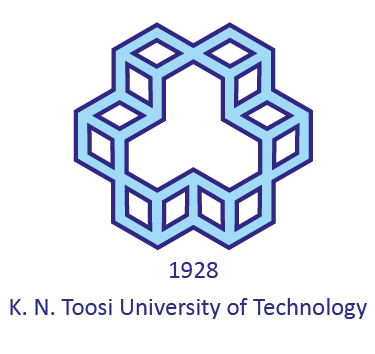
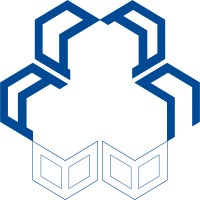
به نام خدا





دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

**مبانی هوشند سازی**

**گزارش تمرین شماره 2**

**سجادرجبی باغستان**

**40005393**

**استاد : جناب دکتر مهدی علیاری**

بهمن 1403

##### فهرست مطالب

|  |  |
| --- | --- |
| عنوان | شماره صفحه |

[سوالات 3](#_Toc187958431)

[سوال اول 3](#_Toc187958432)

[1. 3](#_Toc187958433)

[2. 3](#_Toc187958434)

[سوال دوم 5](#_Toc187958435)

[سوال سوم 20](#_Toc187958436)

[سوال چهارم 23](#_Toc187958437)

# سوالات

## سوال اول

### 

**حذف اطلاعات مهم توسط ReLU:**

فعال‌ساز ReLU ممکن است تمام مقادیر منفی را به صفر تبدیل کند. در نتیجه، اگر اطلاعات مهمی در مقادیر منفی باشد، این اطلاعات از دست می‌روند. این می‌تواند باعث شود که مدل عملکرد بهینه‌ای نداشته باشد.

**اعوجاج خروجی سیگموید:**

خروجی سیگموید با فرض اینکه ورودی آن می‌تواند مقادیر مثبت و منفی باشد طراحی شده است. اگر ورودی سیگموید تنها غیرمنفی باشد (به دلیل ReLU در لایه قبلی)، احتمال تولید شده توسط سیگموید ممکن است به طور غیرطبیعی به سمت ۵/۰ یا 1 متمایل شود. این می‌تواند عملکرد مدل را تحت تأثیر قرار دهد.

**عدم سازگاری معماری با توابع زیان:**

در مسائل طبقه‌بندی دوکلاسه، معمولاً از تابع زیان باینری کراس‌انتروپی (BCE) استفاده می‌شود. این تابع به ورودی سیگموید متکی است که مقادیر متقارن و با توزیع کامل از مثبت و منفی را می‌گیرد. اگر ورودی سیگموید توسط ReLU محدود به مقادیر غیرمنفی شده باشد، محاسبات احتمال و گرادیان دچار اشکال خواهند شد.

### 

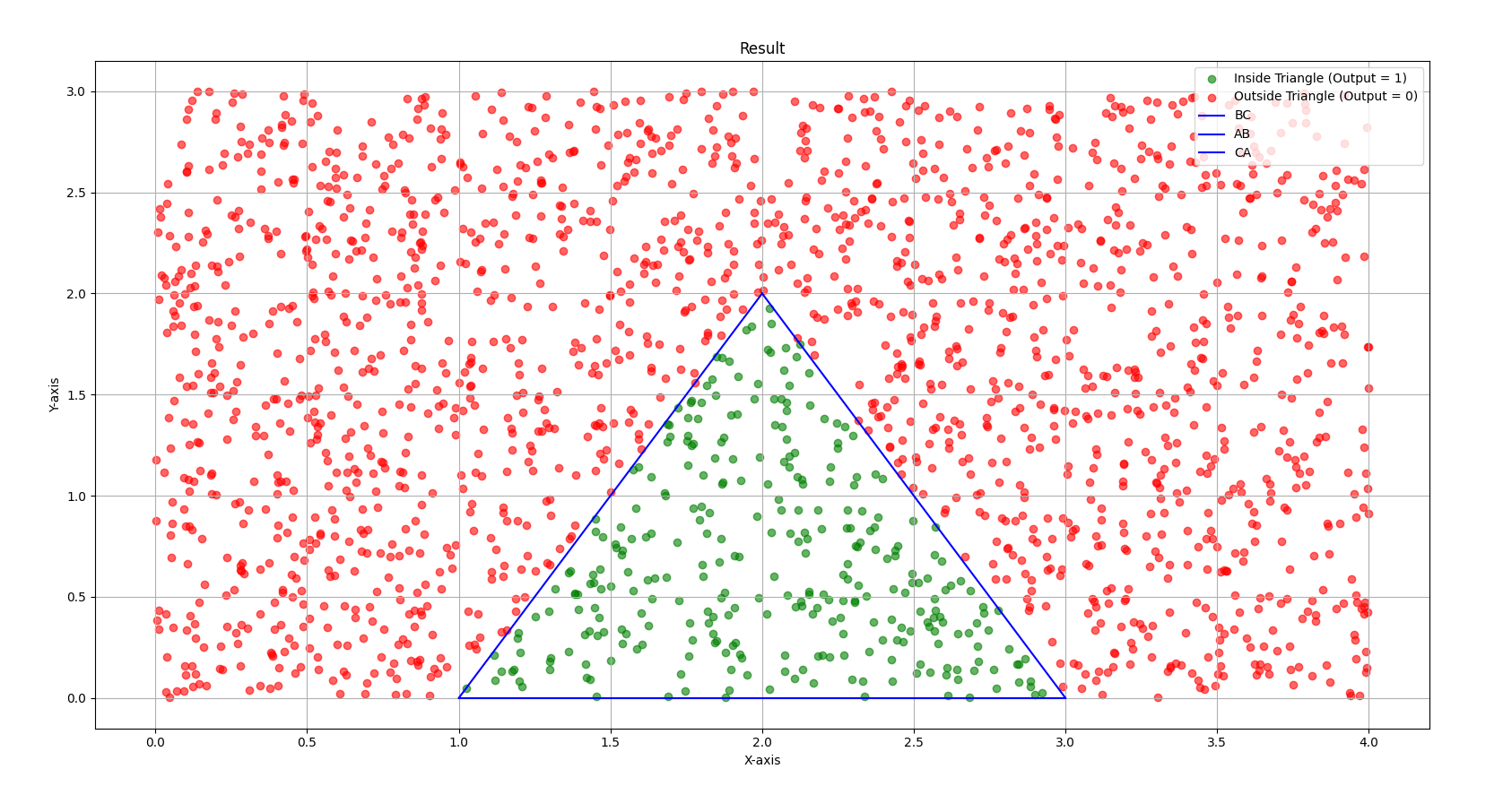
در ReLU، اگر ورودی x ≤ 0 باشد، گرادیان صفر است. این بدان معناست که نورون‌هایی که مقدار ورودی منفی دریافت می‌کنند، به طور کامل غیرفعال می‌شوند و در فرآیند یادگیری شرکت نمی‌کنند. این پدیده که به "مشکل مرگ نورون‌ها" (Dying ReLU Problem) معروف است، می‌تواند باعث شود بخش قابل توجهی از نورون‌ها در طول آموزش بی‌اثر شوند.

اما برای ELU، برای مقادیر x ≤ 0 ، گرادیان ELU برابر αex است که هرگز صفر نمی‌شود.

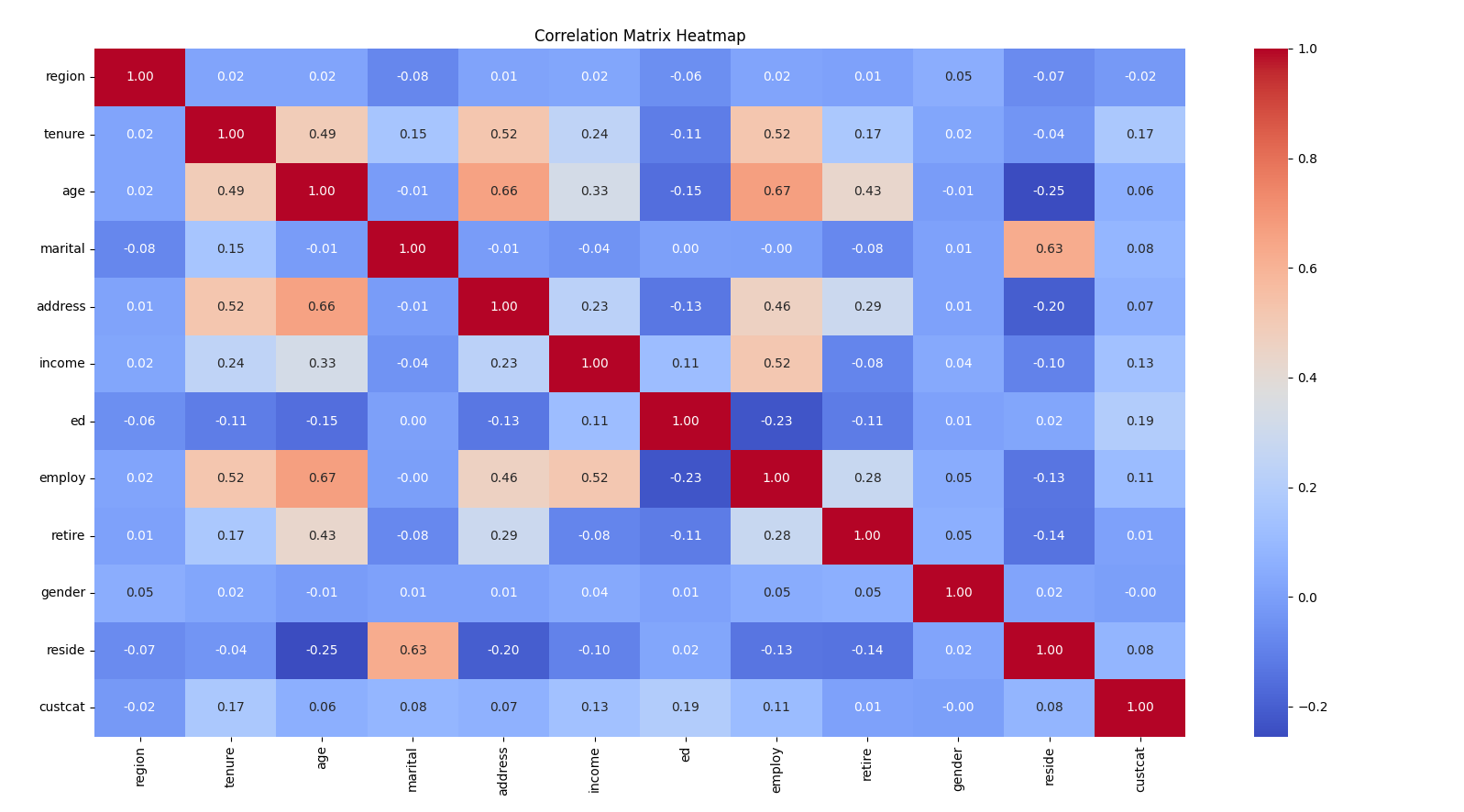
(مگر اینکه α = 0 باشد). این ویژگی تضمین می‌کند که نورون‌ها حتی در نواحی منفی همچنان گرادیان غیرصفر دارند و در فرآیند یادگیری مشارکت می‌کنند.

برای این مسئله، یک تایع تعریف میکنیم که سه شرط را بررسی میکنه که در واقع معدلات سه خط هستند .

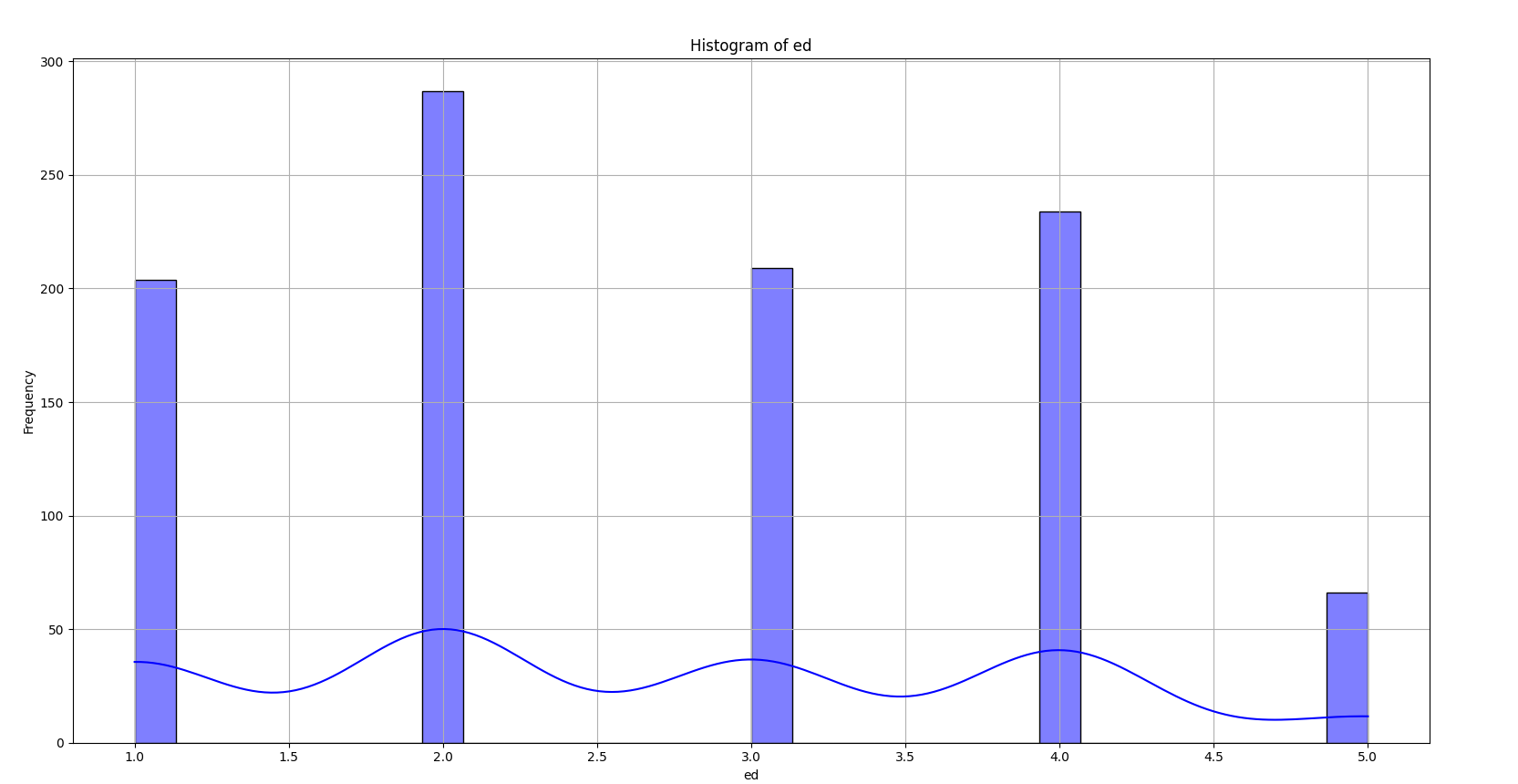
|  |
| --- |
| condition1 = y >= 0  # Above line BC      condition2 = y <= -2 \* x + 6  # Below line AB      condition3 = y <= 2 \* x - 2  # Below line CA |

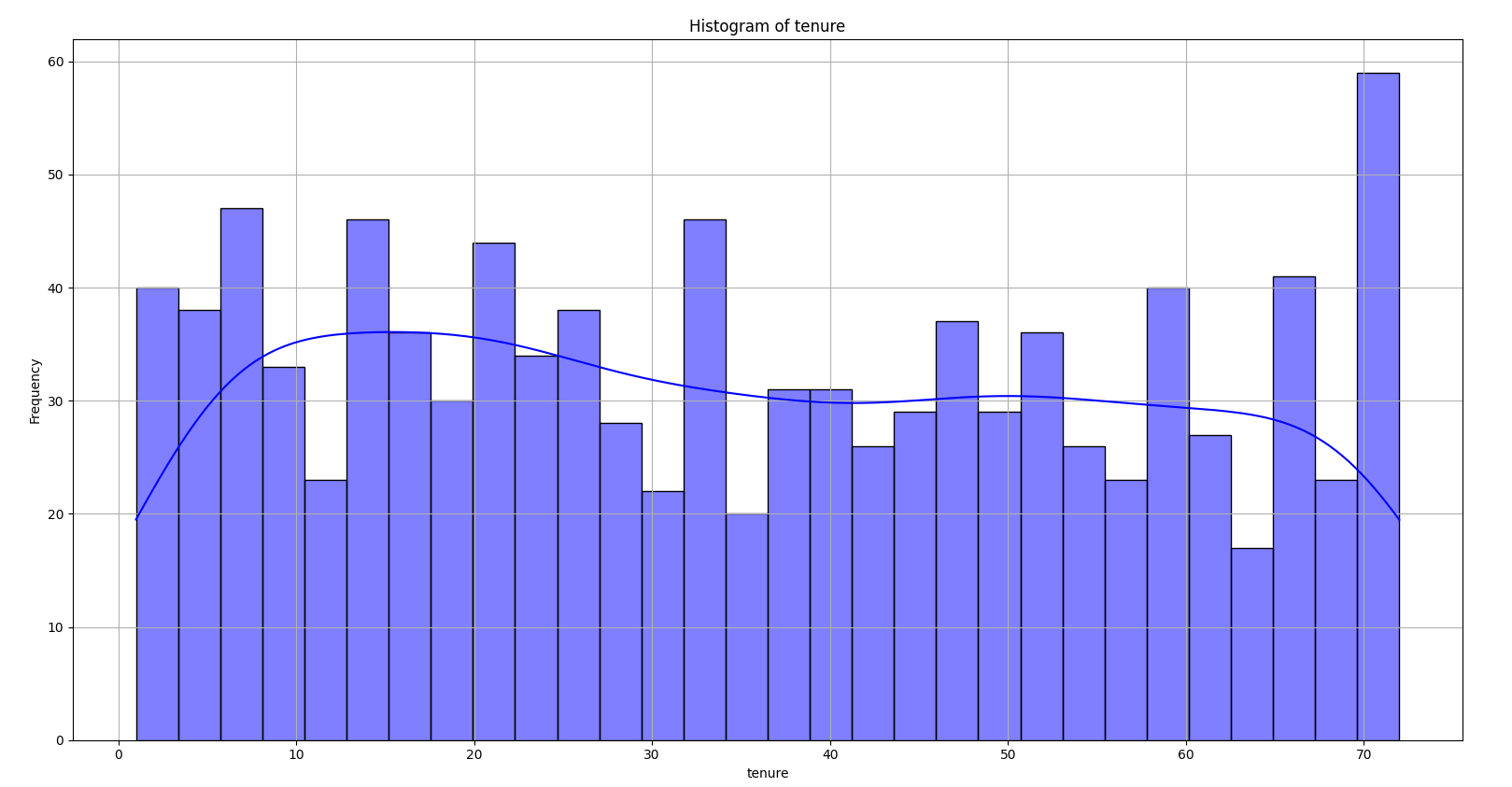


## سوال دوم



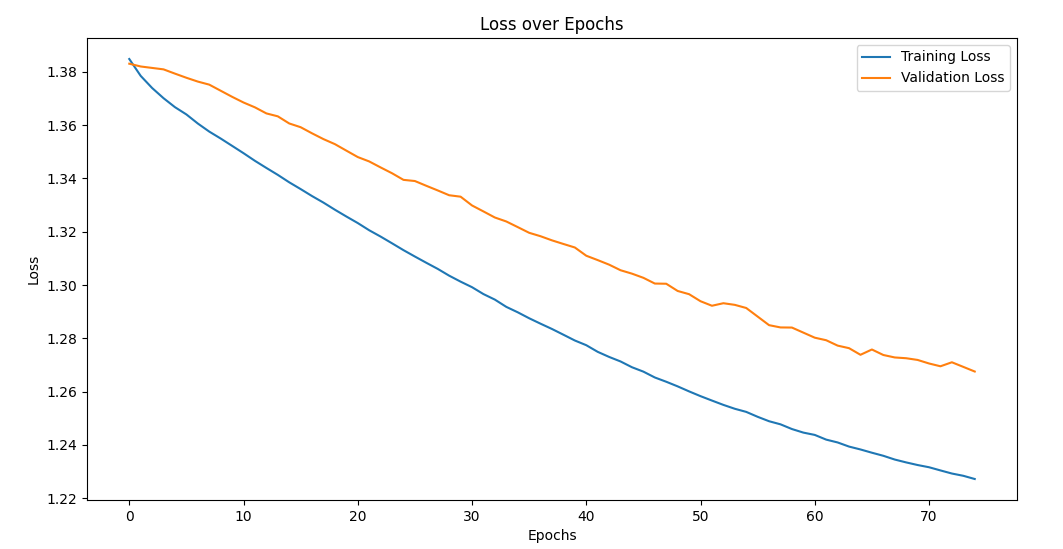
داده‌های موجود همبستگی زیادی با تارگت ما یعنی custcat ندارند و بیشترین همبستگی مربوط به داده ed است که برابر با ۰.۱۹ می‌باشد.





2.4)

**مدل اول با 80 نورون**

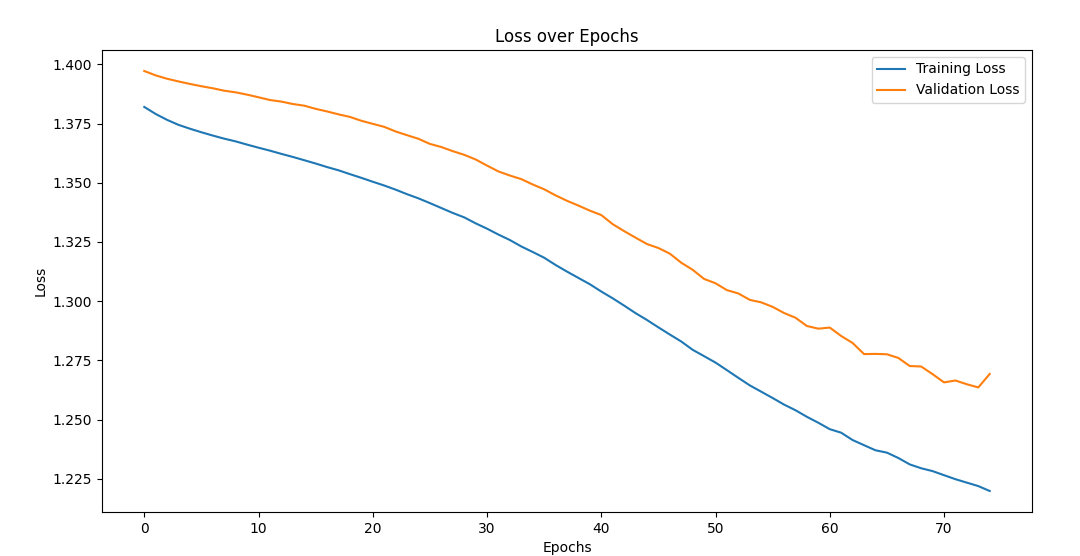


Test Accuracy: 0.38, Test Loss: 1.29

نتیجه به دست آمده(Test Accuracy) تقریبا برای تعداد نورون ها از 10 عدد تا 300 عدد یکسان بود تنها تفاوت اینها هزینه محاسباتی اضافه بود و مقدار Test Accuracy تقریبا بین 0.35 تا 0.41 بود.

یکی دیگر از تفاوت تعداد نورون های بالا با تعداد پایین داشت این بود که مقدار loss با سرعت بیشتری کاهش پیدا میکرد ولی به هر حالا در نتیحه تست تاثیری نداشت.

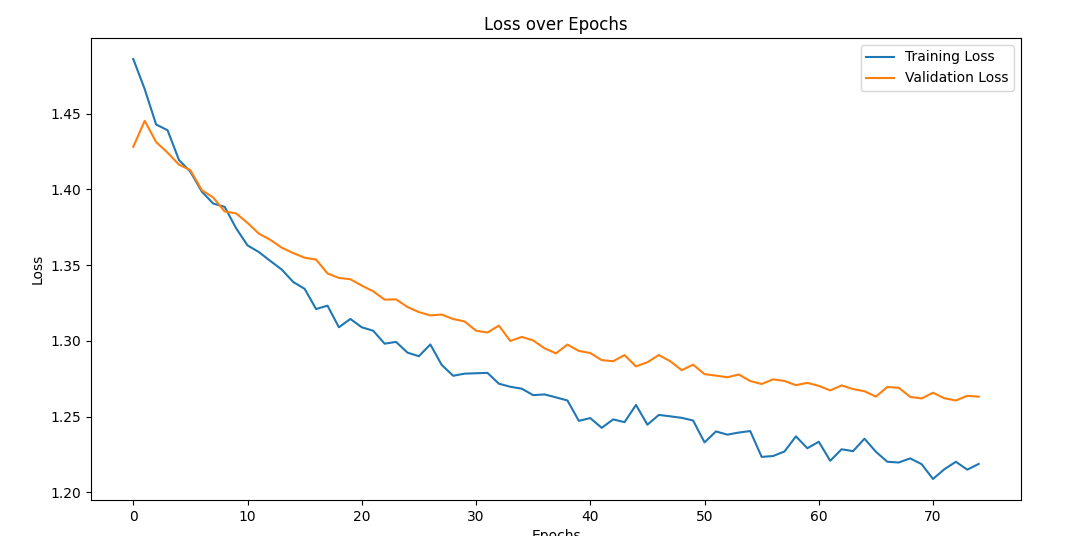
**مدل دوم با 100 نورون در لایه اول و 50 نورون در لایه دوم**



Test Accuracy: 0.38, Test Loss: 1.30

این تغییر در ساختار شبکه کمک خاصی به نتیجه نکرد و دقیقا مثل شبکه قبلی نتیجه ها حدود 0.35 تا 0.41 است.

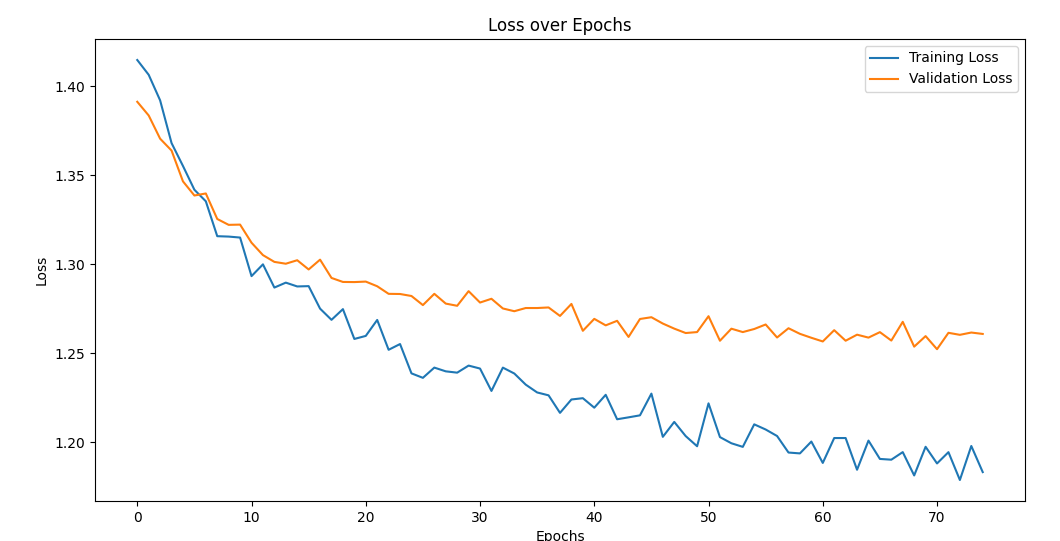
**مدل اول با 80 نورون با لایه نرمال سازی**



Test Accuracy: 0.39, Test Loss: 1.31

با توجه به نمودار سرعت یادگیری مدل افزایش پیدا کرده که از فواید استفاده از لایه نرمالسازی است اما این تکنینک اثر خاصی در نتیجه تست نگذاشته است.

**مدل دوم با 100 نورون در لایه اول و 50 نورون در لایه دوم با لایه نرمال سازی**

****

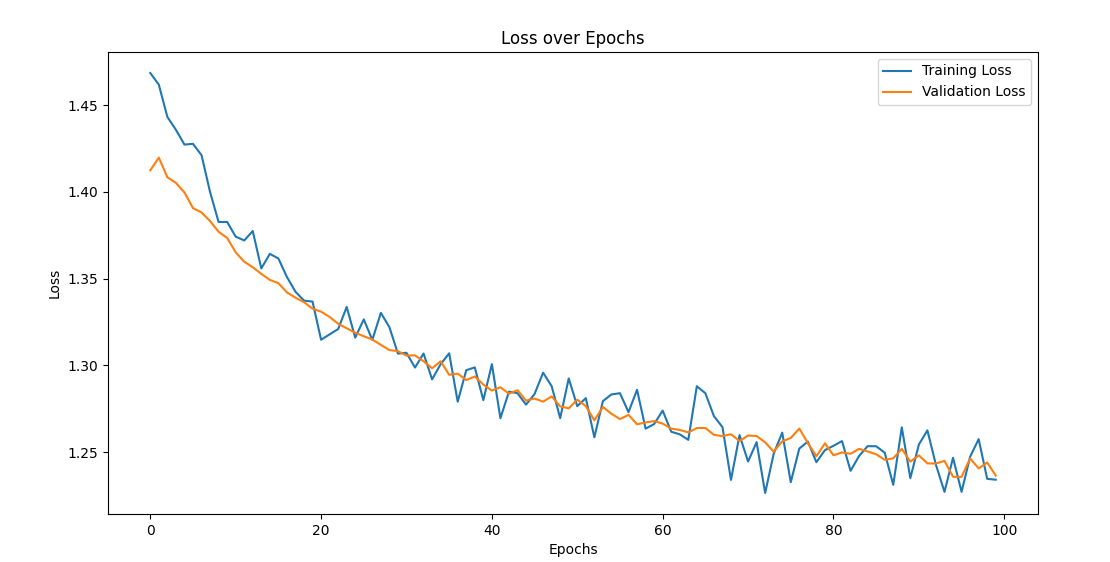
Test Accuracy: 0.38, Test Loss: 1.30

با توجه به نمودار سرعت یادگیری مدل افزایش پیدا کرده که از فواید استفاده از لایه نرمالسازی است اما این تکنینک اثر خاصی در نتیجه تست نگذاشته است.

اضافه کردن**Batch Normalization** اغلب ظرفیت مدل را برای یادگیری پیچیدگی‌های بیشتری از داده‌ها افزایش می‌دهد. این افزایش ظرفیت می‌تواند مدل را بیشتر در معرض Overfitting قرار دهد، به‌خصوص اگر از روش‌های منظم‌سازی (Regularization) مانند Dropout یا Weight Decay به‌اندازه کافی استفاده نکرده باشید.

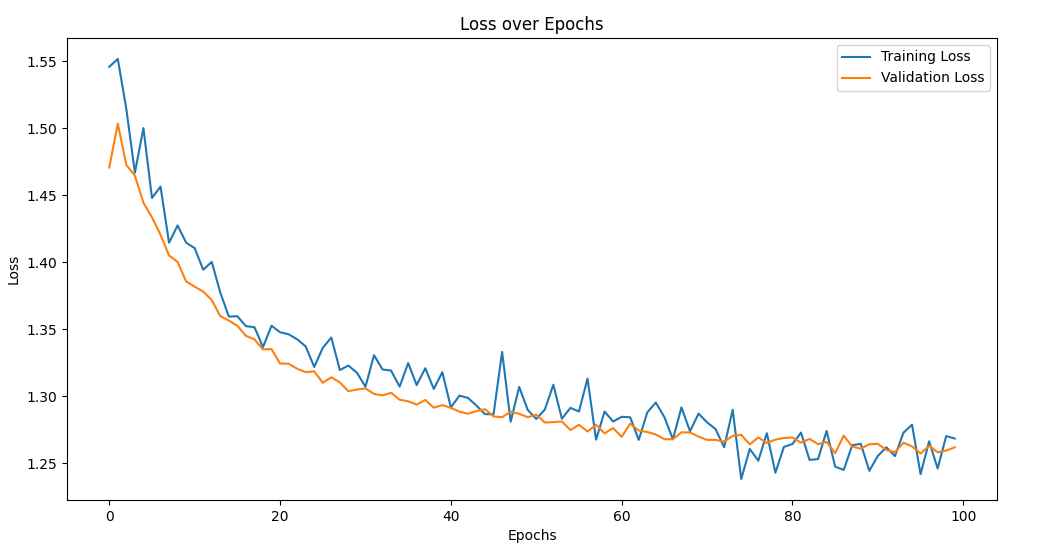
.

**مدل اول با 80 نورون با لایه نرمال سازی و dropout**



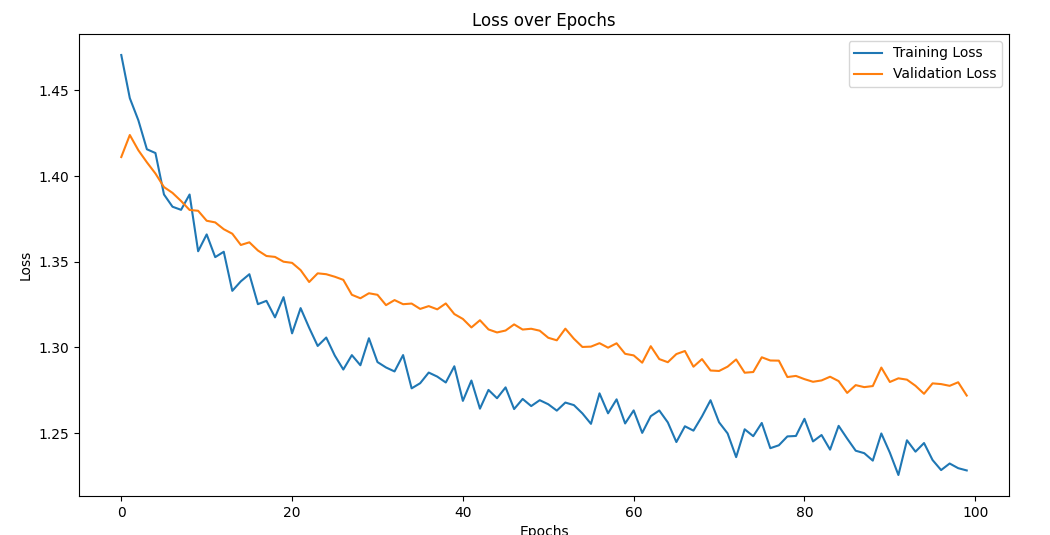
Test Accuracy: 0.39, Test Loss: 1.30

**مدل دوم با 100 نورون در لایه اول و 50 نورون در لایه دوم با لایه نرمال سازی و dropout**



Test Accuracy: 0.38, Test Loss: 1.31

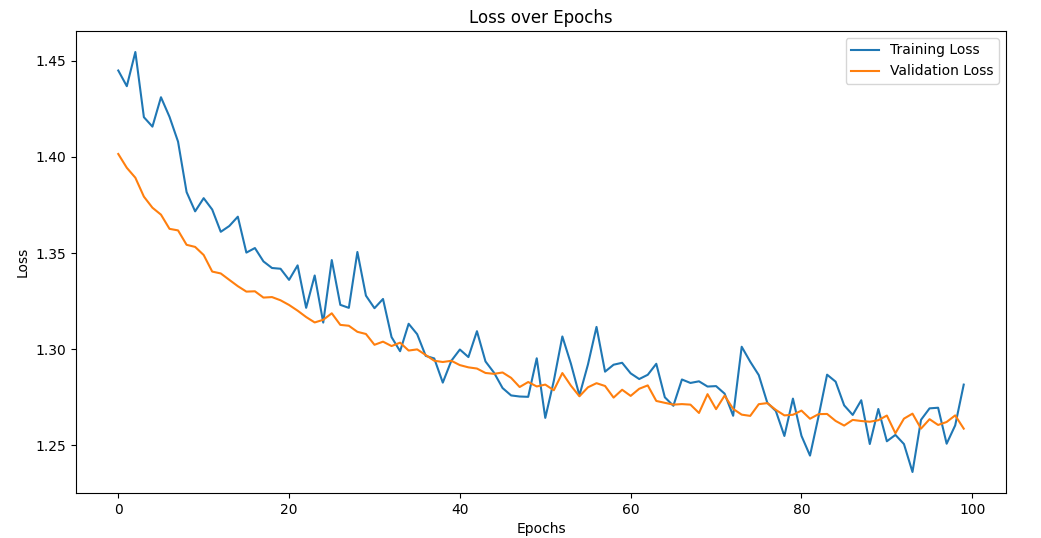
**مدل اول با 80 نورون با لایه نرمال سازی و dropout و L2** Regularization



Test Accuracy: 0.39, Test Loss: 1.30

**مدل دوم با 100 نورون در لایه اول و 50 نورون در لایه دوم با لایه نرمال سازی و dropout و**

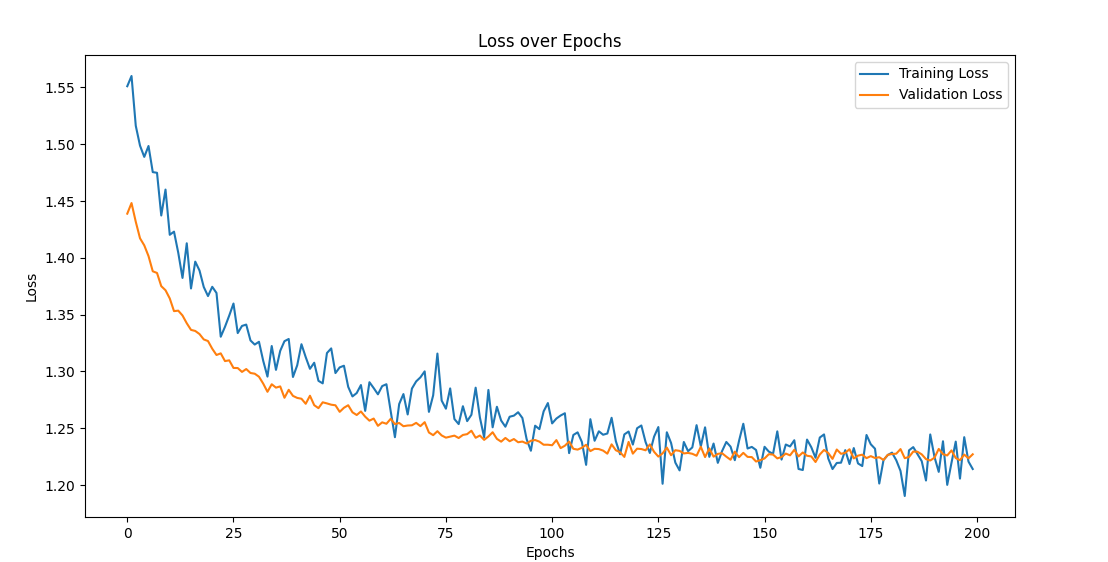
**L2** Regularization

****

Test Accuracy: 0.38, Test Loss: 1.31

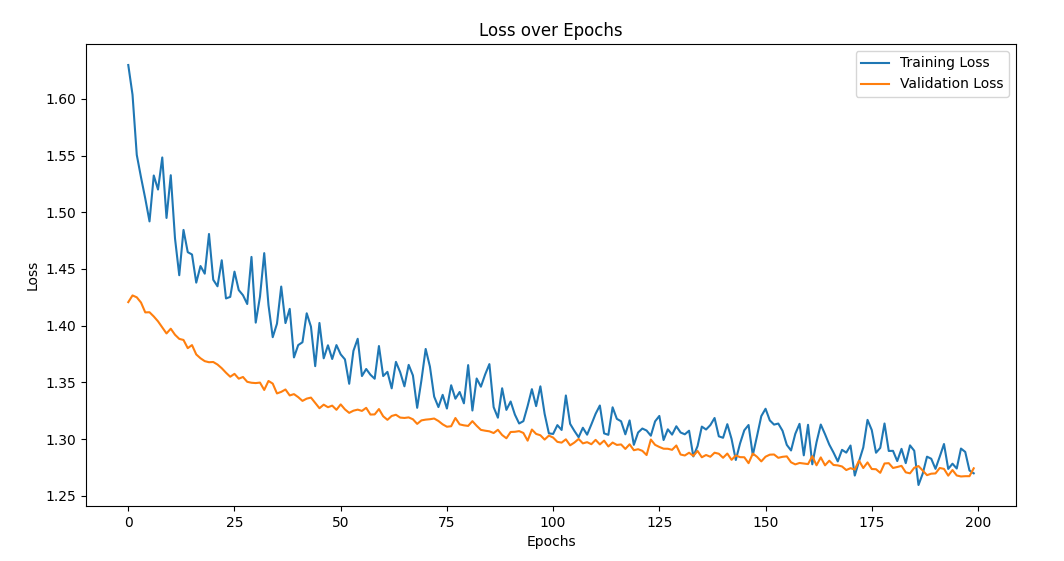
با تشکر.

**مدل اول با 80 نورون و بهینه سازی Adam با لایه نرمال سازی و dropout و L2** Regularization



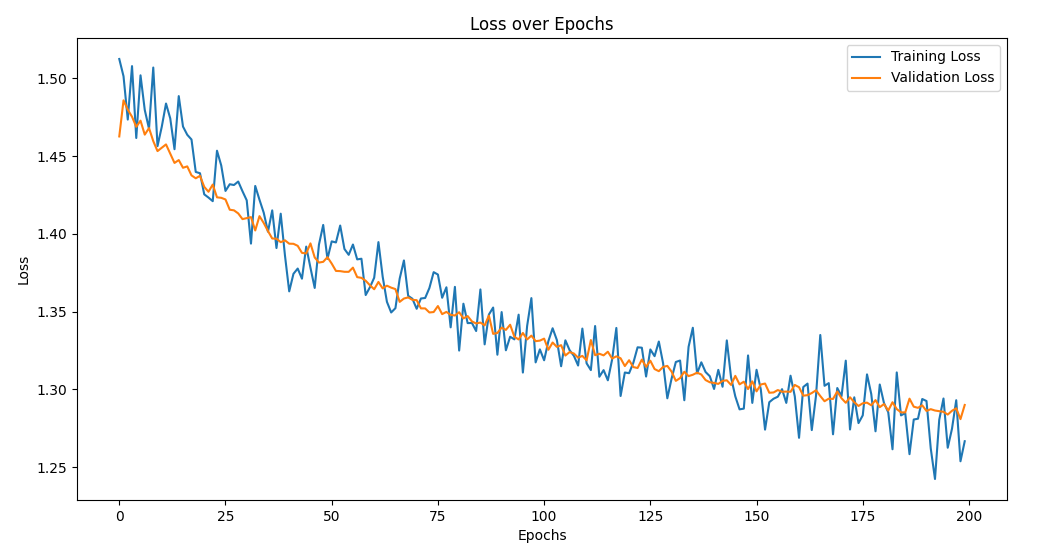
Test Accuracy: 0.38, Test Loss: 1.30

**مدل دوم با 100 نورون در لایه اول و 50 نورون در لایه دوم و بهینه ساز Adam با لایه نرمال سازی و dropout و L2** Regularization



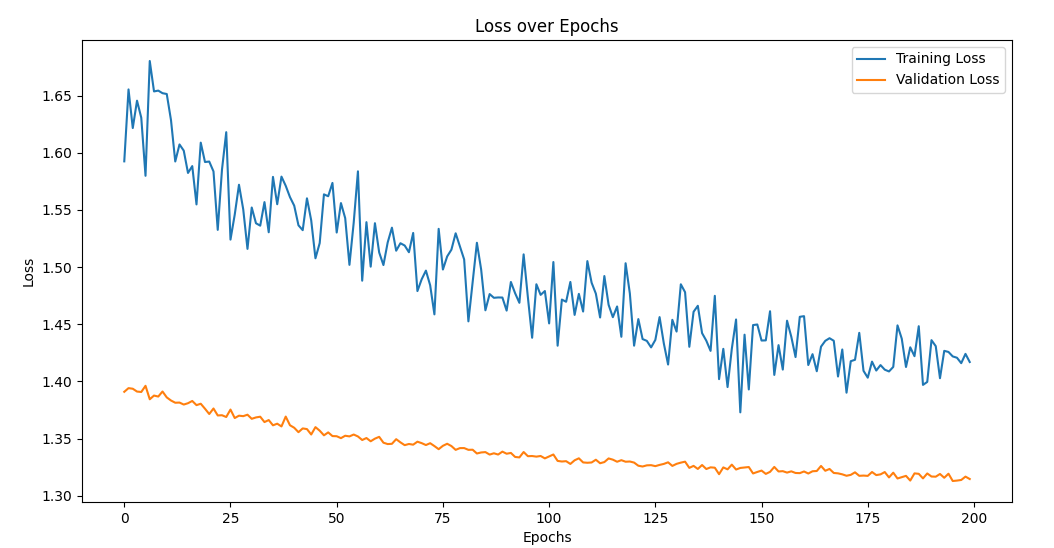
Test Accuracy: 0.34, Test Loss: 1.31

**مدل اول با 80 نورون و بهینه سازی Adadelta با لایه نرمال سازی و dropout و L2** Regularization



Test Accuracy: 0.35, Test Loss: 1.34

**مدل دوم با 100 نورون در لایه اول و 50 نورون در لایه دوم و بهینه ساز Adadelta با لایه نرمال سازی و dropout و L2** Regularization



Test Accuracy: 0.35, Test Loss: 1.33

**مدل اول با 80 نورون و نمرمالساز SGD با لایه نرمال سازی و dropout و L2** Regularization

|  |
| --- |
| Random Samples: Actual vs Predicted  Sample 1: Actual: 2, Predicted: 2  Sample 2: Actual: 2, Predicted: 2  Sample 3: Actual: 3, Predicted: 3  Sample 4: Actual: 0, Predicted: 2  Sample 5: Actual: 0, Predicted: 0  Sample 6: Actual: 1, Predicted: 0  Sample 7: Actual: 2, Predicted: 2  Sample 8: Actual: 0, Predicted: 0  Sample 9: Actual: 1, Predicted: 0  Sample 10: Actual: 2, Predicted: 1 |

**مدل دوم با 100 نورون در لایه اول و 50 نورون در لایه دوم و نرمالساز SGD با لایه نرمال سازی و dropout و L2** Regularization

|  |
| --- |
| Random Samples: Actual vs Predicted  Sample 1: Actual: 1, Predicted: 3  Sample 2: Actual: 2, Predicted: 2  Sample 3: Actual: 3, Predicted: 3  Sample 4: Actual: 3, Predicted: 0  Sample 5: Actual: 1, Predicted: 0  Sample 6: Actual: 3, Predicted: 1  Sample 7: Actual: 1, Predicted: 0  Sample 8: Actual: 0, Predicted: 0  Sample 9: Actual: 2, Predicted: 2  Sample 10: Actual: 2, Predicted: 3 |

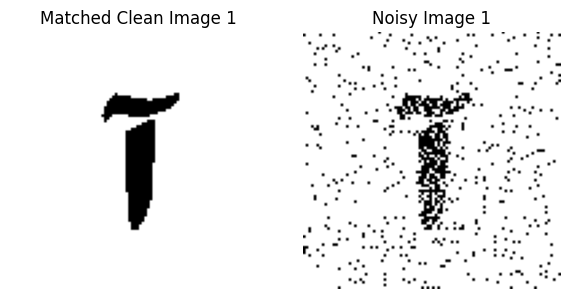
## سوال سوم

تابع اول ابتدا ابعاد تصویر ورودی را استخراج می‌کند و سپس یک مقدار مشخص (مانند یک عامل یا آستانه) برای تشخیص سفید یا سیاه بودن هر پیکسل تعریف می‌کند. سپس مقدار RGB هر پیکسل به‌صورت جداگانه بررسی شده و شدت کلی نور (total intensity) آن محاسبه می‌شود. بر اساس آستانه مشخص‌شده، تعیین می‌شود که پیکسل مورد نظر سفید یا سیاه است و نتیجه به تصویر باینری جدیدی با نام binary\_representation اضافه می‌شود.

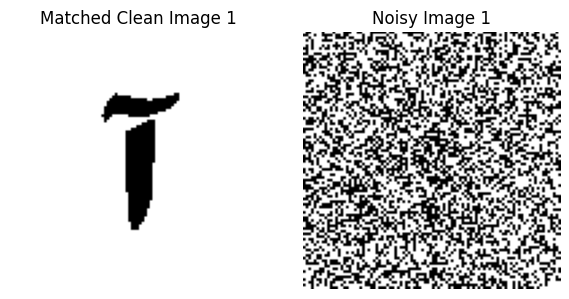
روش جایگزین دیگری برای این کار به این صورت است:  
ابتدا تصویر به مقیاس خاکستری (grayscale) تبدیل می‌شود و به جای مقادیر RGB، فقط یک مقدار شدت روشنایی برای هر پیکسل ذخیره می‌شود. سپس ماتریسی به ابعاد تصویر اصلی (به نام binary\_img )ایجاد می‌شود که در آن پیکسل‌هایی با شدت روشنایی بیشتر از ۷۵ به مقدار True و کمتر از آن به مقدار False تغییر داده می‌شوند. در نهایت، با استفاده از تابع astype(np.int8) این مقادیر به ۱ و ۰ تبدیل می‌شوند.

در تابع دوم برای افزودن نویز به تصویر، می‌توان شدت نویز را با استفاده از متغیری به نام noise\_factor تنظیم کرد. نویز به این صورت تولید می‌شود که برای هر پیکسل، عددی تصادفی در بازه‌ای بین -noise\_factor و +noise\_factor انتخاب شده و به مقدار RGB آن پیکسل اضافه می‌شود.

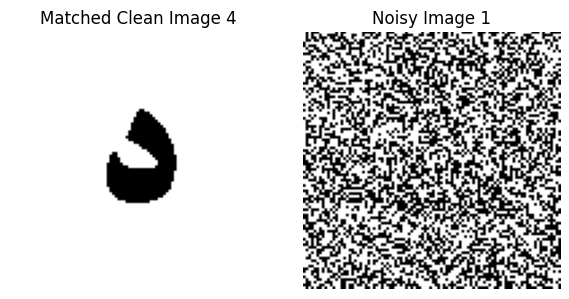
Noise factor = 200



Noise factor = 1000

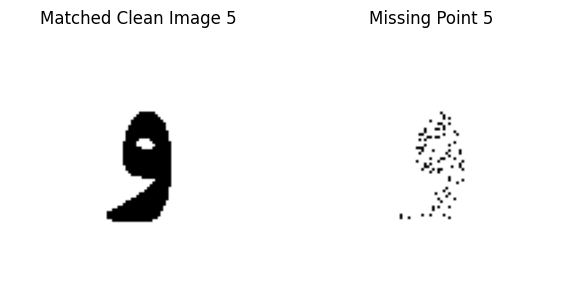


Noise factor = 2000



با نویز زیاد گاهی شبکه اشتباه حدس میزنه.

Noise factor = 500

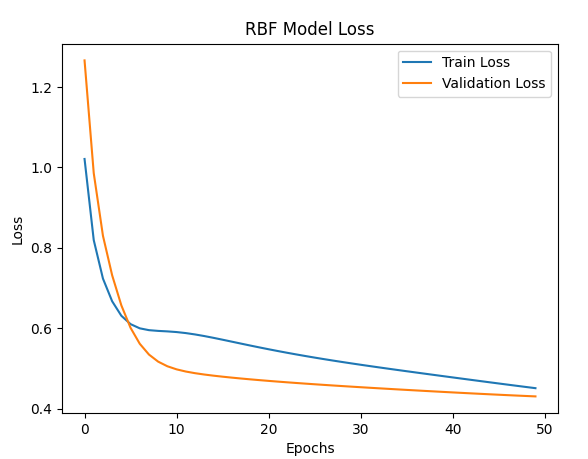


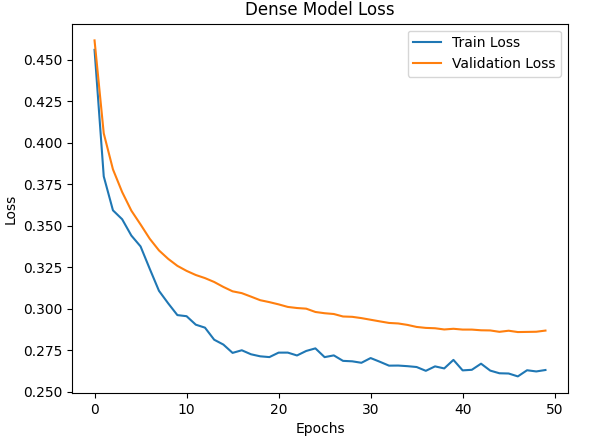
Noise factor = 1000



برای بهبود عملکرد شبکه در عکس‌های Missing Point، می‌توان از روش آگمنتیشن استفاده کنیم.

## سوال چهارم





مدل Dense در مقایسه با مدل RBF عملکرد بهتری داشته و این موضوع از مقدار کمتر loss در این مدل مشخص است. این تفاوت نشان می‌دهد که مدل Dense توانایی بیشتری در حل مسئله مورد نظر دارد.

مدل RBF معمولاً زمانی نتایج بهتری ارائه می‌دهد که داده‌ها دارای پراکندگی خوشه‌ای باشند. اما در این مسئله، روابط داده‌ها پیچیده و غیرخطی بوده و با هدف نهایی ارتباطات غیرساده‌ای دارند. به همین دلیل، مدل Dense با ساختار انعطاف‌پذیرتر و توانایی بهتر در یادگیری روابط پیچیده، انتخاب مناسب‌تری برای این مسئله است.