به نام خدا

1928
K. N. Toosi University of Technology

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

گزارش مینی پروژه ۲

سیده زهرا عربی

۴۰۰۰۷۱۷۳

استاد : آقای دکتر مهدی علیاری

https://github.com/Zahra-Arabi/MJAHMADEE.git

https://colab.research.google.com/drive/1sGE_Ow4Q4WM3v8UkPLl5lMEmMY3_Cz_K?usp=sharing

دی ۱۴۰۳

فهرست مطالب

عنوان شماره صفحه

پرسش ۱

۱.۱

استفاده از ReLU در لایه ماقبل آخر و سیگموید در لایه آخر در یک مسأله طبقه‌بندی دوکلاسه می‌تواند مشکل‌ساز باشد. دلیل این است که:

- ReLU هر مقدار منفی را صفر می‌کند
- خروجی ReLU می‌تواند هر عدد مثبتی باشد (کران بالا ندارد)
- سیگموید ورودی را به بازه (۰،۱) نگاشت می‌کند
- برای طبقه‌بندی دوکلاسه، معمولاً می‌خواهیم خروجی نهایی یک احتمال بین ۰ و ۱ باشد

مشکل اصلی این است که ReLU می‌تواند مقادیر خیلی بزرگ تولید کند که وقتی وارد سیگموید می‌شوند، باعث اشباع تابع سیگموید می‌شوند. این یعنی گرادیان‌های خیلی کوچک و یادگیری کند.

چرا ترکیب ReLU و سیگموید معمول نیست؟

- تکرار غیرخطی بودن **:** استفاده از دو فعال‌ساز غیرخطی پشت سر هم، لزوماً به بهبود عملکرد شبکه منجر نمی‌شود و ممکن است به مشکلاتی مانند ناپدید شدن گرادیان دامن بزنند.
- توزیع خروجی ReLU **:**خروجی نامحدودی دارد، در حالی که سیگموید خروجی را به بازه [۰, ۱] محدود می‌کند. این تفاوت در توزیع خروجی می‌تواند باعث مشکلات در آموزش شود.

چه اتفاقی می‌افتد؟

۱. **عملکرد ضعیف :**ترکیب ReLU و سیگموید ممکن است به عملکرد ضعیف شبکه منجر شود. دلایل این امر می‌تواند ناپایداری آموزش، مشکل در همگرایی و عدم توانایی شبکه در یادگیری ویژگی‌های پیچیده باشد.

۲. **عملکرد قابل قبول :**در برخی موارد، این ترکیب ممکن است به عملکرد قابل قبولی منجر شود. این امر به عوامل مختلفی مانند اندازه شبکه، داده‌های آموزشی و روش‌های تنظیم ابرپارامترها بستگی دارد.

۳. **مشکلات در آموزش :**ممکن است شبکه به دلیل ناپایداری گرادیان یا مشکلات همگرایی، به خوبی آموزش نبیند.

۱.۲

این تابع یک نوع Exponential Linear Unit (ELU) است. برای محاسبه گرادیان:

برای $x >= 0$

- $(\mathrm{ELU(x)})' = 1$

برای $x < 0$

- $(\mathrm{ELU(x)})' = \alpha e^{x}$

مزایای این تابع نسبت به ReLU

- برخلاف ReLU که برای مقادیر منفی کاملاً صفر است، این تابع برای مقادیر منفی یک خروجی منفی تولید می‌کند که می‌تواند میانگین فعال‌سازی‌ها را به صفر نزدیک‌تر کند. این باعث می‌شود که مشکل "dying ReLU" که در آن نورون‌ها کاملاً غیرفعال می‌شوند کمتر رخ دهد.
- گرادیان برای مقادیر منفی صفر نیست که این باعث یادگیری بهتر می‌شود.
- انعطاف‌پذیری بیشتر :پارامتر α در این تابع، به ما اجازه می‌دهد تا شیب تابع را برای مقادیر منفی تنظیم کنیم. این انعطاف‌پذیری می‌تواند به بهبود عملکرد شبکه در برخی مسائل کمک کند.

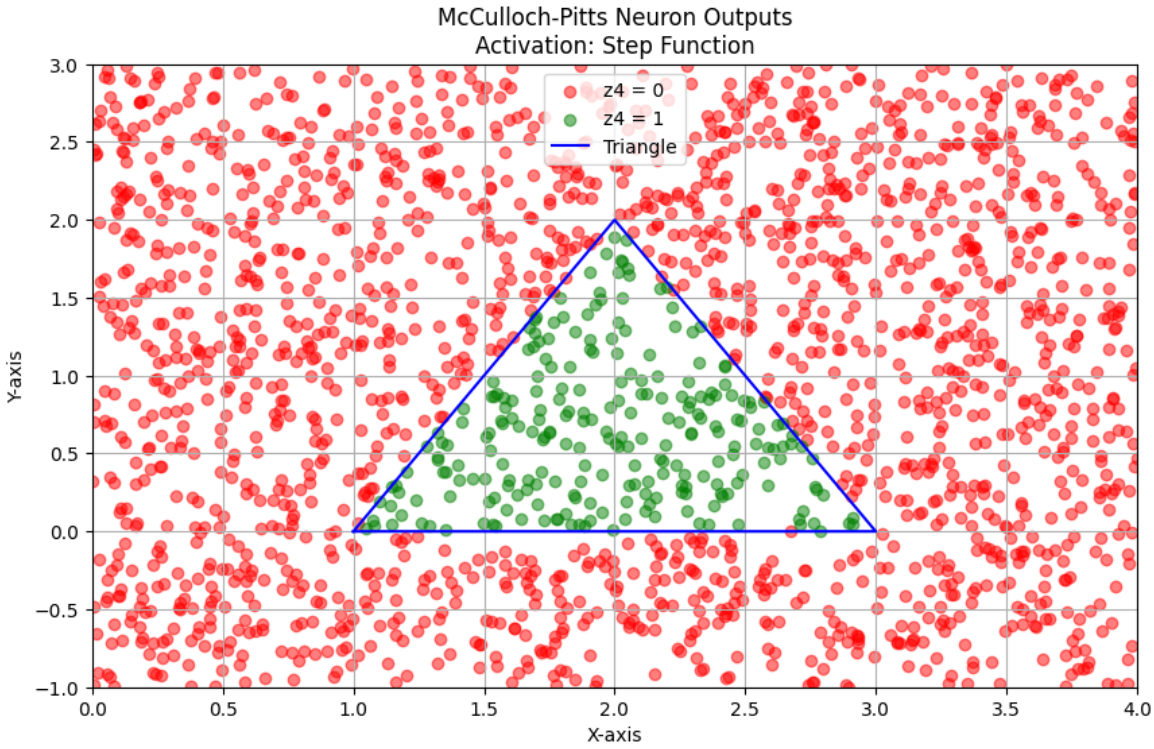
۱.۳

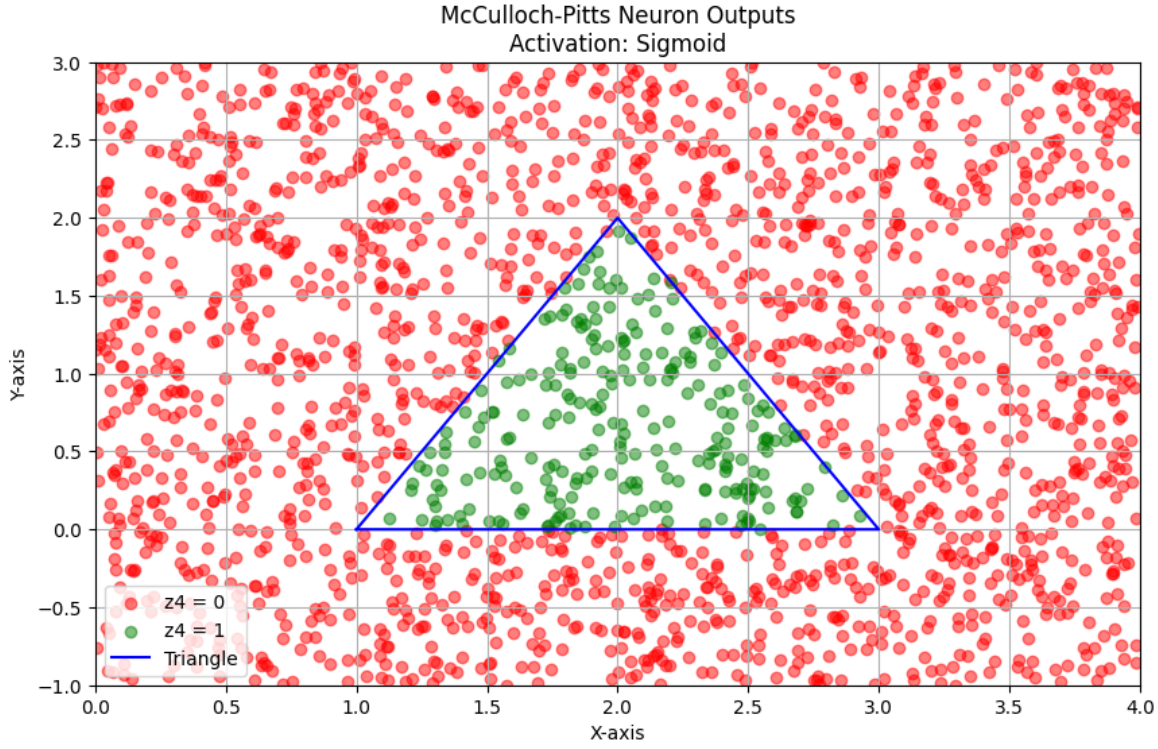
این کد یک شبیه‌سازی بصری از یک نورون مک‌کالاک-پیتس را ارائه می‌دهد که برای تشخیص اینکه آیا نقاط تصادفی درون یک مثلث معین قرار دارند یا نه، استفاده می‌شود.

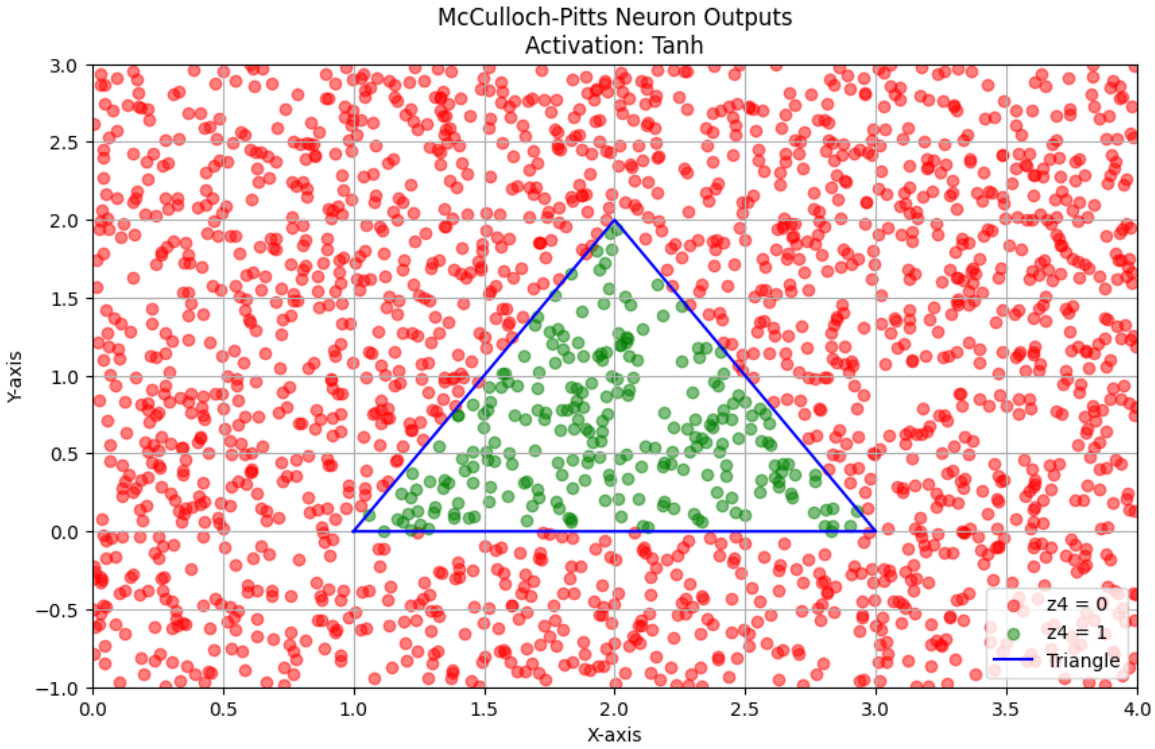
تعریف کلاس :McCullochPittsNeuron این کلاس یک نورون مک‌کالاک-پیتس را تعریف می‌کند که شامل وزن‌ها، بایاس و تابع فعال‌سازی است. سه تابع فعال‌سازی مختلف (پله‌ای، سیگموئید و تانژانت هایپربولیک) تعریف شده‌اند که می‌توانند برای تعیین خروجی نورون بر اساس ورودی‌ها استفاده شوند.

۱. **تولید نقاط تصادفی** :تابع generate_points نقاط تصادفی را در یک محدوده مشخص ایجاد می‌کند.
۲. **تشخیص موقعیت نقطه نسبت به مثلث** :تابع is_inside_triangle بررسی می‌کند که آیا یک نقطه خاص در داخل مثلث مورد نظر قرار دارد یا نه، این تابع با محاسبه و مقایسه مساحت‌های مثلث‌ها کار می‌کند.
۳. **تصویرسازی و طبقه‌بندی نقاط** :تابع plot_triangle_classifier نورون را با وزن‌ها و بایاس مشخص ایجاد می‌کند، سپس نقاط تصادفی را طبقه‌بندی می‌کند و نتایج را بر اساس اینکه آیا نقاط درون مثلث قرار دارند یا خارج از آن هستند، با رنگ‌های متفاوت روی نمودار نمایش می‌دهد. همچنین خود مثلث را نیز روی نمودار ترسیم می‌کند.
٤. **اجرای توابع فعال‌سازی مختلف** :در انتها، این کد تابع plot_triangle_classifier را با هر یک از توابع فعال‌سازی برای نمایش تفاوت‌های نتایج حاصل از هر تابع فراخوانی می‌کند.

این کد ابزاری برای درک نحوه کار نورون‌های مصنوعی و تأثیر توابع فعال‌سازی مختلف بر رفتار آن‌ها در تشخیص الگوها است.


McCulloch-Pitts Neuron Outputs
Activation: Step Function
z4 = 0
z4 = 1
Triangle
Y-axis
X-axis
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


McCulloch-Pitts Neuron Outputs
Activation: Sigmoid
z4 = 0
z4 = 1
Triangle
Y-axis
X-axis
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0



نقاط سبز نشان‌دهنده‌ی نقاطی هستند که طبق تشخیص الگوریتم درون مثلث قرار دارند و نقاط قرمز نشان‌دهنده‌ی نقاطی هستند که بیرون مثلث قرار گرفته‌اند. تابع فعال‌سازی مورد استفاده در این نمودار تابع پله‌ای است که برای تعیین وضعیت نقاط (داخل یا خارج از مثلث) استفاده شده‌است.

تحلیل خروجی:

- تابع فعال‌سازی پله‌ای به طور بسیار مشخصی بین دو کلاس (داخل و خارج مثلث) تفکیک ایجاد می‌کند. در این مورد، خروجی نشان می‌دهد که بیشتر نقاط درون مثلث به درستی به عنوان داخل مثلث تشخیص داده شده‌اند (نقاط سبز) و نقاط خارج از مثلث نیز عمدتاً به درستی تشخیص داده شده‌اند (نقاط قرمز).

بررسی اثر اضافه کردن دو تابع فعال‌ساز مختلف به فرآیند تصمیم‌گیری:

- **تابع فعال‌ساز سیگموئید** :این تابع به جای ارائه خروجی صفر یا یک (مانند تابع پله‌ای)، احتمالی بین صفر تا یک را برمی‌گرداند. برای استفاده در یک نورون مک‌کالاک-پیتس، معمولاً یک آستانه‌ای مانند ۰.۵ در نظر گرفته می‌شود تا تعیین کند که آیا خروجی باید ۰ یا ۱ باشد. این تابع می‌تواند در مواردی که اطلاعات نزدیک به مرز تصمیم هستند، نرم‌تر و انعطاف‌پذیرتر عمل کند.

- **تابع فعال‌ساز تانژانت هایپربولیک :(tanh)** این تابع خروجی‌هایی در بازه‌ی (-۱, ۱) تولید می‌کند و می‌تواند به عنوان تابع فعال‌ساز در شرایطی که تشخیص دقیق‌تری مورد نیاز است، استفاده شود. تابع tanh به خاطر خروجی‌های بیشتر دوطرفه‌اش می‌تواند در تشخیص دقیق‌تر مرزهای بین کلاس‌ها مفید باشد.

نتیجه‌گیری :استفاده از توابع فعال‌ساز مختلف می‌تواند تأثیر زیادی بر روی دقت و نرمی تصمیم‌گیری نورون داشته باشد. تابع پله‌ای بسیار مشخص و قاطع است و برای مسائلی که نیاز به تفکیک واضح دارند مناسب است. در حالی که توابع سیگموئید و tanh می‌توانند انعطاف‌پذیری بیشتری در تشخیص‌های نزدیک به مرز فراهم کنند و احتمال خطا را در این نواحی کاهش دهند.

پرسش ۲

۲.۱

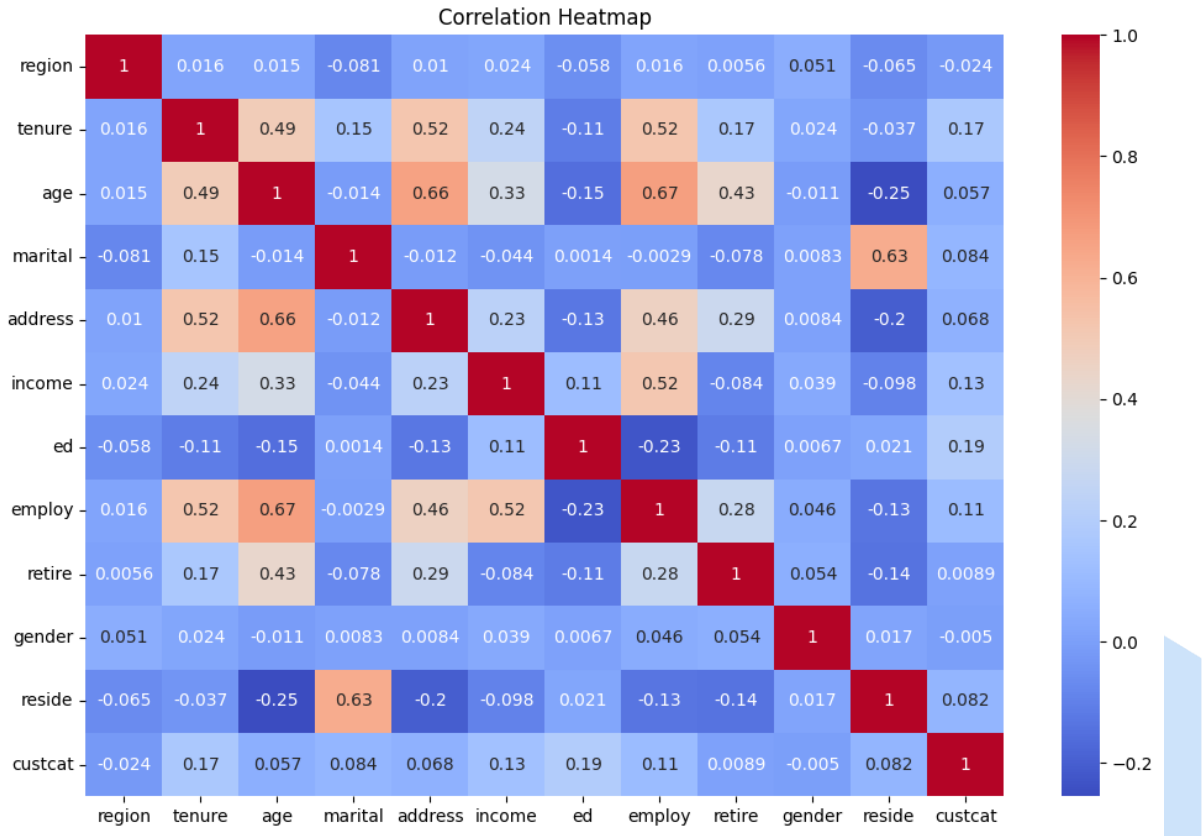
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   region   1000 non-null   int64  
 1   tenure   1000 non-null   int64  
 2   age      1000 non-null   int64  
 3   marital  1000 non-null   int64  
 4   address  1000 non-null   int64  
 5   income   1000 non-null   float64
 6   ed       1000 non-null   int64  
 7   employ   1000 non-null   int64  
 8   retire   1000 non-null   float64
 9   gender   1000 non-null   int64  
 10  reside   1000 non-null   int64  
 11  custcat  1000 non-null   int64  
dtypes: float64(2), int64(10)
memory usage: 93.9 KB
(   region  tenure  age  marital  address  income  ed  employ  retire  gender  \
 0       2      13   44        1        9    64.0   4       5     0.0       0   
 1       3      11   33        1        7   136.0   5       5     0.0       0   
 2       3      68   52        1       24   116.0   1      29     0.0       1   
 3       2      33   33        0       12    33.0   2       0     0.0       1   
 4       2      23   30        1        9    30.0   1       2     0.0       0   

    reside  custcat  
 0       2        0  
 1       6        3  
 2       2        2  
 3       1        0  
 4       4        2  ,
 None,
 (1000, 12))
```

این دیتا شامل دوازده ستون (۱۱ ورودی و یک خروجی است) و ۱۰۰۰ نمونه دارد. و هیچ دیتا null وجود ندارد

ورودی ها: منطقه_ مدت زمان سکونت یا استفاده از خدمات_ سن_وضعیت تاهل_ مدت زمان سکونت در آدرس فعلی_درآمد _ تحصیلات_ مدت زمان اشتغال_ وضعیت بازنشستگی_جنسیت_ تعداد افراد ساکن در خانه

خروجی: دسته بندی مشتری(دارای چهار برچسب پایه، الکترونیکی، پیشرفته و کامل)

۲.۲



مقادیر همبستگی نزدیک به ۱ یا -۱ نشان‌دهنده ارتباط قوی بین متغیرها هستند، در حالی که مقادیر نزدیک به صفر نشان دهنده نبود ارتباط قابل توجه هستند.

تحلیل همبستگی‌های بالا:

سن (Age) و مدت زمان سکونت – (Address) با همبستگی ۰.۶۶

- این همبستگی نشان می‌دهد که افرادی که سن بیشتری دارند، مدت زمان طولانی‌تری را در آدرس فعلی خود سپری کرده‌اند. این موضوع ممکن است به دلیل ثبات بیشتر در زندگی و کمتر تغییر محل سکونت با افزایش سن باشد.

سن (Age) و مدت زمان اشتغال – (Employ) با همبستگی ۰.۶۷

- این رابطه قوی نشان دهنده آن است که با افزایش سن، مدت زمان اشتغال نیز افزایش می‌یابد. این می‌تواند بیانگر ثبات شغلی و افزایش تجربه و مهارت‌های شغلی با گذشت زمان باشد.

مدت زمان سکونت (Address) و مدت زمان اشتغال – (Employ) با همبستگی ۰.۵۲

- این نشان می‌دهد که افرادی که مدت زمان بیشتری در یک آدرس زندگی کرده‌اند، اغلب دوره‌های طولانی‌تری نیز در شغل خود دارند. این می‌تواند به دلیل ثبات عمومی در زندگی آن‌ها باشد.

وضعیت تأهل (Marital) و تعداد ساکنان – (Reside) با همبستگی ۰.۶۳

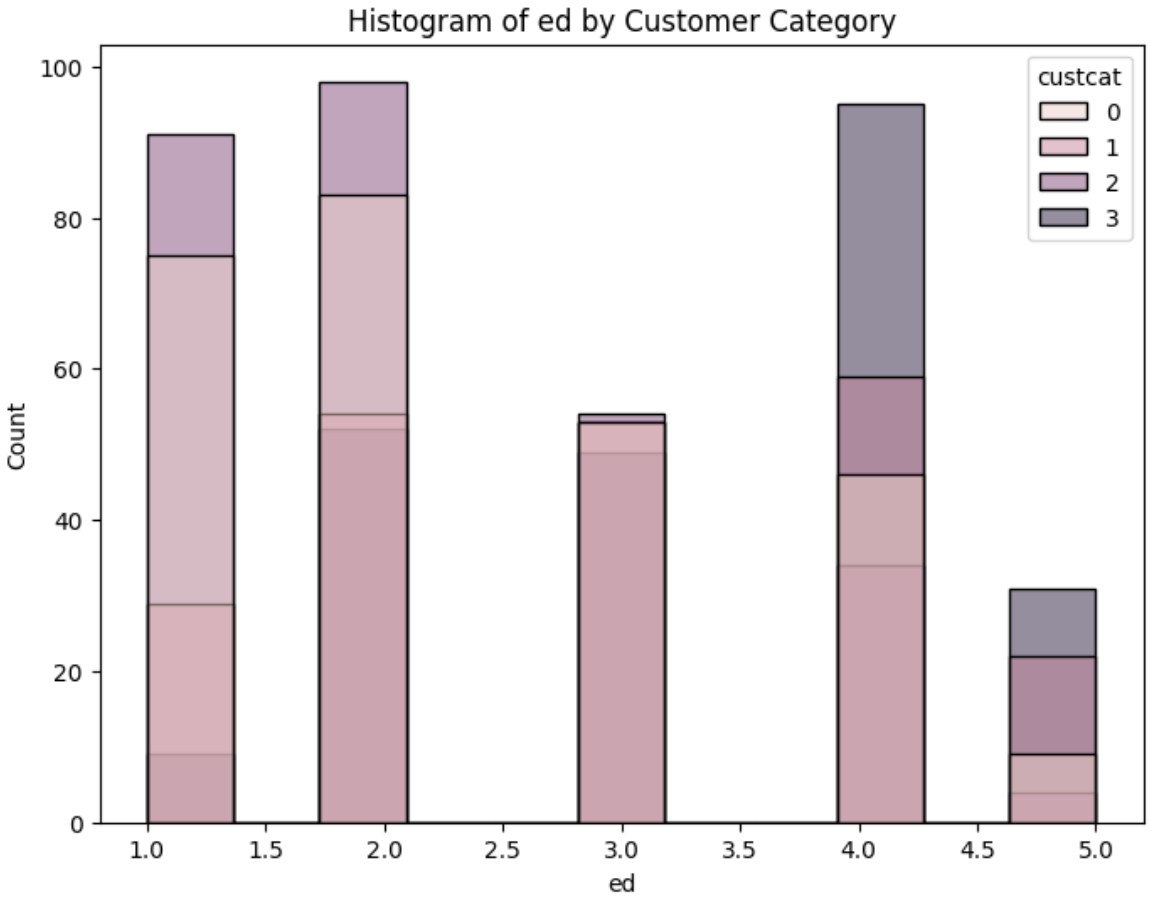
- این همبستگی بالا بین وضعیت تأهل و تعداد افراد ساکن در خانه می‌تواند بیانگر این باشد که افراد متأهل معمولاً با خانواده‌های بزرگ‌تری زندگی می‌کنند.

تحلیل دیگر همبستگی‌ها:

- **وضعیت تأهل (marital) و وضعیت بازنشستگی (retire)** : همبستگی منفی نسبتاً کم (-۰.۰۷۸) نشان دهنده ارتباط ضعیف بین این دو متغیر است.
- **تحصیلات (ed) و درآمد :(income)** همبستگی ۰.۱۱ نشان می‌دهد که بین تحصیلات و درآمد ارتباط ضعیفی وجود دارد، اما انتظار می‌رود افراد با تحصیلات بالاتر درآمد بیشتری داشته باشند.

همبستگی‌های کم اهمیت:

- بسیاری از مقادیر همبستگی نزدیک به صفر هستند، مانند ارتباط بین جنسیت (gender) و سایر متغیرها. این نشان می‌دهد که جنسیت تأثیر ناچیزی بر متغیرهای دیگر دارد.



توزیع تحصیلات (ed) را بر حسب دسته‌بندی مشتریان (custcat) نشان می‌دهد. این هیستوگرام به صورت زیر تحلیل می‌شود:

توزیع تحصیلات:

- **ed** : مقادیر نمایان در نمودار از ۱ تا ۵ را شامل می‌شود، که ممکن است به معنای سطوح مختلف تحصیلاتی باشد (مثلاً ۱: دبستان، ۲: راهنمایی، ۳: دبیرستان، ۴: کارشناسی، ۵: کارشناسی ارشد و بالاتر).

توزیع دسته‌بندی مشتریان:

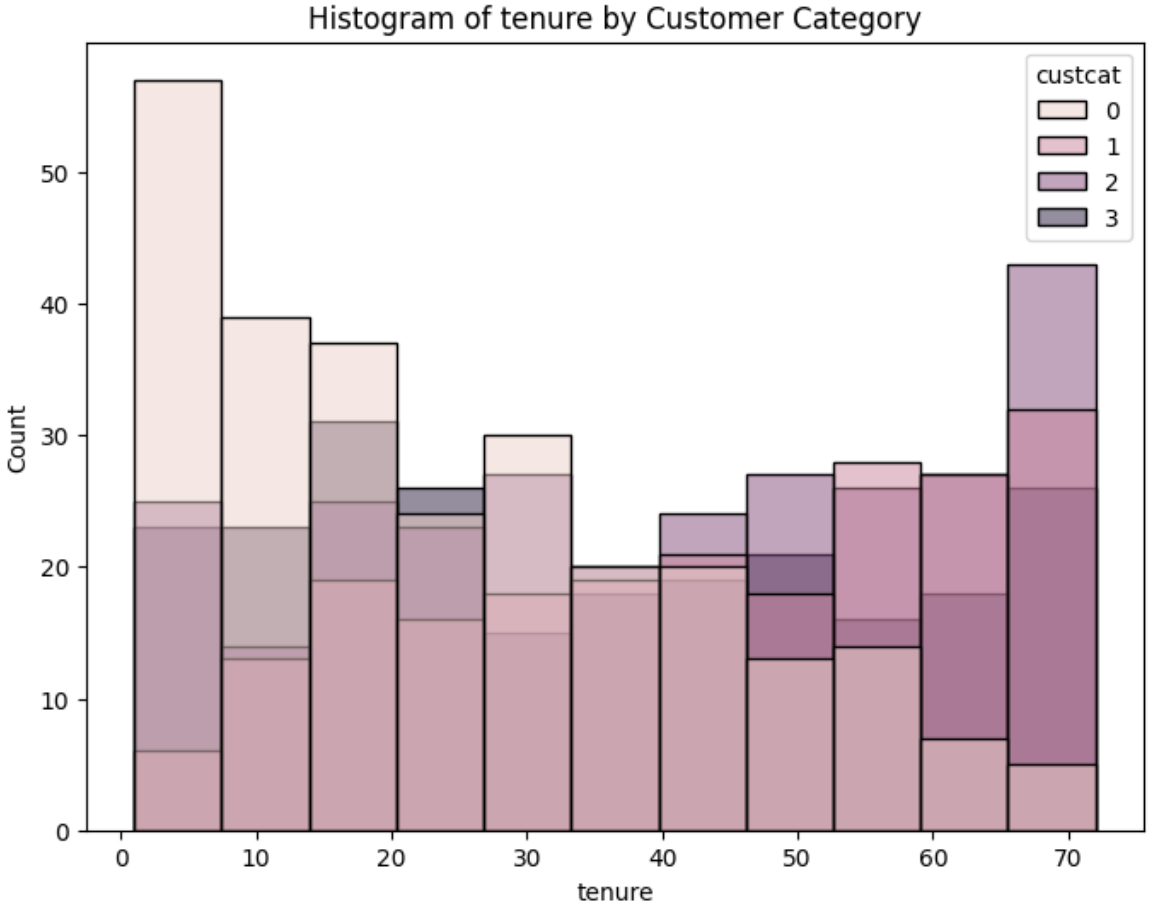
- **custcat**: شامل چهار دسته مختلف است که ممکن است نشان‌دهنده نوع خدمات یا سطح مشتریان باشد. مقادیر ۰ تا ۳ ممکن است. (دارای چهار برچسب پایه، الکترونیکی، پیشرفته و کامل)

تحلیل نمودار:

- **تحصیلات پایه (۱.۰ و ۲.۰)** :این دو سطح بیشترین تعداد افراد را در بین دسته‌های مشتریان دارند. عمده افراد در این سطوح تحصیلاتی متعلق به دسته‌بندی مشتریان ۰ و ۱ هستند.
- **تحصیلات متوسط (3.0)** : تعداد افراد با این سطح تحصیلات کمتر است و عمدتا در دسته مشتری ۰ و ۱ قرار دارند.
- **تحصیلات عالیه (۴.۰ و ۵.۰)** :این سطوح تحصیلاتی تعداد نسبتا کمتری از مشتریان را شامل می‌شوند و اکثر این افراد در دسته ۱ و ۲ قرار دارند. مشتریان با بالاترین سطح تحصیلی (۵.۰) تقریبا فقط در دسته ۱ دیده می‌شوند.

نتیجه‌گیری:

این توزیع نشان می‌دهد که افراد با سطوح تحصیلی پایین‌تر بیشترین جمعیت را تشکیل می‌دهند و عمده آنها در دسته‌های پایین‌تر مشتری قرار دارند. این امر ممکن است نشان‌دهنده نوع خدمات یا محصولاتی باشد که به این دسته‌ها ارائه می‌شود و احتمالا با قیمت پایین‌تر و دسترسی آسان‌تر همراه است. در حالی که افراد با تحصیلات بالاتر تعداد کمتری دارند و بیشتر در دسته‌های بالاتر مشتریان دیده می‌شوند، که می‌تواند نشان‌دهنده خدمات یا محصولات با کیفیت بالاتر و گران‌تر باشد.



توزیع مدت زمان استفاده (tenure) مشتریان را بر اساس دسته‌بندی مشتریان (custcat) نشان می‌دهد. تحلیل این نمودار به صورت زیر است:

توزیع مدت زمان استفاده:

- **Tenure**: مقادیر از ۰ تا ۷۰ نشان داده شده، که می‌تواند بر حسب ماه یا سال باشد، و نشان‌دهنده مدت زمان استفاده مشتریان از خدمات یا محصولات شرکت است.

توزیع دسته‌بندی مشتریان:

- **custcat**: شامل چهار دسته مختلف است که ممکن است نشان‌دهنده نوع خدمات یا سطح مشتریان باشد. مقادیر ۰ تا ۳ ممکن است. (دارای چهار برچسب پایه، الکترونیکی، پیشرفته و کامل)

تحلیل نمودار:

- **مشتریان جدید (tenure) نزدیک به ۰** :اکثریت این گروه در دسته ۰ قرار دارند، که نشان دهنده جذب بالای مشتریان جدید در این دسته است.
- **توزیع مدت زمان استفاده متوسط (بین ۱۰ تا ۵۰)** :این توزیع نشان می‌دهد که مشتریان با دوره‌های متوسط استفاده در تمام دسته‌ها به نسبت برابر پخش شده‌اند، با تمرکز اندکی بیشتر در دسته‌های ۱ و ۲.
- **مشتریان با دوره استفاده طولانی (بیش از ۶۰)** :تعداد قابل توجهی از این مشتریان در دسته ۳ قرار دارند، که ممکن است نشان دهنده وفاداری بالا در این دسته باشد.

نتیجه‌گیری:

این توزیع ممکن است نشان‌دهنده چگونگی جذب و نگهداری مشتریان در شرکت باشد. دسته‌بندی ۰ ممکن است مربوط به خدمات یا محصولات ابتدایی با جذب مشتری بالا باشد، در حالی که دسته‌بندی ۳ می‌تواند شامل خدمات یا محصولات پرمیوم با مشتریان وفادارتر باشد. این اطلاعات می‌توانند برای بهبود استراتژی‌های بازاریابی و فروش به شرکت کمک کنند.

۲.۳

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside
0	0.5	0.169014	0.440678	1.0	0.163636	0.033153	0.75	0.106383	0.0	0.0	0.142857
1	1.0	0.140845	0.254237	1.0	0.127273	0.076552	1.00	0.106383	0.0	0.0	0.714286
2	1.0	0.943662	0.576271	1.0	0.436364	0.064497	0.00	0.617021	0.0	1.0	0.142857
3	0.5	0.450704	0.254237	0.0	0.218182	0.014467	0.25	0.000000	0.0	1.0	0.000000
4	0.5	0.309859	0.203390	1.0	0.163636	0.012658	0.00	0.042553	0.0	0.0	0.428571

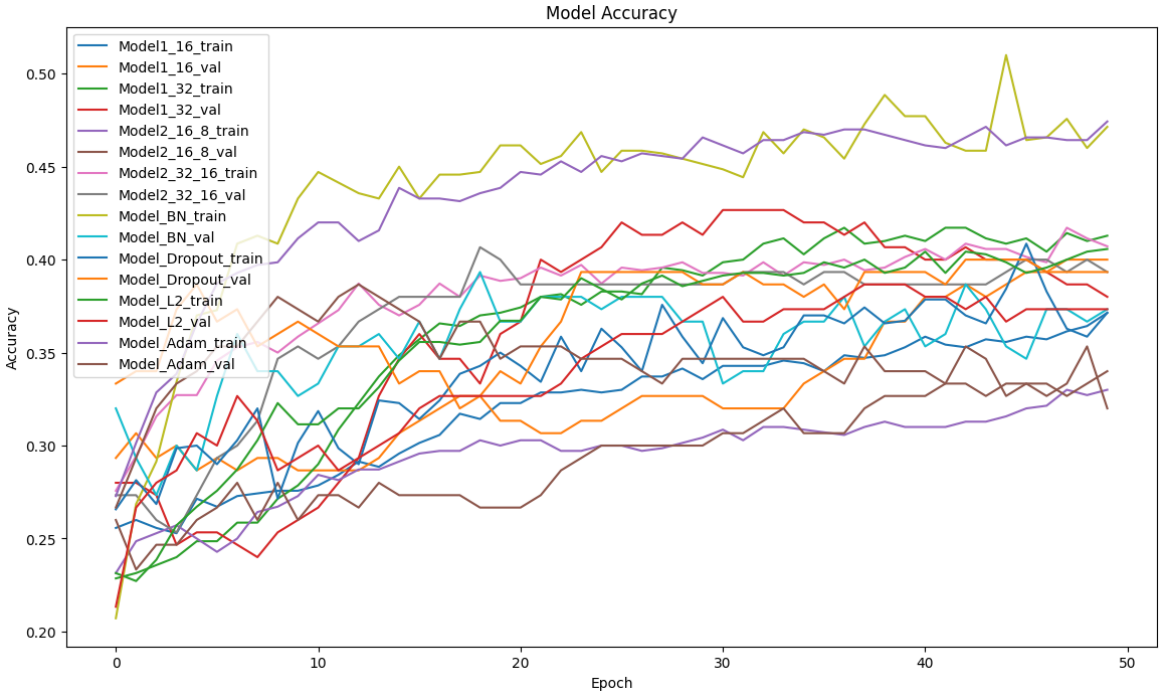
تمام داده های مربوط به ورودی نرمال سازی شده اند، ۳۰ درصد داده ها متعلق به مجموعه تست هستند که ۵۰ درصد مجموعه تست متعلق به داده های اعتبارسنجی هستند.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

هر ویژگی (feature) را طوری مقیاس‌بندی می‌کند که کوچکترین مقدار آن ویژگی به ۰ و بزرگترین مقدار به ۱ تبدیل شود.

این نوع از نرمال‌سازی به خصوص زمانی مفید است که می‌خواهیم تاثیرات نامتعادل بزرگی مقادیر برخی ویژگی‌ها را کاهش دهیم، و به این ترتیب اطمینان حاصل کنیم که الگوریتم‌های یادگیری ماشین که حساس به مقیاس ویژگی‌ها هستند، مثل مدل‌های که از گرادیان استفاده می‌کنند، بدون تحیز عمل کنند.

۲.۴



۱. تأثیر تعداد نورون‌ها بر عملکرد مدل:

مدل با تعداد نورون‌های بیشتر مانند Model1_32 وModel2_32_16 به طور کلی دقت بالاتری در داده‌های آموزش و اعتبار نشان داده است نسبت به مدل‌های با نورون‌های کمتر مانند Model1_16 و .Model2_16_8 این نشان می‌دهد که افزایش تعداد نورون‌ها ممکن است به بهبود توانایی مدل در یادگیری ویژگی‌های پیچیده‌تر کمک کند.

پ.ن: من بهترین مدل را مدل Model1_32 در نظر گرفتم و سایر موارد خواسته شده را روی این مدل پیاده سازی کردم.(زیرا احساس کردم با دولایه مخفی مدل بیش از اندازه یاد خواهد گرفت)

۲. **تأثیر Batch Normalization:**

مدل با Batch Normalization (Model_BN) نشان دهنده بهبود قابل توجه در ثبات و دقت آموزش و اعتبار است. نمودار نشان می‌دهد که دقت این مدل به طور یکنواختی بیشتر است و کمتر دچار نوسان است، که نشان دهنده تأثیر مثبت Batch Normalization در کاهش مشکل overfitting است.

۳. **تأثیر Dropout:**

استفاده از Dropout (Model_Dropout) هم به نظر می‌رسد تأثیر مثبتی در کاهش overfitting داشته است. مدل‌هایی که از Dropout استفاده می‌کنند، دارای دقت آموزش و اعتبار نزدیک‌تری به یکدیگر هستند، که نشان می‌دهد Dropout به مدل کمک کرده است تا عمومی‌تر یاد بگیرد.

٤. **تأثیر L2 Regularization:**

مدل با L2 Regularization (Model_L2) نیز دارای عملکرد بهتری در مقایسه با مدل‌های بدون این تنظیم است، خصوصاً در دقت اعتبار. این نشان می‌دهد که L2 Regularization به کاهش overfitting کمک کرده و مدل را قادر ساخته است تا ویژگی‌های عمومی‌تری را یاد بگیرد.

٥. **تأثیر Optimizers مختلف:**

آزمایش با استفاده از Adam Optimizer (Model_Adam) نشان دهنده بهبود قابل توجه در دقت اعتبار است، که می‌تواند نشان دهنده بهینه‌سازی بهتر در فرآیند یادگیری باشد. این نتیجه نشان می‌دهد که Adam ممکن است در مواردی که SGD به خوبی عمل نکرده است، گزینه بهتری باشد.

بر اساس تحلیل‌های انجام شده، مدل‌هایی که از تکنیک‌هایی مانند Batch Normalization, Dropout و L2 Regularization استفاده می‌کنند و همچنین تکنیک‌های بهینه‌سازی مدرن‌تر مانند Adam ، عموماً عملکرد بهتری دارند. این یافته‌ها می‌توانند در تصمیم‌گیری برای انتخاب تنظیمات مناسب برای شبکه‌های عصبی در پروژه‌های آینده مفید باشند.

۲.۵

```
نتایج تست:
Model1_16 - 0.3200 :دقت تست
Model1_32 - 0.3800 :دقت تست
Model2_16_8 - 0.3600 :دقت تست
Model2_32_16 - 0.4133 :دقت تست
Model_BN - 0.4267 :دقت تست
Model_Dropout - 0.3867 :دقت تست
Model_L2 - 0.3667 :دقت تست
Model_Adam - 0.4000 :دقت تست
```

```
Model1_16:
واقعی: 2, پیش‌بینی: 2
واقعی: 3, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 1
واقعی: 3, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
```

```
Model1_32:
واقعی: 2, پیش‌بینی: 2
واقعی: 3, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 1
واقعی: 3, پیش‌بینی: 3
واقعی: 0, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
```

```
Model2_16_8:
واقعی: 2, پیش‌بینی: 2
واقعی: 0, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 3
واقعی: 0, پیش‌بینی: 1
واقعی: 0, پیش‌بینی: 3
واقعی: 0, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
```

```
Model2_16_8:
واقعی: 2, پیش‌بینی: 2
واقعی: 0, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 3
واقعی: 0, پیش‌بینی: 1
واقعی: 0, پیش‌بینی: 3
واقعی: 0, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
```

```
Model2_32_16:
واقعی: 2, پیش‌بینی: 2
واقعی: 3, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 1
واقعی: 3, پیش‌بینی: 3
واقعی: 1, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
```

```
Model_BN:
واقعی: 2, پیش‌بینی: 2
واقعی: 1, پیش‌بینی: 3
واقعی: 1, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 1
واقعی: 3, پیش‌بینی: 3
واقعی: 0, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 2
واقعی: 3, پیش‌بینی: 3
```

```
Model_Dropout:
واقعی: 2, پیش‌بینی: 2
واقعی: 0, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 1
واقعی: 3, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
```

```
Model_L2:
واقعی: 2, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 1
واقعی: 0, پیش‌بینی: 3
واقعی: 0, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
```

```
Model_Adam:
واقعی: 2, پیش‌بینی: 2
واقعی: 1, پیش‌بینی: 3
واقعی: 1, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 1
واقعی: 3, پیش‌بینی: 3
واقعی: 3, پیش‌بینی: 3
واقعی: 2, پیش‌بینی: 0
واقعی: 0, پیش‌بینی: 0
واقعی: 2, پیش‌بینی: 2
واقعی: 2, پیش‌بینی: 3
```

بهترین عملکرد در داده‌های تست مربوط به مدل Model_BN است با دقت **0.4267**. این نشان می‌دهد که اضافه کردن لایه Batch Normalization به مدل، عملکرد بهتری را در داده‌های تست ارائه داده است.

- **افزایش تعداد نورون‌ها:**
  - افزایش تعداد نورون‌ها در مدل اول از ۱۶ به ۳۲ باعث افزایش دقت تست از ۰.۳۲۰۰ به ۰.۳۸۸۰ شد.
- **افزایش تعداد لایه‌ها:**
  - افزایش تعداد لایه‌ها در مدل دو لایه (Model2_32_16) در مقایسه با مدل تک‌لایه (Model1_32) باعث بهبود عملکرد از ۰.۳۸۸۰ به ۰.۴۱۳۳ شد.
- **Batch Normalization**
  - استفاده از Batch Normalization در مدل باعث بهبود عملکرد نسبت به مدل بدون این ویژگی شد و با دقت ۰.۴۲۶۷ بهترین عملکرد را داشته است.
- **Dropout و L2-Regularization**
  - مدل با Dropout و L2 Regularization نسبت به مدل پایه عملکرد بهتری دارند اما عملکرد آن‌ها از مدل‌های با Batch Normalization کمتر است.
- **بهینه‌ساز Adam**
  - استفاده از Adam نسبت به SGD در مدل‌ها با دقت ۰.۴۰۰۰ عملکرد بهتری ارائه کرده است.

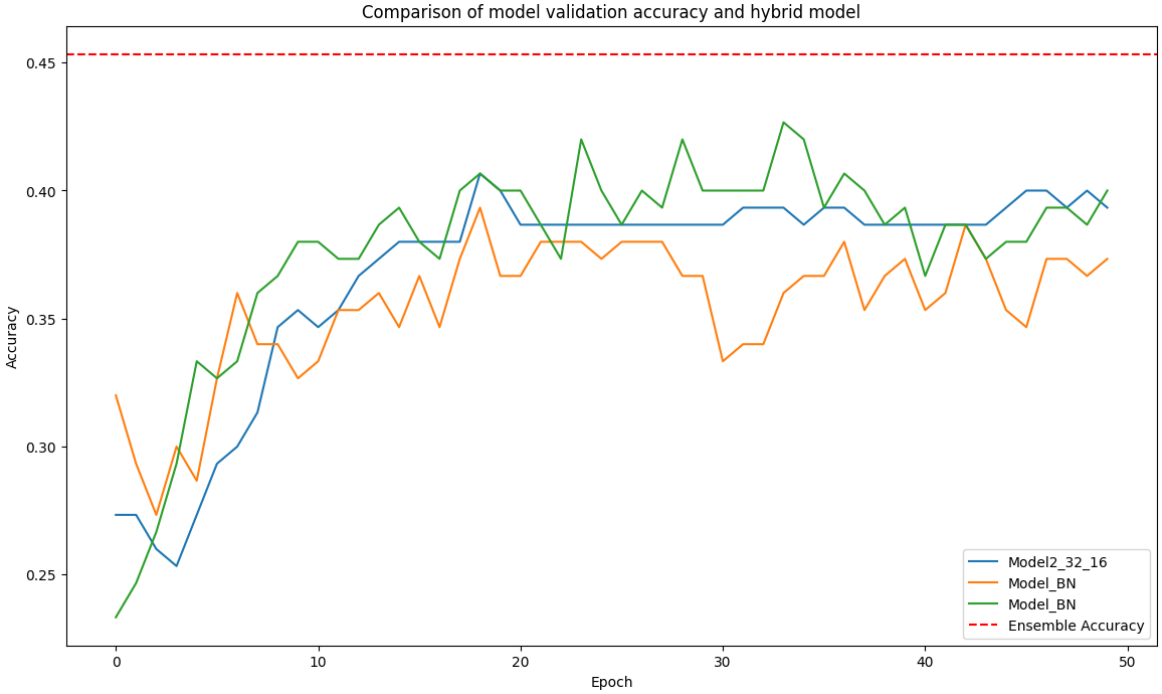
خروجی‌های مدل‌ها برای نمونه‌های تصادفی نشان می‌دهد که اکثر مدل‌ها در تشخیص دسته‌بندی واقعی دچار مشکل هستند و پیش‌بینی‌های اشتباه زیادی دارند.
Model_BN در پیش‌بینی دسته‌بندی‌های درست عملکرد بهتری نسبت به سایر مدل‌ها داشته است.

افزودن Batch Normalization باعث پایداری بیشتر یادگیری و افزایش دقت تست شده است.

۲.۶

در این قسمت دو مدل model_bn و model_1_32 با هم ترکیب کردم که بهترین عملکرد را داشتند.

دقت مدل ترکیبی 0.45 است.



- **مقایسه دقت مدل‌ها:**
  - Model2_32_16 و Model_BN دقت‌های انفرادی ۰.۴۱۳۳ و ۰.۴۲۶۷ داشتند. مدل ترکیبی دقت بالاتری نسبت به هر دو مدل منفرد ارائه داده است.
  - خط قرمز نشان‌دهنده **دقت مدل ترکیبی** است که بالاتر از دقت هر یک از مدل‌های تکی قرار گرفته است. این نشان می‌دهد که ترکیب مدل‌ها به بهبود عملکرد کلی کمک کرده است.
- **مزیت ترکیب مدل‌ها:**
  - ترکیب مدل‌ها معمولاً با کاهش نویز و بهره‌برداری از نقاط قوت هر مدل باعث بهبود دقت می‌شود. در این مورد، Model_BN به دلیل استفاده از Batch Normalization عملکرد پایدارتر و بهتر داشته و **Model2_32_16** به دلیل تعداد نورون‌ها و لایه‌های بیشتر، تنوع بیشتری ارائه داده است. ترکیب این دو مدل نقاط قوت هر دو را ترکیب کرده است.
- **پایداری مدل ترکیبی:**

- دقت مدل ترکیبی به طور قابل ملاحظه‌ای در طول اپیاک‌ها ثابت‌تر است و نویز کمتری دارد. این نتیجه می‌تواند نشان‌دهنده کاهش overfitting یا underfitting باشد که معمولاً در ترکیب مدل‌ها رخ می‌دهد.

- **بهبود دقت:**

- دقت مدل ترکیبی به دلیل میانگین‌گیری پیش‌بینی‌ها، تمایل به کاهش خطاهای پیش‌بینی دارد. این امر به‌ویژه وقتی مدل‌ها از لحاظ معماری یا بهینه‌سازی متفاوت باشند، مؤثر است.

- **نقاط ضعف:**

- ترکیب مدل‌ها ممکن است محاسبات بیشتری نیاز داشته باشد، زیرا هر مدل جداگانه باید پیش‌بینی کند. با این حال، در این مثال به نظر می‌رسد که افزایش دقت ارزش این هزینه محاسباتی اضافی را دارد.

پرسش ۳

۳.۱

سلول اول: تابع تبدیل ImageToBinary

این تابع مسیر فایل تصویر را می گیرد و آن را بر اساس شدت پیکسل به نمایش باینری تبدیل می کند.

Image Opening: فایل تصویر را با استفاده از کتابخانه PIL باز می کند.

پردازش پیکسل: روی هر پیکسل در تصویر تکرار می شود.

محاسبه شدت: برای هر پیکسل، شدت کل را با جمع مقادیر RGB محاسبه می کند.

آستانه گذاری: پیکسل هایی با شدت بالاتر از آستانه مشخص به سفید (-۱) و پیکسل های زیر به سیاه (۱) تبدیل می شوند.

نمایش باینری: نتیجه یک لیست از -۱ و ۱ است که به ترتیب نشان دهنده پیکسل های سفید و سیاه است.

اصلاح بصری: پیکسل های تصویر اصلی بر اساس آستانه به سیاه یا سفید خالص تغییر می کنند و تصویر دستکاری می شود اما ذخیره یا نمایش داده نمی شود.

سلول دوم: ایجاد عملکرد NoisyImages

این عملکرد نویز را به مجموعه ای از تصاویر اضافه می کند و نسخه های جدید و نویز را ذخیره می کند. در اینجا نحوه کار آن آمده است:

مدیریت فایل تصویری: مسیرهای چند تصویر مشخص شده است.

اضافه کردن نویز: برای هر تصویر

تصویر باز شده و پردازش می شود.

نویز تصادفی به مقادیر RGB هر پیکسل اضافه می شود.

تصویر نویزدار در یک فایل جدید ذخیره می شود.

خروجی: مسیرهای تصاویر نویز پس از تولید و ذخیره آنها چاپ می شود.

توضیح توابع بهبود یافته

عملکردهای بهبود یافته ای که نوشته ام از نظر وضوح، کارایی و انعطاف پذیری نسبت به نسخه قبلی چندین مزیت دارد.

تابع: convert_image_to_binary

این تابع با استفاده از تکنیک های مدرن پایتون یک تصویر را به فرمت باینری (سیاه و سفید) تبدیل می کند:

تبدیل تصویر به مقیاس خاکستری: روش image.convert('L') تصویر رنگی را به مقیاس خاکستری تبدیل می کند ('L' مخفف روشنایی). این ساده‌سازی سودمند است زیرا داده‌های تصویر را به یک کانال درخشندگی کاهش می‌دهد و پردازش بعدی را سریع‌تر و پیچیده‌تر می‌کند.

کاربرد آستانه: عبارت np.array(image) < 128 به طور موثر آستانه ای را برای کل تصویر در مقیاس خاکستری اعمال می کند. این بدان معناست که هر پیکسل با روشنایی کمتر از ۱۲۸ (نقطه میانی مقادیر ممکن از ۰ تا ۲۵۵) روی True (نماینده سفید) و بقیه روی False (سیاه) تنظیم می شود. این عملیات برداری شده در مقایسه با رویکرد تکراری پیکسل به پیکسل در کد اصلی بسیار کارآمد است.

نوع بازگشت: این تابع یک تصویر باینری را به عنوان یک آرایه NumPy برمی گرداند، که برای پردازش تصویر بیشتر یا کارهای یادگیری ماشین ایده آل است. این رویکرد از قابلیت های NumPy برای عملیات با کارایی بالا در آرایه های بزرگ استفاده می کند.

عملکرد: add_noise

این تابع نویز را به یک تصویر باینری اضافه می کند:

Noise Generation: الگوی نویز ایجاد می کند که در آن هر پیکسل با توجه به توزیع دوجمله ای دارای سطح احتمال ۱ (تغییر شده) و ۱ سطح ۰ (بدون تغییر) است. این کنترل خوبی بر روی میزان نویز اضافه شده بر اساس پارامتر سطح فراهم می کند.

کاربرد نویز: np.logical_xor (الگو، نویز) نویز را به الگوی اصلی اعمال می کند. عملیات XOR به طور موثر مقدار پیکسل را در هر جایی که نویز ۱ باشد تغییر می دهد و در نتیجه نویز را در الگو یکپارچه می کند. این عملیات همچنین بردار است و آن را بسیار سریعتر از روش های تکراری معادل می کند.

تبدیل نوع داده: .atype(int) نتیجه یک آرایه بولی (ناشی از عملیات XOR) را به یک آرایه عدد صحیح تبدیل می کند که اغلب برای پردازش تصویر یا کارهای تجسم بیشتر مفید است.

مزایا نسبت به کد قبلی

کارایی: هر دو تابع از عملیات بردار شده استفاده می‌کنند که ذاتی NumPy هستند، که معمولاً بسیار سریع‌تر از حلقه زدن از طریق پیکسل‌های جداگانه مانند کد اصلی هستند. این به ویژه برای پردازش تصاویر بزرگ یا هنگام انجام عملیات به صورت انبوه مفید است.

انعطاف پذیری: تابع add_noise امکان تنظیم آسان سطح نویز را فراهم می کند و عملکرد را در سناریوهای مختلف که در آن مقادیر متفاوتی از نویز مورد نظر است، همه کاره می کند.

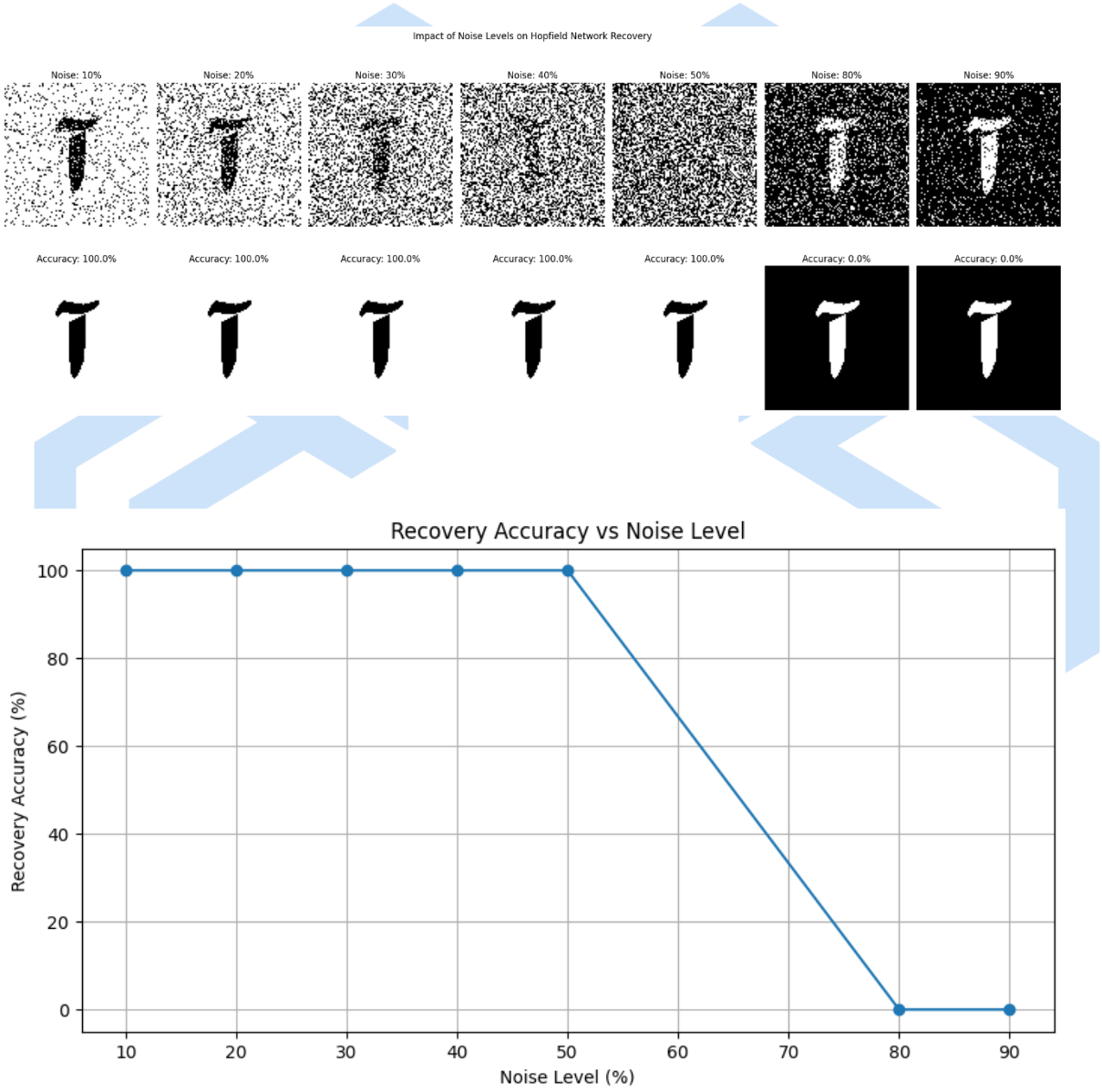
استفاده از NumPy: با استفاده از آرایه ها و عملیات NumPy، این توابع بهتر با پشته محاسباتی علمی پایتون ادغام می شوند و گسترش این عملیات با مراحل پردازش اضافی مانند فیلتر کردن، عملیات مورفولوژی یا ادغام با کتابخانه های یادگیری ماشین را آسان تر می کند.

مقیاس پذیری: استفاده از NumPy و اجتناب از حلقه ها، این توابع را برای تصاویر یا مجموعه داده های بزرگتر مقیاس پذیرتر می کند، که اغلب در برنامه های کاربردی دنیای واقعی مانند تجزیه و تحلیل داده ها، بینایی کامپیوتر یا سیستم های کنترل کیفیت خودکار لازم است.

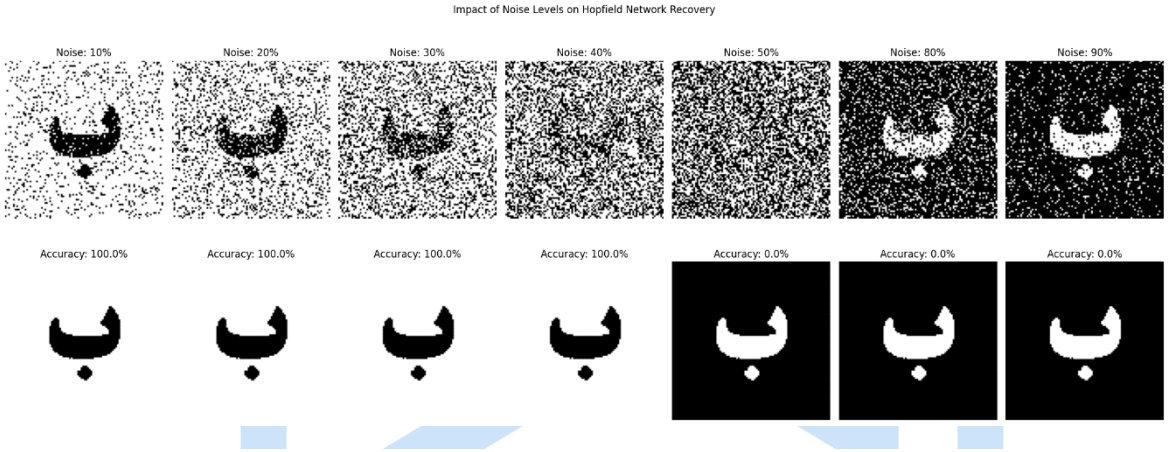
به طور خلاصه، کد بازسازی شده پیشرفت های قابل توجهی را از نظر عملکرد، قابلیت استفاده و یکپارچه سازی با گردش کار پردازش داده های پیشرفته ارائه می دهد.

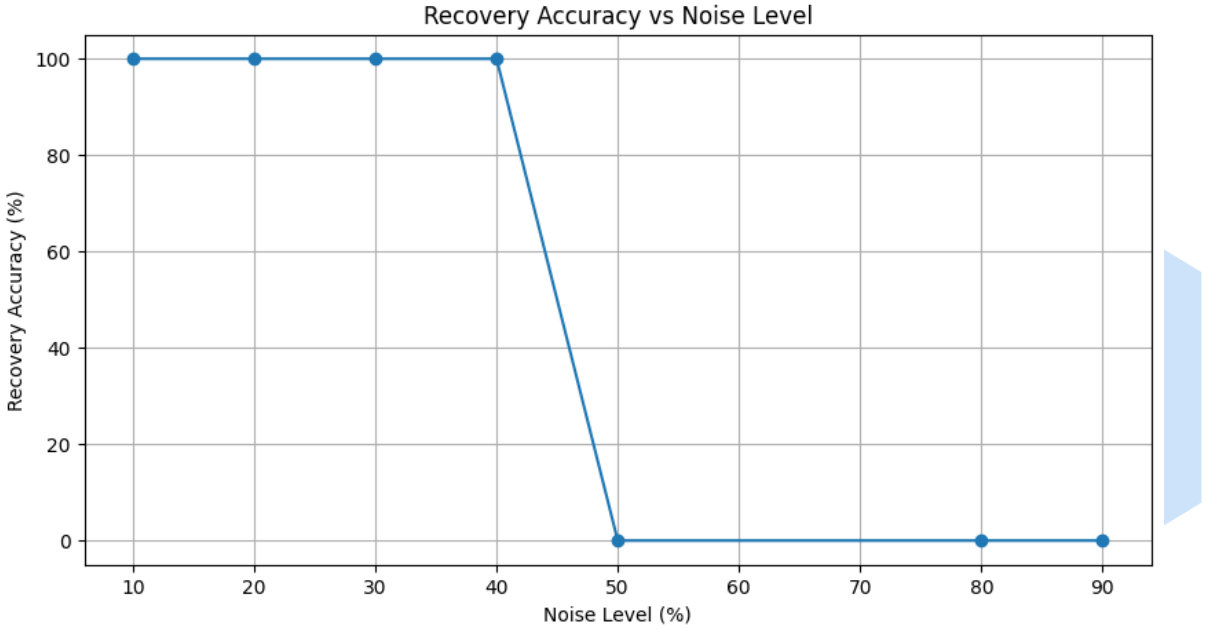
۳.۲

تصاویر زیر انعطاف پذیری شبکه هاپفیلد را در تشخیص و بازسازی الگو در حضور افزایش سطوح نویز نشان می دهد. توانایی شبکه برای یادآوری الگوی اصلی با وجود خرابی نویز، ویژگی معمولی سیستم‌های حافظه انجمنی مانند شبکه‌های هاپفیلد را نشان می‌دهد.

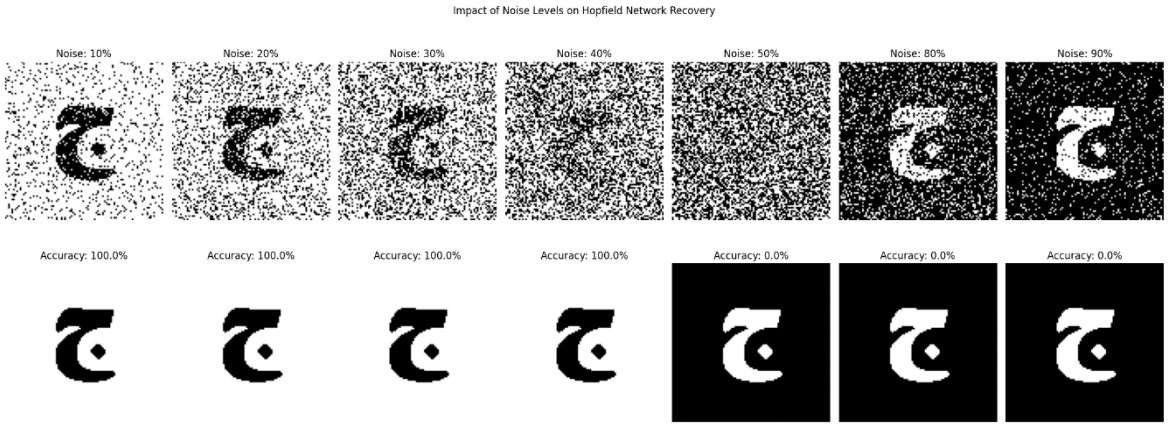


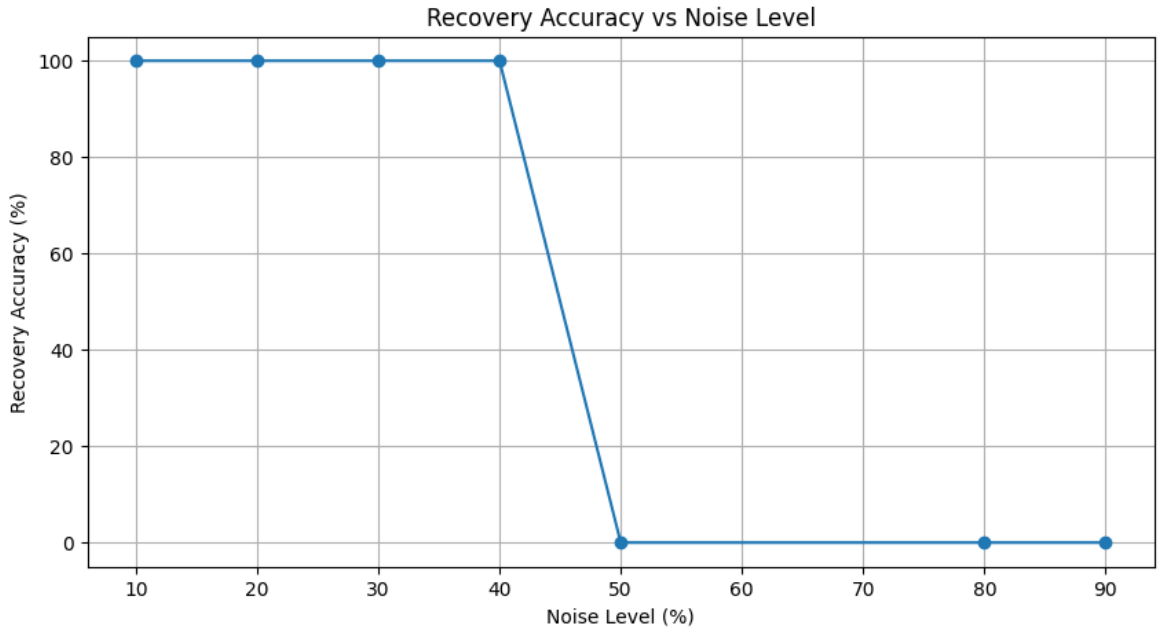
کاراکتر "آ" : شبکه با موفقیت صد درصد تصویر اصلی "آ" را تا ۵۰ درصد نویز به یاد می آورد. فراتر از این سطح، نویز بر قابلیت تشخیص الگو غلبه می کند و منجر به شکست کامل در سطوح نویز ۸۰ و ۹۰ درصد می شود.



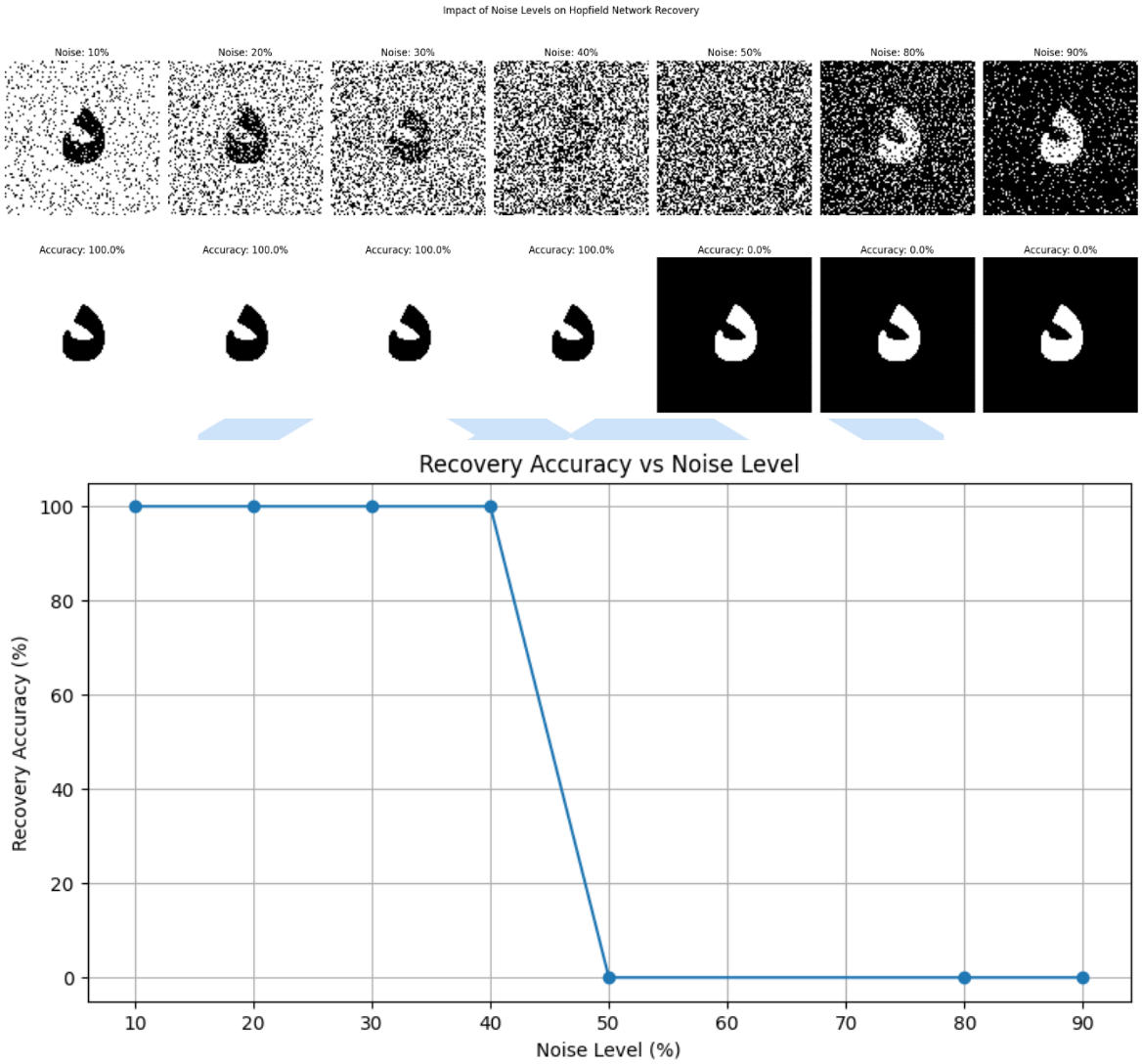


کاراکتر "ب" : "ب" نیز تا ۴۰ درصد نویز به درستی فراخوانی می کند. این الگو در سطوح نویز بالاتر شروع به بدتر شدن می کند و محدودیت های شبکه را نشان می دهد و با بیش از ۵۰ درصد نویز کاملا شکست میخورد.

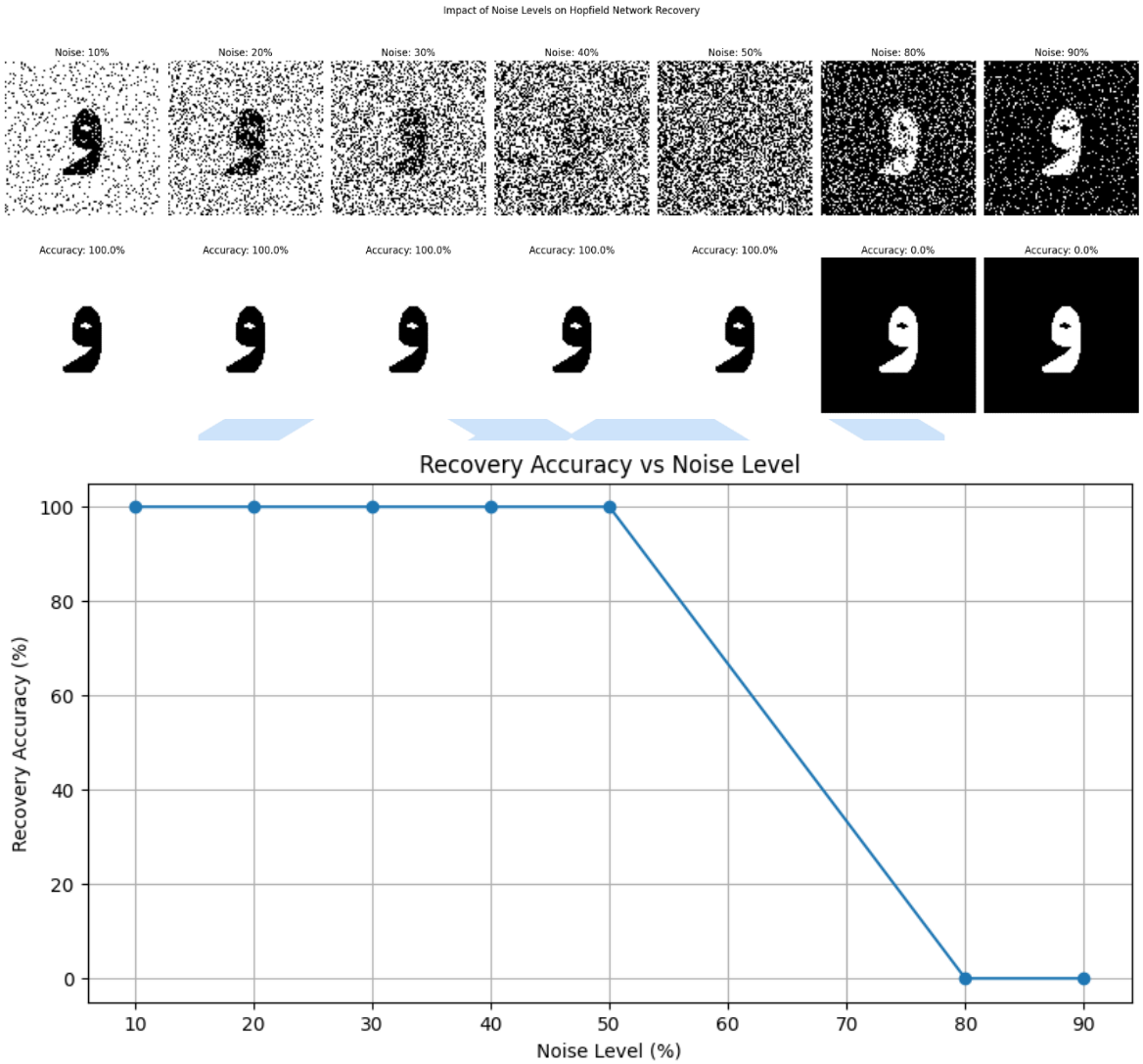




کاراکتر "ج": کاراکتر "ج" کاملاً تا ۴۰٪ نویز حفظ می شود. این الگو در سطوح نویز بالاتر شروع به بدتر شدن می کند و محدودیت های شبکه را نشان می دهد و با بیش از ۵۰ درصد نویز کاملا شکست میخورد.



کاراکتر "د": کاراکتر "د" کاملاً تا ۴۰٪ نویز حفظ می شود. این الگو در سطوح نویز بالاتر شروع به بدتر شدن می کند و محدودیت های شبکه را نشان می دهد و با بیش از ۵۰ درصد نویز کاملا شکست میخورد.



کاراکتر "و" : شبکه با موفقیت صد درصد تصویر اصلی "و" را تا ۵۰ درصد نویز به یاد می آورد. فراتر از این سطح، نویز بر قابلیت تشخیص الگو غلبه می کند و منجر به شکست کامل در سطوح نویز ۸۰ و ۹۰ درصد می شود.

.

در هر مورد، اثربخشی شبکه در مدیریت نویز، محدودیت عملی این نوع مدل را نشان می‌دهد. این رفتار برای شبکه های هاپفیلد معمول است، جایی که دقت فراخوانی به طور قابل توجهی کاهش می یابد زمانی که سطح نویز از ظرفیت شبکه برای متمایز کردن سیگنال از نویز فراتر می رود. این مدل در درک استحکام حافظه انجمنی تحت شرایط مختلف مفید است و می‌تواند به عنوان مبنایی برای مطالعه پیشرفت‌ها یا جایگزین‌هایی در طراحی شبکه عصبی برای انعطاف‌پذیری نویز باشد.

برای تقویت این شبکه، میتوان افزایش تعداد نورون ها را در نظر گرفت که از نظر تئوری می تواند ظرفیت ذخیره سازی و تحمل نویز را بهبود بخشد. از طرف دیگر، بهینه سازی پویایی شبکه، مانند به روز رسانی قوانین یا نرخ یادگیری، ممکن است عملکرد بهتری را در شرایط نویز بالا داشته باشد.

۳.۳

نویز میسینگ (Missing Noise) یا به صورت دقیق‌تر، "داده‌های گمشده به دلیل نویز" به موقعیتی اشاره دارد که در آن بخش‌هایی از داده‌ها در یک سیگنال یا الگو به دلیل حضور نویز یا خطا ناپدید شده‌اند. این نوع نویز می‌تواند به دلایل مختلفی ایجاد شود، مانند خرابی دستگاه‌های سنجش، مشکلات انتقال داده، یا نقص‌های فنی که باعث می‌شود داده‌های دریافتی ناقص یا تحریف شده باشند.

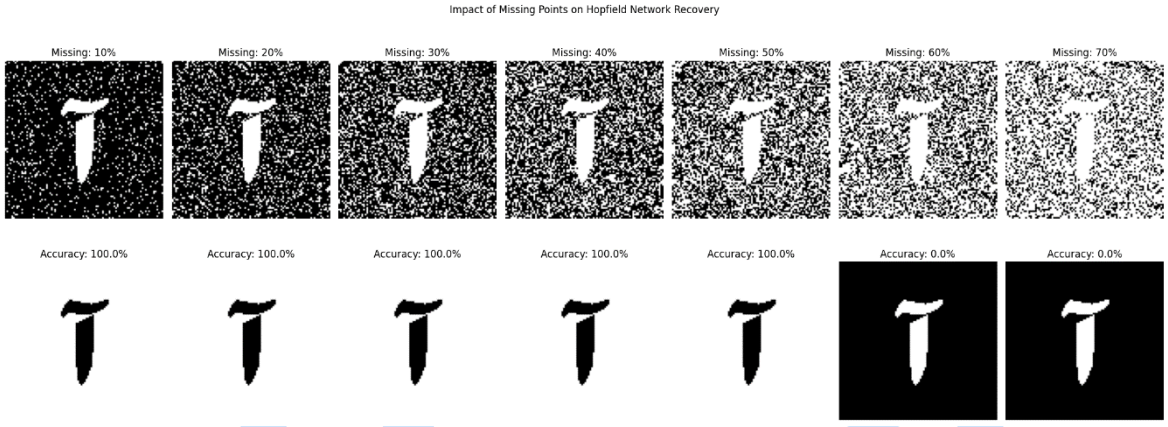
توضیح کلیت تابع add_missing_points

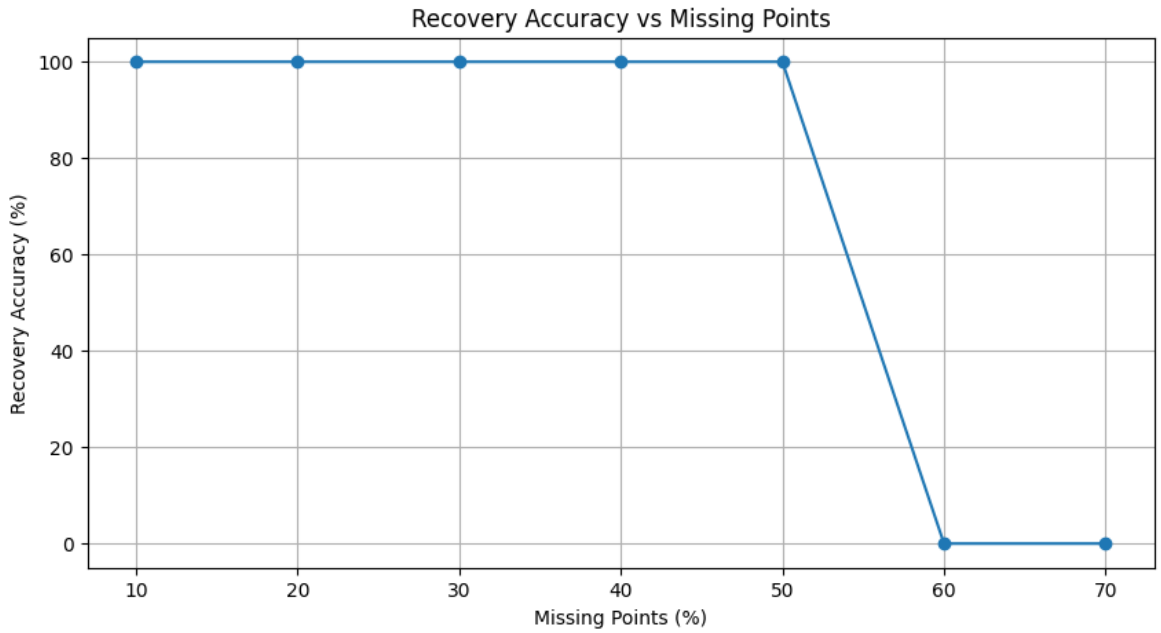
تابع add_missing_points که توضیح داده شد، به منظور شبیه‌سازی این وضعیت در داده‌ها طراحی شده است. این تابع از یک الگوی داده‌ای و یک سطح احتمال به عنوان ورودی استفاده می‌کند تا نقاط گمشده را به صورت تصادفی در الگو ایجاد کند. این کار با استفاده از ماسک تصادفی انجام می‌شود که تعیین می‌کند کدام نقاط باید به عنوان داده‌های گمشده علامت‌گذاری شوند.

فرآیند تابع:

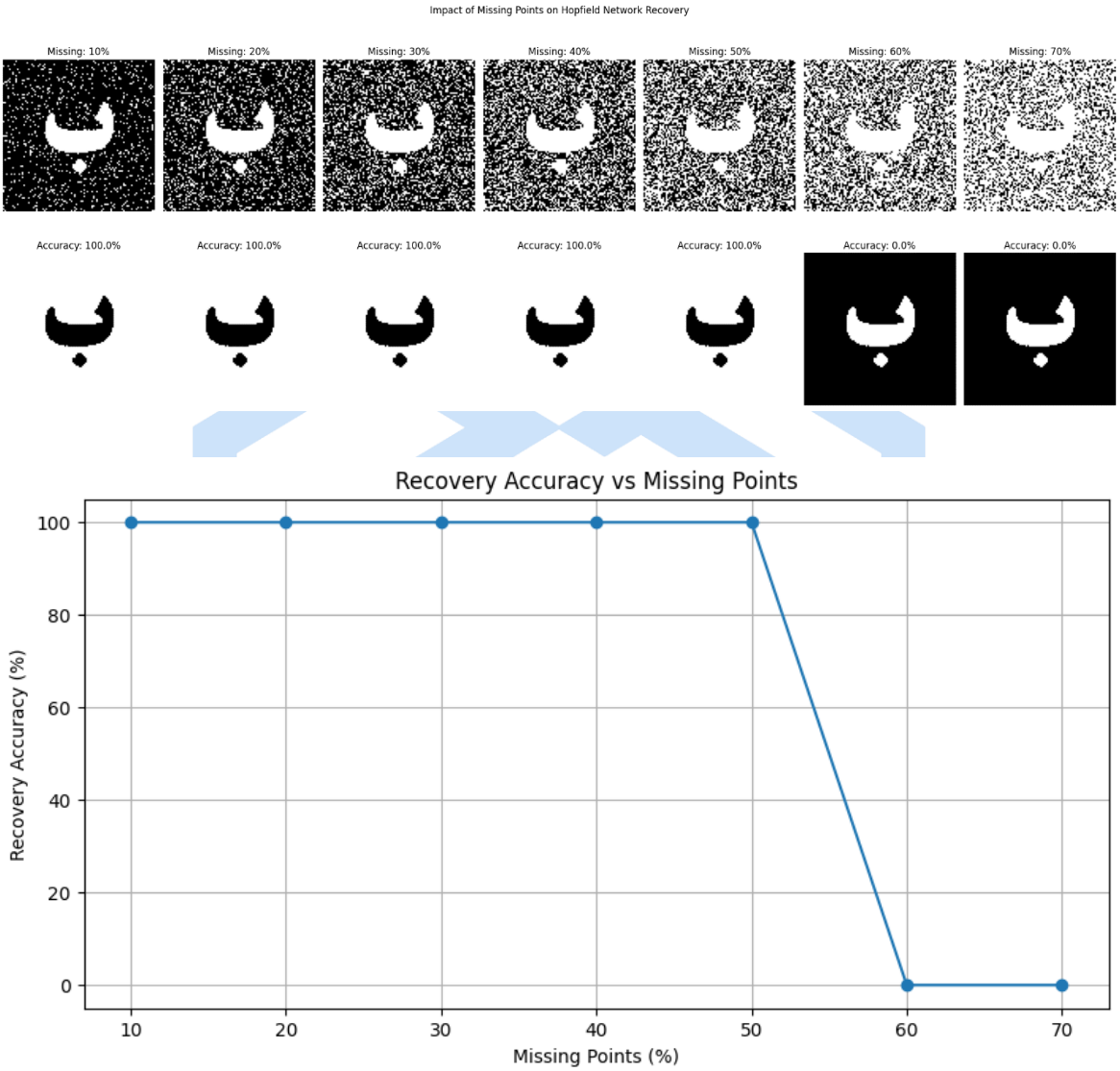
۱. **ایجاد یک ماسک تصادفی** :بر اساس احتمال داده شده ,(level) یک ماسک بولین (صحیح/غلط) تولید می‌شود که هر عنصر آن به طور تصادفی تعیین می‌کند که آیا نقطه مربوطه باید به عنوان گمشده در نظر گرفته شود.

۲. **تعیین نقاط گمشده** :نقاطی که ماسک آن‌ها را به عنوان true نشان می‌دهد، در نسخه کپی شده از الگو، مقدار ۰.۵ را می‌گیرند. این ارزش به صورت نمادین برای نشان دادن داده‌های گمشده انتخاب شده است و می‌تواند بر اساس نیازهای خاص تحلیلی تغییر کند.

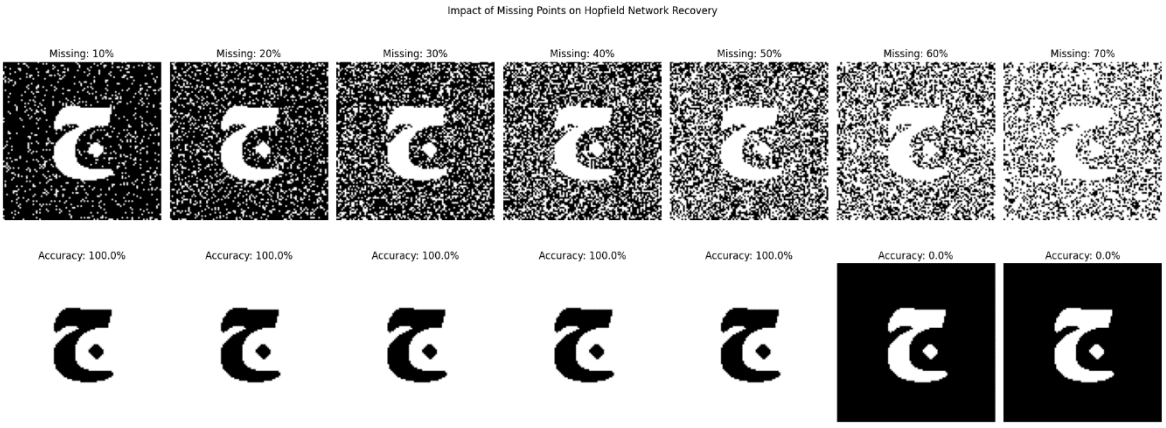


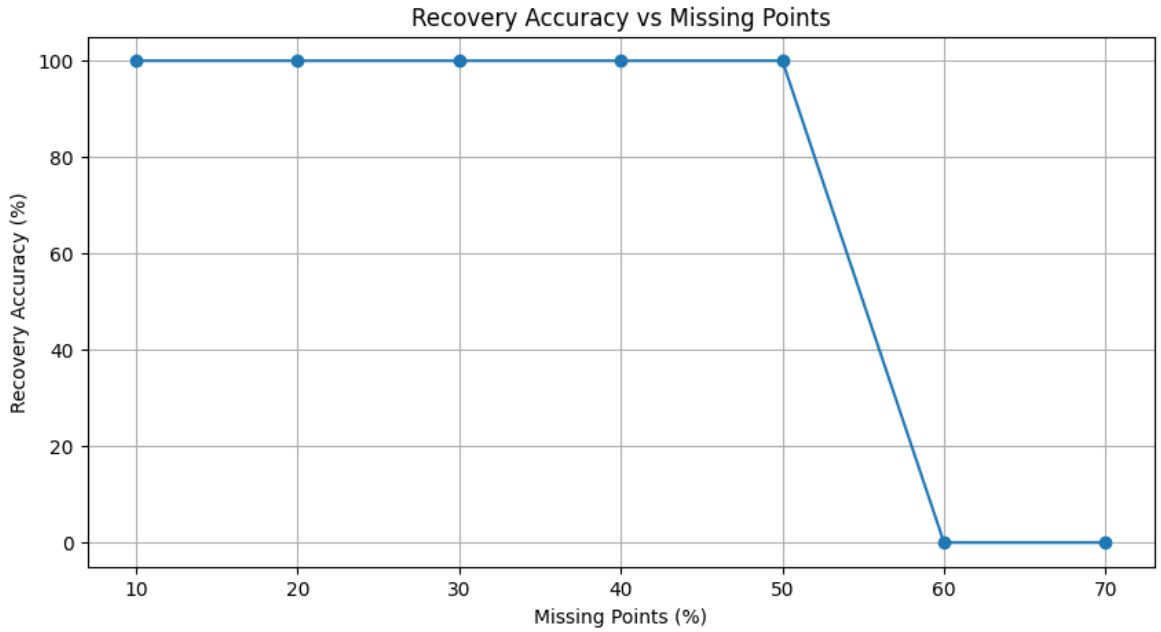


کاراکتر "آ" : شبکه با موفقیت صد درصد تصویر اصلی "آ" را تا ۵۰ درصد نویز به یاد می آورد. فراتر از این سطح، نویز بر قابلیت تشخیص الگو غلبه می کند و منجر به شکست کامل در سطوح نویز ۶۰ و ۷۰ درصد می شود.

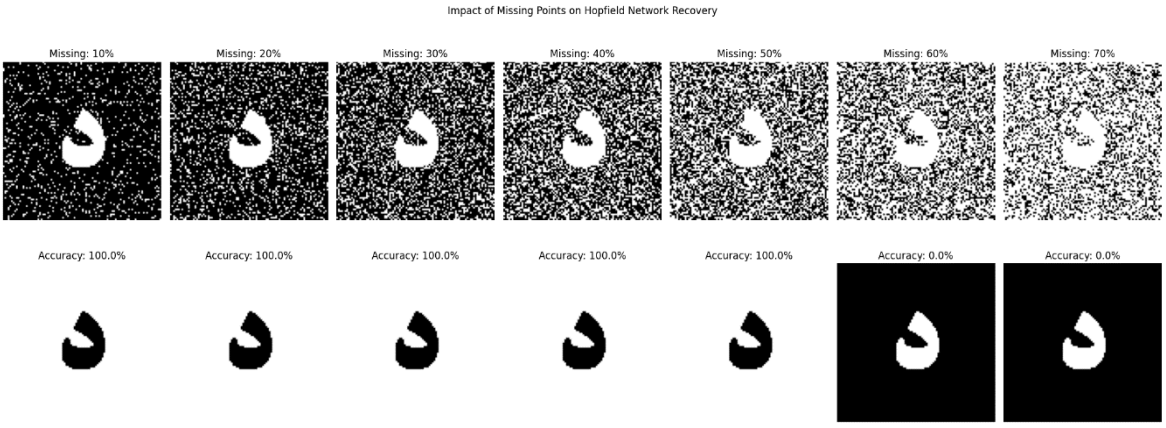


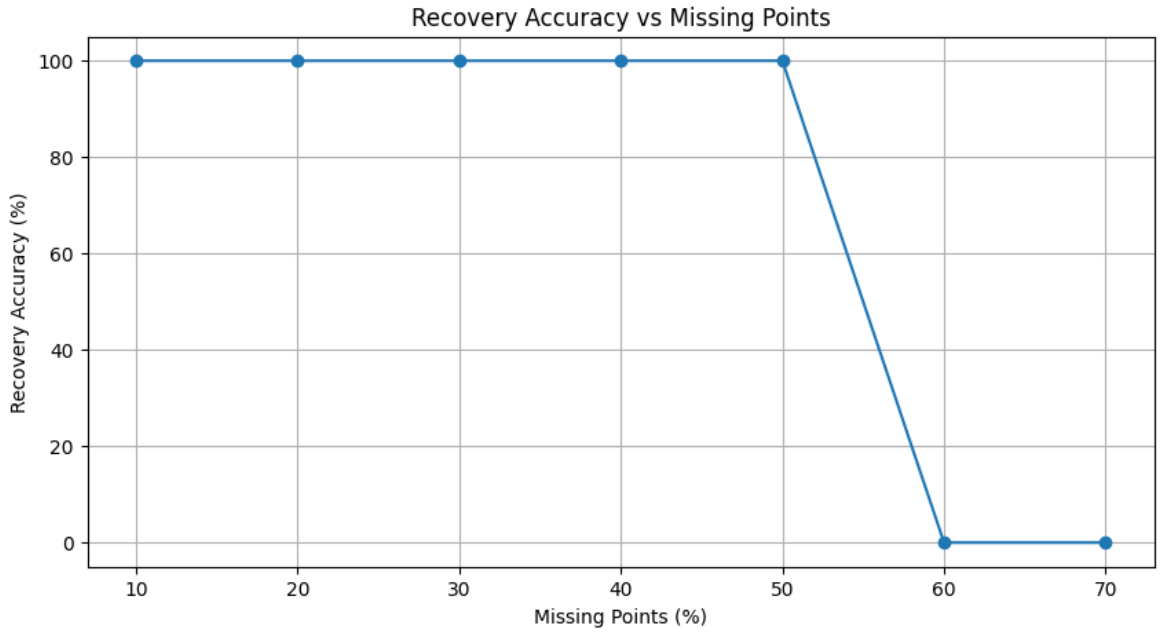
کاراکتر "ب" : شبکه با موفقیت صد درصد تصویر اصلی "ب" را تا ۵۰ درصد نویز به یاد می آورد. فراتر از این سطح، نویز بر قابلیت تشخیص الگو غلبه می کند و منجر به شکست کامل در سطوح نویز ۶۰ و ۷۰ درصد می شود.



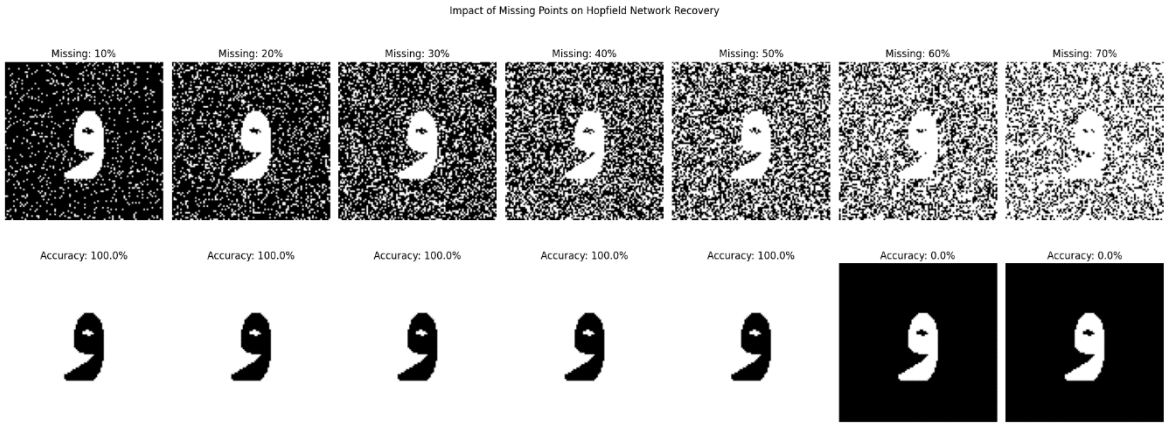


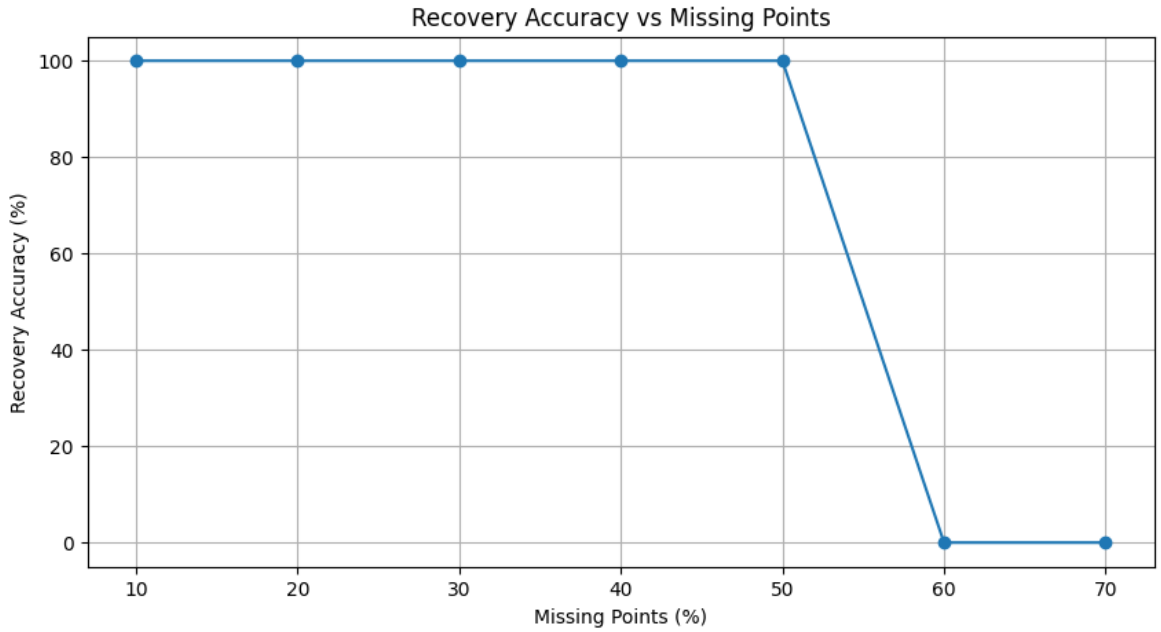
کاراکتر "ج" : شبکه با موفقیت صد درصد تصویر اصلی "ج" را تا ۵۰ درصد نویز به یاد می آورد. فراتر از این سطح، نویز بر قابلیت تشخیص الگو غلبه می کند و منجر به شکست کامل در سطوح نویز ۶۰ و ۷۰ درصد می شود.





کاراکتر "د" : شبکه با موفقیت صد درصد تصویر اصلی "د" را تا ۵۰ درصد نویز به یاد می آورد. فراتر از این سطح، نویز بر قابلیت تشخیص الگو غلبه می کند و منجر به شکست کامل در سطوح نویز ۶۰ و ۷۰ درصد می شود.





کاراکتر "و" : شبکه با موفقیت صد درصد تصویر اصلی "و" را تا ۵۰ درصد نویز به یاد می آورد. فراتر از این سطح، نویز بر قابلیت تشخیص الگو غلبه می کند و منجر به شکست کامل در سطوح نویز ۶۰ و ۷۰ درصد می شود.

شبکه‌های هاپفیلد، که نوعی شبکه عصبی بازگشتی و تکاملی است، برای بازیابی الگوها از حافظه‌های جمعی مورد استفاده قرار می‌گیرند. این شبکه‌ها دارای توانایی بازیابی داده‌ها حتی در حضور نویز هستند، اما این قابلیت تا یک حد خاصی قابل اطمینان است. در زیر دلایلی که چرا شبکه هاپفیلد نمی‌تواند نویز بیش از ۵۰ درصد را بازیابی کند، بررسی شده است:

- **ظرفیت ذخیره‌سازی** :شبکه‌های هاپفیلد دارای محدودیت ظرفیت هستند، به طور معمول که می‌توانند حدود ۰.۱۵ برابر تعداد نورون‌های خود را به عنوان الگوهای مستقل ذخیره کنند. وقتی که نویز بیش از این حد باشد، تعداد نورون‌هایی که اطلاعات نادرست دارند بیشتر می‌شود و شبکه نمی‌تواند به طور موثر الگوها را بازیابی کند.
- **دینامیک بازگشتی** :شبکه‌های هاپفیلد بر پایه دینامیک‌های بازگشتی عمل می‌کنند که در آن‌ها هر نورون با دیگری از طریق وزن‌ها متصل است. نویز بیش از حد می‌تواند این دینامیک‌ها را مختل کند و منجر به پایدار شدن شبکه در حالت‌های غیرمناسب (مینیموم‌های محلی ناخواسته) شود.
- **تداخل** :با افزایش نویز، الگوهای ذخیره شده در شبکه ممکن است شروع به تداخل کنند، به این معنی که بازیابی یک الگو می‌تواند به خطاهایی منجر شود که ناشی از حضور بخش‌هایی از الگوهای دیگر است.
- **خطاهای ورودی** :وقتی نویز به بیش از ۵۰٪ برسد، تعداد خطاها در ورودی‌ها به حدی است که احتمال دارد شبکه نتواند الگوی اصلی را از نویز تشخیص دهد.

راه حل‌ها:

برای مقابله با این محدودیت‌ها و بهبود قابلیت بازیابی شبکه‌های هاپفیلد در حضور نویز بالا، می‌توان رویکردهایی مانند افزایش تعداد نورون‌ها، بهبود الگوریتم‌های یادگیری و وزن‌دهی، استفاده از پیش‌پردازش برای بهبود کیفیت داده‌های ورودی، و استفاده از شبکه‌های عصبی عمیق‌تر یا مدل‌های پیچیده‌تر را در نظر گرفت.

پرسش ۴

آموزش شبکه عصبی

این کد به طور کلی به بررسی عملکرد دو مدل یادگیری عمیق مختلف بر روی مجموعه داده‌های مسکن کالیفرنیا (California Housing) می‌پردازد. هدف اصلی مدل‌ها، پیش‌بینی قیمت خانه‌ها بر اساس ویژگی‌های داده شده (مانند تعداد اتاق‌ها، مساحت، جمعیت و غیره) است. در ادامه، روند کلی کد و ایده‌های پشت مدل‌ها توضیح داده می‌شود:

آماده‌سازی داده‌ها

- **بارگذاری داده‌ها** :مجموعه داده‌ی California Housing از کتابخانه‌ی sklearn بارگذاری می‌شود. این داده شامل ویژگی‌های چندگانه برای پیش‌بینی قیمت خانه‌ها است.
- **نرمال‌سازی داده‌ها** :با استفاده از استانداردسازی(StandardScaler) ، مقیاس تمامی ویژگی‌ها یکسان می‌شود. این کار برای بهبود کارایی مدل‌ها در هنگام یادگیری ضروری است.
- **تقسیم داده‌ها** :داده‌ها به مجموعه‌های آموزشی و آزمایشی تقسیم می‌شوند تا عملکرد مدل‌ها روی داده‌های نادیده‌گرفته‌شده ارزیابی شود.

ساخت مدل‌ها

مدل RBF (Radial Basis Function)

- این مدل از یک لایه‌ی خاص به نام RBF استفاده می‌کند که اساس آن بر پایه‌ی توابع پایه شعاعی (Radial Basis Functions) است.
- **لایه RBF**
  - این لایه برای هر نقطه ورودی فاصله‌ی آن از مراکز مشخصی (Centers) را محاسبه می‌کند.
  - خروجی لایه به صورت نمایی از منفی فاصله‌ها است که بر اساس پارامتر γ تنظیم می‌شود. این مقدار تعیین‌کننده میزان حساسیت مدل به فاصله‌ها است.
- **ویژگی اصلی مدل RBF**
  - این مدل برای داده‌هایی که رفتار غیرخطی دارند مناسب‌تر است. لایه‌ی RBF توانایی مدل‌سازی روابط پیچیده‌تر بین ورودی و خروجی را دارد.

مدل Dense (چند لایه‌ای کلاسیک)

- این مدل شامل دو لایه چگال (Dense) است. لایه‌ی اول ۵۰ نورون با تابع فعال‌سازی ReLU دارد و لایه دوم تنها یک نورون برای پیش‌بینی نهایی.
- **ویژگی اصلی Dense**
  - این مدل بر پایه‌ی معماری کلاسیک شبکه‌های عصبی است و معمولاً در مدل‌سازی روابط خطی و غیرخطی ساده‌تر استفاده می‌شود.

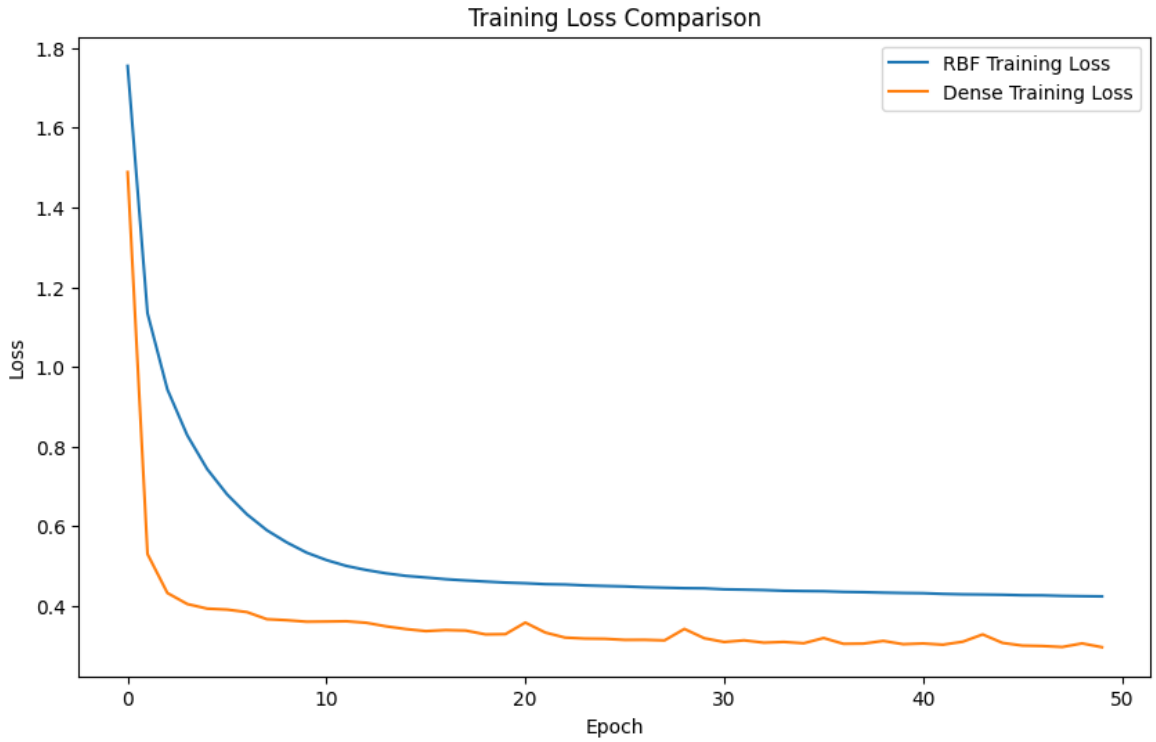
آموزش مدل‌ها

- هر دو مدل با استفاده از الگوریتم Adam و تابع هزینه MSE (میانگین مربعات خطا) آموزش داده می‌شوند.
- داده‌ی آموزشی به دو بخش تقسیم شده (۸۰٪ برای آموزش و ۲۰٪ برای اعتبارسنجی)، و مدل‌ها به مدت ۵۰ دوره آموزش داده می‌شوند.

تفاوت‌ها و تحلیل کلی

۱. **لایه RBF** : مدل RBF به طور مستقیم روابط پیچیده‌تر بین ورودی‌ها و خروجی را یاد می‌گیرد. این مدل معمولاً برای داده‌هایی که الگوهای غیرخطی بیشتری دارند مناسب است.
۲. **مدل Dense** : این مدل انعطاف‌پذیر و عمومی‌تر است و برای طیف گسترده‌ای از مسائل استفاده می‌شود. با این حال، ممکن است در یادگیری روابط پیچیده به تنهایی محدودیت‌هایی داشته باشد.
۳. **کاربردهای متفاوت** : اگر داده‌ی مسکن شامل روابط غیرخطی قوی بین ویژگی‌ها باشد، RBF ممکن است عملکرد بهتری داشته باشد. در مقابل، اگر داده‌ها رفتار خطی‌تر داشته باشند، مدل Dense می‌تواند مناسب‌تر باشد.

ارزیابی مدل شبکه عصبی با دولایه مختلف RBF و Dense



۱. **شروع خطاها:**
   - مدل RBF (خط آبی) در ابتدای آموزش خطای بالاتری نسبت به مدل Dense دارد. این به دلیل پیچیدگی ذاتی لایه RBF است که در شروع نیاز به تنظیم بیشتری دارد.
   - مدل Dense (خط نارنجی) سریع‌تر به خطای پایین‌تر می‌رسد، زیرا این مدل ساده‌تر است و بهینه‌سازی آن راحت‌تر انجام می‌شود.

۲. **کاهش خطا:**
   - خطای مدل RBF به تدریج کاهش می‌یابد و روند آن صاف‌تر است.
   - مدل Dense به سرعت به خطای کمتر می‌رسد اما این کاهش در اوایل آموزش اتفاق می‌افتد و سپس تقریباً ثابت می‌ماند.

۳. **وضعیت نهایی:**
   - هر دو مدل به یک مقدار خطا نزدیک می‌شوند، اما مدل Dense به خطای کمتری در پایان آموزش می‌رسد.
   - این نشان‌دهنده این است که مدل Dense برای این مجموعه داده خاص عملکرد بهتری در یادگیری دارد.

تحلیل عملکرد مدل‌ها

- **مدل RBF**
  - این مدل برای داده‌های با روابط غیرخطی پیچیده طراحی شده است.
  - روند کاهش خطا نشان می‌دهد که این مدل زمان بیشتری برای یادگیری و تنظیم دارد، اما در طولانی‌مدت بهبود پیوسته‌ای نشان می‌دهد.
  - اگرچه این مدل ممکن است در برخی از مسائل غیرخطی بهتر عمل کند، در اینجا به خطای نهایی کمی بالاتر می‌رسد.
- **مدل Dense**
  - این مدل ساده‌تر و عمومی‌تر است و در اینجا عملکرد بهتری نشان داده است.
  - سرعت یادگیری اولیه بالا است، و در مدت زمان کوتاه به خطای پایینی می‌رسد.
  - این موضوع می‌تواند به دلیل این باشد که داده‌های مسکن کالیفرنیا رفتار نسبتاً خطی دارند و مدل Dense برای این نوع داده مناسب‌تر است.

نتیجه‌گیری

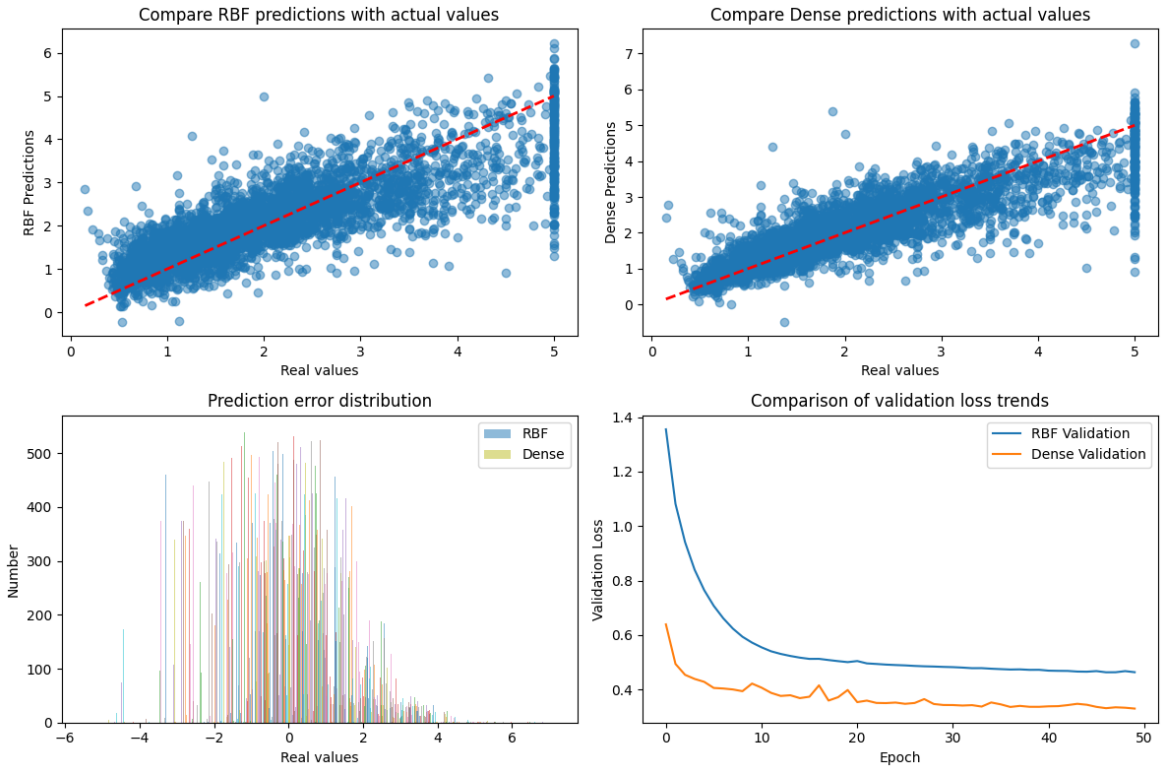
۱. **سرعت یادگیری:**
   - مدل Dense سریع‌تر یاد می‌گیرد.
   - مدل RBF کندتر یاد می‌گیرد اما می‌تواند برای مسائل پیچیده‌تر بهتر باشد.

۲. **مقدار خطا:**
   - در این مسئله خاص، مدل Dense به نظر عملکرد نهایی بهتری دارد و خطای پایین‌تری به دست می‌آورد.

۳. **انتخاب مدل:**
   - برای داده‌های مشابه این مسئله (با روابط خطی یا نیمه‌خطی)، مدل Dense گزینه مناسب‌تری است.
   - اما اگر داده‌ها پیچیدگی بیشتری داشتند، مدل RBF می‌تواند ارزشمندتر باشد.

```
MSE:
RBF: 0.4462
Dense: 0.3161

MAE:
RBF: 0.4838
Dense: 0.3870

R2:
RBF: 0.6595
Dense: 0.7588

RMSE:
RBF: 0.6680
Dense: 0.5623
```

مدل Dense در تمامی معیارها (MSE ، MAE، R^2، RMS) بهتر از RBF عمل کرده است، زیرا داده‌های مسکن کالیفرنیا ساختاری نسبتاً ساده و خطی دارند که Dense برای آن مناسب‌تر است. مدل RBF با وجود عملکرد مناسب، برای مسائل با روابط غیرخطی پیچیده‌تر مؤثرتر خواهد بود. برای این مسئله، Dense بهترین انتخاب است.



نمودار "Compare RBF predictions with actual values" :

- این نمودار پیش‌بینی‌های مدل RBF را در مقایسه با مقادیر واقعی نشان می‌دهد.
- نقاط آبی نمایانگر پیش‌بینی‌های مدل در برابر مقادیر واقعی هستند، و خط قرمز نشان‌دهنده خط ایده‌آل (پیش‌بینی دقیق) است.
- **تحلیل:**
  - توزیع نقاط نسبت به خط قرمز پراکندگی بیشتری دارد، که نشان‌دهنده دقت کمتر مدل RBF است.
  - پیش‌بینی‌ها به صورت کلی الگوی داده‌ها را دنبال می‌کنند اما نوسانات قابل‌توجهی دارند.

نمودار "Compare Dense predictions with actual values" :

- این نمودار مشابه نمودار بالا سمت چپ است، اما برای مدلDense
- **تحلیل:**

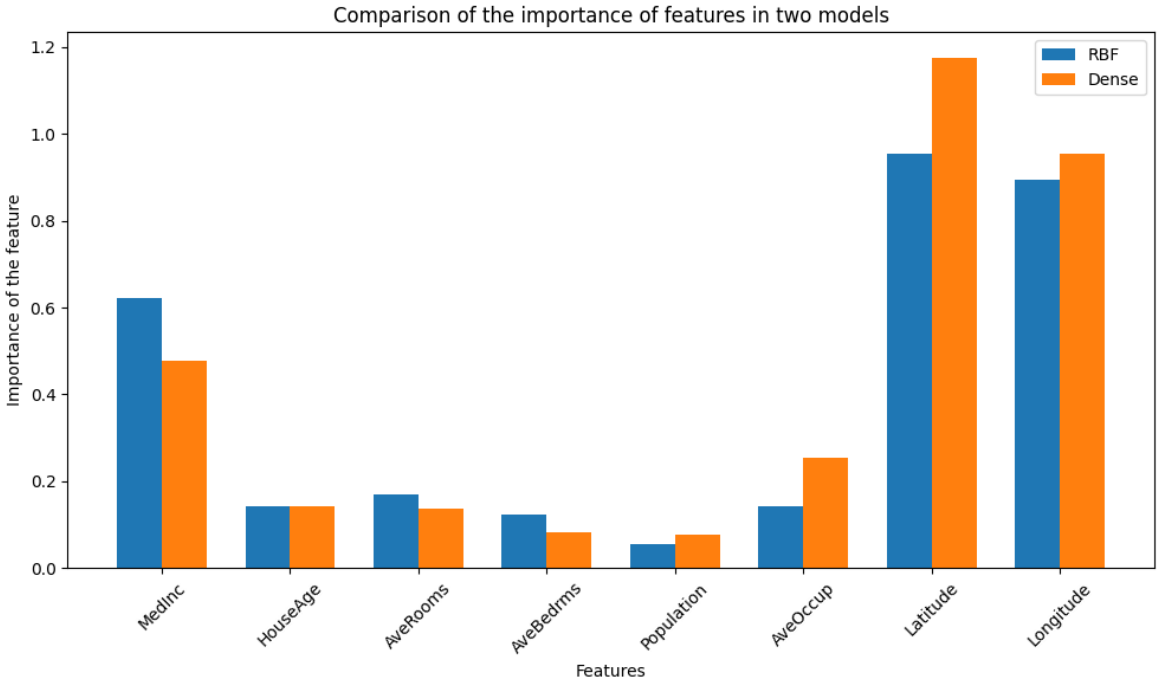
- توزیع نقاط نسبت به مدل RBF نزدیک‌تر به خط قرمز است، که نشان‌دهنده دقت بیشتر مدل Dense است.
- مدل Dense بهتر توانسته روابط بین ورودی و خروجی را یاد بگیرد و مقادیر واقعی را پیش‌بینی کند.

نمودار "Prediction error distribution" :

- این نمودار توزیع خطاهای پیش‌بینی (تفاوت بین مقادیر واقعی و پیش‌بینی شده) را برای هر دو مدل نشان می‌دهد.
- **تحلیل:**
  - خطاهای مدل Dense (نوارهای زرد) به مرکز نزدیک‌تر و دارای انحراف استاندارد کمتری هستند، که نشان‌دهنده پیش‌بینی‌های دقیق‌تر این مدل است.
  - مدل RBF (نوارهای آبی) پراکندگی بیشتری در خطاها دارد و خطاهای بزرگ‌تری تولید کرده است.

نمودار "Comparison of validation loss trends" :

- این نمودار روند کاهش خطای اعتبارسنجی (Validation Loss) برای هر دو مدل را در طول آموزش نشان می‌دهد.
- **تحلیل:**
  - مدل Dense (خط نارنجی) در هر دوره آموزش خطای اعتبارسنجی کمتری دارد و روند کاهش سریع‌تری را نشان می‌دهد.
  - مدل RBF (خط آبی) خطای اعتبارسنجی بیشتری دارد و به آرامی به یک مقدار ثابت می‌رسد، که نشان می‌دهد تطبیق این مدل با داده‌ها دشوارتر بوده است.



این نمودار اهمیت ویژگی‌ها (Feature Importance) در دو مدل RBF و Dense را برای پیش‌بینی قیمت مسکن مقایسه می‌کند. محور افقی ویژگی‌ها را نشان می‌دهد و محور عمودی مقدار اهمیت آن‌ها را برای هر مدل مشخص می‌کند. ستون‌های آبی مربوط به مدل RBF و ستون‌های نارنجی مربوط به مدل Dense هستند.

تحلیل ویژگی‌ها

۱. **MedInc درآمد میانه**
   - در هر دو مدل، این ویژگی اهمیت بالایی دارد، اما مدل RBF وزن بیشتری به آن داده است.
   - این موضوع منطقی است، زیرا درآمد خانوار تأثیر مستقیمی بر قیمت مسکن دارد.

۲. **HouseAge عمر خانه**
   - این ویژگی اهمیت کمی در هر دو مدل دارد و وزن یکسانی در RBF و Dense دریافت کرده است.
   - نشان می‌دهد که عمر خانه تأثیر محدودی در پیش‌بینی قیمت مسکن دارد.

۳. **AveRooms میانگین تعداد اتاق‌ها**

- هر دو مدل وزن مشابهی به این ویژگی اختصاص داده‌اند، اما اهمیت آن نسبت به ویژگی‌های کلیدی دیگر مانند MedInc کمتر است.

٤. **AveBedrms میانگین تعداد اتاق‌های خواب**

- این ویژگی نیز اهمیت کمتری دارد و در هر دو مدل تقریباً به یک میزان استفاده شده است.

٥. **Population جمعیت**

- ویژگی جمعیت در هر دو مدل کم‌اهمیت است، که نشان می‌دهد این متغیر برای پیش‌بینی قیمت خانه در این داده‌ها اطلاعات زیادی فراهم نمی‌کند.

٦. **AveOccup میانگین تعداد افراد ساکن در هر خانه**

- این ویژگی در مدل Dense اهمیت بیشتری نسبت به مدل RBF دارد.
- نشان‌دهنده این است که مدل Dense توانسته اطلاعات بیشتری از این ویژگی استخراج کند.

٧. **Latitude(عرض جغرافیایی) و Longitude (طول جغرافیایی)**

- این دو ویژگی جغرافیایی در هر دو مدل بسیار مهم هستند، اما مدل Dense وزن بیشتری به آن‌ها داده است.
- اهمیت بالای این ویژگی‌ها منطقی است، زیرا مکان خانه یکی از مهم‌ترین عوامل تعیین‌کننده قیمت مسکن است.

مدل **RBF** بیشتر به ویژگی‌های خطی و مستقیم‌تر مانند MedInc متکی است.
مدل **Dense** ویژگی‌های مکانی Latitude و Longitude را بیشتر درک کرده و از آن‌ها استفاده بهینه‌تری کرده است.
برای این مسئله، اهمیت بیشتر ویژگی‌های جغرافیایی در مدل Dense نشان‌دهنده برتری آن در استخراج الگوهای پیچیده‌تر مرتبط با مکان است.