سوال ۱ :

بخش اول:

۱ .تعریف توابع هدف: در این بخش، سه تابع هدف تعریف شده است:
- تابع هدف ۱: تابع سینوس با استفاده از np.sin(x)
- تابع هدف ۲: تابع مربعی با استفاده از x**2 + 2*x + 1
- تابع هدف ۳: تابع جذر مربعی با استفاده از np.sqrt(np.abs(x))

۲ .تولید داده‌های آموزش برای هر تابع هدف: با استفاده از توابع هدف تعریف شده، داده‌های آموزش با استفاده از تابع linspace و تابع های هدف ایجاد می‌شود. همچنین به داده‌های آموزش نویز گوسی با استفاده از np.random.normal اضافه می‌شود.

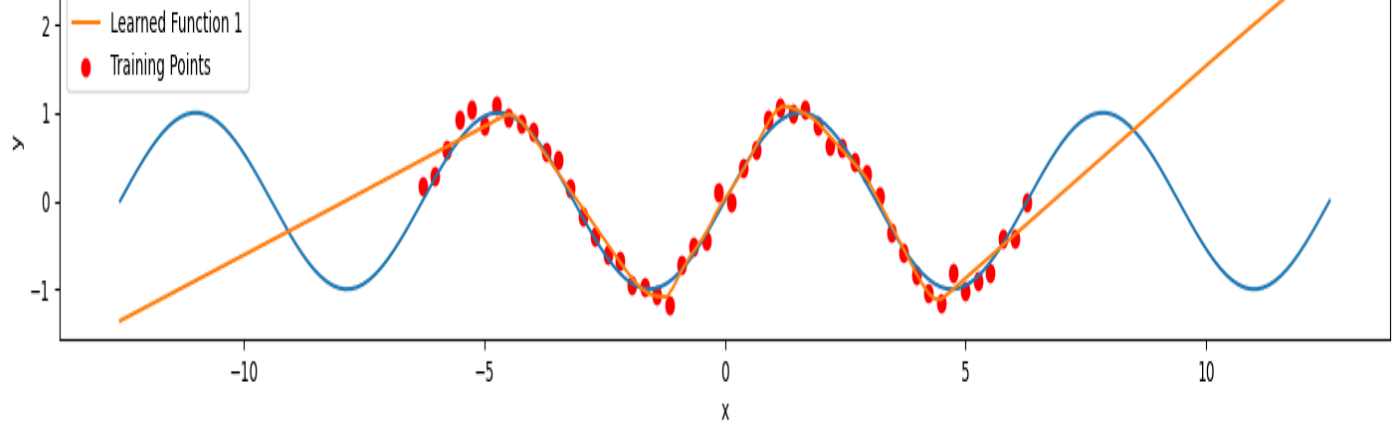
۳ .ایجاد و آموزش MLP برای هر تابع هدف: در این بخش، سه MLPRegressor با ساختار (hidden_layer_sizes=(100,), max_iter=10000)ایجاد می‌شود و روی داده‌های آموزش آموزش داده می‌شوند.

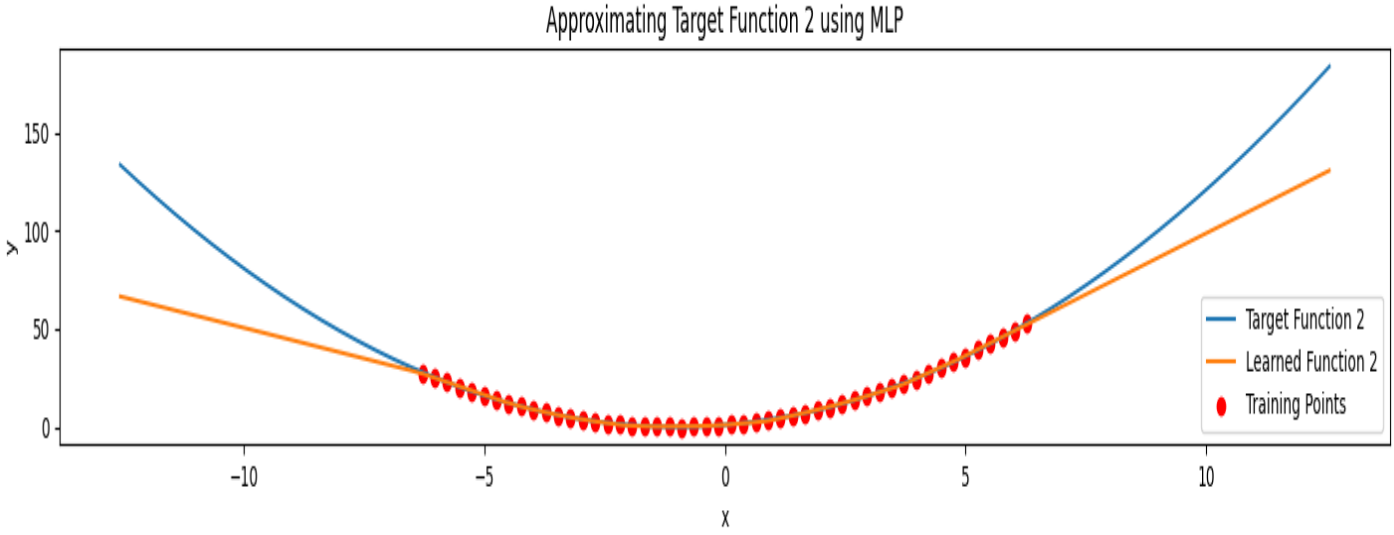
۴ .تولید داده‌های آزمون برای هر تابع هدف: داده‌های آزمون با استفاده از تابع linspace و توابع هدف تعریف شده در بازه‌ی گسترده‌تری نسبت به داده‌های آموزش ایجاد می‌شوند.

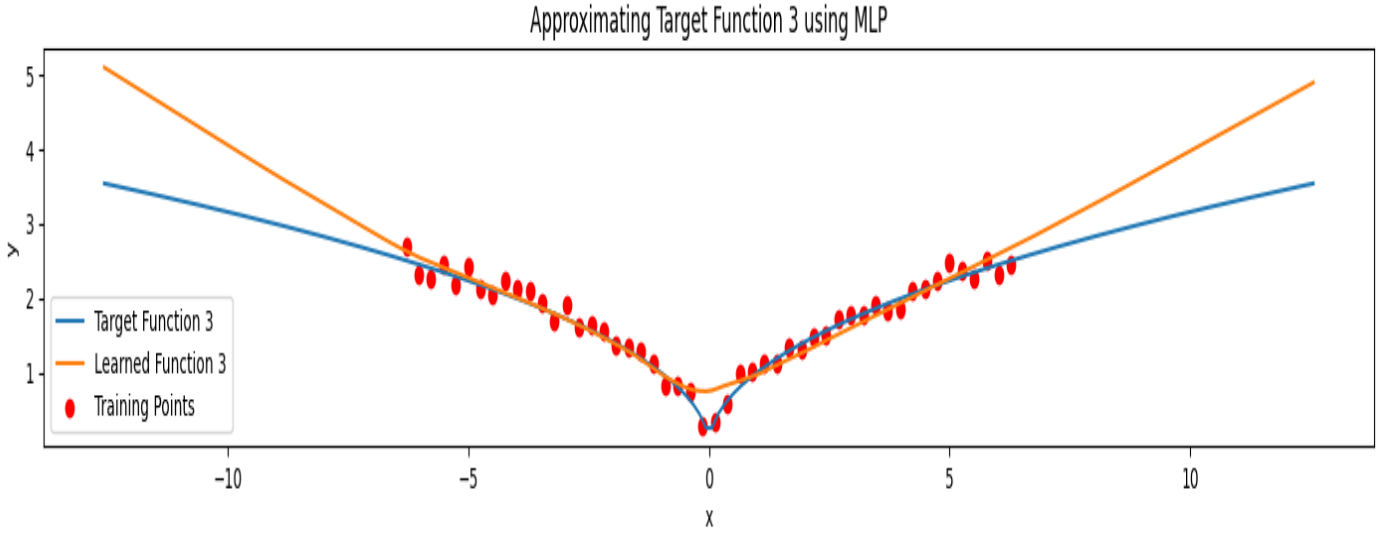
۵ .پیش‌بینی با استفاده از MLP آموزش‌دیده برای هر تابع هدف: برای هر تابع هدف، با استفاده از MLP آموزش‌دیده، مقادیر پیش‌بینی شده برای داده‌های آزمون محاسبه می‌شود.

۶ .محاسبه و چاپ خطای میانگین مربعات برای هر تابع هدف: برای هر تابع هدف، با استفاده از مقادیر پیش‌بینی و مقادیر واقعی، خطای میانگین مربعات محاسبه می‌شود و چاپ می‌شود.

۷ .نمایش توابع هدف و توابع یادگرفته شده: با استفاده از کتابخانهmatplotlib ، توابع هدف و توابع یادگرفته شده برای هر تابع هدف رسم می‌شوند. همچنین نقاط آموزش نیز با استفاده از scatter plot نمایش داده می‌شوند.
در نهایت، نمودارهای حاوی توابع هدف و توابع یادگرفته شده رسم می‌شوند.

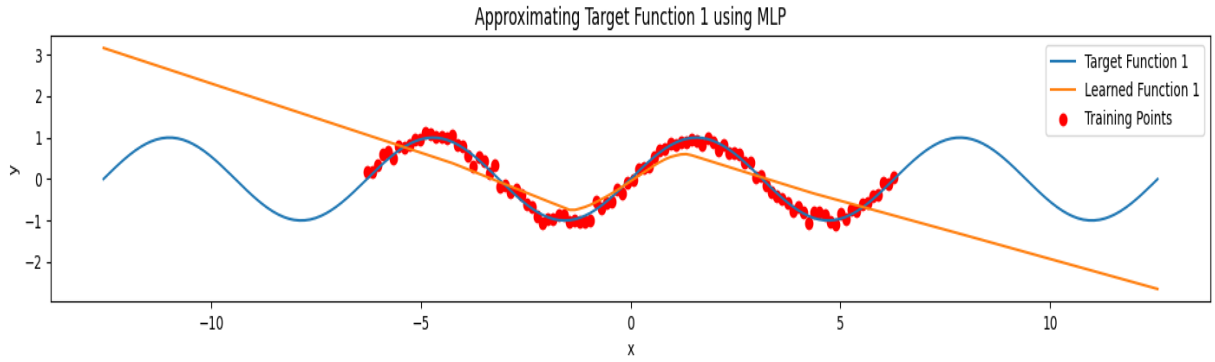

Approximating Target Function 1 using MLP
Target Function 1
Learned Function 1
Training Points
y
x
2
1
0
−1
−10
−5
0
5
10

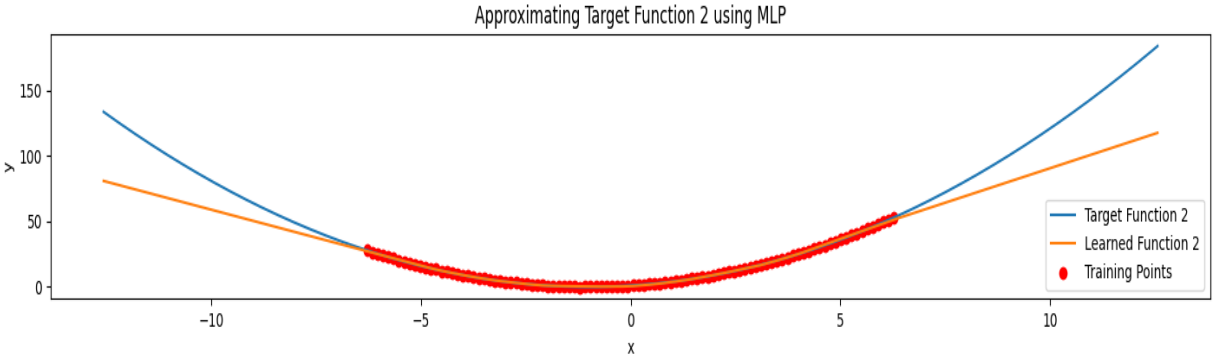

Approximating Target Function 2 using MLP
Target Function 2
Learned Function 2
Training Points
y
x
150
100
50
0
−10
−5
0
5
10

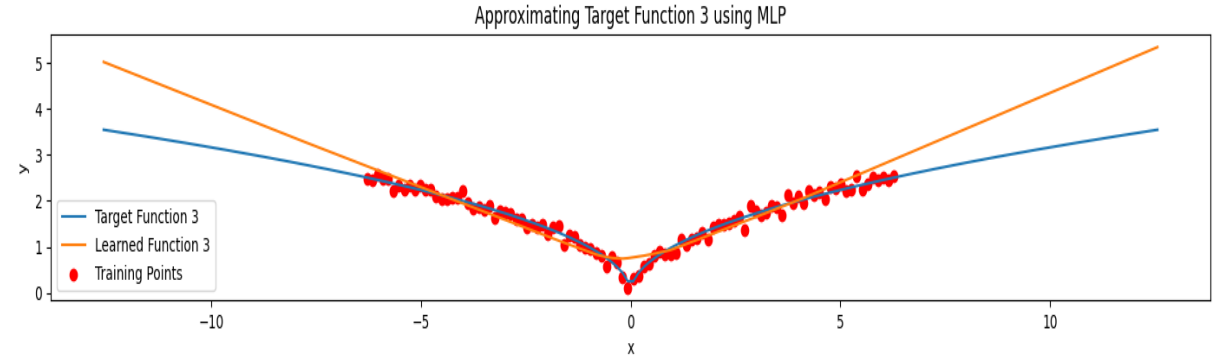

Approximating Target Function 3 using MLP
Target Function 3
Learned Function 3
Training Points
y
x
5
4
3
2
1
−10
−5
0
5
10

تغییر تعداد نقاط آموزشی:

۱ .دقت در پیش‌بینی: با افزایش تعداد نقاط آموزشی، شبکه عصبی بیشتری از الگوها و روابط در داده‌ها یاد می‌گیرد. این به معنی دقت بیشتر در پیش‌بینی مقادیر برای داده‌های آموزشی و آزمون است. با داشتن نقاط آموزشی بیشتر، شبکه عصبی می‌تواند توانایی خود را در تقریب و تعمیم بهتر از داده‌های جدید نشان دهد.
۲ .عملکرد در داده‌های جدید: با افزایش تعداد نقاط آموزشی، شبکه عصبی بهتر می‌تواند الگوها و روابط موجود در داده‌ها را یاد بگیرد و بهترین تقریب را برای تابع هدف ارائه دهد. این به معنی عملکرد بهتر در داده‌های جدید، به ویژه در بازه‌هایی که در داده‌های آموزشی وجود ندارند، است.
۳. انتقال یادگیری: با افزایش تعداد نقاط آموزشی، شبکه عصبی می‌تواند الگوها و روابطی که در داده‌های آموزشی یاد گرفته است را بهتر به داده‌های مشابه دیگر منتقل کند. این به معنی توانایی بهتر در انتقال یادگیری به مسائل مشابه و مشابه‌تر به داده‌های جدید است.







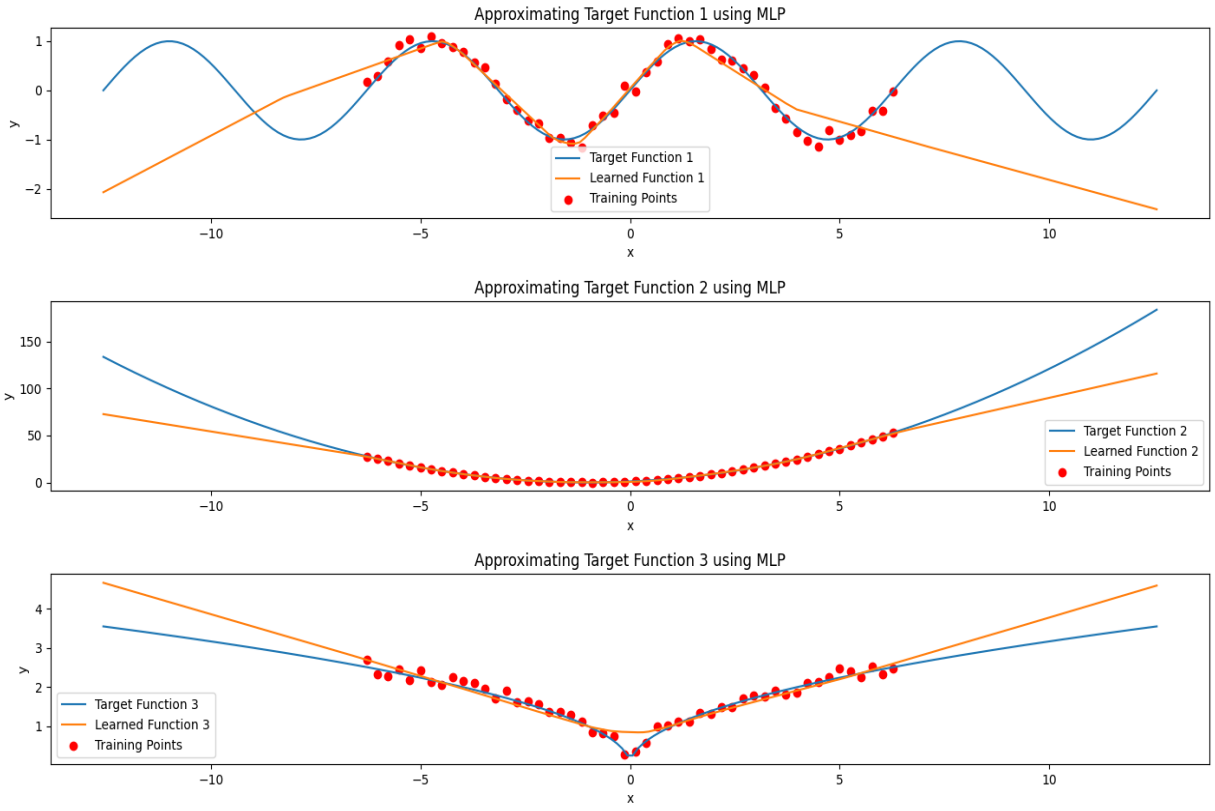
تغییر تعداد نورون ها:

افزایش تعداد نورون‌ها:

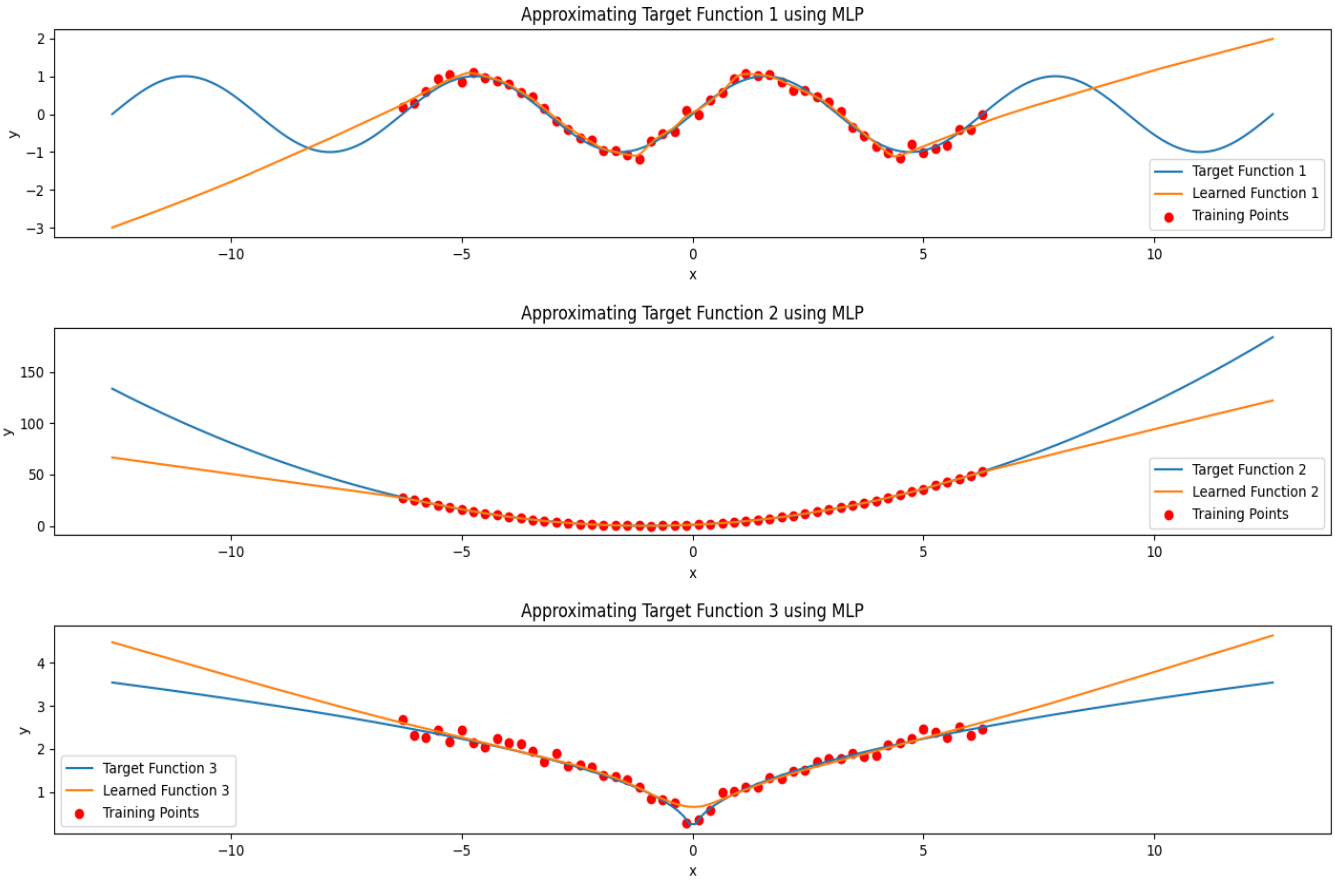
افزایش تعداد نورون‌ها معمولاً منجر به افزایش قدرت تقریبی شبکه می‌شود، به این معنی که شبکه قادر است توابع پیچیده‌تری را یاد بگیرد.
با افزایش تعداد نورون‌ها، شبکه قادر است اطلاعات بیشتری را در لایه‌های مخفی نگه دارد و ویژگی‌های پیچیده‌تری را استخراج کند.
افزایش تعداد نورون‌ها می‌تواند منجر به افزایش انعطاف‌پذیری شبکه در مدل‌سازی روابط پیچیده شود.
کاهش تعداد نورون‌ها:

کاهش تعداد نورون‌ها معمولاً منجر به ساده‌تر شدن مدل می‌شود. این می‌تواند به کاهش پیچیدگی مدل و جلوگیری از بیش‌برازش (overfitting) کمک کند.
با کاهش تعداد نورون‌ها، ممکن است مدل قادر به تقریب دادن توابع پیچیده‌تر نباشد و از برخی اطلاعات جزئی تراکم برداری کند.
کاهش تعداد نورون‌ها می‌تواند زمان آموزش مدل را کاهش دهد، زیرا تعداد پارامترهای قابل آموزش در شبکه کاهش می‌یابد.

کاهش تعداد نورون ها :

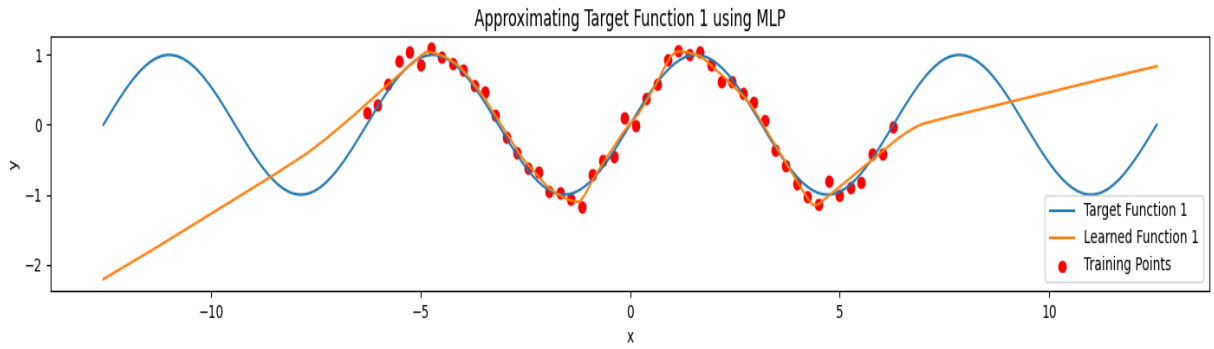


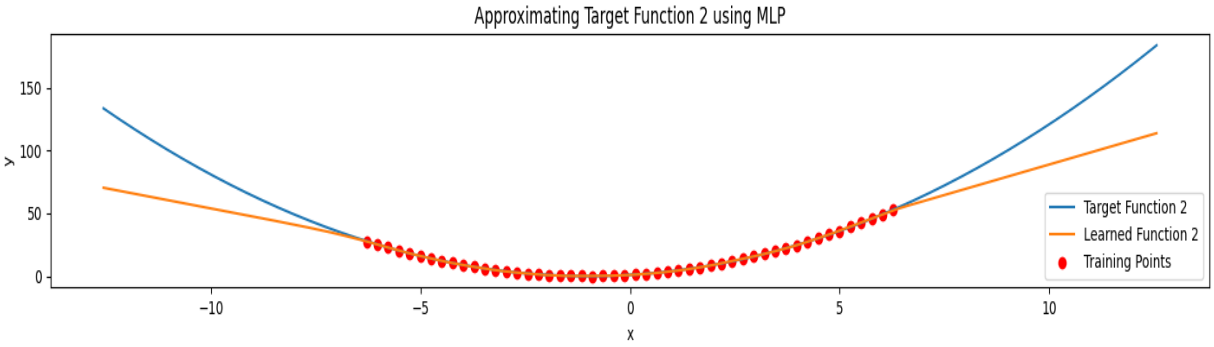
افزایش تعداد نورون ها :

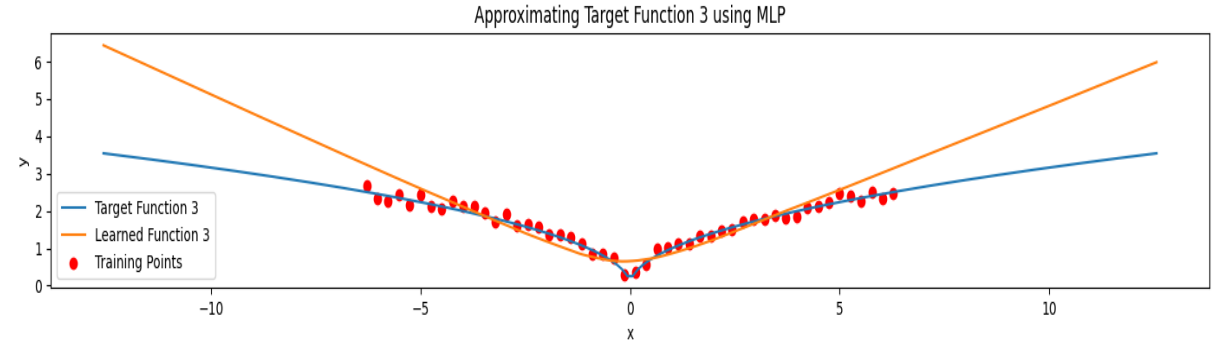


تغییر تعداد چرخه‌های آموزش شبکه

در صورت افزایش تعداد چرخه‌های آموزش، مدل بیشتر از داده‌های آموزش یاد می‌گیرد و می‌تواند جزئیات ریزتر را به خود بیاموزد. این می‌تواند باعث افزایش دقت و دقت پیش‌بینی مدل در داده‌های آموزش شود. با این حال، در صورتی که تعداد چرخه‌های آموزش بیش از حد باشد، ممکن است به بیش‌برازش (overfitting) منجر شود و مدل توانایی خوبی در پیش‌بینی داده‌های جدید نداشته باشد. از طرف دیگر، در صورت کاهش تعداد چرخه‌های آموزش، ممکن است مدل عملکرد کمتری داشته باشد و نتواند به شکل کامل از اطلاعات موجود در داده‌های آموزش استفاده کند. این می‌تواند منجر به کاهش دقت و دقت پیش‌بینی مدل در داده‌های آموزش شود.







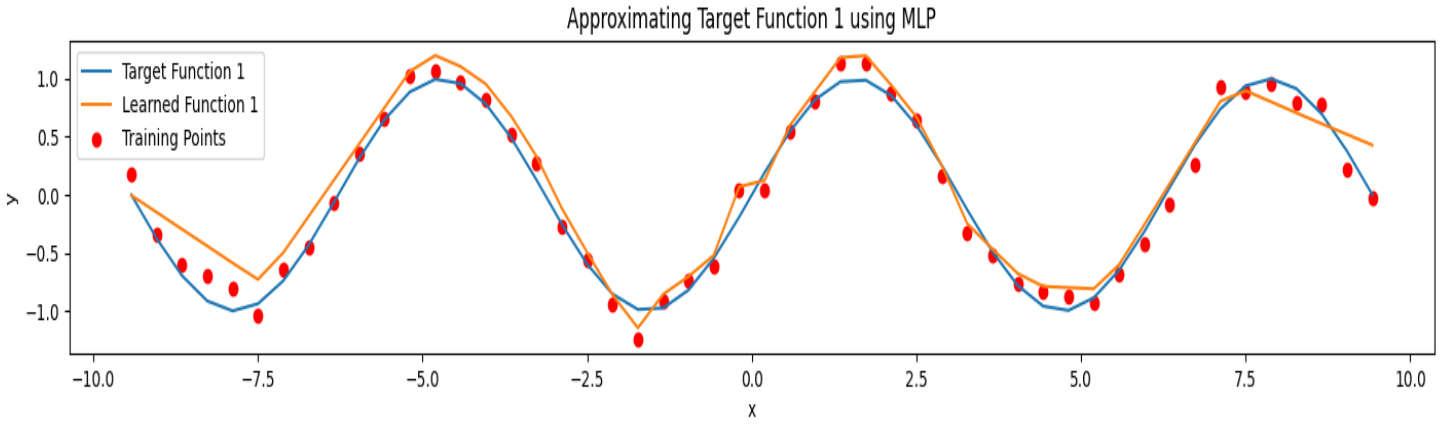
تغییرمحدوده ورودی در بخش تولید داده‌های آموزشی (x_train) و آزمون (x_test):

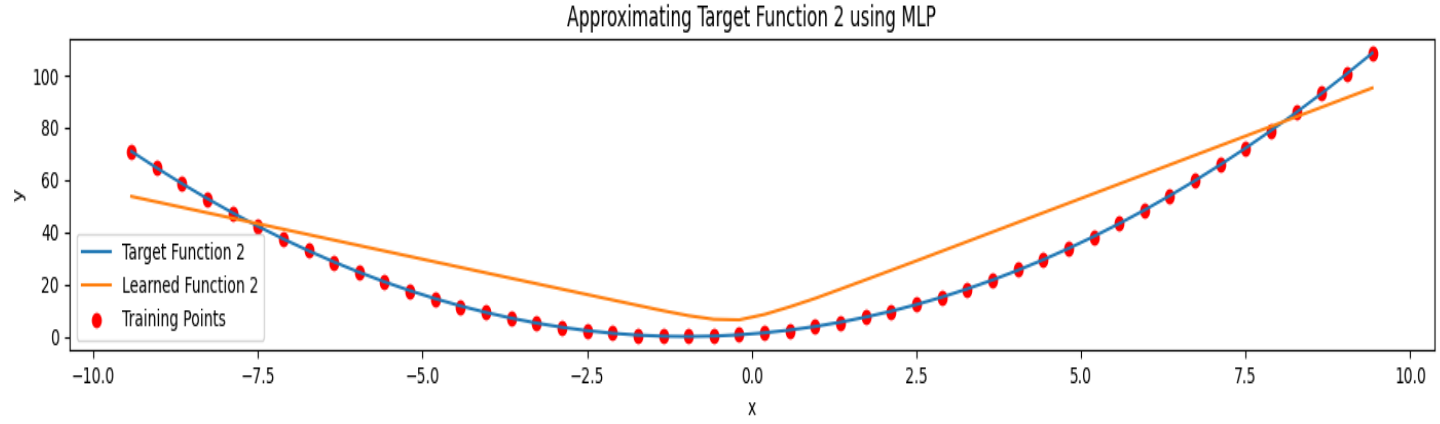
پوشش داده‌ها: با تغییر محدوده ورودی، می‌توانید پوشش داده‌ها را تغییر دهید و دامنه گسترده‌تری از ورودی‌ها را بررسی کنید. این می‌تواند به مدل کمک کند تا الگوهای موجود در داده‌ها را بهتر یاد بگیرد و مدلی انعطاف‌پذیرتر و قادر به پیش‌بینی بهتر در مقابل داده‌های جدید باشد.

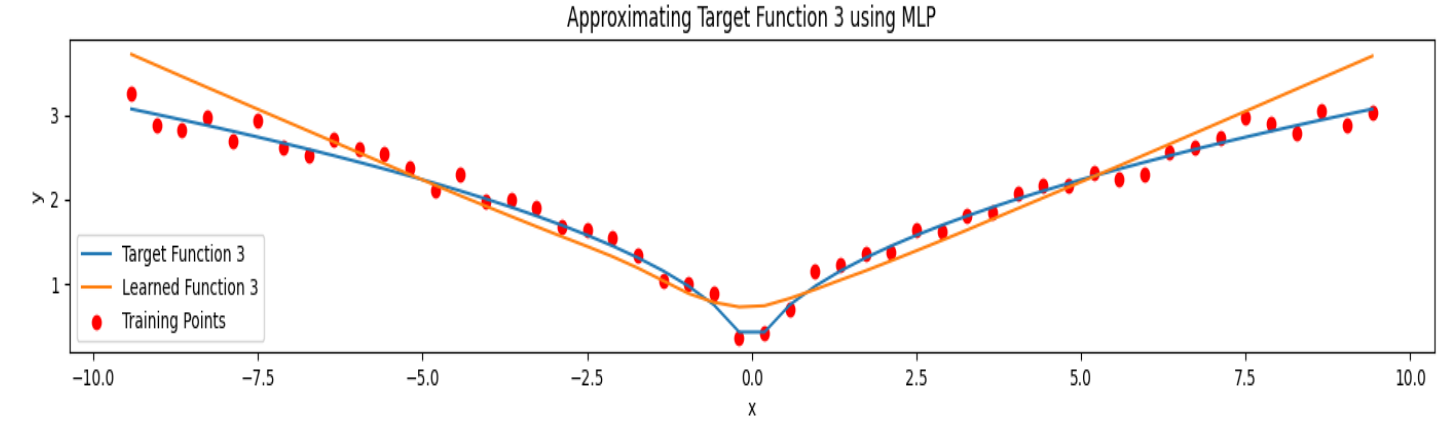
تعمیم‌پذیری: با تغییر محدوده ورودی، می‌توانید مدل را برای تعمیم‌پذیری بهتر آموزش دهید. با ارائه داده‌های آموزشی در محدوده‌های مختلف، مدل می‌تواند بهتر تطبیق پیدا کند و قادر به پیش‌بینی داده‌ها در خارج از محدوده آموزش شود.

بیش‌برازش (Overfitting) یا کاهش‌برازش (Underfitting): تغییر محدوده ورودی می‌تواند تأثیری بر روی بیش‌برازش یا کاهش‌برازش مدل داشته باشد. در صورتی که محدوده ورودی آموزشی بسیار محدود

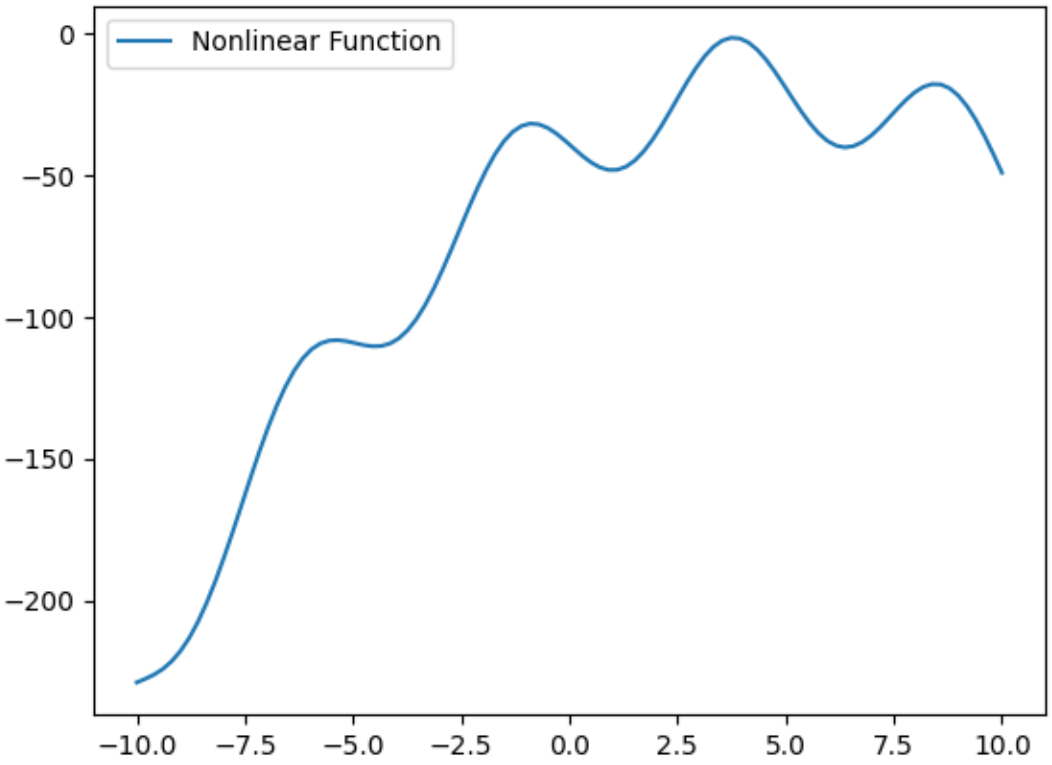
باشد، ممکن است مدل بیش‌برازش کند و در داده‌های جدید نتواند به خوبی عمل کند. از طرف دیگر، اگر محدوده ورودی آموزشی بسیار گسترده باشد، مدل ممکن است کاهش‌برازش داشته باشد و عملکرد ضعیفی در داده‌های جدید داشته باشد. بنابراین، تعیین محدوده ورودی مناسب می‌تواند به کنترل بیش‌برازش یا کاهش‌برازش کمک کند.







بخش دوم :



.

مجموعه ای از مقادیر ورودی x را با استفاده از np.linspace ایجاد می کنیم تا محدوده 10- تا 10 را پوشش دهیم.

مقادیر خروجی اصلی y_original با اعمال تابع_غیرخطی به مقادیر ورودی محاسبه می شود.

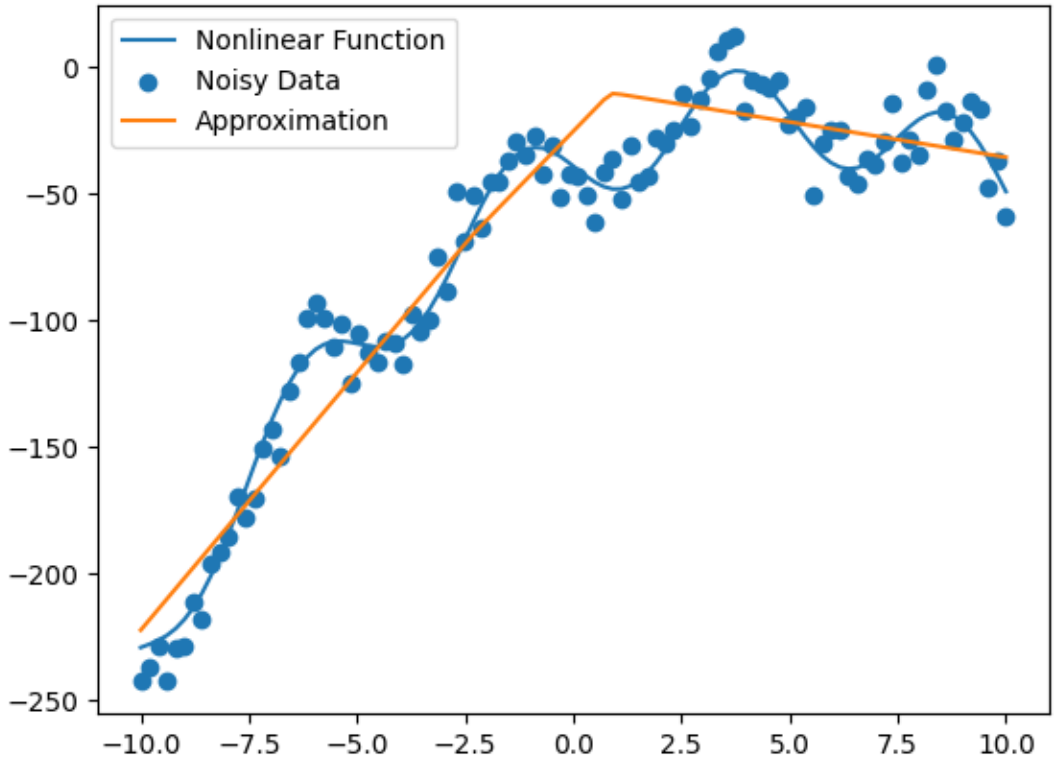
برای شبیه سازی داده های دنیای واقعی مقداری نویز تصادفی به مقادیر خروجی اصلی اضافه می کنیم. مقادیر خروجی نویز در y_train ذخیره می شوند. یک مدل شبکه عصبی ساده ایجاد می کنیم.

این مدل با استفاده از مقادیر ورودی x و مقادیر خروجی نویزدار y_train برای 1000 دوره آموزش داده شده است.

پس از آموزش، از مدل آموزش داده شده برای پیش بینی مقادیر خروجی برای مقادیر ورودی x استفاده می کنیم که نتیجه آن y_approx است.

در نهایت، تابع غیرخطی اصلی و تقریب به دست آمده از شبکه عصبی را رسم می کنیم.

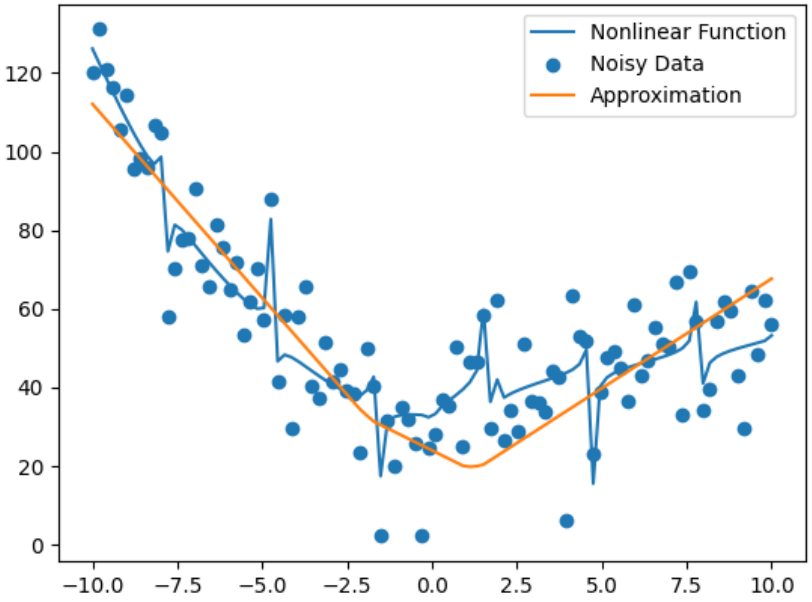
وقتی که بدون نویز آموزش داده می‌شود، مدل عصبی تلاش می‌کند تابع غیرخطی را با دقت بیشتری تقریب بزند. در نتیجه، تقریب حاصل از مدل نزدیک‌تر به تابع اصلی خواهد بود.



اگر نمودار ترسیم شده دارای نقاط پرش ناگهانی باشد، مدل عصبی با تعداد محدود نورون و ساختار ساده ممکن است نتواند این پرش‌های ناگهانی را به درستی تقریب بزند.

در این حالت، اگر نمودار ترسیم شده دارای نقاط پرش ناگهانی باشد، مدل عصبی با تعداد محدود نورون و ساختار ساده ممکن است نتواند این پرش‌های ناگهانی را به درستی تقریب بزند و تفاوت‌های زیادی بین تابع اصلی و تقریبی ایجاد شود.

دلیل این اتفاق این است که مدل‌های ساده با تعداد کم نورون و ساختار ساده معمولاً قدرت تقریب‌زنی محدودتری دارند و نمی‌توانند تغییرات ناگهانی را به درستی تقریب بزنند. زمانی که تابع اصلی دارای پرش‌های ناگهانی است، باید از مدل‌هایی با ساختار پیچیده‌تر و تعداد بیشتر نورون استفاده کنید تا بتوانند این تغییرات ناگهانی را به درستی تقریب بزنند.


Nonlinear Function
Noisy Data
Approximation
120
100
80
60
40
20
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

سوال ۲ :

2 .مجموعه داده CIFAR-10 با استفاده از تابع () keras.datasets.cifar10.load_data بارگذاری می شود. دو تاپل را برمی‌گرداند(x_train_full, y_train_full) :، (x_test, y_test) ، که به ترتیب نشان‌دهنده داده‌های آموزش و آزمایش هستند.

3 .مقادیر پیکسل تصاویر با تقسیم آنها بر 255.0 بین 0 و 1 نرمال می شوند.

4 .داده ها با استفاده از تابع train_test_split به مجموعه های آموزشی، اعتبار سنجی و آزمایش تقسیم می شوند. مجموعه آموزشی شامل 60٪ از داده های آموزشی اصلی، مجموعه اعتبارسنجی شامل 20٪ و مجموعه تست شامل 20٪ باقی مانده است.

5 .برچسب های کلاس برای مجموعه داده CIFAR-10 به عنوان یک لیست تعریف می شوند.

6 .معماری مدل با استفاده از Sequential API از Keras تعریف شده است. این شامل سه جفت لایه Conv2Dو MaxPooling2D برای استخراج ویژگی است، به دنبال آن یک لایه Flatten برای تبدیل نقشه های ویژگی دو بعدی به یک بردار 1 بعدی و دو لایه متراکم برای طبقه بندی.

7 .مدل با بهینه سازAdam ، تابع تلفات متقابل آنتروپی طبقه بندی شده پراکنده و متریک دقت گردآوری شده است.

8 .مدل بر روی داده های آموزشی برای 10 دوره آموزش داده شده است، با استفاده از داده های اعتبارسنجی برای اعتبار سنجی در طول آموزش. تاریخچه آموزش در متغیر history ذخیره می شود.

9 .مدل بر روی داده های آموزش، اعتبار سنجی و آزمایش با استفاده از روش ارزیابی ارزیابی می شود که ضرر و دقت را برمی گرداند.

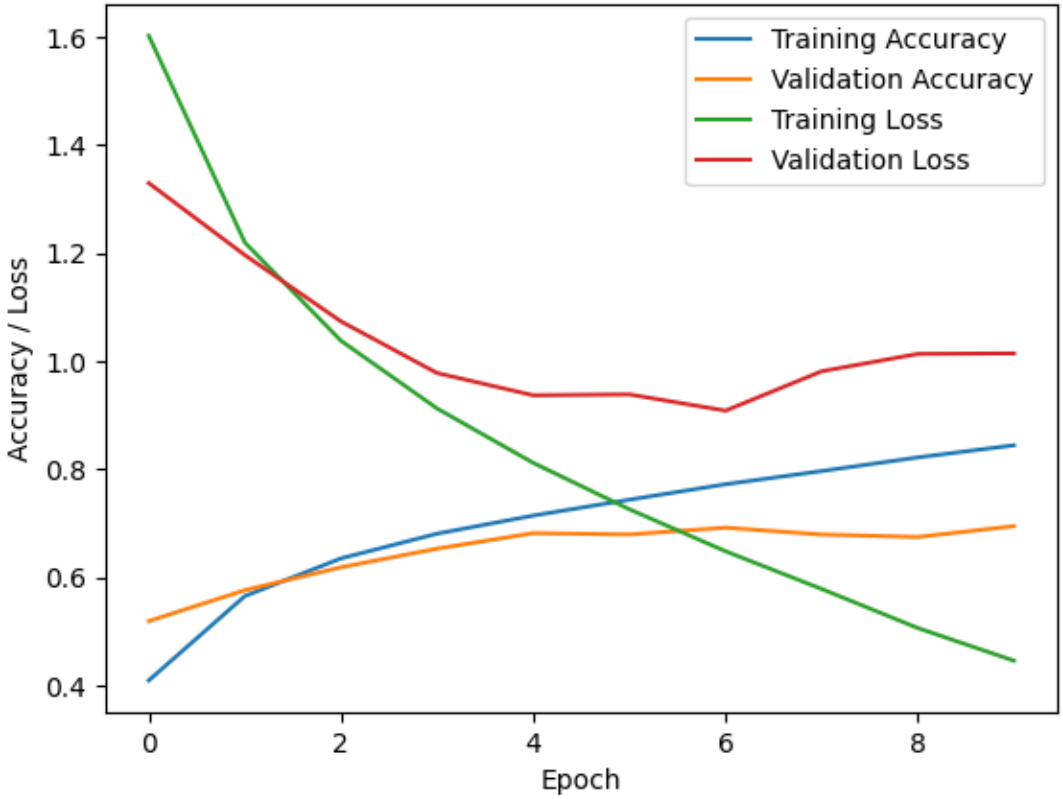
10 .دقت آموزش، صحت اعتبارسنجی و دقت تست چاپ شده است.

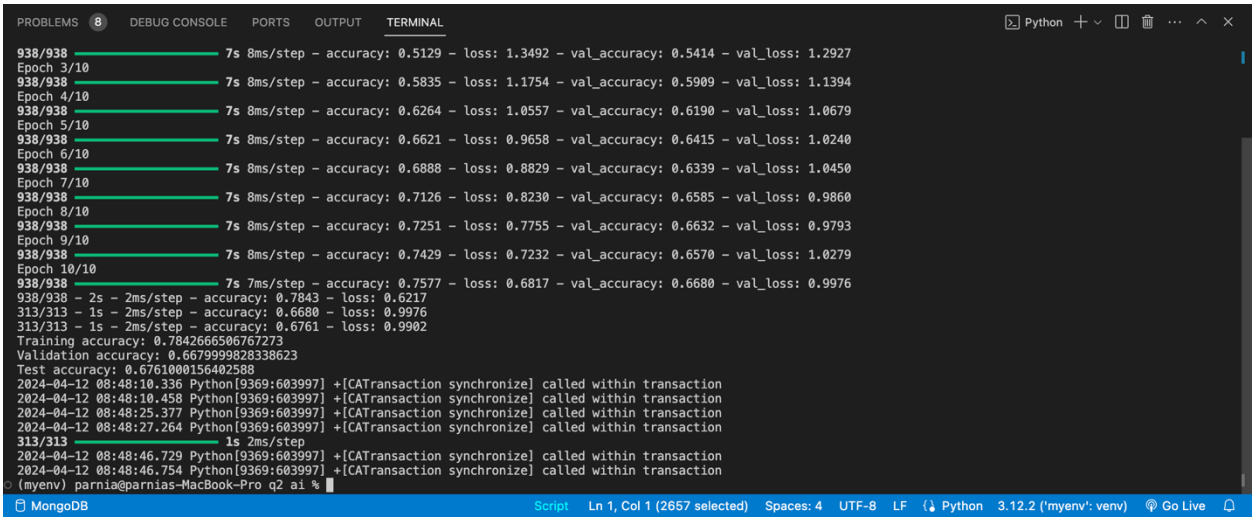
11 .دقت آموزش، دقت اعتبارسنجی، از دست دادن آموزش و از دست دادن اعتبار با استفاده از matplotlibرسم شده است.

12 .مدل با استفاده از روش پیش بینی داده های آزمون را پیش بینی می کند.

13 .پیش بینی های نمونه با استفاده از matplotlib تجسم می شوند. برای هر نمونه، تصویر، برچسب پیش بینی شده و برچسب واقعی نمایش داده می شود.


Predicted: Ship
True: Bird
Predicted: Deer
True: Deer
Predicted: Bird
True: Bird
Predicted: Truck
True: Truck
Predicted: Horse
True: Horse


Training Accuracy
Validation Accuracy
Training Loss
Validation Loss
Accuracy / Loss
Epoch
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
2
4
6
8



سوال ۳ :

بخش اول:

کد تصاویر و برچسب‌های مربوط به آن‌ها را از یک مجموعه داده بارگیری می‌کند، تصاویر را با تغییر اندازه و تبدیل آنها به مقیاس خاکستری از قبل پردازش می‌کند، کدگذاری برچسب و رمزگذاری یک‌بار روی برچسب‌ها را انجام می‌دهد، داده‌ها را به مجموعه‌های آموزشی و آزمایشی تقسیم می‌کند، یک مدل MLPبا استفاده از Keras می‌سازد. ، مدل را بر روی داده های آموزشی آموزش می دهد و عملکرد آن را روی داده های تست ارزیابی می کند.

مراحل اصلی کد عبارتند از:

1.بارگذاری تصاویر و برچسب ها از مجموعه داده.
2.پیش پردازش تصاویر با تغییر اندازه آنها به 32 32xپیکسل و تبدیل آنها به مقیاس خاکستری.
3.رمزگذاری برچسب ها
4.تقسیم داده ها به مجموعه های آموزشی و آزمایشی.
5.عادی سازی داده های تصویر با تقسیم آن بر 255.0.
6.ساخت یک مدل MLP با یک لایه مسطح، یک لایه متراکم با 256 واحد و فعال سازیReLU ، و یک لایه خروجی متراکم با فعال سازیsoftmax.
7.کامپایل مدل با بهینه ساز Adam و از دست دادن متقابل آنتروپی طبقه بندی شده.
8.آموزش مدل بر روی داده های آموزشی برای 10 دوره با سایز دسته ای 32.
9.ارزیابی دقت مدل بر روی داده های تست.

بخش دوم :

تابع `()load_data` به عنوان مرحله اول اجرا، تصاویر و برچسب‌ها را از مسیرهای مشخص شده بارگیری می‌کند. این تابع بر روی دایرکتوری‌های شامل تصاویر افراد حلقه می‌زند و تصاویر را با استفاده از کتابخانه OpenCV می‌خواند. سپس تصاویر را به رنگ خاکستری تبدیل کرده و به ابعاد 64 64xپیکسل تغییر اندازه می‌دهد. تصاویر و برچسب‌ها در لیست‌های `images` و `labels` اضافه می‌شوند. سپس برچسب‌ها با استفاده از `LabelEncoder` از scikit-learn رمزگذاری و با استفاده از تابع `to_categorical` از Keras به بردارهای دودویی تبدیل می‌شوند. سپس داده‌ها با استفاده از تابع `train_test_split`از scikit-learn به مجموعه‌های آموزش، اعتبارسنجی و آزمون تقسیم می‌شوند. در نهایت، داده‌ها به آرایه‌های قابل پردازش در Keras تبدیل می‌شوند. ابعاد داده‌ها برای استفاده در شبکه عصبی کانولوشنال (CNN) تغییر شکل داده می‌شوند.

تابع `()build_model` برای ساخت مدل CNN استفاده می‌شود. این مدل شامل لایه‌های کانولوشنال، لایه‌های ادغام(Pooling) ، لایه‌های تماما متصل (Fully Connected) و لایه‌ی آخر با فعال‌سازی softmaxبرای طبقه‌بندی است. مدل با بهینه‌ساز Adam و تابع هزینه categorical_crossentropy آموزش داده می‌شود.

تابع `()train_model` مدل را با استفاده از داده‌های آموزش و اعتبارسنجی آموزش می‌دهد. ابعاد ورودی و تعداد کلاس‌ها از طریق توابع `X_train[0].shape` و `y_train.shape[1]` تعیین می‌شوند. مدل با استفاده از تابع `fit` آموزش داده می‌شود و پس از آموزش، مدل بر روی داده‌های آزمون ارزیابی می‌شود.

تابع `()evaluate_model` مدل آموزش داده شده را بر روی داده‌های آزمون ارزیابی کرده و دقت طبقه‌بندی را چاپ می‌کند.

تابع `()save_trained_model` مدل آموزش داده شده را در مسیر مشخص شده ذخیره می‌کند.

ابتدا داده‌ها را بارگیری کرده، سپس مدل را آموزش می‌دهد و بر روی داده‌های آزمون ارزیابی می‌کند. سپس مدل آموزش داده شده را ذخیره می‌کند و نمونه‌هایی از تصاویر آزمون را نمایش می‌دهد. و کل فرایند آموزش، ارزیابی و ذخیره‌سازی مدل انجام می‌شود. همچنین تصاویر آزمون به همراه برچسب‌هایشان نمایش داده می‌شوند