Reconhecimento de Caracteres com Melhoria da Tolerância a Ruídos

Lucas S Schiavini
lucaschiavini@hotmail.com

***Resumo*— O trabalho visa identificar parâmetros de uma rede neural e descrever como estes afetam o resultado esperado para caracteres gerados em uma matriz de 63 pixeis a partir da primeira letra do nome do estudante.**

I. Objetivos

O objetivo do projeto é estudar o funcionamento de uma rede neural de forma a manipulá-la e melhorar os resultados de predição de valores. Dessa forma, a rede será treinada com caracteres com e sem ruído e os parâmetros modificados devem melhorar os resultados dos treinamentos.

II. Introdução Teórica

Uma rede neural é uma construção matemática que permite a realização de aprendizado de resultados desejados sem a que o método de obtenção dos resultados seja um algoritmo criado por um ser humano.

Redes neurais nada mais são que uma sequência de operações realizadas em matrizes de dados de entrada. As operações são multiplicações e adições e então aplicações de funções não lineares, como a regressão logística.

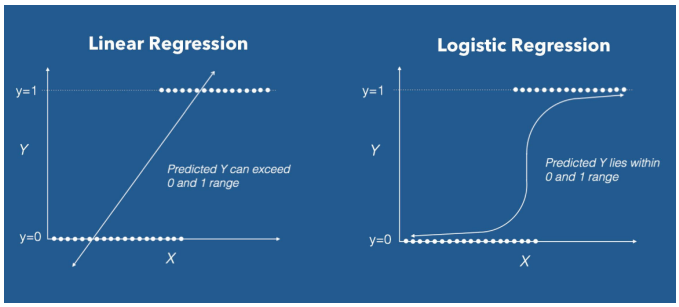

Fig. 1. Regressão logística

Com funções não lineares, é possível treinar camadas de neurônios e fazê-los se comportar de forma a aproximar funções, ou mesmo analisar dados.

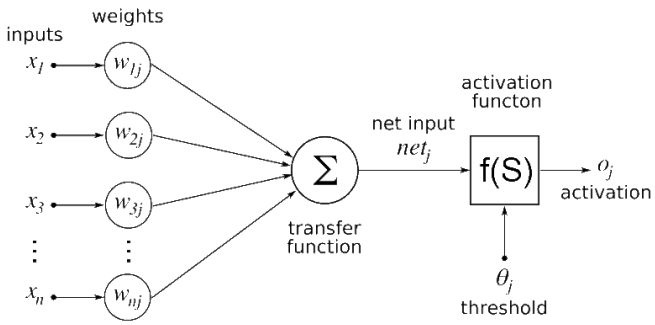

Fig. 2. Rede Neural

Na Figura 2 podemos ver as entradas de cada neurônio, e que o resultado da aplicação das funções não lineares dentro dos neurônios é somado ponderadamente e esse resultado passa por uma função de ativação. Se a função possuir um comportamento semelhante ao da Regressão Logística, é possível verificar que teremos valores estocásticos.

Dessa forma, os neurônios são colocados nessa rede, com mais camadas ou não, de forma a serem "treinados" por um conjunto significativo de dados. A seguir, essa rede com seus pesos ajustados, pode ser colocada a uma sequência de dados de entrada para ser validada. Com um número grande de testes e validações, é possível verificar se o erro entre o valor previsto e o valor dos dados válidos está diminuindo em relação com o tempo e com o número de dados de teste.

Ao fim do processo de teste e validação os dados devem ser suficientes para indicar se aquela rede está treinada o suficientemente bem para prever resultados fora daqueles que foram utilizados para seu treino e para sua validação.

III. Descrição Experimental

O experimento consistiu em adaptar parâmetros de uma rede neural implementada em python de forma a conseguir o mínimo de erro possível para cada tipo de valor de ruído, variando de 0 a 100%.

Primeiramente, criou-se uma sequência de caracteres em formato de uma matriz 7x9 com o primeiro caractere começando pela primeira letra do nome do estudante, nesse caso a letra L.

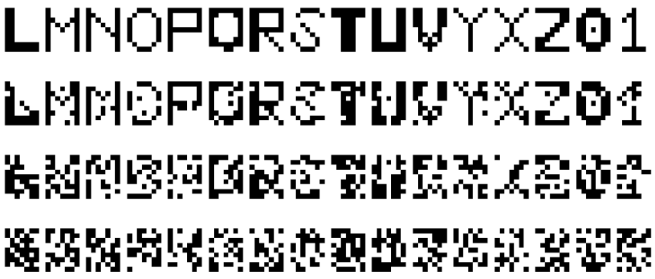

Fig. 3. Caracteres Gerados: 0% de ruído, 10%, 30% e 60%

Depois, utilizou-se o jupyter notebook para a montagem do código e as alterações nos parâmetros. Os parâmetros iniciais foram: N = 10, Nnt = 100, Nnv = 100, Nstep = 10, Neurônios primeira camada = 128, Neurônios segunda camada = 16.

Os parâmetros alterados foram:

- Nnt: Número de amostras de ruído para cada nível de ruído(0 a 100%).

- Número de neurônios na primeira camada da rede.
- Nstep: Passo de ruído.

IV. Resultados

Os resultados foram compilados em gráficos, com a linha verde sendo o treinamento sem ruído e a linha azul o treinamento com ruído. O resultado esperado é que a linha verde sempre fique acima da linha azul, visto que a rede neural não treinou com mudanças nos caracteres, logo deve cometer mais erros com o aumento do ruído.

Assim, temos os seguintes modelos:

A. Modelo inicial

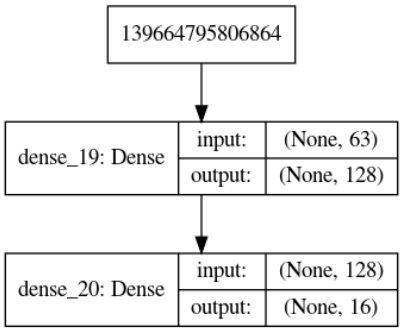

Fig. 4. Modelo Inicial

Dados originais do modelo, cada parâmetro foi modificado de forma a ver o comportamento da rede.

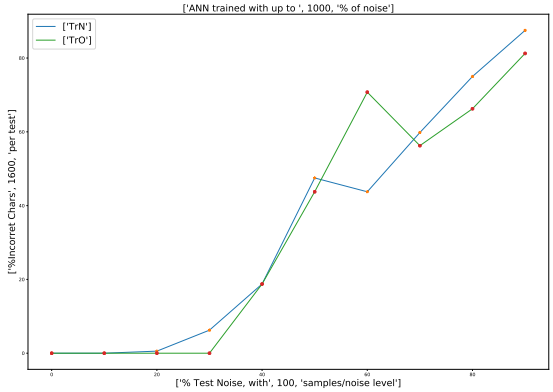

Fig. 5. Erro x Ruído

B. Amostras para treinamento com ruído

Modificando o número de amostras para treinamento com ruído (aumentando de 100 para 150) tem-se:

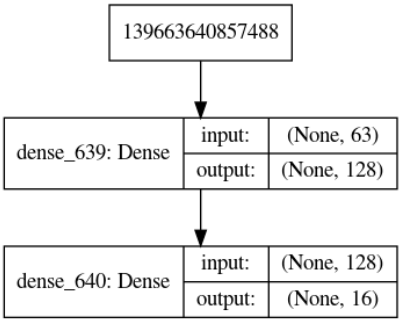

Fig. 6. Modelo Inicial

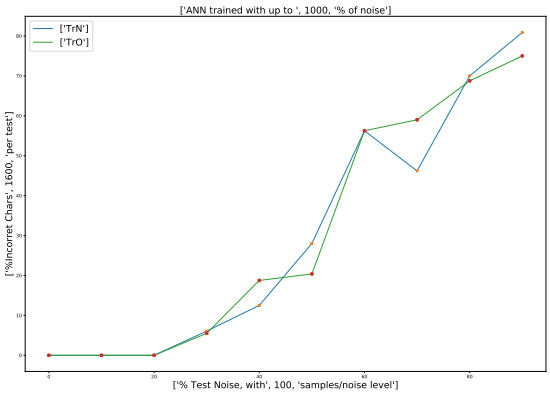

Fig. 7. Erro x Ruído, mudança em Noise Sample = 150

É possível ver na figura 7 que a curva azul possui valores menores de erro em cada passo da porcentagem do ruído.

C. Passo de Ruído

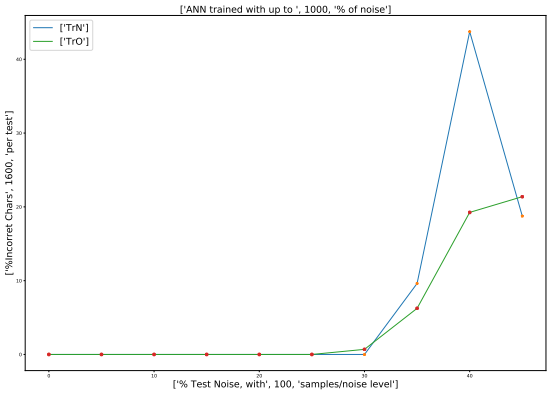

Fig. 8. Erro x Ruído, Valor do Noise Step = 5

Vemos na figura 8 que aumentar o Noise step causa um pico na curva do modelo treinado com ruído já em 30% de ruído. Como o Noise Step só modifica a amostragem de

dados no final, é possível que a função transferência das camadas dos neurônios apenas tenha obtido outro "mínimo" global dado os parâmetros inicializados.

D. Número de Dados ruidosos usados para Validação

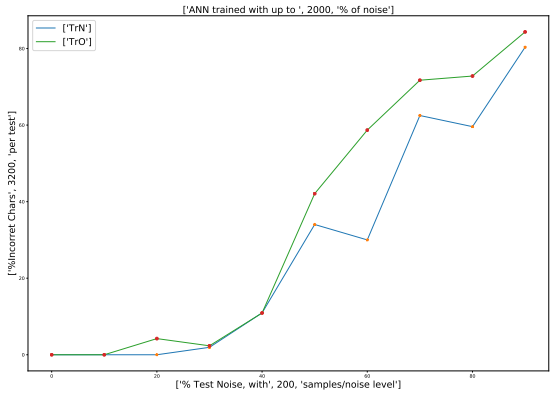

Fig. 9. Erro x Ruído, Numero do Noise Sample Validation = 200

Com a figura 9 temos que com um número maior de amostras para validação, a curva azul começa a indicar maior grau de erros. Isso mostra que um valor grande o suficiente de amostras pra validação aumentando o erro da rede neural significa que a rede não foi bem treinada. A rede pode ter passado pelo processo de 'overfitting' para cada dado, e com valores maiores de ruído, ela não é capaz de gerar resultados satisfatórios.

E. Número de Neurônios primeira Camada

Modificando a quantidade de neurônios na primeira camada de 128 para 160 tem-se:

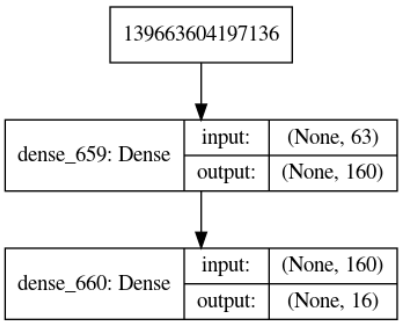

Fig. 10. 160 neurônios na primeira camada

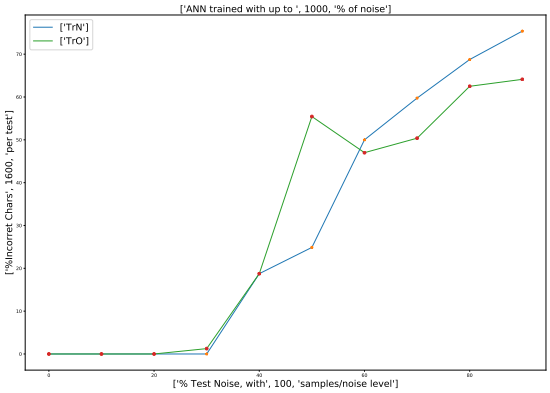

Fig. 11. Erro x Ruído, 160 neurônios

É visível na curva verde da figura 11 que o treino com ruído se tornou mais impreciso. Assim verifica-se que nem sempre aumentar a quantidade de neurônios é suficiente para aproximar do resultado esperado.

F. Utilização de RELU como função de ativação

Foi trocado de sigmoid para RELU a função de ativação dos neurônios da primeira camada.

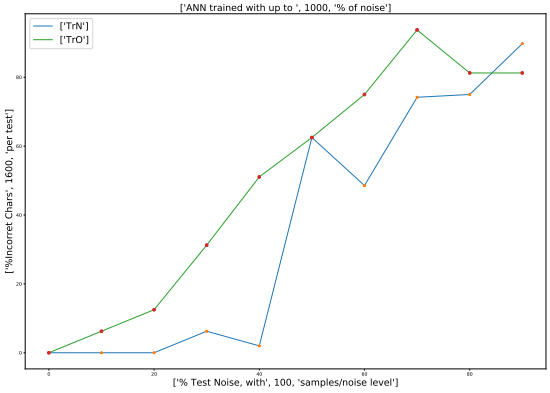

Fig. 12. Erro x Ruído, Função de ativação RELU

Pode ser visto uma aproximação melhor no gráfico, que indica que para essa atividade, a função de ativação relu tende a resultar em um treinamento mais eficiente.

1) Redução de neurônios: De forma a diminuir recursos computacionais, foi reduzido o número de neurônios para 10, 12 e finalmente 16.

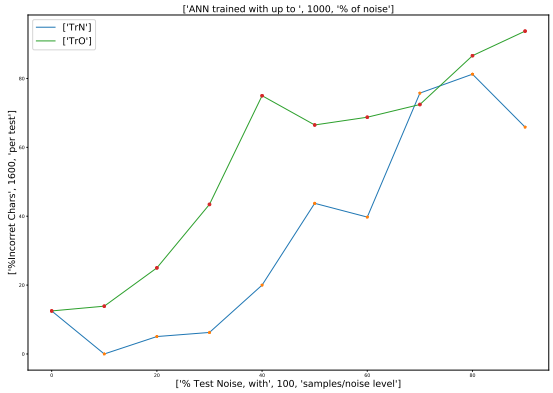

Fig. 13. Erro x Ruído, Função de ativação RELU com 10 Neurônios

Percebe-se maior índices de erro com pouco ruído, indicando que, analisando apenas o número de neurônios, esse número é insuficiente.

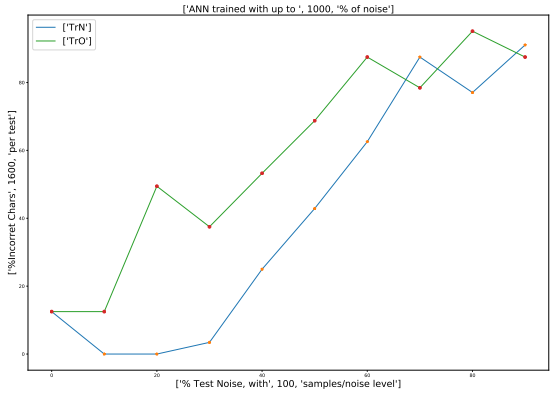

Fig. 14. Erro x Ruído, Função de ativação RELU com 12 Neurônios

Com o aumento do número de neurônios na primeira camada para 12, é possível ver que o gráfico possui menos erro com ruídos intermediários. No entanto, ainda há erro no início do gráfico para ruídos abaixo de 10%.

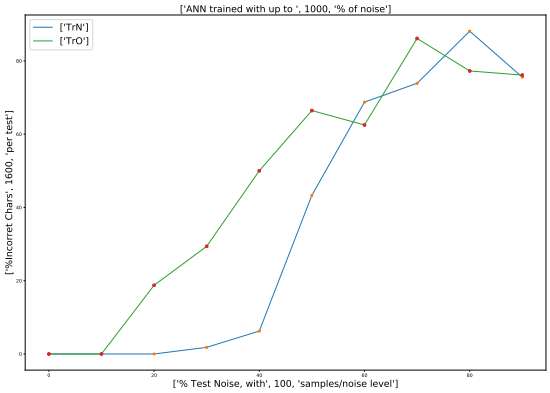

Fig. 15. Erro x Ruído, Função de ativação RELU com 16 Neurônios

Na figura 15, podemos ver que com 16 neurônios treinados com relu, temos um índice de erros muito baixos em até 40% de ruído.

O modelo final ficou:

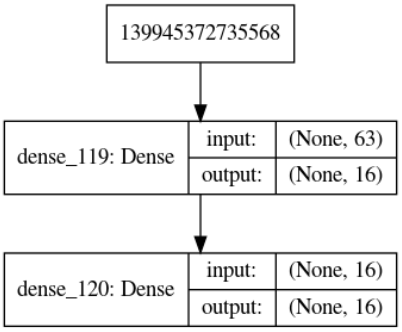

Fig. 16. Modelo de ativação RELU com 16 Neurônios

V. Discussão e Conclusões

Com essa atividade, foi possível entender melhor o funcionamento de uma rede neural, como funcionam os neurônios que nela estão localizados, a forma como pode ser feita sua parametrização e como separar uma faixa de dados para teste e validação.

References

Haykin, S.: Neural Networks and Learning Machines, 3rd Ed., Prentice Hall, 2009

Nick McClure, TensorFlow Machine Learning CookBook, Packt, 2017