Os 300 (Hyper)Parâmetros

Lucas Araujo
Instituto de Informática
Universidade Federal de Goiás
www.deeplearningbrasil.com.br

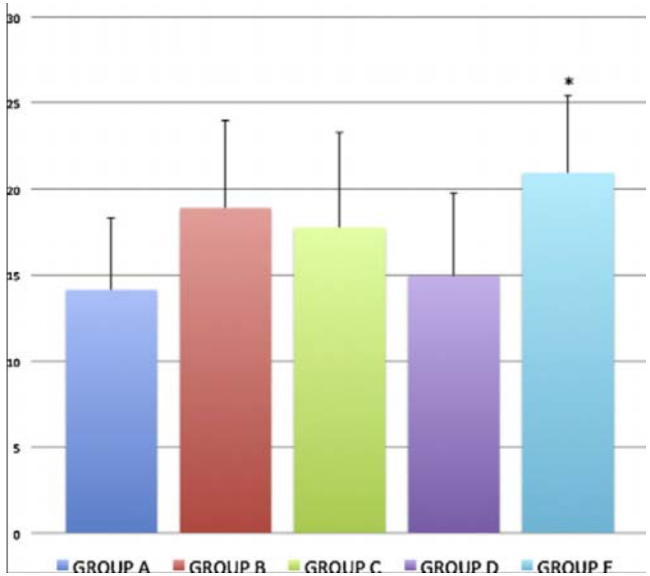
INSTITUTO DE
INFORMÁTICA
UFG

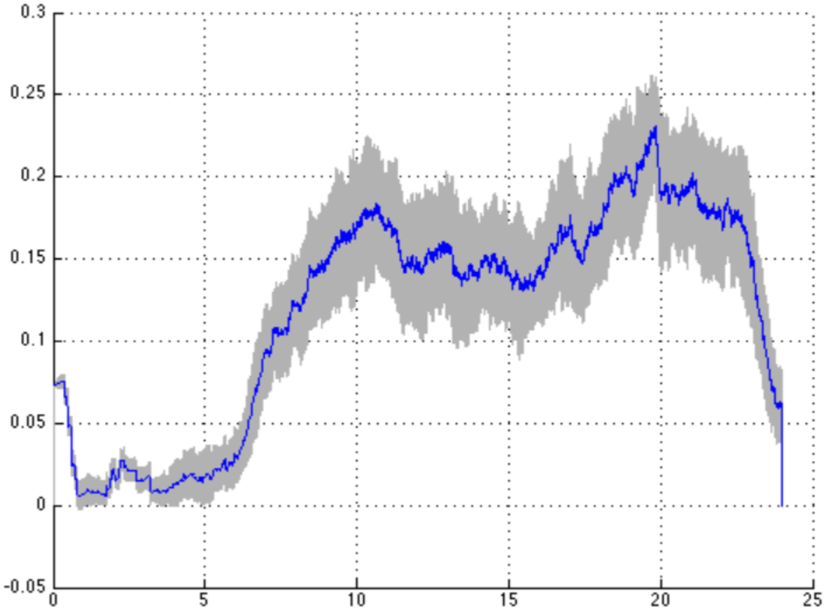
Tópicos:

- Preparação dos Dados
- Definição das Funções Custo e Métrica (classificação vs regressão)
- Interpretação e Correção de Resultados (Ciclo Vicioso de ML)
- Estudo de Caso: RSNA Bone Age Challenge 2017

Preparação de Dados:

- Visualização

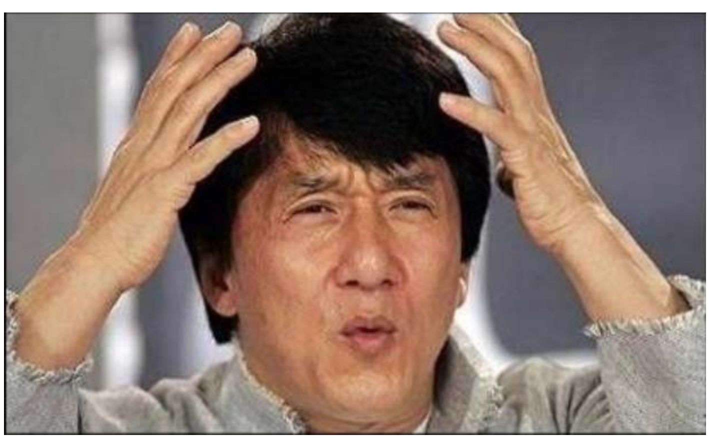


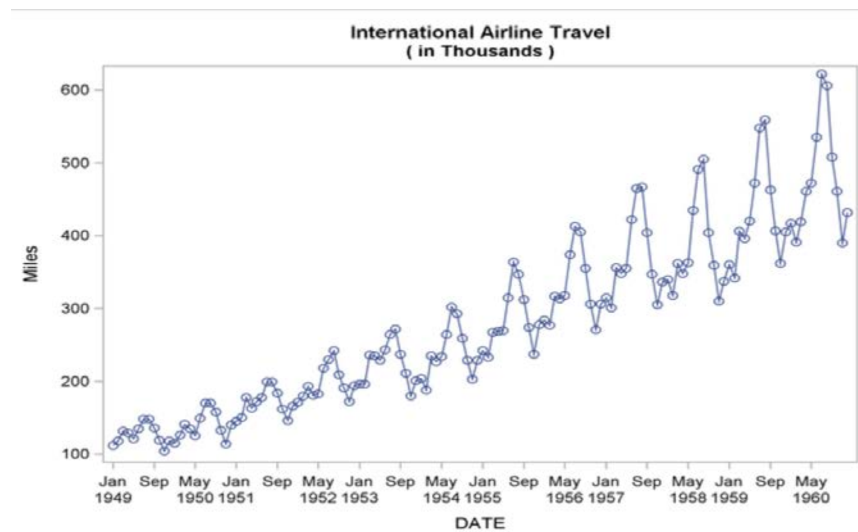


Preparação de Dados:

- Diferentes Tipos de Dados

↓	C1	C2	C3
	length	success	Recoded length
1	2	93.3	0
2	3	83.1	0
3	4	74.1	0
4	5	58.9	0
5	6	54.8	0
6	7	53.1	0
7	8	46.3	0
8	9	31.8	0
9	10	33.5	1
10	11	31.6	1
11	12	25.7	1
12	13	24.0	1
13	14	31.0	1



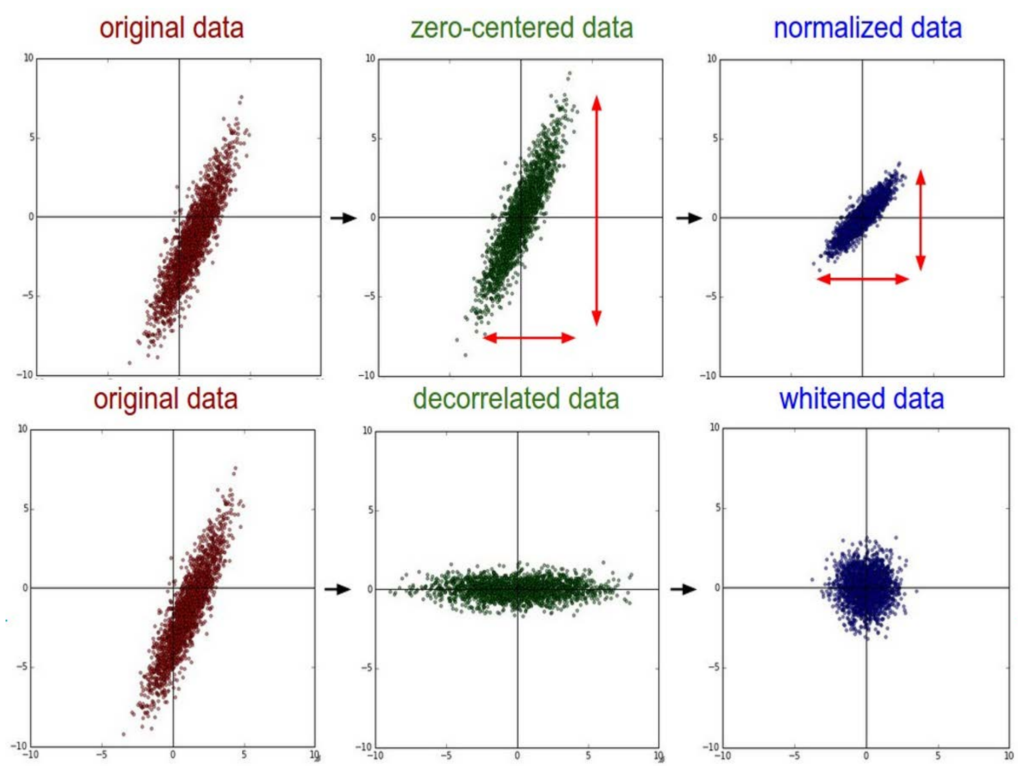


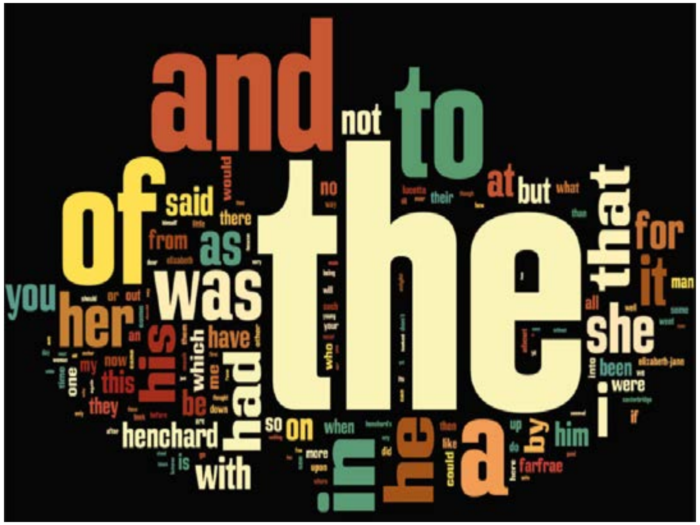
Preparação de Dados:

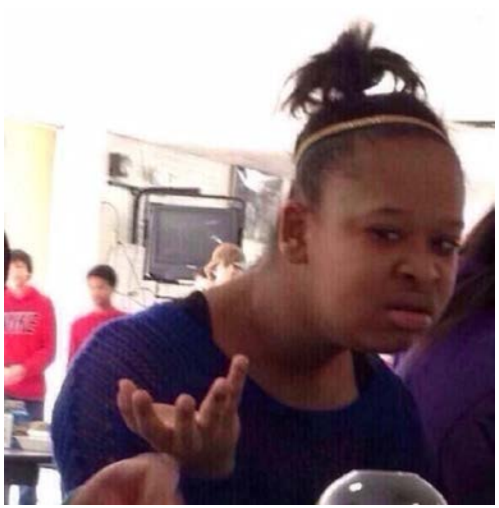
- Preprocessamento

ID	Gender
1	Male
2	Female
3	Not Specified
4	Not Specified
5	Female

ID	Male	Female	Not Specified
1	1	0	0
2	0	1	0
3	0	0	1
4	0	0	1
5	0	1	0







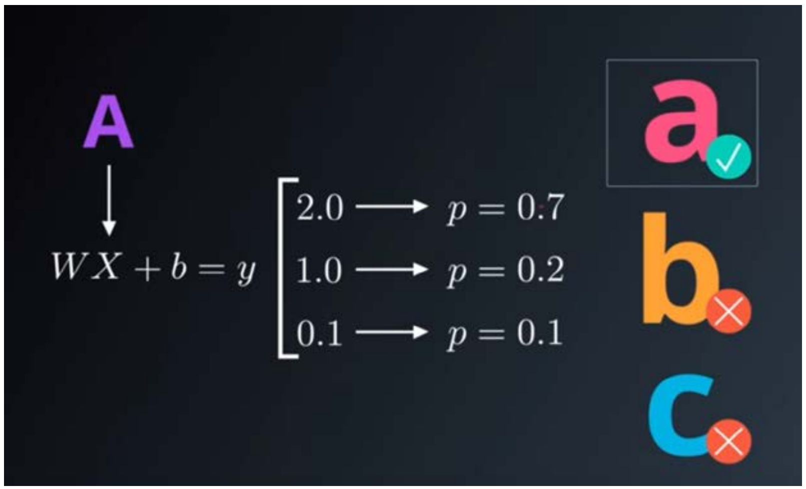
Preparação de Dados:

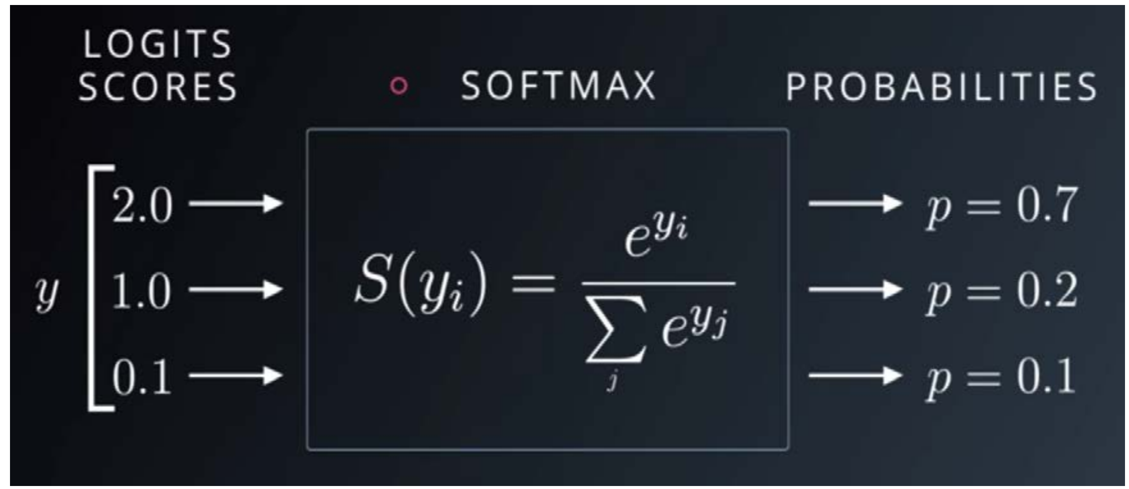
- Divisão treino/validação/teste

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> from sklearn import datasets
>>> from sklearn import svm

>>> iris = datasets.load_iris()
>>> iris.data.shape, iris.target.shape
((150, 4), (150,))
>>> X_train, X_test, y_train, y_test = train_test_split(
...     iris.data, iris.target, test_size=0.4, random_state=0)

>>> X_train.shape, y_train.shape
((90, 4), (90,))
>>> X_test.shape, y_test.shape
((60, 4), (60,))
```

Definição da Função Custo:

- Classificação





Entropia Cruzada: $H(p,q) = -\sum_x p(x) \log q(x)$, *H(p,q) ≠ H(q,p)*

Outros exemplos menos usados: Hinge, divergência de Kullback-Leibler, cosseno do ângulo entre os vetores, etc. (Google is your friend! :))

Definição da Função Custo:

- Regressão

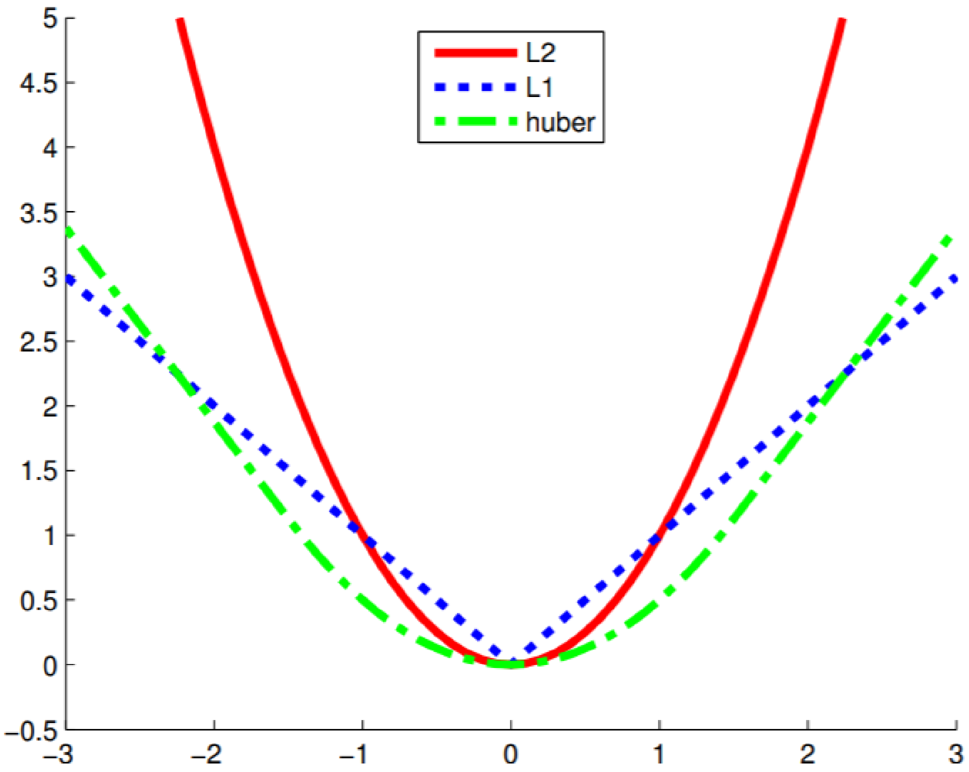
As mais usadas são:

L2 = $\mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}}||\boldsymbol{y} - f(\boldsymbol{x})||^2$

$$S = \sum_{i=0}^{n}(y_i - h(x_i))^2$$

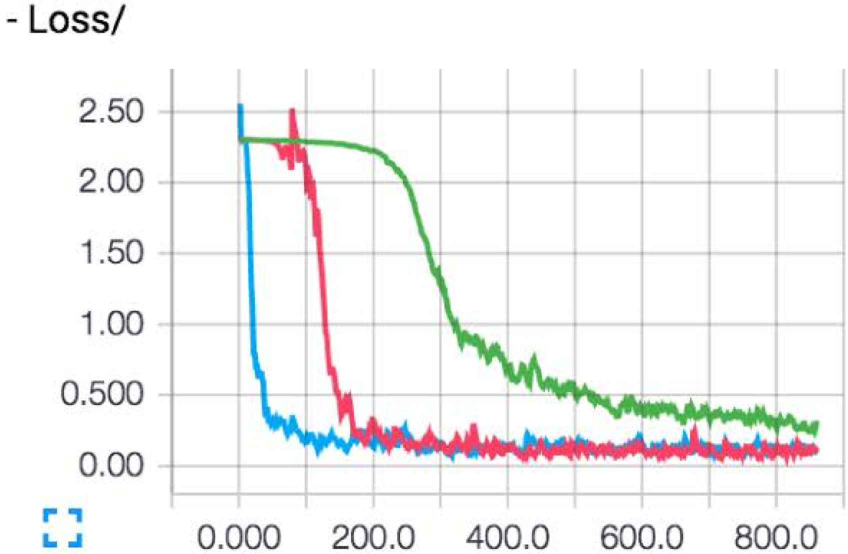
L1 = $\mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}}||\boldsymbol{y} - f(\boldsymbol{x})||_1$

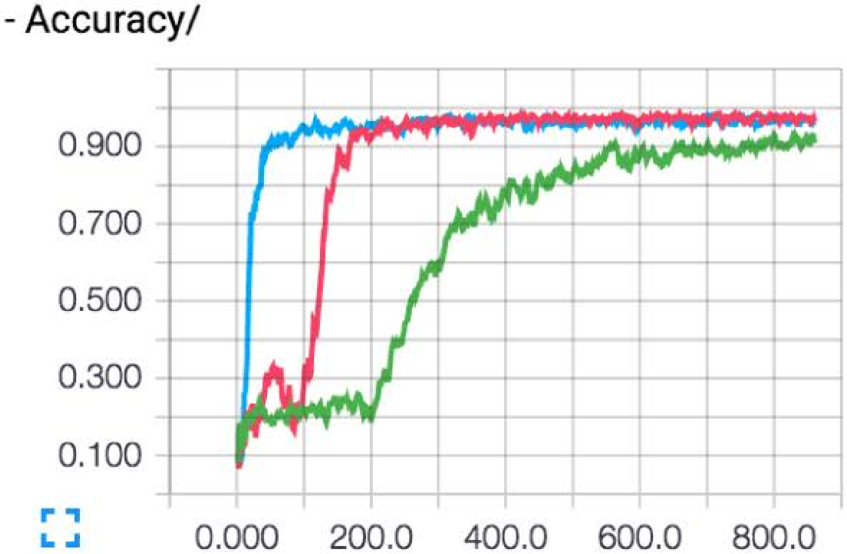
$$S = \sum_{i=0}^{n}|y_i - h(x_i)|$$



Definição da Métrica:

- Diferença entre Custo e Métrica

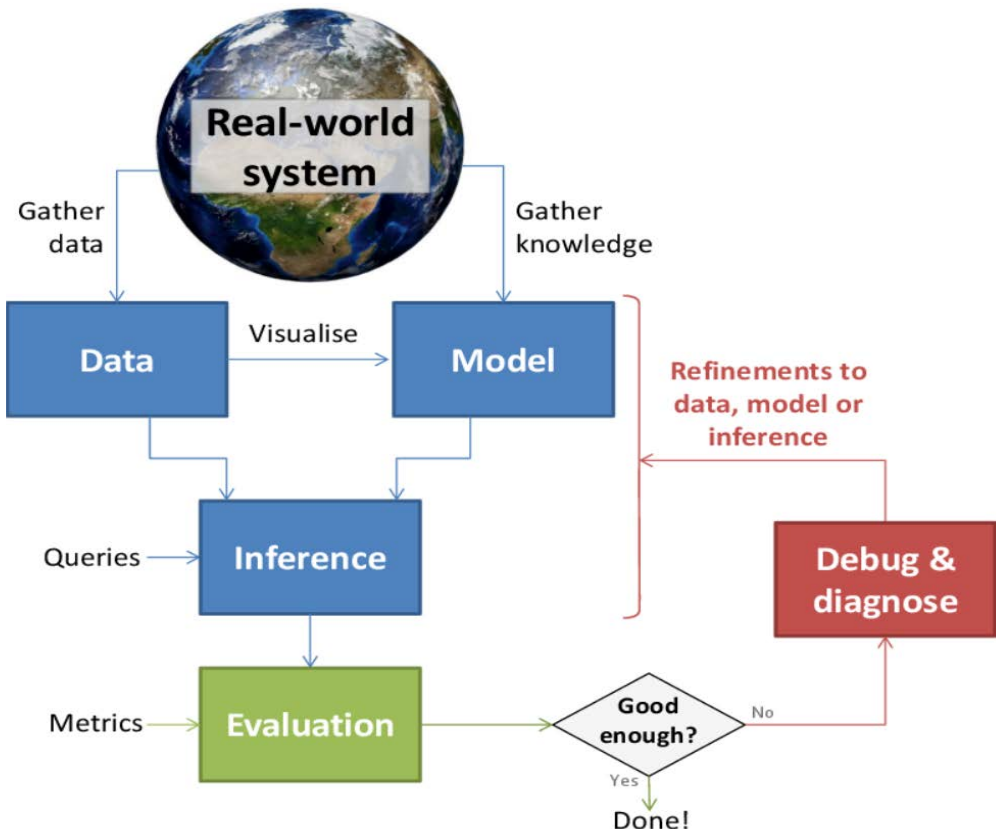


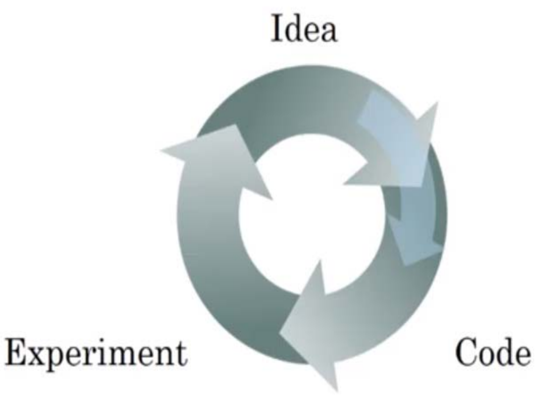


- Exemplos de métricas: acurácia, precisão, recall, F1 score, IoU, MAP, etc.

Interpretação e Correção de Resultados:

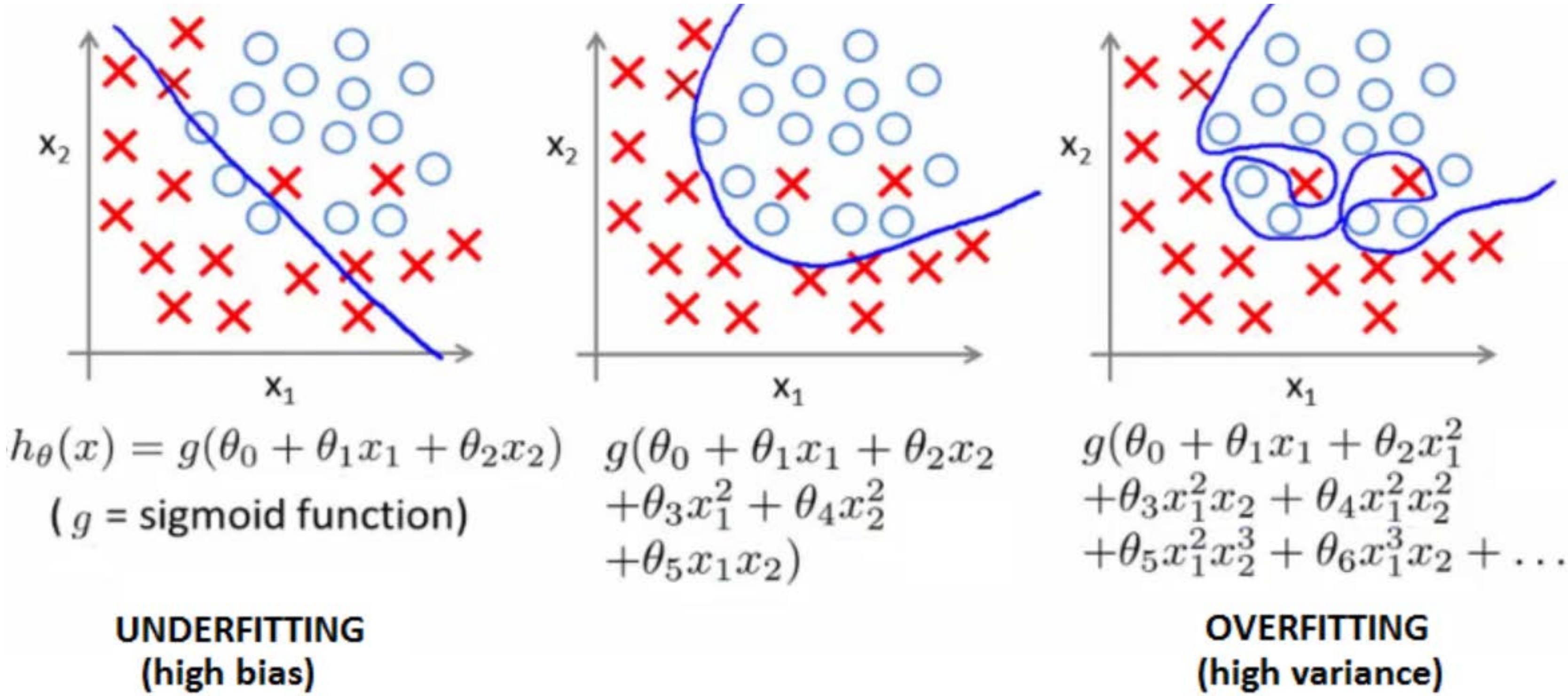
- Ciclo (Vicioso) de ML





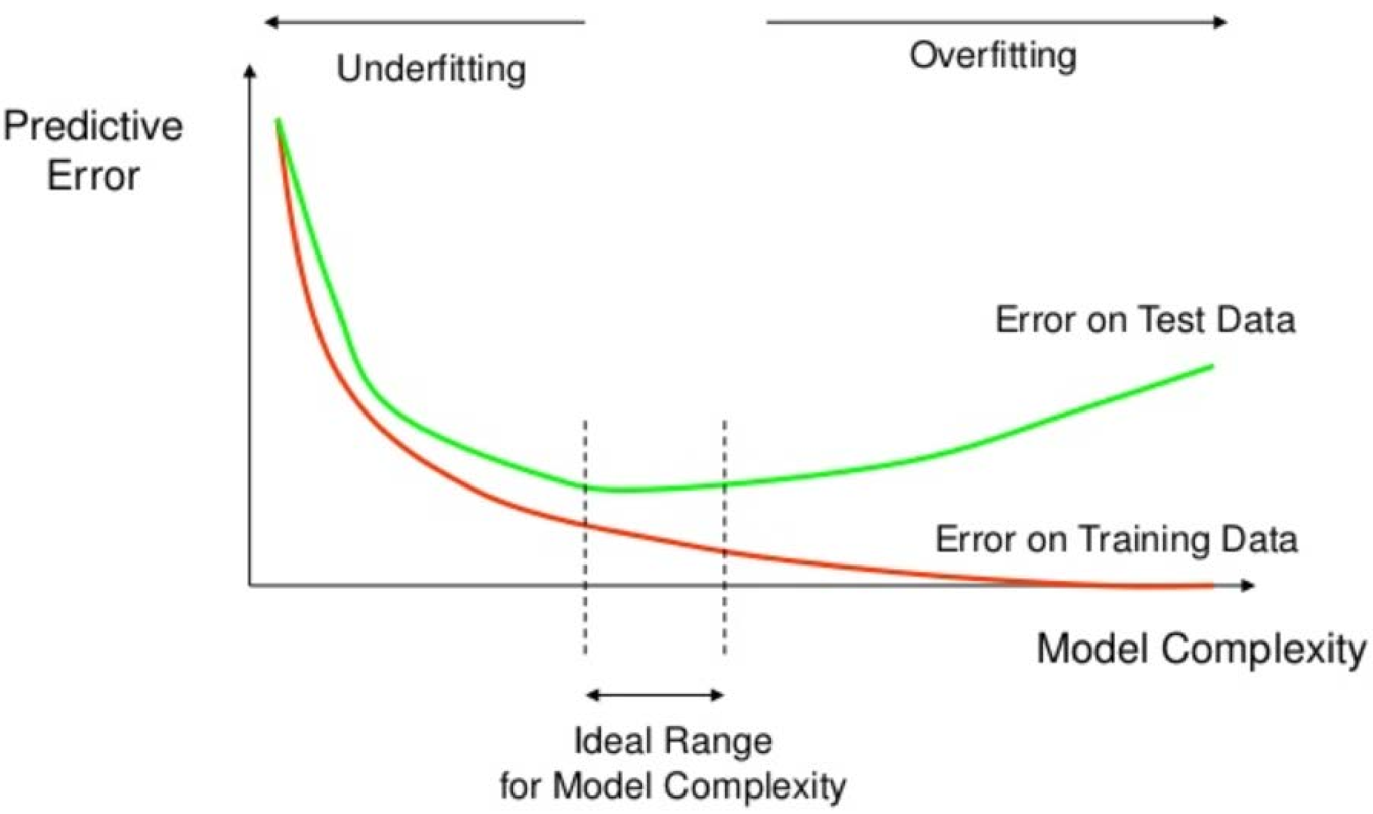
Interpretação e Correção de Resultados:

- Underfit vs Overfit



Interpretação e Correção de Resultados:

- Underfit vs Overfit



Interpretação e Correção de Resultados:

- Parâmetros: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}$ ...
- Hyperparâmetros:
  - Arquitetura da rede (MLP, CNN, RNN, …)
  - Número de camadas e número de unidades por camada (profundidade e largura da rede)
  - Função de ativação
  - Algoritmos de Otimização
  - Taxa de aprendizado
  - Técnicas de regularização
  - […] (sempre tem um que vc tá esquecendo... --')

Interpretação e Correção de Resultados:

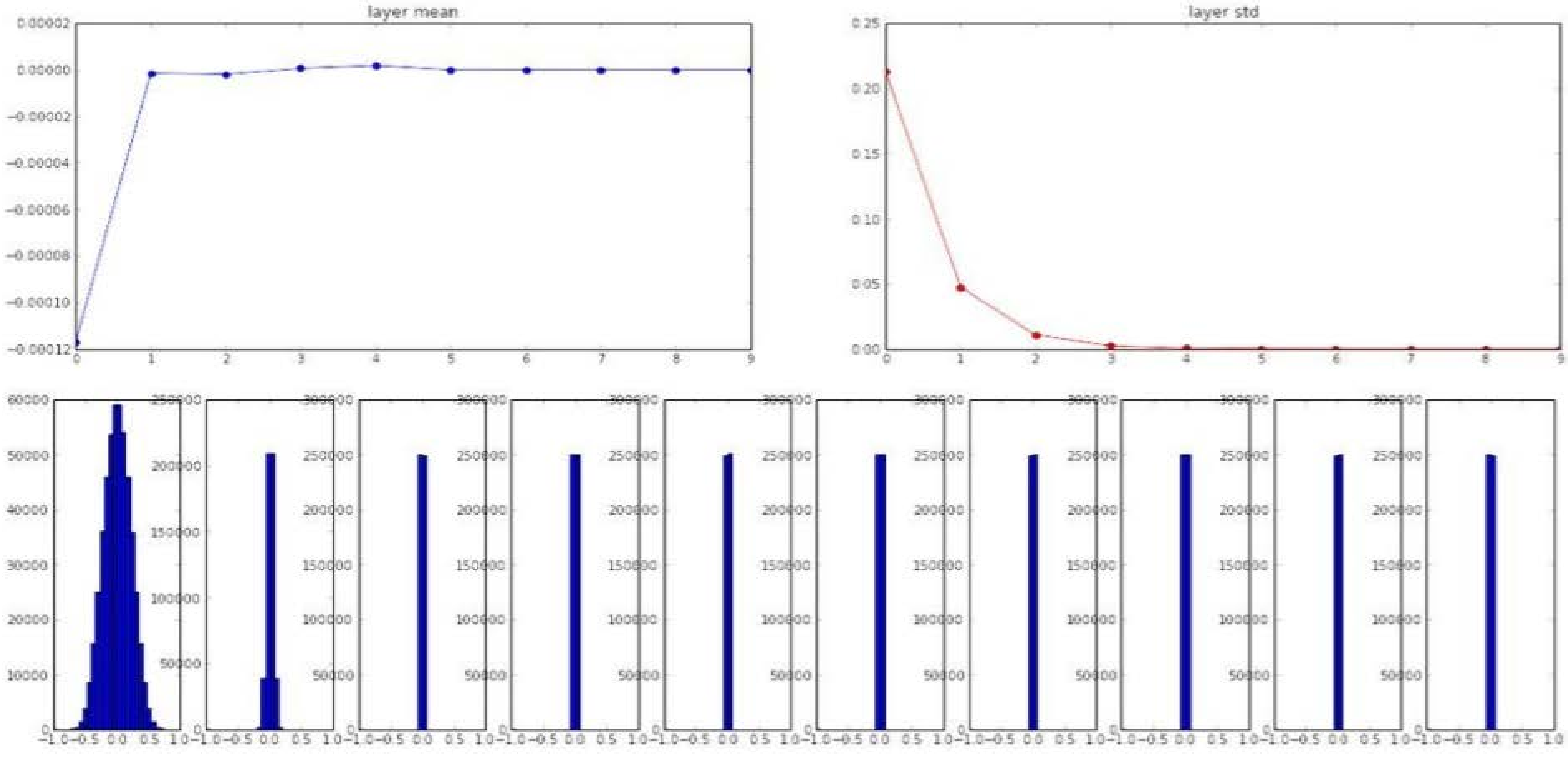
- Inicialização de Parâmetros
  - Iniciar tudo com 0
  - Iniciar tudo com o mesmo valor > 0
  - Iniciar tudo com o mesmo valor < 0
  - Iniciar com valores aleatórios com variância alta
  - Iniciar com valores aleatórios com variância baixa

Interpretação e Correção de Resultados:

- Inicialização de Parâmetros
  - Iniciar tudo com 0
  - Iniciar tudo com o mesmo valor > 0
  - Iniciar tudo com o mesmo valor < 0
  - Iniciar com valores aleatórios com variância alta
  - Iniciar com valores aleatórios com variância baixa

Interpretação e Correção de Resultados:

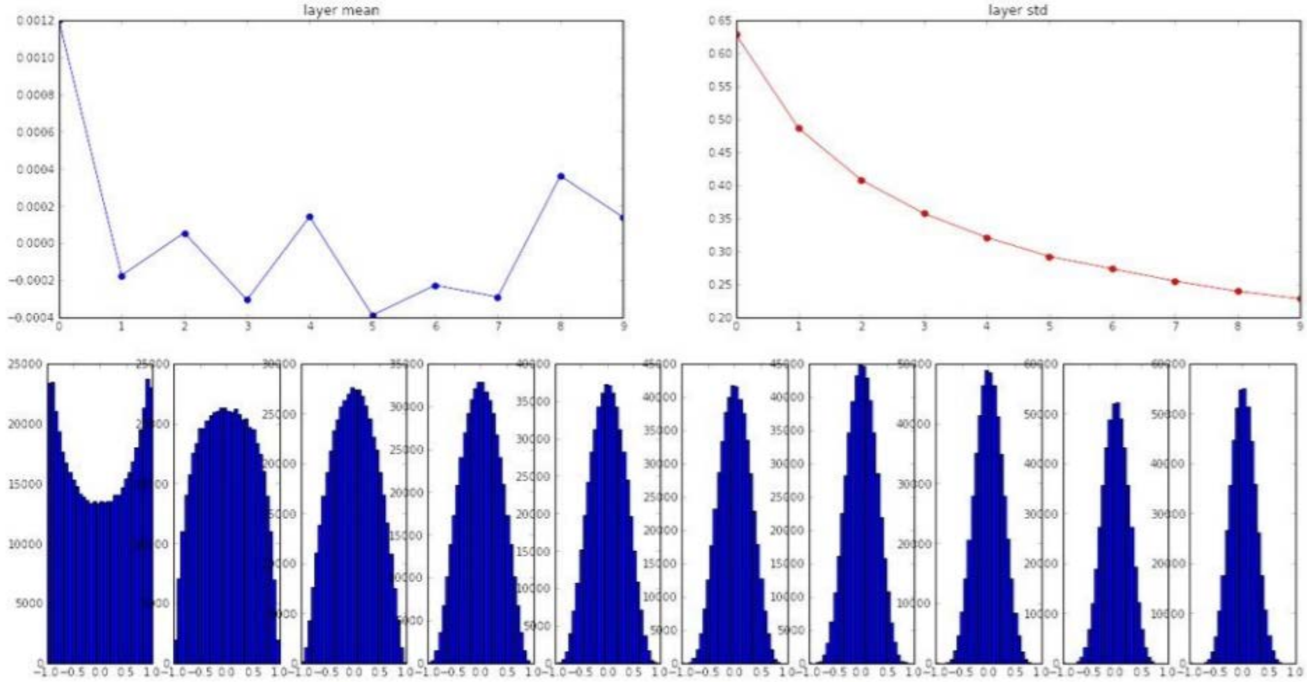
- Inicialização de Parâmetros
  - W = 0.01*np.random.randn(D, H)



Interpretação e Correção de Resultados:

- Inicialização de Parâmetros
  - Xavier Glorot (2010):

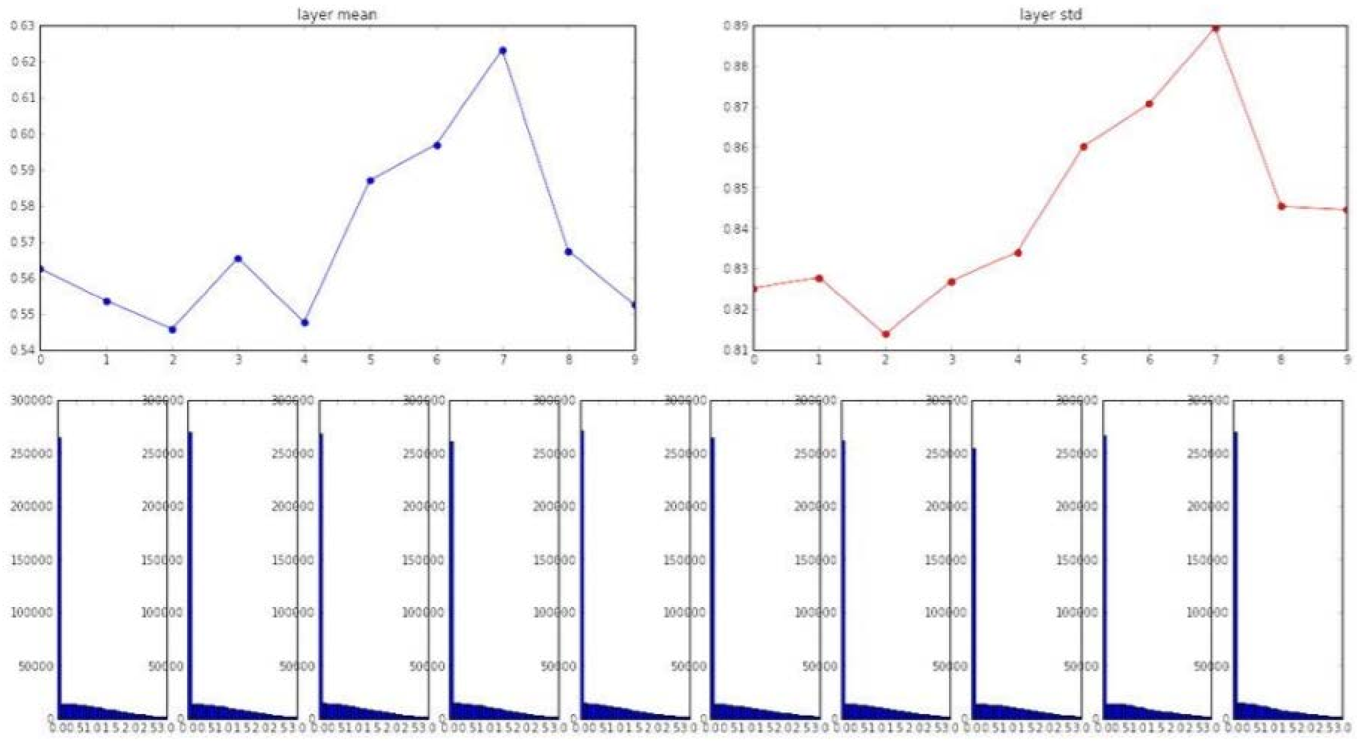
W = 0.01*np.random.randn(D, H) / np.sqrt((D+H)/2)



Interpretação e Correção de Resultados:

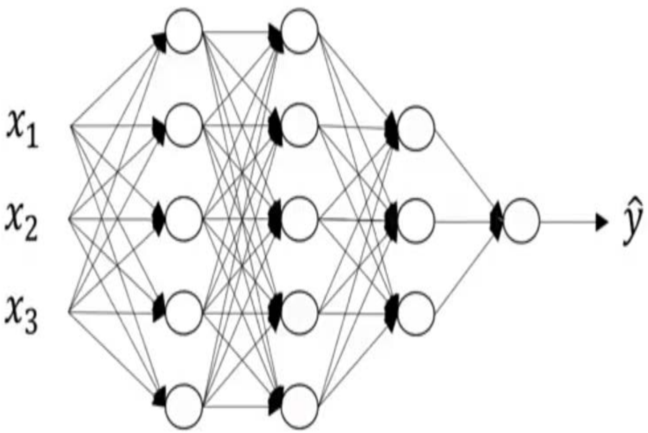
- Inicialização de Parâmetros
  - Kaiming He (2015):

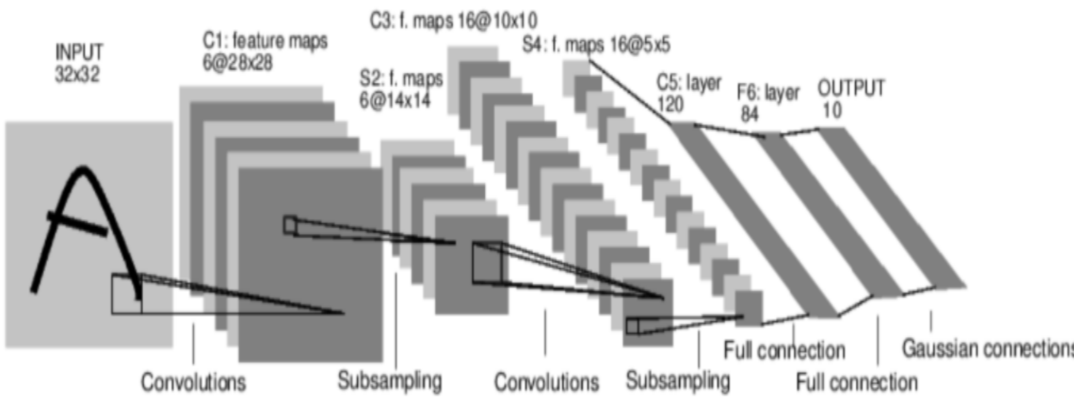
W = 0.01*np.random.randn(D, H) / np.sqrt(D/2)

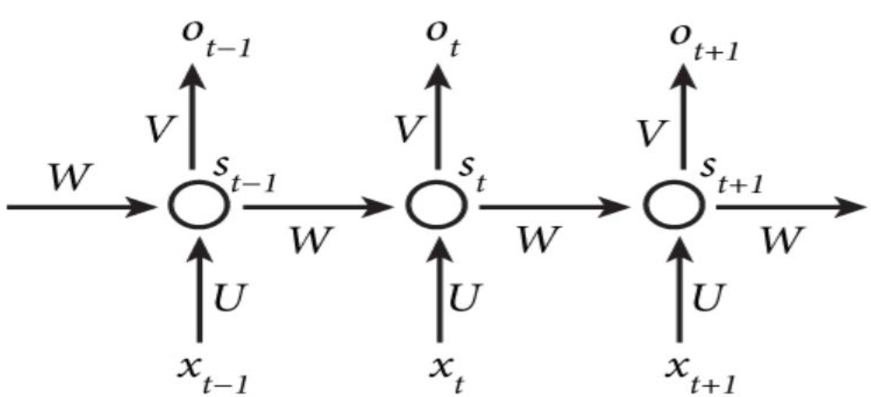


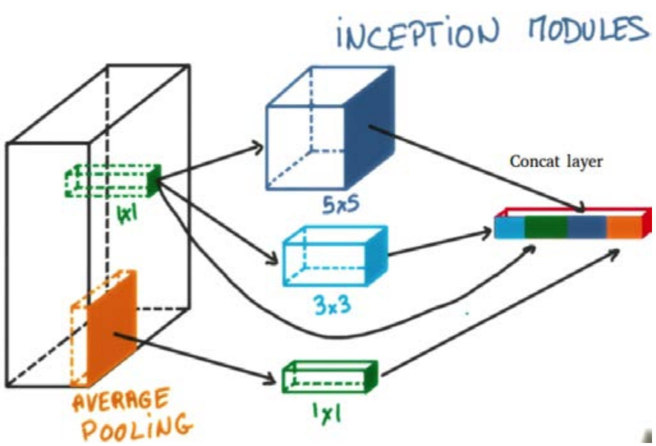
Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Escolha da Arquitetura e das Dimensões







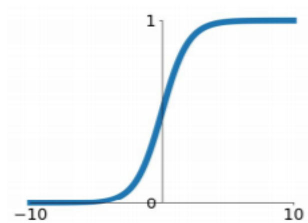


Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Função de Ativação

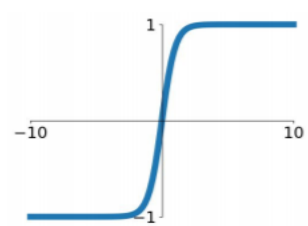
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



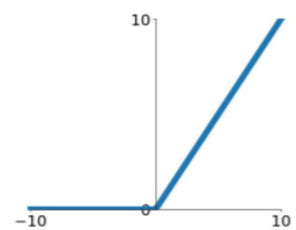
tanh

$\tanh(x)$



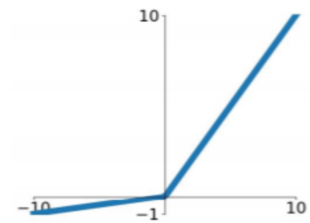
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

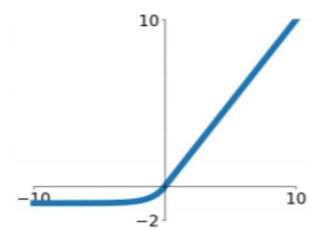


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Algoritmos de Otimização

  Stochastic Gradient Descent (SGD):

    - Batch < len(dataset)

  Gradient Descent com Momento

```
v = mu * v - learning_rate * dx # integrate velocity
x += v # integrate position
```

  Adagrad

```
cache += dx**2
x += - learning_rate * dx / (np.sqrt(cache) + eps)
```

Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Algoritmos de Otimização

RMSprop

```
cache = decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + eps)
```

Adam

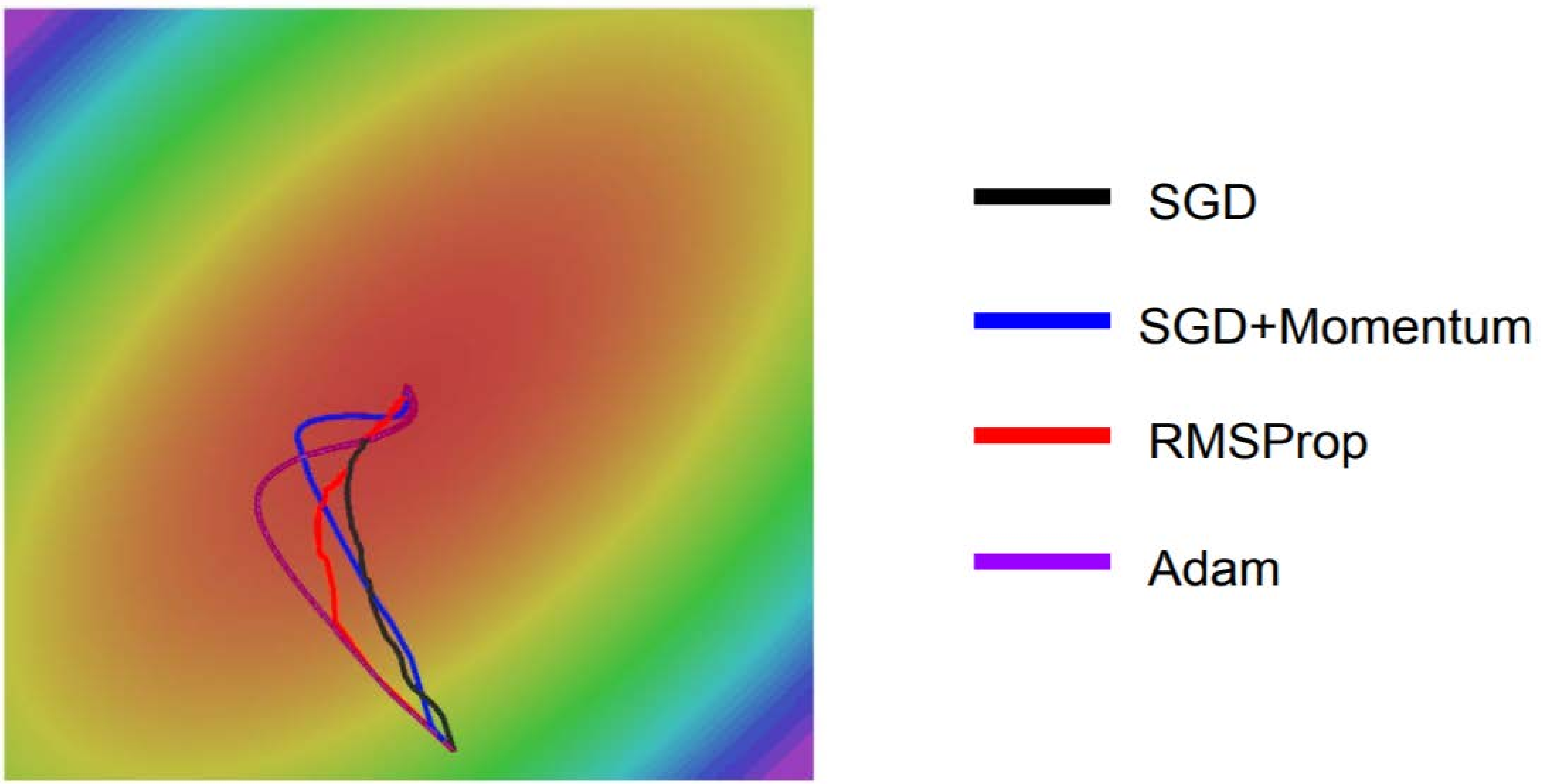
```
m = beta1*m + (1-beta1)*dx
v = beta2*v + (1-beta2)*(dx**2)
x += - learning_rate * m / (np.sqrt(v) + eps)
```

Métodos de 2ª ordem: BFGS e L-BFGS (mais comuns)

$$\boxed{\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \boldsymbol{H}^{-1}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)}$$

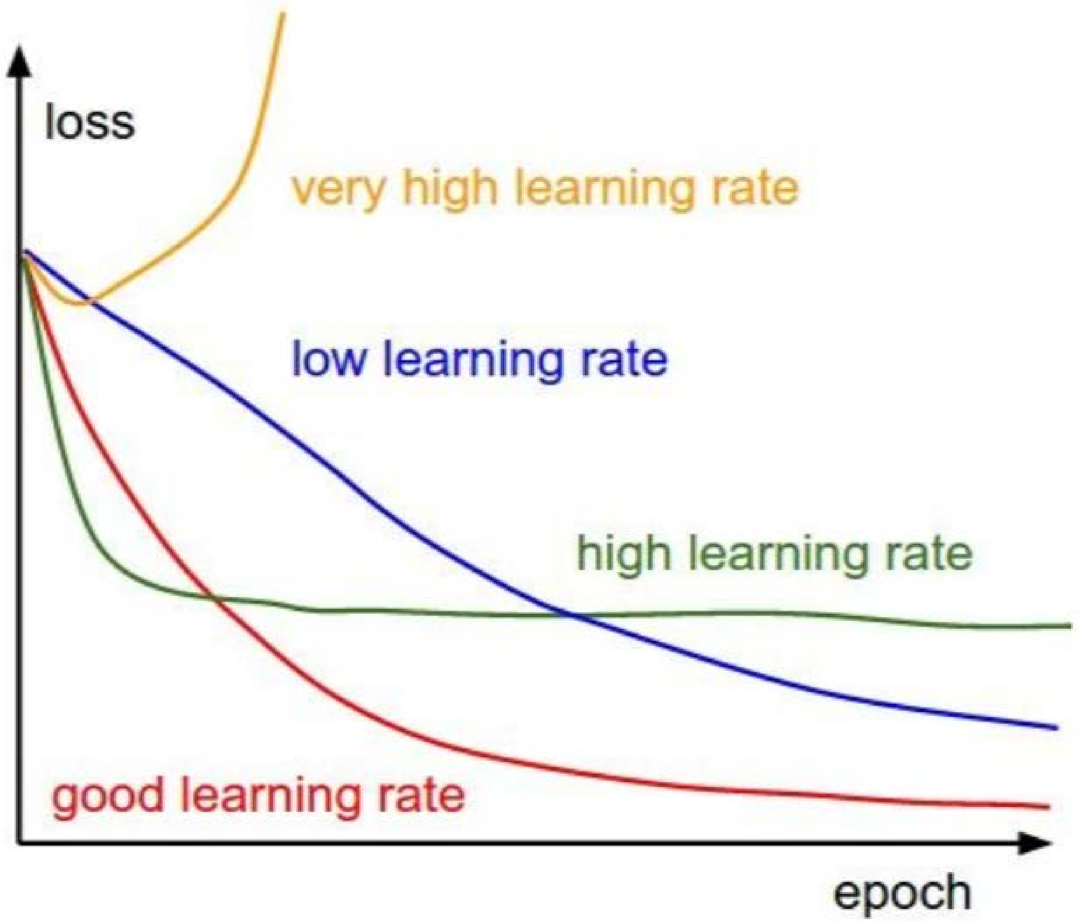
Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Algoritmos de Otimização



Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Taxa de Aprendizado



Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Regularização (weight-decay)

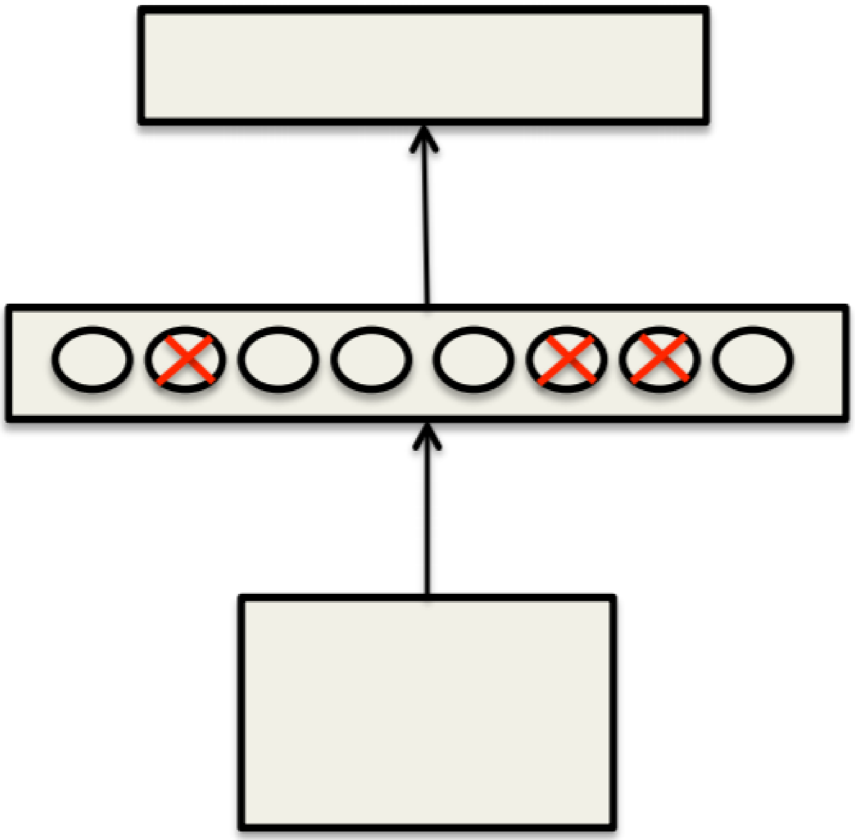
$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

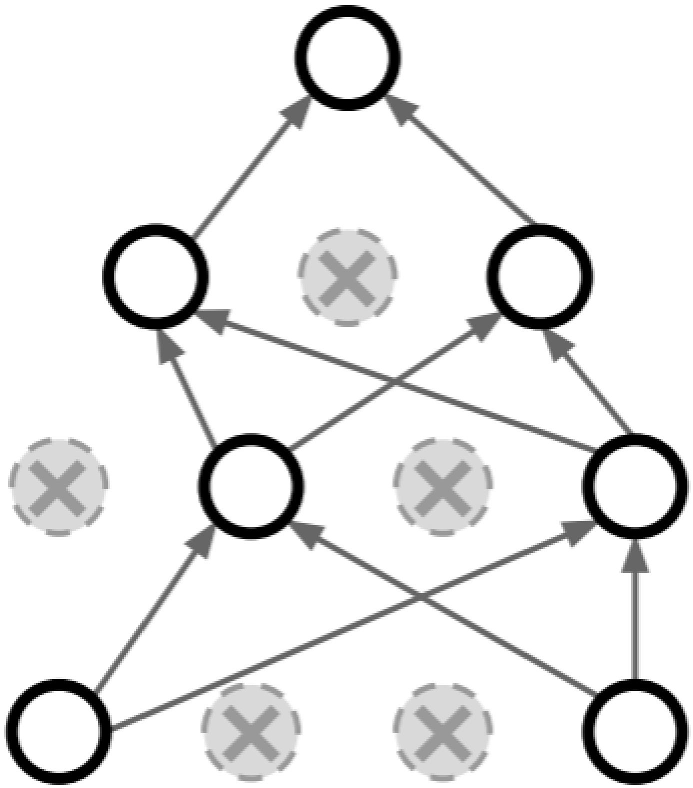
  - L2: $$R(W) = \sum_k \sum_l W_{k,l}^2$$

  - L1: $$R(W) = \sum_k \sum_l |W_{k,l}|$$

Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Dropout





Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Batch (Re)Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

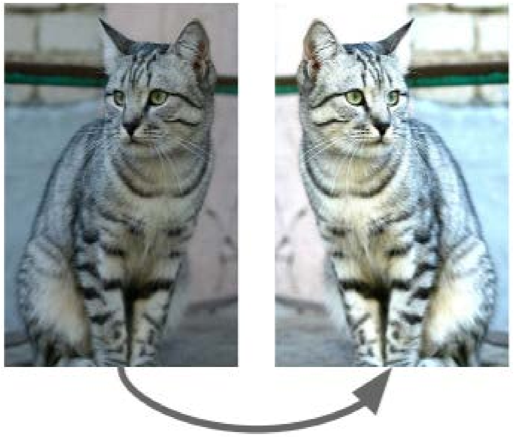
$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

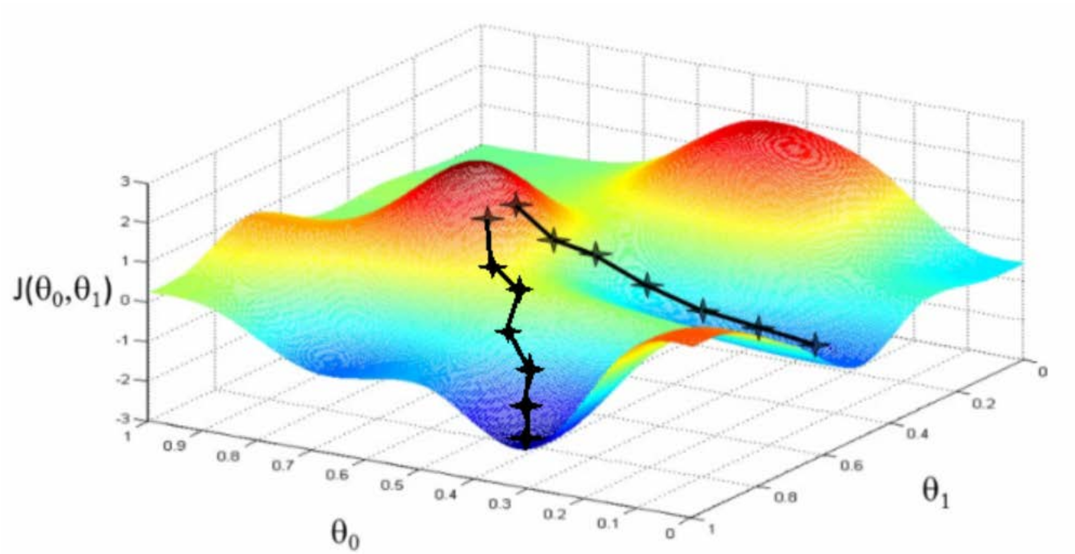
Interpretação e Correção de Resultados:

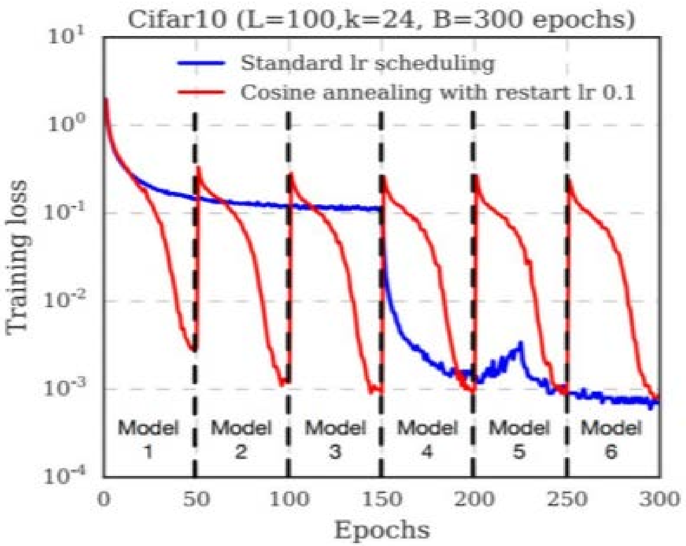
- Hyperparâmetros
  - Data Augmentation



Interpretação e Correção de Resultados:

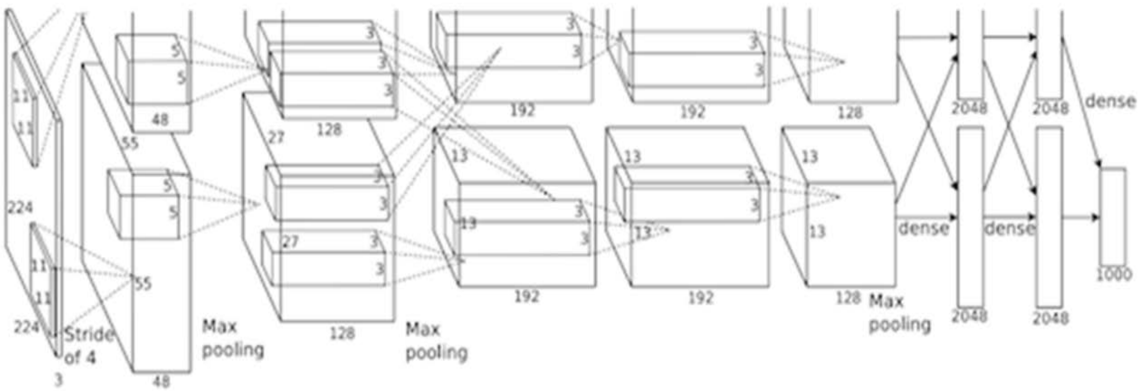
- Hyperparâmetros
  - Ensembles

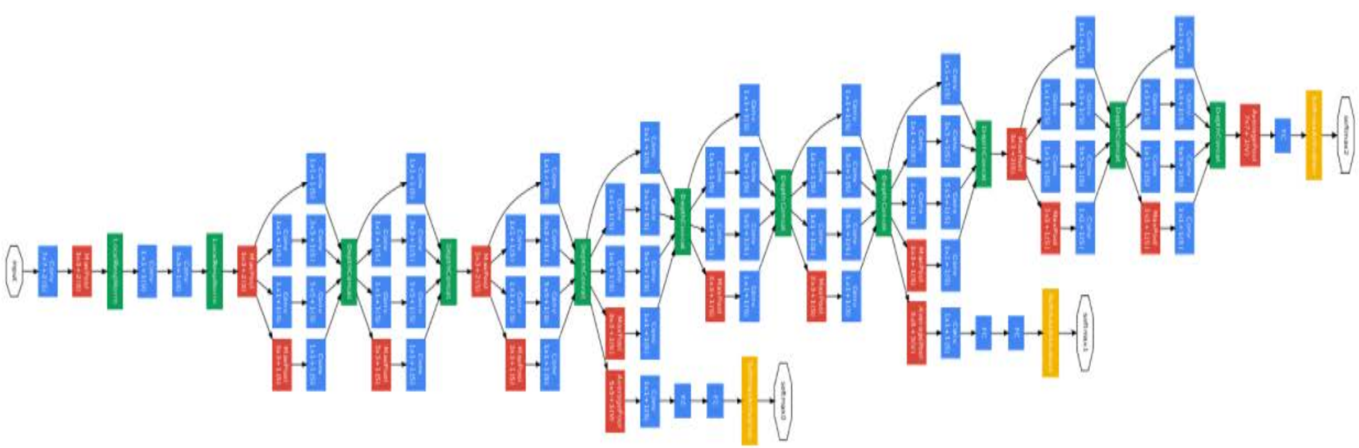


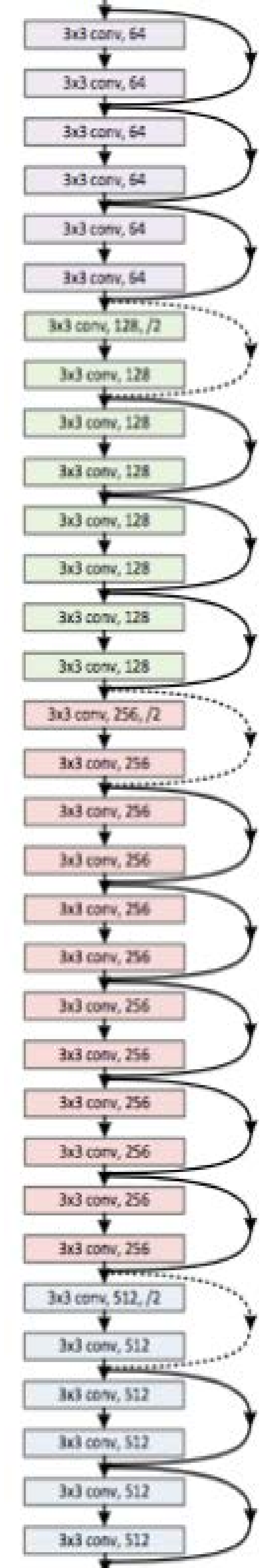


Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Transfer Learning







Interpretação e Correção de Resultados:

- Hyperparâmetros
  - Transfer Learning

	Datasets Similares	**Datasets Distintos**
Muitos dados disponíveis	Treine algumas camadas do modelo base e o classificador de saída	Treine um número maior de camadas (ou todas elas) da rede
Poucos dados disponíveis	Treine apenas o classificador de saída	É... Houston, we have a problem

RSNA 2017 – Pediatric Bone Age Challenge

Objetivo:

- Identificar idade óssea a partir de radiografias infantis.

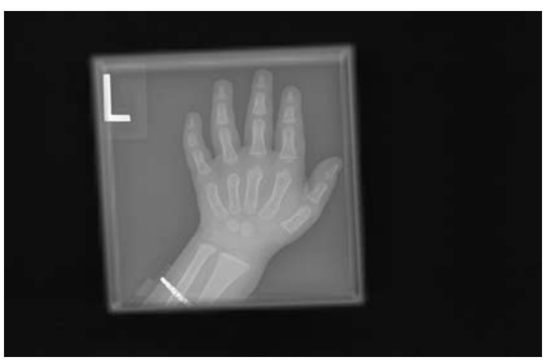
Solução proposta:

- Deep Learning

Time:

- UFG + Unifesp

Resumo do Problema



Entradas:

- Radiografia
- Sexo

Idades:

- 1 mês a 19 anos

Métrica:

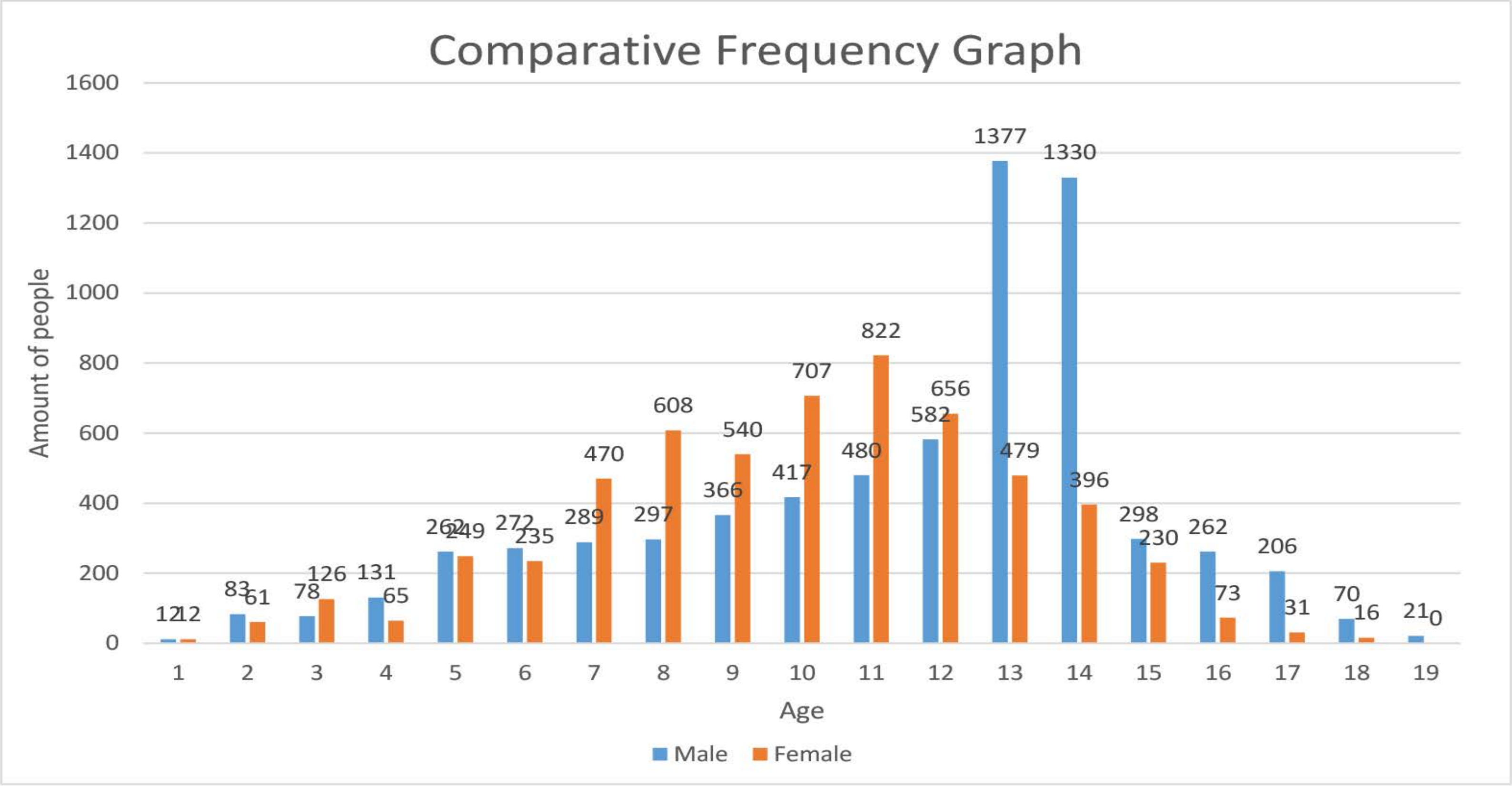
- Erro médio absoluto (MAE)

Restrição:

- Proibido usar dados privados ou redes pré-treinadas em dados privados

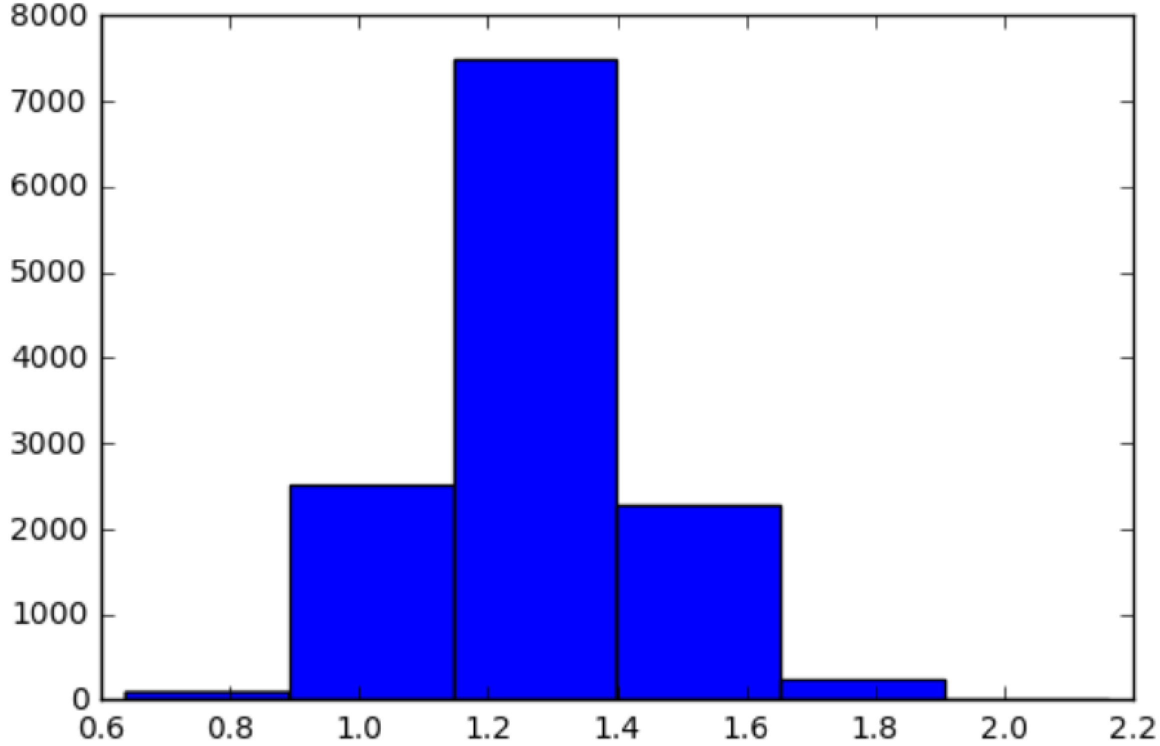
Exploração dos Dados

- Distribuição das radiografias em relação ao sexo



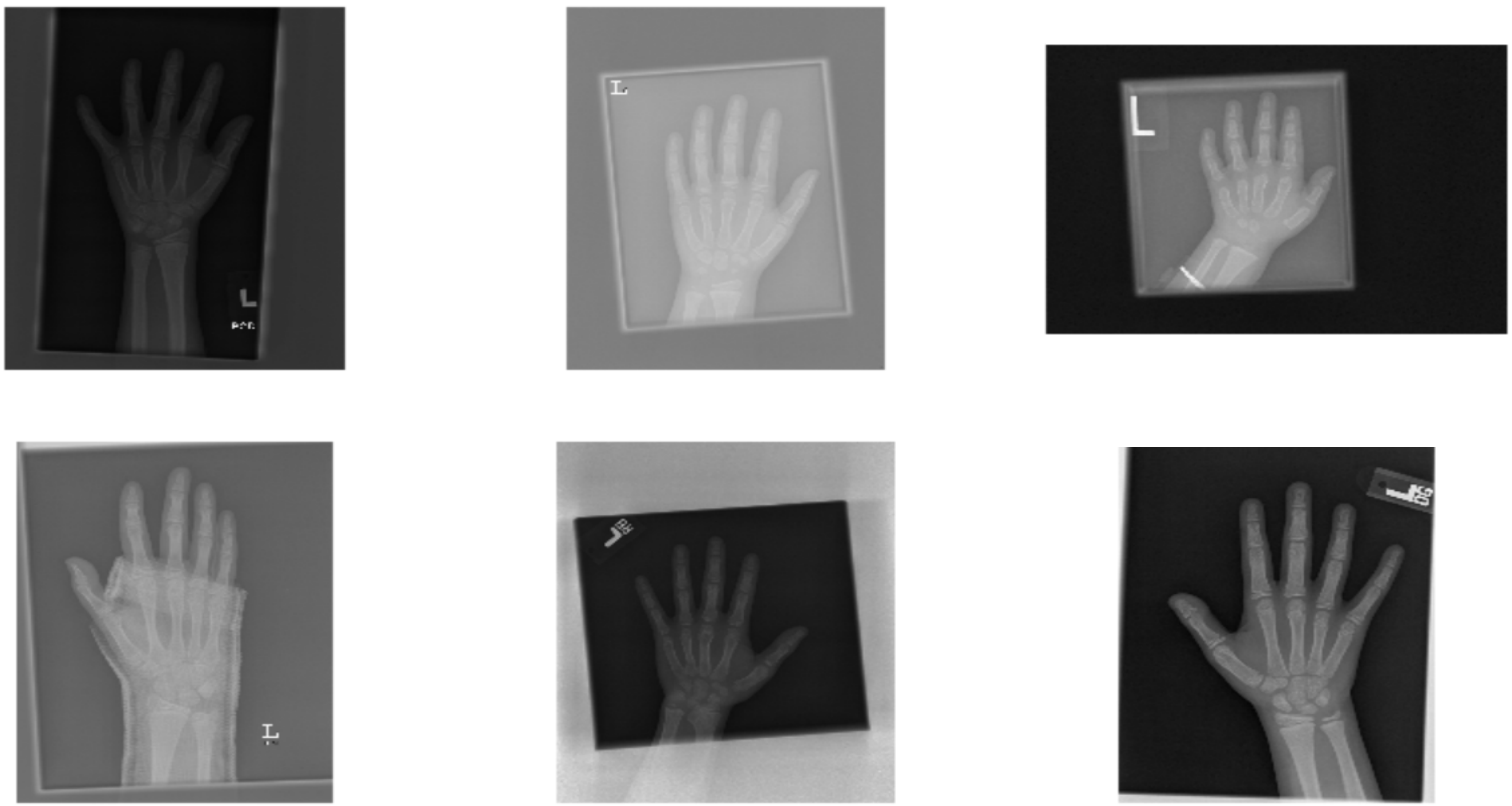
Exploração dos Dados

- Distribuição dos formatos (aspect ratio) das radiografias



Exploração dos Dados

- Variação de Brilho e Contraste entre as radiografias

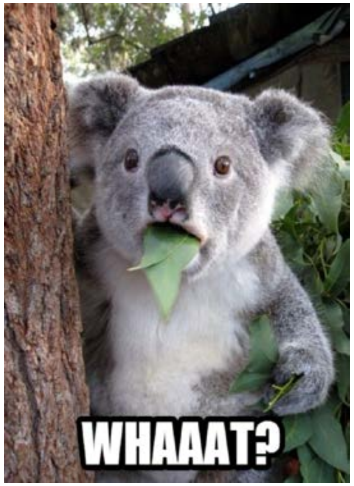


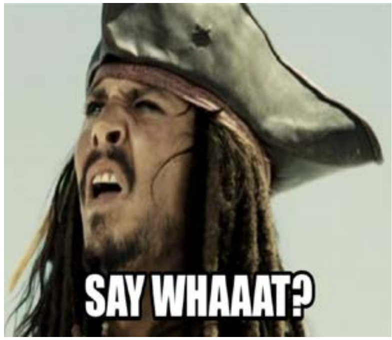
Exploração dos Dados

- Variação estrutural dos dados (ângulo das mãos, presença de outros objetos, presença de duas mãos, etc.)

Exploração dos Dados

- Exemplo inacreditável...





Função Custo e Métrica

- Erro Absoluto Médio

Learn the Details | Phases | Participate | Results | Forums | Team

- Overview
- **Evaluation**
- Terms and Conditions
- Organizers

Participants should submit a csv file with the case ID and the predicted age in months. A prediction should be provided for all cases in each phase. A zip file containing this csv file as well as a text file describing the algorithm is uploaded as the participant's submission. An example test submission is available here

The primary evaluation measure is Mean Absolute Distance (MAD), calculated as the mean of the absolute values of the difference between the model estimates and those of the reference standard bone age.

In case of ties, the concordance correlation coefficient (CCC) will be used to break ties.

Exploração de Arquiteturas

- Inception v4

Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning

Christian Szegedy
Google Inc.
1600 Amphitheatre Pkwy, Mountain View, CA
szegedy@google.com

Sergey Ioffe
sioffe@google.com

Vincent Vanhoucke
vanhoucke@google.com

Alex Alemi
alemi@google.com

Abstract

Very deep convolutional networks have been central to the largest advances in image recognition performance in recent years. One example is the Inception architecture that has been shown to achieve very good performance at relatively low computational cost. Recently, the introduction of residual connections in conjunction with a more traditional architecture has yielded state-of-the-art performance in the 2015 ILSVRC challenge; its performance was similar to the latest generation Inception-v3 network. This raises the question of whether there are any benefit in combining the Inception architecture with residual connections. Here we give clear empirical evidence that training with residual connections accelerates the training of Inception networks significantly. There is also some evidence of residual Inception networks outperforming similarly expensive Inception networks without residual connections by a thin margin. We also present several new streamlined architectures for both residual and non-residual Inception networks. These variations improve the single-frame recognition performance on the ILSVRC 2012 classification task significantly. We further demonstrate how proper activation scaling stabilizes the training of very wide residual Inception networks. With

tion [7], object tracking [18], and superresolution [3]. These examples are but a few of all the applications to which deep convolutional networks have been very successfully applied ever since.

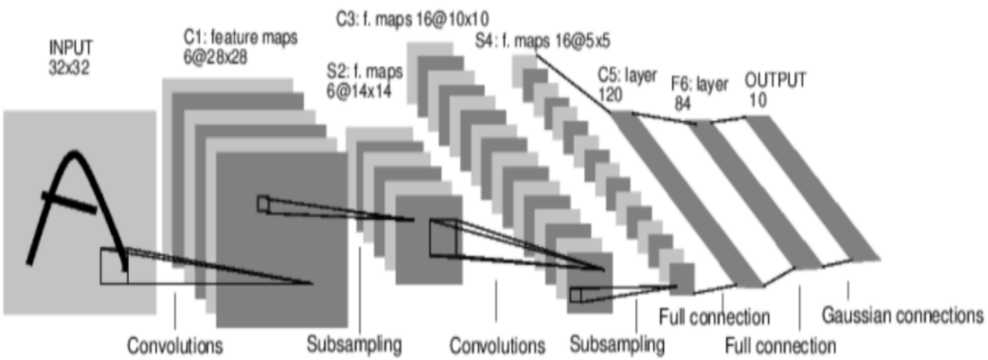
In this work we study the combination of the two most recent ideas: Residual connections introduced by He et al. in [5] and the latest revised version of the Inception architecture [15]. In [5], it is argued that residual connections are of inherent importance for training very deep architectures. Since Inception networks tend to be very deep, it is natural to replace the filter concatenation stage of the Inception architecture with residual connections. This would allow Inception to reap all the benefits of the residual approach while retaining its computational efficiency.

Besides a straightforward integration, we have also studied whether Inception itself can be made more efficient by making it deeper and wider. For that purpose, we designed a new version named Inception-v4 which has a more uniform simplified architecture and more inception modules than Inception-v3. Historically, Inception-v3 had inherited a lot of the baggage of the earlier incarnations. The technical constraints chiefly came from the need for partitioning the model for distributed training using DistBelief [2]. Now, after migrating our training setup to TensorFlow [1] these

```
from keras.models import Sequential
from keras import optimizers
from keras.layers.core import Dense, Dropout
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPool2D, GlobalAveragePooling2D
from keras.models import Model
from keras.layers import Input, Concatenate
import inception_v4

# define cnn model
def inceptionv4():

    #Carrega Incpetion v4
    model = inception_v4.create_model(include_top=False)

    return model

imgs = Input(shape=(598,598,1))
gender = Input(shape=(1,))

x = Conv2D(3, 3, padding='same', activation='relu')(imgs)
x = MaxPool2D()(x)
x = inceptionv4()(x)
x = GlobalAveragePooling2D()(x)
x = Concatenate(axis=-1)([x,gender])
x = Dense(32, activation='relu')(x)
outputs = Dense(1)(x)
inputs = [imgs, gender]
icptv4_model = Model(inputs=inputs, outputs=outputs)

# escolha o modelo a ser treinado:
model = icptv4_model
#model.summary()
```

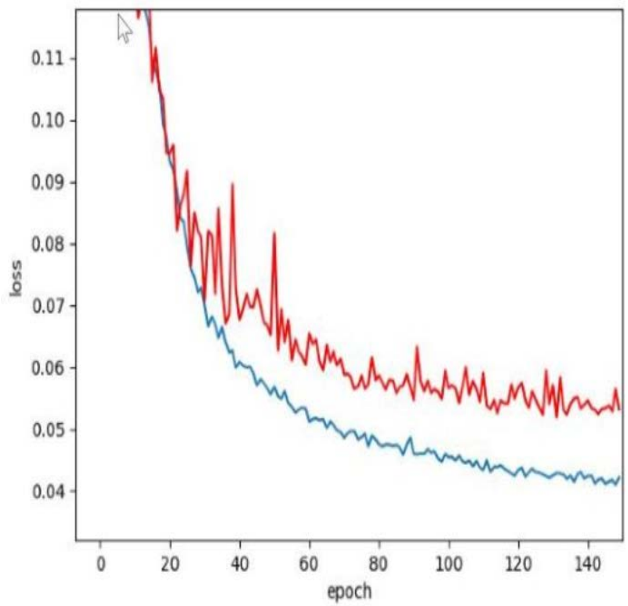
Exploração de Arquiteturas

- Variação da LeNet

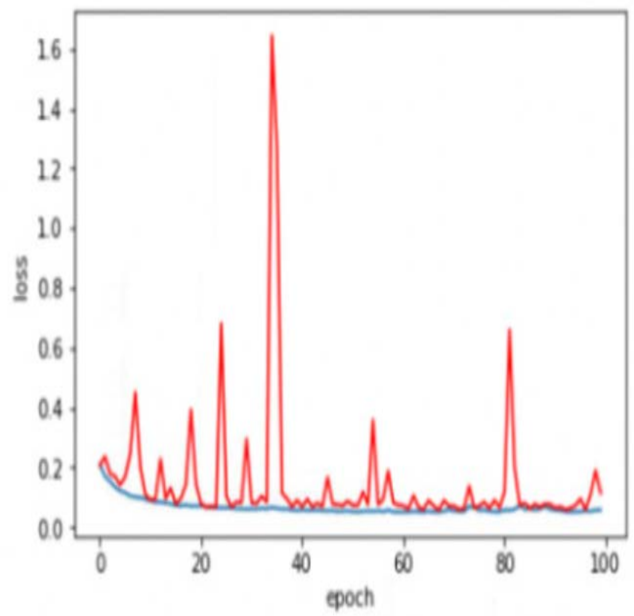



```python
from keras.models import Sequential
from keras import optimizers
from keras.layers.core import Dense, Dropout
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPool2D, GlobalAveragePooling2D
from keras.models import Model
from keras.layers import Input, Concatenate

# define cnn model
cnn_model = Sequential()
cnn_model.add(Conv2D(16, 3, input_shape=(550, 550, 1), padding='same', activation='relu'))
cnn_model.add(MaxPool2D())
cnn_model.add(Conv2D(32, 3, padding='same', activation='relu'))
cnn_model.add(MaxPool2D())
cnn_model.add(Dropout(0.5))
cnn_model.add(Conv2D(32, 3, padding='same', activation='relu'))
cnn_model.add(MaxPool2D())
cnn_model.add(Conv2D(64, 3, padding='same', activation='relu'))
cnn_model.add(MaxPool2D())
cnn_model.add(Conv2D(64, 3, padding='same', activation='relu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(GlobalAveragePooling2D(name='ft_x_layer'))
cnn_model.add(Dense(64, activation='relu'))
cnn_model.add(Dense(1))

# define ft. extractor + gender based on cnn model
base_model = Model(inputs=cnn_model.input, outputs=cnn_model.get_layer('ft_x_layer').output)
imgs = base_model.input
gender = Input(shape=(1,))
x = base_model.output
x = Concatenate(axis=-1)([x,gender])
x = Dense(32, activation='relu')(x)
outputs = Dense(1)(x)
inputs = [imgs, gender]
ft_x_cnn_model = Model(inputs=inputs, outputs=outputs)

# escolha o modelo a ser treinado:
model = ft_x_cnn_model
#model.summary()
```

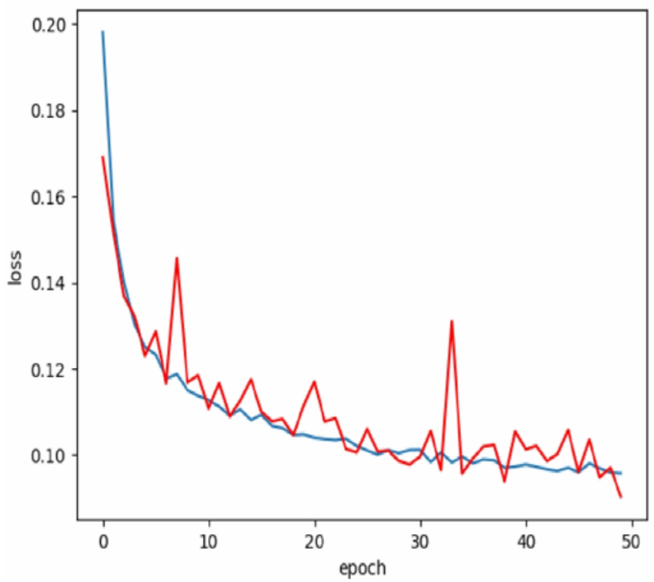
Resultados



0.11
0.10
0.09
0.08
0.07
0.06
0.05
0.04
loss
0
20
40
60
80
100
120
140
epoch

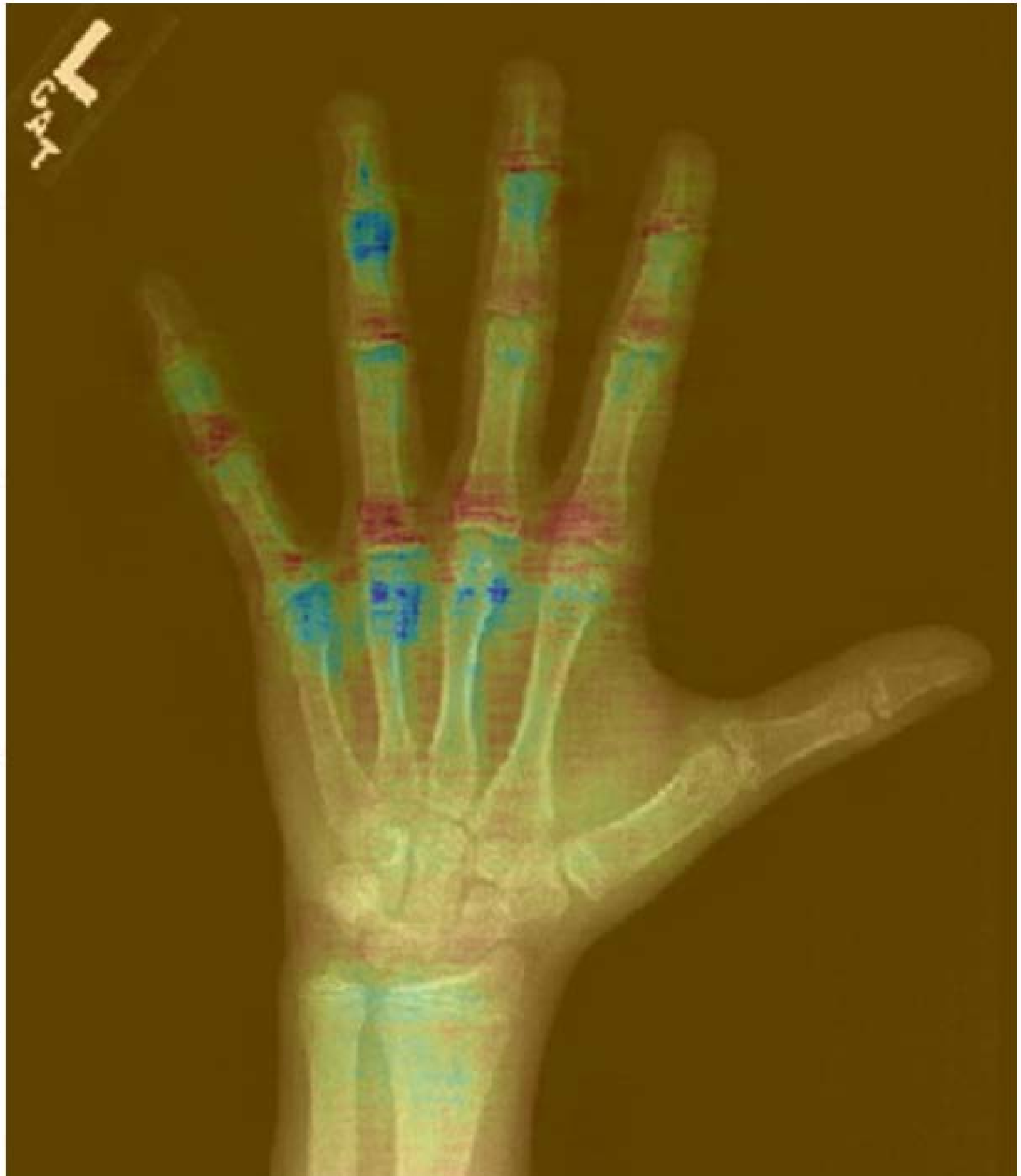


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
loss
0
20
40
60
80
100
epoch

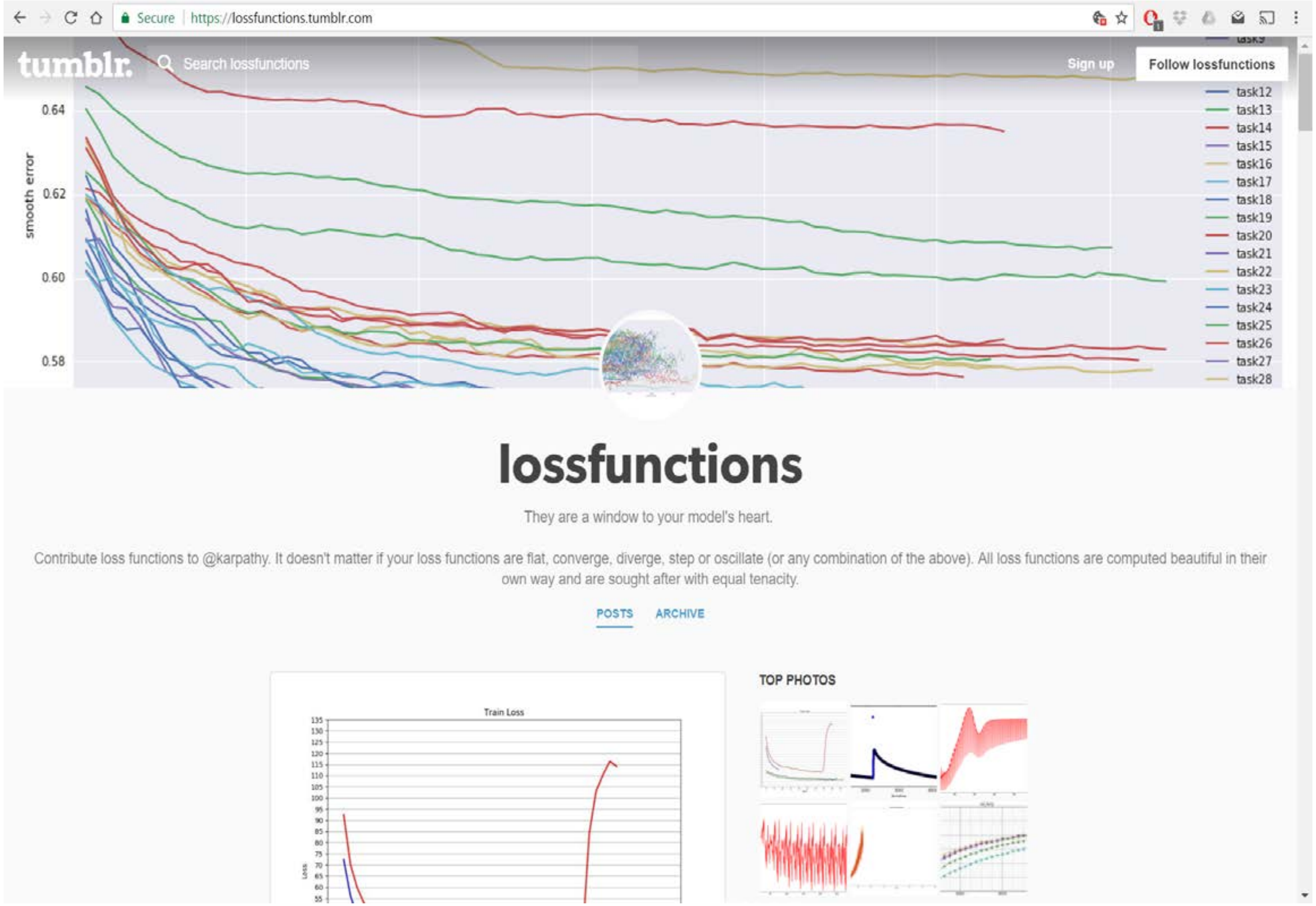


0.20
0.18
0.16
0.14
0.12
0.10
loss
0
10
20
30
40
50
epoch

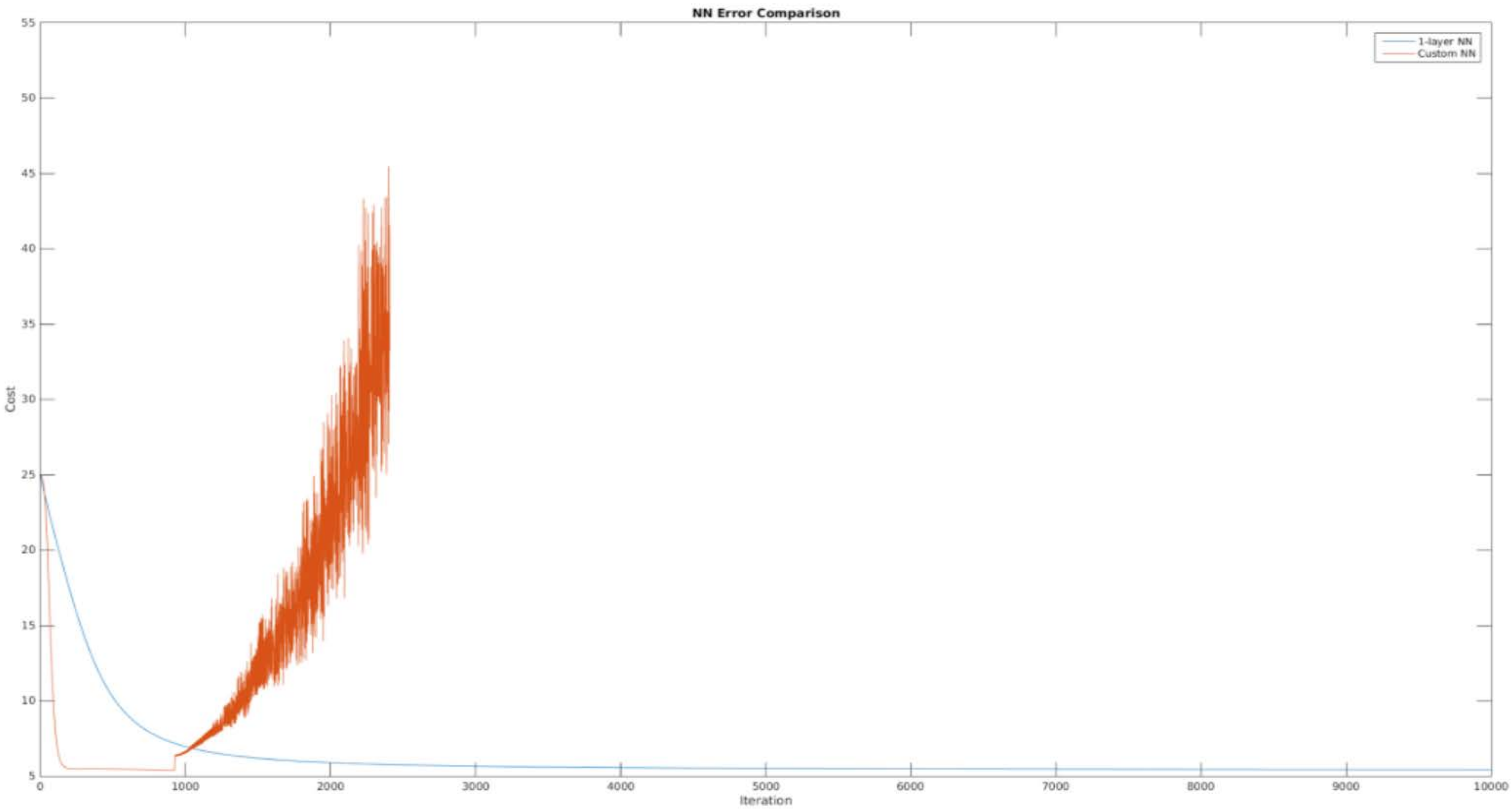
Resultados



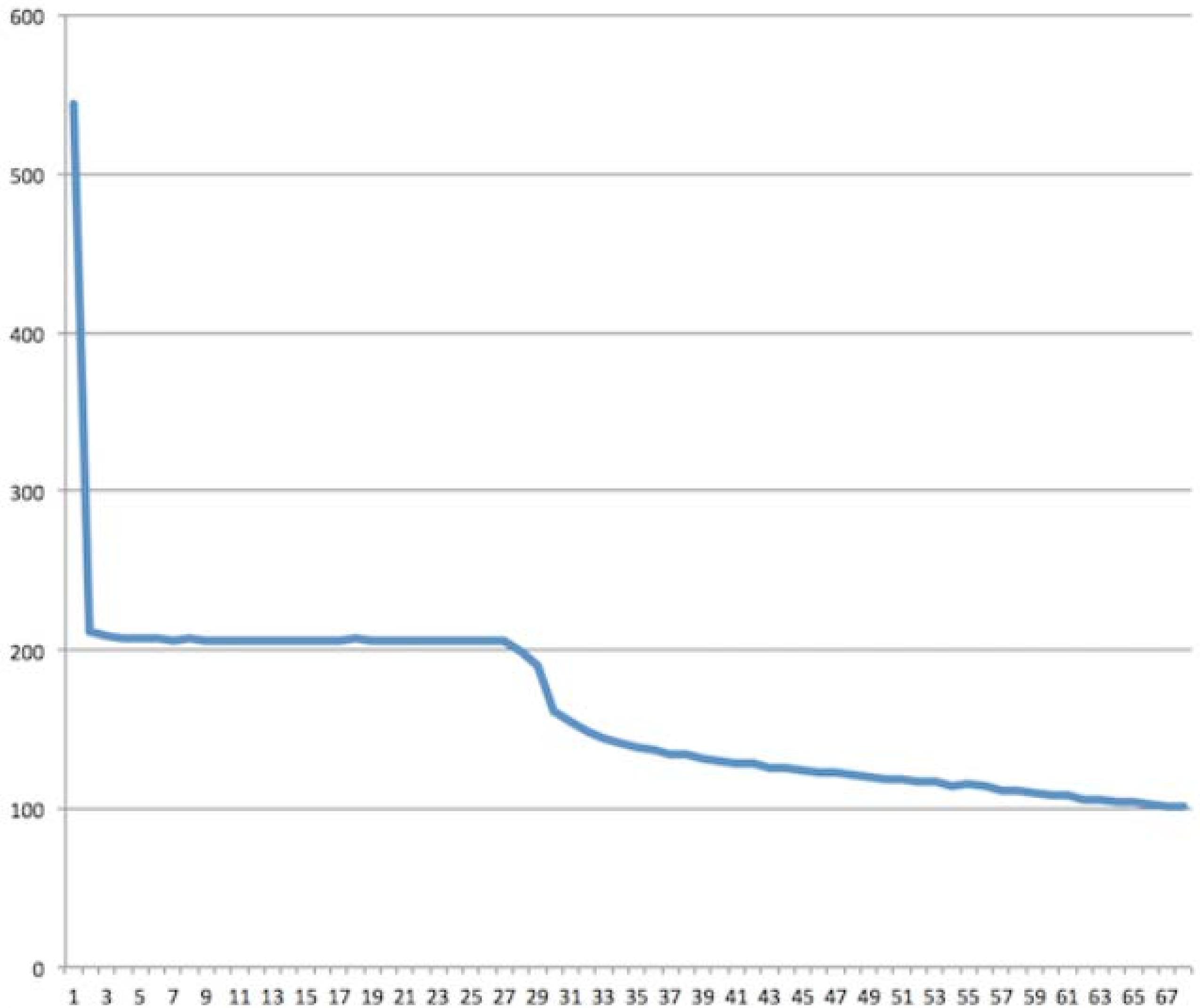
Curiosidades



Curiosidades



Curiosidades



Curiosidades