

Detecção de falhas usando Redes Neurais

RUBEN DARIO BENITES PEREZ
JOAQUIN EDUARDO FIGUEROA BARRAZA
LUIS FELIPE GUARDA BRAUNING
VICTOR RAFAEL LIMA SOUZA

ESCOLA POLITÉCNICA – UNIVERSIDADE DE SÃO PAULO
PSI5886 – Princípios de Neurocomputação
PROF. DR. EMÍLIO DEL MORAL HERNANDEZ
DEZEMBRO DE 2018

Membros da equipe



Luis Felipe
Guarda
Brauning



Ruben Dario
Benites
Perez



Joaquin
Eduardo
Figueroa
Barraza



Victor Rafael
Lima Souza

Conteúdo

- Introdução
- Dados: MFPT
- Ferramenta: Deep Learning Studio
- Metodologia/Implementação
- Resultados
- Conclusões
-

Introdução

- CBM (Condition Based Maintenance) [1,2]
 - Detectar iminência de falha (antes de sua ocorrência)
 - Manutenção preventiva, apenas se necessário
 - Ensaio não-destrutivo
 - Obtenção de dados durante operação
- Uso de redes tipo MLP para classificação do estado dos rolamentos [3,4,5,6]:
 - Sem falhas
 - Falha na pista externa
 - Falha na pista interna).
- Dados de entrada [7,8,9]:
 - medições de vibração (aceleração)

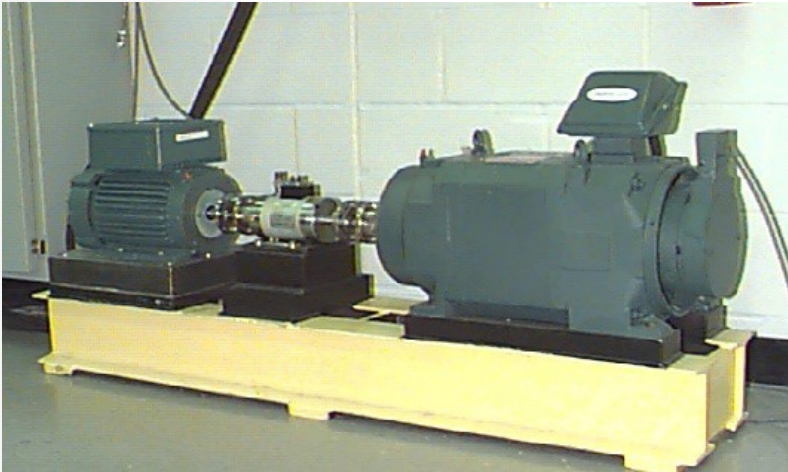
Dados: MFPT

- Society for Machinery Failure Prevention Technology.
- Dados de vibração em rolamentos (aceleração):
 - Data Set da Society for Machinery Failure Prevention Technology [9]
- Três tipos de condições
 - Rolamentos sem falhas
 - Com falha na pista externa
 - Com falha na pista interna
- Dados de entradas:
 - Blocos de 640 frames do dataset [9]

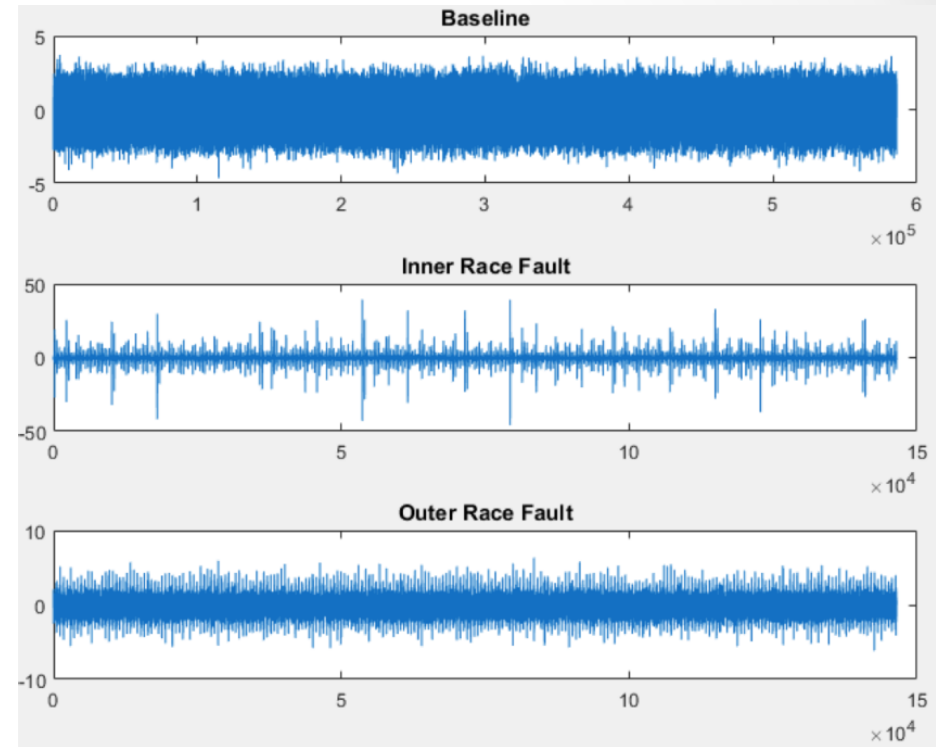


Fonte: Catálogo de revendedor [10]

Dados: MFPT



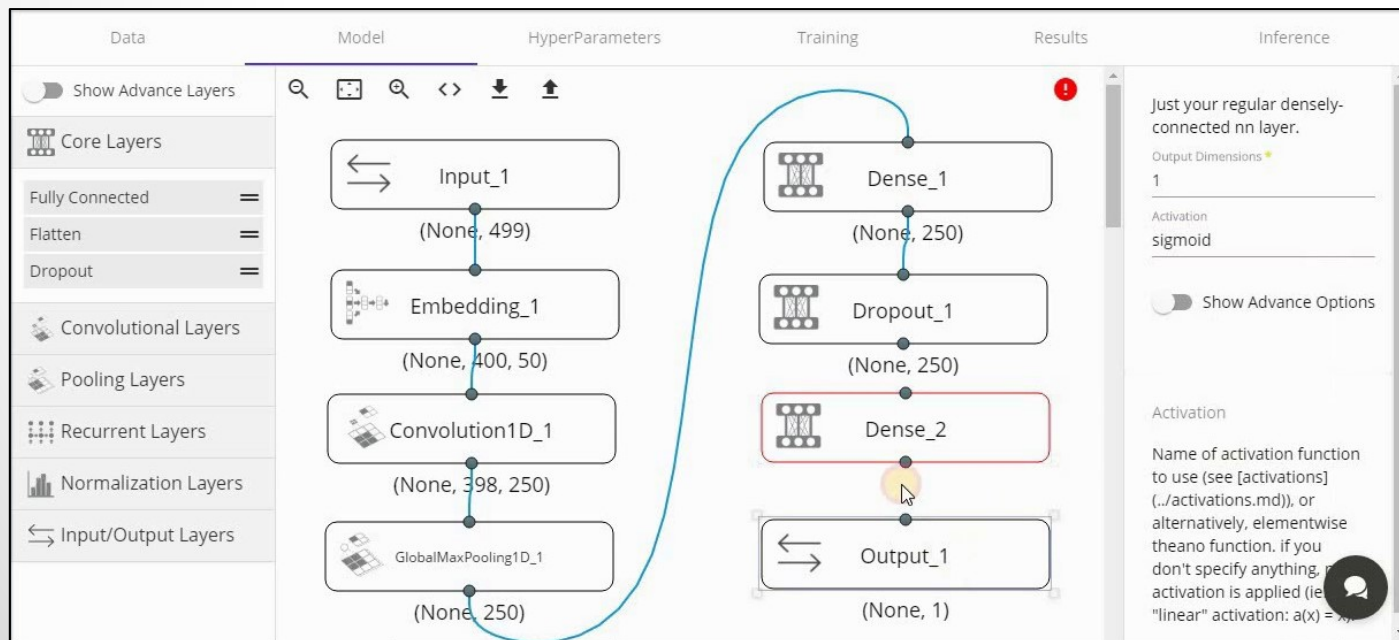
Aparato de teste. Fonte: [11]



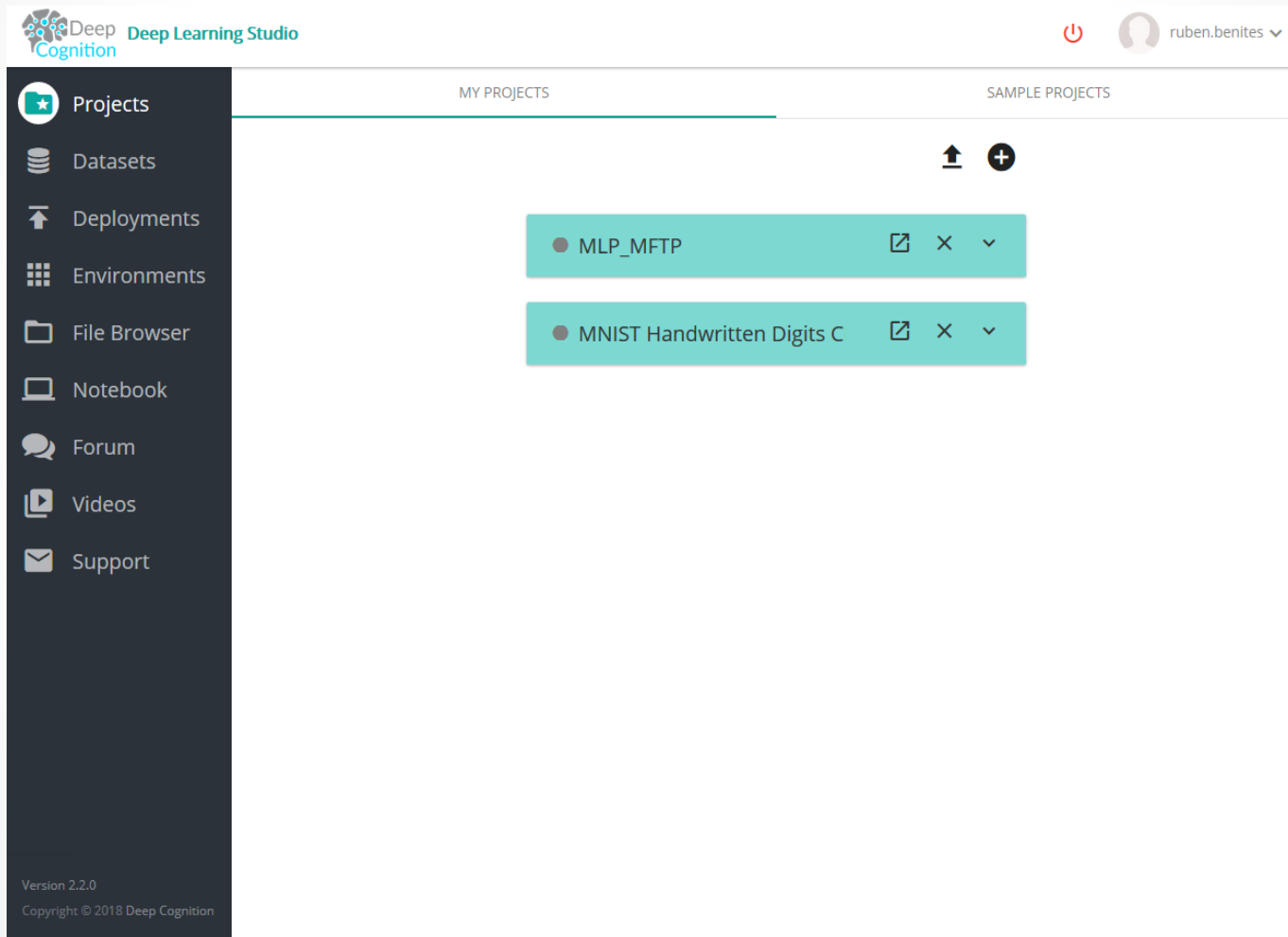
Fonte: (Autores, 2018), utilizando dados de [9]

Deep Learning Studio [12]

- Software para criação de modelos de Machine Learning e Deep Learning.
- Não precisa de muito conhecimento de programação.
- Fácil manipulação da rede e os parâmetros dela.
- Compatibilidade com Tensorflow e Keras.
- Possibilidade de trabalhar com CPU e GPU.



Tutorial Deep Learning Studio



Data Model HyperParameters Training Results Inference/Deploy

Dataset	Access	Train / Validation / Test Split	Training	Validation	Test
MFPT_R_train	- private	0%	4846	1211	0
Caltech101	- public				
cifar-10	- public				
imdb	- public				
iris	- public				
Kaggle_IDC_Cancer	- public				

Memory at a time Shuffle Data

data target

Data Type: Array Numeric

Input or Output: InputPort0 InputPort0

0.0319;-0.0742;-0.0447;0.0154;-0.0347;-0.0321;0.1615;-0.0046;-0....	2
-0.0106;-0.0340;-0.0158;-0.0265;-0.0581;-0.1382;0.1362;0.1119;-0...	0
-0.0725;-0.0959;-0.0325;-0.1265;-0.0812;-0.0592;0.1840;0.1690;-0...	1
-0.0245;-0.0085;-0.0292;-0.0416;-0.0523;-0.0857;0.2264;0.0868;-0...	1
-0.0140;-0.0374;0.0191;-0.0312;-0.0091;-0.1038;0.1940;0.1273;-0....	1
-0.0101;-0.0359;0.0619;-0.0083;-0.0649;-0.0320;0.2589;0.0896;-0....	0
0.1983;0.0226;-0.0356;-0.2040;-0.1351;-0.1169;0.2152;-0.0545;-0....	2
-0.0323;0.0785;-0.0309;-0.0261;-0.0122;-0.0946;0.1176;0.0325;-0....	1

Click on column buttons to configure column specific preprocessing options

Fonte: Autores, 2018

Deep Learning Studio NeuroComp ruben.benites

Data Model HyperParameters Training Results Inference/Deploy

Advance Layers +

Search

↔ Input/Output Layers

Core Layers

- Dense =
- Dropout =
- Flatten =

Convolutional Layers

Pooling Layers

Recurrent Layers

Normalization Layers

⏏ 🔍 🔍 <> ⬇️ ⬆️ ⌛ ✂️

```

graph TD
    Input_2[Input_2] -- "(None, 641)" --> Dense_2[Dense_2]
    Dense_2 -- "(None, 1024)" --> Dense_4[Dense_4]
    Dense_4 -- "(None, 256)" --> Dense_3[Dense_3]
    Dense_3 -- "(None, 3)" --> Output_1[Output_1]
  
```

! Just your regular densely-connected nn layer.

Output Dimensions *
3

Activation

- softmax
- softplus
- softsign
- relu
- tanh
- sigmoid

activation is applied (ie. "linear" activation: $a(x) = x$).

Fonte: Autores, 2018

Data

Model

HyperParameters

Training

Results

Inference/Deploy



Show Advanced Parameters ✓

Number of Epoch

10

Batch Size

32

Loss Function

categorical_crossentropy

Optimizer

Adadelta

lr

1

epsilon

1e-08

decay

0

rho

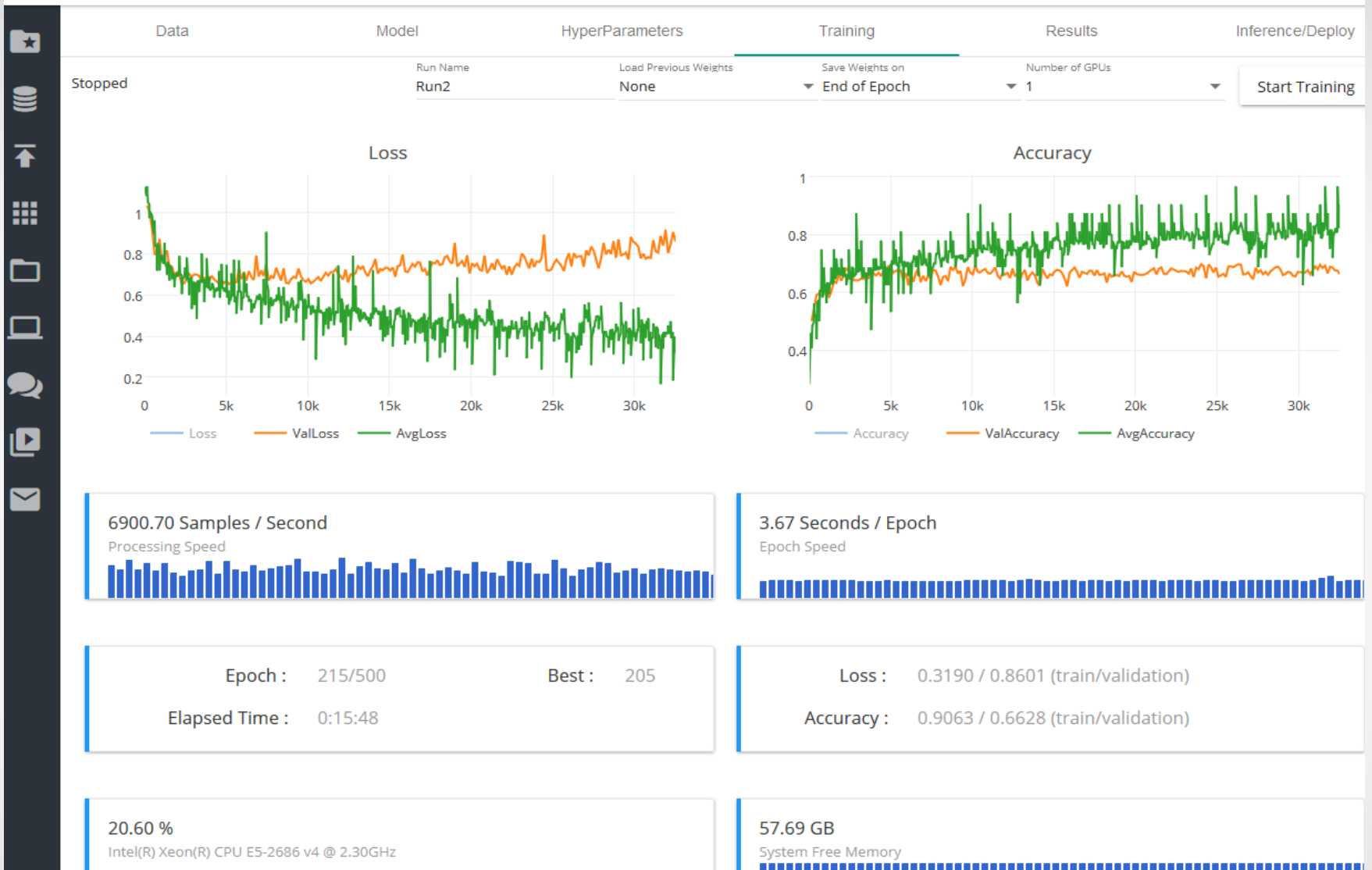
0.95

Float >= 0. learning rate. it is recommended to leave it at the default value.

Float >= 0. fuzz factor.

Float >= 0. learning rate decay over each update.

Float >= 0.



Fonte: Autores, 2018

Metodologia e Implementação

- Análise de arquiteturas e hiperparâmetros para uma MLP
- Estudo da influência de diferentes fatores:
 - Número de camadas
 - Número de neurônios por camada
 - Dropout
 - Função de ativação
 - Otimizador
 - Taxa de aprendizado
- Uso do software Deep Learning Studio [12].

Metodologia

Numero de camadas e neurônios

7 Arquiteturas iniciais

5 Rodadas de simulações

Numero de camadas	Numero de neurônios por camada
1	256
	512
	1024
2	521-1024
	1024-512
3	256-512-1024
	1024-512-256

Metodologia

- Primeira rodada de simulações:

- Todas as 7 MLPs
- Função de ativação: ReLu
- Sem Dropout
- Otimizador: Adam
- Taxa de aprendizado: $1e-3$

- Melhores resultados

destacados em amarelo:

Camadas	Neurônios	Validation Accuracy
1	256	0,759
	512	0,805
	1024	0,813
2	512-1024	0,745
	1024-512	0,7549
3	256-512-1024	0,739
	1024-512-256	0,736

Metodologia

- Segunda rodada de simulações:
 - Melhor MLP para cada número de camadas da rodada anterior
 - **Dropout 20% e 50% (total de 6 MLPs)**
 - Função de ativação: ReLu
 - Otimizador: Adam
 - Taxa de aprendizado: $1e-3$

- Melhor resultado
destacado em amarelo:

Arquitetura	DropOut	Validation Accuracy
1024	0,5	0,813
1024-512	0,5	0,726
256-512-1024	0,5	0,64
1024	0,2	0,8215
1024-512	0,2	0,746
256-512-1024	0,2	0,6422

Metodologia



- **ReLu****
- **Sigmoid**
- **Tanh**

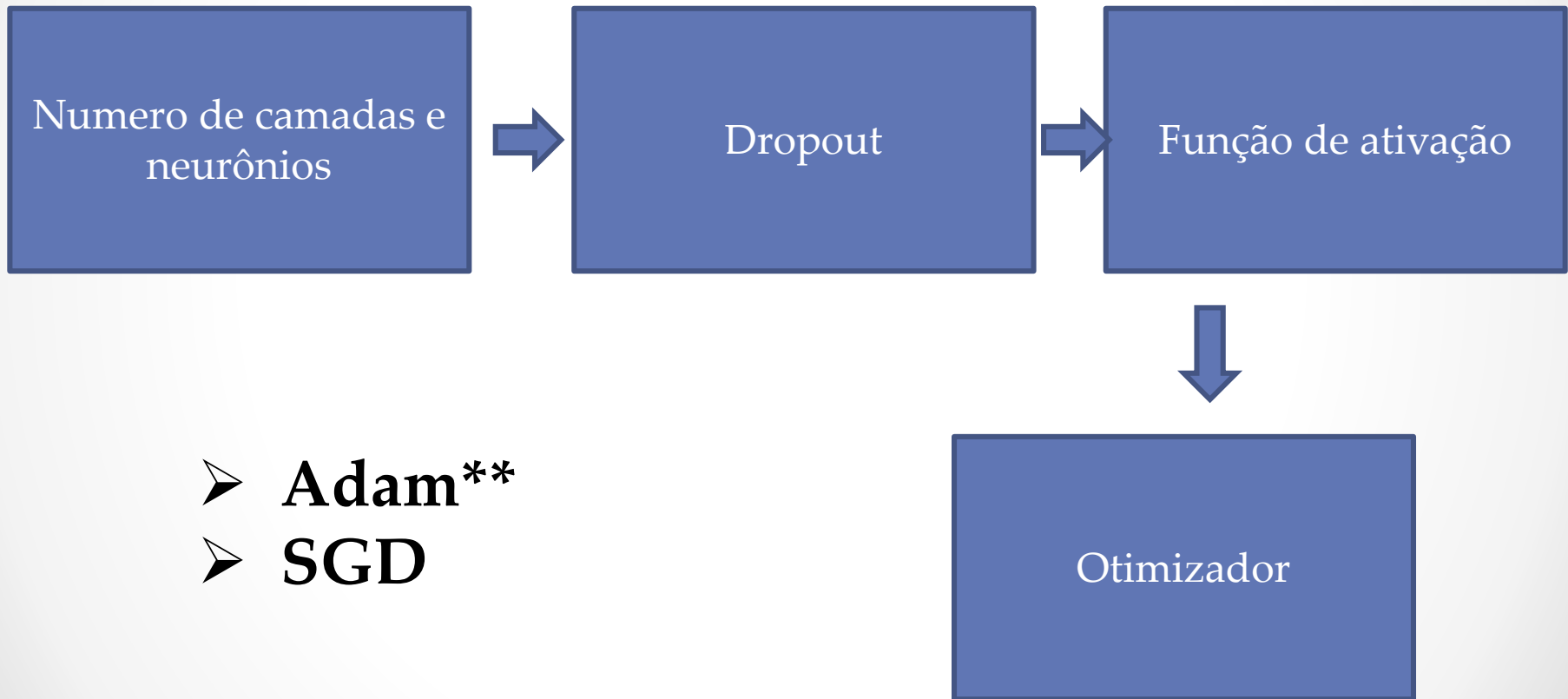
Metodologia

- Terceira rodada de simulações:
 - Melhor MLP: 1 camada com 1024 neurônios
 - Sem Dropout
 - **Função de ativação: Sigmoid e Tangente Hiperbólica (2 MLPs)**
 - Otimizador: Adam
 - Taxa de aprendizado: $1e-3$

Função	Validation Accuracy
tanh	0,781
sigmoid	0,741

- Resultados piores que os obtidos com ReLu:

Metodologia



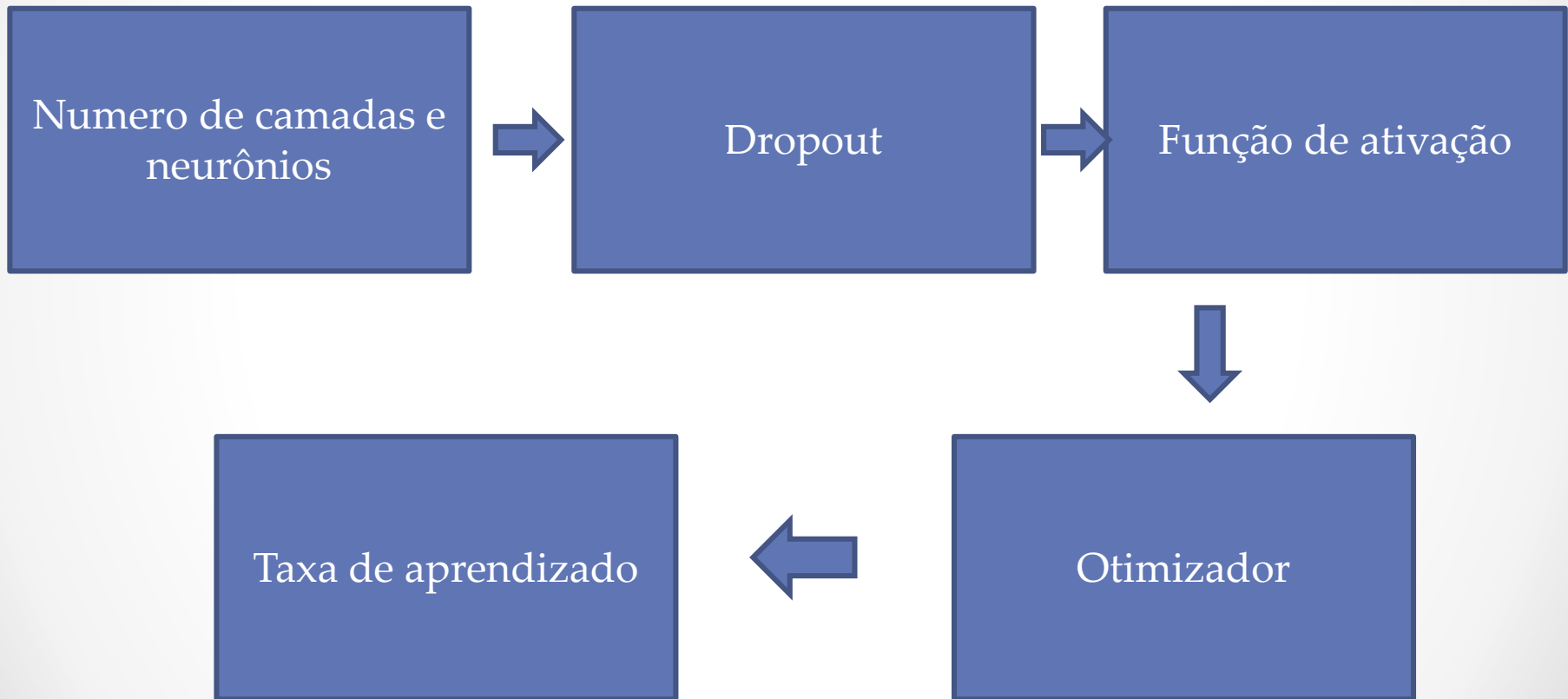
Metodologia

- Quarta rodada de simulações:
 - Melhor MLP: 1 camada com 1024 neurônios
 - Sem Dropout
 - Função de ativação: ReLu
 - **Otimizador: SGD**
 - Taxa de aprendizado: $1e-3$

- Resultados piores que os obtidos com ADAM:

Otimizador	Validation Accuracy
SGD	0,6825

Metodologia



➤ **1E-03**

➤ **1E-04**

Metodologia

- Quinta rodada de simulações:
 - Melhor MLP: 1 camada com 1024 neurônios
 - Sem Dropout
 - Função de ativação: ReLu
 - Otimizador: ADAM
 - **Taxa de aprendizado: 1e-4**

- Resultados piores que os obtidos com taxa de 1e-3:

Taxa de aprendizado	Validation Accuracy
1,00E-04	0,8001

Resultados

- Numero de camadas ocultas: 1
- Numero de neurônios: 1024
- Dropout: 0.2
- Otimizador: Adam
- Taxa de aprendizado: 1E-3
- Precisão modelo: 82.15%
-

Conclusões

- Menor numero de camadas → Melhor precisão
- Maior numero de neurônios → Melhor precisão
- Aplicação de dropout não muda melhor arquitetura.
- ReLu é a melhor função de ativação.
- Otimizador: Adam > SGD
- **O desempenho do modelo depende dos dados e do problema.**

Referências

- [1] S. Telford, M. I. Mazhar, and I. Howard, “Condition Based Maintenance (CBM) in the Oil and Gas Industry : An Overview of Methods and Techniques,” Proc. 2011 Int. Conf. Ind. Eng. Oper. Manag. Kuala Lumpur, Malaysia, January 22 – 24, 2011, pp. 1152–1159, 2011.
- [2] M. C. A. Olde Keizer, S. D. P. Flapper, and R. H. Teunter, “Condition-based maintenance policies for systems with multiple dependent components: A review,” Eur. J. Oper. Res., vol. 261, no. 2, pp. 405–420, 2017.
- [3] S. Khan and T. Yairi, “A review on the application of deep learning in system health management,” Mech. Syst. Signal Process., vol. 107, pp. 241–265, 2018.
- [4] H. Shao, H. Jiang, Y. Lin, and X. Li, “A novel method for intelligent fault diagnosis of rolling bearings using ensemble deep auto-encoders,” Mech. Syst. Signal Process., vol. 102, pp. 278–297, 2018.

Referências

- [5] C. M. Bishop, Pattern Recognition and Machine Learning, vol. 53, no. 9. 2013.
- [6] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep Learning and Its Applications to Machine Health Monitoring: A Survey,” vol. 14, no. 8, pp. 1–14, 2016.
- [7] A. Kumar, R. Shankar, and L. S. Thakur, “A big data driven sustainable manufacturing framework for condition-based maintenance prediction,” J. Comput. Sci., 2017.
- [8] D. Verstraete, M. Engineering, and M. Engineering, “Deep Learning Enabled Fault Diagnosis Using Time-Frequency Image Analysis of Rolling Element Bearings,” pp. 1–29.

Referências

- [9] Society for Machinery Failure Prevention Technology. Data Set. Disponível em <<https://mfpt.org/fault-data-sets/>>. Acesso em 20.11.2018
- [10] Kugellager Express. Disponível em: <<https://www.kugellager-express.de/deep-groove-ball-bearing-6411-open-55x140x33-mm>> Acesso em 27.12.2018
- [11] Bearing Data Center. Case Western Reserve University. Equipamento de teste. Disponível em: <<https://csegroups.case.edu/bearingdatacenter/pages/apparatus-procedures>>. Acesso em 25.11.2018.
- [12] Deep Cognition – Deep Learning Studio. Disponível em: <<https://deepcognition.ai/>>. Acesso em 20.11.2018.

Detecção de falhas usando Redes Neurais

PERGUNTAS?

OBRIGADO!

Dezembro, 2018