گزارش پیاده سازی KAN با Pytorch

شبکه‌های کولموگروف-آرنولد (KAN) برای تقریب توابع چندبعدی استفاده می‌شوند. این شبکه‌ها براساس تئوری‌های کولموگروف و آرنولد برای بازنمایی توابع پیوسته بنا شده‌اند.

Version 1 (Without Grid Extension)

کلاس KANLinear

یک کلاس به نام KANLinear تعریف میکنیم که یک لایه‌ی سفارشی برای شبکه‌های عصبی با استفاده از PyTorch است. این لایه ترکیبی از یک لایه‌ی خطی استاندارد و یک بسط-B اسپلاین است .

in_features - تعداد ویژگی‌های ورودی.

out_features - تعداد ویژگی‌های خروجی.

grid_size - اندازه شبکه برای-B اسپلاین‌ها.

spline_order - ترتیب-B اسپلاین‌ها.

base_activation - تابع فعال‌سازی برای لایه پایه.

reset_parameters -این متد برای مقداردهی اولیه پارامترهای لایه استفاده می‌شود.

torch.nn.init.kaiming_uniform_ - برای مقداردهی وزن‌های پایه به‌طور یکنواخت.

self.spline_weight.data.fill_(0) - وزن‌های اسپلاین‌ها با مقدار صفر مقداردهی می‌شوند.

self.spline_scaler.data.fill_(1) - مقیاس‌کننده‌ی اسپلاین‌ها با مقدار یک مقداردهی می‌شود.

b_splines -این متد برای محاسبه‌ی پایه‌های-B اسپلاین برای ورودی داده شده استفاده می‌شود.

x.unsqueeze(-1) - یک بعد به انتهای x اضافه می‌کند.

bases - محاسبه پایه‌های-B اسپلاین با استفاده از شرایط خاص و ترتیب اسپلاین‌ها.

forward -این متد برای گذر جلو (forward pass) در لایه‌ی KANLinear استفاده می‌شود.

base_output - خروجی لایه‌ی پایه با اعمال تابع فعال‌سازی و وزن‌های پایه.

spline_output - خروجی لایه‌ی اسپلاین با استفاده از محاسبه‌ی-B اسپلاین‌ها و وزن‌های اسپلاین‌ها.

base_output + spline_output - خروجی نهایی که ترکیبی از خروجی لایه پایه و لایه‌ی اسپلاین است.

لایه‌ی KANLinear ترکیبی از یک لایه‌ی خطی استاندارد و یک لایه‌ی B-اسپلاین است که با استفاده از شبکه‌های B-اسپلاین بهبود یافته است. این ترکیب به مدل اجازه می‌دهد تا با استفاده از اسپلاین‌ها، انعطاف‌پذیری بیشتری در یادگیری روابط غیرخطی داشته باشد.

کلاس KAN

یک کلاس به نام KAN تعریف میکنیم که یک مدل شبکه عصبی با استفاده از لایه‌های KANLinear است. این مدل برای پیاده‌سازی یک شبکه عصبی با چندین لایه پنهان و استفاده از B-اسپلاین‌ها طراحی شده است.

layers_hidden - لیستی از تعداد نرون‌ها در هر لایه پنهان.

grid_size - اندازه شبکه برای B-اسپلاین‌ها.

spline_order - ترتیب B-اسپلاین‌ها.

base_activation - تابع فعال‌سازی برای لایه پایه.

self.layers - یک لیست از لایه‌های KANLinear که برای هر جفت از تعداد نرون‌های ورودی و خروجی در layers_hidden ساخته می‌شود.

forward -این متد برای گذر جلو (forward pass) در مدل KAN استفاده می‌شود.

x - ورودی مدل با شکل(batch_size, in_features).

for layer in self.layers - برای هر لایه درself.layers ، ورودی x از طریق لایه عبور می‌کند.

return x - خروجی نهایی که خروجی آخرین لایه است.

مدل KAN یک شبکه عصبی با چندین لایه پنهان است که هر لایه از نوع KANLinear است. این مدل از B-اسپلاین‌ها برای افزایش انعطاف‌پذیری در یادگیری روابط غیرخطی استفاده می‌کند. KAN با استفاده از لیستی از تعداد نرون‌ها در هر لایه پنهان مقداردهی اولیه می‌شود و با یک گذر جلو (forward pass) از تمام لایه‌ها عبور می‌کند تا خروجی نهایی را تولید کند.

Visualization of the 2D Dataset

تولید مجموعه داده

generate_dataset

ترسیم مجموعه داده

-ترسیم داده‌ها با استفاده ازmatplotlib

آماده‌سازی مدل KAN

-تعریف بهینه‌ساز باtorch.optim.Adam

-تعریف تابع خطا باtorch.nn.CrossEntropyLoss

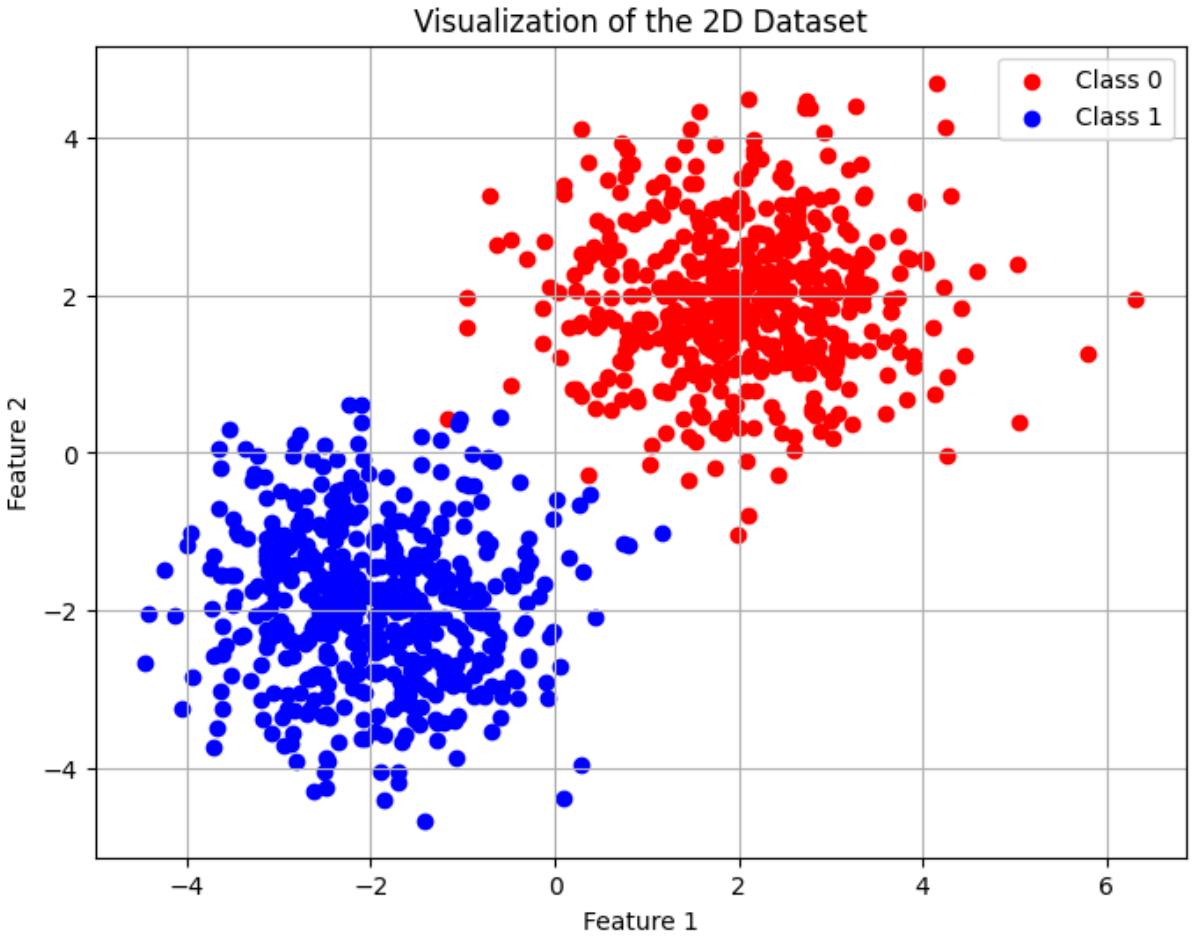
آموزش مدل

-حلقه آموزش مدل با استفاده از for epoch in range(10)

-محاسبه گذر جلو باmodel.forward

-محاسبه خطا باcriterion

- به‌روزرسانی وزن‌ها با optimizer.step



Visualization of the Spiral Dataset

تولید مجموعه داده

-این تابع یک مجموعه داده مارپیچی دوبعدی تولید می‌کند. هر کلاس شامل generate_spiral_dataset تعداد مشخصی نمونه است که به صورت مارپیچی دور یک مرکز قرار دارند.

ترسیم مجموعه داده

-ترسیم داده‌ها با استفاده از matplotlib این بخش داده‌های تولید شده را ترسیم می‌کند. نقاط مربوط به کلاس 0 با رنگ قرمز و کلاس 1 با رنگ آبی نمایش داده می‌شوند.

آماده‌سازی مدل

-مدل شبکه عصبی KAN با ساختار [2, 50, 50, 2] ایجاد می‌شود که پیچیدگی مدل را افزایش KAN می‌دهد تا بتواند با مجموعه داده مارپیچی کار کند.

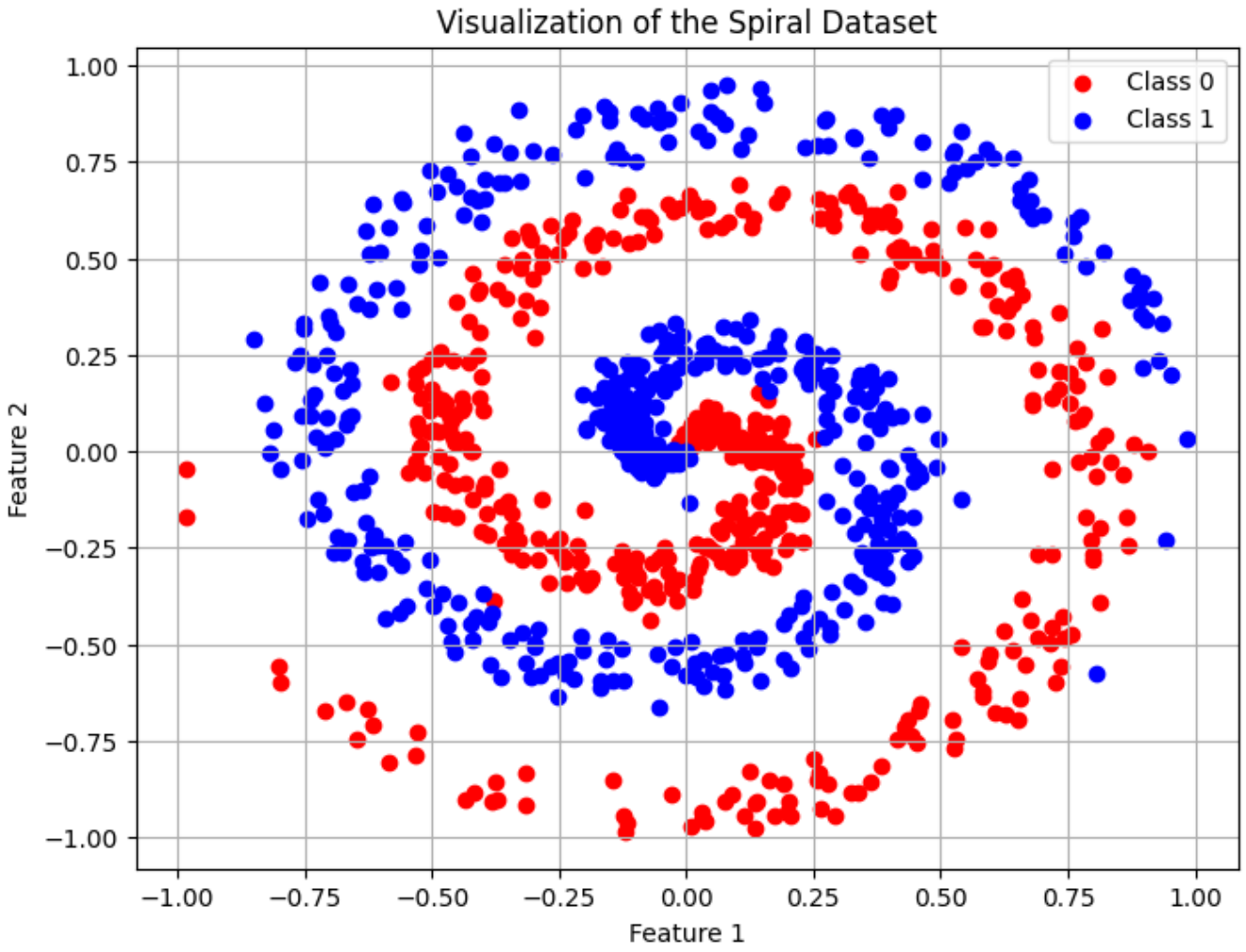
-تعریف بهینه‌ساز با torch.optim.Adam بهینه‌ساز Adam با نرخ یادگیری 0.001 برای تنظیم وزن‌های مدل استفاده می‌شود.

-تعریف تابع خطا با torch.nn.CrossEntropyLoss تابع خطای CrossEntropyLoss برای محاسبه تفاوت بین خروجی مدل و برچسب‌های واقعی استفاده می‌شود.

آموزش مدل

-حلقه آموزش مدل با استفاده از for epoch in range(20) مدل برای 20 دوره آموزش داده می‌شود تا بتواند به خوبی روی مجموعه داده مارپیچی عمل کند.

- محاسبه گذر جلو با model.forward داده‌ها از طریق مدل عبور داده می‌شوند تا خروجی پیش‌بینی شده تولید شود.
- محاسبه خطا با criterion تفاوت بین خروجی پیش‌بینی شده و برچسب‌های واقعی محاسبه می‌شود.
- به‌روزرسانی وزن‌ها با optimizer.step وزن‌های مدل با استفاده از بهینه‌ساز Adam به‌روزرسانی می‌شوند.



Version 2 (With Grid Extension)

کلاس KANLinear

افزودن تابع extend_grid بهKANLinear

-extend_grid این تابع به KANLinear اضافه شده است و اجازه می‌دهد تا اندازه شبکه B-splines را به طور دینامیک تغییر دهد و وزن‌های spline را به‌روز کند. این امکان به شبکه اجازه می‌دهد تا در مواجهه با داده‌هایی با ساختار متفاوت، بهبود یابد و تطبیق پذیر باشد.

-اضافه شدن extend_grid در تابع سازنده __init__ این امکان به لایه اضافه شده است که شبکه-B splinesرا با تغییر اندازه موجودیتهای پارامتری اش به روزرسانی کند.

تغییرات درforward

-استفاده از spline_weight به جای spline_scaler.unsqueeze(-1) spline_weight در متد forwardاز این تغییر برای ساده‌سازی محاسبات استفاده شده است.

افزوده شدن extend_grid بهreset_parameters

-استفاده از extend_grid در reset_parameters این کد از امکان extend_grid در reset_parametersبرای تغییرات اولیه و آماده‌سازی لایه استفاده می‌کند.

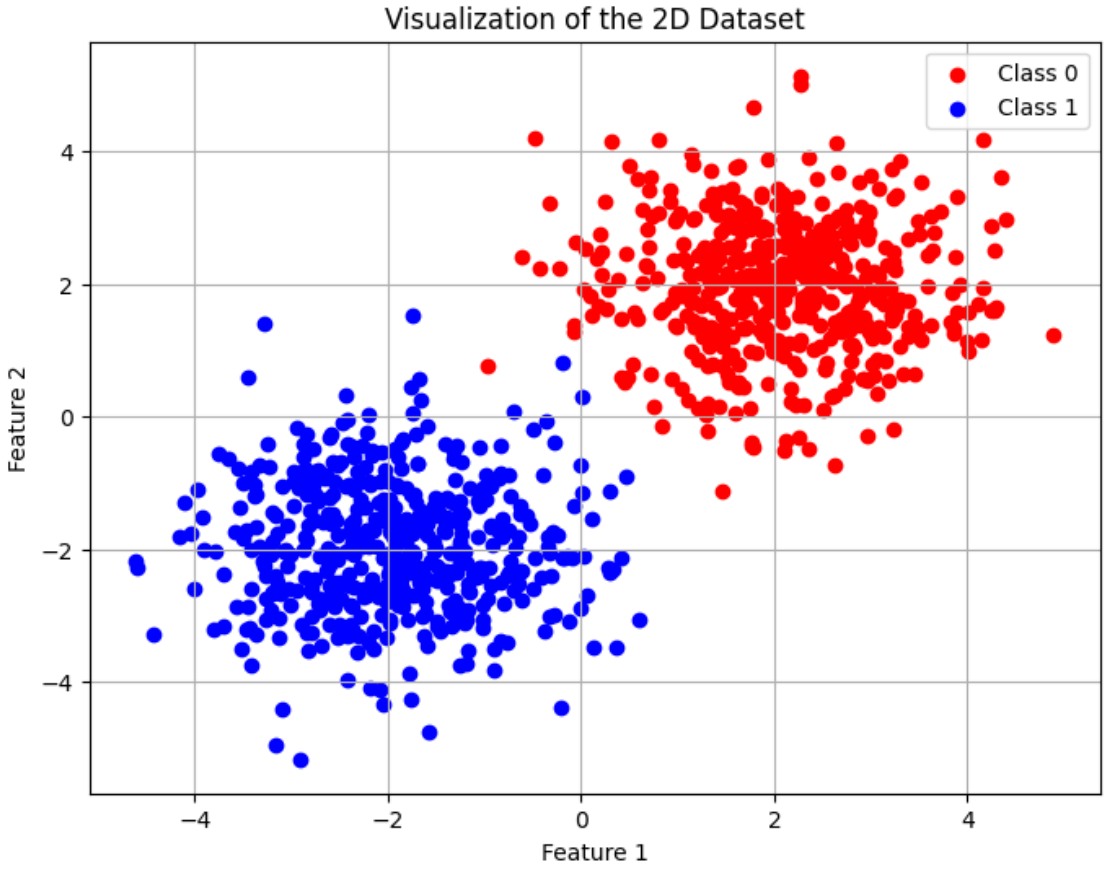
این تغییرات باعث شده است که KANLinear بهترین قابلیت تطبیق را با داده‌های مختلف داشته باشد و قابلیت اضافه کردن انعطاف‌پذیری بیشتری به شبکه عصبی را فراهم کند.

کلاس KAN

با اضافه کردن extend_gridو قابلیت تغییر اندازه شبکهB-splines ، از یک پلتفرم آموزشی قوی‌تر و انعطاف‌پذیرتر برخورداریم که این امکان را به شبکه می‌دهد تا با داده‌های مختلف و در مواجهه با چالش‌های متنوع، بهبود یابد.

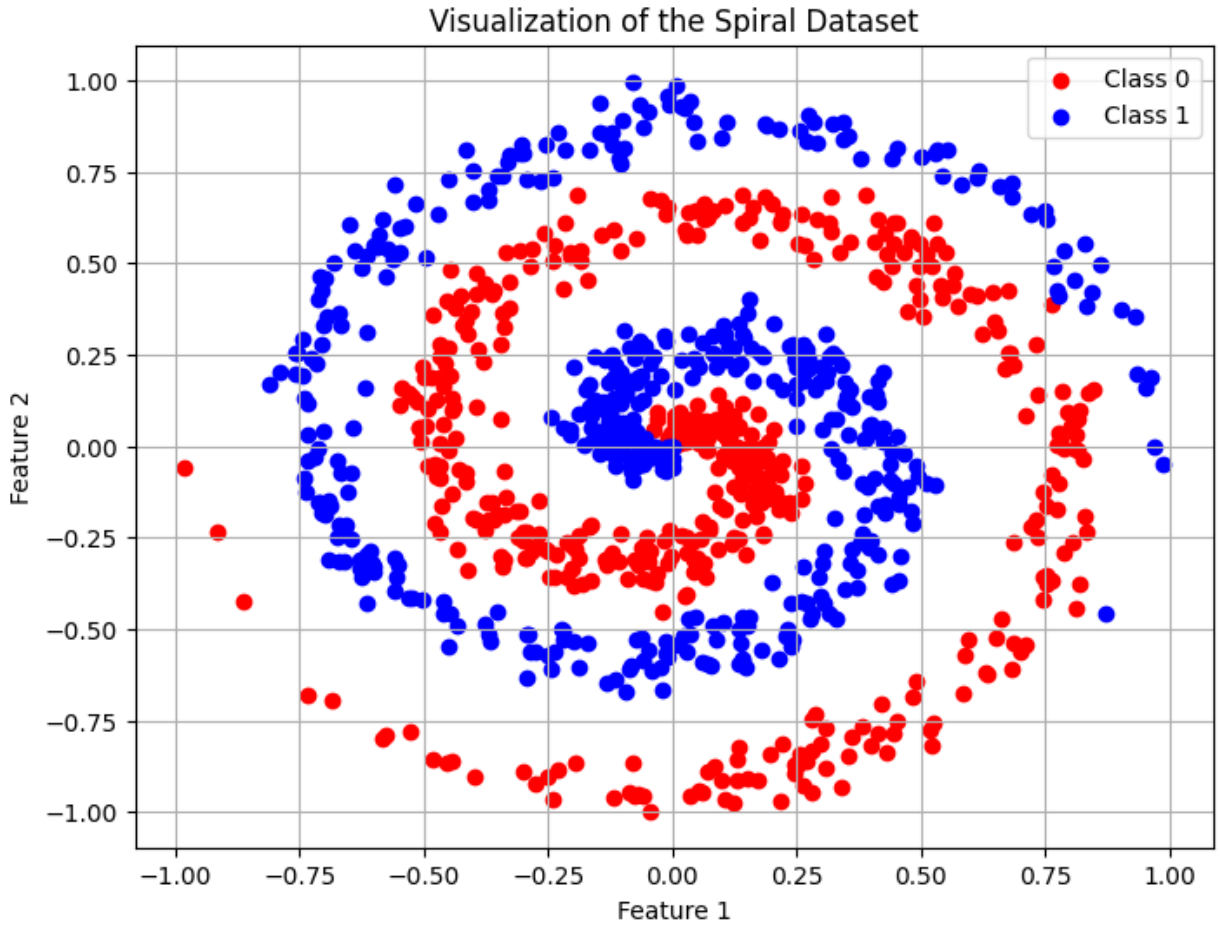
Visualization of the 2D Dataset

در طول آموزش (در این مثال در اپوک 5)، شبکه B-splines با استفاده از تابع extend_gridبه طور دینامیک تغییر می‌کند((layer.extend_grid(new_grid_size=10) ، که این امکان را به مدل می‌دهد تا بهبود یافته و بهتر با داده‌های پیچیده‌تر سازگار شود.



Visualization of the Spiral Dataset

مانند بالا از extended grid استفاده کردیم.



نتیجه گیری

1 .بدون استفاده از شبکه B-splines گسترده

- در مدل‌هایی که بدون استفاده از توسعه شبکه B-splines هستند، اندازه شبکه ثابت است و تغییرات در آن در طول آموزش اعمال نمی‌شود. این معمولاً به محدودیت در قابلیت تطبیق مدل با داده‌های پیچیده‌تر منجر می‌شود، زیرا توانایی مدل در نمایش ساختارهای پیچیده‌تر یا غیرخطی محدود است.

2 .با استفاده از شبکه B-splines و تغییرات در آن

- مدل‌هایی که از شبکه B-splines استفاده می‌کنند و قابلیت تغییر دینامیکی در اندازه شبکه B-splinesرا دارند، قادرند در طول آموزش این انعطاف را داشته باشند و بهبود یافته‌اند. به عنوان مثال، با استفاده از تابعextend_grid ، می‌توان شبکه B-splines را در هر مرحله از آموزش به صورت دینامیک تغییر داد تا با ساختار داده‌های ورودی بهتر سازگار شود.

3 .نتایج عملی برای دقت و عملکرد مدل

- در آزمایش‌ها و آموزش‌هایی که با داده‌های پیچیده‌تر مانند دیتاست مارپیچی انجام شد، مدل که از توسعه شبکه B-splines با استفاده از تابع extend_grid استفاده کرده بود، دقت و عملکرد بهتری نسبت به مدلی که بدون این امکان عمل می‌کرد، نشان داده شد. این تفاوت‌ها نشان می‌دهد که توانایی تطبیق مدل با ساختارهای پیچیده‌تر و افزایش دقت در تشخیص دسته‌ها افزایش می‌یابد.

=> بنابراین، استفاده از شبکه B-splines با امکان تغییر دینامیکی در آن (مانند استفاده از تابع extend_grid) می‌تواند بهبود قابل توجهی در دقت و عملکرد مدل در مواجهه با داده‌های پیچیده‌تر مانند دیتاست مارپیچی داشته باشد، زیرا مدل قادر به یادگیری و نمایش ساختارهای غیرخطی و پیچیده‌تری خواهد بود.