به نام خدا

سند پروژه‌ی اول درس هوش محاسباتی

استاد درس: دکتر هادی تابع‌الحجه

رضا اعلایی

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

نیم‌سال دوم تحصیلی ۰۳ - ۱۴۰۲

فهرست مطالب

توضیحات کلی پروژه ۳
توضیحات کلی مدل ۴
تعداد لایه‌های نهان ۴
تعداد نورون‌های هر لایه ۶
الگوریتم‌های بهینه‌سازی ۷
نرخ یادگیری ۹
شرط توقف ۱۱
توابع فعال‌سازی ۱۱
drop out ۱۳
batch normalization ۱۵
نتایج مدل و تنظیمات نهایی ۱۶

توضیحات کلی پروژه

هدف ما در این پروژه، ارائه یک شبکه تمام متصل چندلایه از پرسپترون‌ها (fully Connected MLP) برای کلاسیفیکیشن مجموعه‌داده‌ای mnist است.

در هر بخش از سند، تاثیر و میزان تاثیر یکی از عوامل زیر، با ثابت نگاه داشتن بقیه عوامل، بررسی می‌شود.

همچنین در هر بخش، نمودارهای accuracy و loss مربوط به آن قسمت ارائه می‌شود که دید بهتری از میزان تاثیرات فراهم آورده شود. عوامل مورد بحث در این مدل عبارتند از:

- تعداد لایه‌های نهان

- تعداد نورون‌های هر لایه

- تاثیر الگوریتم‌های بهینه‌سازی مختلف

- تاثیر نرخ یادگیری

- شرایط توقف متفاوت

- توابع فعال‌سازی متفاوت

همچنین تاثیر دو مورد زیر نیز در ادامه بررسی خواهد شد:

- تاثیر drop out

تاثیر بچ نرمالیزیشن

همچنین در خروجی، پس از انجام هر epoch مقدار دقت، loss، validation accuracy، validation loss نیز محاسبه می‌شود.

توضیحات کلی مدل

در کد مدل دو قسمت مهم‌تر وجود دارد: تابع build_model و قسمت config.

تابع build_model:

اطلاعات مربوط به عوامل مختلف را دریافت کرده و بر اساس آن‌ها مدل را می‌سازد.

قسمت config:

برای انجام تغییرات در کد و آزمایش کردن آن به کار می‌رود، به کمک این ساختار داده‌ای میتوان مدل را بر روی تنظیمات مختلف آموزش داد و نتیجه را مشاهده کرد.

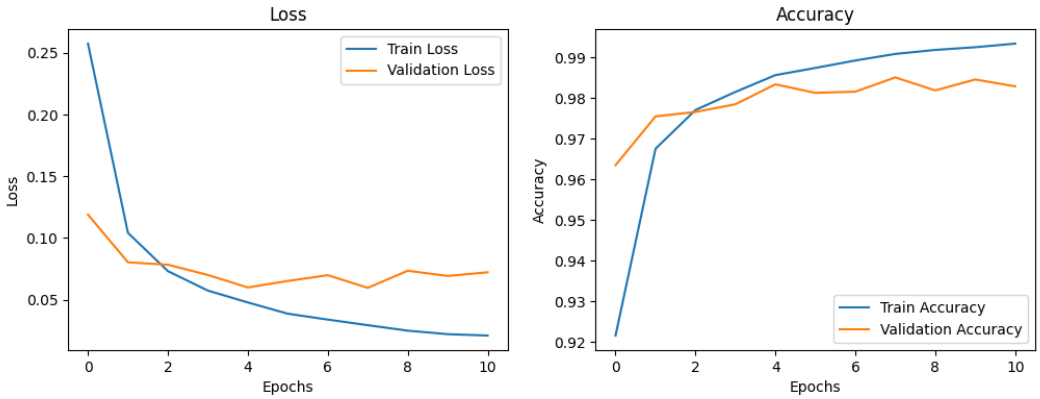
همچنین از تابع plot_config نیز برای رسم نمودارهای accuracy و loss استفاده شده است.

تعداد لایه‌های نهان

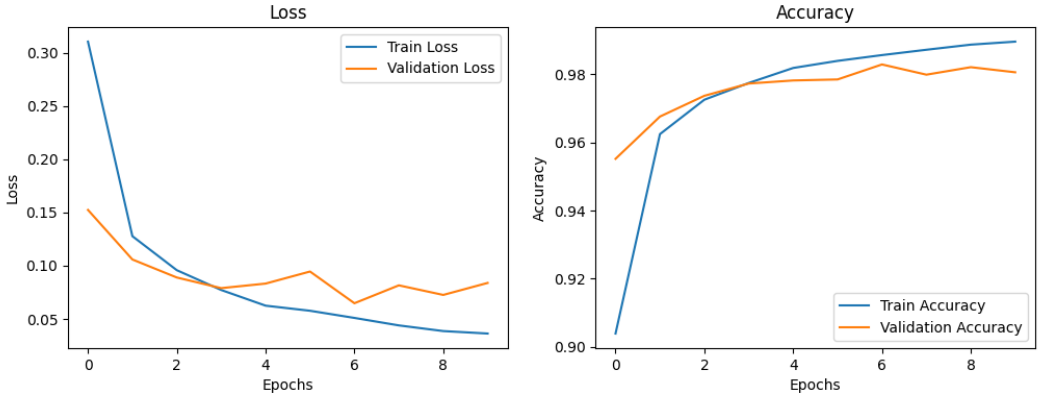
هر MLP از حداقل صفر یا چند لایه‌ی نهان تشکیل شده است. برای این پروژه، تعداد لایه‌های مختلفی تنظیم و امتحان شد.

در زیر برای تعداد لایه‌های مساوی با ۲، ۴، ۷ و ۱۰، نمودارهای مربوطه آمده است.

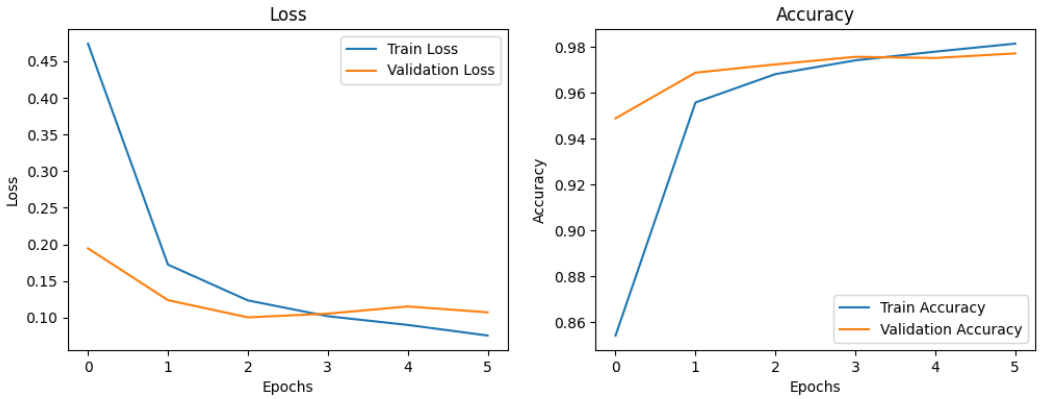
با توجه کردن به نتایج مربوطه به هر epoch و همچنین نتایج کلی مدل، بنظر می‌رسد تعداد ۲ برای لایه‌های نهان عدد مناسبی است؛ چرا که هم به دقت خوب و مقدار loss مناسب می‌رسیم و هم نیازی به سخت‌افزار بیشتر برای اجرای مدل نداریم.



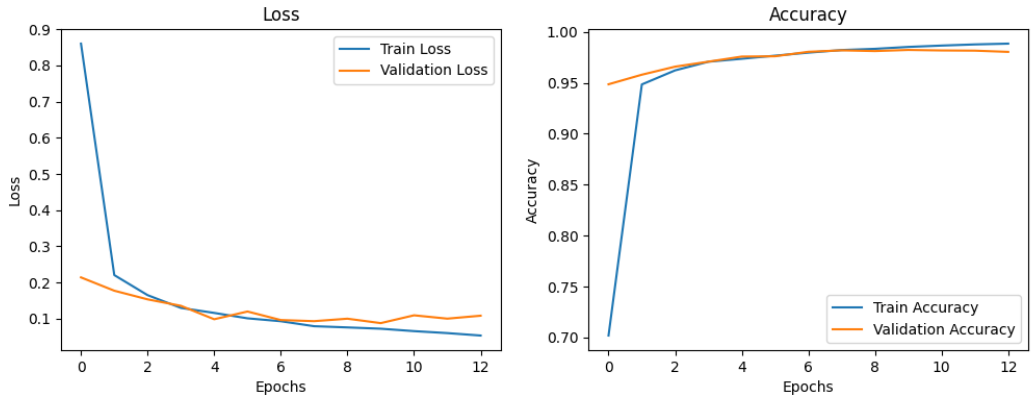
تصویر ۱ – تعداد لایه‌های نهان برابر ۲



تصویر ۲ – تعداد لایه‌های نهان برابر ۴



تصویر ۳ – تعداد لایه‌های نهان برابر ۷

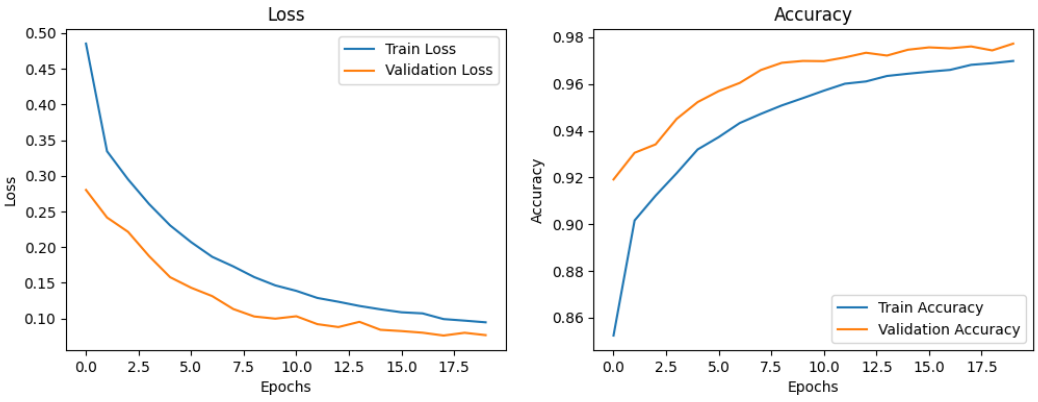


تصویر ۴ – تعداد لایه‌های نهان برابر ۱۰

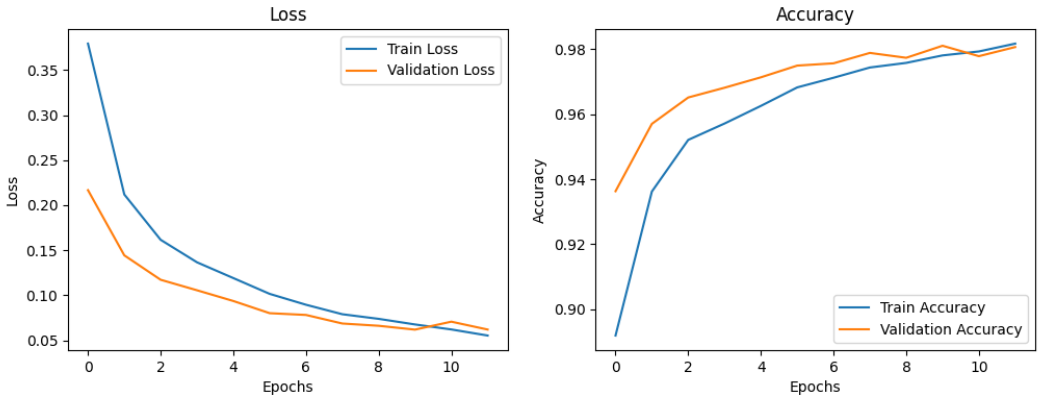
تعداد نورون‌های هر لایه

برای این کار چندین تعداد مختلف برای نورون‌های هر لایه در نظر گرفته شد. برای مقادیر پایین، به وضوح میتوان دید که مدل fit نیست. همچنین در مقادیر بالا، نمودارها، رفتارهایی از overfitting یا بیش برازش از خود نشان می‌دهند.

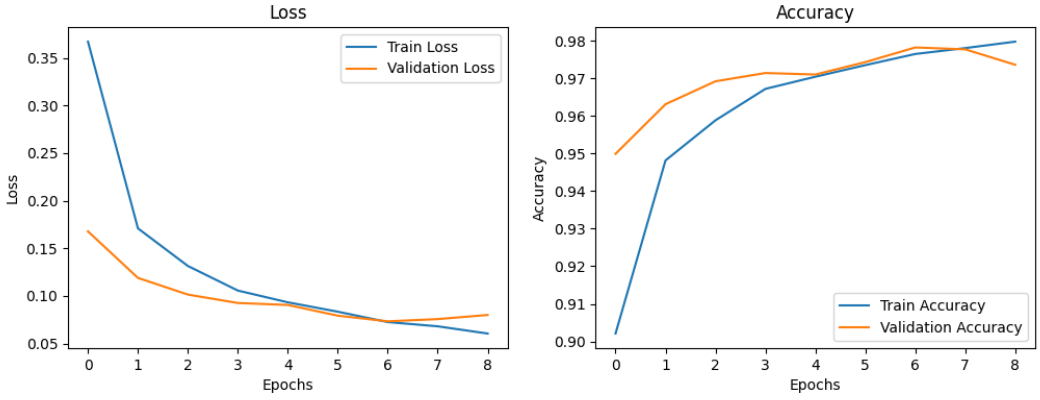
در زیر نمودارها برای سه مقدار ۱۰۰، ۵۰۰ و ۱۰۰۰ آورده شده است. مقدار نهایی این فاکتور در مدل نهایی، ۵۰۰ است.



تصویر ۵ – نورون‌های هر لایه = ۱۰۰



تصویر ۶ – نورون‌های هر لایه = ۵۰۰



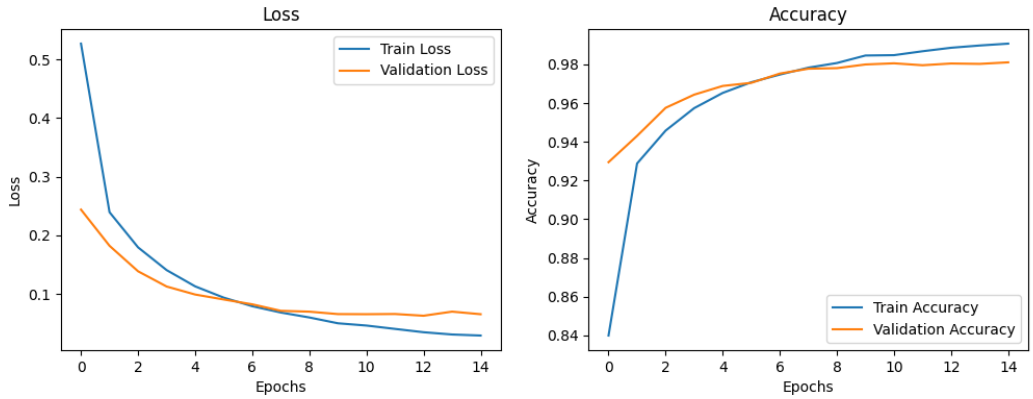
تصویر ۷ – نورون‌های هر لایه = ۱۰۰۰

الگوریتم‌های بهینه‌سازی

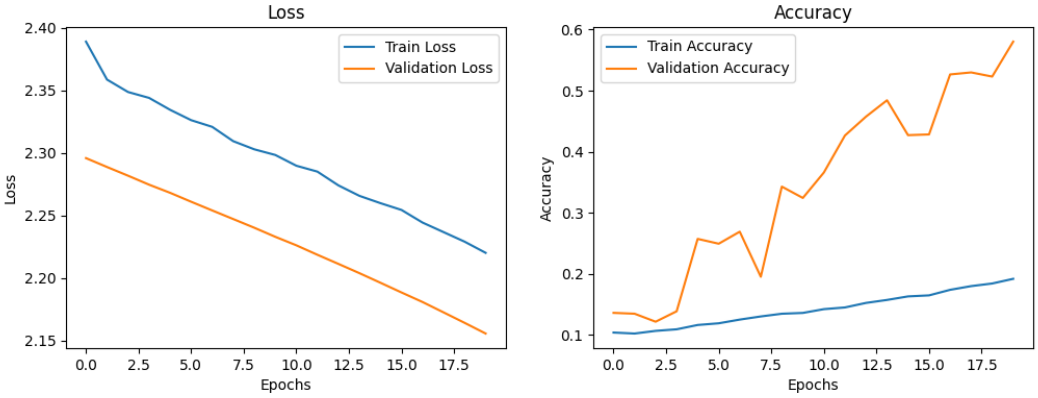
الگوریتم‌های استفاده شده در این مدل Adam، RMSprop و SGD هستند.

بنظر می‌رسد الگوریتم Adam بهتر بر روی مجموعه داده‌ای ما منطبق میشود. همچنین الگوریتم SGD بر طبق بقیه عوامل تنظیم شده کارایی پایین‌تری برای مدل داشت. بطوریکه حتی با تغییر عوامل دیگر نیز به دقت دیگر روش‌ها دست نمی‌یافت.

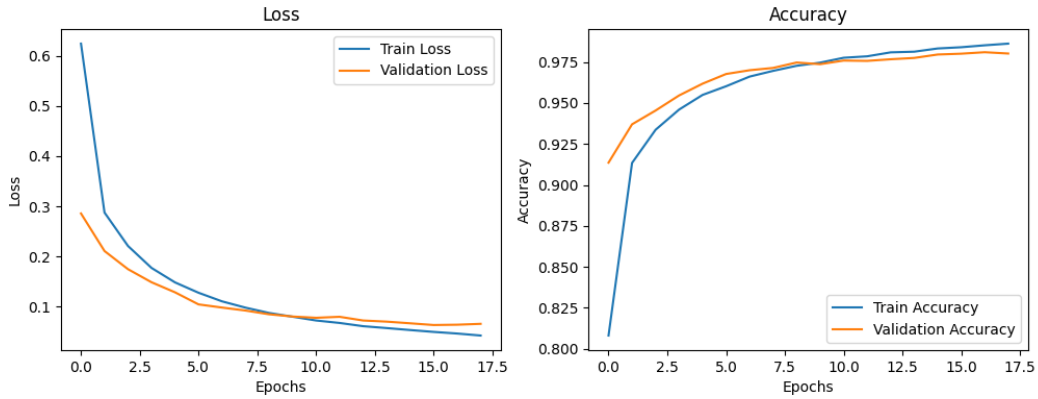
نمودارهای مربوط به هر الگوریتم در ادامه آمده است.



تصویر ۸ – الگوریتم Adam



تصویر ۹ – الگوریتم SGD



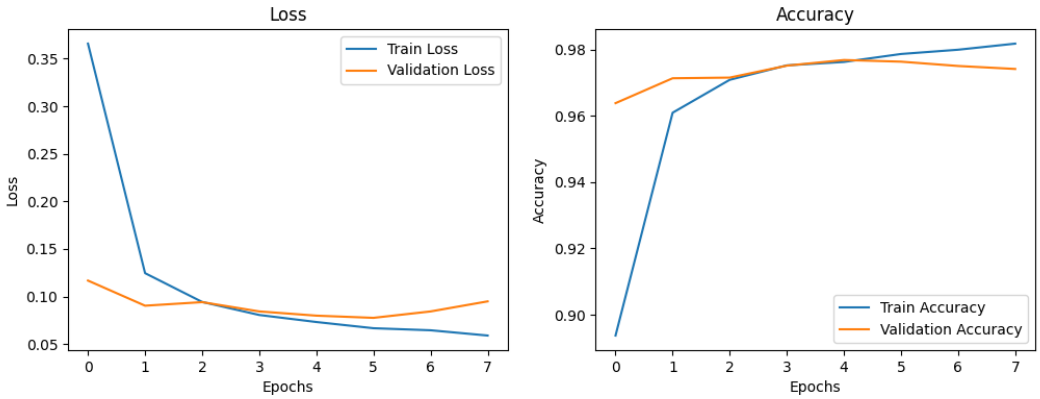
تصویر ۱۰ – الگوریتم RMSprop

نرخ یادگیری

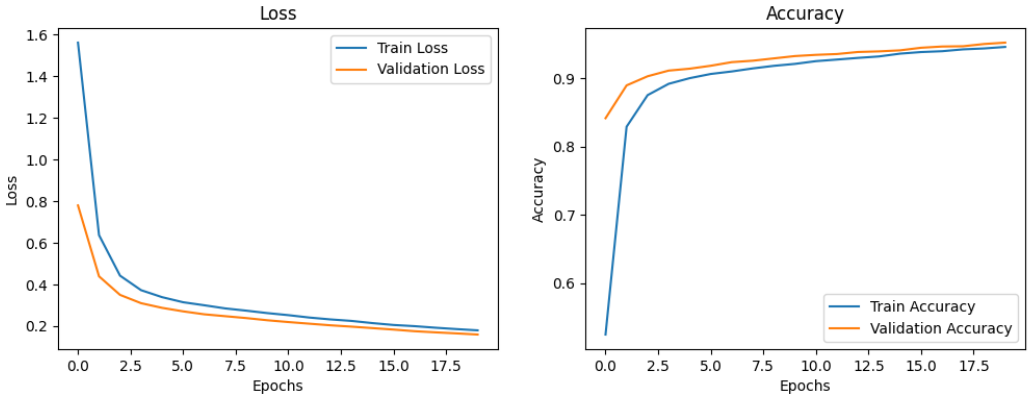
نرخ یادگیری، مقداری بین صفر و یک است.

با آزمودن مقادیر متفاوت برای نرخ یادگیری، نهایتا بهترین نتایج وقتی رخ داد که نرخ یادگیری چیزی نزدیک به ۰.۰۰۱ بود. دوباره با آزمودن اعداد مختلف این بازه عدد ۰.۰۰۰۹۵ مقادیر مورد انتظار از مدل ما را برآورده می‌ساخت.

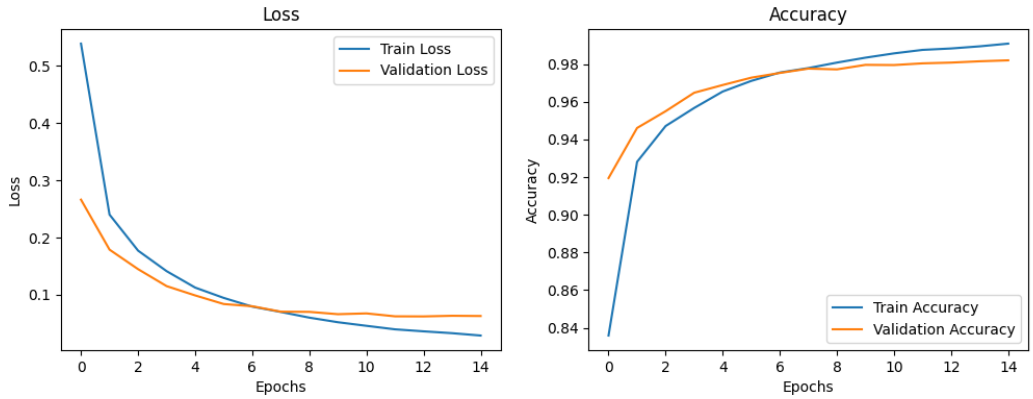
در ادامه نمودارهای مرتبط با مقادیر متفاوت نرخ یادگیری آمده است.



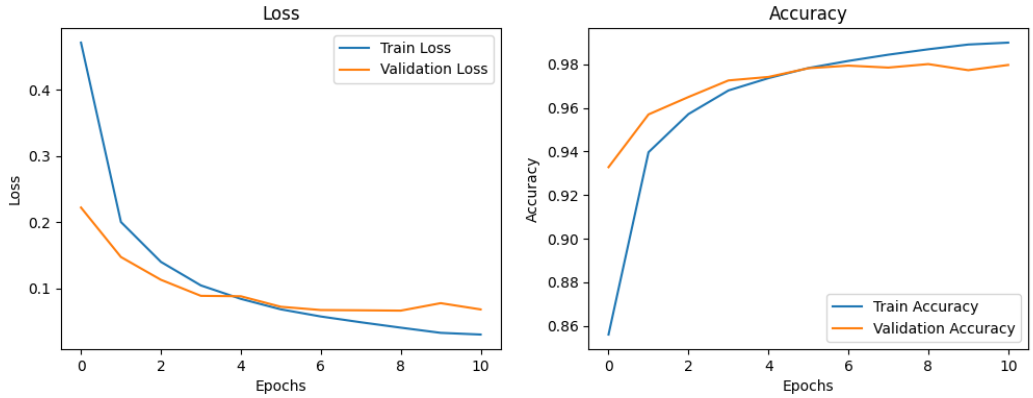
تصویر ۱۱ – نرخ یادگیری = ۰.۰۱



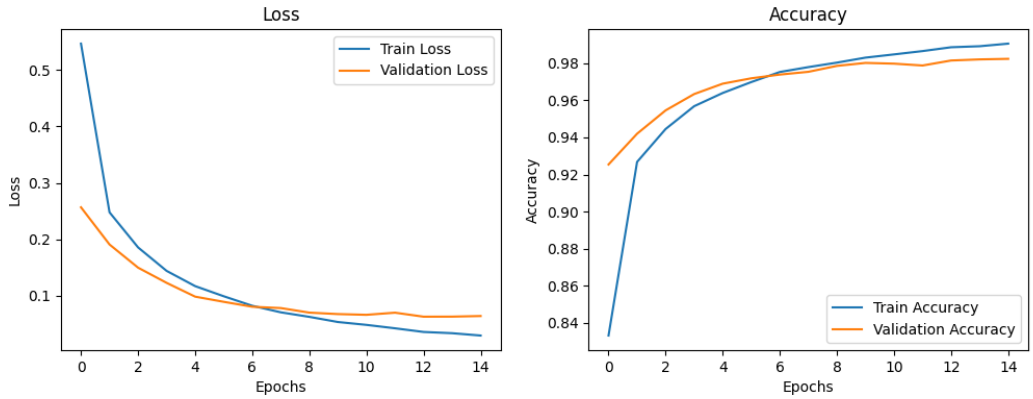
تصویر ۱۲ – نرخ یادگیری = ۰.۰۰۰۱



تصویر ۱۳ – نرخ یادگیری = ۰.۰۰۱



تصویر ۱۴ – نرخ یادگیری = ۰.۰۰۱۵



تصویر ۱۵ – نرخ یادگیری = ۰.۰۰۰۹۵

شرط توقف

در کد مدل ما، شرط توقف بطور خودکار بررسی می‌شود. بدین صورت که با استفاده از تابع EarlyStopping بررسی می‌شود که آیا با گذشت Epochهای مختلف، معیار خاصی تغییر می‌کند یا ثابت می‌ماند. در اینجا مقدار خاص مدنظر ما validation loss است. تابع هربار بررسی میکند که آیا این مقدار نسبت به مقدار قبلش کاهش یافته یا خیر. اگر پس از گذشت چند بار (در اینجا ۲) این مقدار به‌جای کاهش، افزایش یابد؛ مدل ما متوقف میشود.

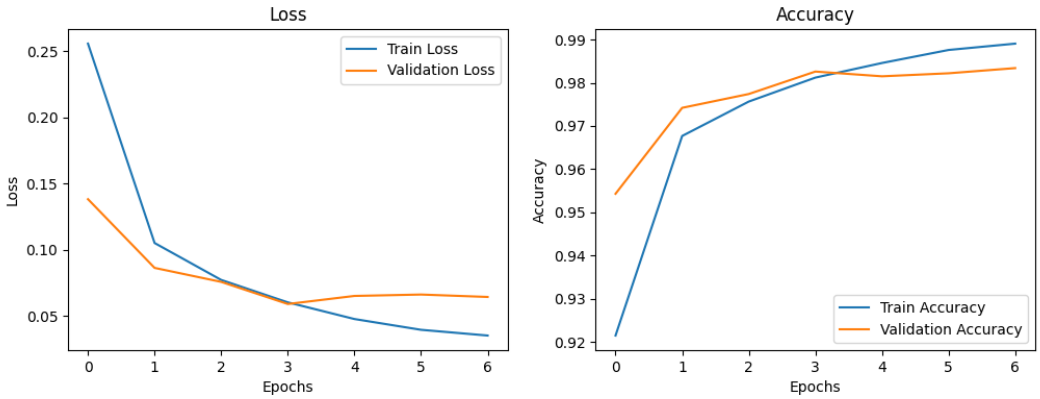
این کار کمک بسیار زیادی به مبحث overfitting و جلوگیری از اتفاق آن میکند.

توابع فعال‌سازی

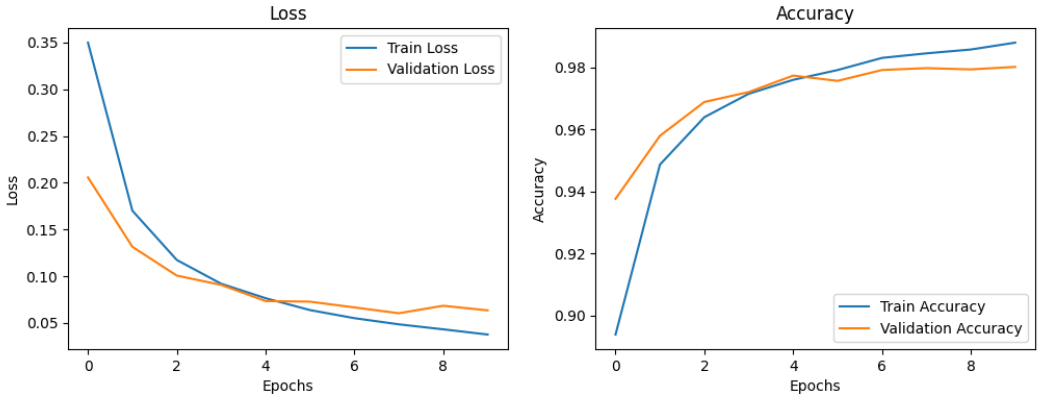
در اینجا ما از سه تابع فعال‌سازی relu، tanh و sigmoid برای آزمودن استفاده کردیم.

انطباق هر سه تابع با مدل ما خوب است، اما تابع سیگموید به طور واضح‌تری کارآتر و مناسب‌تر است.

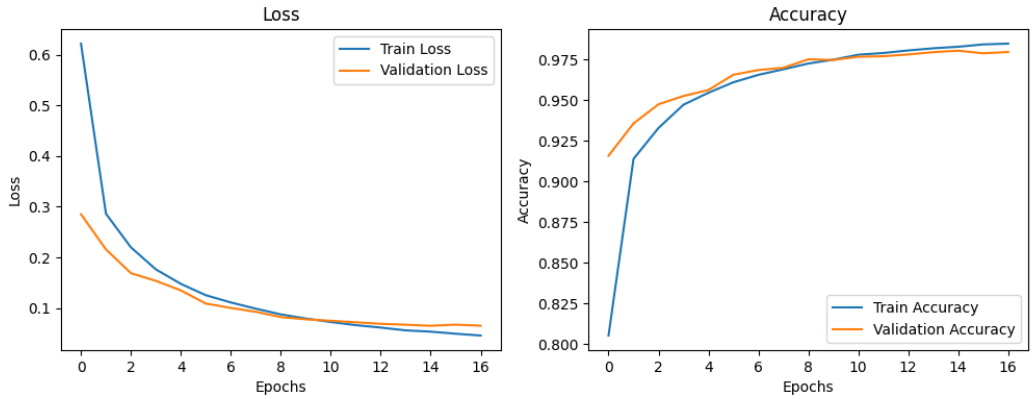
در ادامه نمودارهای مربوط به هر تابع رسم شده‌اند.



تصویر ۱۶ – تاثیر تابع فعال‌سازی relu



تصویر ۱۷ – تاثیر تابع فعال‌سازی tanh

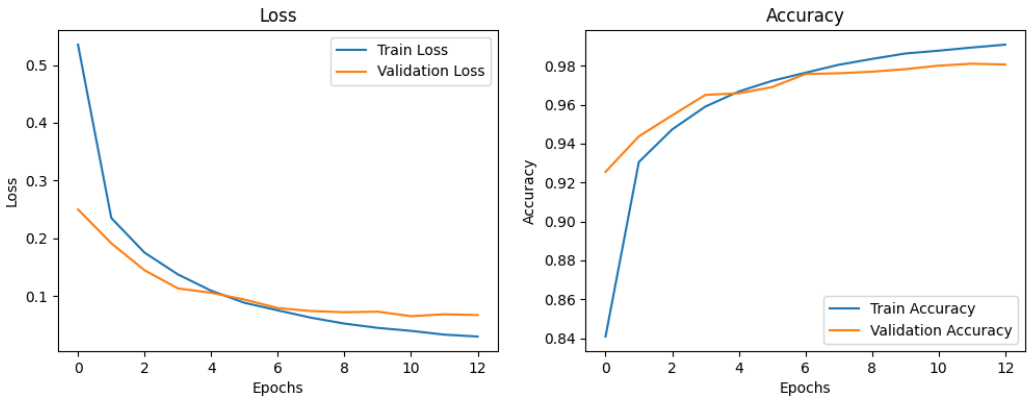


تصویر ۱۸ – تاثیر تابع فعال‌سازی sigmoid

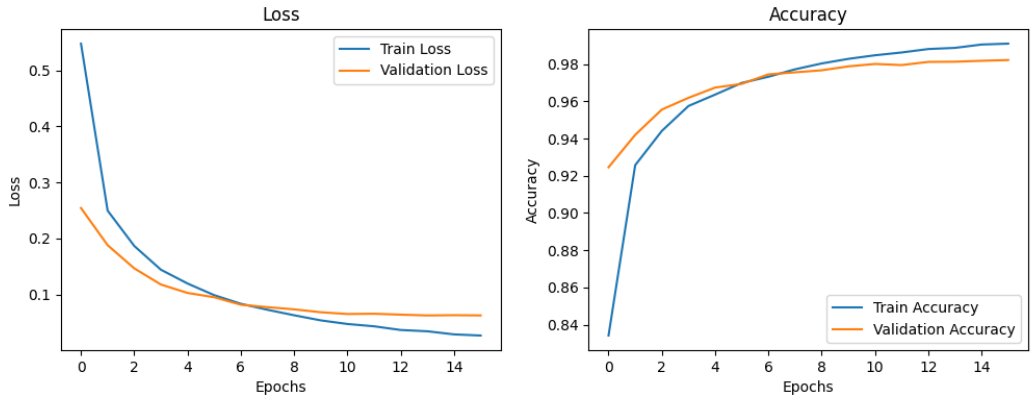
drop out

Dropout یک تکنیک منظم‌سازی است که برای جلوگیری از overfitting در شبکه‌های عصبی استفاده می‌شود. این تکنیک با حذف تصادفی بخشی از نورون‌ها در هر تکرار آموزشی عمل می‌کند. نرخ dropout، که معمولاً بین ۰.۲ و ۰.۵ است، نسبت نورون‌هایی را که حذف می‌شوند تعیین می‌کند. این فرایند باعث می‌شود شبکه ویژگی‌های مقاوم‌تری یاد بگیرد، زیرا نمی‌تواند به حضور نورون‌های خاصی وابسته باشد. در طول آموزش، dropout به ایجاد نوعی یادگیری جمعی کمک می‌کند و با آموزش دادن زیرشبکه‌های مختلف در داخل شبکه اصلی، این کار را انجام می‌دهد. در طول آزمون، dropout خاموش می‌شود و از کل شبکه استفاده می‌شود، اما خروجی به گونه‌ای مقیاس‌بندی می‌شود که نرخ dropout استفاده شده در طول آموزش را جبران کند. این تکنیک به کاهش overfitting و بهبود تعمیم‌دهی مدل کمک می‌کند.

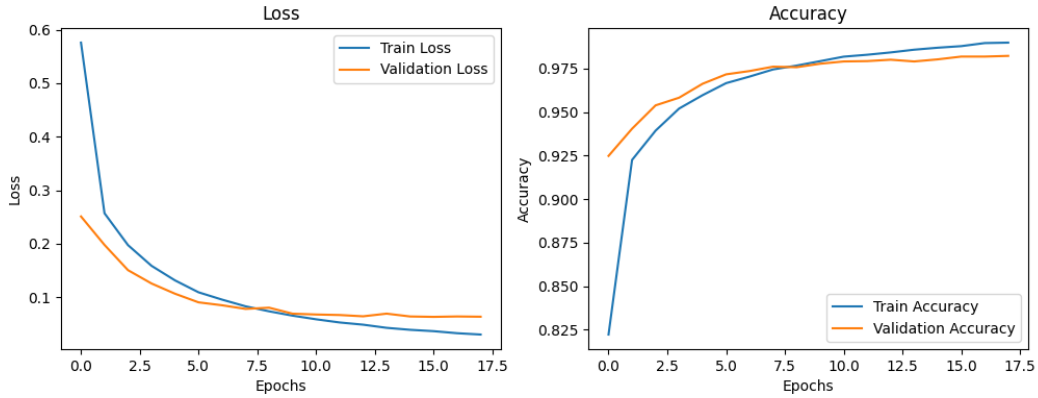
مقادیر مختلفی برای این عامل تست شد که در ادامه نمودارهای آن‌ها مشاهده می‌شود.



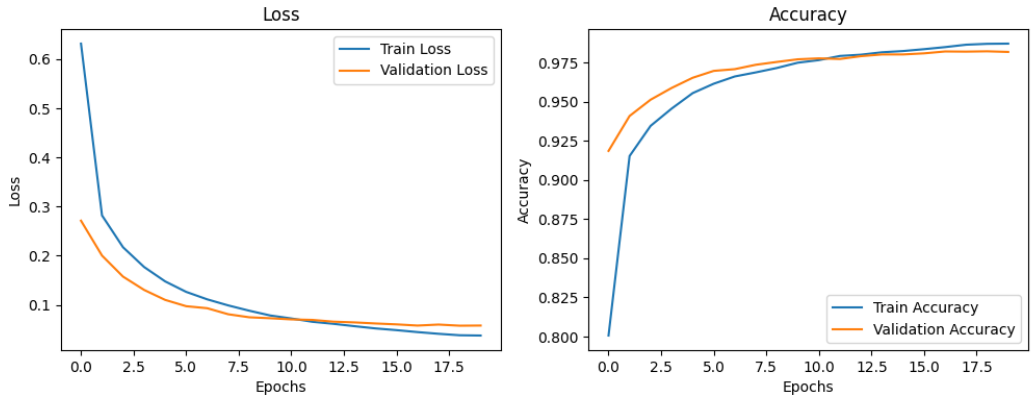
تصویر ۱۹ – نرخ dropout = ۰.۱



تصویر ۲۰ – نرخ dropout = ۰.۲



تصویر ۲۱ – نرخ dropout = ۰.۳

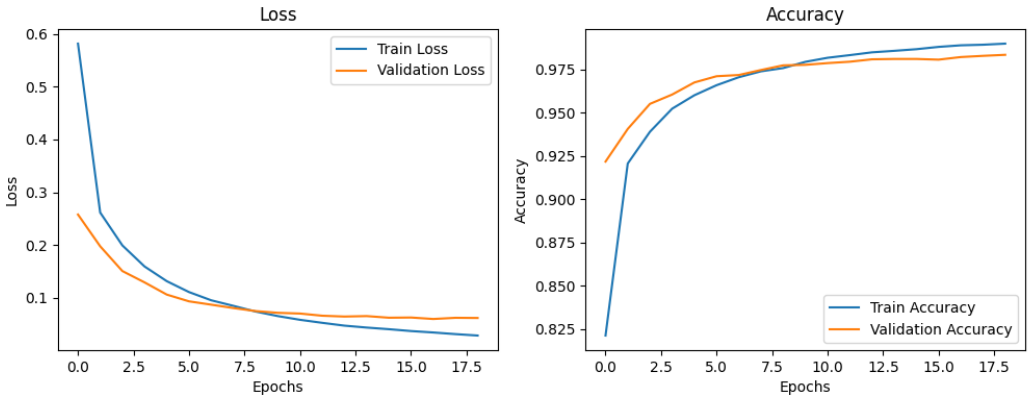


تصویر ۲۲ – نرخ dropout = ۰.۴

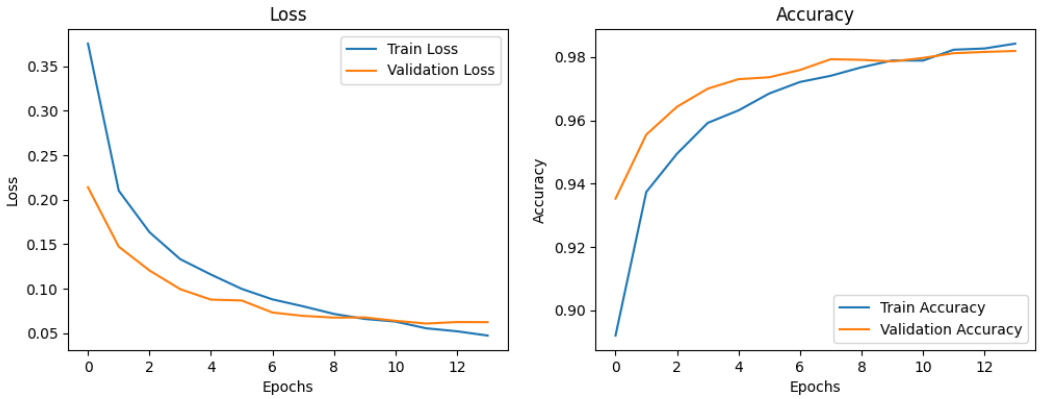
batch normalization

Batch Normalization یک تکنیک است که برای بهبود آموزش شبکه‌های عصبی عمیق با تثبیت و تسریع فرآیند یادگیری استفاده می‌شود. این تکنیک با نرمال‌سازی ورودی هر لایه به صورتی که میانگین صفر و واریانس یک داشته باشد، بر روی هر مینی‌بچ از داده‌ها عمل می‌کند. این نرمال‌سازی قبل از اعمال تابع فعال‌سازی انجام می‌شود و به کاهش تغییر داخلی کوواریانس کمک می‌کند؛ پدیده‌ای که در آن توزیع ورودی‌های هر لایه در طول آموزش تغییر می‌کند و می‌تواند فرآیند یادگیری را کند نماید. با اطمینان از اینکه ورودی‌های هر لایه توزیع ثابتی دارند، نرمال‌سازی دسته‌ای اجازه می‌دهد تا نرخ‌های یادگیری بالاتری استفاده شوند، حساسیت به مقداردهی اولیه کاهش یابد، و می‌تواند به عنوان شکلی از منظم‌سازی عمل کند که گاهی نیاز به dropout را حذف می‌کند.

مقداردهی های مختلف این عامل نیز در ادامه آمده است.



تصویر ۲۳ – batch normalization = false



تصویر ۲۴ – batch normalization = true

نتایج مدل و تنظیمات نهایی

تنظیمات نهایی مدل به شرح زیر است:

```
config = {
    'num_hidden_layers': 2,
    'neurons_per_layer': 500,
    'activation': 'sigmoid',
    'optimizer': 'Adam',
    'learning_rate': 0.00095,
    'dropout_rate': 0.3,
    'batch_norm': True
}
```

همچنین خروجی نهایی مدل با تنظیمات بالا به این شکل است:

Epoch 1/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.8479 - loss: 0.5406 - val_accuracy: 0.9411 - val_loss: 0.2028

Epoch 2/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9312 - loss: 0.2279 - val_accuracy: 0.9591 - val_loss: 0.1396

Epoch 3/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.9504 - loss: 0.1653 - val_accuracy: 0.9647 - val_loss: 0.1123

Epoch 4/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9580 - loss: 0.1383 - val_accuracy: 0.9711 - val_loss: 0.0941

Epoch 5/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.9637 - loss: 0.1131 - val_accuracy: 0.9726 - val_loss: 0.0880

Epoch 6/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9681 - loss: 0.1022 - val_accuracy: 0.9730 - val_loss: 0.0827

Epoch 7/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9720 - loss: 0.0857 - val_accuracy: 0.9767 - val_loss: 0.0760

Epoch 8/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9747 - loss: 0.0760 - val_accuracy: 0.9759 - val_loss: 0.0728

Epoch 9/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - accuracy: 0.9760 - loss: 0.0732 - val_accuracy: 0.9791 - val_loss: 0.0654

Epoch 10/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.9793 - loss: 0.0659 - val_accuracy: 0.9803 - val_loss: 0.0629

Epoch 11/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.9815 - loss: 0.0561 - val_accuracy: 0.9810 - val_loss: 0.0632

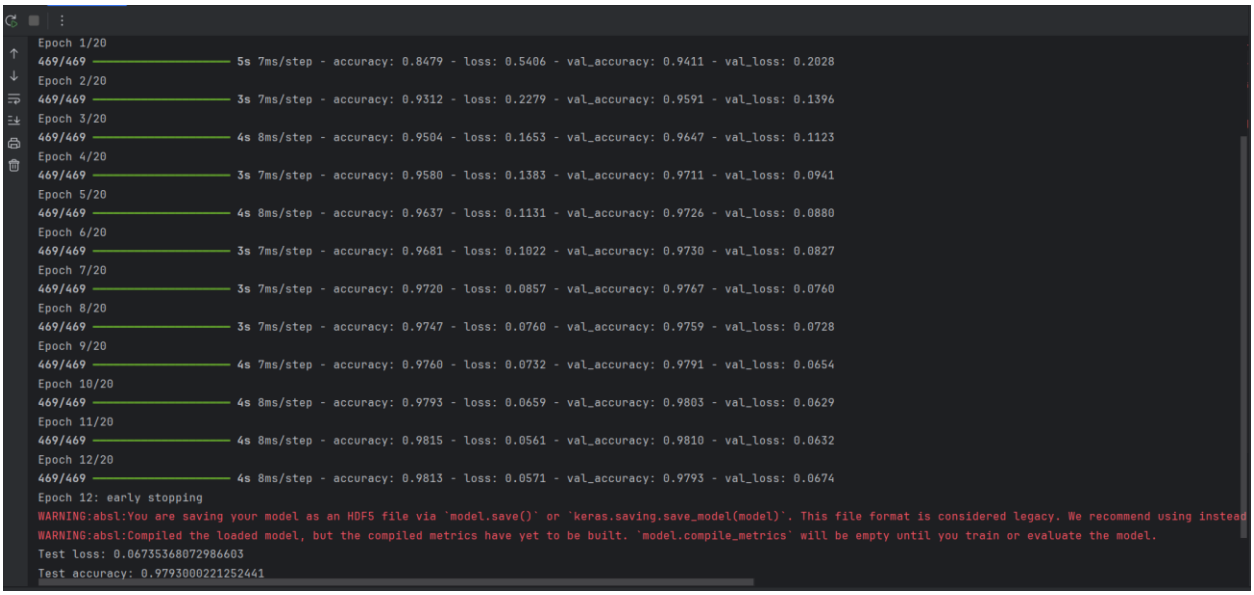
Epoch 12/20

469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.9813 - loss: 0.0571 - val_accuracy: 0.9793 - val_loss: 0.0674

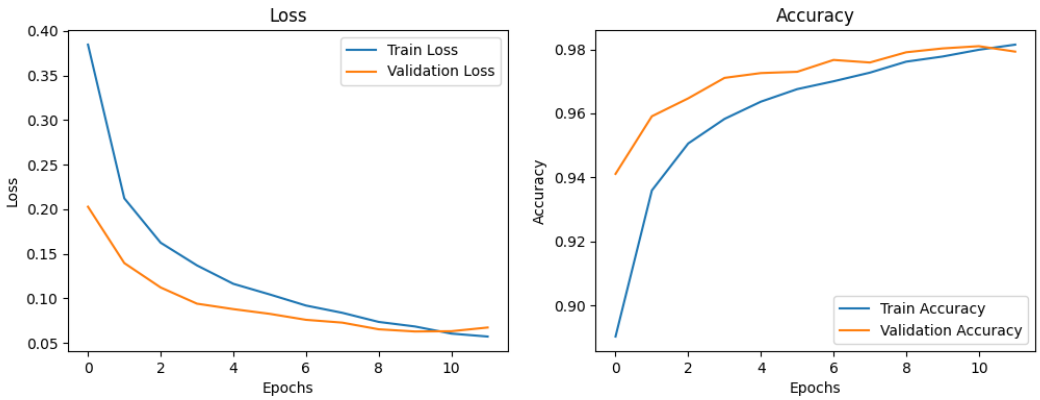
Epoch 12: early stopping

Test loss: 0.06735368072986603

Test accuracy: 0.9793000221252441



تصویر ۲۵ – خروجی کد



تصویر ۲۶ – نمودار مدل نهایی