Introdução ao tópico de redes neurais residuais totalmente conectadas (FRN) e reduzidas (RRN) com teoria do caos

Apresentado por Gabriel Borin Macedo

Instituto de Matemática, Estatística e Computação Científica Unicamp (IMEEC)

15 de Janeiro de 2020

Resumo da apresentação

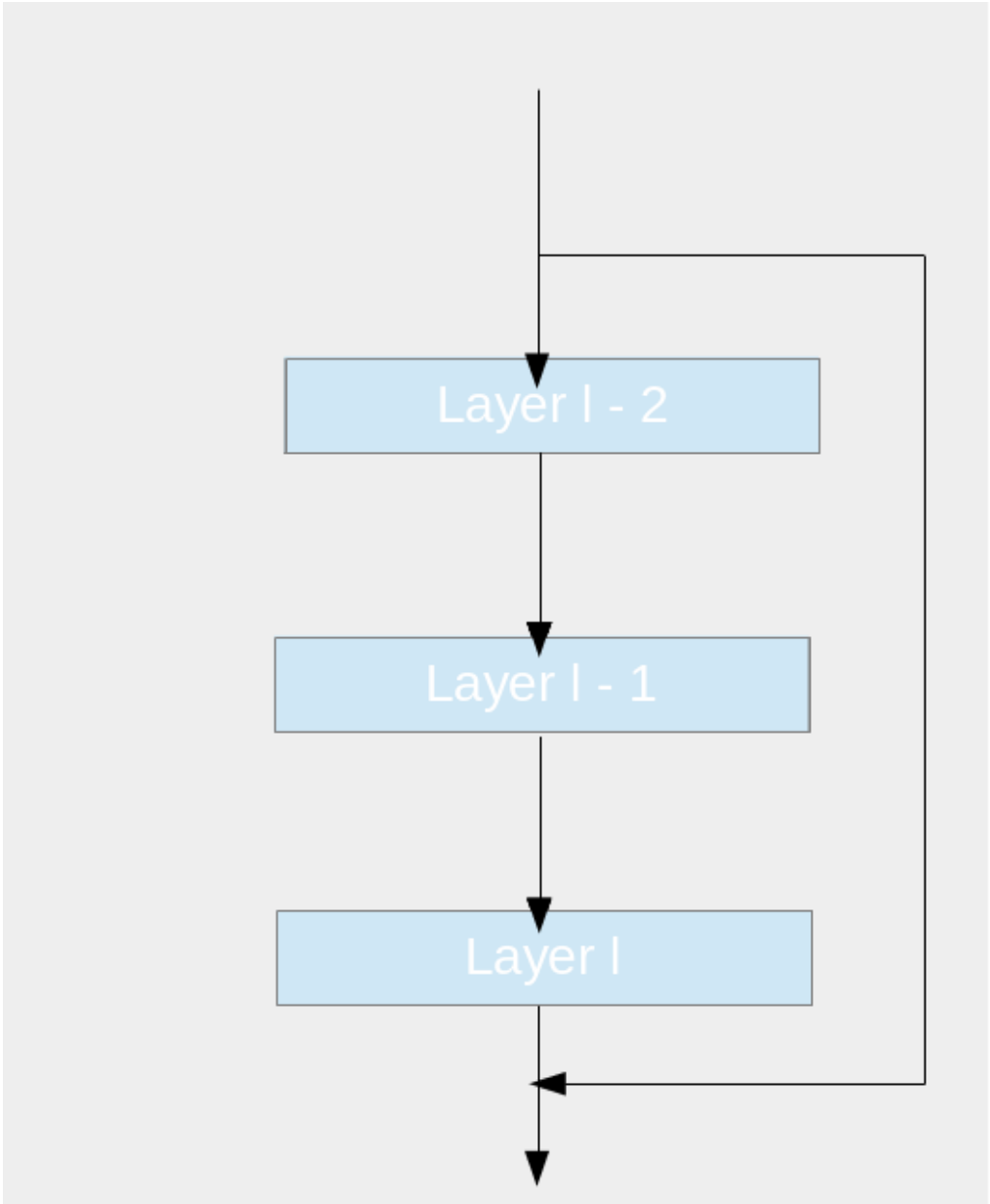
- Breve introdução ao modelo residual.
- Desvantagens do modelo FeedFoward.
- Motivação do modelo residual.
- Discução sobre estabilidade vs instabilidade.
- Definição dos modelos FRN e RRN.
- Descussão dos resultados do paper.
- Implementação dos modelos.
- Comparação com os resultados do paper.
- Conclusões obtidas do paper.

O que são redes neurais residuais ?

- São redes neurais que realizam “saltos” de alguns layers.
- Ou seja, nem todos os dados da rede são processados linearmente.
- Esses saltos são realizados criando novas conexões em algumas camadas da rede.
- È muito comum adicionar essas operações não lineares a cada duas ou três camadas.

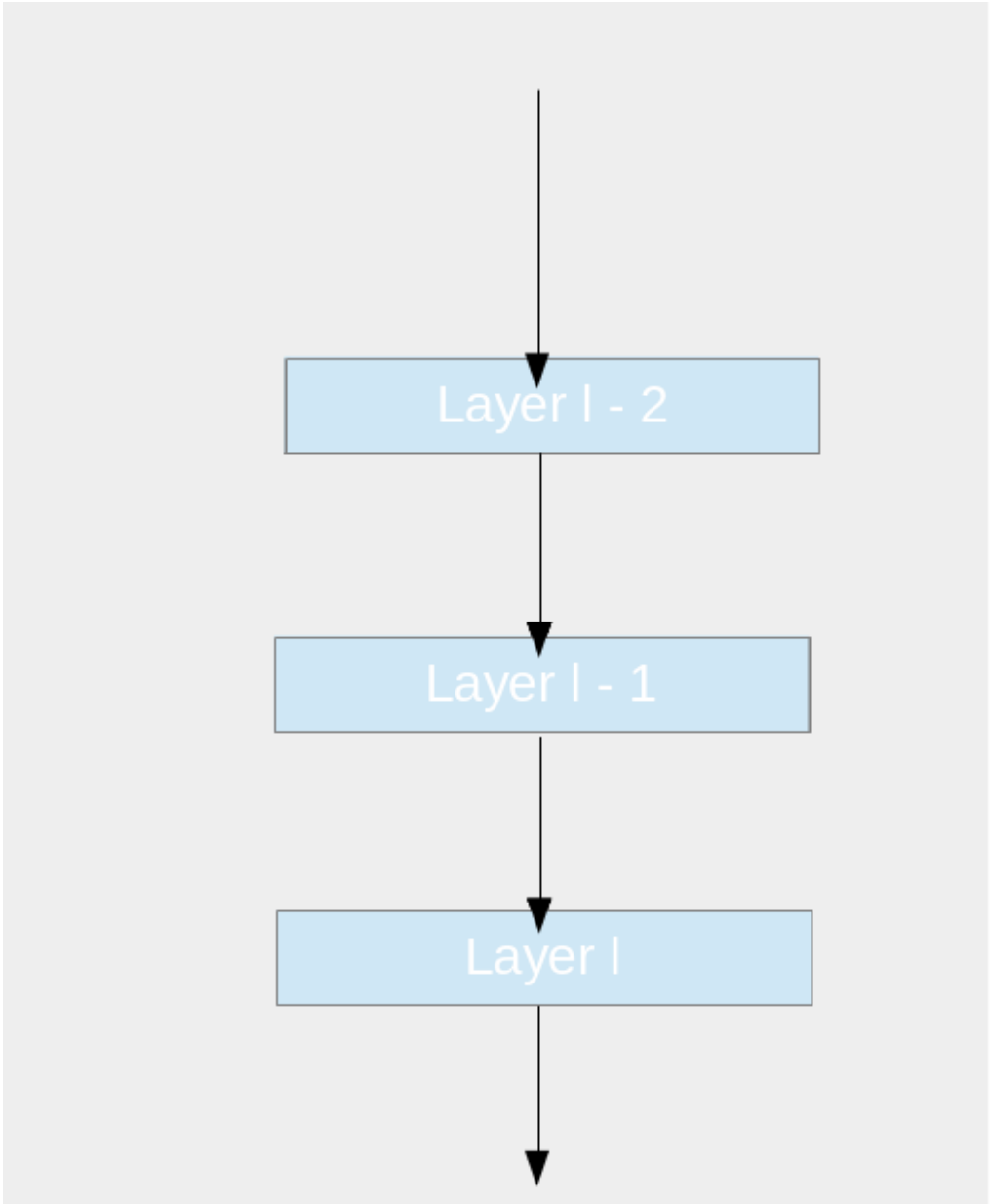
Modelos FeedFoward e residuais

Modelo residual



Modelos FeedFoward e residuais

Modelo FeedFoward



Desvantagens de um modelo FeedFoward

- Redes neurais clássicas que inicializam pesos randomicamente e usam, como exemplo, a função tanh como função de ativação, apresentam comportamento médio exponencial quando propaga seus dados utilizando feedfoward.
- Com isso, exponencialmente a rede perde a noção geométrica da imagem original.
- Além disso, nesse modelo mais clássico, exponencialmente pode ocorrer um drástico desaparecimento ou explosão do gradiente.

Desvantagens de um modelo FeedFoward

- Ou seja, para o caso da explosão do gradiente, exponencialmente são acumulado grandes erros no gradiente e ocorre updates mais longos nos pesos da rede durante a fase do treinamento.
- Com isso, deixa o modelo mais instável e dificulta que ele aprenda a partir do database utilizado.
- Já para o caso do desaparecimento do gradiente, exponencialmente o gradiente se aproxima do valor de zero.
- Consequentemente, deixa a rede mais difícil para treinar e aprender novas “features”.

Motivação do uso de redes neurais residuais

- No paper "Mean Field Residual Networks: On the Edge of Chaos", foi provado teoricamente e empiricamente que adicionando esses "saltos", a rede terá uma dinâmica sub-exponencial no pior caso e polinomial no caso Médio.
- Em relação à "teoria do caos", esses dois comportamentos permitem que as redes neurais residuais transitem na fronteira de estabilidade para o caos e vice-versa.
- Além disso, consegue preservar a geometria espacial da imagem de input e consegue controlar o fluxo de informação do gradiente ao longo do treinamento.
- Este paper inicializou as FRN com diferentes hiperparâmetros para treinar o modelo utilizando o dataset do MNIST.

Motivação do uso de redes neurais residuais

- O tempo de inicialização teórica conseguiu predizer corretamente o tempo de teste de performace dessa rede.
- Isso foi feito realizando a predição da quantidade de "explosão do gradiente".
- Inicialização de pesos como Xavier/Glorot e He (He-et-al) não foram ótimas para o modelo, pois as variáveis de inicialização dependem da profundidade da rede.

Expressão de Xavier

- *Xavier* inicializa todos os seus pesos na rede através de uma distribuição normal com média zero e uma variância específica.
- Geralmente é utilizada como inicializações de modelos com funções de ativação *tanh*.

$$Var(w_i) = \frac{1}{fanIn + fanOut}$$

- Onde *fanIn* é o número de unidades de input no tensor de peso e *fanOut* é o número de de neurônios que o resultado será passado para a próxima camada.
- w_i é o tensor de peso correspondente ao layer i.

Expressão de He ou He-et-al

- *He* é bem similar a inicialização do método *Xavier*, porém é multiplicado por um fator 2.
- Nesse método, todos os pesos são inicializados mantendo em mente o tamanho do layer anterior que ajuda a encontrar o mínimo global da função de custo mais rápido e mais eficiente.
- Todos os pesos ainda são iniciados randomicamente, porém agora essa "aleatoriedade" depende do tamanho do layer anterior.
- Essas características ajudam a manter uma inicialização mais controlada, rápida e mais eficiente para a descida do gradiente.
- É mais recomendando que se utilize a seguinte inicialização para a *He*.

$$Var(w_i) = \frac{2}{fanIn}$$

Aspecto Caótico

- Redes neurais clássicas apresentam comportamento estável ou caótico em cada camada dependendo da variância dos pesos em cada camada durante a inicialização do modelo.
- A distância de cosseno de dois vetores de input convergem para um valor pertencente ao intervalo [0, 1] em ordem polinomial.
- Se esse valor for 1 (vetores paralelos), então seu comportanto é estável.
- Caso contrário, se o valor for 0 (vetores perpendiculares), então seu comportamento é caótico.

Desvantagem de modelos estáveis e caóticos

- Muita estabilidade dificulta que o modelo veja a diferença entre dois vetores distintos de input.
- Em contrapartida, acrescentar um pouco de caos pode aumentar a expressatividade do modelo.
- Muito caos pode fazer que a rede pense que dois vetores similares sejam bem diferentes para o modelo.
- Além disso, essa inicialização controla até onde a informação do gradiente pode ser propagada ao longo da rede.
- As redes com formato caóticos tendem a sofrer com "explosão do gradiente". Enquanto redes neuráis estáveis tendem a sofrer com o problema de desaparecimento do gradiente.

Definição de um neurônio residual totalmente conectado (FRN)

- Uma operação dentro de um neurônio é definida da seguinte forma :

$$x_i^{(l)} = \sum_j v_{ij}^{(l)} * \phi(h_j^{(l)}) + x_i^{(l-1)} + a_i^{(l)}$$

$$h_i^{(l)} = \sum_j w_{ij}^{(l)} * x_j^{(l-1)} + b_i^{(l)}$$

- Considerando uma rede neural com L layers onde cada layer l tem uma quantidade de $N^{(l)}$ neurônios.
- Para facilitar, o layer 0 é o layer de input e todas as camadas escondidas possuem a mesma largura.
- Ou seja, o valor de $N^{(l)} = N$ para todo $l > 0$.
- $h^{(l)}$ é o layer de pré-ativação, $w^{(l)}$ é a matriz de peso do layer l, $b^{(l)}$ é o vetor de bias e ϕ é uma função não linear, por exemplo uma *Tahn* ou *ReLU*.
- Além disso, $v_{ij}^{(l)}$ corresponde ao peso associado ao layer do salto e $a_j^{(l)}$ ao bias associado ao layer do salto.

Definição de um neurónio residual resuzido (RRN)

- Uma operação dentro de um neurónio é definida da seguinte forma :

$$x_i^{(l)} = \phi(h_i^{(l)}) + x_i^{(l-1)}$$

$$h_i^{(l)} = \sum_j w_{ij}^{(l)} * x_j^{(l-1)} + b_i^{(l)}$$

- Onde todos os os parâmetros são os mesmos de um modelo FRN

Comportamento Médio

- É mais interessante analisar o caso médio do algorítmo e como os pesos $v_{ij}^{(l)}$ e $w_{ij}^{(l)}$ e os bias $b_i^{(l)}$ e $a_i^{(l)}$ se comportam.
- Para isso, foi feito uma análise do variância média desses valores utilizando a *distribuição normal gaussiana*.
- Um exemplo disso seria verificar a variância $w_{ij}^{(l)}$ sendo definida como $\Delta w_{ij}^{(l)} = \sigma_w^2 / N^{(l)}$.
- O único valor que é calculado de forma diferente é a variação não linear de h : $\Delta h_i^{(l)} = \sigma_w^2$.
- Assumindo que cada $x_j^{(l-1)}$ é fixo.

Resultados teóricos

- para comparação do comportamento do algorítmo, foi desenvolvido 4 modelos com diferentes funções não lineares.
- Uma RRN com função de ativação *Tanh* (i)
- Uma FRN com a função de ativação *Tanh* (ii)
- Uma FRN com a função de ativação *ReLU* (iii)
- Uma FRN com a função de ativação α-*ReLU* (iv)
- Onde α-*ReLU* é $\psi_\alpha(x) = x^\alpha$ se $x > 0$ e 0 caso contrário.
- Além disso, $\alpha < 1$.
- Todos esses modelos convergiram com o tempo experado do paper.

Algumas expressões

- Quantidade de tamanho : $p^{(l)} =< x^{(l)}, x^{(l)} >$ para $l > 0$ e $p^{(0)} =< x^{(0)}, x^{(0)} > /N$
- Quanidade de corelação : $\gamma^{(l)} =< h^{(l)}, h^{(l)'} >$ para $l > 0$ e $\gamma^{(0)} =< x^{(0)}, x^{(0)} > /N$
- Metrica de expressividade : $s^{(l)} = \frac{1}{2*N} < ||x^{(l)} - x^{(l)'}||^2 >= \frac{1}{2*N}(< (x^{(l)})^2 > -2* < x^{(l)}, x^{(l)'} > + < (x^{(l)'})^2 >) = \frac{1}{2} * (p^{(l)} + p^{(l)'}) - \gamma^{(l)}$
- Distância de cossenos de quantidade ou expressividade angular : $e^{(l)} = \gamma^{(l)} / \sqrt{p^{(l)} * p^{(l)'}}$

Algumas expressões

- Foi assumido que $p^{(l)} = p^{(l)'}$ e $q^{(l)} = q^{(l)'}$ para $l \geq 0$
- Assim, temos as expressões $e^{(l)} = \gamma^{(l)}/p^{(l)}$ e $s^{(l)} = p^{(l)} - \gamma^{(l)} = (1 - e^{(l)}) * p^{(l)}$
- Dado uma função de perda E e o vetor gradiente $(\partial E/\partial x_i^{(L)})_i$
- A quantidade de gradiente é definida sendo $\chi^{(l)} =< (\partial E/\partial x^{(l)})^2 >$
- $\chi_*^{(l)} =< (\partial E/\partial *^{(L)})^2 >$ para * = a, b, w e v

Resultados experimentais

- Foi implementado uma rede residual sem normalização de batch e camadas convolucionais, não apresentando o desempenho prático do modelo.
- Para a parte experimental, os modelos mostraram uma discotomia no que realmente importa para iniciar o modelo.
- Para os modelos (i) e (ii), a qualidade da inicialização dependia da quantidade que o "gradiente explodia" na taxa $R^{(L)} = \chi^{(0)}/\chi^{(L)} \approx e^{\theta(\sqrt{L})}$ (Taxa sub-exponencial).
- Onde $\chi^{(l)}$ corresponde ao tamanho do gradiente no layer l.
- Quanto maior for R, o modelo ficará mais propício a divergir.
- Já o modelo (iv) era determinado pela expressão $s^{(L)} = (1 - e^{(L)}) * p^{(L)} = \theta(p) = \theta(L)$.
- Ou seja, dependia da expressividade randômica da rede.
- Já para o modelo (iii), o mesmo dependia da expressão $s^{(L)} = (1 - e^{(L)}) * p^{(L)} = exp(\sqrt{\theta(L)})$.

Resultados experimentais de *Tanh*

- Para o parâmetro σ_w^2, redes mais profundas são mais favorecidas quanto menor for σ_w.
- Para FRN, o parâmetro σ_v foi fixado sendo $\sigma_v = 1$.
- Além disso, foi comprovado, experimentalmente e teoricamente, que $R \approx \sigma_w * \sqrt{L}$.
- Por que a melhor inicialização não é da forma $R = 1 \Leftrightarrow \sigma_w = 0$?
- Isso ocorre pois quando L e/ou σ_w é pequeno, toda essa dinâmica do gradiente não influencia mais na qualidade da inicialização.
- Com isso, não há "muito espaço para o gradiente explorar". Consequentemente, o modelo fica mais suscetível a não ter tanta expressividade e o desempenho é afetado negativamente.

Resultados experimentais de *Tanh*

- A expressão $\rho^2 = \sigma_a^2/\sigma_v^2$ determina o ponto fixo $e^* < 1$ e sua taxa de convergência.
- Além disso, essa expressão também contribui para a taxa de explosão do gradiente.
- Para amenizar essa problema, os termos σ_v e ρ foram variados de tal forma que a expressão $\sigma_v/\sqrt{1+\rho^2}$ fosse constante.
- Dessa forma, foi obtido que o principal termo do log do crescimento do gradiente se tornasse praticamente constante para cada L e ρ.
- Como resultado final, foi percebido que a performace é maximizada em um valor fixo de L que não depende de ρ, mostrando que a dinâmica do gradiente determine a qualidade da unicialização em tanh resnets.
- Também foi percebido que aumentando o valor de ρ aumenta um pouco a performace do modelo.

Resultado experimentais de *ReLU*

- Variando o parâmetro $\sigma_w^2 \in [0, 1.5]$, enquanto os valores $\sigma_v^2 = 1$, $\sigma_a^2 = \sigma_b^2 = 1/2$ foram fixos.
- Os resultados mostraram problemas númericos juntamente com problemas de "explosão do gradiente".
- Essas regiões mais problemáticas geralmente aconteciam quando $p^{(L)}$ (medida média da magnetude de ativação de um neurônio no layer L) era muito grande.
- Entretanto, para valores abaixo de L com esses problemas, apresentaram as melhores acurácias
- Todas as dinâmicas $s^{(l)}$, $R^{(l)}$ e $p^{(l)}$ foram praticamente as mesmas. Ou seja $e^{\sqrt{\theta(l)}}$.
- Os valores mais altos de $s^{(L)}$, $R^{(L)}$ e $p^{(L)}$ foram os melhores valores para o modelo.

Resultado experimentais de α-*ReLU*

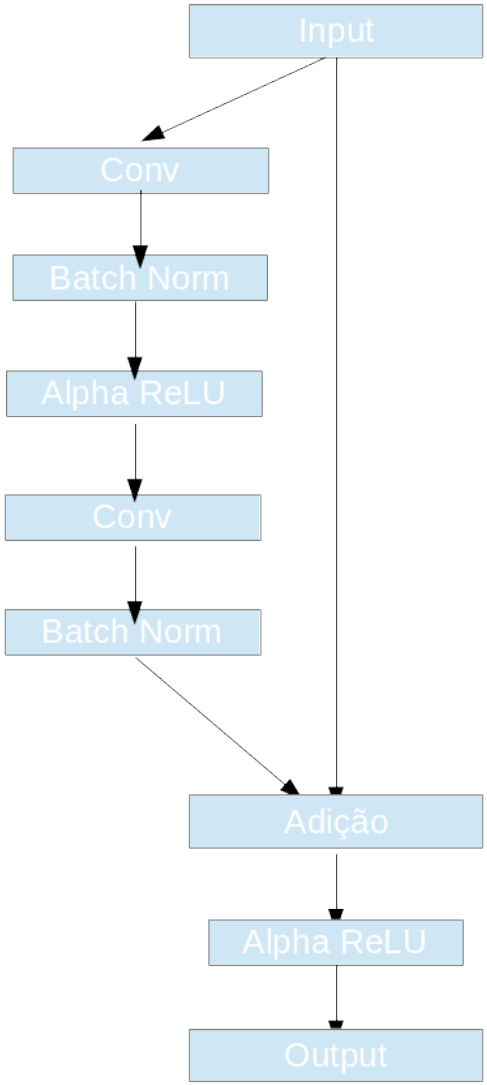
- Todos os parâmetros σ foram fixos e apenas α foi variado.
- Os resultados mostraram um comportamento parecido com *ReLU* : quando o modelo possui muitas camadas, mais sucetível fica para problemas numéricos.
- Porém, para valores logo abaixo de L com esse problema, foram os que apresentaram melhor performace do modelo.
- Por interpolação nos dados de $s^{(l)}$, foi possível verificar que as funções *ReLU* são expressivas e não treináveis.
- Com essas conclusões, foi possível determinar a performace do modelo durante a fase de teste.

Proposta de implementação do modelo

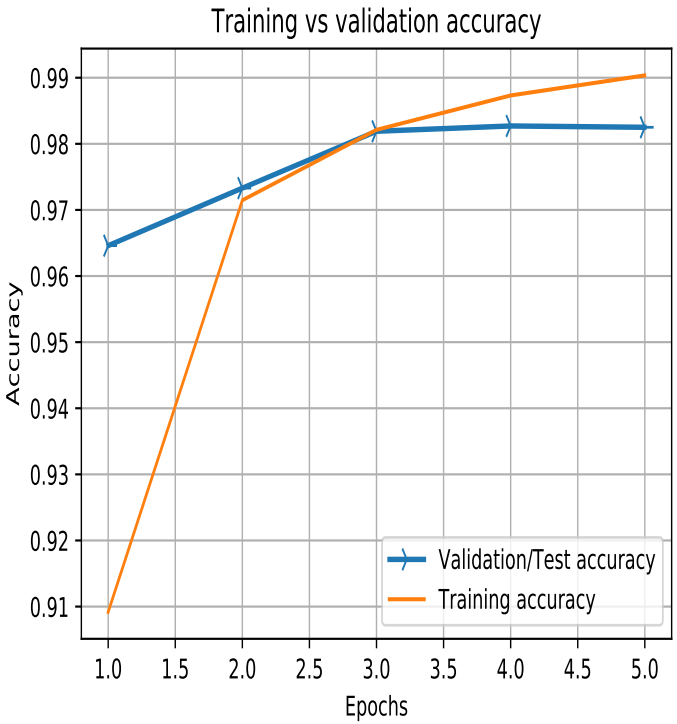
- É possível implementar esse modelo neural utilizando alguma *ResNet* ou "Na mão".
- Com a biblioteca *Keras* e *Tensorflow* do *Python*, foi implementado as operações de convoluções e de "batch normalization".
- Também é possível implementar essa função α-*ReLU* utilizando as funções *lambda* ou utilizar outras *ReLUs* (*Leaky ReLU*, *BReLU*, *RReLU*, dentre outras).
- Foi implementado 7 modelos *FRN* inspirado nos modelos (ii), (iii) e (iv).
- Onde quatro modelos iniciaram seus pesos com a função *He*.
- Os outros três foram iniciados com a função *Xavier*.

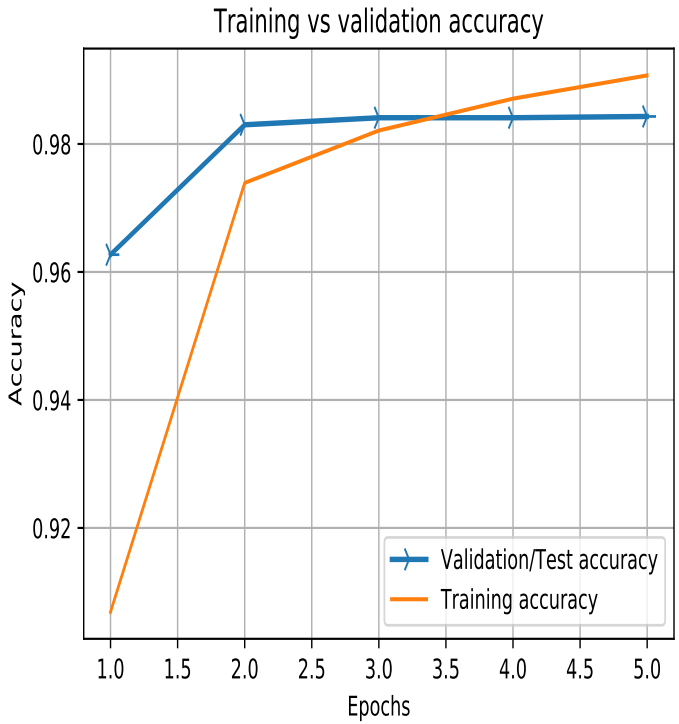
Proposta de implementação da rede

Proposta de implementação do modelo com a adição das convoluções e normalização.

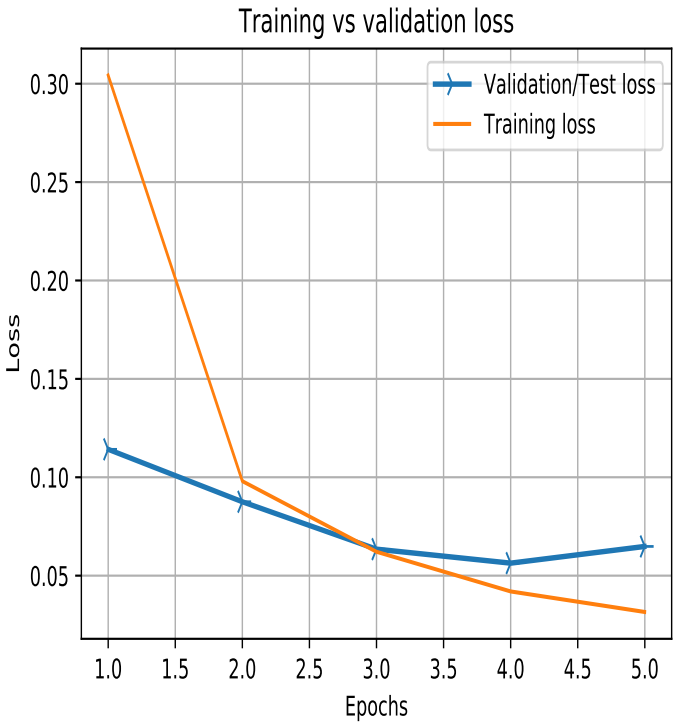


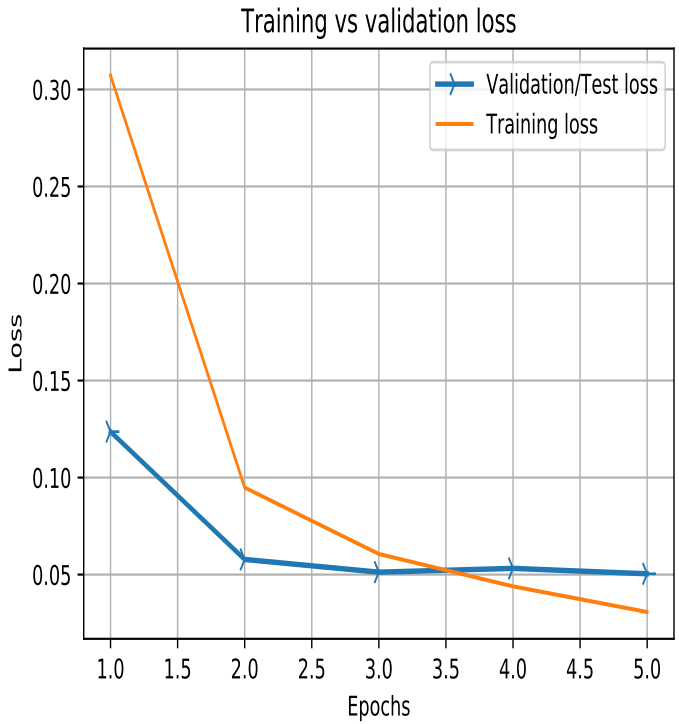
Modelo He vs Xavier Tanh



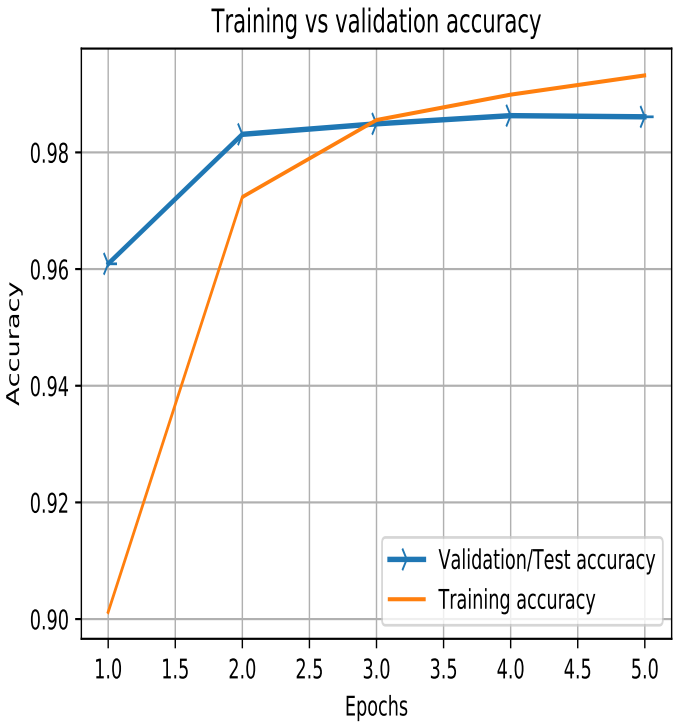


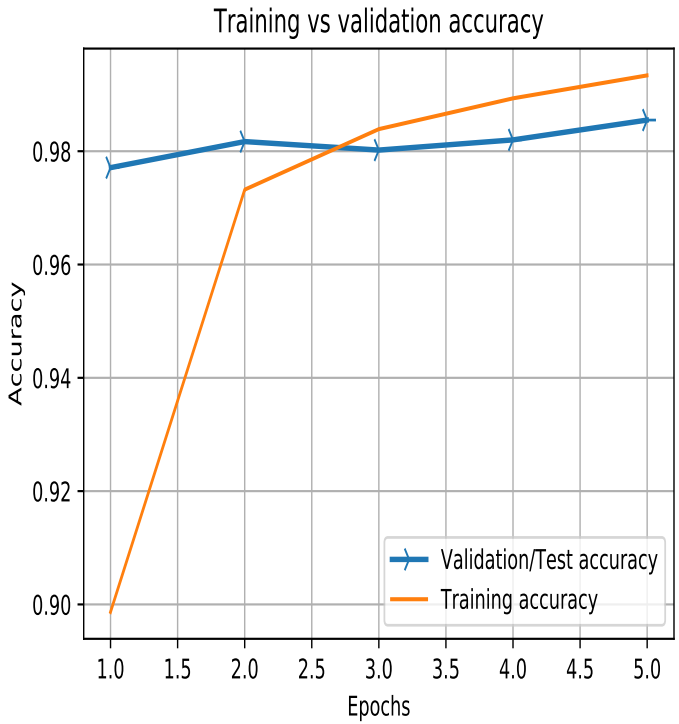
Modelo He vs Xavier Tanh



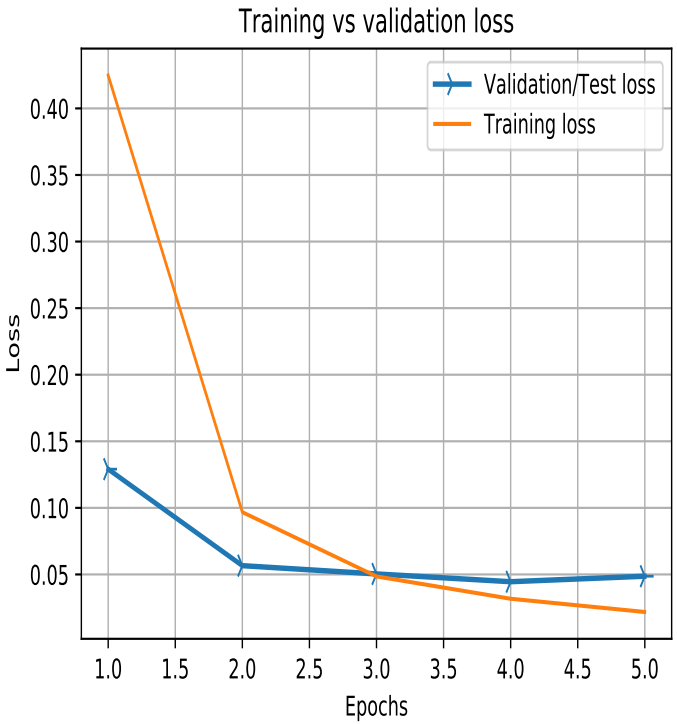


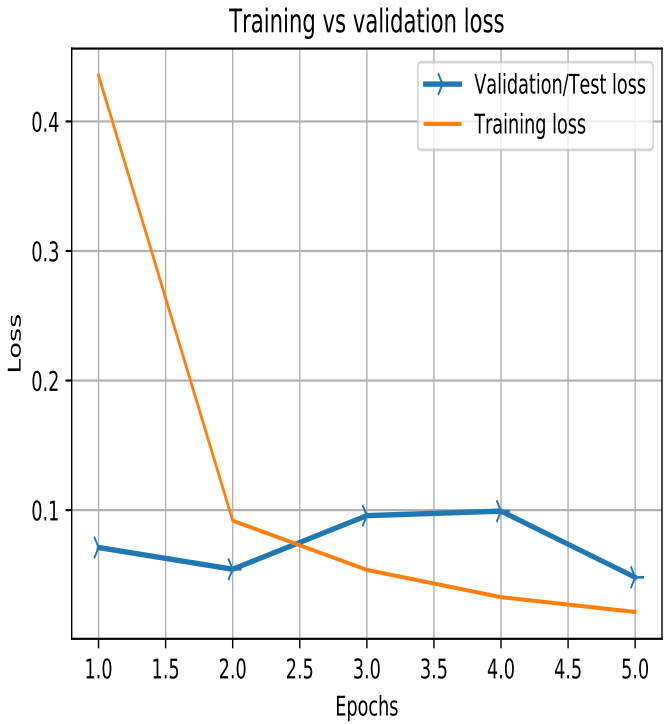
Modelo He vs Xavier Relu



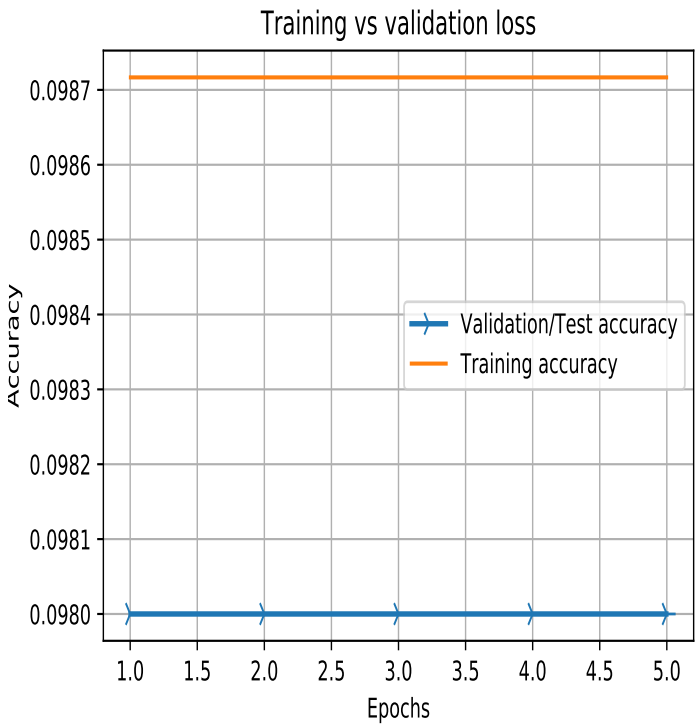


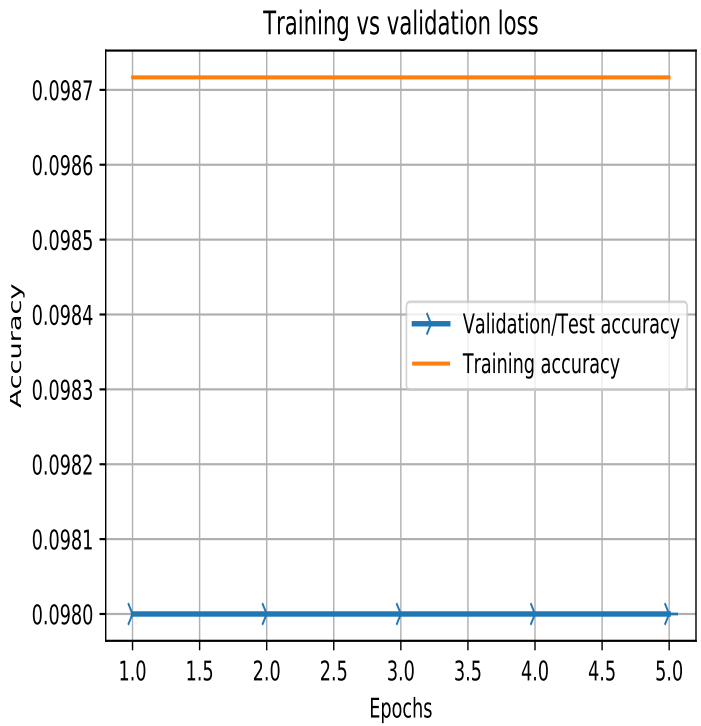
Modelo He vs Xavier ReLU



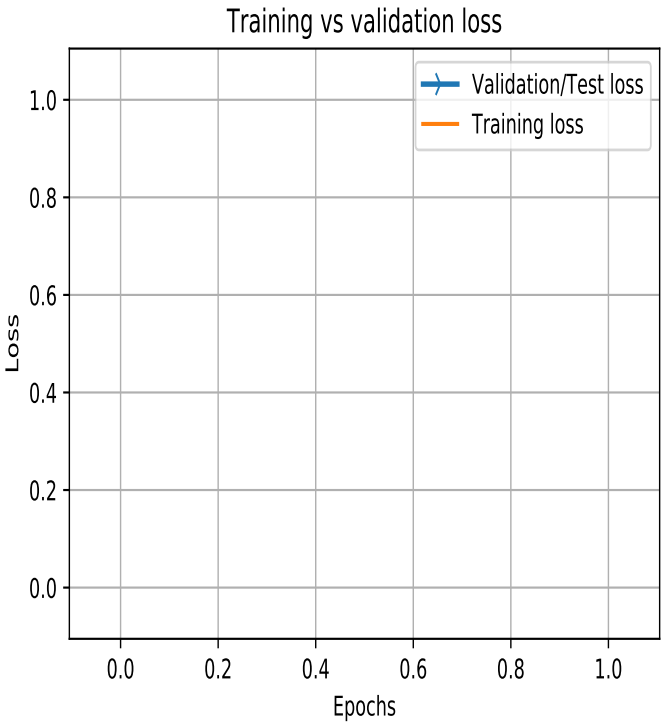


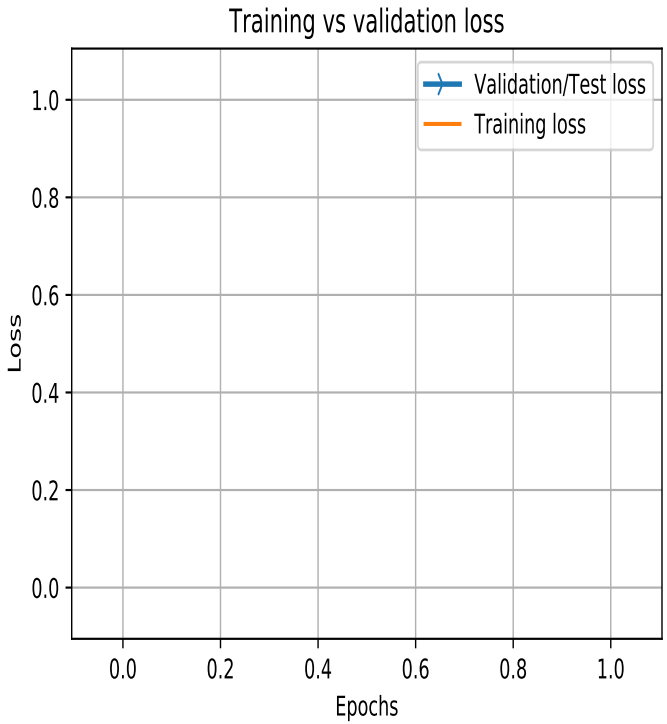
Modelo He vs Xavier α-ReLU



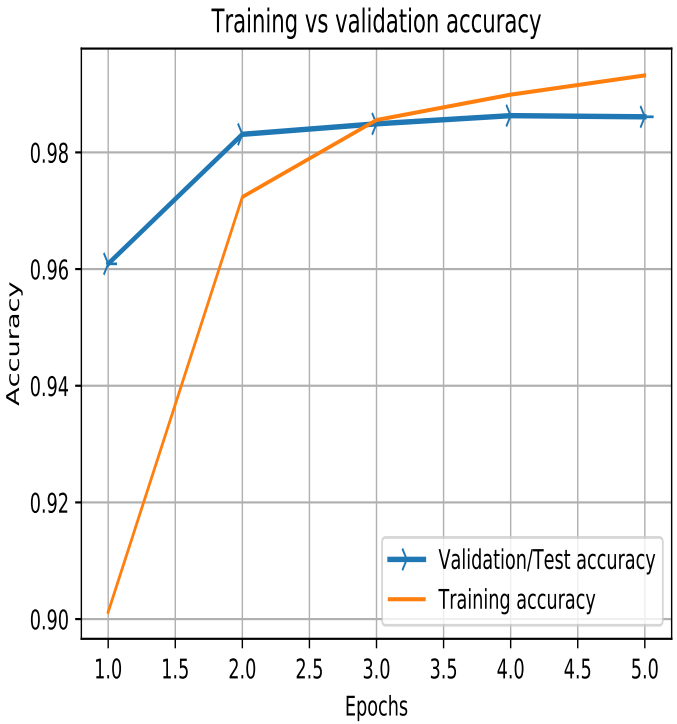


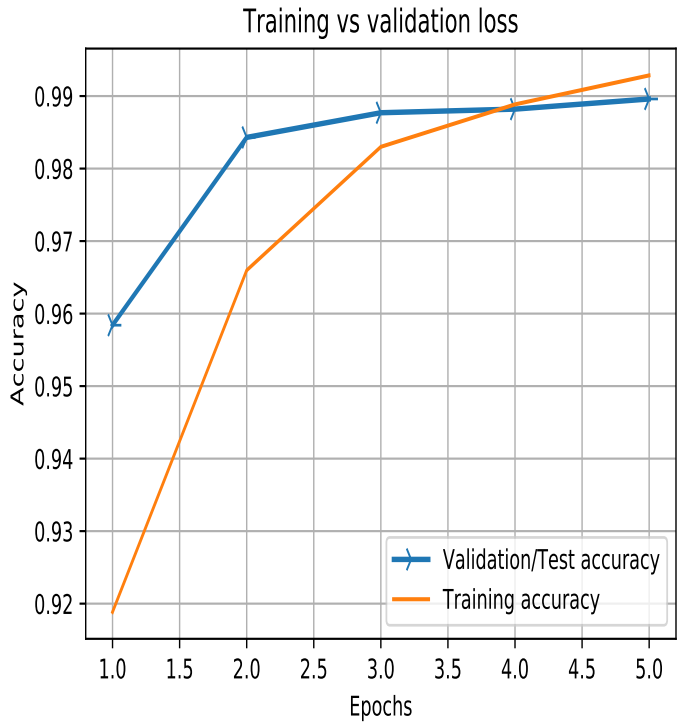
Modelo He vs Xavier α-ReLU



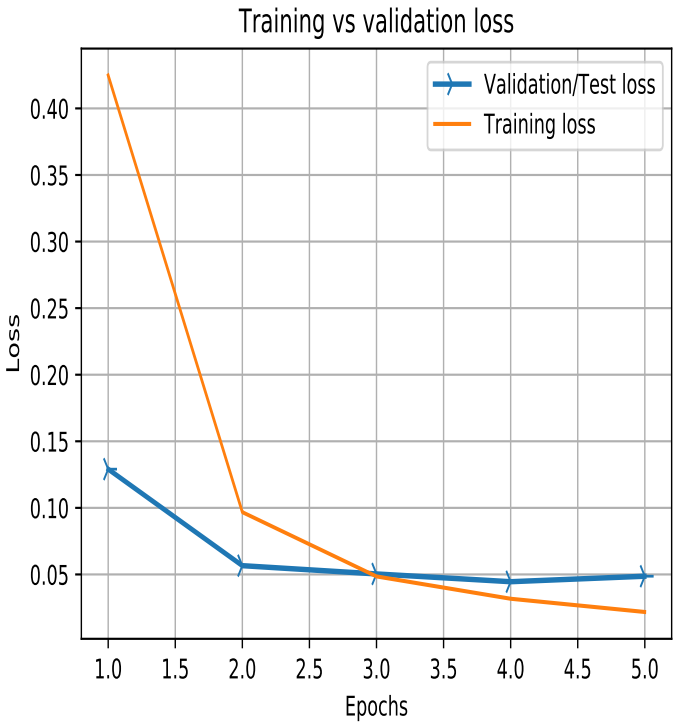


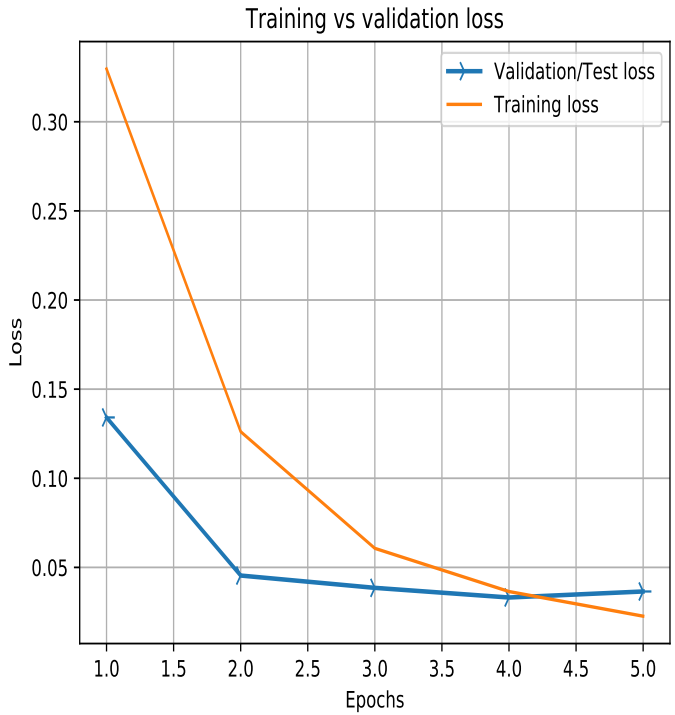
Variância do σ de *He* usando ReLU (seed = 42)



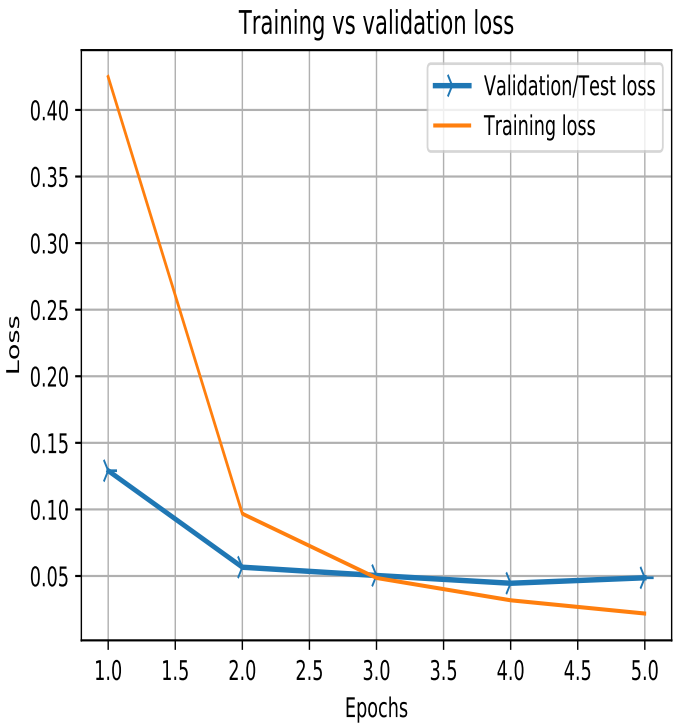


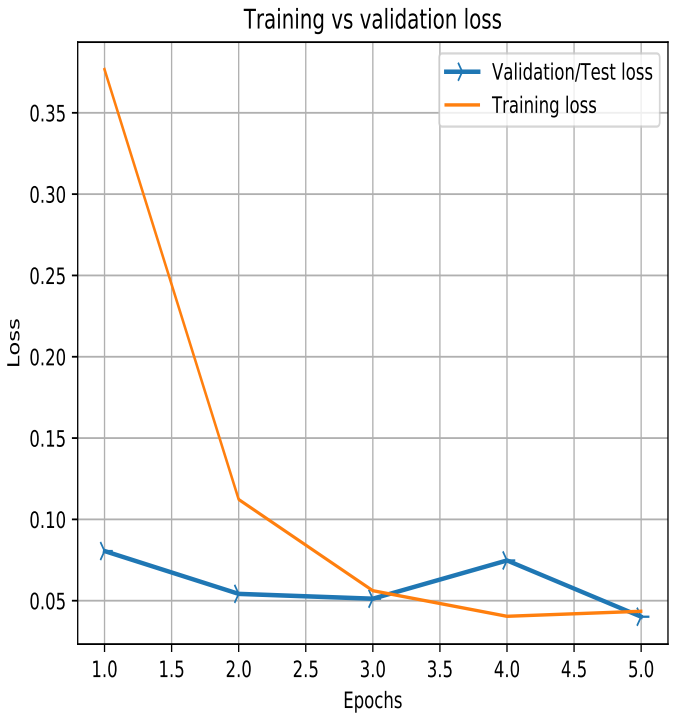
Variância do σ de *He* usando ReLU (seed = 42)



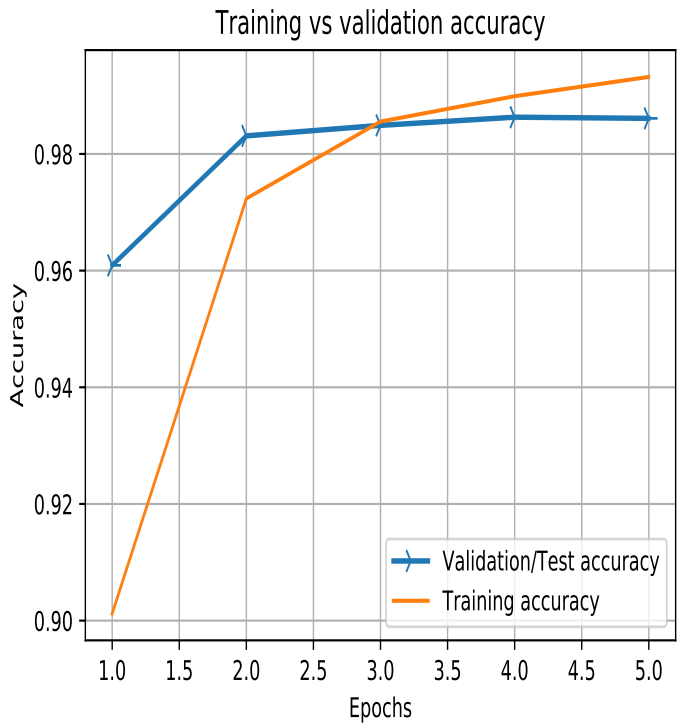


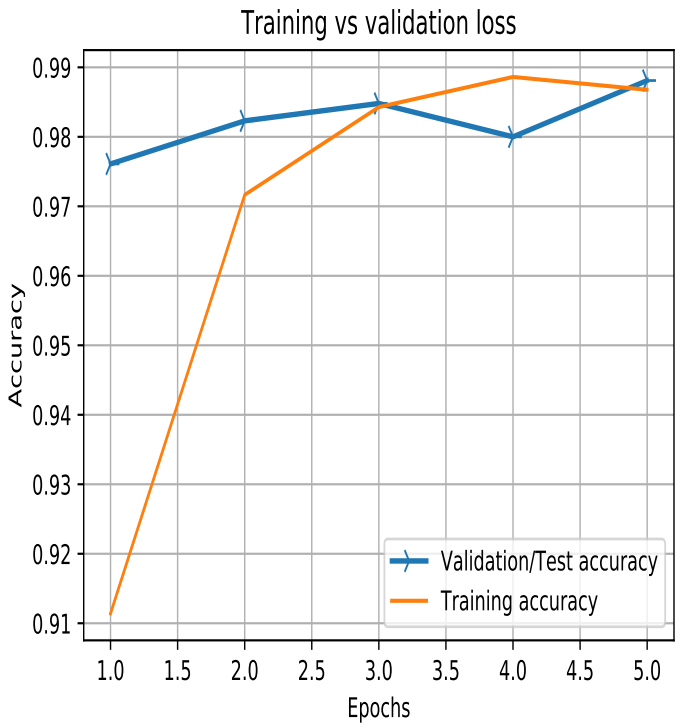
Modelo *He* normal usando ReLU Vs Modelo *He* uniforme usando ReLU





Modelo *He* normal usando ReLU Vs Modelo *He* uniforme usando ReLU





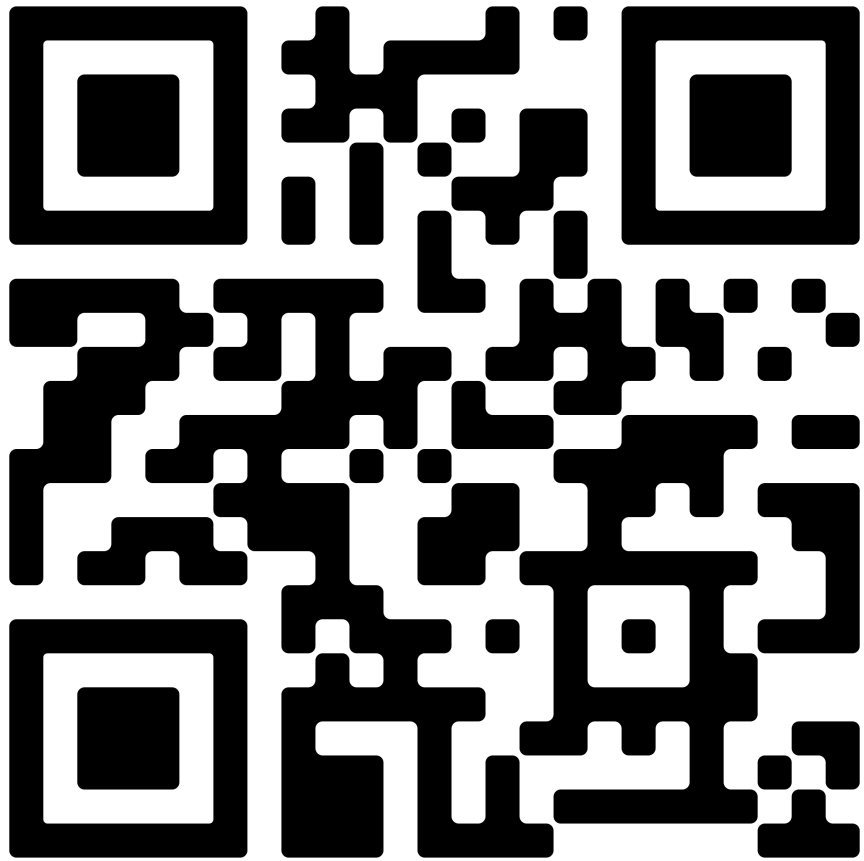
Conclusão

- Foi provado e verificado que todas as redes residuais descritas aqui não colapsam a geometria do input original ou a perda do gradiente de forma exponencial.
- A teoria foi extremamente preditiva em relação ao desempenho do tempo de teste. Apesar de que apenas isto não confirma muito sobre a dinâmica de treinamento.
- Além disso, foi verificado teoricamente e experimentalmente que uma inicialização de pesos correta precisar ser levada em conta para modelos mais profundos, pois prejudicam negativamente o desempenho do modelo.

Conclusão

- As inicializações do tipo *Xavier* e *He* não são bons esquemas de inicialização para redes neurais residuais, devido as suas características mais estáticas.
- Os modelos com função α-*ReLU* apresentaram o pior comportamento. Isto se deve provavelmente pela implementação do modelo.
- Praticamente compromentem bastante a precisão do modelo.
- A única desvantagem de um modelo residal é em relação ao tempo de treinamento : no qual apresenta maior tempo para treinamento.
- Uma rede convolucional apresenta em média 1 minuto por época.
- Enquanto que um modelo residual apresenta em média 24 minutos por época.
- Ambos os modelos Residuais foram treinados em uma *GPU* (GeForce 940MX - 4GB)

- Email : borinmacedo@gmail.com
- Github : https://github.com/borin98/ApresetacaoResnets



Obrigado Pela atenção !!