Análise de desempenho de diferentes funções de ativação para redes neurais convolucionais classificadoras de imagens.

Lucas Hattori Costa

Bacharelado em Engenharia Mecânica

Escola Politécnica da Universidade de São Paulo (POLI-USP)

Bolsista SESu-MEC pelo Programa de Educação Tutorial (PET-Mecânica) - 2018

lucashattori@usp.br / (11) 96145-5238

1. Introdução

Inteligência artificial e aprendizado de máquinas são duas áreas de conhecimento que obtiveram grandes avanços nas últimas décadas, principalmente em questões de reconhecimento de padrões que se utilizam de Deep Learning. Prova disso são os trabalhos envolvendo redes neurais profundas (DNN, do inglês, *Deep Neural Networks*) em modelos acústicos [4] , em reconhecimento de imagens [1] ou processamento de linguagem natural [3]. Em cada um desses casos, o algoritmo criado se baseia na escolha de um conjunto de parâmetros, como o número de épocas de treinamento, o número de camadas, o tamanho do *kernel* de cada camada convolucional, e, por fim, a função de ativação utilizada.

2. Objetivos

O uso de redes neurais convolucionais (CNN) aplicadas a reconhecimento de padrões e visão computacional está sendo consolidado e aprofundado intensamente. Visando contornar problemas como tempo gasto, poder computacional envolvido e outros, foram desenvolvidos estudos acerca das características de diferentes funções de ativação quando aplicadas em diferentes tipos de dados. Este estudo visa analisar o desempenho, características e diferenças de funções de ativações quando aplicadas à uma CNN classificadora de um banco de dados composto por imagens de letras representadas em LIBRAS. As funções de ativação abordadas no projeto foram ReLU (Rectified Linear Unit) e sua versão parametrizada, sigmoide ou logística, a ELU (Exponential Linear Unit) e a função exponencial normalizada, denominada *softmax*.Tais funções estão definidas respectivamente em (1), (2), (3), (4) e (6).

$$f(x) = \begin{cases} 0, \text{ se } & x \leq 0, \\ x, \text{ se } & x > 0. \end{cases} \tag{1}$$

$$f(x) = \begin{cases} \alpha x, \text{ se } & x \leq 0 com 0 < \alpha << 1, \\ x, & \text{se } \quad x > 0. \end{cases} \tag{2}$$

$$f(x) = \frac{1}{1 + exp(-\lambda x)} \tag{3}$$

$$f(x) = \begin{cases} \alpha(exp(x) - 1), \text{ se } & x < 0, \\ x, \text{ se } & x \geq 0. \end{cases} \tag{4}$$

Para se entender a função *softmax*, é necessário defini-la como uma uma função que recebe um vetor z de dimensão K e retorna um vetor σ (z) com K entradas dadas por 0 ou 1 e dado que a somatória dessas entradas somam 1, de forma que σ (z) possui dimensão vetorial (K-1), como escrito em (5).

$$\sigma : \mathbb{R}^K \rightarrow \sigma \in \mathbb{R}^K | \sigma_i > 0, \sum \sigma_i = 1 \tag{5}$$

$$\sigma(z)_j = \frac{exp(z_j)}{\sum exp(z_k)} for j = 1, 2, 3, ... \tag{6}$$

3. Metodologia

A comparação entre as funções se deu por meio de uma CNN criada a partir da linguagem Python por meio do framework *Keras* e usando back-end no software *TensorFlow*. A arquitetura da rede se manteve semelhante entre os testes. Foram utilizadas 3 camadas convolucionais e intercaladas por camadas de *pooling*. Cada neurônio de camada convolucional possui dois parâmetros (o peso *w* e o viés *b*) que é ajustado a cada época usando o algoritmo *AdaGrad* tal qual implementado no framework utilizado. Além disso, o erro de cada época é calculado pelo método *binary crossentropy* definido por (7), onde *y* a distribuição de probabilidade de classes real para x e $\hat{y}$ é o valor obtido, como definido em [2].

$$loss(x_i) = -\sum y_j * log(\hat{y}_j + \epsilon), onde 0 < \epsilon << 1 \tag{7}$$

O banco de dados utilizado é composto, inicialmente, por 9074 imagens coloridas de representação em LIBRAS. O processo de aumento de dados consistiu em transformar para escala de cinza, redimensionar, e rotacionar 180º horizontalmente.

4. Resultados e Discussão

O desempenho das funções se traduz em diversos aspectos da rede. Entre eles, um dos mais expressivos é a acurácia de validação, pois mostra a eficácia da rede em classificar imagens fora do seu banco de dados de treino. O desempenho das funções estudadas nesse aspecto estão representados na Figura 1.

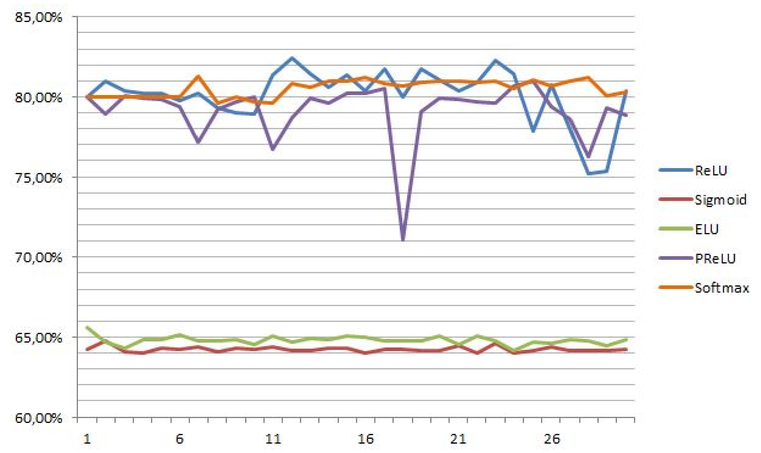

Figura 1: *Perfomance em acurácia de validação das funções ReLU, Sigmoid, ELU, PReLU e Softmax em função da época de treinamento.*

Outro resultado expressivo é o erro de validação, definida por (7), cujo desempenho das funções está representado na Figura 2.

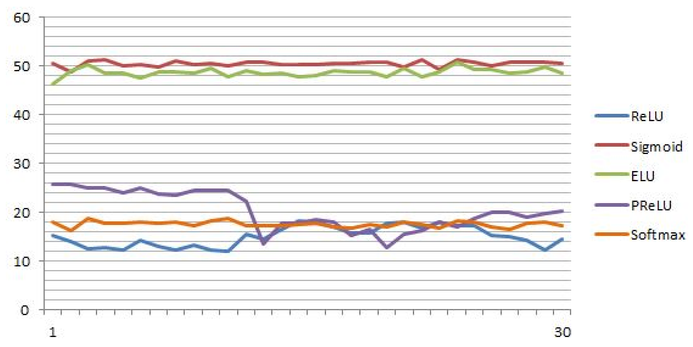

Figura 2: *Perfomance em perda de validação das funções ReLU, Sigmoid, ELU, PReLU e Softmax em função da época de treinamento.*

5. Conclusão

Em cada aspecto analisado, existe uma função ótima a ser utilizada. No quesito acurácia de validação e erro de validação, como mostrado, a com melhor desempenho é a *Softmax*, seguida pela ReLU e sua versão parametrizada. Entretanto, se outro quesito fosse priorizado, como o tempo de treinamento, por exemplo, os resultados poderiam ser outros.

Referências

[1] Kaiming H. e Xiangyu Z. e Shaoqing R. e Jian S. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*, 2015.

[2] Ponti M. e Costa G.B.P. Como funciona o Deep Learning. *Tópicos em Gerenciamento de Dados e Informações* , ICMC-USP, São Carlos-SP, Brasil, 2017.

[3] Zhang C. e Woodland P.C. Parameterised Sigmoid and ReLU Hidden Activation Functions for DNN Acoustic Modelling *INTERSPEECH 2015*,Dresden, Alemanha, 2015.

[4] Maas A. e Hannun A. e Ng A. Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proceedings of the 30 th International Conference on Machine Learning*, Atlanta, EUA, 2013.