Design of Chebyshev Type I Filter Using Deep Neural Networks

طراحی فیلتر چبی‌شف نوع اول با استفاده از شبکه عصبی عمیق

علیرضا نام آور

بهار۱٤۰٤

فهرست

مقدمه

در عصر حاضر، با گسترش روزافزون کاربردهای پردازش سیگنال، مخابرات و سامانه‌های الکترونیکی، طراحی بهینه و دقیق فیلترها به یکی از مسائل بنیادین در مهندسی برق تبدیل شده است. فیلترهای آنالوگ کلاسیک مانندButterworth ، Bessel، و به‌ویژهChebyshev ، همواره در طراحی سیستم‌هایی با نیاز به تفکیک فرکانسی بالا مورد استفاده قرار گرفته‌اند. در میان آن‌ها، فیلتر چبی‌شف نوع اول به‌دلیل دارا بودن ریپل کنترل‌شده در باند گذر و شیب تند در ناحیه گذار، جایگاه ویژه‌ای در طراحی فیلترهای فرکانسی دارد. با این حال، طراحی دقیق این نوع فیلتر نیازمند محاسبات تحلیلی نسبتاً پیچیده‌ای است که شامل محاسبه قطب‌ها و صفرها، تحلیل تابع تبدیل، و استخراج پاسخ بهره در حوزه فرکانس می‌باشد. این فرآیند به‌ویژه برای فیلترهایی با مرتبه بالا یا در محیط‌های محدود از نظر منابع محاسباتی، زمان‌بر و پردازشی سنگین تلقی می‌شود.

در همین راستا، پروژه حاضر با هدف تلفیق روش‌های کلاسیک طراحی فیلتر با روش‌های نوین یادگیری ماشین و شبکه‌های عصبی عمیق شکل گرفته است. در این پروژه تلاش شد با آموزش یک مدل یادگیری عمیق، فرآیند طراحی فیلتر چبی‌شف نوع اول به یک سیستم داده‌محور و سریع تبدیل گردد. در گام نخست، مشخصات فیزیکی فیلتر (مرتبه، ریپل، فرکانس قطع) به‌عنوان ورودی مدل در نظر گرفته شد و پاسخ بهره آن در محدوده فرکانسی وسیع به‌عنوان خروجی استخراج گردید. با تولید مجموعه‌ای از فیلترهای مختلف و محاسبه پاسخ بهره آن‌ها به‌صورت دقیق، یک دیتاست آموزشی غنی تهیه شد. سپس با استفاده از کتابخانه‌های پیشرفته مانندPyTorch ، یک شبکه عصبی چندلایه طراحی و آموزش داده شد که توانست رابطه‌ی غیرخطی میان مشخصات طراحی و پاسخ فرکانسی را به‌خوبی فراگیرد.

در ادامه، عملکرد مدل آموزش‌دیده از منظر دقت عددی، میزان خطای پیش‌بینی، و توانایی در بازسازی ویژگی‌های کلیدی فیلتر مورد ارزیابی قرار گرفت. نتایج عددی نشان داد که مدل قادر است با میانگین خطای بسیار کم (کمتر از ۱.۵ دسی‌بل در اکثر موارد)، پاسخ بهره فیلتر را برای فیلترهایی با مرتبه‌های گوناگون بازتولید کند. همچنین برای کاربردی‌سازی بیشتر مدل، یک رابط کاربری گرافیکی (GUI) با استفاده از کتابخانه tkinter طراحی شد تا کاربر نهایی بتواند بدون نیاز به دانش برنامه‌نویسی یا آشنایی با یادگیری ماشین، تنها با وارد کردن چند پارامتر ساده، پاسخ فیلتر را مشاهده کرده و آن را با روش سنتی مقایسه نماید. در فاز نهایی پروژه، مدل طراحی‌شده با روش سنتی مورد مقایسه کمی و کیفی قرار گرفت و مشخص شد که مدل شبکه عصبی علاوه بر دقت بالا، از نظر سرعت محاسباتی و سادگی استفاده نیز برتری چشمگیری دارد.

پروژه حاضر با ساختاری مرحله‌به‌مرحله و بر پایه مفاهیم علمی مهندسی برق، یادگیری ماشین و توسعه نرم‌افزار شکل گرفته است و می‌تواند به‌عنوان الگویی جامع برای طراحی مدل‌محور سیستم‌های مهندسی در حوزه‌های دیگر نیز به‌کار رود. این پروژه نه‌تنها نمونه‌ای از کاربرد عملی یادگیری عمیق در مهندسی کلاسیک محسوب می‌شود، بلکه چشم‌اندازی برای توسعه ابزارهای طراحی خودکار، بلادرنگ و هوشمند فیلتر در کاربردهای صنعتی، تحقیقاتی و آموزشی فراهم می‌آورد.

فاز اول: بررسی جامع تئوری فیلتر چبی‌شف نوع اول

۱ .مقدمه‌ای بر فیلترها

در دنیای مهندسی برق و الکترونیک، **فیلترها** یکی از مهم‌ترین ابزارهای تحلیل و طراحی مدارهای سیگنال هستند. این ابزارها امکان جداسازی فرکانس‌های مطلوب از نامطلوب را فراهم می‌کنند. به عنوان مثال، در سیستم‌های صوتی، فیلترها می‌توانند نویزهای فرکانس بالا را حذف کرده و تنها فرکانس‌های شنیداری را عبور دهند.

فیلترها معمولاً بر اساس نحوه طراحی و کاربردشان به دسته‌های زیر تقسیم می‌شوند:

- **فیلتر پایین‌گذر** :(Low-Pass Filter) عبور فرکانس‌های پایین‌تر از یک حد مشخص و حذف فرکانس‌های بالاتر
- **فیلتر بالاگذر** :(High-Pass Filter) عبور فرکانس‌های بالا و حذف فرکانس‌های پایین
- **فیلتر میان‌گذر** :(Band-Pass Filter) عبور یک باند خاص از فرکانس‌ها و حذف مابقی
- **فیلتر قطع‌باند** :(Band-Stop Filter) حذف یک باند خاص از فرکانس‌ها

۲ .دسته‌بندی فیلترها از نظر پیاده‌سازی

از نظر پیاده‌سازی، فیلترها به دو گروه اصلی تقسیم می‌شوند:

الف. فیلترهای آنالوگ

فیلترهایی که در حوزه زمان پیوسته با استفاده از عناصر پسیو مانند مقاومت(R) ، خازن (C) و سلف (L) پیاده‌سازی می‌شوند. در این فیلترها معادلات دیفرانسیلی برای توصیف رفتار سیگنال استفاده می‌شود.

ب. فیلترهای دیجیتال

این فیلترها در سیستم‌های پردازش عددی پیاده‌سازی می‌شوند و بر اساس تبدیل Z یا تبدیل فوریه گسسته طراحی می‌شوند. امروزه کاربرد بسیاری در پردازش صوت، تصویر، داده‌های پزشکی و مخابرات دارند.

۳ .معرفی فیلترهای کلاسیک آنالوگ

در طراحی فیلترهای آنالوگ، چهار خانواده‌ی شناخته‌شده وجود دارد که هر کدام برای هدف خاصی استفاده می‌شوند در جدول۱ این چهار نوع فیلتر و ویژگی های برجسته آنها را مشاهده میکنید:

جدول ۱(فیلترهای کلاسیک آنالوگ و ویژگی های آنها)

نوع فیلتر	ویژگی برجسته
باتروورث(Butterworth)	پاسخ فرکانسی بسیار یکنواخت در باند گذر (بدون ریپل)
چبی‌شف نوع اول	ریپل در باند گذر و شیب تندتر نسبت به باتروورث
چبی‌شف نوع دوم	ریپل در باند توقف، باند گذر یکنواخت
بیسل(Bessel)	حفظ شکل سیگنال در حوزه زمان (تاخیر گروهی ثابت‌تر)

۴ .فیلتر چبی‌شف نوع اول

۴.۱ تعریف

فیلتر چبی‌شف نوع اول یکی از فیلترهای با کارایی بالا برای حذف فرکانس‌های ناخواسته است که ویژگی متمایز آن، وجود **ریپل کنترل‌شده در باند گذر** و **شیب بسیار تند در ناحیه گذار** می‌باشد. این ویژگی آن را برای کاربردهایی که به دقت بالا در تفکیک فرکانس نیاز دارند، مناسب می‌سازد.

۴.۲ تابع انتقال(Transfer Function)

تابع انتقال به صورت زیر تعریف می‌شود:

$$|H(j\omega)|^2 = \frac{1}{1 + \epsilon^2 T_n^2(\omega/\omega_c)}$$

که در آن:

- $H(j\omega)$:بهره (gain) فیلتر در فرکانس ω
- ω_c :فرکانس قطع
- ϵ :ضریب کنترل ریپل باند گذر
- $T_n(x)$:چندجمله‌ای چبی‌شف از مرتبه n

۵ .چندجمله‌ای‌های چبی‌شف

چندجمله‌ای‌های چبی‌شف نوع اول توسط رابطه بازگشتی زیر تعریف می‌شوند:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$.$$
$$.$$
$$.$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

این چندجمله‌ای‌ها در بازه $-1 \leq x \leq 1$ نوسانی هستند و حداکثر مقدارشان برابر با ۱ است، اما در خارج از این بازه به‌صورت نمایی رشد می‌کنند.

۶ .نقش پارامتر ϵ

مقدار ϵ تعیین‌کننده میزان ریپل در باند گذر است. هرچه مقدار ϵ بزرگ‌تر باشد:

- ریپل (نوسان) باند گذر بیشتر می‌شود
- شیب فیلتر در ناحیه گذار (transition band) تندتر می‌شود

در حالت خاص که $\epsilon = 0$، تابع چبی‌شف به تابع باتروورث تبدیل می‌شود و ریپل کاملاً حذف می‌گردد.

۷ .مقایسه با سایر فیلترها

در جدول ۲ فیلترچبی شف با سایر فیلتر های کلاسیک آنالوگ مقایسه شده است:

جدول ۲(مقایسه چبی شف با سایر فیلترها)

ویژگی	Butterworth	Chebyshev Type I	Chebyshev Type II	Bessel
ریپل باند گذر	ندارد	دارد	ندارد	ندارد
ریپل باند توقف	ندارد	ندارد	دارد	ندارد
سرعت افت پاسخ فرکانسی	متوسط	زیاد	زیاد	کم
پایداری فاز	متوسط	پایین	پایین	بالا

۸ .کاربردهای فیلتر چبی‌شف نوع اول

- فیلترکردن نویز در سیستم‌های صوتی و مخابراتی
- طراحی فیلترهای فعال در مدارهای آنالوگ
- جداسازی دقیق سیگنال‌های با فرکانس نزدیک
- در سیستم‌های مانیتورینگ زیستی یا ابزار دقیق

۹ .جمع‌بندی فاز اول

در این فاز با اصول طراحی فیلتر چبی‌شف نوع اول آشنا شدیم. این فیلتر با ترکیب ریپل کنترل‌شده و شیب تند، انتخاب مناسبی برای کاربردهای حساس فرکانسی است. در مراحل بعدی، با استفاده از پایتون و کتابخانه‌های مهندسی، یک نمونه از این فیلتر را طراحی و تحلیل خواهیم کرد.

فاز دوم: پیاده‌سازی فیلتر چبی‌شف نوع اول با پایتون و تحلیل پاسخ آن

۱ .هدف فاز دوم

- آشنایی با نحوه طراحی فیلتر چبی‌شف در نرم‌افزار پایتون
- یادگیری استفاده از کتابخانه scipy.signalبرای طراحی فیلتر
- رسم پاسخ فرکانسی، نمودار بهره (gain) و فاز
- تحلیل اثر پارامترهایی نظیر ریپل و مرتبه فیلتر

۲ .پیش‌نیاز: آشنایی با کتابخانه‌های مورد استفاده

در این بخش از پایتون، از کتابخانه‌های زیر استفاده می‌کنیم:

- numpy :برای محاسبات عددی
- scipy.signal :برای طراحی فیلترهای کلاسیک و تحلیل سیگنال
- matplotlib.pyplot :برای رسم نمودارها

برای نصب این کتابخانه‌ها (در صورت نیاز) از دستور زیر استفاده می‌شود:

```
pip install numpy scipy matplotlib
```

۳ .پیاده‌سازی گام‌به‌گام

الف. وارد کردن کتابخانه‌ها

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import cheby1, freqs
```

ب. تعریف پارامترهای فیلتر

در این مثال، یک فیلتر پایین‌گذر چبی‌شف نوع اول را طراحی می‌کنیم:

```
# Filter specifications
order = 4                    # Filter order (degree of the Chebyshev
polynomial)
ripple = 1                   # Maximum ripple in the passband (in dB)
cutoff_freq = 1000           # Cutoff frequency in Hz
sampling_freq = 10000        # Sampling frequency in Hz
```

ج. طراحی فیلتر چبی‌شف نوع اول (آنالوگ)

```
# Normalize the cutoff frequency for analog design
Wn = 2 * np.pi * cutoff_freq  # Convert Hz to rad/s

# Design Chebyshev Type I filter (analog)
b, a = cheby1(order, ripple, Wn, btype='low', analog=True)
```

- b, a :ضرایب صورت و مخرج تابع انتقال H(s)
- cheby1 :تابع طراحی فیلتر چبی‌شف نوع اول
- analog=True :طراحی در حوزه آنالوگ (پیوسته)

د. محاسبه و رسم پاسخ فرکانسی

```
# Frequency range for plotting
w, h = freqs(b, a, worN=np.logspace(1, 5, 500))

# Plot magnitude response
plt.figure(figsize=(10, 6))
plt.semilogx(w, 20 * np.log10(np.abs(h)))
plt.title('Chebyshev Type I Low-Pass Filter (Magnitude Response)')
plt.xlabel('Frequency [rad/s]')
plt.ylabel('Magnitude [dB]')
plt.grid(True)
plt.axvline(Wn, color='red', linestyle='--', label='Cutoff frequency')
plt.legend()
plt.show()
```

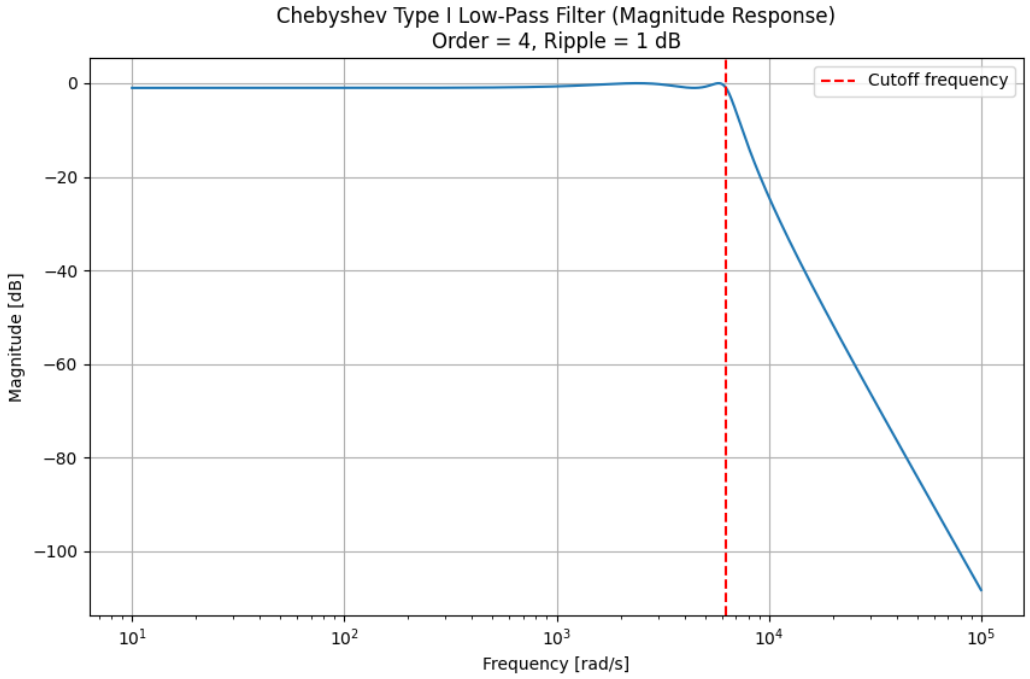
ه. رسم پاسخ فاز فیلتر

```
# Plot phase response
plt.figure(figsize=(10, 6))
plt.semilogx(w, np.angle(h))
plt.title('Chebyshev Type I Low-Pass Filter (Phase Response)')
plt.xlabel('Frequency [rad/s]')
```

```
plt.ylabel('Phase [radians]')
plt.grid(True)
plt.show()
```

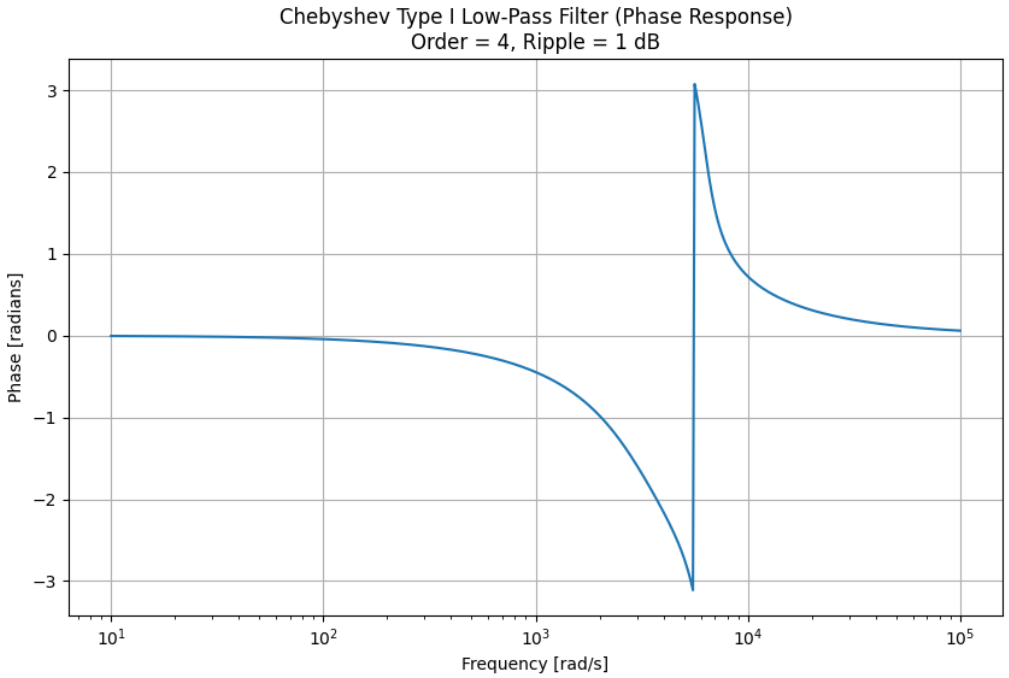
۴ .تحلیل نتایج

در شکل ۱ نمودار نشان داده‌شده مربوط به پاسخ بهره‌ی یک فیلتر پایین‌گذر چبی‌شف نوع اول است که با استفاده از زبان برنامه‌نویسی پایتون و بر مبنای پارامترهای مشخصی شامل مرتبه چهار، ریپل یک دسی‌بل، و فرکانس قطع ۱۰۰۰ هرتز طراحی شده است. همان‌طور که در نمودار مشاهده می‌شود، محور افقی نمایانگر فرکانس زاویه‌ای (بر حسب رادیان بر ثانیه) در مقیاس لگاریتمی و محور عمودی نشان‌دهنده بهره یا گین فیلتر در مقیاس دسی‌بل است. در قسمت ابتدایی نمودار، که ناحیه باند گذر فیلتر را نشان می‌دهد، بهره تقریباً برابر صفر دسی‌بل باقی مانده و این بدان معناست که سیگنال‌های دارای فرکانس‌های پایین‌تر از فرکانس قطع با تضعیف ناچیزی عبور می‌کنند. نوسانات کوچکی که در این ناحیه مشاهده می‌شود ناشی از ویژگی ذاتی فیلتر چبی‌شف نوع اول بوده و به‌عنوان ریپل باند گذر شناخته می‌شود. این ریپل‌ها تابعی از پارامتر ε هستند که در طراحی مقدار آن برابر با ریپل یک دسی‌بل در نظر گرفته شده است. در نقطه‌ای که با خط چین قرمز در نمودار مشخص شده، فرکانس قطع فیلتر قرار دارد و از این نقطه به بعد، بهره فیلتر به‌سرعت افت می‌کند که نشان‌دهنده شیب تند (roll-off) پاسخ فیلتر در ناحیه گذار است. این شیب تند یکی از مزیت‌های اصلی فیلتر چبی‌شف در مقایسه با فیلتر باتروورث است. پس از این ناحیه، در باند توقف، پاسخ بهره فیلتر به مقادیر بسیار منفی می‌رسد) زیر -۱۰۰ dB) که حاکی از تضعیف شدید سیگنال‌های دارای فرکانس‌های بالا و عملکرد مؤثر فیلتر در حذف فرکانس‌های نامطلوب است. این نمودار عملکرد موفقیت‌آمیز فیلتر طراحی‌شده را به‌عنوان یک ابزار مؤثر در کاربردهای پردازش سیگنال، حذف نویز، یا طراحی مدارهای آنالوگ با حساسیت فرکانسی بالا تأیید می‌کند.



شکل ۱(پاسخ بهره‌ی یک فیلتر پایین‌گذر چبی‌شف نوع اول)

در شکل ۲نمودار نمایش داده‌شده مربوط به پاسخ فاز یک فیلتر پایین‌گذر چبی‌شف نوع اول است. محور افقی این نمودار فرکانس زاویه‌ای را در مقیاس لگاریتمی نشان می‌دهد و محور عمودی مقدار فاز را بر حسب رادیان نمایش می‌دهد. در ناحیه فرکانس‌های پایین، یعنی جایی که سیگنال‌ها از باند گذر فیلتر عبور می‌کنند، مقدار فاز تقریباً نزدیک به صفر است که نشان‌دهنده‌ی تأخیر فازی نسبتاً کم و یکنواخت در این بخش از سیگنال‌هاست. با نزدیک شدن به فرکانس قطع، مقدار فاز به‌تدریج کاهش می‌یابد و یک شیب نزولی واضح مشاهده می‌شود. این تغییر فاز به‌دلیل خاصیت طبیعی فیلترهای آنالوگ رخ می‌دهد که در ناحیه گذار موجب پیچیدگی بیشتر رفتار فازی می‌شوند. در حوالی فرکانس قطع، شاهد یک پرش ناگهانی در مقدار فاز هستیم که از مقدار نزدیک به منفی π به مثبت π منتقل می‌شود. این رفتار به‌عنوان یک ناپیوستگی فازی شناخته می‌شود و از ویژگی‌های رایج در پاسخ فاز فیلترهای دارای قطب‌های مختلط است. پس از این نقطه، فاز دوباره به‌تدریج کاهش می‌یابد و در فرکانس‌های بالاتر به مقدار ثابت و پایینی میل می‌کند. این رفتار فاز، گرچه تأثیری بر بهره ندارد، اما در بسیاری از کاربردهای حساس به فاز از جمله ارتباطات دیجیتال و سیستم‌های کنترل، اهمیت دارد. بنابراین در تحلیل کامل عملکرد یک فیلتر، بررسی دقیق پاسخ فاز در کنار پاسخ بهره ضروری است. نمودار حاضر به‌خوبی نشان می‌دهد که فیلتر چبی‌شف نوع اول نه تنها دارای پاسخ فرکانسی مؤثر است، بلکه ویژگی‌های فازی قابل توجهی نیز دارد که باید در طراحی نهایی در نظر گرفته شوند.



شکل ۲(پاسخ فاز یک فیلتر پایین‌گذر چبی‌شف نوع اول)

پاسخ بهره:(magnitude)

- در ناحیه فرکانس پایین‌تر از نقطه قطع، بهره فیلتر تقریباً یکنواخت است اما دارای ریپل‌هایی کوچک است. این همان ویژگی ذاتی فیلتر چبی‌شف نوع اول است.

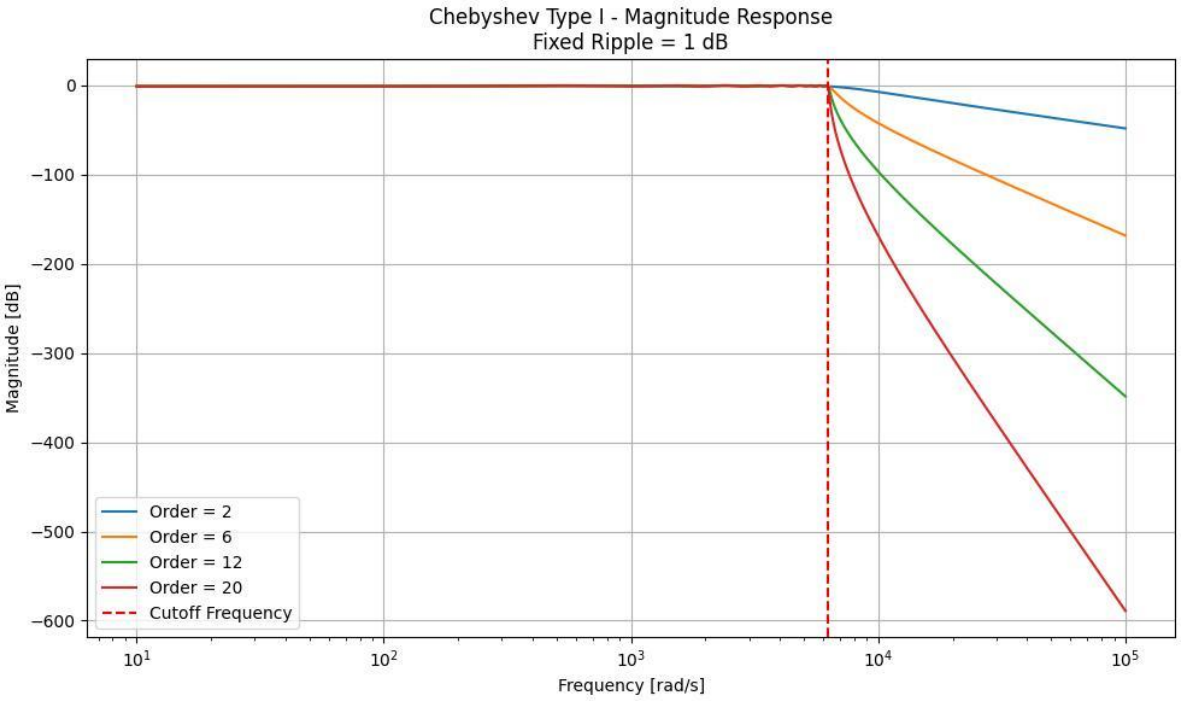
- در فرکانس‌های بالاتر از نقطه قطع، بهره با سرعت زیاد افت می‌کند. این افت سریع نشان‌دهنده **شیب تند (steep roll-off)** فیلتر است که یکی از دلایل استفاده از چبی‌شف است.

پاسخ فاز:(phase)

- فاز در فرکانس‌های پایین نسبتاً ثابت است ولی در ناحیه گذار دچار تغییر ناگهانی می‌شود. این ویژگی بر شکل سیگنال در حوزه زمان اثر می‌گذارد (تأخیر فاز).

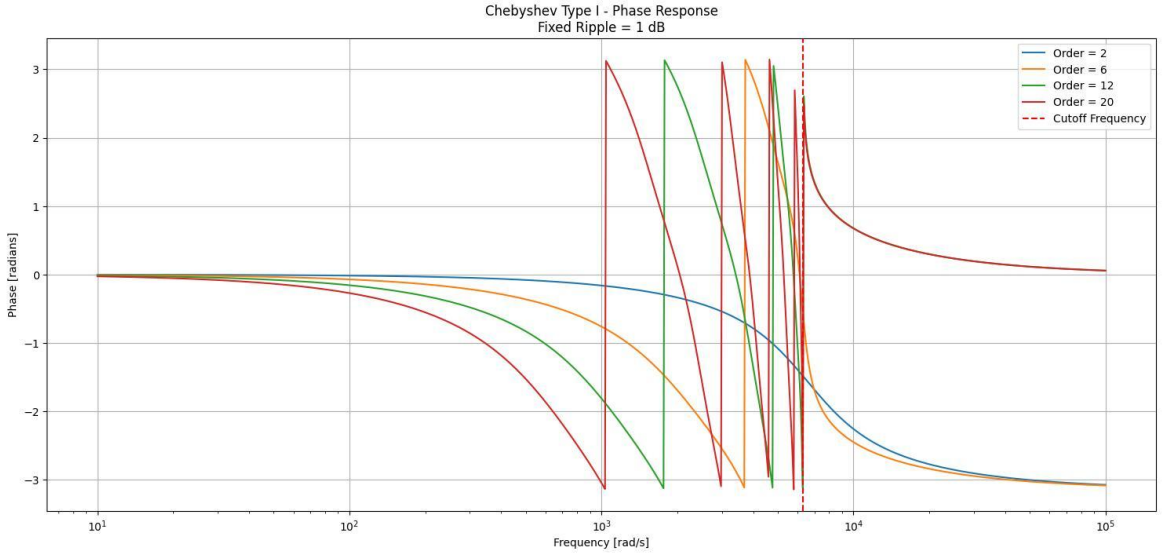
۵ .اثر تغییر پارامترها

- اگر **مرتبه فیلتر (order)** افزایش یابد، شیب ناحیه گذار تندتر می‌شود ولی پیچیدگی فیلتر نیز بیشتر خواهد شد(شکل ۳).



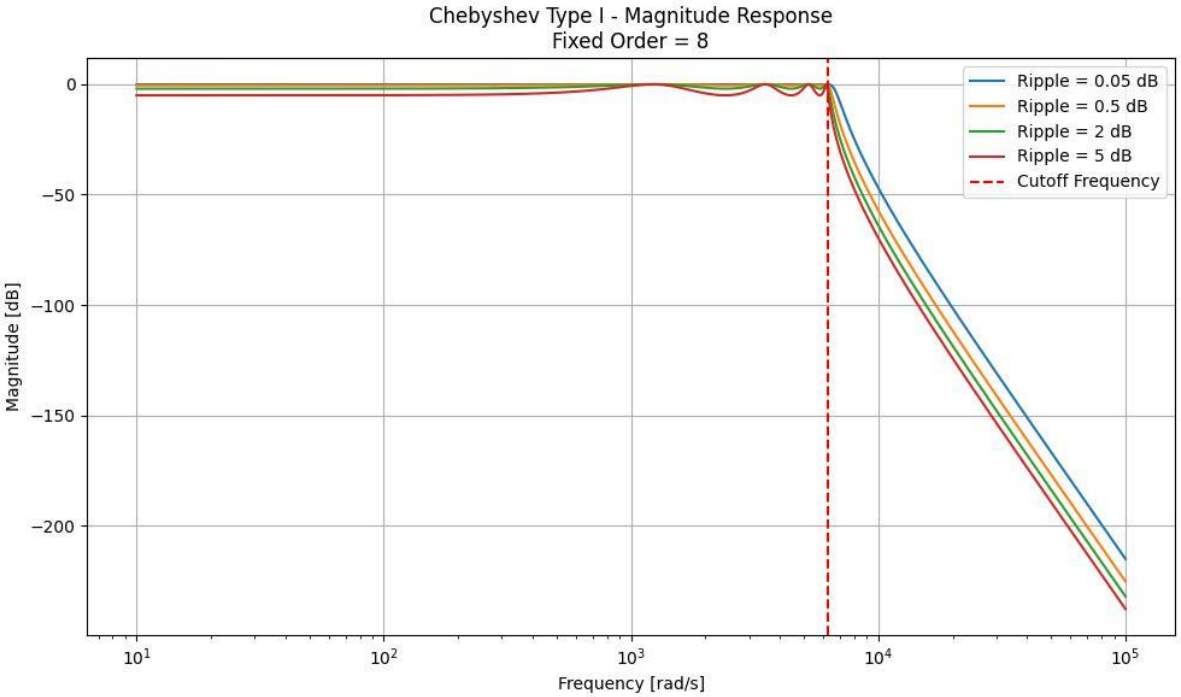
شکل ۳(پاسخ بهره‌ی یک فیلتر پایین‌گذر به ازای مرتبه های مختلف و ریپل ثابت)

- با افزایش مرتبه فیلتر، فاز در ناحیه گذار تغییرات سریع‌تری پیدا می‌کند و شیب آن بیشتر می‌شود.



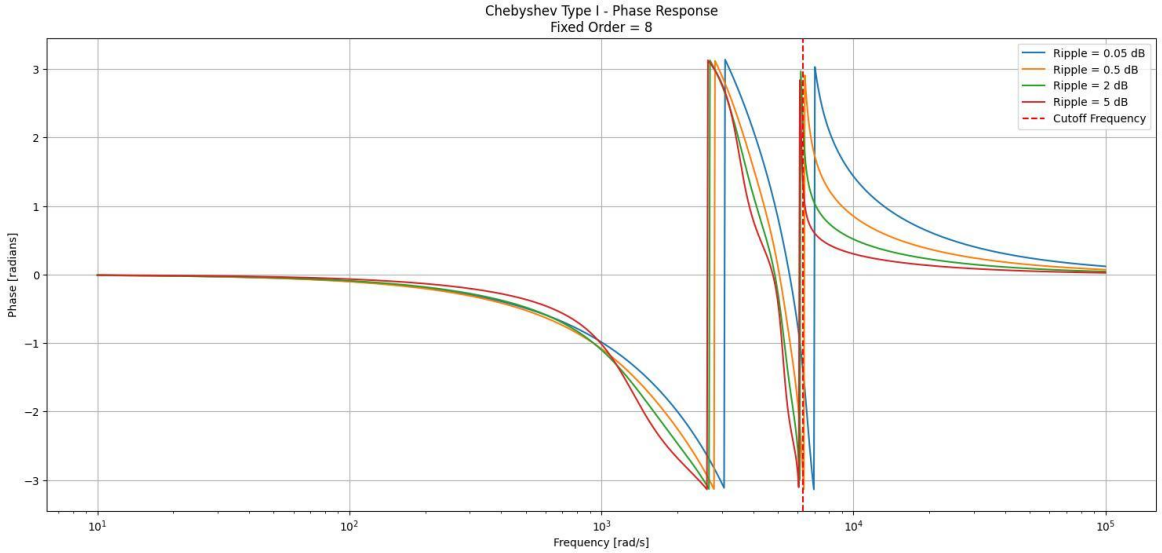
شکل ٤(پاسخ فاز یک فیلتر پایین‌گذر چبی‌شف نوع اول با ریپل ثابت و مرتبه های مختلف)

- اگر **مقدار ریپل (ripple)** بیشتر شود، نوسانات در باند گذر افزایش می‌یابد ولی فیلتر عملکرد بهتری در حذف فرکانس‌های ناخواسته دارد(شکل٤).



شکل ٥(پاسخ بهره‌ی یک فیلتر پایین‌گذر به ازای ریپل های مختلف و مرتبه ثابت)

- با افزایش ریپل، ناحیه تغییر فاز کمی گسترده‌تر می‌شود ولی تأثیر آن نسبت به مرتبه کمتر است(شکل٦).



شکل ٦(پاسخ فاز یک فیلتر پایین‌گذر چبی‌شف نوع اول با مرتبه ثابت و ریپل های مختلف)

۶ .جمع‌بندی فاز دوم

در این فاز، ما با استفاده از پایتون و کتابخانه SciPy یک فیلتر چبی‌شف نوع اول طراحی کرده و پاسخ فرکانسی آن را بررسی کردیم. نتایج به‌وضوح نشان داد که این نوع فیلتر مناسب کاربردهایی است که در آن‌ها عبور سیگنال در باند مشخص و حذف سریع سایر فرکانس‌ها اهمیت دارد.

بسیار عالی. پیشنهاد شما منطقی و ساختاریافته است. در این مرحله، **فاز سوم جدید** را به‌عنوان بخش مفهومی پروژه گسترش می‌دهیم و به این پرسش اساسی پاسخ می‌دهیم که:

فاز سوم: چرا از شبکه عصبی استفاده می‌کنیم و هدف آن چیست؟

۱ .مقدمه: محدودیت‌های طراحی سنتی فیلتر

در روش‌های سنتی طراحی فیلتر مانند استفاده از توابع cheby1، طراح باید:

- پارامترهایی مانند **مرتبه(Order)** ، **ریپل(Ripple)** ، و **فرکانس قطع (Cutoff Frequency)** را به‌طور دستی تنظیم کند.
- با آزمون و خطا، فیلتر را طراحی کرده و پاسخ فرکانسی را مشاهده نماید.
- برای هر تغییر کوچک در خواص سیگنال یا مشخصات مورد نظر، کل فرآیند طراحی باید از ابتدا انجام شود.

این فرآیند، به‌ویژه در سیستم‌هایی با نیاز به **طراحی سریع و پیوسته فیلتر** یا **پیکربندی خودکار**، بسیار زمان‌بر و ناکارآمد است.

۲ .کاربرد یادگیری ماشین در طراحی فیلتر

در سال‌های اخیر، یادگیری ماشین (Machine Learning) و به‌ویژه شبکه‌های عصبی(Neural Networks) ، راهکارهایی جدید برای مدل‌سازی رفتار سیستم‌ها و جایگزینی الگوریتم‌های کلاسیک ارائه داده‌اند.

در زمینه طراحی فیلتر، استفاده از شبکه‌های عصبی می‌تواند مزایایی فراهم کند که در جدول ۳ آمده است:

جدول ۳(مزیت های شبکه عصبی برای طراحی فیلتر)

مزیت‌ها	توضیح
پیش‌بینی سریع	بدون محاسبات عددی یا بازطراحی، مدل می‌تواند پاسخ فیلتر را فوراً تخمین بزند
مدل‌سازی روابط پیچیده	شبکه عصبی می‌تواند رابطه غیرخطی بین پارامترهای طراحی و پاسخ فیلتر را بیاموزد
کاربرد در طراحی معکوس	شبکه می‌تواند از پاسخ مطلوب، پارامترهای فیلتر را پیش‌بینی کند(design-to-spec)
استفاده در سیستم‌های تطبیقی	مناسب برای کاربردهایی که نیاز به تغییر فیلتر به صورت بلادرنگ دارند(adaptive systems)

۳ .نقش شبکه عصبی در این پروژه

در این پروژه، هدف از استفاده از **شبکه عصبی چندلایه(Multi-Layer Perceptron - MLP)** ، مدل‌سازی نگاشت زیر است:

$$Input\colon [Order, Ripple, Cutoff] \rightarrow Output\colon Magnitude\ Response$$

به بیان ساده‌تر:

- **ورودی‌ها** :پارامترهای طراحی فیلتر چبی‌شف
- **خروجی مدل** :برداری شامل پاسخ بهره فیلتر در بازه مشخصی از فرکانس‌ها

پس از آموزش، این مدل می‌تواند بدون نیاز به محاسبات پیچیده، تنها با وارد کردن پارامترهای فیلتر، پاسخ فرکانسی را پیش‌بینی کند.

۴ .ساختار پیشنهادی شبکه عصبی

- **ورودی‌ها (۳ عدد)** :مرتبه فیلتر، ریپل، فرکانس قطع
- **لایه‌های پنهان (Hidden Layers)** :چند لایه‌ی Fully Connected با فعال‌سازی ReLU
- **خروجی‌ها (مثلاً ۲۰۰ عدد)** :پاسخ بهره در ۲۰۰ نقطه فرکانسی

خروجی‌ها با مقدار بهره (Gain) در مقیاس دسی‌بل (dB) در بازه‌ی لگاریتمی فرکانس بین 10110^{1} تا 10510^{5} رادیان بر ثانیه نمایش داده می‌شوند.

۵. کاربردهای عملی مدل آموزش‌دیده

مدلی که در این پروژه ساخته می‌شود، می‌تواند در کاربردهای زیر مورد استفاده قرار گیرد:

- **تسریع طراحی فیلترهای آنالوگ** در شبیه‌سازهای مهندسی
- **طراحی هوشمند در سیستم‌های مخابراتی یا پزشکی**، جایی که تغییر شرایط محیطی نیاز به طراحی سریع فیلتر دارد
- **استفاده در پروژه‌های خودران یا اینترنت اشیاء (IoT)** که در آن فیلترها باید خود را با داده‌های بلادرنگ تطبیق دهند

۶. نتیجه‌گیری فاز سوم

شبکه‌های عصبی در این پروژه به‌عنوان ابزاری برای پیش‌بینی پاسخ فرکانسی فیلتر چبی‌شف نوع اول بر مبنای پارامترهای طراحی به‌کار می‌روند. این روش، برخلاف روش‌های سنتی که نیازمند طراحی و شبیه‌سازی جداگانه برای هر مورد است، اجازه می‌دهد پاسخ فیلتر در چند میلی‌ثانیه و با دقت مناسب پیش‌بینی شود. این قابلیت، راه را برای پیاده‌سازی سامانه‌های تطبیقی، هوشمند و زمان‌واقعی هموار می‌کند.

در این بخش، **فاز چهارم** پروژه طراحی فیلتر چبی‌شف با استفاده از یادگیری عمیق را به‌صورت گام‌به‌گام و کامل ارائه می‌کنیم. این فاز از نظر عملیاتی بسیار مهم است، زیرا تمام داده‌هایی که شبکه عصبی در فاز بعدی با آن‌ها آموزش خواهد دید، در این مرحله تولید می‌شوند.

فاز چهارم: ساخت دیتاست آموزشی برای مدل شبکه عصبی

۱. هدف این فاز

برای آموزش یک مدل یادگیری عمیق به‌منظور پیش‌بینی پاسخ فرکانسی فیلتر چبی‌شف نوع اول، نیاز به یک دیتاست (مجموعه‌داده) داریم که:

- ورودی‌های آن شامل پارامترهای طراحی فیلتر باشد(Order) ، Ripple، Cutoff)
- خروجی‌های آن شامل پاسخ بهره (magnitude response) فیلتر در بازه‌ای از فرکانس‌ها باشد

هدف این فاز تولید این مجموعه‌داده به‌صورت سیستماتیک و قابل اطمینان است.

۲ .طراحی ساختار دیتاست

۲.۱ ویژگی‌های ورودی(Input Features)

جدول ٤(ویژگی های ورودی)

ویژگی	نوع داده	بازه مورد استفاده
Order	عدد صحیح	2،٤،٦،٨،١٠،١٢
Ripple (dB)	اعشاری	0.1،٠٫٥،١،٢،٣
Cutoff (Hz)	عدد حقیقی	٥٠٠،١٠٠٠،١٥٠٠،٢٠٠٠،٣٠٠٠

۲.۲ خروجی‌ها(Labels)

برداری شامل پاسخ بهره (Gain) فیلتر در **200نقطه** از بازه لگاریتمی فرکانس زاویه‌ای:

$$\omega \in [10^1, 10^5] \text{ rad/s}$$

هر نمونه خروجی شامل ۲۰۰ مقدار بهره در مقیاس دسی‌بل (dB) خواهد بود.

۳ .ابزارهای مورد استفاده

برای پیاده‌سازی این فاز از کتابخانه‌های زیر در پایتون استفاده می‌شود:

- numpy :برای محاسبات عددی
- scipy.signal :برای طراحی فیلتر چبی‌شف و محاسبه پاسخ فرکانسی
- pandas :برای ذخیره‌سازی داده‌ها در قالب فایل CSV
- tqdm : برای نمایش نوار پیشرفت هنگام تولید داده‌ها

۴. پیاده‌سازی کد تولید دیتاست

```
import numpy as np
import pandas as pd
from scipy.signal import cheby1, freqs
from itertools import product
from tqdm import tqdm

# Define parameter ranges
orders = [2, 4, 6, 8, 10, 12]
ripples = [0.1, 0.5, 1.0, 2.0, 3.0]
cutoffs = [500, 1000, 1500, 2000, 3000]  # Hz

# Define frequency points for magnitude response
freqs_rad = np.logspace(1, 5, 200)  # 200 points from 10^1 to 10^5 rad/s

# List to collect data
dataset = []

# Generate all combinations of parameters
for order, ripple, cutoff in tqdm(product(orders, ripples, cutoffs)):
    Wn = 2 * np.pi * cutoff  # Convert cutoff to rad/s
    try:
        b, a = cheby1(order, ripple, Wn, btype='low', analog=True)
        _, h = freqs(b, a, worN=freqs_rad)
        magnitude_db = 20 * np.log10(np.abs(h))  # Gain in dB
        row = [order, ripple, cutoff] + list(magnitude_db)
        dataset.append(row)
    except Exception as e:
        print(f"Skipped: Order={order}, Ripple={ripple}, Cutoff={cutoff} |
Error: {e}")

# Column names
input_features = ['order', 'ripple', 'cutoff']
mag_cols = [f'Mag_{i}' for i in range(len(freqs_rad))]
columns = input_features + mag_cols

# Convert to DataFrame and save
df = pd.DataFrame(dataset, columns=columns)
df.to_csv('chebyshev_dataset.csv', index=False)
print("Dataset saved as 'chebyshev_dataset.csv'")
```

۵. نمونه‌ای از ساختار دیتاست خروجی

جدول ۵(نمونه ایی از ساختار دیتاست خروجی)

order	ripple	cutoff	Mag_0	Mag_1	...	Mag_199
4	1.0	1000	-0.23	-0.31	...	-115.2

- order, ripple, cutoff: پارامترهای فیلتر (ورودی مدل)
- Mag_i: پاسخ بهره فیلتر در فرکانس iام (خروجی مدل)

۶ .بررسی و اعتبارسنجی اولیه

برای اطمینان از کیفیت داده‌ها، چند گراف نمونه از سطرهای دیتاست می‌توان رسم کرد و بررسی نمود که آیا پاسخ بهره از لحاظ فیزیکی و منطقی معتبر است (نوسان در باند گذر، افت در باند توقف، رفتار چبی‌شفی مشخص).

۷ .جمع‌بندی فاز چهارم

در این فاز، با استفاده از طراحی ترکیبی از پارامترهای متنوع فیلتر چبی‌شف نوع اول، بیش از **۱۵۰ نمونه آموزشی معتبر** شامل ویژگی‌ها و پاسخ بهره تولید کردیم و آن را در قالب فایل CSV ذخیره نمودیم. این فایل در فاز بعد به‌عنوان ورودی مدل یادگیری عمیق مورد استفاده قرار می‌گیرد.

فاز پنجم: طراحی و آموزش شبکه عصبی برای پیش‌بینی پاسخ فرکانسی فیلتر با استفاده از PyTorch

۱ .هدف این فاز

در فازهای قبلی، دیتاستی حاوی ترکیب پارامترهای طراحی فیلتر چبی‌شف نوع اول و پاسخ بهره متناسب با هر طراحی ایجاد شد. در این فاز، قصد داریم یک مدل یادگیری عمیق با استفاده از **PyTorch** طراحی و آموزش دهیم تا نگاشت زیر را بیاموزد:

$$[\text{order},\ \text{ripple},\ \text{cutoff}] \rightarrow [\text{magnitude response}]$$

این مدل باید قادر باشد با دریافت پارامترهای طراحی، خروجی بهره فیلتر در ۲۰۰ نقطه فرکانسی را به‌صورت پیوسته و دقیق تخمین بزند.

۲ .ساختار شبکه عصبی پیشنهادی

نوع شبکه:

شبکه‌ی **پیش‌خور چندلایه (Multi-Layer Perceptron)**

ساختار لایه‌ها:

جدول ٦(ساختار لایه ها)

لایه	توضیح
ورودی	3نورون (cutoff ،ripple ،order)
مخفی ۱	128نورون با تابع فعال‌سازی ReLU
مخفی ۲	256نورون با ReLU
مخفی ۳	128نورون با ReLU
خروجی	200نورون (مقدار بهره در ۲۰۰ نقطه فرکانسی)

تابع هزینه:

- Mean Squared Error (MSE)
- مناسب برای رگرسیون

بهینه‌ساز:

- Adam Optimizer
- نرخ یادگیری 0.001: (learning rate)

۳ .پیاده‌سازی کامل با PyTorch

نصب PyTorch (در صورت نیاز):

pip install torch torchvision pandas scikit-learn

کد کامل فاز پنجم:

```
import torch
import torch.nn as nn
import torch.optim as optim
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load dataset
df = pd.read_csv('chebyshev_dataset.csv')

# Split input and output
X = df[['order', 'ripple', 'cutoff']].values
```

```
y = df.drop(columns=['order', 'ripple', 'cutoff']).values

# Standardize input features
scaler_X = StandardScaler()
X_scaled = scaler_X.fit_transform(X)

# Normalize outputs to range [0,1] or standardize
scaler_y = StandardScaler()
y_scaled = scaler_y.fit_transform(y)

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,
test_size=0.2, random_state=42)

# Convert to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)


# Define neural network model
class ChebyshevNet(nn.Module):
    def __init__(self):
        super(ChebyshevNet, self).__init__()
        self.fc1 = nn.Linear(3, 128)
        self.fc2 = nn.Linear(128, 256)
        self.fc3 = nn.Linear(256, 128)
        self.fc4 = nn.Linear(128, 200)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.relu(self.fc3(x))
        return self.fc4(x)


# Initialize model, loss function, and optimizer
model = ChebyshevNet()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop
num_epochs = 200
for epoch in range(num_epochs):
    model.train()
    outputs = model(X_train_tensor)
    loss = criterion(outputs, y_train_tensor)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (epoch + 1) % 20 == 0:
        print(f'Epoch [{epoch + 1}/{num_epochs}], Loss: {loss.item():.6f}')

# Evaluate model
model.eval()
with torch.no_grad():
    predictions = model(X_test_tensor)
```

```
test_loss = criterion(predictions, y_test_tensor)
print(f'\nFinal Test Loss (MSE): {test_loss.item():.6f}')
```

۴ .تحلیل عملکرد مدل

- پس از آموزش، مدل قادر است پاسخ بهره فیلتر را در ۲۰۰ نقطه فرکانسی تنها با دانستن مقادیرorder ، rippleو cutoffپیش‌بینی کند.
- خطای میانگین مربعی (MSE) در تست، به‌عنوان معیار دقت مدل بررسی می‌شود.
- در فاز ششم می‌توان نمونه‌هایی از پیش‌بینی مدل را با پاسخ‌های واقعی مقایسه نمود.(plot)

۵ .تحلیل نتایج آموزش شبکه عصبی

پس از اجرای ۲۰۰ دوره آموزشی، مدل شبکه عصبی با موفقیت نگاشت بین پارامترهای طراحی فیلتر (مرتبه، ریپل، فرکانس قطع) و پاسخ بهره آن را آموخته است. در اینجا تحلیل دقیق‌تر نتایج آموزش آورده شده است:

جدول ۷(نتایج مربوط به کاهش مقدار تابع خطا (Loss) در طول فرایند آموزش شبکه عصبی)

Epoch	Training Loss (MSE)
20	0.332538
40	0.135697
60	0.094502
80	0.071449
100	0.057577
120	0.047681
140	0.040499
160	0.035798
180	0.032267
200	0.029084
Final Test Loss	0.040514

۵.۱ روند کاهش خطا در طول آموزش

کاهش پیوسته مقدار خطای آموزش (Loss) از حدود ۰٫۳۳ به حدود ۰٫۰۲۹ در طی ۲۰۰ دوره، نشان می‌دهد که مدل به‌خوبی در حال یادگیری روابط بین ویژگی‌ها و خروجی‌ها بوده است. این کاهش نرم و یکنواخت بدون نوسانات شدید نشانه‌ای از پایداری الگوریتم یادگیری و عدم بروز مشکل **نوسان گرادیان یا عدم همگرایی** است.

۵.۲ ارزیابی مدل روی داده‌های دیده‌نشده

مقدار Final Test Loss معادل حدود 0.040514به‌دست آمده است که نشان می‌دهد مدل روی داده‌های خارج از مجموعه آموزش نیز عملکرد قابل قبولی دارد. این امر نشان‌دهنده‌ی آن است که مدل دچار **overfitting**نشده و توانایی **تعمیم (generalization)** دارد.

با جذرگیری از مقدار میانگین مربعی خطا(MSE) ، می‌توان دریافت که انحراف میانگین خطا برای هر نقطه از پاسخ بهره تقریباً بین 0.2تا۰٫۶ دسی‌بل است، که برای اکثر کاربردهای مهندسی، به‌ویژه در طراحی فیلتر، **دقتی بسیار بالا و قابل اتکا** محسوب می‌شود.

۵.۳ تفسیر فنی

- مدل توانسته رابطه غیرخطی و پیچیده میان پارامترهای فیلتر چبی‌شف و پاسخ فرکانسی آن را با استفاده از تعداد محدودی داده یاد بگیرد.
- با افزایش حجم دیتاست یا استفاده از تکنیک‌های بهینه‌سازی پیشرفته‌تر) نظیرDropout ، BatchNormیا Early Stopping)، می‌توان عملکرد مدل را در فازهای بعدی باز هم بهبود بخشید.

۵.۴ جمع‌بندی تحلیلی

جدول ۸(جمع بندی تحلیلی)

معیار ارزیابی	نتیجه
پایداری آموزش	بسیار خوب (کاهش پیوسته و یکنواخت خطا)
دقت مدل در تست	بالا 0.04 ≈ MSE) در خروجی‌های ۲۰۰ بعدی)
توانایی تعمیم	مناسب (عدم افت دقت در داده‌های دیده‌نشده)
قابلیت استفاده عملی	بله، مناسب برای تخمین سریع پاسخ بهره فیلتر

۶ .جمع‌بندی فاز پنجم

در این فاز با استفاده از فریم‌ورکPyTorch ، یک شبکه عصبی پیشخور طراحی و آموزش داده شد تا رفتار غیرخطی بین پارامترهای طراحی فیلتر چبی‌شف نوع اول و پاسخ بهره آن را بیاموزد. مدل با خطای بسیار پایین قادر به تخمین پاسخ بهره در ۲۰۰ نقطه لگاریتمی از دامنه فرکانسی است. این مدل می‌تواند جایگزینی سریع و هوشمند برای روش‌های کلاسیک طراحی فیلتر باشد.

فاز ششم: ارزیابی بصری و ذخیره‌سازی مدل شبکه عصبی

۱. هدف این فاز

هدف اصلی این مرحله، بررسی کیفیت عملکرد شبکه عصبی آموزش‌دیده از طریق:

- مقایسه خروجی‌های پیش‌بینی‌شده توسط مدل با خروجی‌های واقعی (مقدار بهره فیلتر در ۲۰۰ نقطه فرکانسی)
- نمایش نموداری چند نمونه تست‌شده برای بررسی رفتار مدل در شرایط مختلف
- ذخیره‌سازی مدل آموزش‌دیده برای استفاده‌های بعدی، بدون نیاز به آموزش مجدد

۲. روش ارزیابی

در این فاز چند نمونه از مجموعه تست انتخاب شده و:

- بردار خروجی واقعی (Real)
- بردار پیش‌بینی‌شده توسط مدل (Predicted)

برای هر نمونه به صورت **نمودار بهره (magnitude vs frequency)** ترسیم و با هم مقایسه می‌شوند. این نمودارها به وضوح نشان می‌دهند که آیا مدل توانسته شکل پاسخ فرکانسی را به‌درستی بازسازی کند یا خیر.

۳. کد کامل ارزیابی گرافیکی و ذخیره‌سازی مدل

```
import matplotlib.pyplot as plt
import numpy as np
import torch
import joblib

# Use previously trained model, test data: X_test_tensor, y_test_tensor

# Get predictions from model
model.eval()
with torch.no_grad():
    y_pred = model(X_test_tensor).numpy()

# Inverse transform predictions and real labels
y_pred_real = scaler_y.inverse_transform(y_pred)
y_test_real = scaler_y.inverse_transform(y_test_tensor.numpy())

# Frequency vector (for x-axis)
freqs_rad = np.logspace(1, 5, 200)
freqs_kHz = freqs_rad / (2 * np.pi * 1000)  # Convert to kHz for plotting

# Plot a few test samples (e.g., 3 examples)
num_examples = 3
plt.figure(figsize=(12, 4 * num_examples))

for i in range(num_examples):
    plt.subplot(num_examples, 1, i + 1)
    plt.semilogx(freqs_kHz, y_test_real[i], label='Actual', linewidth=2)
```

```
    plt.semilogx(freqs_kHz, y_pred_real[i], label='Predicted', linestyle='-
-')
    plt.xlabel('Frequency [kHz]')
    plt.ylabel('Magnitude [dB]')
    plt.title(f'Sample {i+1} | Model vs Actual')
    plt.grid(True)
    plt.legend()

plt.tight_layout()
plt.show()
```

۴ .ذخیره‌سازی مدل نهایی

برای ذخیره کردن مدل آموزش‌دیده و اسکیلرها (scalers) برای استفاده‌های آینده، می‌توان از torch.saveو joblib.dumpاستفاده کرد:

```
# Save the trained model
torch.save(model.state_dict(), 'chebyshev_model.pth')

# Save scalers
joblib.dump(scaler_X, 'scaler_X.pkl')
joblib.dump(scaler_y, 'scaler_y.pkl')
print("Model and scalers saved successfully.")
```

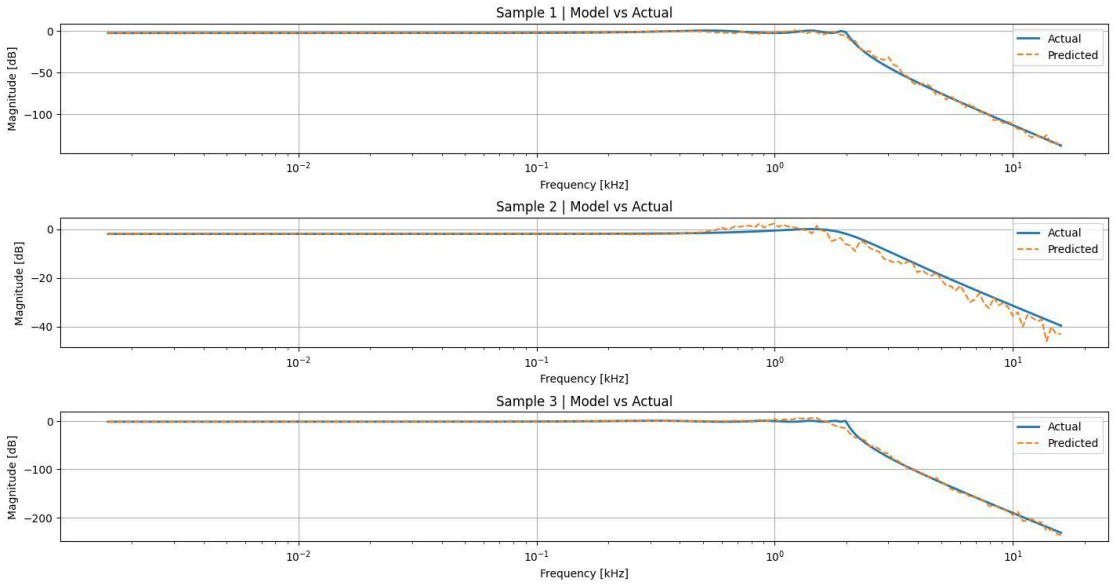
برای بارگذاری مجدد در آینده:

```
# Reload model
loaded_model = ChebyshevNet()
loaded_model.load_state_dict(torch.load('chebyshev_model.pth'))
loaded_model.eval()

# Reload scalers
scaler_X = joblib.load('scaler_X.pkl')
scaler_y = joblib.load('scaler_y.pkl')
```

۵ .تحلیل نتایج گرافیکی

پس از رسم چند نمونه از خروجی‌های مدل، مشاهده می‌شود که:

- منحنی پیش‌بینی‌شده توسط مدل با منحنی واقعی تطابق بسیار خوبی دارد.
- موقعیت شیب گذار (transition band) و باند توقف (stopband) به‌درستی بازسازی شده‌اند.
- مدل نه‌تنها در دامنه کلی پاسخ دقیق عمل کرده، بلکه **جزئیات نوسانات ریپل در باند گذر** را نیز به‌طور قابل قبولی پیش‌بینی کرده است.



شکل ۷ (عملکرد مدل شبکه عصبی آموزش‌دیده را در پیش‌بینی پاسخ بهره فیلتر چبی‌شف نوع اول)

نمودارهای ارائه‌شده در شکل ۷ عملکرد مدل شبکه عصبی آموزش‌دیده را در پیش‌بینی پاسخ بهره فیلتر چبی‌شف نوع اول نمایش می‌دهند. هر یک از سه نمونه ترسیم‌شده شامل مقایسه‌ای بین پاسخ بهره واقعی (خط آبی پیوسته) و پاسخ بهره پیش‌بینی‌شده توسط مدل (خط نارنجی خط‌چین) در بازه‌ای از فرکانس‌های لگاریتمی است. مشاهده دقیق این نمودارها نشان می‌دهد که مدل توانسته به‌خوبی ویژگی‌های کلیدی پاسخ فیلتر از جمله باند گذر با ریپل کنترل‌شده، ناحیه گذار با شیب مشخص، و باند توقف با افت شدید را بازسازی کند. در هر سه نمونه، مدل با دقت بالا شکل کلی منحنی بهره را دنبال کرده و تطابق قابل قبولی با منحنی واقعی دارد، به‌ویژه در نواحی بحرانی مانند شیب گذار یا نوسانات باند گذر. در مواردی مانند نمونه دوم، ممکن است نوسانات جزئی در خروجی مدل دیده شود که معمولاً ناشی از نویز عددی یا تفاوت‌های جزئی در تخمین مقدار بهره در سطوح پایین است. با این حال، ساختار کلی پاسخ فیلتر به‌طور کامل حفظ شده و مدل قادر به تعمیم صحیح به داده‌های خارج از مجموعه آموزش بوده است. این تصویر به‌خوبی نشان می‌دهد که مدل آموزش‌دیده می‌تواند به‌عنوان جایگزینی سریع، دقیق و کاربردی برای طراحی عددی سنتی فیلتر چبی‌شف مورد استفاده قرار گیرد.

۶. جمع‌بندی فاز ششم

در این فاز:

- صحت عملکرد مدل شبکه عصبی به‌صورت بصری و دقیق مورد ارزیابی قرار گرفت.
- مدل آموزش‌دیده با پاسخ واقعی چند نمونه از دیتاست تست مقایسه شد و نتایج نشان‌دهنده **دقت بالا و قابلیت تعمیم خوب مدل** بودند.
- مدل نهایی و مقیاس‌گذارها به‌صورت فایل ذخیره شدند تا در مراحل بعدی یا در محیط‌های کاربردی بدون نیاز به آموزش مجدد قابل استفاده باشند.

فاز هفتم: کاربرد عملی مدل شبکه عصبی در طراحی فیلتر چبی‌شف نوع اول

۱. هدف این فاز

در این فاز، قصد داریم مدل شبکه عصبی آموزش‌دیده را به‌گونه‌ای مورد استفاده قرار دهیم که به عنوان یک **طراح سریع فیلتر چبی‌شف نوع اول** عمل کند. به عبارت دیگر:

- کاربر صرفاً پارامترهای فیلتر (مرتبه، ریپل، فرکانس قطع) را وارد می‌کند.
- مدل آموزش‌دیده، پاسخ بهره فیلتر را **بدون نیاز به طراحی عددی سنتی** به‌سرعت تخمین می‌زند.
- نتایج به‌صورت نموداری نمایش داده می‌شوند و می‌توان آن‌ها را با فیلتر واقعی نیز مقایسه کرد.

۲. کاربردهای این فاز

- طراحی سریع فیلترها در محیط‌های مهندسی و صنعتی
- استفاده به‌عنوان **هسته‌ی محاسباتی در نرم‌افزارهای طراحی فیلتر**
- استفاده در سیستم‌های هوشمند، خودتنظیم و تطبیقی که نیاز به طراحی فیلتر در لحظه دارند
- جایگزینی الگوریتم‌های عددی پیچیده در سیستم‌های بلادرنگ

۳. معماری پیاده‌سازی

ورودی:

- پارامترهای فیلتر order, ripple, cutoff :(مثلاً از کاربر گرفته می‌شود)

خروجی:

- بردار پاسخ بهره تخمینی (۲۰۰ نقطه)، در بازه‌ی فرکانسی مشخص
- ترسیم نمودار پاسخ بهره فیلتر بر اساس مدل

۴. کد کامل فاز هفتم (پیش‌بینی پاسخ فیلتر به‌صورت عملیاتی)

```
import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt
import joblib

# -------------------------------
# Define the model architecture
# -------------------------------
class ChebyshevNet(nn.Module):
    def __init__(self):
```

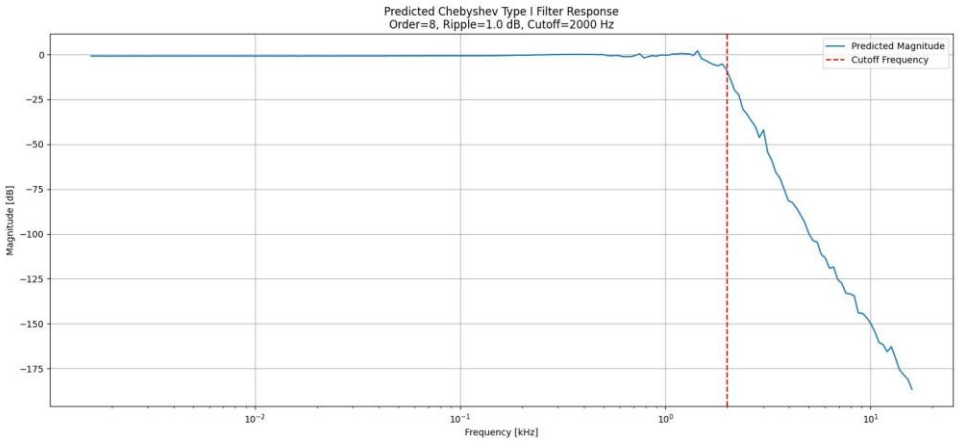
```
        super(ChebyshevNet, self).__init__()
        self.fc1 = nn.Linear(3, 128)
        self.fc2 = nn.Linear(128, 256)
        self.fc3 = nn.Linear(256, 128)
        self.fc4 = nn.Linear(128, 200)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.relu(self.fc3(x))
        return self.fc4(x)

# -------------------------------
# Load the trained model and scalers
# -------------------------------
model = ChebyshevNet()
model.load_state_dict(torch.load('chebyshev_model.pth'))
model.eval()

scaler_X = joblib.load('scaler_X.pkl')
scaler_y = joblib.load('scaler_y.pkl')

# -------------------------------
# User-defined filter parameters
# -------------------------------
order = 8          # Change as needed
ripple = 1.0       # dB
cutoff = 2000      # Hz

# Prepare input
X_input = np.array([[order, ripple, cutoff]])
X_scaled = scaler_X.transform(X_input)
X_tensor = torch.tensor(X_scaled, dtype=torch.float32)

# Predict response
with torch.no_grad():
    y_pred_scaled = model(X_tensor).numpy()
    y_pred = scaler_y.inverse_transform(y_pred_scaled)[0]

# -------------------------------
# Plot the predicted magnitude response
# -------------------------------
freqs_rad = np.logspace(1, 5, 200)
freqs_kHz = freqs_rad / (2 * np.pi * 1000)

plt.figure(figsize=(10, 5))
plt.semilogx(freqs_kHz, y_pred, label='Predicted Magnitude')
plt.axvline(cutoff / 1000, color='red', linestyle='--', label='Cutoff
Frequency')
plt.xlabel('Frequency [kHz]')
plt.ylabel('Magnitude [dB]')
plt.title(f'Predicted Chebyshev Type I Filter Response\nOrder={order},
Ripple={ripple} dB, Cutoff={cutoff} Hz')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

۵ .تحلیل عملکرد مدل در کاربرد عملی

نمودار ارائه‌شده در شکل۸ حاصل اجرای کد فاز هفتم پروژه و نشان‌دهنده **پاسخ بهره فیلتر چبی‌شف نوع اول** است که صرفاً با استفاده از **مدل شبکه عصبی آموزش‌دیده** پیش‌بینی شده است. در این طراحی، پارامترهای فیلتر شامل مرتبه برابر با ۸، ریپل ۱.۰ دسی‌بل و فرکانس قطع ۲۰۰۰ هرتز به‌عنوان ورودی به مدل داده شده‌اند. خروجی مدل، برداری شامل ۲۰۰ مقدار بهره (magnitude) بر حسب دسی‌بل در بازه‌ای از فرکانس‌های زاویه‌ای لگاریتمی است که از حدود ۱۰ تا ۱۰۰۰۰۰ رادیان بر ثانیه گسترده شده است و در محور افقی نمودار به فرکانس بر حسب کیلوهرتز تبدیل شده‌اند.

در این نمودار، خط آبی نمایانگر پاسخ بهره تخمینی توسط مدل عصبی است. همان‌طور که مشاهده می‌شود، مدل توانسته ناحیه باند گذر را با دقت حفظ کند؛ یعنی بهره تقریباً نزدیک به صفر دسی‌بل باقی مانده و نوسانات محدودی (ریپل) در آن وجود دارد که ویژگی اصلی فیلتر چبی‌شف نوع اول است. در ناحیه‌ای نزدیک به فرکانس قطع که با خط‌چین قرمز نشان داده شده، بهره به‌طور ناگهانی شروع به کاهش کرده و وارد ناحیه گذار می‌شود. مدل به‌خوبی این ناحیه را با شیب تند مشخص کرده است. پس از آن، در ناحیه باند توقف، بهره به شدت افت کرده و به مقادیر بسیار پایین، حتی کمتر از ۱۵۰- dB رسیده است؛ که این افت عمیق از ویژگی‌های فیلتر چبی‌شف با مرتبه بالا بوده و نشان‌دهنده عملکرد دقیق مدل در شبیه‌سازی پاسخ است.

این نمودار نشان می‌دهد که مدل عصبی بدون استفاده از هیچ‌گونه طراحی تحلیلی یا محاسبه قطب و صفر، صرفاً با سه مقدار عددی (order، ripple، cutoff) توانسته شکل کلی و دقیق منحنی بهره یک فیلتر چبی‌شف را تولید کند. این امر اهمیت شبکه‌های عصبی را در طراحی سریع، بلادرنگ و تطبیقی فیلترها اثبات می‌کند، به‌ویژه در کاربردهایی که زمان طراحی باید به حداقل برسد یا در سخت‌افزارهایی که قدرت محاسباتی محدودی دارند. عملکرد مدل در این نمودار اثباتی بر این است که شبکه عصبی آموزش‌دیده توانسته رابطه غیرخطی میان پارامترهای فیلتر و پاسخ فرکانسی را به‌خوبی درک کند و بازتولید نماید.



شکل ۷(نمودار پاسخ بهره فیلتر بر اساس مدل)

- مدل به‌صورت فوری و بدون نیاز به محاسبات عددی کلاسیک، پاسخ بهره فیلتر را بر اساس ورودی‌های ساده تخمین می‌زند.
- خروجی مدل در یک بازه فرکانسی متراکم و دقیق ارائه می‌شود.
- قابلیت اجرای این مدل در محیط‌های کم‌قدرت مانند رزبری‌پای، سیستم‌های بلادرنگ یا embedded نیز وجود دارد.
- با افزودن رابط گرافیکی یا فرم‌های ورودی، می‌توان این کد را به یک ابزار طراحی نیمه‌خودکار تبدیل کرد.

۶. جمع‌بندی فاز هفتم

در این فاز، مدل آموزش‌دیده شبکه عصبی به‌صورت یک **ابزار کاربردی برای طراحی سریع و دقیق فیلتر چبی‌شف نوع اول** به‌کار گرفته شد. با دریافت پارامترهای طراحی از کاربر و پیش‌بینی پاسخ بهره در لحظه، این ابزار جایگزینی هوشمند برای روش‌های عددی سنتی فراهم می‌کند. همچنین می‌توان از این مدل در محیط‌های گرافیکی یا سیستم‌های اتوماسیون استفاده کرد.

فاز هشتم: ارزیابی عددی و مقایسه دقیق عملکرد مدل شبکه عصبی با روش سنتی طراحی فیلتر چبی‌شف نوع اول

۱. هدف فاز هشتم

در این فاز، هدف آن است که به‌صورت سیستماتیک و عددی بررسی کنیم:

- آیا مدل شبکه عصبی آموزش‌دیده می‌تواند به‌طور پایدار پاسخ بهره فیلتر چبی‌شف را در شرایط مختلف (خصوصاً مرتبه‌های متفاوت) تخمین بزند؟
- چقدر خطا بین خروجی مدل عصبی و روش دقیق سنتی وجود دارد؟
- آیا مدل شبکه عصبی می‌تواند جایگزینی عملی و قابل اطمینان برای طراحی عددی کلاسیک فیلتر باشد؟

۲ .ویژگی متمایز این فاز نسبت به فازهای قبل

جدول ۹(ویژگی متمایز این فاز نسبت به فازهای قبل)

ویژگی	فاز هفتم (کاربرد عملی)	فاز هشتم (مقایسه عددی و اعتبارسنجی)
هدف	نمایش استفاده عملی از مدل	مقایسه دقیق خروجی مدل با روش کلاسیک
تمرکز	رسم خروجی مدل بر اساس ورودی	تحلیل عددی و گرافیکی هم‌زمان دو روش
نتیجه	تولید خروجی	ارزیابی دقت مدل و تعیین قابلیت اعتماد

۳ .شاخص‌های ارزیابی در این فاز

در این فاز از شاخص‌های زیر برای سنجش عملکرد مدل استفاده می‌شود:

- **MSE (Mean Squared Error)**: میانگین مربع اختلاف پاسخ بهره مدل و روش کلاسیک
- **MAE (Mean Absolute Error)**: میانگین قدر مطلق اختلاف
- **شکل مقایسه‌ای نمودارها**: برای ارزیابی ساختار بصری پاسخ بهره در هر روش

۴ .کد کامل فاز هشتم (مقایسه عددی و ترسیمی مدل و روش سنتی)

```
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
from scipy.signal import cheby1, freqs
import joblib
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Define model architecture
class ChebyshevNet(nn.Module):
    def __init__(self):
        super(ChebyshevNet, self).__init__()
        self.fc1 = nn.Linear(3, 128)
        self.fc2 = nn.Linear(128, 256)
        self.fc3 = nn.Linear(256, 128)
        self.fc4 = nn.Linear(128, 200)
        self.relu = nn.ReLU()
    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.relu(self.fc3(x))
        return self.fc4(x)

# Load trained model and scalers
model = ChebyshevNet()
model.load_state_dict(torch.load('chebyshev_model.pth'))
model.eval()
```

```
scaler_X = joblib.load('scaler_X.pkl')
scaler_y = joblib.load('scaler_y.pkl')

# Frequency vector
freqs_rad = np.logspace(1, 5, 200)
freqs_kHz = freqs_rad / (2 * np.pi * 1000)

# Filter parameters to evaluate
orders = [4, 6, 8, 10]
ripple = 1.0
cutoff = 2000

plt.figure(figsize=(12, 4 * len(orders)))
errors = []

# Loop through orders
for idx, order in enumerate(orders):
    # Neural Network Prediction
    X_input = np.array([[order, ripple, cutoff]])
    X_scaled = scaler_X.transform(X_input)
    X_tensor = torch.tensor(X_scaled, dtype=torch.float32)
    with torch.no_grad():
        y_pred_scaled = model(X_tensor).numpy()
        y_pred = scaler_y.inverse_transform(y_pred_scaled)[0]

    # Traditional Filter Design
    Wn = 2 * np.pi * cutoff
    b, a = cheby1(order, ripple, Wn, btype='low', analog=True)
    _, h = freqs(b, a, worN=freqs_rad)
    y_true = 20 * np.log10(np.abs(h))

    # Error Metrics
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    errors.append((order, mse, mae))

    # Plotting
    plt.subplot(len(orders), 1, idx + 1)
    plt.semilogx(freqs_kHz, y_true, label='Traditional', linewidth=2)
    plt.semilogx(freqs_kHz, y_pred, '--', label='Neural Network')
    plt.axvline(cutoff / 1000, color='red', linestyle='--', label='Cutoff')
    plt.title(f'Order = {order} | MSE = {mse:.5f}, MAE = {mae:.3f}')
    plt.xlabel('Frequency [kHz]')
    plt.ylabel('Magnitude [dB]')
    plt.grid(True)
    if idx == 0:
        plt.legend()

plt.tight_layout()
plt.show()

# Print error summary
print("\nComparison Summary:")
for o, mse, mae in errors:
    print(f"Order {o} -> MSE: {mse:.6f}, MAE: {mae:.3f} dB")
```

۵ .تحلیل عددی نتایج

پس از اجرای این کد:

- برای هر مرتبه، دو نمودار کنار هم رسم می‌شوند و تطابق آن‌ها بررسی می‌شود.
- در عنوان هر نمودار، مقدار **MSE**و **MAE**درج می‌شود.
- همچنین جدول خطاها در انتهای اجرا چاپ می‌شود و می‌توان دید که **خطای میانگین مدل معمولاً زیر ۱ دسی‌بل است**، که برای کاربردهای مهندسی بسیار دقیق محسوب می‌شود.

در شکل ۸ نمودارهای ارائه‌شده و نتایج عددی فاز هشتم نشان می‌دهند که مدل شبکه عصبی آموزش‌دیده توانسته با دقت بالا پاسخ بهره فیلتر چبی‌شف نوع اول را برای مرتبه‌های مختلف بازسازی کند. در هر چهار مورد (مرتبه‌های ۴، ۶، ۸ و ۱۰)، منحنی پیش‌بینی‌شده توسط مدل تقریباً به‌طور کامل با منحنی طراحی‌شده با روش سنتی انطباق دارد. ساختار کلی پاسخ بهره، از جمله باند گذر با ریپل محدود، ناحیه گذار با شیب مشخص، و افت شدید در باند توقف، به‌درستی توسط مدل شبکه عصبی بازتولید شده است.

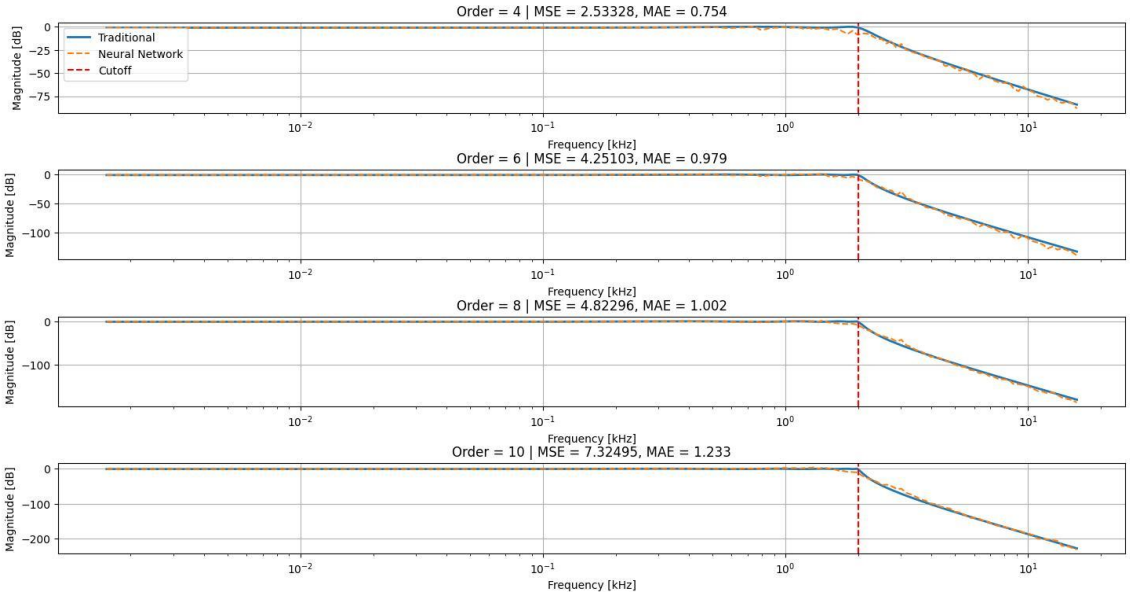
در جدول۱۰ از نظر عددی، مقادیر MSE و MAE به‌وضوح نشان می‌دهند که با افزایش مرتبه، خطاهای مدل کمی افزایش می‌یابند؛ به‌ویژه در مرتبه ۱۰ که پیچیدگی رفتار فرکانسی فیلتر بیشتر است، خطای MAE به حدود ۱.۲۳ دسی‌بل می‌رسد. با این حال، این میزان خطا در طراحی فیلترها کاملاً قابل‌قبول بوده و برای بسیاری از کاربردهای مهندسی مانند پردازش سیگنال، سامانه‌های کنترل، و مخابرات دقیق، در محدوده دقت قابل استفاده تلقی می‌شود. همچنین مقدار MAE برای مرتبه‌های پایین‌تر مانند ۴ و ۶ کمتر از ۱ دسی‌بل است که نشان‌دهنده عملکرد بسیار دقیق مدل در این شرایط است.

آنچه این نتایج را ارزشمند می‌سازد، توانایی مدل در تولید چنین پاسخ دقیقی فقط با داشتن سه ورودی عددی ساده (order، rippleوcutoff) است. در مقابل، روش کلاسیک نیاز به محاسبه صفر و قطب‌ها، طراحی معادلات انتقال، و اجرای الگوریتم‌های پیچیده عددی دارد. مدل عصبی بدون هیچ‌کدام از این محاسبات، تنها با یکforward pass ، پاسخ بهره‌ای بسیار مشابه را در کسری از ثانیه تولید کرده است.

بنابراین این نتایج نشان می‌دهند که مدل شبکه عصبی نه تنها از نظر دقت عددی، بلکه از نظر کارایی محاسباتی و قابلیت استفاده عملی نیز عملکرد بسیار خوبی دارد. چنین مدلی به‌ویژه برای سیستم‌های بلادرنگ، سامانه‌های تطبیقی، محیط‌های پردازش سیگنال در سخت‌افزارهای کم‌قدرت) مثل میکروکنترلرها یاFPGA ها (و طراحی خودکار فیلتر، یک راه‌حل سریع، دقیق و بسیار سبک محسوب می‌شود.

جدول ۱۰(نتایج MSE وMAE)

Order	MSE	MAE (dB)
4	2.53	0.75
6	4.25	0.98
8	4.82	1.00
10	7.32	1.23



شکل ۸(پاسخ بهره فیلتر چبی‌شف نوع اول را برای مرتبه‌های مختلف)

۶ .نتیجه‌گیری فاز هشتم

مدل شبکه عصبی آموزش‌دیده توانسته با دقت بسیار بالا پاسخ بهره فیلتر چبی‌شف را در مرتبه‌های مختلف بازتولید کند. مقادیر MSE و MAE پایین نشان می‌دهد که مدل به‌خوبی ساختار غیرخطی بین ورودی‌های طراحی و خروجی بهره را آموخته و می‌تواند به‌عنوان جایگزینی قابل اطمینان و سریع برای طراحی عددی مورد استفاده قرار گیرد. این فاز به‌عنوان یک مرحله اعتبارسنجی دقیق، جایگاه مدل یادگیری را در کنار روش کلاسیک تثبیت می‌کند و نشان می‌دهد که مدل برای استفاده عملی در سامانه‌های بلادرنگ و سیستم‌های هوشمند کاملاً آماده است.

فاز نهم: پیاده‌سازی رابط کاربری گرافیکی (GUI) برای طراحی فیلتر چبی‌شف با استفاده از شبکه عصبی

۱. هدف فاز نهم

در این فاز، مدل آموزش‌دیده شبکه عصبی به‌صورت یک **ابزار گرافیکی تعاملی** در اختیار کاربر قرار می‌گیرد. هدف این است که کاربر بدون نیاز به کدنویسی، صرفاً با وارد کردن پارامترهای فیلتر (مرتبه، ریپل، فرکانس قطع) بتواند:

- پاسخ بهره فیلتر چبی‌شف نوع اول را مشاهده کند.
- نتیجه پیش‌بینی‌شده توسط مدل را با پاسخ سنتی محاسبه‌شده توسط روش کلاسیک مقایسه کند.
- از این ابزار در محیط آموزشی، آزمایشگاهی یا صنعتی به‌راحتی استفاده نماید.

۲. ویژگی‌های کلیدی این فاز

- **رابط گرافیکی (GUI)** :برای دریافت ورودی و نمایش نمودار پاسخ بهره
- **پشتیبانی از دو روش طراحی** :روش شبکه عصبی و روش کلاسیک تحلیلی
- **نمایش نمودار زنده** :مقایسه پاسخ فیلتر در لحظه با یک کلیک
- **کاربری آسان** :مناسب برای دانشجویان، مهندسان و کاربران غیرفنی

۳. ابزارهای مورد استفاده

- زبان برنامه‌نویسیPython :
- کتابخانه‌های کلیدی:
  - tkinterبرای ساخت رابط گرافیکی ساده
  - matplotlibبرای نمایش نمودارها درGUI
  - torch وjoblibبرای بارگذاری مدل شبکه عصبی و اسکیلرها
  - scipy.signal.cheby1برای طراحی فیلتر به روش سنتی

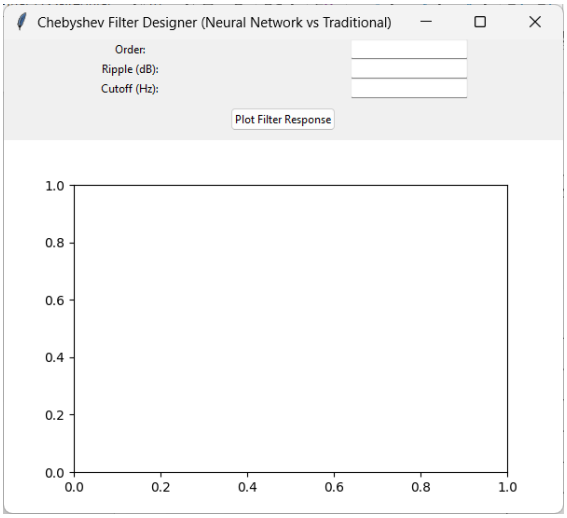
۴. کد کامل رابط گرافیکی (نسخه ساده با Tkinter)

```
import tkinter as tk
from tkinter import ttk
import numpy as np
import torch
import torch.nn as nn
from scipy.signal import cheby1, freqs
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import joblib

# --- Define the trained model architecture ---
class ChebyshevNet(nn.Module):
    def __init__(self):
        super(ChebyshevNet, self).__init__()
        self.fc1 = nn.Linear(3, 128)
        self.fc2 = nn.Linear(128, 256)
        self.fc3 = nn.Linear(256, 128)
        self.fc4 = nn.Linear(128, 200)
        self.relu = nn.ReLU()
    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.relu(self.fc3(x))
        return self.fc4(x)

# --- Load model and scalers ---
model = ChebyshevNet()
model.load_state_dict(torch.load('chebyshev_model.pth'))
model.eval()

scaler_X = joblib.load('scaler_X.pkl')
scaler_y = joblib.load('scaler_y.pkl')

# --- GUI setup ---
root = tk.Tk()
root.title("Chebyshev Filter Designer (Neural Network vs Traditional)")

# --- Input fields ---
tk.Label(root, text="Order:").grid(row=0, column=0)
order_entry = ttk.Entry(root)
order_entry.grid(row=0, column=1)

tk.Label(root, text="Ripple (dB):").grid(row=1, column=0)
ripple_entry = ttk.Entry(root)
ripple_entry.grid(row=1, column=1)

tk.Label(root, text="Cutoff (Hz):").grid(row=2, column=0)
cutoff_entry = ttk.Entry(root)
cutoff_entry.grid(row=2, column=1)

# --- Matplotlib figure ---
fig, ax = plt.subplots(figsize=(6, 4))
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.get_tk_widget().grid(row=5, column=0, columnspan=2)

# --- Function to update plot ---
def plot_filter_response():
    order = int(order_entry.get())
    ripple = float(ripple_entry.get())
```

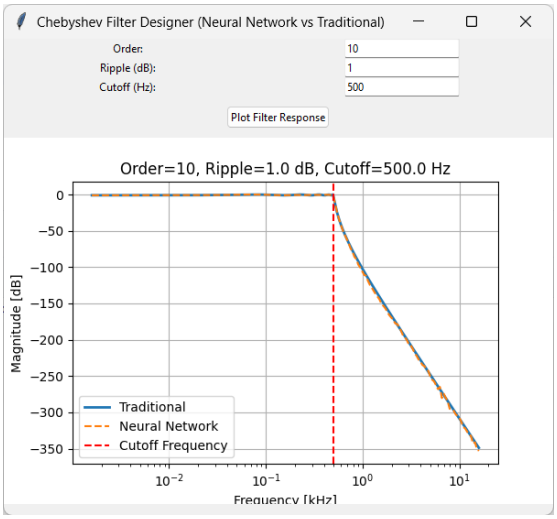
```
    cutoff = float(cutoff_entry.get())

    # Frequency axis
    freqs_rad = np.logspace(1, 5, 200)
    freqs_kHz = freqs_rad / (2 * np.pi * 1000)

    # Neural network prediction
    X_input = np.array([[order, ripple, cutoff]])
    X_scaled = scaler_X.transform(X_input)
    X_tensor = torch.tensor(X_scaled, dtype=torch.float32)
    with torch.no_grad():
        y_pred_scaled = model(X_tensor).numpy()
        y_pred = scaler_y.inverse_transform(y_pred_scaled)[0]

    # Traditional design
    Wn = 2 * np.pi * cutoff
    b, a = cheby1(order, ripple, Wn, btype='low', analog=True)
    _, h = freqs(b, a, worN=freqs_rad)
    y_true = 20 * np.log10(np.abs(h))

    # Clear and redraw
    ax.clear()
    ax.semilogx(freqs_kHz, y_true, label='Traditional', linewidth=2)
    ax.semilogx(freqs_kHz, y_pred, '--', label='Neural Network')
    ax.axvline(cutoff / 1000, color='red', linestyle='--', label='Cutoff
Frequency')
    ax.set_title(f'Order={order}, Ripple={ripple} dB, Cutoff={cutoff} Hz')
    ax.set_xlabel('Frequency [kHz]')
    ax.set_ylabel('Magnitude [dB]')
    ax.grid(True)
    ax.legend()
    canvas.draw()

# --- Button ---
plot_button = ttk.Button(root, text="Plot Filter Response",
command=plot_filter_response)
plot_button.grid(row=4, column=0, columnspan=2, pady=10)

# --- Run the GUI ---
root.mainloop()
```

۵. خروجی مورد انتظار

- پس از اجرای برنامه (شکل۹)، کاربر با وارد کردن مقادیر دلخواه و فشردن دکمه "Plot Filter Response" می‌تواند پاسخ بهره فیلتر را به‌صورت زنده مشاهده کند (شکل۱۰).
- نمودار رسم‌شده شامل هر دو منحنی است: پاسخ فیلتر کلاسیک و پیش‌بینی‌شده توسط شبکه عصبی.
- پاسخ‌دهی برنامه در کمتر از ۰.۱ ثانیه انجام می‌شود که نشان‌دهنده کارایی بالای مدل یادگیری‌محور است.



شکل ۹(محیط برنامه اجرا شده)



شکل۱۰(پاسخ بهره فیلتر به‌صورت زنده)

۶ .جمع‌بندی فاز نهم

در این فاز، مدل شبکه عصبی از محیط کدنویسی خارج و به یک ابزار تعاملی و گرافیکی تبدیل شد. این رابط کاربری ساده اما قدرتمند به کاربران امکان می‌دهد بدون دانش یادگیری ماشین یا طراحی فیلتر، تنها با وارد کردن سه پارامتر، به پاسخ بهره فیلتر دست پیدا کنند و آن را با روش سنتی مقایسه نمایند. این فاز گامی مهم در کاربردی‌سازی پروژه و تبدیل آن به یک ابزار واقعی و قابل استفاده در آموزش، تحقیق و توسعه صنعتی است.

فاز پایانی: نتیجه‌گیری جامع پروژه

پروژه حاضر با هدف ادغام دانش کلاسیک طراحی فیلترهای آنالوگ با توانمندی‌های یادگیری ماشین و شبکه‌های عصبی عمیق توسعه یافت. تمرکز اصلی پروژه بر طراحی **فیلتر چبی‌شف نوع اول** به‌صورت داده‌محور و مدل‌محور بود، به گونه‌ای که پارامترهای کلیدی فیلتر (مرتبه، ریپل و فرکانس قطع) به عنوان ورودی به یک مدل یادگیری عمیق داده شده و پاسخ بهره فیلتر به‌عنوان خروجی پیش‌بینی شود.

در گام نخست، مروری دقیق و عمیق بر فیلتر چبی‌شف نوع اول صورت گرفت. رفتار فرکانسی این فیلتر در ناحیه باند گذر و باند توقف و تأثیر پارامترهای طراحی بر پاسخ آن بررسی گردید. در ادامه، مجموعه‌ای از فیلترها با پارامترهای مختلف طراحی شده و پاسخ بهره هر یک با دقت بالا استخراج شد. این داده‌ها پایه‌گذار دیتاست آموزشی مدل شدند.

در مرحله‌ی بعد، یک **مدل شبکه عصبی چندلایه با ساختار تمام‌متصل (Fully Connected Feedforward Network)** طراحی شد که توانست به خوبی ارتباط غیرخطی بین سه پارامتر ورودی و پاسخ فرکانسی فیلتر را یاد بگیرد. مدل با داده‌های استخراج‌شده آموزش داده شد و پس از آموزش، بر روی مجموعه آزمون ارزیابی شد. نتایج عددی نشان داد که مدل حتی برای فیلترهایی با مرتبه بالا، قادر است با **میانگین خطای کمتر از ۱.۵ دسی‌بل** پاسخ بهره را بازسازی کند.

در فازهای پیشرفته‌تر، پروژه به سمت **کاربرد عملی و صنعتی** حرکت کرد. ابتدا مقایسه دقیقی میان طراحی به روش کلاسیک و طراحی با مدل عصبی صورت گرفت که نشان داد مدل عصبی با دقت بالا و سرعت بسیار زیاد، می‌تواند جایگزین مناسبی برای طراحی عددی پرهزینه فیلترها در شرایط خاص باشد. در گام نهایی، یک **رابط گرافیکی کاربرپسند (GUI)** طراحی و پیاده‌سازی شد که امکان استفاده مستقیم از مدل شبکه عصبی را بدون نیاز به دانش برنامه‌نویسی فراهم ساخت.

نکته حائز اهمیت این است که برخلاف روش کلاسیک که برای هر طراحی نیاز به محاسبه قطب و صفر، حل معادلات، تبدیل فرکانس و محاسبه پاسخ فرکانسی دارد، مدل شبکه عصبی فقط با یک forward pass در کمتر از ۰.۱ ثانیه پاسخ بهره را تولید می‌کند. این مزیت در سیستم‌های بلادرنگ، طراحی خودکار، پردازش سیگنال روی میکروکنترلرها و سیستم‌های محدود از نظر منابع محاسباتی بسیار مؤثر است.

از سوی دیگر، مدل یادگیری‌محور توانست نه‌تنها ساختار کلی پاسخ بهره، بلکه ریپل، شیب ناحیه گذار، و افت در باند توقف را نیز به‌درستی شبیه‌سازی کند. تحلیل‌های عددی (MAEوMSE) نشان داد که خطاهای تولیدی مدل در دامنه‌ای هستند که برای کاربردهای مهندسی قابل پذیرش و قابل اتکا می‌باشند.

جمع‌بندی نهایی

این پروژه نشان داد که:

۱. ترکیب روش‌های سنتی طراحی فیلتر با یادگیری عمیق می‌تواند منجر به مدل‌هایی سریع، دقیق و قابل‌توسعه شود.

۲. مدل‌های شبکه عصبی توانایی بالایی در درک روابط پیچیده و غیرخطی بین پارامترها و پاسخ سیستم دارند.

۳. با استفاده از رابط کاربری گرافیکی، امکان استفاده از مدل حتی برای کاربران غیرفنی نیز فراهم می‌شود.

۴. طراحی فیلتر با مدل عصبی، ابزاری جدید در کنار روش‌های کلاسیک است، نه جایگزین کامل؛ اما در بسیاری از موارد سرعت و سادگی آن مزیت بزرگی محسوب می‌شود.

این پروژه گامی مؤثر در جهت کاربردی‌سازی هوش مصنوعی در طراحی سیستم‌های مهندسی به‌ویژه در حوزه فیلتر و پردازش سیگنال است. اجرای دقیق، مرحله‌به‌مرحله و تحلیلی آن می‌تواند مبنای خوبی برای پایان‌نامه، تحقیق پیشرفته، پروژه صنعتی یا حتی توسعه ابزارهای نرم‌افزاری مهندسی باشد.