In [1]:

```
# imports necessários
import os
import string

import cv2
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Model
from keras.layers import Input, Flatten, Dense, Dropout, Conv2D, MaxPooling2D, BatchNormalization
from keras.backend import clear_session
```

```
Using TensorFlow backend.
```

In [54]:

```
%matplotlib inline
```

In [2]:

```
# constantes
DIR_DATASET = './dataset/samples/samples/'  # local do dataset
LEN_DATASET = 1070  # 1070 imagens no dataset
LEN_CAPTCH = 5  # 5 letras/números

TARGET_WORDS = string.ascii_lowercase + string.digits
LEN_WORDS = len(TARGET_WORDS)

IMG_SHAPE = (50, 200, 1)  # input shape do modelo
TRAIN_SIZE = int(0.92 * LEN_DATASET) # dataset de treino com 92% do dataset total
```

In [4]:

```
# remove os modelos predefinidos
clear_session()

# input do modelo
input_ = Input(shape=IMG_SHAPE)

# camadas de convolução
conv1 = Conv2D(16, kernel_size=(3, 3), padding='same', activation='relu')(input_)
maxp1 = MaxPooling2D(pool_size=(2, 2), padding='same')(conv1)
conv2 = Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu')(maxp1)
maxp2 = MaxPooling2D(pool_size=(2, 2), padding='same')(conv2)
conv3 = Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu')(maxp2)
batc1 = BatchNormalization()(conv3)
maxp3 = MaxPooling2D(pool_size=(2, 2), padding='same')(batc1)

# camadas densas
flat1 = Flatten()(maxp3)  # 3D para 1D

# densa para o primeiro caractere
dens1 = Dense(64, activation='relu')(flat1)
drop1 = Dropout(0.5)(dens1)
outp1 = Dense(LEN_WORDS, activation='sigmoid')(drop1)

# densa para o segundo caractere
dens2 = Dense(64, activation='relu')(flat1)
drop2 = Dropout(0.5)(dens2)
outp2 = Dense(LEN_WORDS, activation='sigmoid')(drop2)

# densa para o terceiro caractere
dens3 = Dense(64, activation='relu')(flat1)
drop3 = Dropout(0.5)(dens3)
outp3 = Dense(LEN_WORDS, activation='sigmoid')(drop3)

# densa para o quarto caractere
dens4 = Dense(64, activation='relu')(flat1)
drop4 = Dropout(0.5)(dens4)
outp4 = Dense(LEN_WORDS, activation='sigmoid')(drop4)

# densa para o quinto caractere
dens5 = Dense(64, activation='relu')(flat1)
drop5 = Dropout(0.5)(dens5)
outp5 = Dense(LEN_WORDS, activation='sigmoid')(drop5)
```

In [5]:

```
# construção do modelo
model = Model(input_, [outp1, outp2, outp3, outp4, outp5])

# compilando o modelo
model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=["accuracy"])
```

In [6]:

```
# estrutura da rede
```

```
model.summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to                     
==================================================================================================
input_1 (InputLayer)            (None, 50, 200, 1)   0                                            
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 50, 200, 16)  160         input_1[0][0]                    
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 25, 100, 16)  0           conv2d_1[0][0]                   
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 25, 100, 32)  4640        max_pooling2d_1[0][0]            
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 13, 50, 32)   0           conv2d_2[0][0]                   
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 13, 50, 32)   9248        max_pooling2d_2[0][0]            
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 13, 50, 32)   128         conv2d_3[0][0]                   
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 7, 25, 32)    0           batch_normalization_1[0][0]      
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 5600)         0           max_pooling2d_3[0][0]            
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           358464      flatten_1[0][0]                  
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 64)           358464      flatten_1[0][0]                  
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 64)           358464      flatten_1[0][0]                  
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 64)           358464      flatten_1[0][0]                  
__________________________________________________________________________________________________
dense_9 (Dense)                 (None, 64)           358464      flatten_1[0][0]                  
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 64)           0           dense_1[0][0]                    
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 64)           0           dense_3[0][0]                    
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 64)           0           dense_5[0][0]                    
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 64)           0           dense_7[0][0]                    
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 64)           0           dense_9[0][0]                    
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 36)           2340        dropout_1[0][0]                  
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 36)           2340        dropout_2[0][0]                  
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 36)           2340        dropout_3[0][0]                  
__________________________________________________________________________________________________
dense_8 (Dense)                 (None, 36)           2340        dropout_4[0][0]                  
__________________________________________________________________________________________________
dense_10 (Dense)                (None, 36)           2340        dropout_5[0][0]                  
==================================================================================================
Total params: 1,818,196
Trainable params: 1,818,132
Non-trainable params: 64
__________________________________________________________________________________________________
```

In [7]:

```
# carregando e processando o dataset

# dados vazios para preencher no laço for
X = np.zeros((LEN_DATASET,) + IMG_SHAPE)
Y = np.zeros((LEN_CAPTCH, LEN_DATASET, LEN_WORDS))

for index, image_name in enumerate(os.listdir(DIR_DATASET)):
    image_path = os.path.join(DIR_DATASET, image_name)  # diretório da imagem

    # pegando o nome da imagem que é o valor do captch
    image_target, _ = os.path.splitext(image_name)

    # carregando imagem, convertendo para cinza, pixel da imagem entre 0 e 1 e adicionando uma dim
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    image = image / 255.
    image = image[:, :, np.newaxis]

    # preenchendo com 1 onde tem a letra/número do output
    target_encode = np.zeros((LEN_CAPTCH, LEN_WORDS))
    for line, word in enumerate(image_target):
        target_encode[line, TARGET_WORDS.index(word)] = 1

    X[index] = image
    Y[:, index] = target_encode
```

In [8]:

```
# separação dos dados de treino e de teste
x_train, x_test = X[:TRAIN_SIZE], X[TRAIN_SIZE:]
y_train, y_test = Y[:, :TRAIN_SIZE], Y[:, TRAIN_SIZE:]

x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[8]:

```
((984, 50, 200, 1), (5, 984, 36), (86, 50, 200, 1), (5, 86, 36))
```

In [9]:

```
# treinamento do modelo
history = model.fit(x_train, [y_train[0], y_train[1], y_train[2], y_train[3], y_train[4]],
                    batch_size=32,  # treinar a cada 32 dados
                    epochs=30,  # treinar 30 vezes
                    validation_split=0.2,  # 20% dos dados utilizar para validar o modelo
                    verbose=1  # apenas printa a barra do treinamento
                   )
```

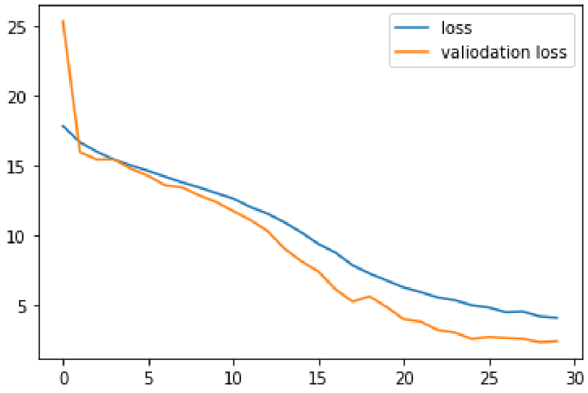
```
WARNING:tensorflow:From /home/sena/Documents/program-files/anaconda3/envs/captch/lib/python3.6/site
-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 787 samples, validate on 197 samples
Epoch 1/30
787/787 [==============================] - 9s 11ms/step - loss: 17.8486 - dense_2_loss: 3.5491 - de
nse_4_loss: 3.4678 - dense_6_loss: 3.6070 - dense_8_loss: 3.6197 - dense_10_loss: 3.6051 - dense_2_
acc: 0.0534 - dense_4_acc: 0.0445 - dense_6_acc: 0.0368 - dense_8_acc: 0.0496 - dense_10_acc: 0.045
7 - val_loss: 25.3514 - val_dense_2_loss: 4.9354 - val_dense_4_loss: 5.6727 - val_dense_6_loss: 4.5
626 - val_dense_8_loss: 4.8101 - val_dense_10_loss: 5.3704 - val_dense_2_acc: 0.0406 - val_dense_4_
acc: 0.0457 - val_dense_6_acc: 0.0000e+00 - val_dense_8_acc: 0.0508 - val_dense_10_acc: 0.0355
Epoch 2/30
787/787 [==============================] - 7s 9ms/step - loss: 16.6783 - dense_2_loss: 3.2660 - den
se_4_loss: 3.2514 - dense_6_loss: 3.3933 - dense_8_loss: 3.3996 - dense_10_loss: 3.3680 - dense_2_a
cc: 0.0584 - dense_4_acc: 0.0546 - dense_6_acc: 0.0534 - dense_8_acc: 0.0546 - dense_10_acc: 0.0610
- val_loss: 15.9715 - val_dense_2_loss: 3.1441 - val_dense_4_loss: 3.1133 - val_dense_6_loss: 3.188
9 - val_dense_8_loss: 3.3201 - val_dense_10_loss: 3.2050 - val_dense_2_acc: 0.0558 - val_dense_4_ac
c: 0.0711 - val_dense_6_acc: 0.1015 - val_dense_8_acc: 0.0609 - val_dense_10_acc: 0.0609
Epoch 3/30
787/787 [==============================] - 7s 9ms/step - loss: 16.0044 - dense_2_loss: 3.1255 - den
se_4_loss: 3.1418 - dense_6_loss: 3.2513 - dense_8_loss: 3.2664 - dense_10_loss: 3.2194 - dense_2_a
cc: 0.0750 - dense_4_acc: 0.0572 - dense_6_acc: 0.0775 - dense_8_acc: 0.0826 - dense_10_acc: 0.0762
- val_loss: 15.4445 - val_dense_2_loss: 2.9802 - val_dense_4_loss: 3.0339 - val_dense_6_loss: 3.137
1 - val_dense_8_loss: 3.2203 - val_dense_10_loss: 3.0730 - val_dense_2_acc: 0.1015 - val_dense_4_ac
c: 0.0406 - val_dense_6_acc: 0.1218 - val_dense_8_acc: 0.0964 - val_dense_10_acc: 0.1218
Epoch 4/30
787/787 [==============================] - 7s 9ms/step - loss: 15.4395 - dense_2_loss: 3.0191 - den
se_4_loss: 3.0540 - dense_6_loss: 3.1282 - dense_8_loss: 3.1491 - dense_10_loss: 3.0890 - dense_2_a
cc: 0.1017 - dense_4_acc: 0.0635 - dense_6_acc: 0.0762 - dense_8_acc: 0.0712 - dense_10_acc: 0.0699
- val_loss: 15.4577 - val_dense_2_loss: 3.0305 - val_dense_4_loss: 3.0755 - val_dense_6_loss: 3.148
9 - val_dense_8_loss: 3.1856 - val_dense_10_loss: 3.0172 - val_dense_2_acc: 0.1320 - val_dense_4_ac
c: 0.0406 - val_dense_6_acc: 0.1371 - val_dense_8_acc: 0.0914 - val_dense_10_acc: 0.1168
Epoch 5/30
787/787 [==============================] - 7s 9ms/step - loss: 15.0224 - dense_2_loss: 2.9274 - den
se_4_loss: 2.9605 - dense_6_loss: 3.0787 - dense_8_loss: 3.0494 - dense_10_loss: 3.0065 - dense_2_a
cc: 0.0978 - dense_4_acc: 0.0826 - dense_6_acc: 0.0813 - dense_8_acc: 0.0915 - dense_10_acc: 0.0699
- val_loss: 14.7810 - val_dense_2_loss: 2.8118 - val_dense_4_loss: 2.9144 - val_dense_6_loss: 3.038
1 - val_dense_8_loss: 3.0902 - val_dense_10_loss: 2.9265 - val_dense_2_acc: 0.1117 - val_dense_4_ac
c: 0.0964 - val_dense_6_acc: 0.1320 - val_dense_8_acc: 0.1269 - val_dense_10_acc: 0.0711
Epoch 6/30
787/787 [==============================] - 7s 9ms/step - loss: 14.6390 - dense_2_loss: 2.8430 - den
se_4_loss: 2.9136 - dense_6_loss: 2.9948 - dense_8_loss: 2.9977 - dense_10_loss: 2.8899 - dense_2_a
cc: 0.0851 - dense_4_acc: 0.0826 - dense_6_acc: 0.1004 - dense_8_acc: 0.1042 - dense_10_acc: 0.0915
- val_loss: 14.2792 - val_dense_2_loss: 2.7724 - val_dense_4_loss: 2.8336 - val_dense_6_loss: 2.957
9 - val_dense_8_loss: 2.8829 - val_dense_10_loss: 2.8324 - val_dense_2_acc: 0.1726 - val_dense_4_ac
c: 0.1066 - val_dense_6_acc: 0.2132 - val_dense_8_acc: 0.1675 - val_dense_10_acc: 0.1168
Epoch 7/30
787/787 [==============================] - 7s 9ms/step - loss: 14.2144 - dense_2_loss: 2.7264 - den
se_4_loss: 2.8452 - dense_6_loss: 2.9139 - dense_8_loss: 2.9003 - dense_10_loss: 2.8286 - dense_2_a
cc: 0.1182 - dense_4_acc: 0.0788 - dense_6_acc: 0.1017 - dense_8_acc: 0.1067 - dense_10_acc: 0.1067
- val_loss: 13.6014 - val_dense_2_loss: 2.5955 - val_dense_4_loss: 2.7256 - val_dense_6_loss: 2.781
8 - val_dense_8_loss: 2.7577 - val_dense_10_loss: 2.7409 - val_dense_2_acc: 0.1218 - val_dense_4_ac
c: 0.0761 - val_dense_6_acc: 0.2183 - val_dense_8_acc: 0.1726 - val_dense_10_acc: 0.1218
Epoch 8/30
787/787 [==============================] - 7s 9ms/step - loss: 13.8077 - dense_2_loss: 2.6075 - den
se_4_loss: 2.7882 - dense_6_loss: 2.8154 - dense_8_loss: 2.8144 - dense_10_loss: 2.7823 - dense_2_a
cc: 0.1423 - dense_4_acc: 0.0915 - dense_6_acc: 0.1182 - dense_8_acc: 0.1169 - dense_10_acc: 0.0978
- val_loss: 13.4490 - val_dense_2_loss: 2.4815 - val_dense_4_loss: 2.7009 - val_dense_6_loss: 2.797
9 - val_dense_8_loss: 2.7049 - val_dense_10_loss: 2.7637 - val_dense_2_acc: 0.1421 - val_dense_4_ac
c: 0.1726 - val_dense_6_acc: 0.1269 - val_dense_8_acc: 0.1472 - val_dense_10_acc: 0.1421
Epoch 9/30
787/787 [==============================] - 7s 9ms/step - loss: 13.4495 - dense_2_loss: 2.4874 - den
se_4_loss: 2.7436 - dense_6_loss: 2.7726 - dense_8_loss: 2.7520 - dense_10_loss: 2.6939 - dense_2_a
cc: 0.1398 - dense_4_acc: 0.0978 - dense_6_acc: 0.1220 - dense_8_acc: 0.1067 - dense_10_acc: 0.1258
- val_loss: 12.8808 - val_dense_2_loss: 2.3328 - val_dense_4_loss: 2.5793 - val_dense_6_loss: 2.628
6 - val_dense_8_loss: 2.6234 - val_dense_10_loss: 2.7168 - val_dense_2_acc: 0.1777 - val_dense_4_ac
```

```
c: 0.1878 - val_dense_6_acc: 0.2081 - val_dense_8_acc: 0.1421 - val_dense_10_acc: 0.1421
Epoch 10/30
787/787 [==============================] - 7s 9ms/step - loss: 13.0445 - dense_2_loss: 2.3948 - den
se_4_loss: 2.6597 - dense_6_loss: 2.6859 - dense_8_loss: 2.6978 - dense_10_loss: 2.6062 - dense_2_a
cc: 0.1372 - dense_4_acc: 0.0864 - dense_6_acc: 0.1233 - dense_8_acc: 0.1499 - dense_10_acc: 0.1410
- val_loss: 12.3969 - val_dense_2_loss: 2.1598 - val_dense_4_loss: 2.5265 - val_dense_6_loss: 2.527
2 - val_dense_8_loss: 2.6054 - val_dense_10_loss: 2.5781 - val_dense_2_acc: 0.1980 - val_dense_4_ac
c: 0.1371 - val_dense_6_acc: 0.1472 - val_dense_8_acc: 0.0609 - val_dense_10_acc: 0.1574
Epoch 11/30
787/787 [==============================] - 7s 9ms/step - loss: 12.6475 - dense_2_loss: 2.2307 - den
se_4_loss: 2.5815 - dense_6_loss: 2.5926 - dense_8_loss: 2.7100 - dense_10_loss: 2.5326 - dense_2_a
cc: 0.1804 - dense_4_acc: 0.0953 - dense_6_acc: 0.1321 - dense_8_acc: 0.1487 - dense_10_acc: 0.1385
- val_loss: 11.7458 - val_dense_2_loss: 1.9123 - val_dense_4_loss: 2.3632 - val_dense_6_loss: 2.447
9 - val_dense_8_loss: 2.5664 - val_dense_10_loss: 2.4561 - val_dense_2_acc: 0.1320 - val_dense_4_ac
c: 0.1523 - val_dense_6_acc: 0.2081 - val_dense_8_acc: 0.1726 - val_dense_10_acc: 0.1472
Epoch 12/30
787/787 [==============================] - 9s 11ms/step - loss: 12.0428 - dense_2_loss: 2.0637 - de
nse_4_loss: 2.5037 - dense_6_loss: 2.5045 - dense_8_loss: 2.5615 - dense_10_loss: 2.4093 - dense_2_
acc: 0.2020 - dense_4_acc: 0.1271 - dense_6_acc: 0.1372 - dense_8_acc: 0.1512 - dense_10_acc: 0.146
1 - val_loss: 11.1141 - val_dense_2_loss: 1.6703 - val_dense_4_loss: 2.2866 - val_dense_6_loss: 2.2
893 - val_dense_8_loss: 2.5239 - val_dense_10_loss: 2.3440 - val_dense_2_acc: 0.1777 - val_dense_4_
acc: 0.1320 - val_dense_6_acc: 0.2284 - val_dense_8_acc: 0.1980 - val_dense_10_acc: 0.1523
Epoch 13/30
787/787 [==============================] - 7s 9ms/step - loss: 11.5680 - dense_2_loss: 1.9218 - den
se_4_loss: 2.3829 - dense_6_loss: 2.4030 - dense_8_loss: 2.5280 - dense_10_loss: 2.3322 - dense_2_a
cc: 0.2859 - dense_4_acc: 0.1347 - dense_6_acc: 0.1576 - dense_8_acc: 0.1715 - dense_10_acc: 0.1563
- val_loss: 10.3262 - val_dense_2_loss: 1.5062 - val_dense_4_loss: 2.1276 - val_dense_6_loss: 2.115
1 - val_dense_8_loss: 2.4272 - val_dense_10_loss: 2.1500 - val_dense_2_acc: 0.4772 - val_dense_4_ac
c: 0.1624 - val_dense_6_acc: 0.3299 - val_dense_8_acc: 0.2335 - val_dense_10_acc: 0.0964
Epoch 14/30
787/787 [==============================] - 7s 9ms/step - loss: 10.9452 - dense_2_loss: 1.7394 - den
se_4_loss: 2.2451 - dense_6_loss: 2.3473 - dense_8_loss: 2.4120 - dense_10_loss: 2.2014 - dense_2_a
cc: 0.3469 - dense_4_acc: 0.1512 - dense_6_acc: 0.1499 - dense_8_acc: 0.2020 - dense_10_acc: 0.1919
- val_loss: 9.0640 - val_dense_2_loss: 1.0576 - val_dense_4_loss: 1.7936 - val_dense_6_loss: 1.9753
- val_dense_8_loss: 2.2881 - val_dense_10_loss: 1.9494 - val_dense_2_acc: 0.7614 - val_dense_4_acc:
0.3604 - val_dense_6_acc: 0.2843 - val_dense_8_acc: 0.3401 - val_dense_10_acc: 0.1472
Epoch 15/30
787/787 [==============================] - 7s 9ms/step - loss: 10.2195 - dense_2_loss: 1.5574 - den
se_4_loss: 2.0828 - dense_6_loss: 2.2585 - dense_8_loss: 2.2950 - dense_10_loss: 2.0258 - dense_2_a
cc: 0.4295 - dense_4_acc: 0.2198 - dense_6_acc: 0.1982 - dense_8_acc: 0.2084 - dense_10_acc: 0.2084
- val_loss: 8.1513 - val_dense_2_loss: 0.8386 - val_dense_4_loss: 1.5124 - val_dense_6_loss: 1.8605
- val_dense_8_loss: 2.1486 - val_dense_10_loss: 1.7913 - val_dense_2_acc: 0.6954 - val_dense_4_acc:
0.5381 - val_dense_6_acc: 0.3503 - val_dense_8_acc: 0.1726 - val_dense_10_acc: 0.1218
Epoch 16/30
787/787 [==============================] - 7s 9ms/step - loss: 9.3910 - dense_2_loss: 1.3365 - dens
e_4_loss: 1.7956 - dense_6_loss: 2.1068 - dense_8_loss: 2.1684 - dense_10_loss: 1.9837 - dense_2_ac
c: 0.5121 - dense_4_acc: 0.3240 - dense_6_acc: 0.2554 - dense_8_acc: 0.2325 - dense_10_acc: 0.2135
- val_loss: 7.4175 - val_dense_2_loss: 0.5581 - val_dense_4_loss: 1.3286 - val_dense_6_loss: 1.7528
- val_dense_8_loss: 2.0036 - val_dense_10_loss: 1.7744 - val_dense_2_acc: 0.8579 - val_dense_4_acc:
0.6041 - val_dense_6_acc: 0.3452 - val_dense_8_acc: 0.2792 - val_dense_10_acc: 0.1827
Epoch 17/30
787/787 [==============================] - 7s 9ms/step - loss: 8.7691 - dense_2_loss: 1.1876 - dens
e_4_loss: 1.6598 - dense_6_loss: 2.0301 - dense_8_loss: 2.0067 - dense_10_loss: 1.8849 - dense_2_ac
c: 0.5705 - dense_4_acc: 0.3825 - dense_6_acc: 0.2465 - dense_8_acc: 0.2834 - dense_10_acc: 0.2859
- val_loss: 6.1474 - val_dense_2_loss: 0.4008 - val_dense_4_loss: 0.8411 - val_dense_6_loss: 1.6233
- val_dense_8_loss: 1.7972 - val_dense_10_loss: 1.4850 - val_dense_2_acc: 0.9188 - val_dense_4_acc:
0.7563 - val_dense_6_acc: 0.4315 - val_dense_8_acc: 0.3706 - val_dense_10_acc: 0.5076
Epoch 18/30
787/787 [==============================] - 7s 9ms/step - loss: 7.8677 - dense_2_loss: 1.0019 - dens
e_4_loss: 1.4387 - dense_6_loss: 1.9013 - dense_8_loss: 1.8198 - dense_10_loss: 1.7060 - dense_2_ac
c: 0.6429 - dense_4_acc: 0.4295 - dense_6_acc: 0.2948 - dense_8_acc: 0.3304 - dense_10_acc: 0.3837
- val_loss: 5.2617 - val_dense_2_loss: 0.1808 - val_dense_4_loss: 0.6742 - val_dense_6_loss: 1.2788
- val_dense_8_loss: 1.5724 - val_dense_10_loss: 1.5555 - val_dense_2_acc: 0.9695 - val_dense_4_acc:
0.8426 - val_dense_6_acc: 0.5533 - val_dense_8_acc: 0.4873 - val_dense_10_acc: 0.5228
Epoch 19/30
787/787 [==============================] - 7s 9ms/step - loss: 7.2726 - dense_2_loss: 0.9376 - dens
e_4_loss: 1.2906 - dense_6_loss: 1.8052 - dense_8_loss: 1.6877 - dense_10_loss: 1.5516 - dense_2_ac
c: 0.6645 - dense_4_acc: 0.5222 - dense_6_acc: 0.3329 - dense_8_acc: 0.3952 - dense_10_acc: 0.4409
- val_loss: 5.6296 - val_dense_2_loss: 0.2296 - val_dense_4_loss: 0.7296 - val_dense_6_loss: 1.2968
- val_dense_8_loss: 2.0424 - val_dense_10_loss: 1.3312 - val_dense_2_acc: 0.9188 - val_dense_4_acc:
0.7868 - val_dense_6_acc: 0.5381 - val_dense_8_acc: 0.4975 - val_dense_10_acc: 0.6091
Epoch 20/30
787/787 [==============================] - 7s 9ms/step - loss: 6.7770 - dense_2_loss: 0.8278 - dens
e_4_loss: 1.2091 - dense_6_loss: 1.7272 - dense_8_loss: 1.5555 - dense_10_loss: 1.4574 - dense_2_ac
c: 0.6785 - dense_4_acc: 0.5286 - dense_6_acc: 0.3494 - dense_8_acc: 0.4307 - dense_10_acc: 0.4740
- val_loss: 4.8744 - val_dense_2_loss: 0.2272 - val_dense_4_loss: 0.5135 - val_dense_6_loss: 1.1348
- val_dense_8_loss: 1.6922 - val_dense_10_loss: 1.3067 - val_dense_2_acc: 0.9340 - val_dense_4_acc:
0.8579 - val_dense_6_acc: 0.5381 - val_dense_8_acc: 0.5838 - val_dense_10_acc: 0.6548
Epoch 21/30
787/787 [==============================] - 7s 9ms/step - loss: 6.2805 - dense_2_loss: 0.7780 - dens
e_4_loss: 1.1247 - dense_6_loss: 1.5937 - dense_8_loss: 1.4610 - dense_10_loss: 1.3230 - dense_2_ac
c: 0.7065 - dense_4_acc: 0.5616 - dense_6_acc: 0.3748 - dense_8_acc: 0.4600 - dense_10_acc: 0.4905
- val_loss: 4.0032 - val_dense_2_loss: 0.1034 - val_dense_4_loss: 0.4095 - val_dense_6_loss: 0.9718
- val_dense_8_loss: 1.4065 - val_dense_10_loss: 1.1119 - val_dense_2_acc: 0.9797 - val_dense_4_acc:
0.8934 - val_dense_6_acc: 0.7056 - val_dense_8_acc: 0.5939 - val_dense_10_acc: 0.6650
Epoch 22/30
```

```
787/787 [==============================] - 7s 9ms/step - loss: 5.9425 - dense_2_loss: 0.7016 - dens
e_4_loss: 0.9947 - dense_6_loss: 1.5563 - dense_8_loss: 1.4068 - dense_10_loss: 1.2831 - dense_2_ac
c: 0.7255 - dense_4_acc: 0.5845 - dense_6_acc: 0.3825 - dense_8_acc: 0.4574 - dense_10_acc: 0.5133
- val_loss: 3.8220 - val_dense_2_loss: 0.0774 - val_dense_4_loss: 0.3328 - val_dense_6_loss: 0.9708
- val_dense_8_loss: 1.4677 - val_dense_10_loss: 0.9732 - val_dense_2_acc: 0.9848 - val_dense_4_acc:
0.9239 - val_dense_6_acc: 0.6497 - val_dense_8_acc: 0.5939 - val_dense_10_acc: 0.7208
Epoch 23/30
787/787 [==============================] - 7s 9ms/step - loss: 5.5611 - dense_2_loss: 0.6845 - dens
e_4_loss: 0.9571 - dense_6_loss: 1.4917 - dense_8_loss: 1.3386 - dense_10_loss: 1.0892 - dense_2_ac
c: 0.7357 - dense_4_acc: 0.6302 - dense_6_acc: 0.3901 - dense_8_acc: 0.4765 - dense_10_acc: 0.5845
- val_loss: 3.2247 - val_dense_2_loss: 0.1158 - val_dense_4_loss: 0.3201 - val_dense_6_loss: 0.8287
- val_dense_8_loss: 0.9499 - val_dense_10_loss: 1.0102 - val_dense_2_acc: 0.9645 - val_dense_4_acc:
0.9137 - val_dense_6_acc: 0.7310 - val_dense_8_acc: 0.7056 - val_dense_10_acc: 0.7259
Epoch 24/30
787/787 [==============================] - 7s 9ms/step - loss: 5.3589 - dense_2_loss: 0.5769 - dens
e_4_loss: 0.9782 - dense_6_loss: 1.4302 - dense_8_loss: 1.2886 - dense_10_loss: 1.0851 - dense_2_ac
c: 0.7649 - dense_4_acc: 0.6074 - dense_6_acc: 0.4320 - dense_8_acc: 0.5133 - dense_10_acc: 0.5642
- val_loss: 3.0397 - val_dense_2_loss: 0.0778 - val_dense_4_loss: 0.3070 - val_dense_6_loss: 0.7868
- val_dense_8_loss: 0.9789 - val_dense_10_loss: 0.8891 - val_dense_2_acc: 0.9746 - val_dense_4_acc:
0.9086 - val_dense_6_acc: 0.7766 - val_dense_8_acc: 0.6751 - val_dense_10_acc: 0.7310
Epoch 25/30
787/787 [==============================] - 7s 9ms/step - loss: 4.9937 - dense_2_loss: 0.5442 - dens
e_4_loss: 0.8878 - dense_6_loss: 1.3298 - dense_8_loss: 1.2064 - dense_10_loss: 1.0255 - dense_2_ac
c: 0.7802 - dense_4_acc: 0.6429 - dense_6_acc: 0.4803 - dense_8_acc: 0.5400 - dense_10_acc: 0.6201
- val_loss: 2.5845 - val_dense_2_loss: 0.0342 - val_dense_4_loss: 0.2762 - val_dense_6_loss: 0.6649
- val_dense_8_loss: 0.8277 - val_dense_10_loss: 0.7815 - val_dense_2_acc: 0.9848 - val_dense_4_acc:
0.9188 - val_dense_6_acc: 0.7766 - val_dense_8_acc: 0.7513 - val_dense_10_acc: 0.7411
Epoch 26/30
787/787 [==============================] - 7s 9ms/step - loss: 4.8329 - dense_2_loss: 0.5442 - dens
e_4_loss: 0.8368 - dense_6_loss: 1.3562 - dense_8_loss: 1.1231 - dense_10_loss: 0.9727 - dense_2_ac
c: 0.7840 - dense_4_acc: 0.6836 - dense_6_acc: 0.4651 - dense_8_acc: 0.5553 - dense_10_acc: 0.6099
- val_loss: 2.7216 - val_dense_2_loss: 0.0499 - val_dense_4_loss: 0.2837 - val_dense_6_loss: 0.7084
- val_dense_8_loss: 0.9296 - val_dense_10_loss: 0.7499 - val_dense_2_acc: 0.9949 - val_dense_4_acc:
0.9442 - val_dense_6_acc: 0.7919 - val_dense_8_acc: 0.7157 - val_dense_10_acc: 0.7614
Epoch 27/30
787/787 [==============================] - 7s 9ms/step - loss: 4.4985 - dense_2_loss: 0.4840 - dens
e_4_loss: 0.8171 - dense_6_loss: 1.2022 - dense_8_loss: 1.1059 - dense_10_loss: 0.8894 - dense_2_ac
c: 0.7878 - dense_4_acc: 0.6671 - dense_6_acc: 0.5286 - dense_8_acc: 0.5591 - dense_10_acc: 0.6658
- val_loss: 2.6489 - val_dense_2_loss: 0.0401 - val_dense_4_loss: 0.3173 - val_dense_6_loss: 0.6812
- val_dense_8_loss: 0.9400 - val_dense_10_loss: 0.6702 - val_dense_2_acc: 0.9898 - val_dense_4_acc:
0.9239 - val_dense_6_acc: 0.7614 - val_dense_8_acc: 0.7513 - val_dense_10_acc: 0.7868
Epoch 28/30
787/787 [==============================] - 7s 9ms/step - loss: 4.5433 - dense_2_loss: 0.5133 - dens
e_4_loss: 0.8204 - dense_6_loss: 1.2689 - dense_8_loss: 1.0424 - dense_10_loss: 0.8982 - dense_2_ac
c: 0.7980 - dense_4_acc: 0.6684 - dense_6_acc: 0.4968 - dense_8_acc: 0.5909 - dense_10_acc: 0.6353
- val_loss: 2.5936 - val_dense_2_loss: 0.0723 - val_dense_4_loss: 0.2952 - val_dense_6_loss: 0.6991
- val_dense_8_loss: 0.8426 - val_dense_10_loss: 0.6845 - val_dense_2_acc: 0.9848 - val_dense_4_acc:
0.9137 - val_dense_6_acc: 0.7766 - val_dense_8_acc: 0.7563 - val_dense_10_acc: 0.7919
Epoch 29/30
787/787 [==============================] - 7s 9ms/step - loss: 4.1950 - dense_2_loss: 0.4195 - dens
e_4_loss: 0.7656 - dense_6_loss: 1.1902 - dense_8_loss: 1.0019 - dense_10_loss: 0.8178 - dense_2_ac
c: 0.8297 - dense_4_acc: 0.6989 - dense_6_acc: 0.5095 - dense_8_acc: 0.5921 - dense_10_acc: 0.6785
- val_loss: 2.3484 - val_dense_2_loss: 0.0483 - val_dense_4_loss: 0.2964 - val_dense_6_loss: 0.5594
- val_dense_8_loss: 0.7459 - val_dense_10_loss: 0.6984 - val_dense_2_acc: 0.9848 - val_dense_4_acc:
0.9340 - val_dense_6_acc: 0.8122 - val_dense_8_acc: 0.7817 - val_dense_10_acc: 0.7665
Epoch 30/30
787/787 [==============================] - 7s 9ms/step - loss: 4.0956 - dense_2_loss: 0.4592 - dens
e_4_loss: 0.7226 - dense_6_loss: 1.1706 - dense_8_loss: 0.9776 - dense_10_loss: 0.7656 - dense_2_ac
c: 0.8196 - dense_4_acc: 0.7179 - dense_6_acc: 0.5388 - dense_8_acc: 0.5807 - dense_10_acc: 0.6823
- val_loss: 2.4200 - val_dense_2_loss: 0.0483 - val_dense_4_loss: 0.2663 - val_dense_6_loss: 0.6528
- val_dense_8_loss: 0.7768 - val_dense_10_loss: 0.6758 - val_dense_2_acc: 0.9746 - val_dense_4_acc:
0.9340 - val_dense_6_acc: 0.7970 - val_dense_8_acc: 0.7970 - val_dense_10_acc: 0.7868
```

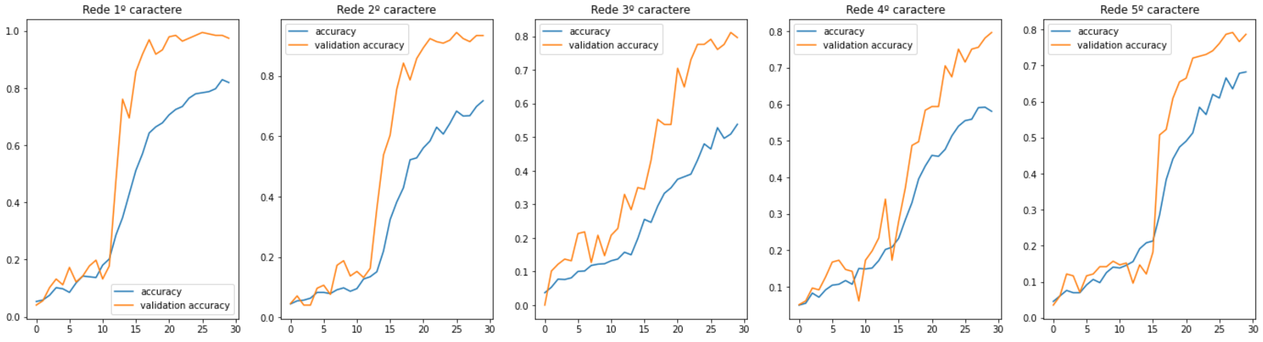
In [55]:

```python
# loss do modelo
fig, axs = plt.subplots(1, 1, figsize=(6, 4))

axs.plot(history.history['loss'], label='loss')
axs.plot(history.history['val_loss'], label='valiodation loss')
axs.legend()

plt.show()
```



In [56]:

```python
# accuracy de cada rede de cada caractere
fig, axs = plt.subplots(1, 5, figsize=(24, 6))

axs[0].set_title('Rede 1º caractere')
axs[0].plot(history.history['dense_2_acc'], label='accuracy')
axs[0].plot(history.history['val_dense_2_acc'], label='validation accuracy')
axs[0].legend()

axs[1].set_title('Rede 2º caractere')
axs[1].plot(history.history['dense_4_acc'], label='accuracy')
axs[1].plot(history.history['val_dense_4_acc'], label='validation accuracy')
axs[1].legend()

axs[2].set_title('Rede 3º caractere')
axs[2].plot(history.history['dense_6_acc'], label='accuracy')
axs[2].plot(history.history['val_dense_6_acc'], label='validation accuracy')
axs[2].legend()

axs[3].set_title('Rede 4º caractere')
axs[3].plot(history.history['dense_8_acc'], label='accuracy')
axs[3].plot(history.history['val_dense_8_acc'], label='validation accuracy')
axs[3].legend()

axs[4].set_title('Rede 5º caractere')
axs[4].plot(history.history['dense_10_acc'], label='accuracy')
axs[4].plot(history.history['val_dense_10_acc'], label='validation accuracy')
axs[4].legend()

plt.show()
```

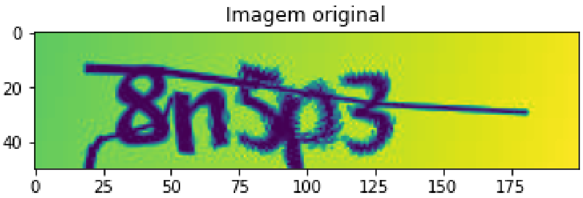


In [57]:

```python
# predição da imagem 8n5p3.png (questão)

DIR_TARGET = './dataset/samples/8n5p3.png'

original_image = cv2.imread(DIR_TARGET, cv2.IMREAD_GRAYSCALE)  # carrega a imagem
target_image = original_image / 255.  # valores da imagem entre 0 e 1

# reshape na imagem para passar pelo modelo
# 50 x 200 para 1 x 50 x 200 x 1
target_image = target_image[np.newaxis, :, :, np.newaxis]

# predição da imagem
predicts = np.array(model.predict(target_image))
predicts = np.reshape(predicts, (5, 36))

# tradução do output do predict para os caracteres do captch
captch = ''
for predict in predicts:
    max_value = predict.max()
    if max_value == 0.:
        captch += '-'
        continue

    max_index = predict.argmax()
    captch += TARGET_WORDS[max_index]
```

In [58]:

```
# visualização da imagem original
fig, axs = plt.subplots(1, 1, figsize=(6, 4))

axs.set_title('Imagem original')
axs.imshow(original_image)

plt.show()
```



In [59]:

```
print(f'Captch extraído da imagem {DIR_TARGET} foi: {captch}')
```

```
Captch extraído da imagem ./dataset/samples/8n5p3.png foi: 8n5p3
```

In [ ]: