Utilização de técnicas de aprendizagem profunda para estimativa de fechamento de poços verticais em rochas salinas

Projeto
Tópicos Especiais em Computação Visual e Inteligente
Aprendizagem Profunda – PPGI017-10, 2019.2
Prof. Tiago F. Vieira

Ricardo A. Fernandes
Matrícula: 2019105350 (PPGEC/UFAL)

Maceió, 12 de fevereiro de 2019

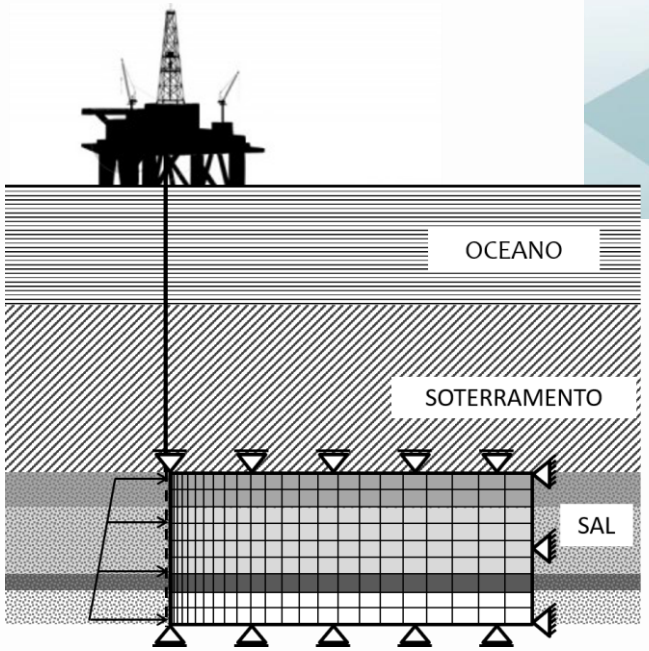
PROJETO Motivação

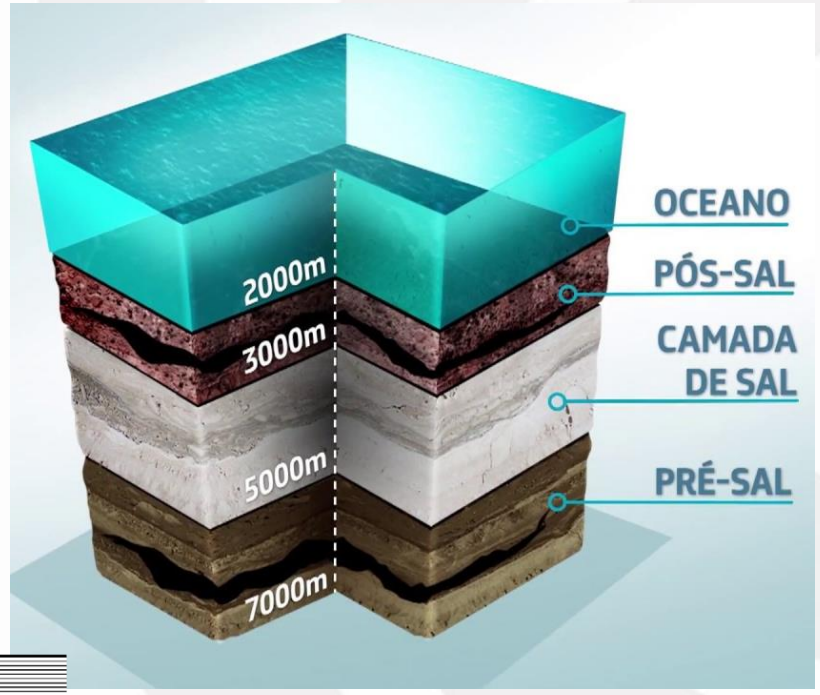
Rochas salinas:

- Presença associada a reservatórios de petróleo
- Impermeabilidade: boas rochas selantes
- Fluência: deformação no tempo
- Peso de fluido: controle do fechamento
- Fechamento do poço: aprisionamento de coluna

Modelagem computacional de rochas salinas:

- Simulação do fechamento do poço
- Experimentação de cenários diferentes
- Projetos mais seguros e eficientes





Fonte: **Petrobras**

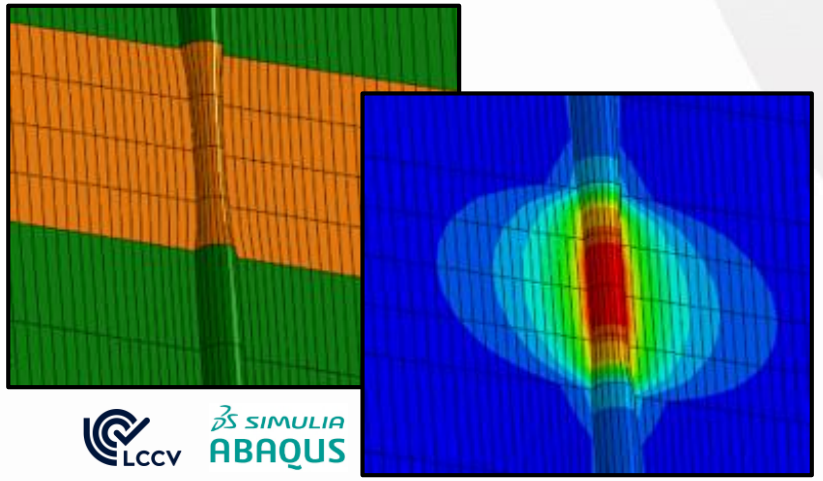
PROJETO Objetivos

Objetivo geral:

- Estimar fechamento de poços verticais em rochas salinas

Objetivos específicos:

- Dimensionar peso de fluido ideal
- Obter estimativas de repasse (alargamento do poço)
- Aplicar técnicas de aprendizado de máquina para estimativa do fechamento
  - Fechamento da borda do poço
    - ao longo da profundidade do modelo
    - ao longo do tempo de análise
  - Redes com camadas densas
  - Problema de regressão

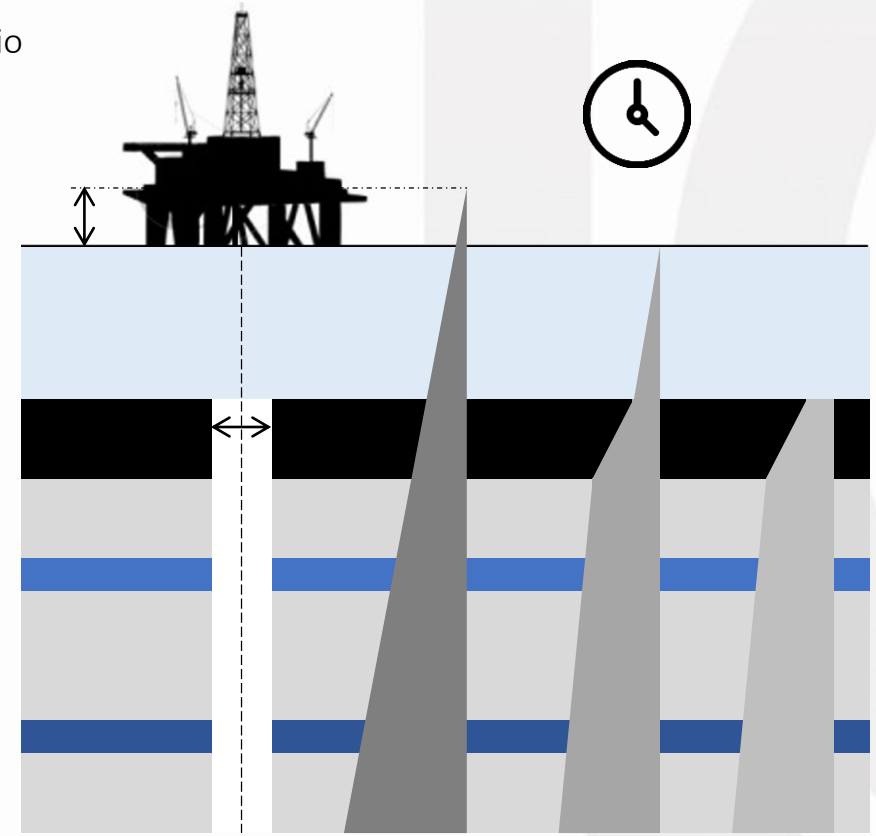


PROJETO Metodologia

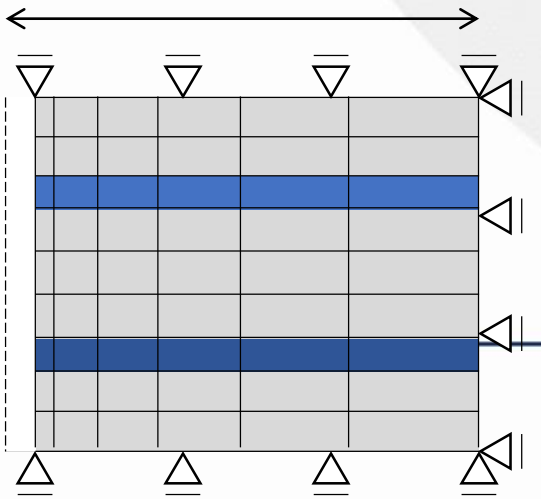
Definição do cenário (variáveis de entrada)

- Airgap
- Diâmetro da broca
- Densidade do fluido de perfuração
- Tempo de análise
- Perfil de tensão geostática
- Perfil de temperatura
- Perfil de litologias (rochas)
- Parâmetros do modelo numérico
  - Raio externo
  - Refinamento vertical
  - Refinamento radial e bias

Cenário



Modelo numérico



PROJETO Metodologia

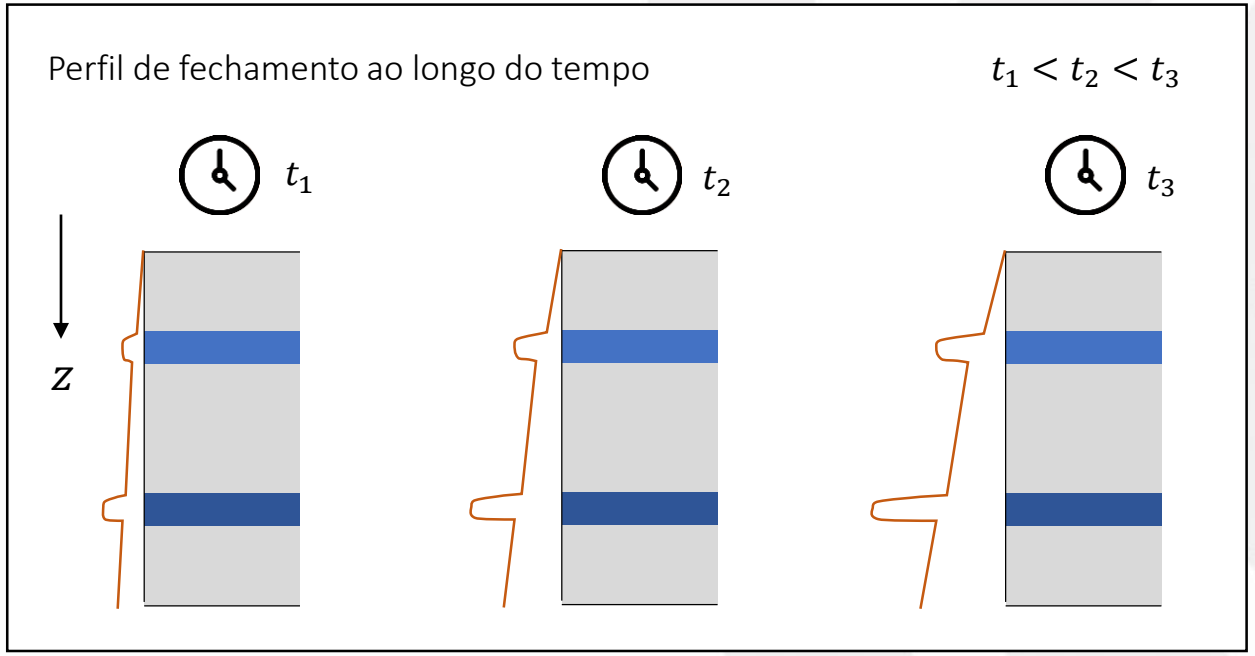
Definição do cenário (variáveis de entrada)

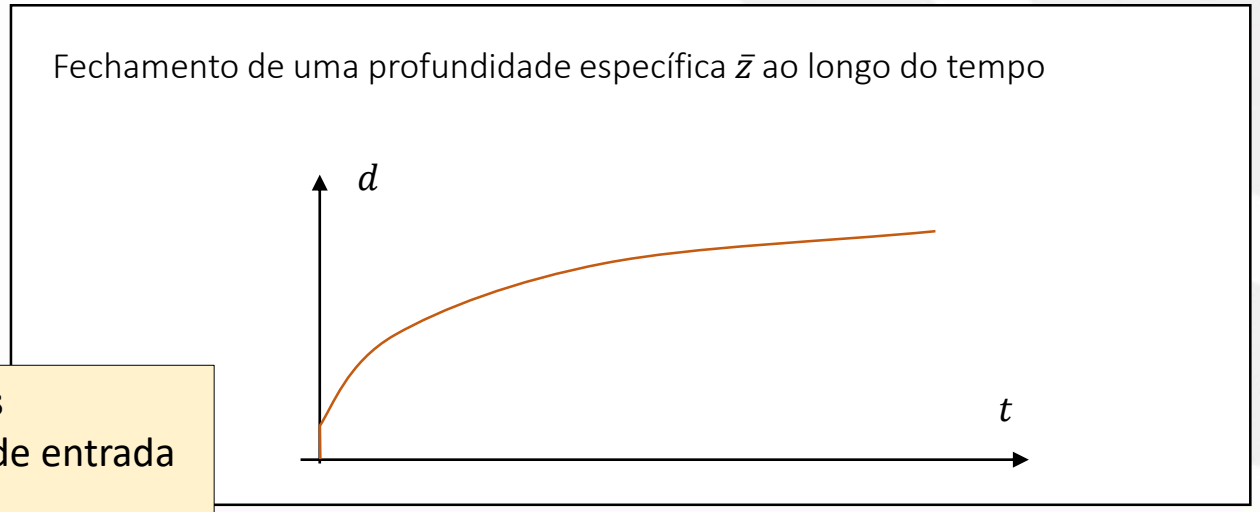
- Airgap
- Diâmetro da broca
- Densidade do fluido de perfuração
- Tempo de análise
- Perfil de tensão geostática
- Perfil de temperatura
- Perfil de litologias (rochas)
- Parâmetros do modelo numérico
  - Raio externo
  - Refinamento vertical
  - Refinamento radial e bias

Resultado da simulação numérica

- Sequência temporal do perfil de fechamento da borda do poço

$d(z,t)$





Simulação de diversos cenários

- Intervalos para as variáveis de entrada
- Combinação de valores

→ Geração de dados para treinamento e teste da rede

PROJETO Metodologia

Definição do cenário
(variáveis de entrada)
- Airgap
- Diâmetro da broca
- Densidade do fluido de perfuração
- Tempo de análise
- Perfil de tensão geostática
- Perfil de temperatura
- Perfil de litologias (rochas)
- Parâmetros do modelo numérico
  - Raio externo
  - Refinamento vertical
  - Refinamento radial e bias

Resultado da simulação numérica
(variável de saída)
- Sequência temporal do perfil de fechamento da borda do poço

Hipóteses

Airgap = 0
(Tempo de análise)
Instantes de interesse da análise
(Tensão geostática)
- Gradiente água: 10 kPa/m
- Gradiente soterramento: 22 kPa/m
- Gradiente sal: 21 kPa/m

(Temperatura)
- Fundo do mar: 4 degC
- Gradiente soterramento: 30 degC/km
- Gradiente sal: 12 degC/km

(Perfil de litologias)
- Litologia da profundidade de interesse
- Espessura desta litologia (intercalação)
- Camadas de halita acima e abaixo

(Parâmetros do modelo numérico)
- Raio externo = 25 m
- Refinamento vertical = 1 m
- Refinamento radial (bias) = 80 elem. (10)

<u>Nova</u> definição do cenário
(variáveis de entrada)
- Lâmina d'água (X_1)
- Soterramento (X_2)
- Diâmetro da broca (X_3)
- Densidade fluido perfuração (X_4)
- Altura de sal (X_5)
- Espessura da intercalação (X_6)

(parâmetros de entrada)
- Litologia (rocha) da intercalação (P_1)
- Instantes de interesse (P_2)

<u>Novo</u> resultado da simulação numérica
(variável de saída)
- Deslocamento da profundidade de interesse (Y)

PROJETO Metodologia

Nova definição do cenário

(variáveis de entrada)

- Lâmina d'água (X_1)
- Soterramento (X_2)
- Diâmetro da broca (X_3)
- Densidade fluido perfuração (X_4)
- Altura de sal (X_5)
- Espessura da intercalação (X_6)

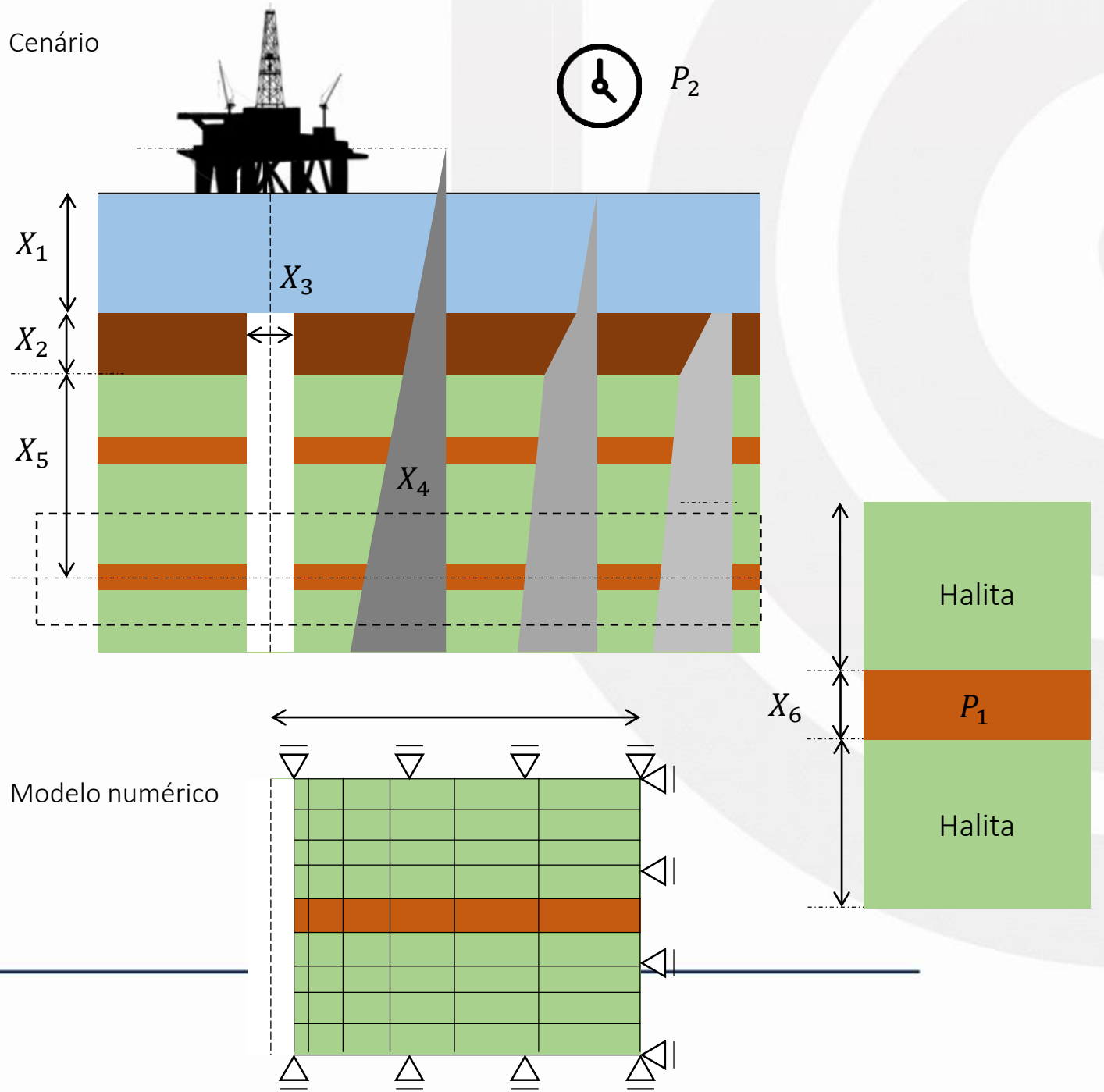
(parâmetros de entrada)

- Litologia (rocha) da intercalação (P_1)
- Instantes de interesse (P_2)

Novo resultado da simulação numérica

(variável de saída)

- Deslocamento da profundidade de interesse (Y)



PROJETO Metodologia

Definição de entradas

- Intervalo de variação das variáveis
  - X_1: $[0, 2000]$ m
  - X_2: $[500, 4000]$ m
  - X_3: $\left(17\frac{1}{2}\ \ 14\frac{3}{4},\ \ 12\frac{1}{4}\right)$ pol
  - X_4: $(10,\ \ 12,\ \ 14)$ lb/gal
  - X_5: $[50, 2000]$ m
  - X_6: $[0.5, 5]$ m
- Intervalo de variação dos parâmetros
  - P_1: ((H)alita, (T)aquidrita)
  - P_2: (0, 24, 48) h

Definição das saídas

- Caso com intercalação de halita
  - $Y_{\mathrm{H,0h}},\ Y_{\mathrm{H,24h}},\ Y_{\mathrm{H,48h}}$
- Caso com intercalação de taquidrita
  - $Y_{\mathrm{T,0h}},\ Y_{\mathrm{T,24h}},\ Y_{\mathrm{T,48h}}$

Geração dos dados

$$\begin{matrix} X_{1\mathrm{H}}^{(1)} & \cdots & X_{6\mathrm{H}}^{(1)} & \rightarrow & Y_{\mathrm{H,0h}}^{(1)} & Y_{\mathrm{H,24h}}^{(1)} & Y_{\mathrm{H,48h}}^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{1\mathrm{H}}^{(n)} & \cdots & X_{6\mathrm{H}}^{(n)} & \rightarrow & Y_{\mathrm{H,0h}}^{(n)} & Y_{\mathrm{H,24h}}^{(n)} & Y_{\mathrm{H,48h}}^{(n)} \end{matrix}$$

1170 casos com intercalação de halita

$$\begin{matrix} X_{1\mathrm{T}}^{(1)} & & X_{6\mathrm{T}}^{(1)} & \rightarrow & Y_{\mathrm{T,0h}}^{(1)} & Y_{\mathrm{T,24h}}^{(1)} & Y_{\mathrm{T,48h}}^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{1\mathrm{T}}^{(m)} & \cdots & X_{6\mathrm{T}}^{(m)} & \rightarrow & Y_{\mathrm{T,0h}}^{(m)} & Y_{\mathrm{T,24h}}^{(m)} & Y_{\mathrm{T,48h}}^{(m)} \end{matrix}$$

660 casos com intercalação de taquidrita

1830 casos

Casos

- Cada caso corresponde a uma simulação numérica
- Simulações simples duram minutos e complexas duram horas

Modelos NN

- Consideram-se 6 modelos
- Casos com halita
  - estimando deslocamento em 0 h
  - estimando deslocamento em 24 h
  - estimando deslocamento em 48 h
- Casos com taquidrita
  - estimando deslocamento em 0 h
  - estimando deslocamento em 24 h
  - estimando deslocamento em 48 h

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Importar bibliotecas

- Definir modelo
  - Nome do modelo
  - Parâmetro de litologia
    - 0, para halita
    - 1, para taquidrita
  - Nome do parâmetro de desloc.
    - ‘d0’
    - ‘d24’
    - ‘d48’

```
# Load standard Python libs
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import time

# Ignore info and warning (gpu) messages from tensorflow
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# Load Keras libs
from keras import metrics, regularizers, optimizers
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import Adam, SGD
from keras.callbacks import EarlyStopping

# For reproducibility purposes
seed = 2020
np.random.seed(seed)


# Define model name, lithol and label parameters
#--------------------------------------------------------------------------
model_name = 'model_01'
lithol = 0
label = 'd0'

```

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Carregar dados
  - Arquivo .csv
- Definir colunas
  - Inputs
  - Labels
- Função filtragem de dados
  - Depende de litologia e label
- Aplicar filtragem de dados

```
30  # Load data
31  #-------------------------------------------------------------------------
32
33  # Load data from CSV
34  dataset_org  = pd.read_csv('./labeled_data.csv')
35
36  # Select columns
37  lithol_col = 'lithol'
38  label_cols = ['d0', 'd24', 'd48']
39  dataset = pd.DataFrame(dataset_org, columns=[
40      'wd', 'ovbd', 'od', 'mud', 'sh', 'lithol', 'interc',
41      'd0', 'd24', 'd48'])
42
43  # Function to get a data subset
44  def data_subset (df, lithol, label, lithol_col='lithol', label_cols=['d0', 'd24', 'd48']):
45      ignore_cols = [lithol_col] + label_cols
46      ignore_cols.remove(label)
47      data = df.loc[df.lithol==lithol, :].drop(ignore_cols, axis=1)
48      return data
49
50  # Select data subset
51  print('lithol:', lithol,'label:', label)
52  data = data_subset(dataset, lithol, label)
53  # print(data.describe())
54
```

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Função para split dos dados
  - Treinamento
  - Validação
- Aplicação do split
  - 80% treinamento
  - 20% validação
- Separar dados em
  - Variáveis de entrada (X)
  - Label (Y)

```
56  # Split data for training and validation
57  #-------------------------------------------------------------------------
58
59  # Function to split a range of dataframe into train/validate subranges
60  def train_validate_split (df, train_part=0.6, validate_part=0.2):
61
62      total_size = train_part + validate_part
63      train_frac = train_part / total_size
64
65      m = len(df)
66      perm = np.random.permutation(df.index)
67
68      train_end = int(train_frac * m)
69      train = perm[:train_end]
70      validate = perm[train_end:]
71
72      return train, validate
73
74  # Split index ranges
75  train_size, valid_size = (80, 20)
76  train, valid = train_validate_split (data, train_part=train_size, validate_part=valid_size)
77
78  # Extract data for training and validation into x and y vectors
79  x_train = data.loc[train, :].drop(label, axis=1)
80  y_train = data.loc[train, [label]]
81  x_valid = data.loc[valid, :].drop(label, axis=1)
82  y_valid = data.loc[valid, [label]]
83
```

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Função para normalização dos dados
  - Distribuição normal padrão
    - Média = 0
    - Desvio padrão = 1
- Aplicar normalização única
  - Treinamento
  - Validação
- Mostrar número de:
  - Variáveis
  - Amostras
    - Treinamento
    - Validação

```
84
85  # Prepare data for training and validation of the Keras model
86  #-------------------------------------------------------------------------
87
88  # Function to get statistics about data frame
89  def norm_stats (d1, d2):
90      ds = np.append(d1, d2, axis=0)
91      mu = np.mean(ds, axis=0)
92      sigma = np.std(ds, axis=0)
93      return (mu, sigma)
94
95  # Training and validation data arrays
96  x_train_arr = np.array(x_train)
97  y_train_arr = np.array(y_train)
98  x_valid_arr = np.array(x_valid)
99  y_valid_arr = np.array(y_valid)
100
101 # Calculare mean and standard deviation of data
102 (x_mean, x_std) = norm_stats (x_train_arr, x_valid_arr)
103 (y_mean, y_std) = norm_stats (y_train_arr, y_valid_arr)
104
105 # Normalise training and validation data
106 xn_train_arr = (x_train_arr - x_mean) / x_std
107 yn_train_arr = (y_train_arr - y_mean) / y_std
108 xn_valid_arr = (x_valid_arr - x_mean) / x_std
109 yn_valid_arr = (y_valid_arr - y_mean) / y_std
110
111 print('Training shape: ', xn_train_arr.shape)
112 print('Training samples: ', xn_train_arr.shape[0])
113 print('Validation samples: ', xn_valid_arr.shape[0])
114
```

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Definir parâmetros para modelo Keras
  - Número de camadas
  - Neurônios por camada
  - Funções de ativação
  - Kernel
    - Initializer
    - Regularizer
  - Bias regularizer
  - Dropout
  - Otimizer
    - Tipo
    - Learning rate
  - Número de épocas
  - Tamanho do batch
  - Callbacks

```
115
116   # Create Keras model
117   #------------------------------------------------------------------------
118   # Time counter
119   start = time.time()
120   nodes = [36, 36, 36]
121   print('Nodes:',nodes)
122   activation = ['relu', 'relu', 'relu', 'relu', 'relu']
123   kernel_initializer = ['normal', 'normal', 'normal', 'normal', 'normal']
124   kernel_regularizer = [None, None, None, None, None]
125   bias_regularizer = [None, None, None, None, None]
126   dropout = [0., 0., 0., 0., 0.]
127   # optimizer = Adam(learning_rate=0.001)
128   optimizer = SGD(learning_rate=0.1, momentum=0.8, nesterov=False)
129
130   # Define how many epochs of training should be done and what is the batch size
131   epochs = 10000
132   batch_size = 128
133   print('Epochs: ', epochs)
134   print('Batch size: ', batch_size)
135
136   # Specify Keras callbacks
137   keras_callbacks = []
138   # keras_callbacks = [EarlyStopping(monitor='val_mean_absolute_error', patience=20, verbose=0)]
139
```

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Função genérica para definição do modelo Keras

- Problema de regressão
  - Não especificar função de ativação na última camada
  - Usar função de perda 'Mean Squared Error'

Table 4.1 Choosing the right last-layer activation and loss function for your model

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

Fonte: Chollet F. (2018) Deep Learning with Python. Manning Publications Co.

  - Usar 'Mean Absolute Error' como métrica ao invés da acurácia (não faz sentido em problemas de regressão)

Fonte: https://www.kaggle.com/ironfrown/deep-learning-house-price-prediction-keras

```
139
140  # Define model with n layers, optimizer, dropouts and L1/L2 regularisers
141  def set_model (x_size, y_size):
142
143      model = Sequential()
144
145      model.add(Dense(nodes[0],
146                      activation=activation[0],
147                      kernel_initializer=kernel_initializer[0],
148                      kernel_regularizer=kernel_regularizer[0],
149                      bias_regularizer=bias_regularizer[0],
150                      input_shape=(x_size,)
151                      ))
152      model.add(Dropout(dropout[0]))
153
154      for i in range(1,len(nodes)):
155          model.add(Dense(nodes[i],
156                      activation=activation[i],
157                      kernel_initializer=kernel_initializer[i],
158                      kernel_regularizer=kernel_regularizer[i],
159                      bias_regularizer=bias_regularizer[i]
160                      ))
161          model.add(Dropout(dropout[i]))
162
163      model.add(Dense(y_size))
164
165      model.compile(loss='mse',
166                    optimizer=optimizer,
167                    metrics=[metrics.mae])
168      return model
169
170  # Create the model
171  model = set_model (xn_train_arr.shape[1], yn_train_arr.shape[1])
172  # model.summary()
173
```

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Aplicar método de ajuste (Fit)

```
175   # Fit/Train Keras model
176   #-------------------------------------------------------------------------
177
178   # Fit model and record the history and validation
179   history = model.fit(xn_train_arr, yn_train_arr,
180                       batch_size=batch_size,
181                       epochs=epochs,
182                       verbose=0,
183                       validation_data=(xn_valid_arr, yn_valid_arr),
184                       callbacks=keras_callbacks)
185
186
```

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Avaliar medidas de 'Loss'
  - Treinamento
  - Validação
- Avaliar erro absoluto percentual
  - usando método 'Predict'
    - Treinamento
    - Validação

```
# Evaluate and report performance of the trained model
#------------------------------------------------------------------------------
train_score = model.evaluate(xn_train_arr, yn_train_arr, verbose=0)
valid_score = model.evaluate(xn_valid_arr, yn_valid_arr, verbose=0)

print('Train MAE: %.2e, Train Loss: %.2e' % (train_score[1], train_score[0]))
print('Valid MAE: %.2e, Valid Loss: %.2e' % (valid_score[1], valid_score[0]))

# Evaluate prediction errors
yn_train_arr_pred = model.predict(xn_train_arr)
y_train_arr_pred = y_mean + yn_train_arr_pred * y_std
train_rel_err = abs(y_train_arr_pred / y_train_arr - 1)

print('Train Minimum Error: %.3f%%' % np.min(train_rel_err*100))
print('Train Maximum Error: %.3f%%' % np.max(train_rel_err*100))
print('Train Mean Error: %.3f%%' % np.mean(train_rel_err*100))

yn_valid_arr_pred = model.predict(xn_valid_arr)
y_valid_arr_pred = y_mean + yn_valid_arr_pred * y_std
valid_rel_err = abs(y_valid_arr_pred / y_valid_arr - 1)

print('Valid Minimum Error: %.3f%%' % np.min(valid_rel_err*100))
print('Valid Maximum Error: %.3f%%' % np.max(valid_rel_err*100))
print('Valid Mean Error: %.3f%%' % np.mean(valid_rel_err*100))

# Elapsed time
dt = time.time() - start
print('Time elapsed: %f sec' % dt)
```

PROJETO Metodologia

Rotina desenvolvida (walkthrough)

- Função para plotar histórico
  - ‘Loss’
    - Trainamento
    - Validação
  - ‘Mean Absolute Error’
    - Treinamento
    - Validação

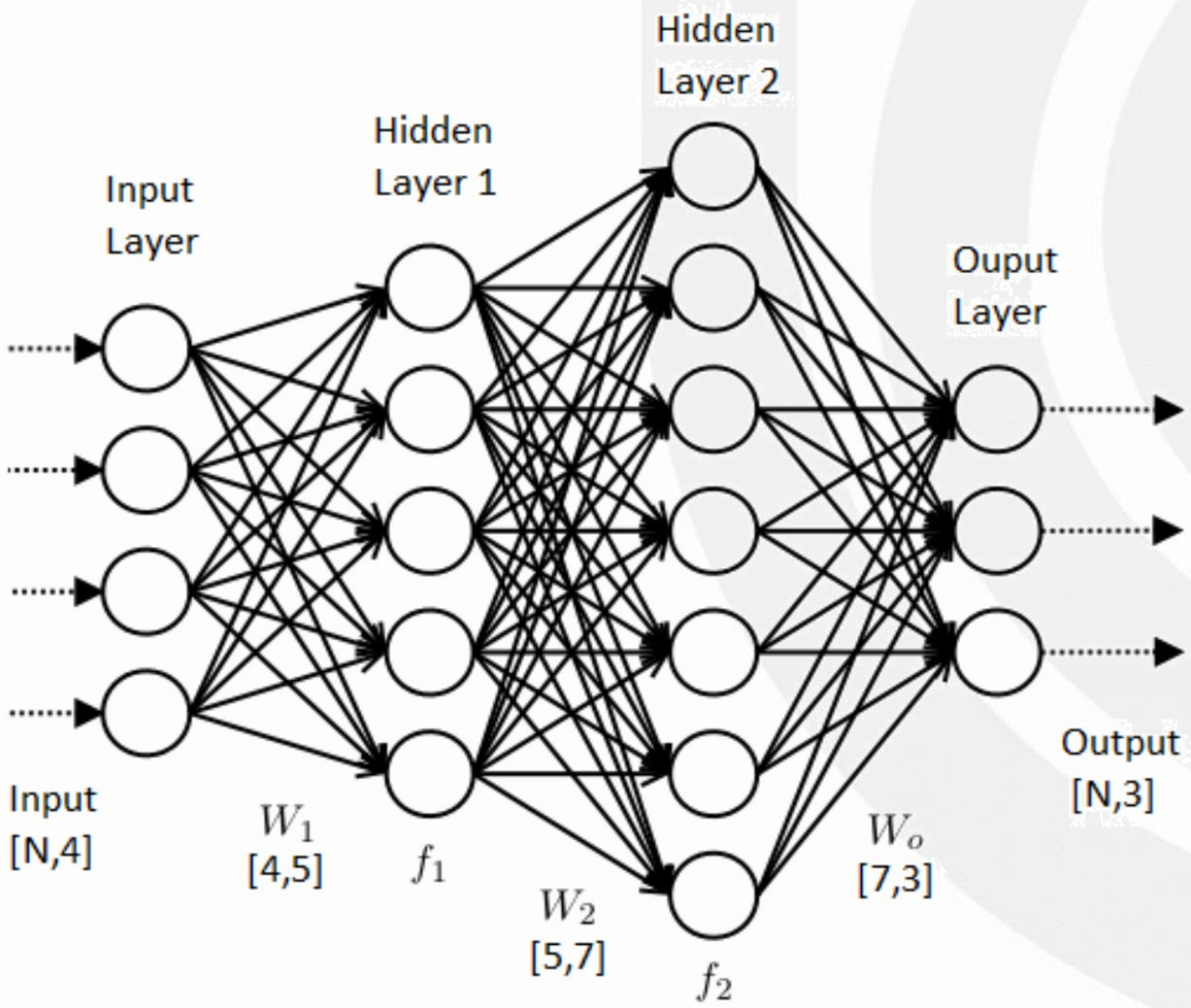
```
216  # This function allows plotting of the training history
217  def plot_hist (h, xsize=6, ysize=10):
218
219      # Prepare plotting
220      fig_size = plt.rcParams['figure.figsize']
221      plt.rcParams['figure.figsize'] = [xsize, ysize]
222      fig, axes = plt.subplots(nrows=4, ncols=4, sharex=True)
223
224      # summarize history for MAE
225      plt.subplot(211)
226      plt.plot(h['mean_absolute_error'])
227      plt.plot(h['val_mean_absolute_error'])
228      plt.title('Training vs Validation MAE')
229      plt.ylabel('MAE')
230      plt.xlabel('Epoch')
231      plt.xscale('log')
232      plt.yscale('log')
233      plt.legend(['Train', 'Validation'], loc='upper left')
```

```
235      # summarize history for loss
236      plt.subplot(212)
237      plt.plot(h['loss'])
238      plt.plot(h['val_loss'])
239      plt.title('Traning vs Validation Loss')
240      plt.ylabel('Loss')
241      plt.xlabel('Epoch')
242      plt.xscale('log')
243      plt.yscale('log')
244      plt.legend(['Train', 'Validation'], loc='upper left')
245
246      # Plot it
247      plt.draw()
248      # plt.show()
249      plt_name = f'dlsalt_{model_name}_val.png'
250      plt.savefig(plt_name)
251      print('Saving', plt_name)
252      return
253
254  # Now plot the training history
255  plot_hist(history.history, xsize=8, ysize=12)
```

PROJETO Resultados

Descrição: modelo 01
(Halita, Deslocamento instantâneo em 0h)

- 3 hidden layers
  - 36 neurônios em cada
- Output layer
  - 1 único neurônio
- Optimizador
  - Teste com Adam
  - SGD (Stochastic Gradient Descent)
    - Learning rate = 0.1
    - Momentum = 0.8
- Amostras: 1170
  - Treinamento: 935 (80%)
  - Validação: 235 (20%)
- Tamanho do batch = 128
- Número de épocas: 100 k



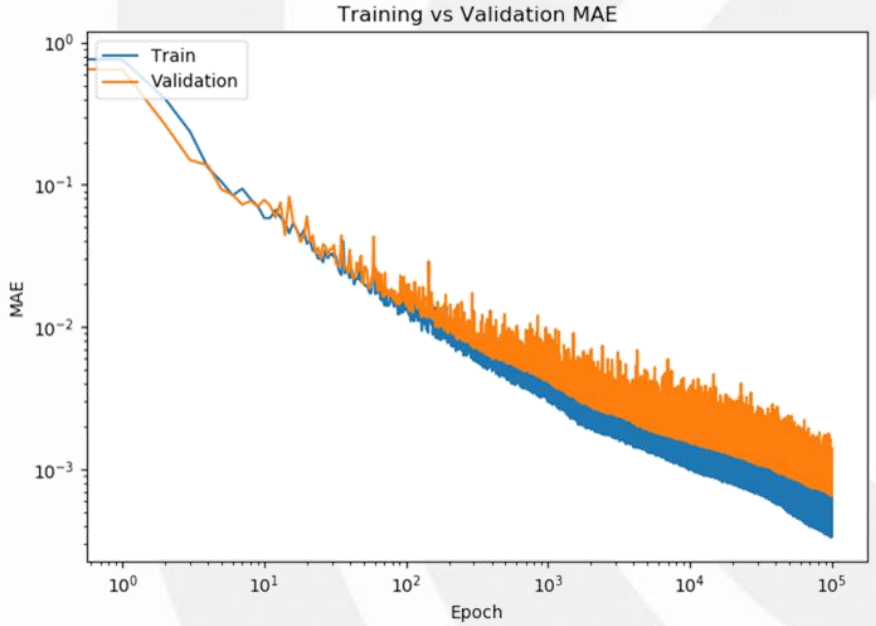
PROJETO Resultados

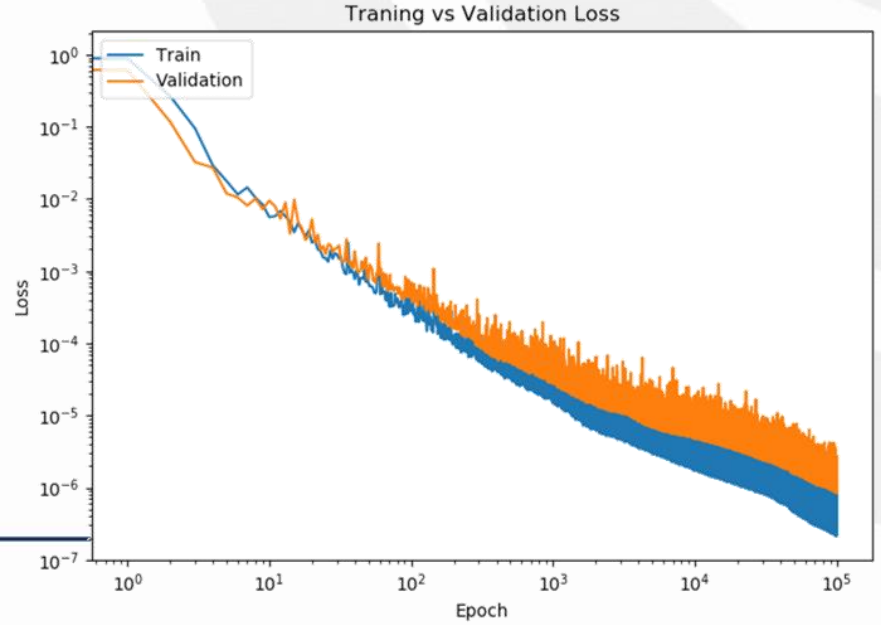
Descrição: modelo 01
(Halita, deslocamento instantâneo em 0h)

- 3 hidden layers
  - 36 neurônios em cada
- Output layer
  - 1 único neurônio
- Optimizador
  - Teste com Adam
  - SGD (Stochastic Gradient Descent)
    - Learning rate = 0.1
    - Momentum = 0.8
- Amostras: 1170
  - Treinamento: 935 (80%)
  - Validação: 235 (20%)
- Tamanho do batch = 128
- Número de épocas: 100 k
- Elapsed time: 10 min

Metric evaluation		
Train	MAE	4.33e-04
	Loss	3.21e-07
Valid	MAE	7.33e-04
	Loss	9.84e-07

Predict values		
Train	Mean APE	0.055%
	Max. APE	2.752%
Valid	Mean APE	0.101%
	Max. APE	3.123%





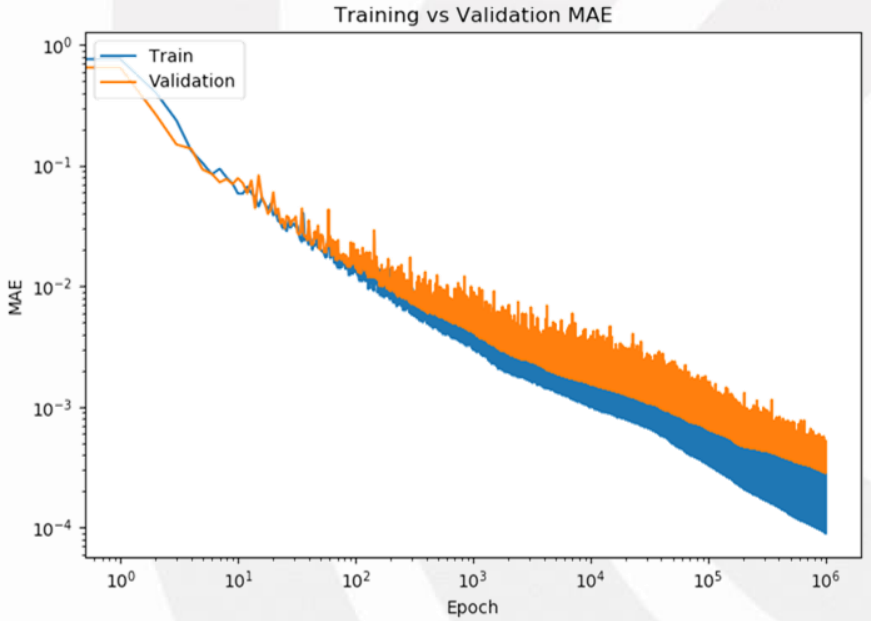
PROJETO Resultados

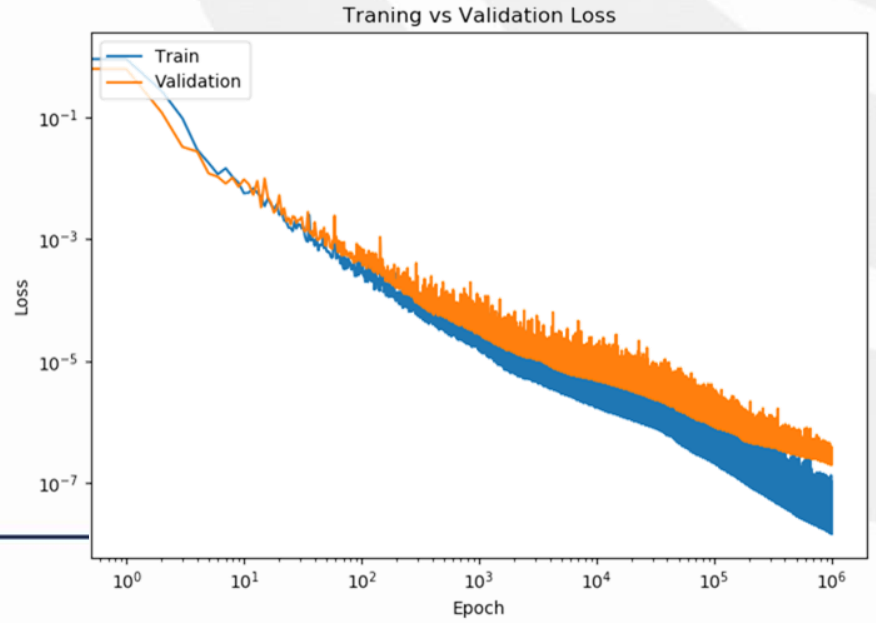
Descrição: modelo 01
(Halita, deslocamento instantâneo em 0h)

- 3 hidden layers
  - 36 neurônios em cada
- Output layer
  - 1 único neurônio
- Optimizador
  - Teste com Adam
  - SGD (Stochastic Gradient Descent)
    - Learning rate = 0.1
    - Momentum = 0.8
- Amostras: 1170
  - Treinamento: 935 (80%)
  - Validação: 235 (20%)
- Tamanho do batch = 128
- Número de épocas: 100 k → **1 M**
- Elapsed time: 10 min → **2 h**

Metric evaluation		
Train	MAE	4.33e-04 → **9.44e-05**
	Loss	3.21e-07 → **1.55e-08**
Valid	MAE	7.33e-04 → **2.92e-04**
	Loss	9.84e-07 → **2.05e-07**

Predict values		
Train	Mean APE	0.055% → **0.014%**
	Max. APE	2.752% → **0.537%**
Valid	Mean APE	0.101% → **0.041%**
	Max. APE	3.123% → **1.030%**





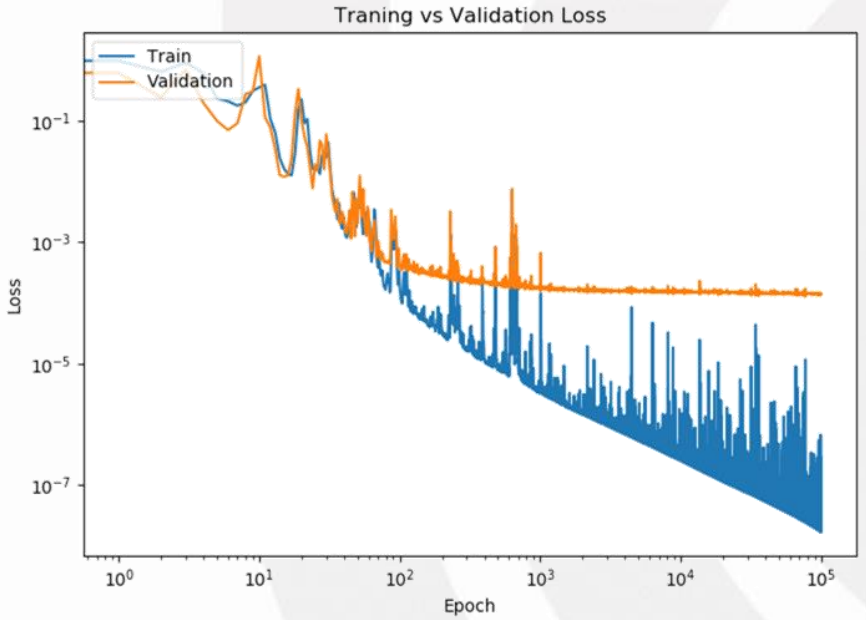
PROJETO Resultados

Descrição: modelo 02
(Halita, deslocamento em 24h)

- Buscando arquitetura ideal

Metric evaluation		
Train	MAE	1.21e-04
	Loss	2.61e-08
Valid	MAE	5.10e-03
	Loss	1.39e-04

Predict values		
Train	Mean APE	1.297%
	Max. APE	115.498%
Valid	Mean APE	3.332%
	Max. APE	61.812%



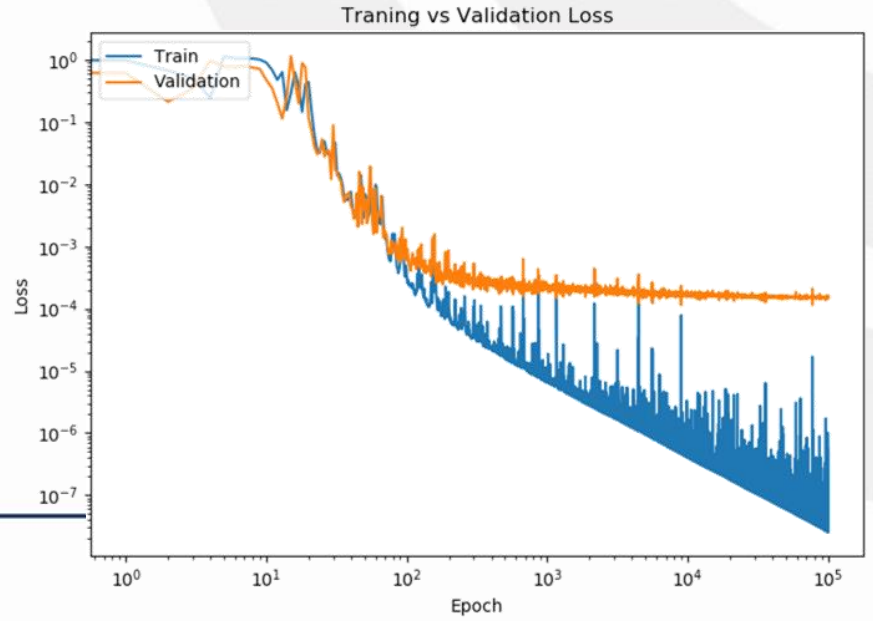
- Erro absoluto médio (validação): 0.0128 cm

Descrição: modelo 03
(Halita, deslocamento em 48h)

- Buscando arquitetura ideal

Metric evaluation		
Train	MAE	1.01e-04
	Loss	2.58e-08
Valid	MAE	5.30e-03
	Loss	1.51e-04

Predict values		
Train	Mean APE	2.605%
	Max. APE	344.504%
Valid	Mean APE	5.120%
	Max. APE	107.422%



- Erro absoluto médio (validação): 0.0276 cm

PROJETO Resultados

Descrição: modelo 04

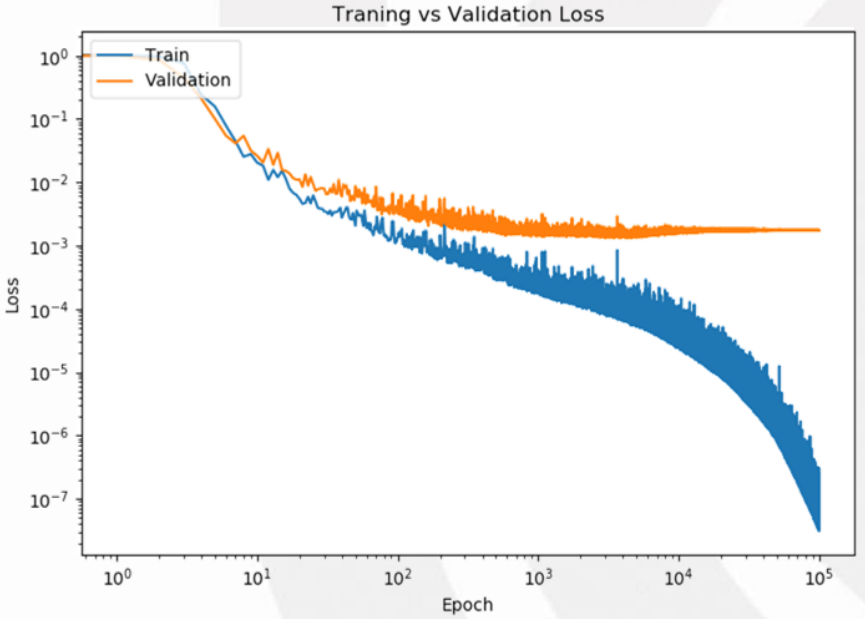
(Taquidrita, deslocamento instantâneo em 0h)

- Buscando arquitetura ideal

Metric evaluation		
Train	MAE	1.63e-04
	Loss	4.93e-08
Valid	MAE	2.81e-02
	Loss	1.74e-03

Predict values		
Train	Mean APE	0.043%
	Max. APE	3.466%
Valid	Mean APE	2.847%
	Max. APE	131.856%

- Erro absoluto médio (validação): 0.00279 cm



Descrição: modelo 05

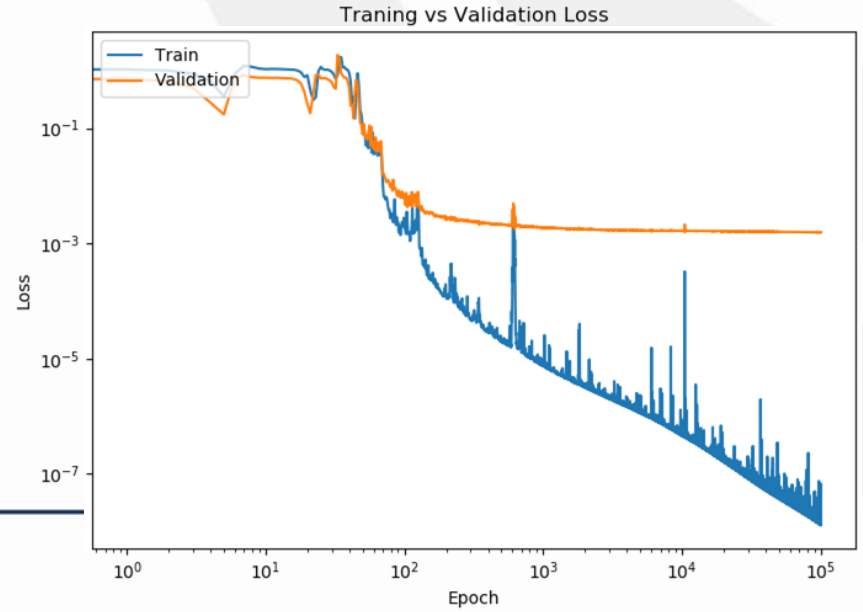
(Taquidrita, deslocamento em 24h)

- Buscando arquitetura ideal

Metric evaluation		
Train	MAE	5.73e-05
	Loss	1.30e-08
Valid	MAE	1.55e-02
	Loss	1.56e-03

Predict values		
Train	Mean APE	2.947%
	Max. APE	196.738%
Valid	Mean APE	6.368%
	Max. APE	169.192%

- Erro absoluto médio (validação): 0.229 cm



PROJETO Resultados

Descrição: modelo 06

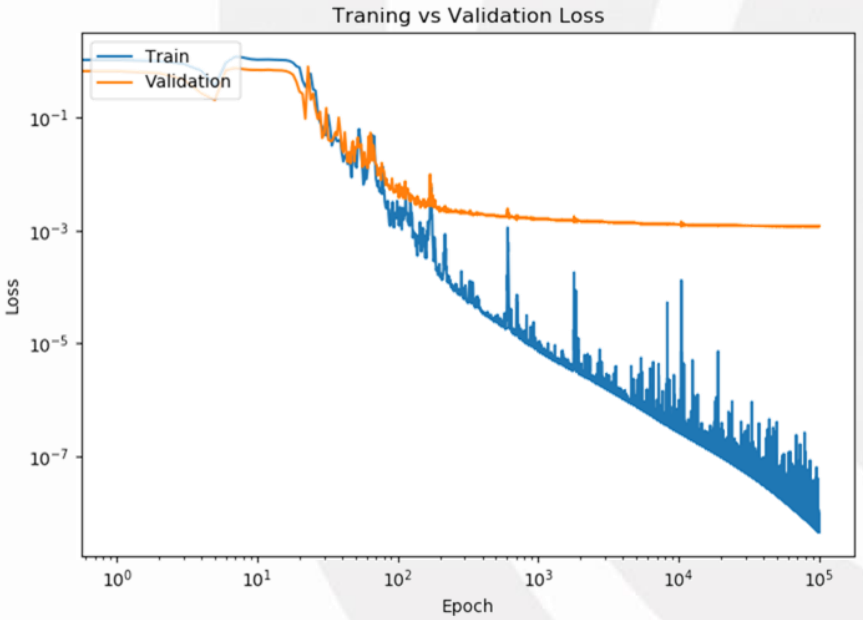
(Taquidrita, deslocamento instantâneo em 0h)

- Buscando arquitetura ideal

Metric evaluation		
Train	MAE	4.11e-05
	Loss	4.81e-09
Valid	MAE	1.38e-02
	Loss	1.20e-03

Predict values		
Train	Mean APE	3.578%
	Max. APE	463.561%
Valid	Mean APE	6.913%
	Max. APE	93.337%

- Erro absoluto médio (validação): 0.37 cm



PROJETO Conclusões

- Os modelos de halita (modelos 01-03) se mostraram mais bem comportados que os modelos envolvendo taquidrita (modelos 04-06)
- Os modelos com saída de deslocamento instântaneo em 0h (modelo 01 e 04) se mostratam mais bem comportados que os demais
- O modelo 01 forneceu resultados muito bons para a predição dos valores de treinamento e validação, sendo todos os por volta ou menores que 1%
- Os demais modelos forneceram erros médios aceitáveis, no entanto, erros máximos ainda elevados
  - Treinamento: erros médios menores, erros máximos maiores
  - Validação: erros médios maiores, erros máximos menores
- Ainda buscam-se os hiperparâmetros e arquitetura de rede ideal para os modelos 02-06
  - Variação automatizada de parâmetros
  - Checagem das métricas de erros
  - Determinação da rede “ótima”

PROJETO Referências

- **Machine Learning / Deep Learning**
  - Chollet F. (2018) Deep Learning with Python. Manning Publications Co.
  - Kaggle
    - https://www.kaggle.com/ironfrown/deep-learning-house-price-prediction-keras
  - Machine Learning Mastery
    - https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
    - https://machinelearningmastery.com/applied-deep-learning-in-python-mini-course/
    - https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/
    - https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/
- **Problema de fechamento de poços em rocha salina**
  - Ferreira J.F., Fernandes R.A., Fernandes C.N.A., Lages E.N. (2019) An automed strategy for simulation of wellbore drilling in salt rocks. XL CILAMCE
  - Lins G.K.L., Fernandes R.A., Lira W.W.M., Lages E.N. (2019) Study of viscoelastic laws applied to salt rock models. XL CILAMCE
  - Costa A.M., Poiate Jr E., Amaral C.S., Gonçalves C.J.C., Falcao J.L. (2010) Geomechanics applied to the well design through salt layers in Brazil: a history of success. ARMA 10-239
  - Poiate Jr E., Costa A.M. Falcao J.L. (2006) Well design of drilling through thick evaporite layers in Santos Basin – Brazil. IADC/SPE 99161

Utilização de técnicas de aprendizagem profunda para estimativa de fechamento de poços verticais em rochas salinas

Projeto
Tópicos Especiais em Computação Visual e Inteligente
Aprendizagem Profunda – PPGI017-10, 2019.2
Prof. Tiago F. Vieira

Ricardo A. Fernandes
Matrícula: 2019105350 (PPGEC/UFAL)

Maceió, 12 de fevereiro de 2019