به نام خدا

مبانی سیستم های هوشمند

گزارشکار مینی پروژه دوم

محدثه فیضی- 40007933

استاد درس: دکتر علیاری

.1.1

تابع فعال‌سازی ReLU : در صورتی‌که ورودی کمتر از صفر باشد، این تابع فعال‌سازی صفر و در ورودی‌های بزرگتر از صفر، مقدار خام را خروجی می‌دهد. بازه‌ی خروجی به صورت $[0,\infty)$ است.

تابع فعال‌سازی سیگموید: تابع سیگموید و مشتق آن ساده هستند و مدت زمان ساخت مدل را کاهش می‌دهند، اما از آنجایی که بازه مشتق آن کوتاه است، در این تابع با مشکل info loss مواجه هستیم. هرچه شبکه عصبی ما لایه‌های بیشتری داشته باشد (عمیق‌تر باشد)، در هر لایه اطلاعات بیشتری فشرده‌سازی و حذف می‌شوند، در نتیجه داده‌های بیشتری از بین می‌روند. بازه‌ی خروجی به صورت $[0,1]$ است.

← **ناسازگاری بازه‌ها** :خروجی لایه ReLU می‌تواند مقادیر بزرگی تولید کند که به عنوان ورودی برای تابع سیگموید استفاده می‌شود. وقتی ورودی سیگموید بسیار بزرگ باشد، مقدار خروجی آن به ۱ نزدیک می‌شود. اگر ورودی سیگموید به دلیل ReLU نزدیک به صفر یا خیلی کوچک باشد، خروجی سیگموید نزدیک به 0.5 خواهد بود که منجر به سردرگمی در تصمیم‌گیری می‌شود.

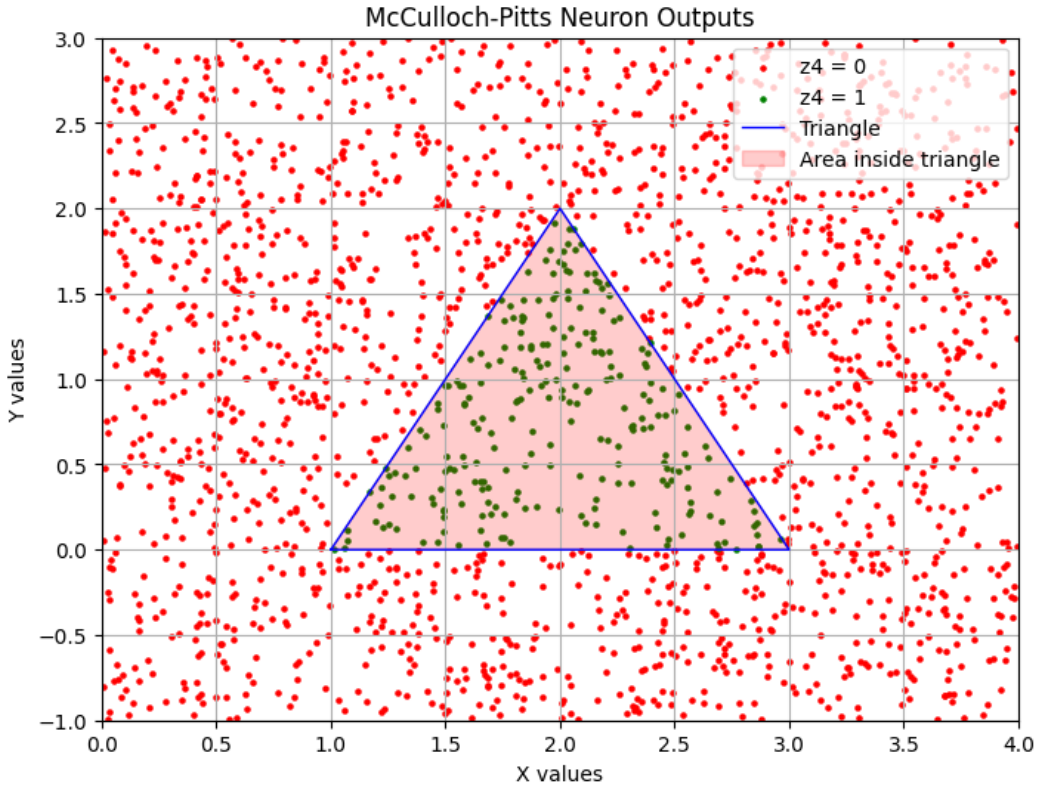
مقادیر اشباع‌شده سیگموید (خیلی نزدیک به ۰ یا ۱) باعث گرادیان‌های بسیار کوچک در فاز یادگیری می‌شوند. همچنین چون خروجی سیگموید عمدتاً اشباع می‌شود، مدل ممکن است یادگیری ضعیفی داشته باشد یا نتواند تصمیم‌های دقیقی بگیرد. مقادیر ورودی بزرگ به سیگموید ممکن است مشکلات overflow ایجاد کنند.

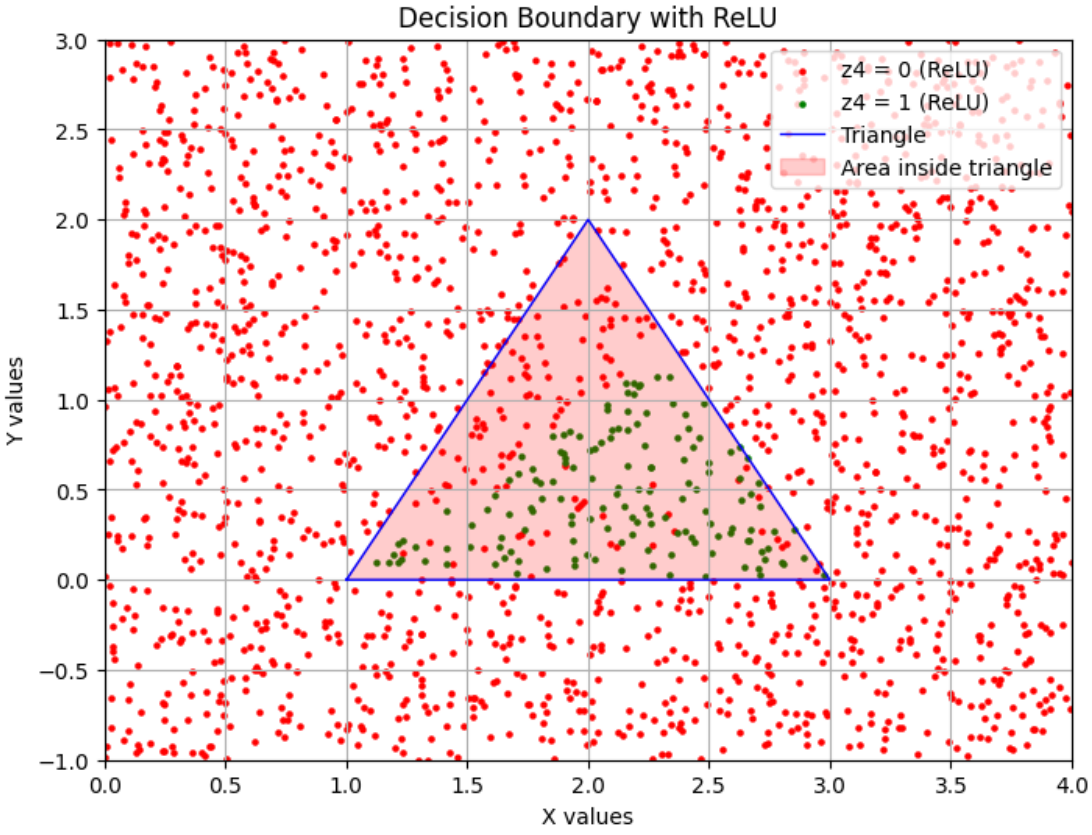
.1.2

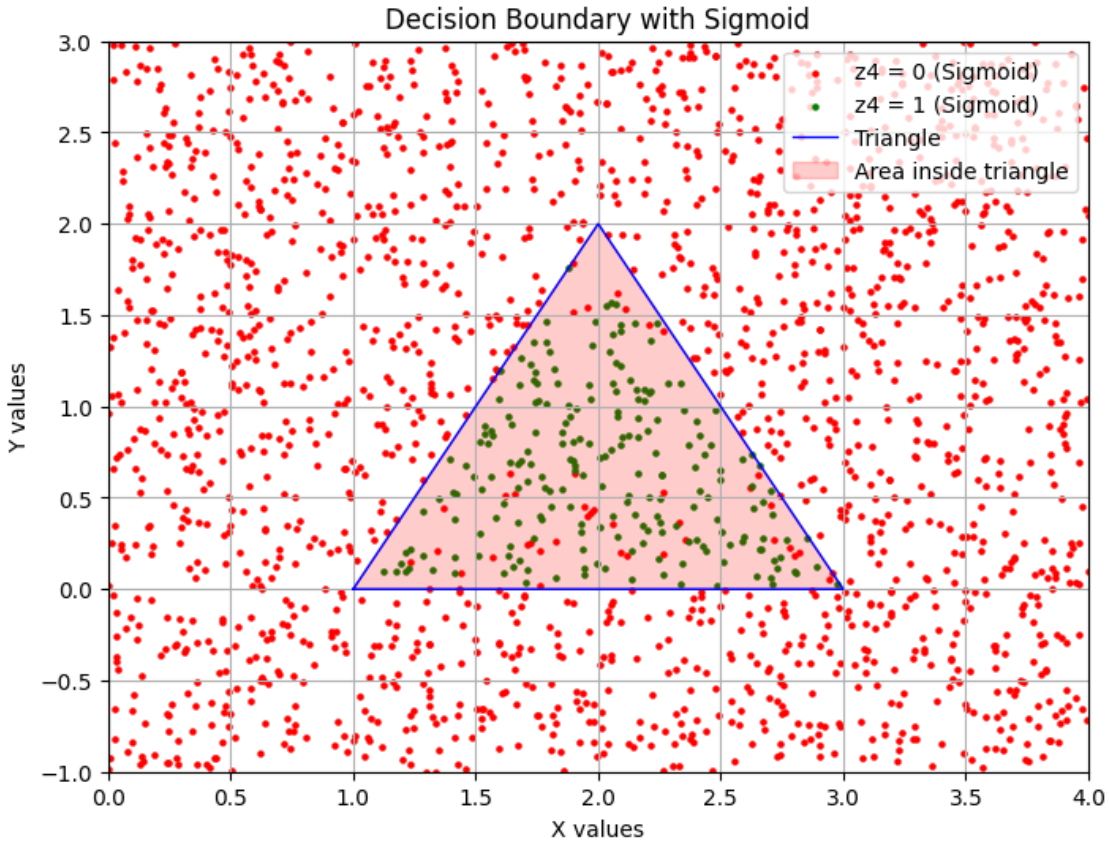
درReLU اگر مقدار ورودی منفی باشد، خروجی صفر می‌شود و گرادیان نیز صفر است. این مشکل باعث می‌شود نرون‌ها در ناحیه‌ای از فضا غیرفعال شوند(Dead Neurons) ولی درELU ، برای مقادیر منفی ورودی، خروجی صفر نیست (بلکه به آرامی به سمت $-\alpha$ همگرا می‌شود) و گرادیان نیز مقدار غیرصفر دارد که باعث می‌شود نرون‌ها حتی در مقادیر ورودی منفی نیز بتوانند یادگیری داشته باشند. برخلاف ReLU که یک نقطه شکست در $x = 0$ دارد، ELU یک انتقال صاف‌تر بین ناحیه مثبت و منفی فراهم می‌کند. این صافی به پایداری و همگرایی بهتر شبکه کمک می‌کند. همچنین ELU تمایل دارد خروجی‌هایی نزدیک به صفر تولید کند (به ویژه برای مقادیر منفی)، که به بهبود پایداری آماری جریان داده در طول شبکه کمک می‌کند.

$$\frac{\partial ELU(x)}{\partial x} = \begin{cases} 1 & if\ x > 0 \\ \alpha e^{x} & if\ x < 0 \end{cases}$$

.1.3


McCulloch-Pitts Neuron Outputs
z4 = 0
z4 = 1
Triangle
Area inside triangle
Y values
X values


Decision Boundary with ReLU
z4 = 0 (ReLU)
z4 = 1 (ReLU)
Triangle
Area inside triangle
Y values
X values



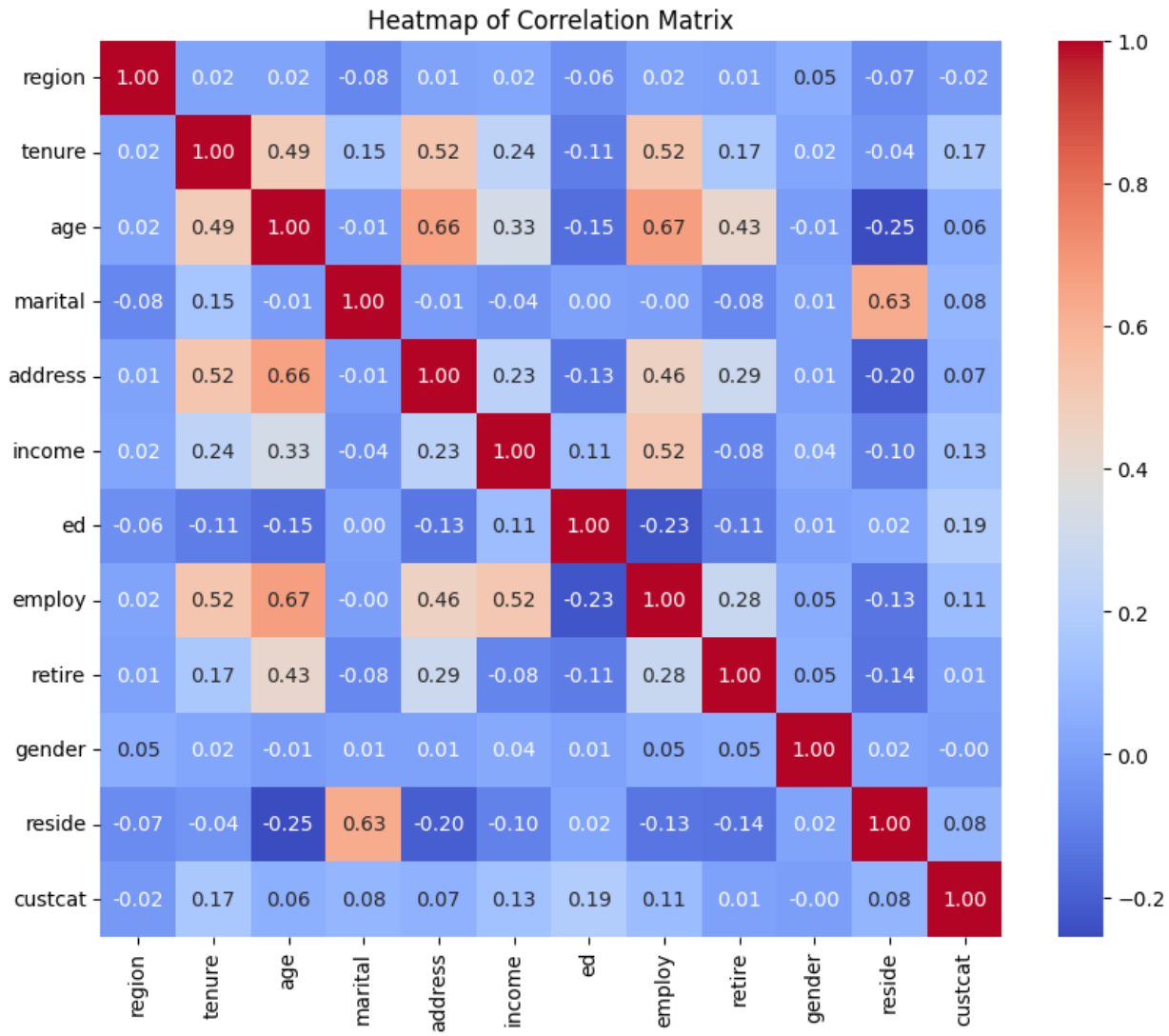
توابع فعال‌ساز به هندسه وابسته نیستند: ReLU یا Sigmoid به‌تنهایی برای بررسی قرارگیری نقاط در داخل مثلث مناسب نیستند. این توابع به اختلاف ورودی‌ها حساس‌اند، اما برای تطبیق دقیق با شکل مثلث باید تصمیم‌گیری به مقادیر دقیق هندسی بستگی داشته باشد.

ReLU: صرفاً مقدار ورودی را به صفر یا مقدار مثبت نگاشت می‌کند. این تابع اطلاعات مربوط به هندسه یا مکان دقیق نقاط را به شکلی که برای حل مسئله شما لازم است، در نظر نمی‌گیرد.

Sigmoid: مقدار ورودی را به بازه [0,1] فشرده می‌کند، اما باز هم هیچ ارتباط مستقیمی با مرزهای هندسی مثلث ندارد.

توابع فعال‌ساز برای این طراحی شده‌اند که در شبکه‌های عصبی، ورودی‌ها را به خروجی‌های غیرخطی تبدیل کنند. هدف آن‌ها تسهیل یادگیری ویژگی‌های پیچیده از داده‌هاست. اما مسئله شما یک مسئله منطقی است (داخل یا خارج مثلث) که به فرمول‌بندی ریاضی دقیق (مانند مساحت‌ها) نیاز دارد، نه ویژگی‌های غیرخطی.

.2.2

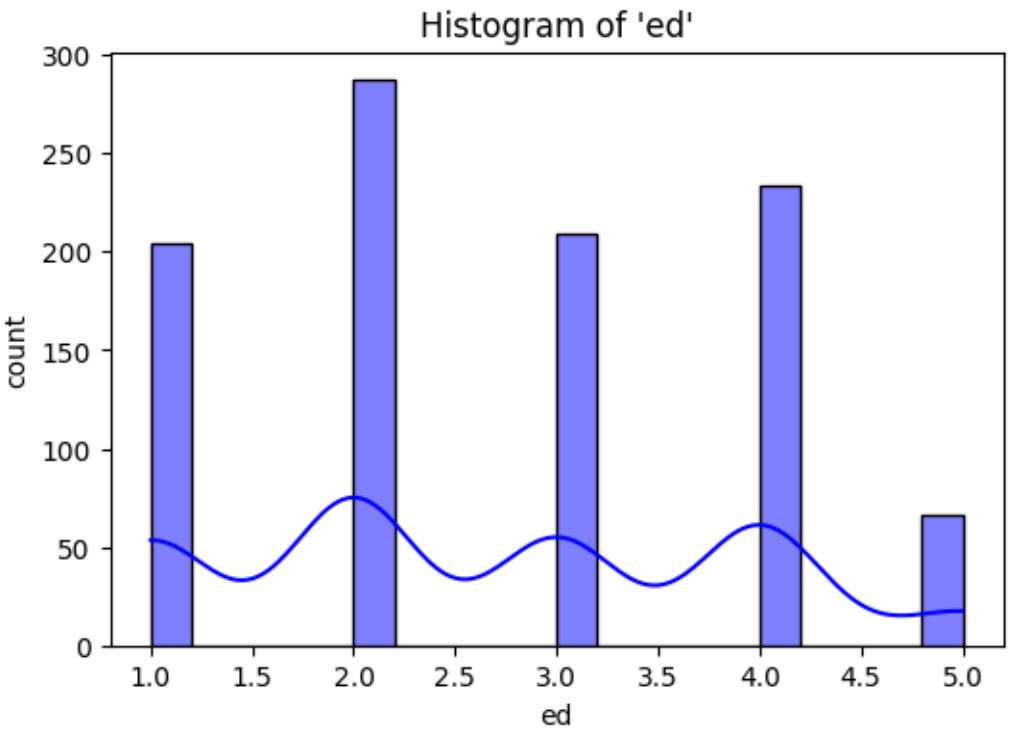


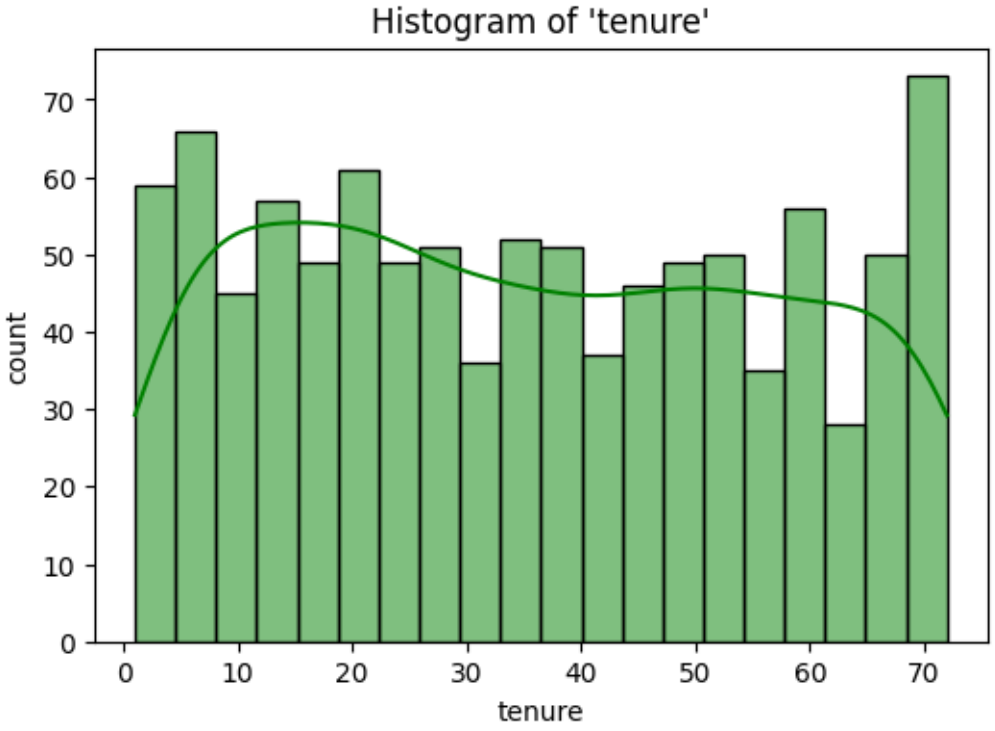
هیت مپ:

رنگ‌های نزدیک به ۱ (معمولاً قرمز) نشان‌دهنده همبستگی مثبت قوی هستند.

رنگ‌های نزدیک به -۱ (معمولاً آبی) نشان‌دهنده همبستگی منفی قوی هستند.

مقادیر نزدیک به ۰ نشان‌دهنده همبستگی ضعیف یا عدم همبستگی هستند.


Histogram of 'ed'
count
ed
0
50
100
150
200
250
300
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Histogram of 'tenure'
count
tenure
0
10
20
30
40
50
60
70

2.3.

Training set: 750 samples, Validation set: 150, Test set: 150

2.4.

افزایش تعداد نرون‌ها و تأثیر آن:

با افزایش تعداد نرون‌ها (از 16 به 32)، دقت مدل‌ها بهبود پیدا کرده است. مدل 1 از **58%** به **78.67%** و مدل 2 از **62.67%** به **72%** افزایش یافته است. این نشان می‌دهد که یک مدل با تعداد نرون بیشتر، توانایی یادگیری ویژگی‌های پیچیده‌تر از داده‌ها را دارد.

کاهش دقت با تعداد نرون‌های بسیار زیاد (64 نرون): وقتی تعداد نرون‌ها از 32 به 64 افزایش یافته، دقت مدل کاهش پیدا کرده است: مدل 1 از **78.67%** به **76%** و مدل 2 از **72%** به **70.67%** کاهش پیدا کرده است. این کاهش ممکن است به دلیل **overfitting** باشد، زیرا مدل با تعداد نرون‌های زیاد ممکن است بیش از حد به داده‌های آموزشی وابسته شود و در داده‌های تست عملکرد ضعیف‌تری داشته باشد.

- دقت مدل 1 در تمامی آزمایش‌ها بهتر یا برابر با مدل 2 بوده است. با 32 نرون، مدل 1 دقت بالاتری نسبت به مدل 2 نشان می‌دهد (**78.67%**در مقابل **72%**) این نشان می‌دهد که افزودن یک لایه مخفی اضافی (مدل 2) همیشه باعث بهبود دقت نمی‌شود. اضافه کردن یک لایه مخفی ممکن است در برخی موارد به پیچیدگی بیش از حد منجر شود.

تاثیر Batch Normalization:

در هر دو مقدار Dropout ، استفاده از Batch Normalization بهبود چشمگیری در عملکرد مدل‌ها داشته است.

بدون Dropout(0.0): مدل 1 دقت از**38.67%** به **82%** و مدل 2 دقت از **50.67%** به **72%** افزایش یافته است.

با Dropout(0.3): مدل 1 دقت از**39.33%** به **74.67%** و مدل 2 دقت از **46.67%** به **74%** افزایش یافته است.

این نشان می‌دهد که Batch Normalization می‌تواند نقش مهمی در پایدارسازی و تسریع یادگیری شبکه‌های عصبی ایفا کند.

تاثیر Dropout:

اضافه کردن Dropout (با نرخ 0.3) در حضور **Batch Normalization** باعث کاهش دقت مدل 1 شده است:
دقت مدل 1 از **82%** به **74.67%** کاهش یافته است.
دقت مدل 2 از **72%** به**74%** تغییر کمی داشته است.
بدون حضور **Batch Normalization**:
دقت در هر دو مدل در سطح بسیار پایینی باقی مانده و Dropout به تنهایی نتوانسته عملکرد را بهبود بخشد.

به طور کلی، مدل 1 عملکرد بهتری نسبت به مدل 2 داشته است، به خصوص زمانی که Batch Normalization فعال بوده است. بدون Dropout و با Batch Normalization، مدل 1 بهترین دقت را ارائه داده.**(82%)**

در برخی موارد مانند Dropout = 0.3 و بدون Batch Normalization ، مدل 2 عملکرد بهتری نسبت به مدل 1 داشته است (**46.67%**در برابر **39.33%**)، اما این بهبود قابل توجه نیست.

تاثیر انواع بهینه‌ساز:
بهینه‌ساز Adam معمولاً دقت بالاتری ارائه داده است، به خصوص برای **مدل2**. به عنوان مثال:
با Dropout = 0.0 و بدون Batch Normalization ، مدل 2 دقت **99.33%** داشته که بسیار بالاست.
با Dropout = 0.3 و بدون Batch Normalization نیز مدل 2 دقت خوبی (**85.33%**) ارائه کرده است.
عملکرد مدل 1 با Adam در بعضی شرایط Dropout = 0.3 و Batch Normalization بهتر از RMSprop بوده است.

بهینه‌ساز RMSprop در برخی شرایط دقت کمتری نسبت به Adam نشان داده است:
مثلاً برای مدل 2 با Dropout = 0.3 و Batch Normalization ، دقت**66.67%** به دست آمده که از Adam کمتر است.
با این حال، در ترکیب با Dropout = 0.0 و بدون Batch Normalization ، عملکرد آن تقریباً مشابه Adam است (**99.33%** برای مدل 2)

Adam عملکرد بهتری در تنظیمات مختلف ارائه داده است. **RMSprop** در تنظیمات خاص مانند بدون Dropout و بدون Batch Normalization عملکرد مشابهی با Adam داشته است.
برای این داده‌ها، مدل 2 با Adam، Dropout = 0.0، و بدون Batch Normalization بهترین ترکیب است و دقت بسیار بالایی (**99.33%**)ارائه می‌دهد.

تاثیر L2 Regularization (learning rate:0.0001):

L2 Regularization به عنوان روشی برای جلوگیری از overfitting با کاهش وزن‌های بزرگ در مدل عمل می‌کند.

مدل‌هایی با L2 Regularization عموماً عملکرد پایدارتری نشان داده‌اند. به‌ویژه در مدل‌هایی که از Dropout یا Batch Normalization استفاده نمی‌کنند، از افت شدید دقت جلوگیری کرده است. عملکرد مدل 2 در حضور L2 Regularization به‌وضوح نسبت به مدل 1 بهتر بوده است.

بهینه‌ساز Adam با L2 Regularization و بدونDropout :

مدل 1: دقت **78.67%** و مدل 2: دقت **91.33%**. این نشان می‌دهد که L2 Regularization در کاهش نوسانات مدل به‌خصوص در لایه‌های عمیق‌تر مؤثر بوده است.

باDropout = 0.3 : عملکرد مدل‌ها کاهش یافت مدل 1، 78% و مدل 2، 87.33%، اما L2 Regularization همچنان به کنترل افت دقت کمک کرده است.

بهینه‌ساز Adam با L2 Regularization و بدونDropout :

مدل 1: دقت **82.00%** و مدل 2: دقت **76.67%**. این نتایج نشان می‌دهد که L2 Regularization در RMSprop تأثیر متوسطی داشته و عملکرد را به‌اندازه Adam بهبود نداده است.

با Dropout = 0.3 : مدل‌ها دقت کمتری داشتند، اما L2 Regularization همچنان باعث شد دقت مدل 2 در سطح قابل قبولی باقی بماند (مدل 1 **78.67%**، مدل 2، **80.00%**)

با Batch Normalization: L2 Regularization به کاهش افت عملکرد کمک کرده است.

Adam، Dropout = 0.0، باBatch Normalization : مدل 1 **78.67%**، مدل 2 .**91.33%**

در حضور Dropout و Batch Normalization ، تأثیر L2 Regularization کمتر بود، اما همچنان مفید.

با Batch Normalization: L2 Regularization بیشترین تأثیر را در جلوگیری از overfitting داشته است.

Adam، Dropout = 0.0، بدونBatch Normalization : مدل 1 **63.33%**، مدل 2 **98.67%**

L2 Regularization به‌ویژه برای مدل‌های عمیق‌تر (مدل 2) بسیار مفید بوده و توانسته است از overfitting جلوگیری کند. در ترکیب با **Adam**، تأثیر بیشتری نسبت به RMSprop داشته است. در شرایطی که Dropout وجود ندارد، کمک کرده تا دقت مدل به حداکثر ممکن برسد. اضافه کردن Dropout تأثیر L2 Regularization را تا حدی کاهش داده، اما همچنان مدل‌ها پایدارتر شده‌اند.

نتایج بدست آمده **پیش‌بینی برای 10 نمونه تصادفی:**

بهینه‌ساز Adam ، بدونDropout ، بدون Batch Normalization ، و با L2 Regularization (0.0001): **100%**

بهینه‌ساز RMSprop ، بدونDropout ، بدون Batch Normalization ، و با L2 Regularization (0.0001): **97.33%**

مدل 2 (Adam): بر اساس دقت کلی بالای 0.9867 و پیش‌بینی‌های درست برای نمونه‌های تصادفی، به نظر می‌رسد که **Adam** به‌طور کلی عملکرد بهتری در شبیه‌سازی و یادگیری از داده‌ها داشته باشد.

مدل 2(RMSProp) : هرچند که دقت کلی 0.9800 برای **RMSprop** خوب است، اما کمی از **Adam** عقب‌تر است. این ممکن است به دلیل ویژگی‌های خاص در داده‌ها یا نحوه تنظیمات این optimizer باشد.

نتایج مدل ترکیبی:

بهبود قابل‌توجهی نشان نداده است و دقت به 0.6333 محدود شده است. این مسئله می‌تواند به چند دلیل باشد:

شباهت ساختاری بین مدل‌ها: هر دو مدل یکی با Adam و دیگری با RMSpropساختار شبکه مشابهی دارند. از آنجا که شبکه‌های عصبی مشابه با تنظیمات مشابهی کار می‌کنند، ممکن است نتایج پیش‌بینی‌های آنها بسیار نزدیک به هم باشند. بنابراین ترکیب پیش‌بینی‌های آنها نمی‌تواند تنوع یا اطلاعات جدیدی به تصمیم نهایی اضافه کند

عدم تفاوت قابل توجه در عملکرد آنها: وقتی دو مدل عملکردی نزدیک به هم داشته باشند، ترکیب آنها ممکن است بهبود جزئی یا حتی عدم بهبود نشان دهد.

عدم استفاده از تنوع در داده‌ها: اگر دو مدل روی همان داده‌ها آموزش دیده باشند و تفاوت اساسی در معماری، تنظیمات، یا نحوه یادگیری نداشته باشند، ترکیب آن‌ها تأثیر زیادی نخواهد داشت.

روش ساده برای ترکیب: روش ترکیب پیش‌بینی‌ها که در اینجا اکثریت آرا (majority voting) است، بسیار ساده است و ممکن است قدرت مدل ترکیبی را محدود کند. روش‌های پیشرفته‌تر مانند میانگین احتمال (probability averaging) یا یادگیری یک مدل (meta-learner) برای ترکیب نتایج می‌توانند دقت ترکیبی را بهبود دهند.

3.1.

تابع convertImageToBinary:

این تابع یک تصویر را دریافت کرده و آن را به نمایش باینری تبدیل می‌کند. پیکسل‌های روشن (سفید) به مقدار -1 و پیکسل‌های تیره (سیاه) به مقدار 1 تبدیل می‌شوند.

عملکرد تابع:

تصویر را از مسیر داده‌شده باز می‌کند. عرض و ارتفاع تصویر را دریافت کرده و به پیکسل‌ها دسترسی پیدا می‌کند. برای هر پیکسل مقادیر RGBرا می‌خواند، مجموع شدت (intensity) را محاسبه می‌کند، اگر شدت از مقدار مشخصی براساس factor بیشتر باشد، پیکسل سفید می‌شود و مقدار 1- به لیست اضافه می‌شود و در غیر این صورت، پیکسل سیاه شده و مقدار 1 به لیست اضافه می‌شود. تصویر به صورت اصلاح‌شده (با تغییر رنگ‌ها) نیز نمایش داده می‌شود. خروجی نهایی لیستی از مقادیر باینری است که نماینده تصویر است.

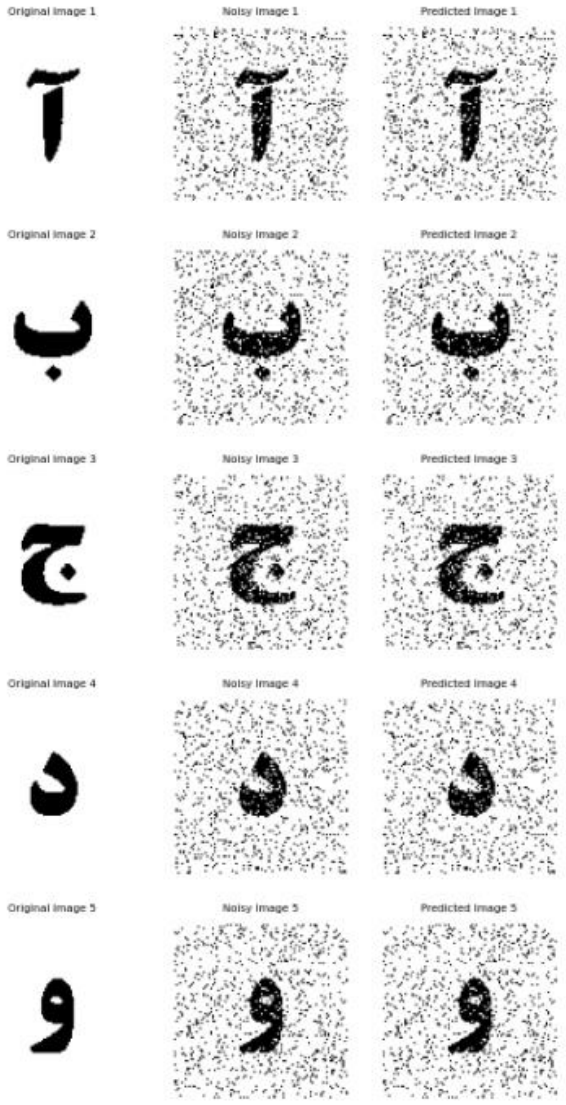
تابع getNoisyBinaryImage:

این تابع، نویز تصادفی به تصویر اضافه می‌کند و نسخه‌ای نویزی از آن تولید و ذخیره می‌کند.

عملکرد تابع:

تصویر ورودی را باز می‌کند. عرض و ارتفاع تصویر را دریافت کرده و به پیکسل‌ها دسترسی پیدا می‌کند. برای هر پیکسل یک مقدار نویز تصادفی تولید می‌کند، نویز را به مقادیر RGB اضافه می‌کند، مقادیر RGB را در بازه مجاز (0 تا 255) محدود می‌کند، مقدار نویزی را به پیکسل‌ها اعمال می‌کند، تصویر نویزی را در مسیر خروجی ذخیره می‌کند.

.3.2



در نویز کم 20٪: شبکه باید به راحتی قادر به بازیابی الگوی صحیح باشد. خروجی پیش‌بینی‌شده باید مشابه با تصویر اصلی باشد.
در نویز متوسط 40٪تا 60٪: شبکه ممکن است کمی اشتباهات جزئی داشته باشد، اما هنوز هم باید بتواند الگو را به‌درستی بازسازی کند.
در نویز بالا 80٪: شبکه ممکن است برای بازسازی الگو با مشکلات جدی مواجه شود. در این حالت، احتمالاً پیش‌بینی‌ها با خطاهای زیادی روبه‌رو می‌شوند و الگوهای بازیابی‌شده با نویز زیاد تفاوت‌های زیادی با تصویر اصلی دارند.
در نویز 100٪: با نویز کامل، شبکه احتمالاً نمی‌تواند هیچ‌گونه اصلاحی انجام دهد و نتایج پیش‌بینی‌شده کاملاً بی‌معنی خواهند بود. زیرا شبکه هاپفیلد معمولاً برای بازسازی الگوها به حداقل اطلاعات از تصویر نیاز دارد.

شبکه‌های عصبی هاپفیلد به‌ویژه در صورتی که میزان نویز بالا باشد، ممکن است نتوانند به درستی به وضعیت پایدار همگرا شوند. این موضوع به دلیل این است که شبکه عصبی هاپفیلد برای بازسازی تصاویر نویزی که اطلاعات زیادی از تصویر اصلی را از دست داده‌اند، به طور طبیعی

ضعیف عمل می‌کند. همچنین شبکه نمی‌تواند شباهت‌های لازم برای بازسازی صحیح تصویر را بیابد. در چنین شرایطی، ممکن است شبکه به وضعیت‌های پایدار اشتباه همگرا شود که باعث می‌شود تصویر پیش‌بینی‌شده از تصویر اصلی بسیار دور باشد.

.3.3

عملکرد شبکه هافیلد با افزایش میزان نقاط گم‌شده کاهش می‌یابد و ممکن است از یک حد خاص(در کد: 0.3 به بالا) به بعد دچار اختلال شود. این شبکه‌ها برای بازیابی الگوهای ناقص طراحی شده‌اند. این شبکه‌ها می‌توانند با دقت نسبی الگوهای گم‌شده یا نویزی را بازسازی کنند، اما وقتی درصد نقاط گم‌شده از یک حد خاص فراتر رود، شبکه به دلیل کمبود اطلاعات صحیح، قادر به شبیه‌سازی دقیق تصویر اصلی نخواهد بود.

راه‌حل‌ها و روش‌های بهبود عملکرد:

- افزایش تعداد تکرارها یا گام‌ها در فرآیند بازیابی: می‌توان تعداد مراحل پیش‌بینی را افزایش داد. در صورتی که شبکه هافیلد بیشتر تلاش کند تا به همگرایی برسد، احتمال موفقیت آن در بازسازی تصویر افزایش می‌یابد، حتی در صورتی که درصد نقاط گم‌شده بالا باشد.

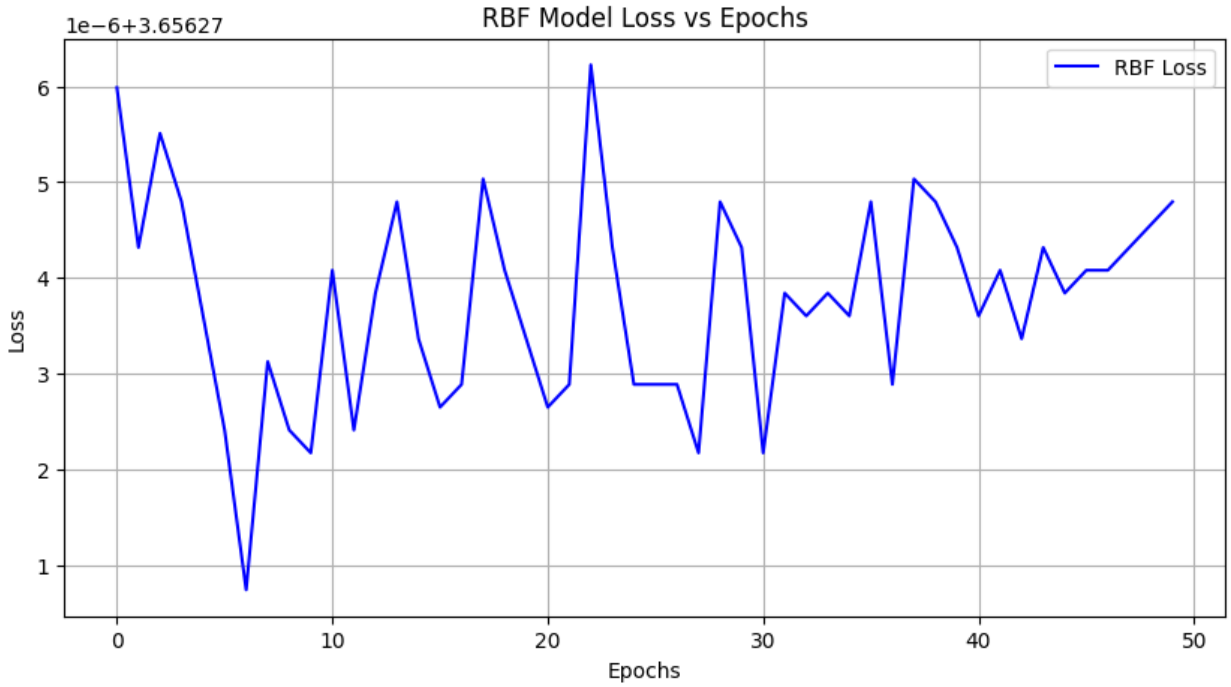
- استفاده از شبکه‌های هافیلد چندلایه(Multilayer Hopfield Network) : این شبکه‌ها می‌توانند ظرفیت بیشتری برای یادگیری و بازیابی الگوهای پیچیده‌تر داشته باشند و در برابر داده‌های نویزی مقاوم‌تر هستند.

- استفاده از تکنیک‌های پیش‌پردازش برای کاهش نویز یا بازسازی داده‌های گم‌شده: برای کاهش اثرMissing Points ، می‌توان از تکنیک‌هایی مانند **تقویت تصویر (image inpainting)** یا **interpolation** استفاده کرد تا نقاط گم‌شده پیش از ورودی به شبکه، بازسازی شوند. به طور خاص، **فیلتر کردن نویز** و **پر کردن نقاط گم‌شده با استفاده از روش‌های هوشمندانه‌تر** می‌تواند دقت پیش‌بینی شبکه را بهبود دهد.

- استفاده از شبکه‌های عصبی پیشرفته‌تر مانند شبکه‌های عصبی کانولوشنی(CNN) : این شبکه‌ها برای شناسایی ویژگی‌های پیچیده‌تر در تصاویر مناسب هستند و قادرند در برابر نویز و Missing Points مقاومت بیشتری نشان دهند

- تقویت شبکه با استفاده از داده‌های بیشتری: استفاده از مجموعه داده‌های بزرگتر و متنوع‌تر برای آموزش شبکه هافیلد می‌تواند توانایی شبکه در شبیه‌سازی بهتر و دقیق‌تر تصاویر را تقویت کند. شبکه هافیلد با داده‌های بیشتر می‌تواند به‌طور بهتری ویژگی‌های تصویر را یاد بگیرد و در برابر نویز و داده‌های گم‌شده مقاومت کند.

.4.6



4.7.

هر مدل به مدت 50 دوره روی داده‌های آموزشی با اندازه دسته 32 آموزش داده می‌شود.

خروجی evaluate برای مدل RBF فقط loss است زیرا تنها معیار تعریف‌شده برای آن MSE است ولی برای مدل Dense شامل دو مقدار است loss (MSE) و MAE در نتیجه مقدار loss هر مدل نمایش داده می‌شود تا بتوان عملکرد دو مدل را مقایسه کرد. مدل با مقدار MSE کمتر عملکرد بهتری دارد چون نشان‌دهنده خطای کمتر در پیش‌بینی است.

توجیه عملکرد:

لایه RBF به دلیل استفاده از هسته‌های شعاعی (Radial Basis Functions) و محاسبه فاصله، ممکن است در شناسایی روابط پیچیده و غیرخطی بین ویژگی‌ها بهتر عمل کند. اگر پارامترها به‌درستی تنظیم نشوند، ممکن است مدل به‌خوبی عمل نکند.
لایه‌های Dense به دلیل توانایی یادگیری مستقیم وزن‌ها از داده‌ها، معمولاً برای بسیاری از مسائل مناسب هستند. اگر داده‌ها به‌خوبی نرمال‌سازی شده باشند و معماری مدل مناسب انتخاب شود، عملکرد بهتری خواهد داشت. عملکرد خوب آن به دلیل تنظیم بهینه تعداد نورون‌ها و لایه‌ها برای مسئله است.

چون در نتایج به دست آمده خطای Dense کمتر از RBF است مدل Dense بهتر عمل کرده است. این می‌تواند به دلیل انعطاف‌پذیری و تعداد زیاد پارامترهای قابل یادگیری باشد. اگر برعکس بود مدل RBF بهتر عمل کرده بود و این می‌تواند نشان‌دهنده قدرت این مدل در شناسایی روابط غیرخطی بین ویژگی‌ها باشد.

RBF Model Loss (MSE): 3.76038098335266

Dense Model Loss (MSE): 0.2710062265396118