به نام خدا

گزارش پروژه دوم – شبکه عصبی **نیایش خانی ۹۹۵۲۱۲۳۵**

این پروژه با جستجو درباره‌ی شبکه های عصبی و مدل های مختلفی که در این حوزه استفاده می شود، آغاز شد. پس از مطالعه‌ی این موضوعات، به بررسی نمونه‌هایی از پروژه‌های مشابه پرداختم.

چالش برانگیز ترین بخش این پروژه زمان بردن train و تست به دلیل زیاد بودن دیتا بود که با انجام هر تغییر کوچکی، باید مدت زمان زیادی صرف می شد تا پروژه دوباره ران شود و خروجی ها را ببینم.

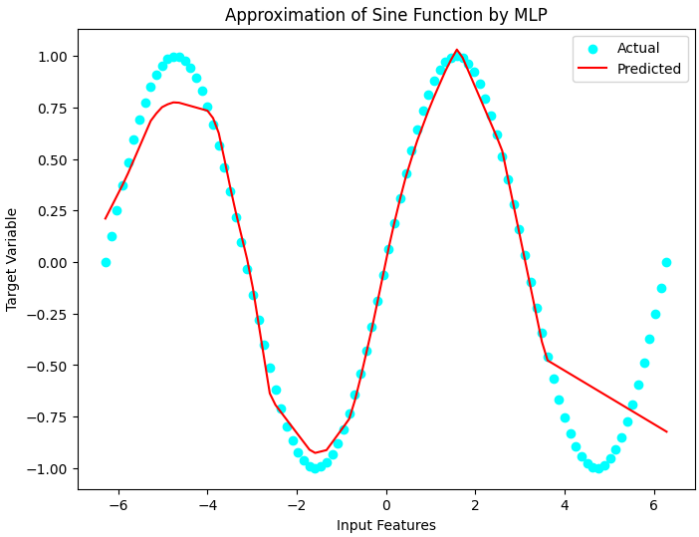
در ادامه به سراغ توضیحات هر بخش می‌رویم.

۱) بخش اول :

در این بخش ما یک مدل شبکه عصبی از نوع MLP برای تقریب تابع سینوسی ایجاد کرده‌ایم.
این کد به ترتیب مراحل زیر را انجام می‌دهد:

- ابتدا هزار نمونه داده برای ویژگی‌های ورودی از توزیع یکنواخت بین -۲*π و ۲*π تولید می‌کنیم.
- سپس مقدار مورد نظر را با استفاده از تابع سینوس محاسبه می‌کنیم و به ویژگی‌های ورودی و مقدار مورد نظر نویز اضافه می‌کنیم تا مسئله را کمی پیچیده‌تر کنیم.
- یک مدل MLP تعریف می‌کنیم که دو لایه پنهان دارد. لایه اول ۱۶ نورون و لایه دوم ۸ نورون.
- مدل را با تابع خطای میانگین مربعات و الگوریتم بهینه‌سازی Adam کامپایل کرده و بر روی داده‌های آموزش آموزش می‌دهیم.
- در آخر مدل آموزش دیده را بر روی داده‌های تست اعمال کرده و نتایج را به صورت نموداری نشان می‌دهیم.

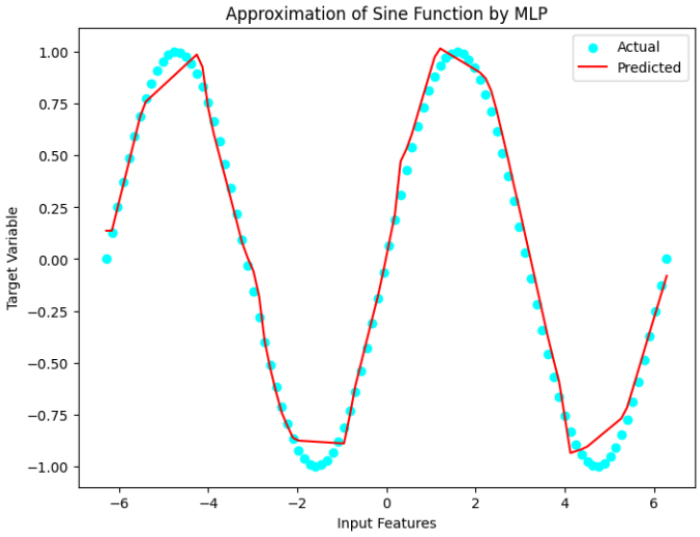
خروجی به صورت زیر خواهد بود :



حال با اعمال تغییراتی که در داکیومنت سوال آمده است، می‌توانیم تاثیر هر پارامتر را بر عملکرد مدل خود مشاهده کنیم.

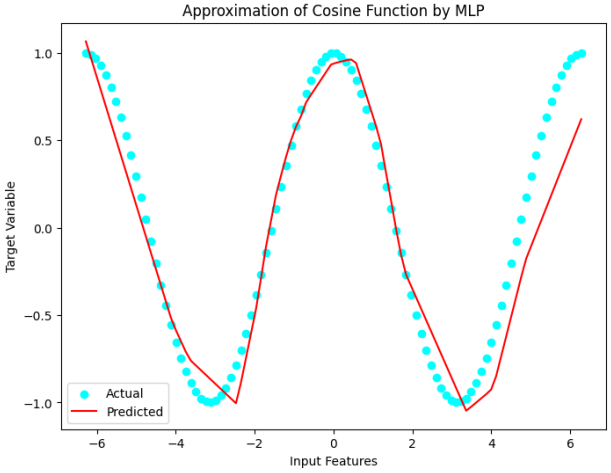
۱) خروجی با تغییر تعداد نقاط ورودی : افزایش تعداد نمونه ها می تواند توانایی مدل را برای یادگیری و تعمیم بهبود بخشد، اما هزینه و زمان محاسباتی را نیز افزایش می دهد.
در اینجا، به جای ۱۰۰۰ نمونه، ۵۰۰۰ نمونه تولید می کنیم.
۲) میزان پیچیدگی تابع مورد نظر که در ادامه بررسی می کنیم.
۳) تعداد لایه ها و نورون های هر لایه : تعداد نورون های لایه اول هیدن را افزایش می دهیم . این کار می‌تواند به مدل کمک کند تا الگوهای پیچیده‌تری را بیاموزد. با این حال، تعداد بیش از حد لایه ها یا نورون ها می تواند منجر به overfit شود، جایی که مدل داده های آموزشی را خیلی خوب یاد می گیرد و در داده های جدید ضعیف عمل می کند. همچنین هزینه محاسباتی را افزایش می دهد.
٤) افزایش تعداد چرخه های شبکه : دوره‌های بیشتر به مدل فرصت‌های بیشتری برای یادگیری از داده‌ها می‌دهد، اما دوره‌های بیش از حد می‌تواند منجر به تطبیق بیش از حد شود. همچنین، پس از یک نقطه خاص، دوره های اضافی عملکرد را بهبود نمی بخشد و فقط هزینه محاسباتی را افزایش می دهد. ما مدل را برای ۳۰۰ دوره با batch sizeکوچکتر - ۱۶ - آموزش خواهیم داد.
۵) در آخر دامنه ورودی را افزایش می دهیم و به جای ۱۰۰، ۲۰۰ نقطه داده آزمایشی ایجاد می‌کنیم.

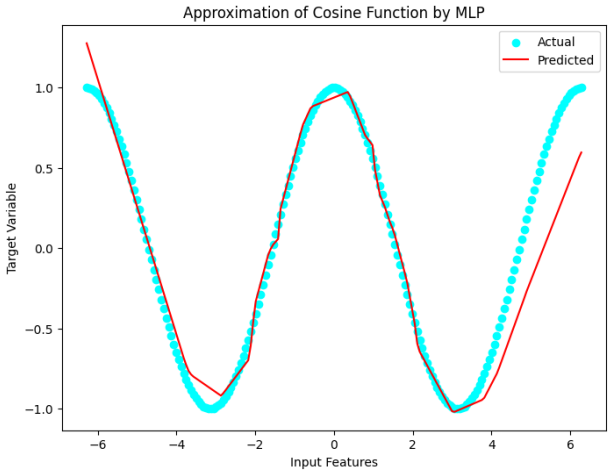
همانطور که مشاهده می شود، دقت بیشتر شده است و خطای بسیار کمی میان تابع اصلی و تابع train شده وجود دارد.



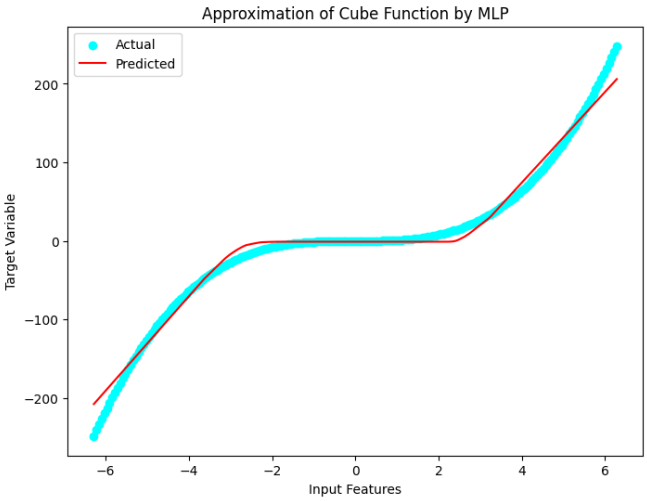
تابع بعدی کسینوس است که همان مراحل قبل را بر روی آن اجرا می‌کنیم و خروجی های زیر را دریافت می‌کنیم.

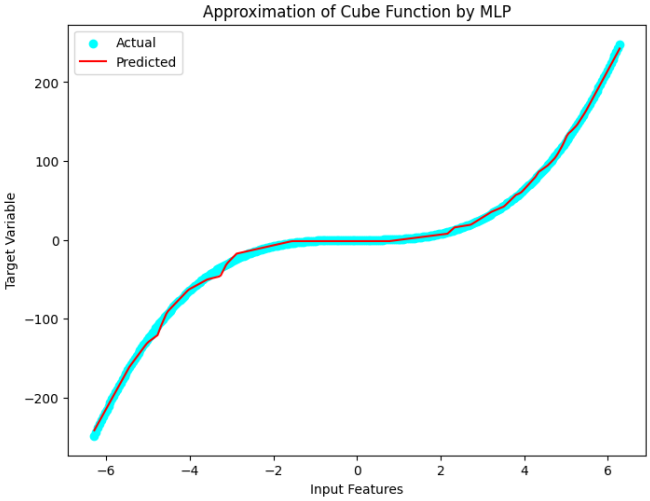
خروجی اول برای داده های اولیه و خروجی دوم بعد از اعمال تغییرات روی پارامتر های مدل است.





سپس تابع x^3 را امتحان می کنیم. همانطور که مشاهده می‌شود ابتدا خروجی مناسبی دریافت نمی‌شود اما با تغییر پارامترها به نتیجه بهتری می‌رسیم.





بخش دوم : در کد ارائه شده، یک خط قرمز را در یک تصویر تشخیص می دهیم که همان تصویر موجود در داک پروژه است و با استفاده از رگرسیون چند جمله ای یک منحنی بر روی آن قرار می دهیم. سپس، سعی می کنیم از یک مدل پرسپترون چند لایه (MLP) برای تقریب منحنی استفاده کنیم.

اگر نموداری که ترسیم می‌کنیم جهش‌های ناگهانی داشته باشد، مانند شکل L، ممکن است نشان دهد که رگرسیون چند جمله‌ای مدل مناسبی برای داده‌ها نیست. رگرسیون چند جمله ای سعی می کند یک منحنی چند جمله ای را با داده ها منطبق کند، که ممکن است تغییرات ناگهانی یا ناپیوستگی ها را به خوبی ثبت نکند.

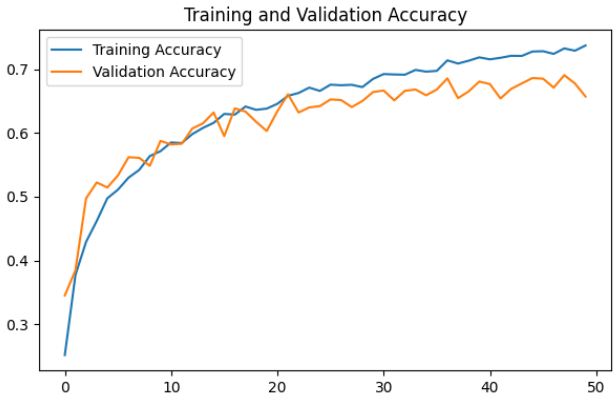
۲) دسته بندی تصاویر :

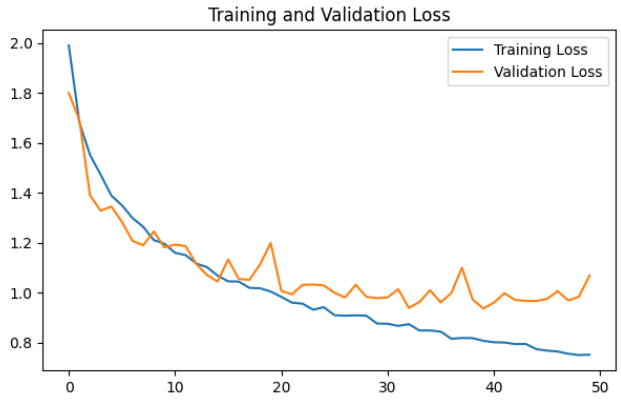
این کد برای آموزش یک مدل شبکه‌ی عصبی کانولوشنال (CNN) بر روی مجموعه داده CIFAR-10 به منظور دسته‌بندی تصاویر استفاده می‌شود. داده‌های آموزش و آزمون ابتدا بارگذاری می‌شوند، سپس پیش‌پردازش مانند نرمال‌سازی و تبدیل برچسب‌ها انجام می‌شود. سپس مدل CNN با لایه‌های کانولوشنال، گیری حداکثر، و لایه‌های کاملا متصل تعریف می‌شود. مدل سپس کامپایل و با استفاده از داده‌های آموزشی آموزش داده می‌شود. برای افزایش داده و جلوگیری از بیش‌برازش از تکنیک‌های افزایش داده استفاده می‌شود. در نهایت، عملکرد مدل بر روی داده‌های آزمون ارزیابی شده و نمودارهایی از عملکرد مدل و نمونه‌هایی از تصاویر آزمون و پیش‌بینی‌های مدل نمایش داده می‌شود.

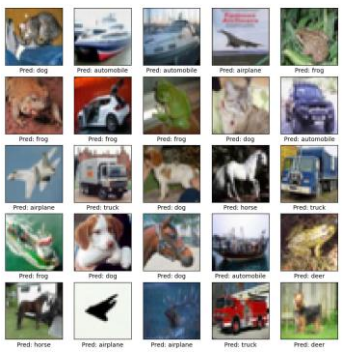
در این کد از نموداری برای نمایش تغییرات دقت و loss بر روی داده های آموزشی در هر epoch استفاده شده است. همچنین تکنیک های generation image نیز پیاده سازی شده است.

نمونه ای از خروجی :

```
Epoch 48/50
313/313 [==============================] - 16s 53ms/step - loss: 0.7552 - accuracy: 0.7324 - val_loss: 0.9684 - val_accuracy: 0.6905
Epoch 49/50
313/313 [==============================] - 16s 52ms/step - loss: 0.7499 - accuracy: 0.7288 - val_loss: 0.9835 - val_accuracy: 0.6775
Epoch 50/50
313/313 [==============================] - 18s 57ms/step - loss: 0.7515 - accuracy: 0.7370 - val_loss: 1.0680 - val_accuracy: 0.6570
```







۳) تشخیص چهره با استفاده از MLP :

ابتدا تصاویر از پوشه‌ای بارگذاری می‌شوند و برچسب‌های مربوط به آن‌ها از یک فایل CSV خوانده می‌شوند. این فایل CSV شامل نام تصاویر و برچسب مربوط به آن‌ها است.

برای کاهش ابعاد ویژگی‌های تصاویر و افزایش سرعت آموزش، از روش کاهش ابعاد PCA استفاده می‌شود. این کار باعث می‌شود تعداد ویژگی‌های ورودی به مدل کاهش یابد و در نتیجه مدل سریع‌تر آموزش داده شود.

سپس یک مدل شبکه‌ی عصبی چند لایه‌ای (MLP) با استفاده از کتابخانه scikit-learn تعریف و آموزش داده می‌شود. این مدل قادر به یادگیری الگوهای پیچیده‌تری از تصاویر است.

پس از آموزش مدل، این مدل بر روی داده‌های آزمون پیش‌بینی می‌شود و دقت آن بر اساس برچسب‌های واقعی محاسبه می‌شود. این کار به ما اطلاع می‌دهد که مدل به چه اندازه به درستی برچسب‌های تصاویر را پیش‌بینی کرده است.