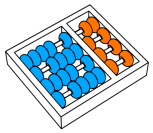
Universidade Estadual de Campinas
Instituto de Computação



MO434 - Deep Learning

Identificação: Christian Massao Konishi – RA 214570

Resumo

Inserir resumo

8 de maio de 2023

1 Exercício 1 – Exemplo de uma primeira rede neural para imagens

- Notebook: *first-image-neuralnet.ipynb*

1.1 Descrição

O notebook consiste na implementação simples de uma rede neural, seu treinamento e avaliação sobre a a base de dados Fashion-MNIST[1]. A base de dados consiste em 60.000 imagens em escala de cinza de 28x28 pixeis, que foram divididos em conjunto de treinamento, validação e teste, com tamanhos de 30.000, 15.000 e 15.000, respectivamente.

A rede neural proposta no enunciado em 3 camadas densamente conectadas©om inicialização de Xavier, com função de ativação ReLU e operação de Dropout com probabilidade de 25%. A função de custo utilizada é a entropia cruzada, com otimizador Adam e taxa de aprendizado 10^{-5}.

Depois de 40 épocas de treinamento, o gráfico de treinamento (Figura 1) indicou que houve um *underfitting*, caracterizado pelo custo de validação menor do que o de treinamento. A acurácia obtida no teste foi de 0,85.

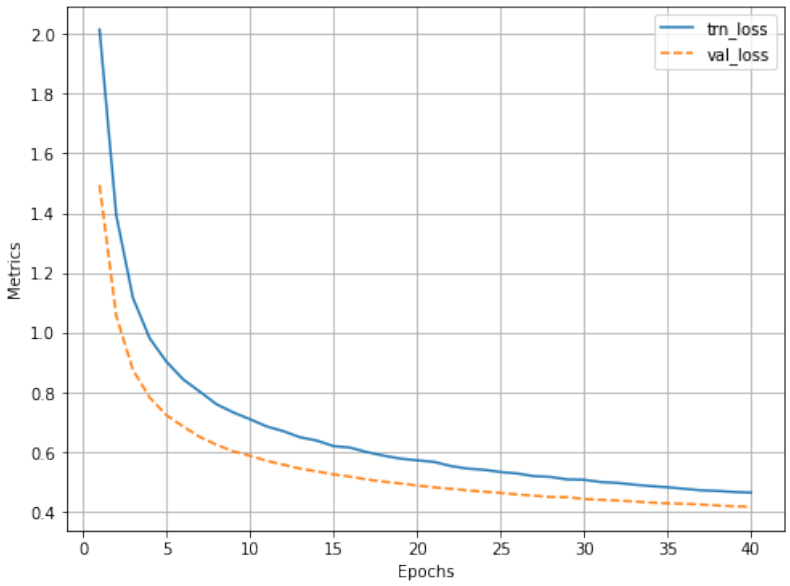


Figura 1: Gráfico do custo de treinamento e de validação ao longo das épocas de treinamento.

O exercício proposto envolve a modificação do modelo e dos processos de treinamento de forma a solucionar o problema encontrado, mas sem causar *overfitting*.

1.2 Solução do Exercício Proposto

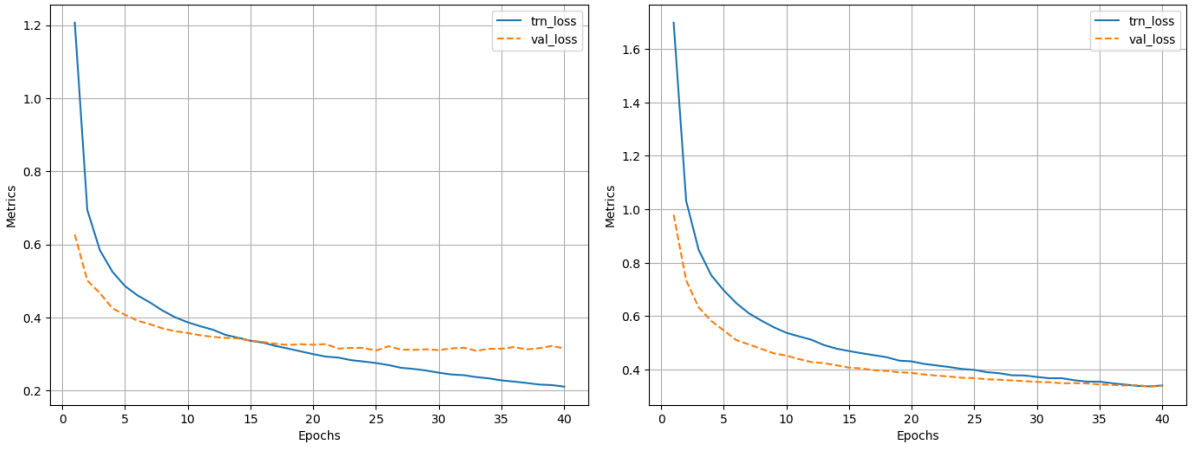
Diferentes abordagens para reduzir o fenômeno de *underfitting* podem ser aplicadas, entre elas o aumento da taxa de aprendizado ou o aumento de épocas, aumentar a complexidade da rede neural, aumentando a quantidade de camadas escondidas, por exemplo, entre outras possibilidades. Outra possibilidade seria a redução da probabilidade associada à operação de *dropout*, que poderia estar

exagerada, prejudicando a rede – apesar de que o valor de 0,25 não é considerado elevado, então é improvável que isso seja um problema.

A primeira alteração feita foi adicionar uma camada escondida extra, entre a segunda e terceira, com entrada de 200 *features* e saída de 100, fazendo-se as alterações necessárias na primeira camada densamente conectada para encaixá-la. A acurácia final não sofreu alteração significativa, nesse sentido, a mudança poderia ter sido descartada, mas para fins de estudo, ela foi mantida.

A segunda mudança foi aumentar a taxa de aprendizado, visto que observando a Figura 1, a rede ainda parecia estar convergindo ao fim do treinamento. Aumentar a taxa para 10^{-4} demonstrou-se exagerado, visto que houve um *overfitting* do modelo, caracterizado pelo função de custo de treinamento, que continuou a reduzir, mesmo com a estabilidade do custo de validação (Figura 2-a). É possível que com mais épocas, o custo de validação chegasse até mesmo a aumentar.

Voltando a taxa de aprendizado para 10^{-5} e aumentando-a em passos de 1.10^{-5}, em 4.10^{-5} o gráfico começa a apresentar um comportamento de *overfitting*, portanto 3.10^{-5} foi a taxa de aprendizado final utilizada (Figura 2-b).



(a) $lr = 4.10^{-5}$ (b) $lr = 3.10^{-5}$

Figura 2: Gráfico do custo de treinamento e de validação ao longo das épocas de treinamento para uma taxa de aprendizado de 4.10^{-5} e 3.10^{-5}.

Para a taxa de aprendizado final, a acurácia obtida na taxa de treinamento foi de 87,83%. De fato, o ganho de acurácia em relação ao treinamento inicial foi pequeno, mas a configuração de execução de algoritmo encontrada eliminou o *underfitting* sem causar *overfitting*.

2 Exercício 2 – Introdução ao Pytorch Convnet

- Notebook: *introducing-pytorch-convnet.ipynb*

2.1 Descrição

O notebook em questão contém uma série de demonstrações de conceitos básicos do Pytorch, como as operações de convolução, a função de ativação, *pooling*. Além de *skip connections*, camadas densamente conectadas, *dropout* etc. Por fim, há um processo de construção e treinamento de uma rede

neural, desde sua construção, definição de função de perda e otimizador, processo de treinamento e validação.

O problema do processo que foi realizado é que houve *overfitting* (Figura 3), noticiado pelo custo de treinamento consistentemente menor do que o de validação. O exercício em questão consiste numa exploração dos elementos e hiperparâmetros utilizados para melhorar o resultado obtido. A acurácia inicial é de 0,93.

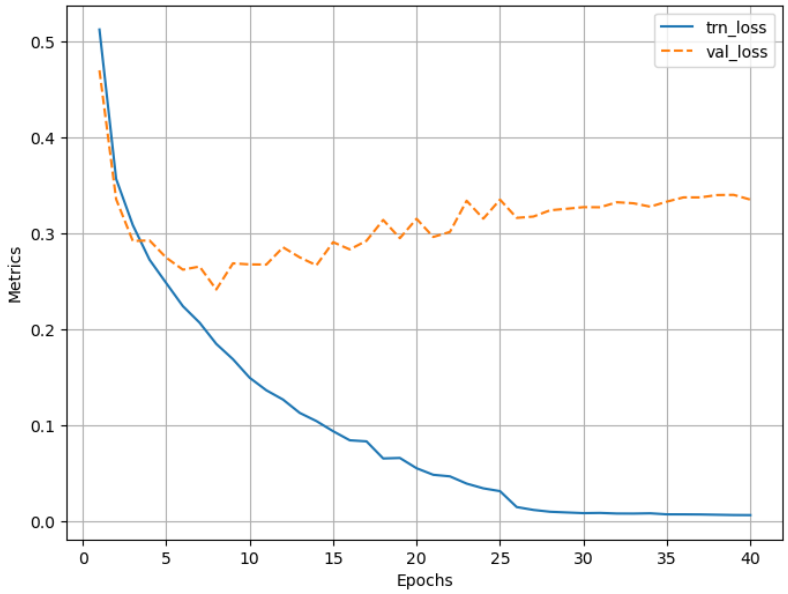


Figura 3: Gráfico do custo de treinamento e de validação ao longo das épocas de treinamento.

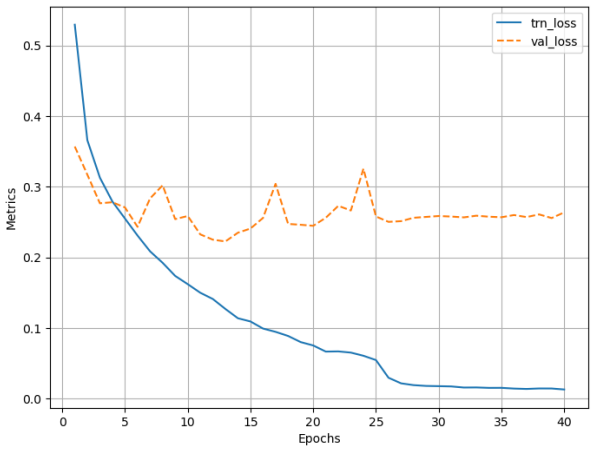
2.2 Resolução do Exercício Proposto

Como está ocorrendo um fenômeno de *overfitting*, a primeira alteração feita foi aumentar a regularização L2 o otimizador, alterando o valor de *weight_decay* para $0,001$ (antes era $0,0001$). Contudo o efeito não foi positivo, a acurácia foi de 0,928 e o *overfitting* permaneceu (Figura 4-a), portanto a alteração foi descartada.

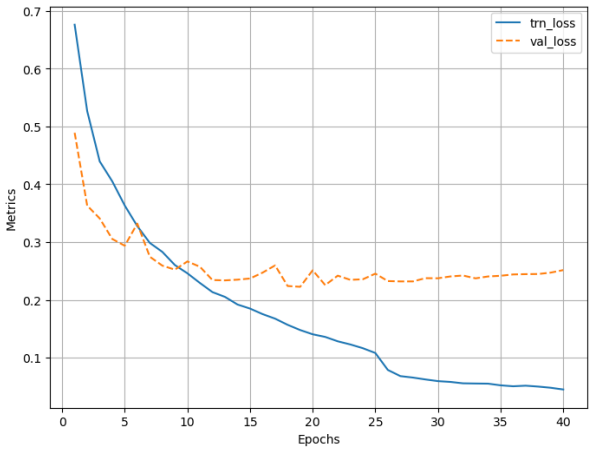
Em seguida, a probabilidade da operação de *dropout* foi aumentada, de 20% para 50% de chance de desconectar uma ligação da camada densa. A acurácia não apresentou melhoria, permaneceu em 0,93, mas o efeito de *overfitting* pareceu ser atenuado (Figura 4-b), mesmo que ainda não solucionado. De qualquer forma, a alteração foi mantida.

Por fim, a última modificação testada foi adicionar algumas operações de *data augmentation* no conjunto de treinamento. Para isso, foi feito uma classe de *dataset* personalizada, que funciona como *wrapper* da original, mas podendo aplicar *data augmentation* apenas no conjunto de treinamento.

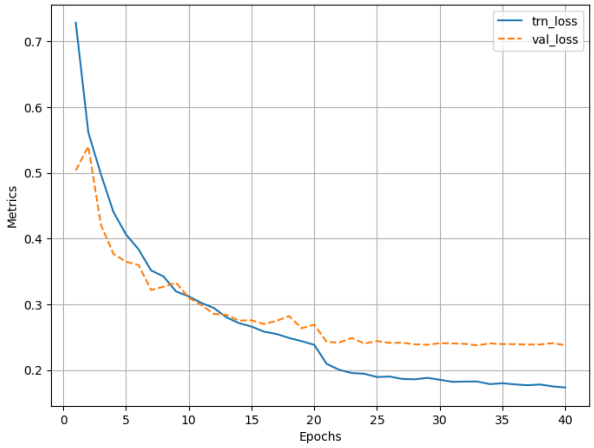
Foram adicionadas duas operações: (i) com uma chance de 50%, pode ocorrer uma flip horizontal da imagem; (ii) é aplicado um filtro gaussiano com tamanho de kernel 3 e sigma aleatório entre 0,1 e 2. Com essa abordagem, os efeitos do sobreajuste foram bastante minimizados (Figura 4-c), mas a eficácia obtida foide 0,92%.



(a) *weight_decay* $= 0,001$



(b) dropout $= 0,5$



(c) *Data Augmentation*

Figura 4: Gráfico do custo de treinamento e de validação ao longo das épocas de treinamento para diferentes modificações no modelo.

Referências

[1] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.