتر عمل می‌کند.

[10] overfit



شکل10 نمودار های دقت و ضرر بهینه‌ساز RMSprop

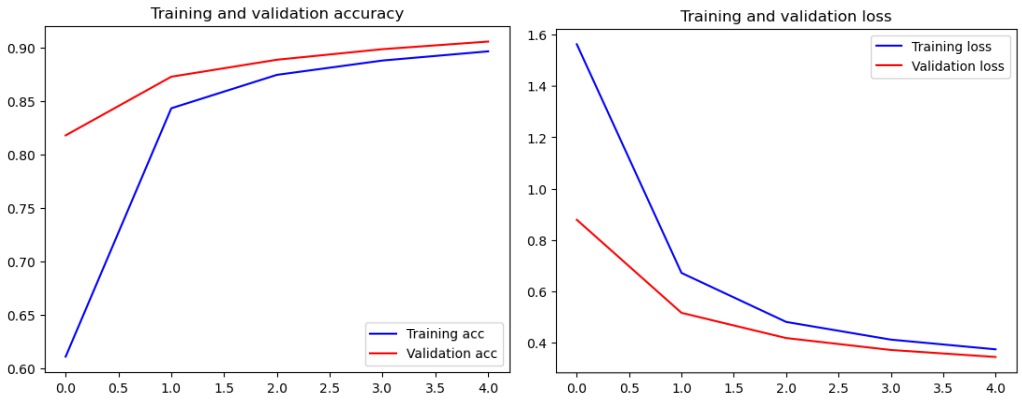
باتوجه‌به نمودارهای به‌دست‌آمده می‌توان ادعا کرد خروجی تقریباً با Adam برابر است و تمامی تحلیل‌های قبل از قبیل هم‌گرایی و بیش برازش برای این بهینه‌ساز نیز صادق است.

حال بهینه‌ساز را به SGD تغییر می‌دهیم و بدون تغییر در سایر اجزای مدل، فرایند آموزش را انجام می‌دهیم.

ایپاک	دقت (Accuracy)	ضرر (Loss)
۱	۰.۶۱۰۷	۱.۵۶۲۱
۲	۰.۸۴۳۲	۰.۶۷۱۵
۳	۰.۸۷۴۵	۰.۴۸۰۷
۴	۰.۸۸۷۹	۰.۴۱۱۷
۵	۰.۸۹۶۴	۰.۳۷۴۰
مقدار نهایی	۰.۹۰۵۶	۰.۳۴۳۹

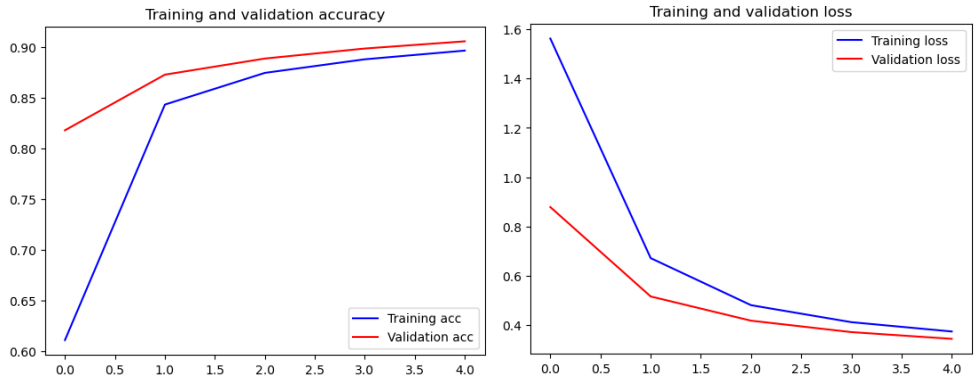
جدول 4 عملکرد با بهینه‌ساز SGD

به نظر می‌رسد که این بهینه‌ساز در دقت ضعیف‌تر عمل کرده است و برای دقت بالاتر شاید تعداد ایپاک بیشتری نیاز داشته باشد؛ اما باتوجه‌به نمودارهای دقت و ضرر که در شکل ۱۱ مشاهده می‌شود همگرایی به‌خوبی صورت می‌گیرد و بیش برازش نیز رخ نمی‌دهد.



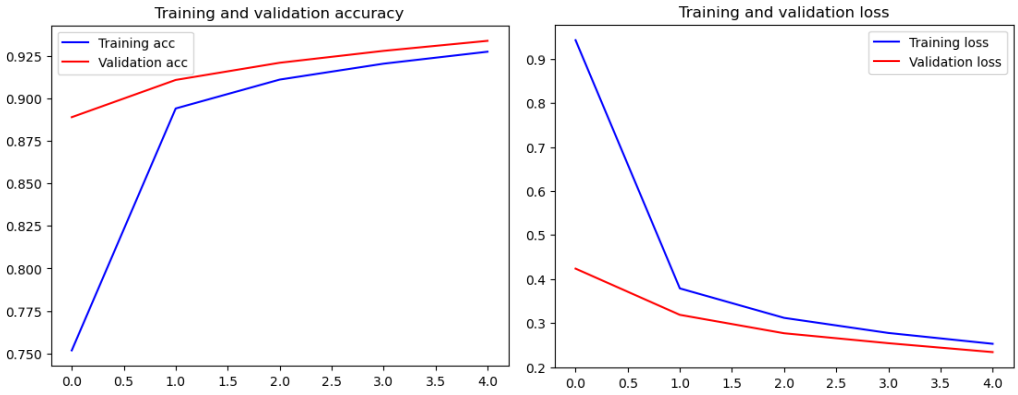
شکل11 نمودار های ضرر و عملکرد با بهینه‌ساز SGD

برای بهبود عملکرد این بهینه‌ساز می‌توان نرخ یادگیری را تغییر داد و تنظیم نمود. برای امتحان این موضوع نرخ یادگیری را به ۰.۰۱، ۰.۰۳، ۰.۰۰۰۱ قرار می‌دهیم و نتایج را بررسی می‌کنیم. ابتدا نرخ یادگیری را ۰.۰۱ قرار می‌دهیم و عملکرد دقت مدل به ۰.۹۰۳۶ می‌رسد که از عملکرد مقدار اولیه نرخ یادگیری ضعیف‌تر است؛ ولی از نظر نموداری خیلی تغییری نمی‌کند.



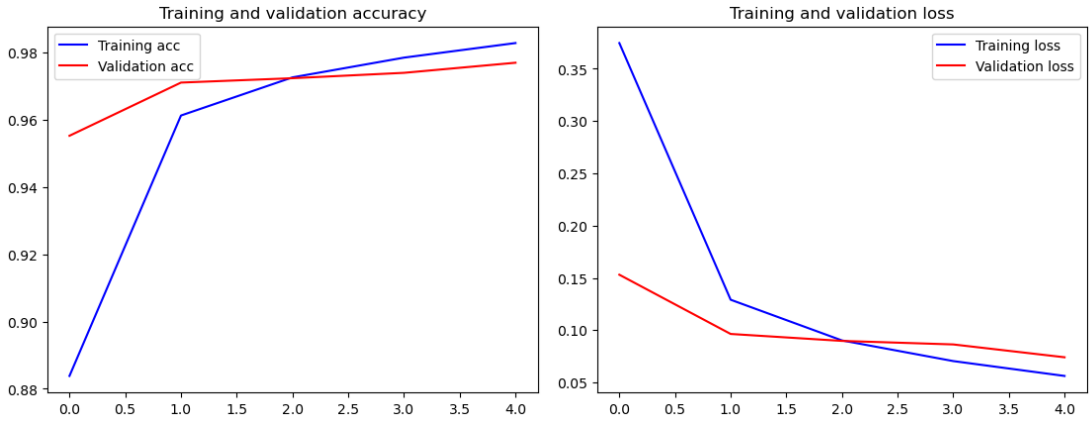
شکل12 نمودار عملکرد و ضرر با بهینه‌ساز SGD و نرخ یادگیری ۰.۰۱

با تغییر نرخ یادگیری به ۰.۰۳ دقت مدل به ۰.۹۳۳۷ می‌رسد که از دو نرخ قبلی بهتر است همچنین همگرایی به‌خوبی در حال رخ‌دادن است و بیش برازش اتفاق نمی‌افتد.



شکل13 نمودار عملکرد و ضرر با بهینه‌ساز SGD و نرخ یادگیری ۰.۰۳

با تغییر نرخ یادگیری به ۰.۰۰۰۱ دقت مدل به ۰.۱۱ می‌رسد که بسیار پایین است و مدل در این تعداد ایپاک به‌خوبی آموزش نمی‌بیند.
با تغییر نرخ یادگیری به ۰.۰۵ دقت مدل به ۰.۹۷۶۸ می‌رسد که از قبل بهتر است؛ اما همگرایی ضعیف‌تر می‌شود.

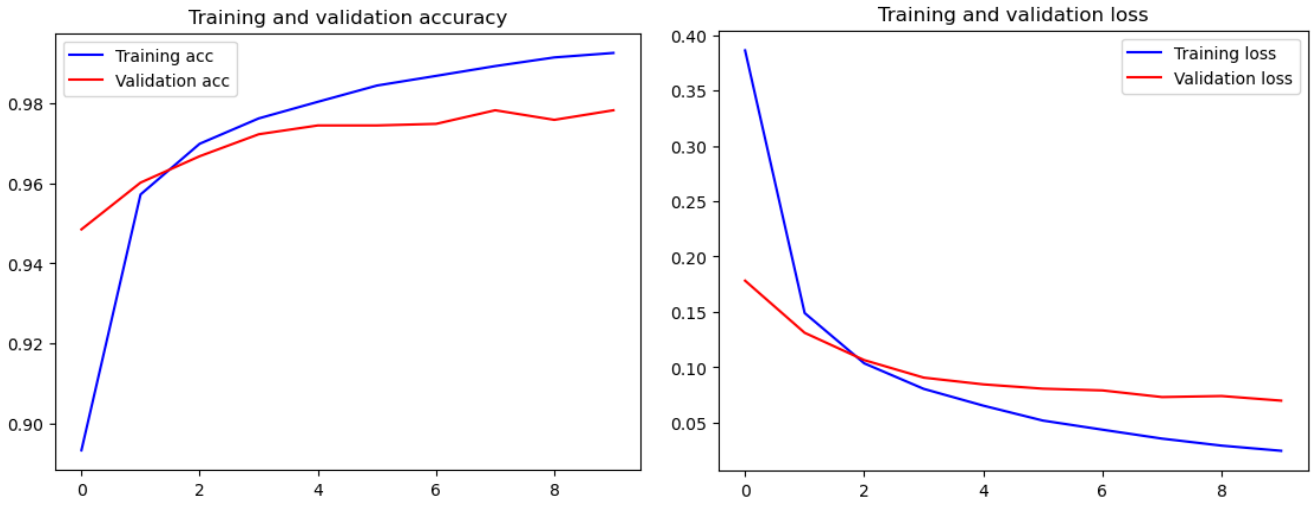


شکل14 نمودار عملکرد و ضرر با بهینه‌ساز SGD و نرخ یادگیری ۰.۰۵

با بررسی نتایج به‌دست‌آمده از عملکرد بهینه‌سازهای مختلف با توجه سادگی و کارایی بهینه‌ساز Adam و همچنین تعیین نرخ یادگیری به‌صورت تطبیقی در حین آموزش به‌صورت خودکار، در ادامه نیز از این بهینه‌ساز استفاده می‌شود و سایر بهینه‌سازها را برای مدل‌های دیگر بررسی نمی‌کنیم.

۲-۶ مدل شماره دو:

این مدل دقیقاً مثل مدل اول دارای دولایه پنهان ۱۲۰ و ۸۴ نورونی است. بهینه‌ساز، تابع هزینه و معیار دقت دقیقاً مثل مدل قبلی است. در این مدل ما تعداد ایپاک‌ها را ۲ برابر کردیم و ۱۰ بار همهٔ داده‌ها را مشاهده کردیم. در این حالت دقت مدل به ۰.۹۸۷۲ می‌رسد که نسبت به مدل قبلی بهتر است. اما باتوجه‌به شکل ۱۵، می‌توان نشانه‌های بیش برازش را مشاهده کرد. درحالی‌که دقت آموزش کمی افزایش می‌یابد و سطح بالایی را حفظ می‌کند، دقت اعتبارسنجی پس از رسیدن به اوج، کمی کاهش می‌یابد. این اختلاف ممکن است نشانگر بیش برازش زودهنگام باشد، جایی که مدل الگوهای خاص داده‌های آموزش یاد می‌گیرد که به داده‌های اعتبارسنجی تعمیم نمی‌یابند. بااین‌حال، کاهش دقت اعتبارسنجی چندان شدید نیست و دقت به طور نسبتاً بالا می‌ماند. این نشان می‌دهد که اگر بیش برازش اتفاق می‌افتد، در این مرحله به طور شدید نیست و با افزایش بیشتر ایپاک‌ها احتمال آن بیشتر می‌شود.



شکل15 نمودار عملکرد مدل دو

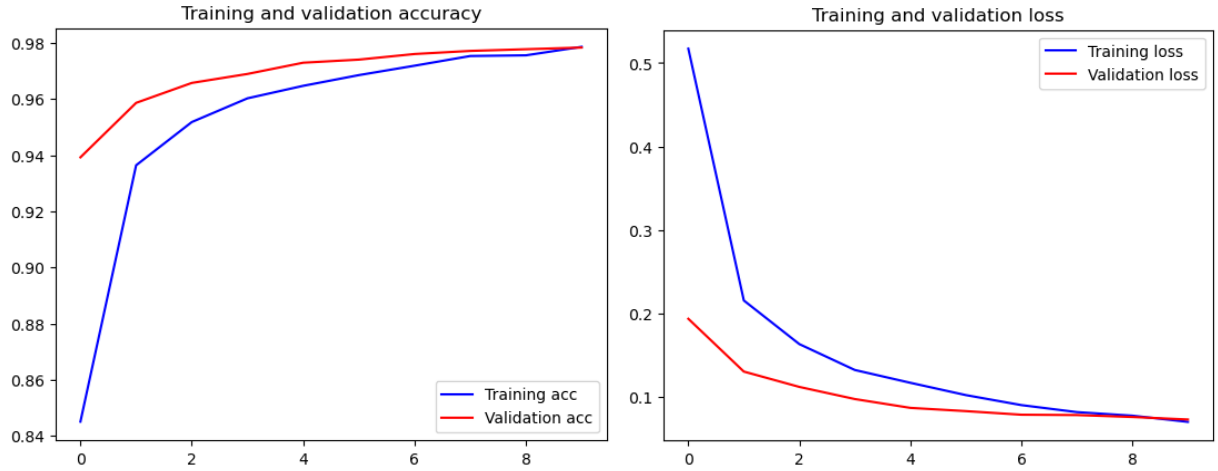
۳-۶ مدل شماره سه:

در این مدل ما قصد داریم از ۱۰ ایپاک استفاده کنیم به‌طوری‌که از بیش برازش جلوگیری شود. برای این امر ما باید لایهٔ Drop out اضافه کنیم. Dropout یک تکنیک مهم در شبکه‌های عصبی است که برای جلوگیری از بیش برازش به کار می‌رود. در این تکنیک، در هر مرحله از آموزش، به طور تصادفی بخشی از واحدهای یک‌لایه به طور موقت غیرفعال شده و از فرایند آموزش خارج می‌شوند. این باعث می‌شود که شبکهٔ عصبی به داده‌هایی که از پیش ندیده است عادت کند و الگوهایی کلی‌تر را یاد بگیرد. این کار باعث کاهش امکان بیش برازش مدل و افزایش عملکرد آن بر روی داده‌های جدید می‌شود. استفاده از drop out فوایدی مثل جلوگیری از بیش برازش، افزایش عمومیت مدل، افزایش مقاومت در برابر نویز و بهبود عملکرد دارد. برای نمونه ما دولایه Drop out در مدل اولیه، یکی بعد از لایهٔ پنهان اول و دیگری قبل از لایه خروجی قرار می‌دهیم و به مدل زیر می‌رسیم.

Layer (type)	Output Shape	Param #
flatten_9 (Flatten)	(None, 784)	0
dense_27 (Dense)	(None, 120)	94,200
dropout (Dropout)	(None, 120)	0
dense_28 (Dense)	(None, 84)	10,164
dropout_1 (Dropout)	(None, 84)	0
dense_29 (Dense)	(None, 10)	850

شکل 16 مدل شماره سه

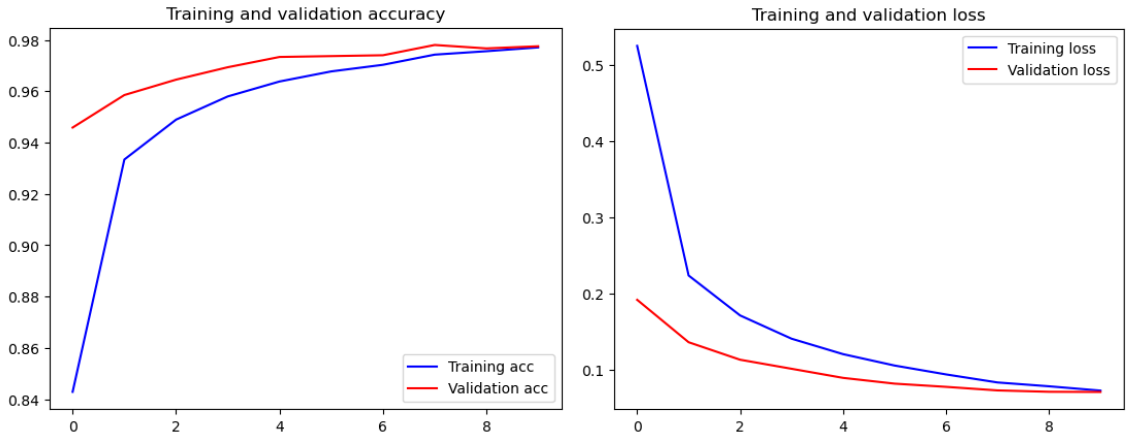
سپس مدل را با تمامی پارامترها و هایپرپارامترهای قبلی آموزش می‌دهیم و به ۰.۹۷۹۰ می‌رسیم. با افزایش تعداد ایپاک‌ها به‌دقت بالاتر هم می‌توان دست‌یافت. حال با بررسی نمودارهای دقت و ضرر می‌بینیم که مدل همگرایی بسیار خوبی دارد و همچنین باتوجه‌به اینکه دقت اعتبارسنجی، دقت آموزش را دنبال می‌کند و از آن پایین‌تر نمی‌آید، درگیر بیش برازش نشده است به‌علاوه این لایه‌ها باعث شده که تعادل خوبی بین یادگیری و عمومیت مدل به وجود آید و در نتیجه عملکرد بهتری نیز داشته باشیم.



شکل17 نمودار عملکرد و ضرر مدل سه

۶-۴ مدل شماره چهار:

در این مدل ما عیناً از مدل شماره سه استفاده می‌کنیم؛ اما به‌جای اینکه داده‌های اعتبارسنجی ما همان داده‌های آزمون باشند، آمدیم ۹۰ درصد از داده‌های آموزش را برای آموزش و ۱۰ درصد مابقی برای اعتبارسنجی در نظر گرفتیم، سپس آموزش را با این دو انجام داده و عملکرد را بر روی مجموعه‌داده آزمون بررسی کردیم. دقت عملکرد مدل بر روی مجموعه‌آزمون در این مدل ۰.۹۷۶۷ شد و از نظر نموداری نیز تغییر خاصی نکرد. برای جداسازی داده‌های آموزش و اعتبارسنجی از کتابخانه scikit-learn استفاده کردیم. شکل ۱۸ نمودارهای این مدل را نشان می‌دهد.



شکل18 عملکرد مدل شماره چهار

۷ نتیجه‌گیری:

در این تمرین ما با مجموعه‌دادهٔ MNSIT آشنا شدیم و مقداری عملیات پیش‌پردازش را بر روی آن انجام دادیم. سپس با نحوهٔ ساخت مدل MLP و کامپایل و آموزش آشنا شدیم. انواع پارامترها و هایپرپارامترهای موجود را بررسی کردیم. در مدل‌های مختلفی که ساختیم سعی کردیم از هایپرپارامترهای مختلف استفاده کنیم تا نتایج را بتوانیم بررسی کنیم و سعی کنیم به مدل بهتری دست پیدا کنیم. بعد از آموزش مدل‌های مختلف می‌توان ادعا کرد به‌دقت حدود ۹۸ درصد رسیدیم که البته همچنان می‌توان برای بهبود آن اقداماتی را صورت داد و به درصدهای بالاتر نیز دست پیدا کرد به‌عنوان‌مثال استفاده از شبکه‌های CNN می‌تواند در بهبود عملکرد تأثیر داشته باشد.

در ادامه با استفاده از شبکه های عصبی کانولوشنی و همچنین استفاده از معماری معروف AlexNet این تمرین را انجام می دهیم تا بتوانیم نتایج به دست آمده در مرحله قبلی و این مرحله را با هم مقایسه کنیم، همچنین از چند تکنیک یادگیری ماشین برای طبقه بندی استفاده کردیم که در ادامه به بررسی آن ها خواهیم پرداخت.

۸ شبکه عصبی کانولوشنی

شبکه‌های عصبی کانولوشنی Convolutional Neural Networks یکی از انواع شبکه‌های عصبی مصنوعی هستند که به طور خاص برای پردازش داده‌های ساختاریافته مثل تصاویر طراحی شده‌اند. این شبکه‌ها به طور گسترده در زمینه‌های مختلفی از جمله تشخیص تصویر، تشخیص الگو، و بینایی ماشین استفاده می‌شوند.

۸.۱ ساختار و اجزای اصلی شبکه‌های عصبی کانولوشنی:

لایه‌های کانولوشن:(Convolutional Layers)

وظیفه این لایه‌ها استخراج ویژگی‌ها از داده ورودی است. هر لایه کانولوشن شامل چندین فیلتر (Kernel) است که بر روی تصویر ورودی اعمال می‌شوند و نقشه‌های ویژگی (Feature Maps) تولید می‌کنند.
فیلترها به صورت ماتریس‌های کوچکی هستند که بر روی تصویر لغزیده می‌شوند و مقادیر خروجی را محاسبه می‌کنند.

لایه‌های پولینگ:(Pooling Layers)

این لایه‌ها برای کاهش ابعاد نقشه‌های ویژگی و کاهش تعداد پارامترها و محاسبات استفاده می‌شوند. دو نوع پولاژینگ رایج عبارتند از Max Pooling وAverage Pooling

Max Pooling بزرگترین مقدار از هر ناحیه کوچکی از نقشه ویژگی را انتخاب می‌کند، در حالی که Average Pooling میانگین مقادیر را می‌گیرد.

لایه‌های تمام‌متصل:(Fully Connected Layers)

این لایه‌ها مشابه لایه‌های شبکه‌های عصبی معمولی (MLP) هستند و به عنوان طبقه‌بند نهایی عمل می‌کنند. در اینجا، نقشه‌های ویژگی تخت می‌شوند و به یک بردار یک‌بعدی تبدیل می‌شوند که سپس به لایه‌های تمام‌متصل ارسال می‌شوند.

لایه‌های نرمال‌سازی و فعال‌سازی:(Normalization and Activation Layers)

این لایه‌ها به شبکه کمک می‌کنند تا بهتر یاد بگیرد و مشکلات ناشی از نرمال‌سازی داده‌ها را کاهش دهند.
تابع‌های فعال‌سازی رایج شامل ReLU (Rectified Linear Unit) و Sigmoid هستند.

در این تمرین ما از دو مدل شبکه عصبی کانولوشنی استفاده کردیم که در ادامه به بررسی آن ها می پردازیم. لازم به ذکر است مراحل پیش پردازش عینا مانند نسخه قبلی می باشد فقط لیبل های آموزش و تست را one-hot کدگذاری نکرده ایم .

۸.۲ مدل شماره ۱

در ابتدا یک لایه ورودی با استفاده از InputLayer تعریف کردیم که شکل ورودی داده ها را مشخص می کند که در این تمرین برابر با ۱*۲۸*۲۸ هست. سپس لایه کانولوشن اول با ۱۶ فیلتر و اندازه کرنل ۳*۳ و تابع فعال ساز ReLU تعریف کردیم تا ویژگی ها محلی را استخراج کند. بعد از آن یک لایه BatchNormalization تعریف کردیم تا از بیش برازش جلوگیری کنیم. سپس از یک MaxPool استفاده کردیم تا محاسبات را کاهش دهیم و وابستگی به مکان را کاهش دهیم. در نهایت یک لایه Dropout با نرخ ۰.۲۵ قرار دادیم تا از بیش برازش جلوگیری

کنیم. در ادامه عینا لایه های فوق را تکرار کردیم فقط برای لایه کانولوشن دوم از ۳۲ فیلتر با اندازه کرنل ۵*۵ استفاده کردیم. در نهایت با یک لایه Flatten خروجی را به یک بردار تک بعدی تبدیل کردیم تا به لایه های Dense بدهیم. در این مرحله یک لایه Dense با تمام متصل با ۱۲۸ نورون و فعال ساز ReLU گذاشته ایم و بعد از آن یک BatchNormalization و Dropout با نرخ ۰.۵ . در نهایت با توجه به تعداد کلاس هایمان یک لایه Dense دیگر با ۱۰ نورون تعریف کردیم و از فعال ساز softmax استفاده کردیم .

۸.۲.۱ کامپایل مدل ۱

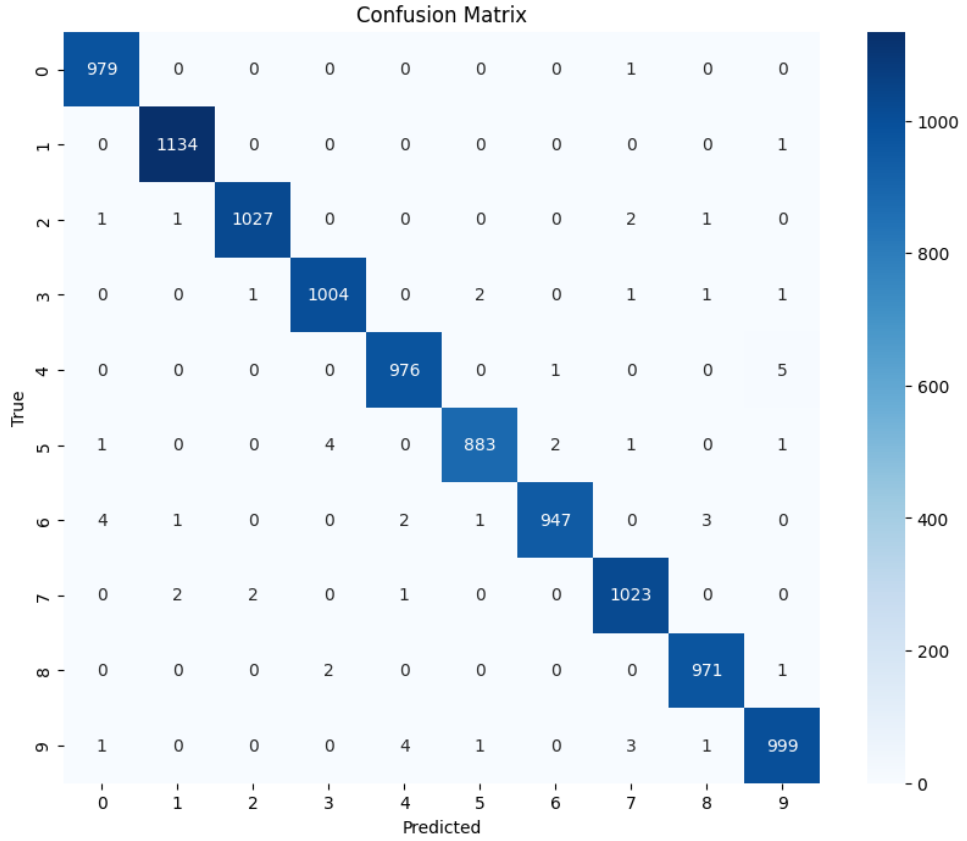
مدل تعریف شده در مرحله قبلی را با استفاده از بهینه ساز Adam که قبلا راجع به آن صحبت شده است و تابع ضرر sparse_categorical_crossentropy و با معیار accuracy کامپایل کردیم. دلیل استفاده از این تابع ضرر این است که ما در این نسخه لیبل هارا one-hot کدگذاری نکرده ایم.

۸.۲.۲ آموزش مدل ۱

در این مرحله با اندازه دسته های ۳۲ تایی و تعداد ۲۰ ایپاک با استفاده از داده های آموزش موجود مدل را آموزش می دهیم. لازم به ذکر است در این مرحله از داده های اعتبارسنجی استفاده نکردیم.

۸.۲.۳ عملکرد مدل ۱ و ماتریس درهم ریختگی

با بررسی عملکرد مدل آموزش دیده بر روی داده های تست به دقت ۹۹.۴ درصد می رسیم که از بهترین مدل آموزش دیده شده توسط ما با شبکه های عصبی ساده حدود ۲ درصد بهبود داشته است. در شکل شماره ۱۹ ماتریس درهم ریختگی این مدل را مشاهده می کنیم. مثلا برای عدد ۰ ، ۹۷۹ پیش بینی صحیح داشته و فقط یک مورد از ۰ هارا ۷ پیش بینی کرده است. یا برای ۱ ها که به نظر ساده تر از بقیه اعداد می رسند ۱۱۳۴ مورد را درست و فقط یک مورد را ۹ پیش بینی کرده است. برای عددی مثل ۴ ، ۹۷۶ مورد را درست و ۵ مورد را ۹ و ۱ مورد را ۶ پیش بینی کرده است. در کل می توان گفت عملکرد مدل خوب بوده است .



شکل۱۹ ماتریس درهم ریختگی مدل شماره ۱

۸.۳ مدل شماره ۲

در این مدل نیز ما از لایه های کانولوشنی و MaxPooling استفاده کردیم . ابتدا یک لایه کانولوشن ۳۲ فیلتر و سایز کرنل ۳*۳ استفاده کردیم و سپس یک MaxPooling ۲*۲ زده ایم. مجدد لایه کانولوشن با ۶۴ فیلتر و سایز کرنل ۳*۳ استفاده کردیم و لایه MaxPooling را تکرار کرده ایم، سپس آخرین لایه کانولوشن را تکرار کرده و بعد از آن یک GlobalAveragePooling گذاشته ایم که از لایه Flatten که تعداد پارامتر های زیادی تولید می کرد استفاده نکنیم و همچنین بتوانیم هر سایز عکسی که داشتیم را به مدل بدهیم و وابسته با اندازه عکس های مجموعه آموزش نباشیم.
در ادامه یک لایه تمام متصل با ۶۴ نورون و فعال ساز ReLU و dropout با نرخ ۰.۵ گذاشتیم. در پایان هم مثل مدل قبلی یک لایه تمام متصل با ۱۰ نورون و فعال ساز softmax استفاده کرده ایم.

۸.۳.۱ کامپایل مدل

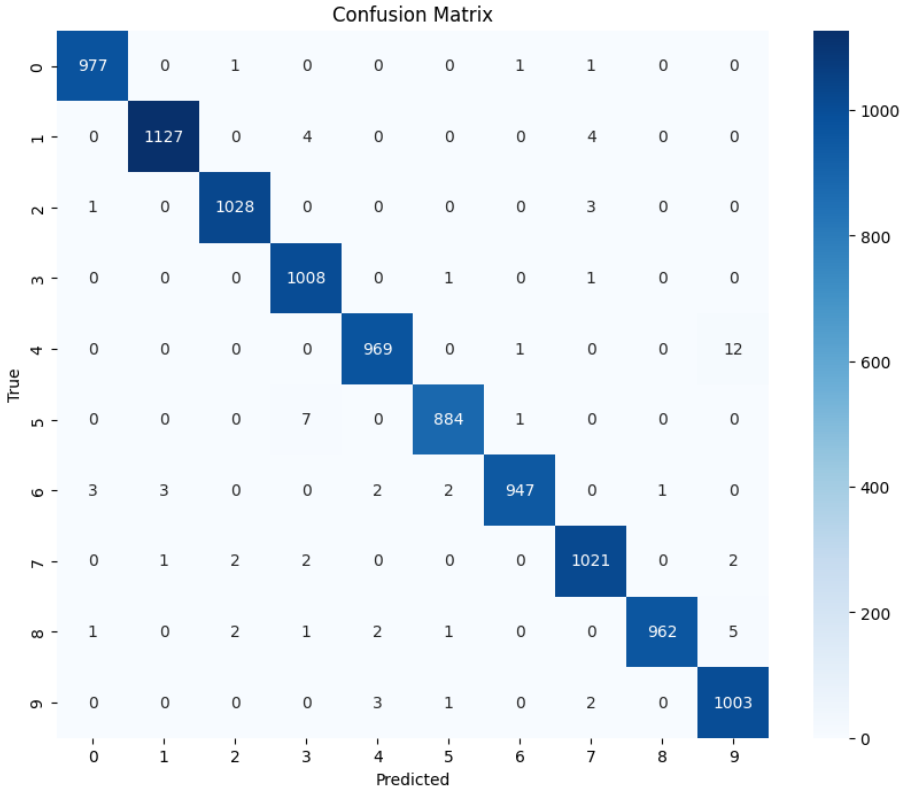
عینا مانند مدل ۱ می باشد و از بیان مجدد خودداری کردم

۸.۳.۲ آموزش

در مرحله آموزش نیز دقیقا مثل مدل ۱ عمل کردیم فقط با دستور validation_split=0.2 ، ۰.۲ از مجموعه داده آموزش را برای اعتبار سنجی کنار گذاشتیم تا هر مرحله اعتبار سنجی نیز انجام دهیم.

۸.۳.۳ عملکرد مدل ۲ و ماتریس درهم ریختگی

با بررسی عملکرد مدل آموزش دیده بر روی داده های تست به دقت ۹۹.۲ درصد می رسیم که از بهترین مدل آموزش دیده شده توسط ما با شبکه های عصبی ساده حدود ۲ درصد بهبود داشته است ولی از مدل شماره ۱ حدود ۰.۲ ضعیف تر عمل کرده است. در شکل شماره ۲۰ ماتریس درهم ریختگی این مدل را مشاهده می کنیم. مثلا برای عدد ۰ ، ۹۷۷ پیش بینی صحیح داشته و یک مورد از ۰ هارا و ۷ دیگری را ۶ پیش بینی کرده است. یا برای ۱ ها که به نظر ساده تر از بقیه اعداد می رسند ۱۱۲۴ مورد را درست و ۴ مورد را ۷ و ۴ مورد را ۳ پیش بینی کرده است. برای عددی مثل ۴ ، ۹۶۹ مورد را درست و ۱۲ مورد را ۹ و ۱ مورد را ۶ پیش بینی کرده است. در کل می توان گفت عملکرد مدل نسبت به مدل قبلی ضعیف تر بوده است.



شکل ۲۰ ماتریس درهم ریختگی مدل شماره ۲

۸.۴ مدل شماره ۳ (الکس نت)

در این مرحله ما از ساختار کلی معماری الکس نت استفاده کردیم ولی با توجه به سادگی بیشتر مساله ما از فیلتر های کم تر با اندازه کوچکتر استفاده کردیم. همچنین در لایه های تمام متصل نیز از نورون های کم تری استفاده کردیم. در این مدل ما از ۵ لایه کانولوشن ، ۱ لایه فلتن ، ۲ لایه تمام متصل و یک لایه خروجی گذاشته ایم.

در لایه های کانولوشن از کانولوشن با تعداد فیلتر های ۳۲ ، ۶۴ ، ۱۲۸ ، ۱۲۸ ، ۶۴ و با سایز کرنل ۳*۳ ، padding = same و فعال ساز ReLU همچنین از BatchNormalization و MaxPooling استفاده کردیم. سپس از Flatten استفاده کرده و مدل را به تمام متصل ها وصل کردیم که در لایه اول ۱۰۲۴ نورون دارد و BatchNormalization و Dropout با نرخ ۰.۵ ، سپس در لایه بعدی ۵۱۲ نورون و موارد لایه قبل تکرار شد. در هر دو لایه از ReLU استفاده شد.

۸.۴.۱ کامپایل مدل

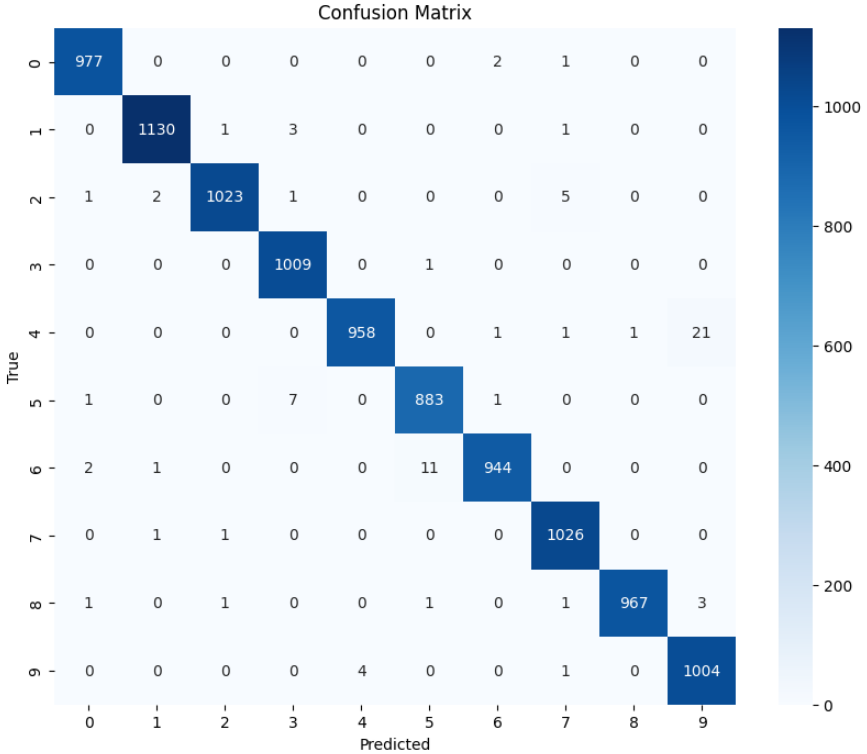
عینا مانند مدل ۱ و ۲ می باشد و از بیان مجدد خودداری کردم

۸.۴.۲ آموزش

در مرحله آموزش نیز دقیقا مثل مدل ۲ عمل کردیم.

۸.۴.۳ عملکرد مدل ۲ و ماتریس درهم ریختگی

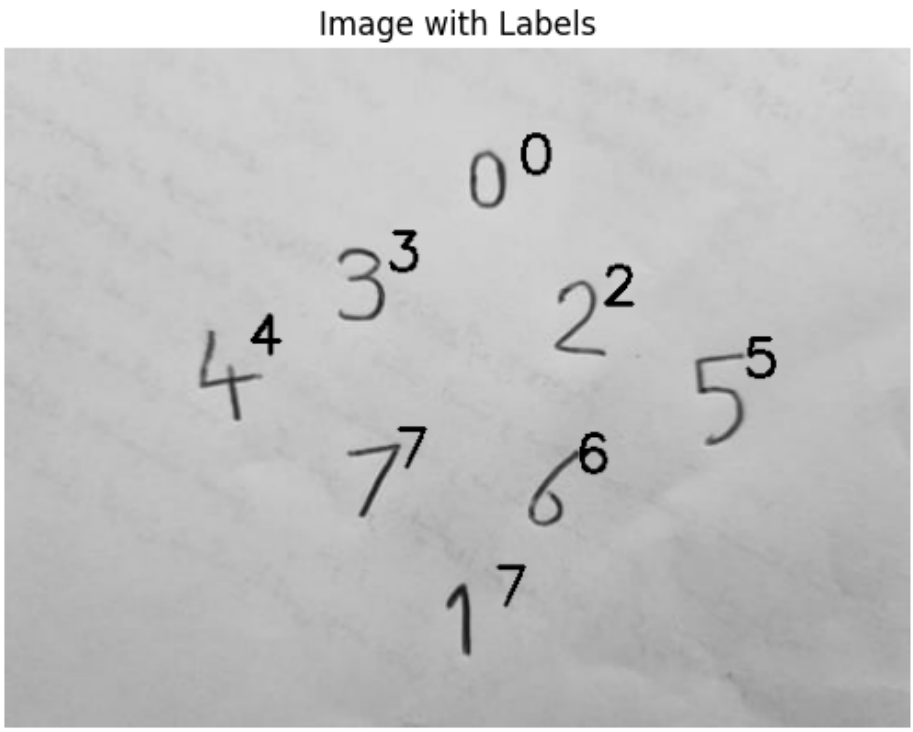
با بررسی عملکرد مدل آموزش دیده بر روی داده های تست به دقت ۹۹.۲ درصد می رسیم که از بهترین مدل آموزش دیده شده توسط ما با شبکه های عصبی ساده حدود ۲ درصد بهبود داشته است ولی از مدل شماره ۱ حدود ۰.۲ ضعیف تر عمل کرده است و برابر با مدل شماره ۲ است. در شکل شماره ۲۱ ماتریس درهم ریختگی این مدل را مشاهده می کنیم. مثلا برای عدد ۰ ، ۹۷۷ پیش بینی صحیح داشته و یک مورد از ۰ هارا ۷ و ۲ مورد دیگری را ۶ پیش بینی کرده است. یا برای ۱ ها که به نظر ساده تر از بقیه اعداد می رسند ۱۱۳۰ مورد را درست و ۱ مورد را ۷ و ۳ مورد را ۳ و ۱ مورد را ۲ پیش بینی کرده است. برای عددی مثل ۴ ، ۹۵۸ مورد را درست و ۲۱ مورد را ۹ و ۱ مورد را ۶ ، ۱ مورد را ۷ و ۱ مورد را ۸ پیش بینی کرده است. در کل می توان گفت عملکرد مدل نسبت به مدل های قبلی ضعیف تر بوده است.البته لازم به ذکر است که این نتایج حاصل یک بار اجرا روی مدل های مختلف است و شاید با اجرا های متعدد دیگر ، نتایج متفاوت و قابل قبول تری به دست می آمد.



شکل۲۱ ماتریس درهم ریختگی مدل شماره ۳

۸.۵ بررسی عملکرد مدل شماره ۳ بر روی یک عکس ورودی

در این مرحله عکسی را که تعدادی عدد دست نویس روی آن نوشته ایم را با استفاده از کتابخانه cv2 بارگذاری می کنیم. سپس در قطعه کد مربوطه عدد های موجود در عکس را پیدا کرده و برش می زنیم. سپس عملیات پیش پردازش و تغییر ابعاد را انجام داده و هر عکس را به مدل می دهیم تا پیش بینی کند چه عددی نوشته شده است. در نهایت جواب مدل را در بالای سمت راست عدد دست نوشته شده قرار می دهیم. مشاهده می کنیم که عدد ۱ را به اشتباه ۷ تشخیص داده و سایر اعداد را به درستی تشخیص داده است.



شکل ۲۲ پیش بینی اعداد نوشته شده با استفاده از مدل شماره ۳

۹ استفاده از مدل های یادگیری ماشین

در این مرحله ما از الگوریتم های KNN ، SVM و Logistic Regression که در یادگیری ماشین با آن ها آشنا شدیم استفاده کردیم تا بتوانیم نتیجه استفاده از شبکه های عصبی و الگوریتم های یادگیری ماشین را برای این تسک بررسی کنیم. در ابتدا مجموعه داده ها را لود کرده و عکس های موجود در مجموعه داده ی آموزش و تست را به شکل تعداد سمپل و تعداد فیچر تبدیل کردیم. برای به دست آوردن تعداد فیچر تمام پیکس های عکس را زیر هم چیده ایم که حاصل ضرب ۲۸ در ۲۸ برابر با ۷۸۴ فیچر به ما داد. سپس با هر ۳ الگوریتم و با استفاده از کتابخانه scikit-learn آموزش را انجام دادیم و عملکرد هر مدل را بررسی کردیم که در جدول شماره ۵ نتایج را مشاهده می کنید.

جدول ۵ عملکرد الگوریتم های یادگیری ماشین

مدل	دقت
KNN with K = 3	97.05
SVM	86.2
Logistic Regression	92.55

می توان به این نتیجه رسید که الگوریتم KNN نسبتا دقت خوبی در بین این ۳ الگوریتم دارد اما باز هم استفاده از شبکه های عصبی کانولوشنی دقت بهتری را به ما خواهد داد. در ادامه یک جدول مقایسه ای از دقت تمامی مدل هایی که تا اینجا به آن ها پرداختیم را خواهیم داشت .

جدول ۶ مقایسه عملکرد مدل های مختلف

مدل	دقت
MLP_Model_1	97.20
MLP_Model_2	97.82
MLP_Model_3	97.82
MLP_Model_4	97.67
CNN_Model_1	**99.43**
CNN_Model_2	**99.26**
CNN_Model_3	**99.21**
KNN with K=3	97.05
SVM	86.2
Logistic Regression	92.55

۱۰ مجموعه داده ارقام دست نوشته فارسی

در ادامه این تمرین از مجموعه داده هدی استفاده کردیم تا مدل های قبلی را برای تشخیص اعداد دست نوشته فارسی نیز استفاده کنیم. این مجموعه داده اولین مجموعه ی بزرگ ارقام دست نویس فارسی است. داده های این مجموعه از حدود ۱۲۰۰۰ فرم ثبت نام آزمون سراسری کارشناسی ارشد سال ۱۳۸۴ و آزمون کاردانی پیوسته دانشگاه جامع علمی و کاربردی سال ۱۳۸۳ استخراج شده است . فرمت این مجموعه داده به صورت فایل متلب می باشد. با استفاده از فایل پایتونی و تابع نوشته شده توسط مهندس اخوان پور[11]، داده ها را به آموزش و تست تقسیم بندی کرده و اندازه هر عکس را به ۲۸*۲۸ تبدیل می کنیم. سپس عملیات پیش پردازش داده ها را که در مراحل قبلی استفاده کردیم نیز استفاده می کنیم تا داده ها نرمال و به صورت one-hot کد گذاری شوند. سپس دو مدل، مدل شماره ۱ و مدل شماره ۳ (الکس نت) را تعریف می کنیم و مثل قبل مدل را کامپایل و آموزش را انجام می دهیم. این دو مرحله عینا مانند قسمت قبلی می باشد و از ذکر مجدد آن خودداری کرده ایم فقط با توجه به one-hot کردن لیبل ها از categorical_crossentropy استفاده کردیم. در جدول شماره دقت های حاصل از این آموزش با ۱۰۰۰۰ عکس آموزش و ۲۰۰۰ عکس تست را مشاهده می کنید.

جدول ۷ مقایسه عملکرد دو مدل بر روی مجموعه داده هدی

مدل	دقت
CNN_1	98.8
CNN_2(AlexNet)	98.9

در پایان می توان نتیجه گرفت که استفاده از CNN ها در معماری پیشنهادی می توان دقت را بهبود بخشد.

با تشکر از زحمات شما

[11] https://github.com/Alireza-Akhavan/deeplearning-tensorflow2-notebooks/blob/master/dataset.py