Diagnóstico com Deep Autoencoder

Criado por: Leonardo Franco de Godói

Data: 27/09/2019

Descrição:

O dataset utilizado, contendo dados obtidos de rolamentos, está disponível em Case Western Reserve University bearing data center, através do link:

http://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website (http://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website).

O dataset contempla três tipos de falhas mecânicas, associadas a:

- Elementos rolantes.
- Anel interno.
- Anel externo.

Além disso, o dataset é também dividido entre duas condições de operação em termos de velocidade de rotação - configuração de carga:

- 20-0
- 30-2

Os dados foram obtidos através de 8 sensores:

- 1 (vibração do motor).
- 2, 3, 4 (vibração da caixa de engrenagens planetária nas direções X, Y e Z).
- 5 (torque do motor).
- 6, 7, 8 (vibração da caixa de engrenagens paralela nas direções X, Y e Z).

Esta implementação visa o diagnóstico de um rolamento através da classificação de sua condição entre 5 possíveis estados:

- Saudável (Health).
- Falha no anel interno (Inner).
- Falha no anel externo (Outer).
- Falha de elemento rolante (Ball).
- Falha combinada dos anéis interno e externo (Comb).

Um Deep Autoencoder (Deep AE) com a configuração (256-128-64-32-64-128-256) é aplicado sobre um dataset misto, contendo ocorrências de todos os estados mencionados. Após o treinamento do Autoencoder, uma camada para classificação é conectada ao gargalo (contendo a informação comprimida) e um novo treino, desta vez supervisionado, é realizado.

Importando os pacotes necessários.

In [2]:

```
import pandas as pd
import numpy as np
from keras.layers import Input, Dense
from keras.models import Model
import keras
from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

Informando o diretório dos datasets originais.

In [3]:

```
# Definindo o diretório dos datasets originais
dir_health = 'health_30_2.csv'
dir_inner = 'inner_30_2.csv'
dir_outer = 'outer_30_2.csv'
dir_ball = 'ball_30_2.csv'
dir_comb = 'comb_30_2.csv'
```

Número de classes (possíveis estados).

In [4]:

```
num_classes = 5
```

Definindo os parâmetros da rede neural.

In [5]:

```
hidden_layer1 = 256
hidden_layer2 = 128
hidden_layer3 = 64
hidden_layer4 = 32
NUM_EPOCHS = 100
BATCH_SIZE = 1024
act_func = 'relu'
```

Carregando os datasets originais (8 sensores x 1048560 medições).

- *Dataset saudável (health).*
- *Dataset com falha no anel interno (inner).*
- *Dataset com falha no anel externo (outer).*
- *Dataset com falha nos elementos rolantes (ball).*
- *Dataset com falha combinada dos anéis (comb).*

In [6]:

```
dataset_health = pd.read_csv(dir_health, sep='\t', header=-1)
dataset_inner = pd.read_csv(dir_inner, sep='\t', header=-1)
dataset_outer = pd.read_csv(dir_outer, sep='\t', header=-1)
dataset_ball = pd.read_csv(dir_ball, sep='\t', header=-1)
dataset_comb = pd.read_csv(dir_comb, sep='\t', header=-1)
```

Removendo a última coluna (vazia).

In [7]:

```
dataset_health = dataset_health.iloc[:,:-1]
dataset_inner = dataset_inner.iloc[:,:-1]
dataset_outer = dataset_outer.iloc[:,:-1]
dataset_ball = dataset_ball.iloc[:,:-1]
dataset_comb = dataset_comb.iloc[:,:-1]
```

Definindo rótulos para as colunas e ajustando as linhas.

In [9]:

```
dataset_health.columns = ['Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4',
                          'Sensor 5', 'Sensor 6', 'Sensor 7', 'Sensor 8']
dataset_inner.columns = dataset_health.columns
dataset_inner.index = dataset_health.index
dataset_outer.columns = dataset_health.columns
dataset_outer.index = dataset_health.index
dataset_ball.columns = dataset_health.columns
dataset_ball.index = dataset_health.index
dataset_comb.columns = dataset_health.columns
dataset_comb.index = dataset_health.index
```

Definindo os conjuntos de treinamento e validação.

In [10]:

```
train_size = 50000
test_size = 20000
X_train = dataset_health[:train_size].append([dataset_inner[:train_size],
                                          dataset_outer[:train_size],
                                          dataset_ball[:train_size],
                                          dataset_comb[:train_size]])
X_test = dataset_health[:test_size].append([dataset_inner[:test_size],
                                          dataset_outer[:test_size],
                                          dataset_ball[:test_size],
                                          dataset_comb[:test_size]])
y_train = np.repeat(np.array([0,1,2,3,4]), train_size)
y_test = np.repeat(np.array([0,1,2,3,4]), test_size)
```

Convertendo os vetores de classes em matrizes binárias.

In [11]:

```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Estruturando o modelo do Autoencoder.

In [12]:

```
my_input = Input(shape=(X_train.shape[1],))
encoded = Dense(hidden_layer1, activation=act_func)(my_input)
encoded = Dense(hidden_layer2, activation=act_func)(encoded)
encoded = Dense(hidden_layer3, activation=act_func)(encoded)
encoded = Dense(hidden_layer4, activation=act_func)(encoded) # Gargalo
decoded = Dense(hidden_layer3, activation=act_func)(encoded)
decoded = Dense(hidden_layer2, activation=act_func)(decoded)
decoded = Dense(hidden_layer1, activation=act_func)(decoded)
decoded = Dense(X_train.shape[1], activation='sigmoid')(decoded)
autoencoder = Model(my_input, decoded)
```

```
WARNING:tensorflow:From C:\Users\leona\Anaconda3\lib\site-packages\tensorf
low\python\framework\op_def_library.py:263: colocate_with (from tensorflo
w.python.framework.ops) is deprecated and will be removed in a future vers
ion.
Instructions for updating:
Colocations handled automatically by placer.
```

Compilando o modelo.

In [13]:

```
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

Criando um mecanismo de Early Stopping baseado na perda para evitar overfitting.

In [14]:

```
es_loss = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
```

Treinando o Autoencoder.

In [15]:

```
autoencoder.fit(X_train, X_train,
                epochs=NUM_EPOCHS,
                batch_size=BATCH_SIZE,
                callbacks=[es_loss],
                shuffle=True,
                validation_data=(X_test, X_test))
```

```
WARNING:tensorflow:From C:\Users\leona\Anaconda3\lib\site-packages\tensorf
low\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math
_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 250000 samples, validate on 100000 samples
Epoch 1/100
250000/250000 [==============================] - 5s 20us/step - loss: -0.9
130 - val_loss: -1.7650
Epoch 2/100
250000/250000 [==============================] - 4s 15us/step - loss: -1.0
647 - val_loss: -1.7650
Epoch 00002: early stopping
```

Out[15]:

```
<keras.callbacks.History at 0x1268680fa20>
```

Inserindo a camada de classificação.

In [16]:

```
output = Dense(num_classes, activation='softmax')(encoded)
```

Redefinindo o modelo da rede neural.

In [17]:

```
autoencoder = Model(my_input, output)
```

Compilando o novo modelo.

In [18]:

```
autoencoder.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accura
cy'])
```

Treinando de forma supervisionada.

In [19]:

```
autoencoder.fit(X_train,
                y_train,
                epochs=NUM_EPOCHS,
                batch_size=BATCH_SIZE,
                validation_data=(X_test, y_test))
```

```
Train on 250000 samples, validate on 100000 samples
Epoch 1/100
250000/250000 [==============================] - 6s 22us/step - loss: 1.16
06 - acc: 0.4519 - val_loss: 0.7680 - val_acc: 0.6090
Epoch 2/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.76
76 - acc: 0.5641 - val_loss: 0.7246 - val_acc: 0.6102
Epoch 3/100
250000/250000 [==============================] - 5s 18us/step - loss: 0.74
70 - acc: 0.5734 - val_loss: 0.7188 - val_acc: 0.6041
Epoch 4/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.73
66 - acc: 0.5792 - val_loss: 0.7000 - val_acc: 0.6167
Epoch 5/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.73
09 - acc: 0.5827 - val_loss: 0.6901 - val_acc: 0.6219
Epoch 6/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.72
80 - acc: 0.5857 - val_loss: 0.7015 - val_acc: 0.6046
Epoch 7/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.72
16 - acc: 0.5928 - val_loss: 0.6907 - val_acc: 0.6269
Epoch 8/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.71
27 - acc: 0.6053 - val_loss: 0.6705 - val_acc: 0.6518
Epoch 9/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.70
61 - acc: 0.6136 - val_loss: 0.6672 - val_acc: 0.6616
Epoch 10/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.69
74 - acc: 0.6235 - val_loss: 0.6857 - val_acc: 0.6428
Epoch 11/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.69
00 - acc: 0.6326 - val_loss: 0.6568 - val_acc: 0.6535
Epoch 12/100
250000/250000 [==============================] - 5s 18us/step - loss: 0.68
36 - acc: 0.6386 - val_loss: 0.6518 - val_acc: 0.6628
Epoch 13/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.67
91 - acc: 0.6424 - val_loss: 0.6376 - val_acc: 0.6850
Epoch 14/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.67
66 - acc: 0.6438 - val_loss: 0.6373 - val_acc: 0.6831
Epoch 15/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.67
36 - acc: 0.6472 - val_loss: 0.6301 - val_acc: 0.6914
Epoch 16/100
250000/250000 [==============================] - 4s 18us/step - loss: 0.67
14 - acc: 0.6478 - val_loss: 0.6414 - val_acc: 0.6546
Epoch 17/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.66
88 - acc: 0.6507 - val_loss: 0.6441 - val_acc: 0.6759
Epoch 18/100
250000/250000 [==============================] - 5s 19us/step - loss: 0.66
73 - acc: 0.6511 - val_loss: 0.6540 - val_acc: 0.6794
Epoch 19/100
250000/250000 [==============================] - 5s 20us/step - loss: 0.66
63 - acc: 0.6541 - val_loss: 0.6270 - val_acc: 0.6891
Epoch 20/100
250000/250000 [==============================] - 4s 17us/step - loss: 0.66
54 - acc: 0.6546 - val_loss: 0.6294 - val_acc: 0.6910
```

```
Epoch 21/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.66
39 - acc: 0.6549 - val_loss: 0.6287 - val_acc: 0.6872
Epoch 22/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.66
31 - acc: 0.6570 - val_loss: 0.6587 - val_acc: 0.6610
Epoch 23/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.66
28 - acc: 0.6563 - val_loss: 0.6303 - val_acc: 0.6886
Epoch 24/100
250000/250000 [==============================] - 4s 17us/step - loss: 0.66
05 - acc: 0.6589 - val_loss: 0.6468 - val_acc: 0.6755
Epoch 25/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.66
09 - acc: 0.6588 - val_loss: 0.6430 - val_acc: 0.6775
Epoch 26/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.65
99 - acc: 0.6595 - val_loss: 0.6255 - val_acc: 0.6895
Epoch 27/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.65
81 - acc: 0.6611 - val_loss: 0.6207 - val_acc: 0.6948
Epoch 28/100
250000/250000 [==============================] - 4s 17us/step - loss: 0.65
75 - acc: 0.6624 - val_loss: 0.6350 - val_acc: 0.6821
Epoch 29/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.65
67 - acc: 0.6620 - val_loss: 0.6263 - val_acc: 0.6904
Epoch 30/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.65
53 - acc: 0.6643 - val_loss: 0.6236 - val_acc: 0.6963
Epoch 31/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.65
43 - acc: 0.6650 - val_loss: 0.6164 - val_acc: 0.6967
Epoch 32/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.65
37 - acc: 0.6648 - val_loss: 0.6248 - val_acc: 0.6928
Epoch 33/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.65
38 - acc: 0.6651 - val_loss: 0.6210 - val_acc: 0.6983
Epoch 34/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.65
28 - acc: 0.6669 - val_loss: 0.6254 - val_acc: 0.6912
Epoch 35/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.65
18 - acc: 0.6677 - val_loss: 0.6307 - val_acc: 0.6865
Epoch 36/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.65
07 - acc: 0.6679 - val_loss: 0.6225 - val_acc: 0.6976
Epoch 37/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.65
06 - acc: 0.6679 - val_loss: 0.6211 - val_acc: 0.6981
Epoch 38/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.65
03 - acc: 0.6692 - val_loss: 0.6261 - val_acc: 0.6842
Epoch 39/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.65
03 - acc: 0.6691 - val_loss: 0.6478 - val_acc: 0.6699
Epoch 40/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.65
18 - acc: 0.6671 - val_loss: 0.6144 - val_acc: 0.6994
Epoch 41/100
```

```
250000/250000 [==============================] - 4s 17us/step - loss: 0.64
96 - acc: 0.6695 - val_loss: 0.6201 - val_acc: 0.6918
Epoch 42/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.64
85 - acc: 0.6701 - val_loss: 0.6110 - val_acc: 0.7073
Epoch 43/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.64
77 - acc: 0.6712 - val_loss: 0.6074 - val_acc: 0.7063
Epoch 44/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.64
83 - acc: 0.6705 - val_loss: 0.6190 - val_acc: 0.6941
Epoch 45/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.64
70 - acc: 0.6716 - val_loss: 0.6080 - val_acc: 0.7095
Epoch 46/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.64
73 - acc: 0.6717 - val_loss: 0.6179 - val_acc: 0.6978
Epoch 47/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.64
67 - acc: 0.6717 - val_loss: 0.6103 - val_acc: 0.7040
Epoch 48/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.64
63 - acc: 0.6727 - val_loss: 0.6343 - val_acc: 0.6864
Epoch 49/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.64
58 - acc: 0.6731 - val_loss: 0.6092 - val_acc: 0.7051
Epoch 50/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.64
55 - acc: 0.6732 - val_loss: 0.6046 - val_acc: 0.7133
Epoch 51/100
250000/250000 [==============================] - 4s 16us/step - loss: 0.64
46 - acc: 0.6738 - val_loss: 0.6098 - val_acc: 0.7046
Epoch 52/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.64
39 - acc: 0.6748 - val_loss: 0.6054 - val_acc: 0.7144
Epoch 53/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.64
44 - acc: 0.6743 - val_loss: 0.6099 - val_acc: 0.7057
Epoch 54/100
250000/250000 [==============================] - 4s 17us/step - loss: 0.64
45 - acc: 0.6743 - val_loss: 0.6143 - val_acc: 0.7049
Epoch 55/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.64
39 - acc: 0.6741 - val_loss: 0.6094 - val_acc: 0.7114
Epoch 56/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.64
40 - acc: 0.6752 - val_loss: 0.6031 - val_acc: 0.7141
Epoch 57/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.64
30 - acc: 0.6757 - val_loss: 0.5943 - val_acc: 0.7206
Epoch 58/100
250000/250000 [==============================] - 4s 17us/step - loss: 0.64
23 - acc: 0.6764 - val_loss: 0.6047 - val_acc: 0.7104
Epoch 59/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.64
24 - acc: 0.6765 - val_loss: 0.6045 - val_acc: 0.7155
Epoch 60/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.64
21 - acc: 0.6761 - val_loss: 0.5964 - val_acc: 0.7190
Epoch 61/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.64
```

```
21 - acc: 0.6761 - val_loss: 0.5956 - val_acc: 0.7199
Epoch 62/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.64
12 - acc: 0.6772 - val_loss: 0.6126 - val_acc: 0.7117
Epoch 63/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.64
09 - acc: 0.6776 - val_loss: 0.6099 - val_acc: 0.7075
Epoch 64/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.64
06 - acc: 0.6773 - val_loss: 0.6031 - val_acc: 0.7118
Epoch 65/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.64
04 - acc: 0.6782 - val_loss: 0.6008 - val_acc: 0.7200
Epoch 66/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.63
96 - acc: 0.6781 - val_loss: 0.6003 - val_acc: 0.7154
Epoch 67/100
250000/250000 [==============================] - 4s 18us/step - loss: 0.64
03 - acc: 0.6783 - val_loss: 0.6120 - val_acc: 0.7039
Epoch 68/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.63
91 - acc: 0.6790 - val_loss: 0.5979 - val_acc: 0.7102
Epoch 69/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.63
88 - acc: 0.6790 - val_loss: 0.6034 - val_acc: 0.7131
Epoch 70/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.63
99 - acc: 0.6785 - val_loss: 0.6111 - val_acc: 0.7102
Epoch 71/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.63
84 - acc: 0.6788 - val_loss: 0.5945 - val_acc: 0.7190
Epoch 72/100
250000/250000 [==============================] - 4s 15us/step - loss: 0.63
87 - acc: 0.6790 - val_loss: 0.6072 - val_acc: 0.7066
Epoch 73/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.63
84 - acc: 0.6794 - val_loss: 0.5998 - val_acc: 0.7156
Epoch 74/100
250000/250000 [==============================] - 3s 14us/step - loss: 0.63
79 - acc: 0.6798 - val_loss: 0.6071 - val_acc: 0.7048
Epoch 75/100
250000/250000 [==============================] - 3s 13us/step - loss: 0.63
87 - acc: 0.6785 - val_loss: 0.5970 - val_acc: 0.7173
Epoch 76/100
250000/250000 [==============================] - 6s 23us/step - loss: 0.63
76 - acc: 0.6795 - val_loss: 0.5979 - val_acc: 0.7141
Epoch 77/100
250000/250000 [==============================] - 6s 24us/step - loss: 0.63
72 - acc: 0.6807 - val_loss: 0.5939 - val_acc: 0.7185
Epoch 78/100
250000/250000 [==============================] - 7s 28us/step - loss: 0.63
75 - acc: 0.6804 - val_loss: 0.5897 - val_acc: 0.7233
Epoch 79/100
250000/250000 [==============================] - 8s 31us/step - loss: 0.63
76 - acc: 0.6795 - val_loss: 0.5987 - val_acc: 0.7135
Epoch 80/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.63
59 - acc: 0.6808 - val_loss: 0.6020 - val_acc: 0.7119
Epoch 81/100
250000/250000 [==============================] - 8s 32us/step - loss: 0.63
78 - acc: 0.6791 - val_loss: 0.5957 - val_acc: 0.7188
```

```
Epoch 82/100
250000/250000 [==============================] - 5s 19us/step - loss: 0.63
62 - acc: 0.6806 - val_loss: 0.6311 - val_acc: 0.7032
Epoch 83/100
250000/250000 [==============================] - 3s 11us/step - loss: 0.63
66 - acc: 0.6800 - val_loss: 0.5927 - val_acc: 0.7195
Epoch 84/100
250000/250000 [==============================] - 4s 14us/step - loss: 0.63
61 - acc: 0.6807 - val_loss: 0.5956 - val_acc: 0.7203
Epoch 85/100
250000/250000 [==============================] - 5s 19us/step - loss: 0.63
52 - acc: 0.6809 - val_loss: 0.5950 - val_acc: 0.7215
Epoch 86/100
250000/250000 [==============================] - 3s 12us/step - loss: 0.63
48 - acc: 0.6816 - val_loss: 0.5892 - val_acc: 0.7234
Epoch 87/100
250000/250000 [==============================] - 3s 10us/step - loss: 0.63
42 - acc: 0.6820 - val_loss: 0.5939 - val_acc: 0.7167
Epoch 88/100
250000/250000 [==============================] - 3s 10us/step - loss: 0.63
62 - acc: 0.6803 - val_loss: 0.6072 - val_acc: 0.7026
Epoch 89/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.634
0 - acc: 0.6824 - val_loss: 0.5962 - val_acc: 0.7180
Epoch 90/100
250000/250000 [==============================] - 3s 10us/step - loss: 0.63
39 - acc: 0.6824 - val_loss: 0.5899 - val_acc: 0.7224
Epoch 91/100
250000/250000 [==============================] - 2s 10us/step - loss: 0.63
43 - acc: 0.6828 - val_loss: 0.5947 - val_acc: 0.7213
Epoch 92/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.635
0 - acc: 0.6813 - val_loss: 0.5853 - val_acc: 0.7259
Epoch 93/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.634
8 - acc: 0.6814 - val_loss: 0.6022 - val_acc: 0.7136
Epoch 94/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.634
6 - acc: 0.6816 - val_loss: 0.5923 - val_acc: 0.7174
Epoch 95/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.632
8 - acc: 0.6828 - val_loss: 0.6083 - val_acc: 0.7067
Epoch 96/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.632
3 - acc: 0.6835 - val_loss: 0.5960 - val_acc: 0.7159
Epoch 97/100
250000/250000 [==============================] - 2s 10us/step - loss: 0.63
35 - acc: 0.6815 - val_loss: 0.5911 - val_acc: 0.7228
Epoch 98/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.632
6 - acc: 0.6830 - val_loss: 0.5829 - val_acc: 0.7267
Epoch 99/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.633
4 - acc: 0.6824 - val_loss: 0.6013 - val_acc: 0.7094
Epoch 100/100
250000/250000 [==============================] - 2s 9us/step - loss: 0.633
0 - acc: 0.6827 - val_loss: 0.5893 - val_acc: 0.7233
```

Out[19]:

```
<keras.callbacks.History at 0x12686f9d208>
```

Avaliando o desempenho da rede.

In [20]:

```
score = autoencoder.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.5893179732513427
Test accuracy: 0.72334
```