Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco

Classificação de caracteres (EMNIST) usando CNN

Aprendizado de Máquina

Marcelo Gervazoni Carbonera

12 de Julho de 2019

Base de Dados EMNIST

Extended MNIST (Modified National Institute of Standards and Technology)

- Imagens de caracteres escritos à mão.
- 375974 imagens de 45x45 pixels (1 ou 0).
- Distribuídas em 82 classes.

Base de Dados EMNIST

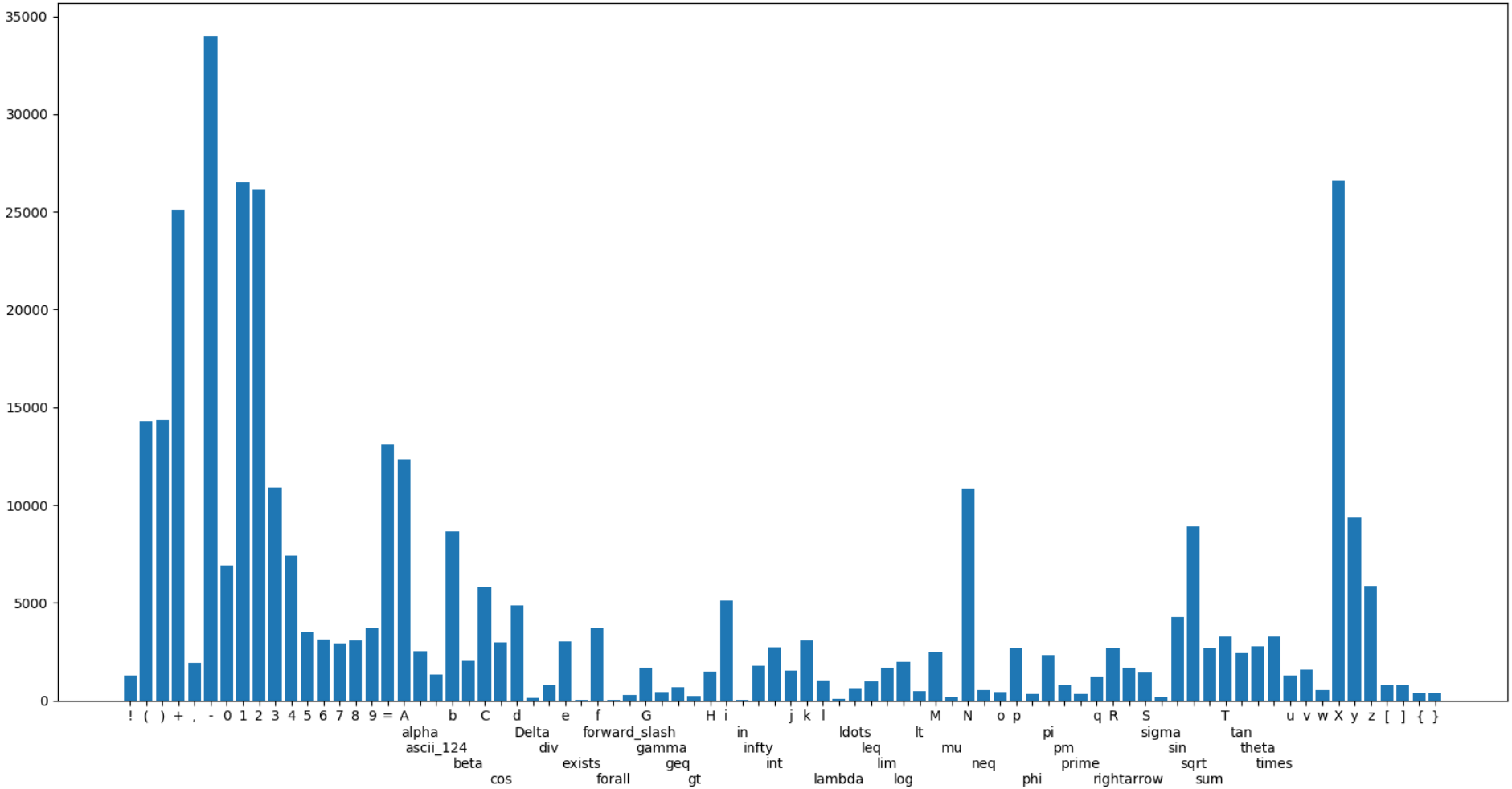
Extended MNIST (Modified National Institute of Standards and Technology)

- Imagens de caracteres escritos à mão.
- 375974 imagens de 45x45 pixels (1 ou 0).
- Distribuídas em 82 classes.

Dados

- Imagens estão dispostas em pastas distintas para cada classe.
- Classes: '!', '(', ')', '+', ',', '-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '=', 'A', 'alpha', 'ascii_124', 'b', 'beta', 'C', 'cos', 'd', 'Delta', 'div', 'e', 'exists', 'f', 'forall', 'forward_slash', 'G', 'gamma', 'geq', 'gt', 'H', 'i', 'in', 'infty', 'int', 'j', 'k', 'l', 'lambda', 'ldots', 'leq', 'lim', 'log', 'lt', 'M', 'mu', 'N', 'neq', 'o', 'p', 'phi', 'pi', 'pm', 'prime', 'q', 'R', 'rightarrow', 'S', 'sigma', 'sin', 'sqrt', 'sum', 'T', 'tan', 'theta', 'times', 'u', 'v', 'w', 'X', 'y', 'z', '[', ']', '{', '}'

Classes desbalanceadas:



Código

Codigos:

```
1  import numpy as np
2  import os
3  import random as rd
4  from skimage import io, color
5
6  import keras
7  from keras.models import Sequential
8  from keras.layers import Dense, Dropout, Flatten
9  from keras.layers import Conv2D, MaxPooling2D
10 from keras.layers.convolutional import Convolution2D
11
```

Códigos:

```
12  ## Porcentagem de treinamento:
13  tx_treinamento = 0.2
14
15  ## Imprime nome das pastas
16  base = os.getcwd()
17  i = 0
18  VarDir = [];
19  for root, dirs, files in os.walk(os.getcwd(), topdown=False):
20      for name in dirs:
21          i=i+1
22          VarDir.append(os.path.join(root, name))
23
24  # Extrair nomes das classes
25  k = VarDir.pop(82)
26  classes = []
27  for i in range(0, len(VarDir)):
28      x = VarDir[i].split(os.sep)
29      classes.append(x[-1])
30
```

Códigos:

```
31  # Percorremos as pastas. Para cada uma, obter lista com nomes das imagens
32  SomaImagens = 0
33  x_treina = []
34  y_treina = []
35  x_teste = []
36  y_teste = []
37  # i caminha em diretórios (classes distintas)
38  numero_classes = len(VarDir)
39  #numero_classes = 2
40  for i in range(0, numero_classes): #len(VarDir)
41      # Percorrendo arquivos na pasta:
42      ListaArquivo = os.listdir(VarDir[i])
43
44      # Soma de imagens carregadas
45      SomaImagens = SomaImagens + len(ListaArquivo)
46      print('Imagens carregadas:', SomaImagens)
47
48      # Separar aleatoriamente imagens para treinamento e teste
49      rd.shuffle(ListaArquivo)
50
```

Códigos:

```
51      # Numero de linhas e colunas na primeira imagem:
52       Im = io.imread(VarDir[i] + '\\' + ListaArquivo[0])
53       lin, col = Im.shape
54
55       # j caminha sobre figuras nas pastas especificadas por i
56       total_imagens_na_pasta = len(ListaArquivo)
57       for j in range(0,total_imagens_na_pasta):
58           # Abre figura
59           Im = io.imread(VarDir[i] + '\\' + ListaArquivo[j])
60           Im = color.rgb2gray(Im) > 127
61      # Porcentagem treinamento ou teste
62           if(j/total_imagens_na_pasta <= tx_treinamento): # Treinamento
63               x_treina.append(Im)
64               y_treina.append(i) # i é a classe
65           else: # Teste
66               x_teste.append(Im)
67               y_teste.append(i) # i é a classe
68
```

Códigos:

```
69  x_treina = np.array(x_treina)
70  y_treina = np.array(y_treina)
71  x_teste = np.array(x_teste)
72  y_teste = np.array(y_teste)
73
74  ################
75  # Etapa do CNN #
76  ################
77  x_treina = x_treina.reshape(len(x_treina),45,45,1)
78  x_teste = x_teste.reshape(len(x_teste),45,45,1)
79
80  batch_size = 128
81  #num_classes = 10 #Variavel numero_classes
82  epochs = 20
83
84  print('x_treina shape:', x_treina.shape)
85  print(x_treina.shape[0], 'train samples')
86  print(x_teste.shape[0], 'test samples')
87
```

Códigos:

```
88  # convert class vectors to binary class matrices
89  y_treina = keras.utils.to_categorical(y_treina, numero_classes)
90  y_teste = keras.utils.to_categorical(y_teste, numero_classes)
91
92  model = Sequential()
93  model.add(Convolution2D(32, kernel_size=(3, 3),
94                    activation='relu',
95                    input_shape=(45,45,1)))
96  model.add(Convolution2D(64, (3, 3), activation='relu'))
97  model.add(MaxPooling2D(pool_size=(2, 2)))
98  model.add(Dropout(0.25))
99  model.add(Flatten())
100 model.add(Dense(128, activation='relu'))
101 model.add(Dropout(0.5))
102 model.add(Dense(numero_classes, activation='softmax'))
103
104 model.compile(loss=keras.losses.categorical_crossentropy,
105               optimizer=keras.optimizers.Adadelta(),
106               metrics=['accuracy'])
107
```

Códigos:

```
108 model.fit(x_treina, y_treina,
109           batch_size=batch_size,
110           epochs=epochs,
111           verbose=1,
112           validation_data=(x_teste, y_teste))
113 score = model.evaluate(x_teste, y_teste, verbose=0)
114 print('Test loss:', score[0])
115 print('Test accuracy:', score[1])
116
117
```

Salvar modelo:

```
118 with open("model.json","w") as json_file:
119    json_file.write(model_json)
120 model.save_weights("model.h5")
121
```

Carregar modelo salvo:

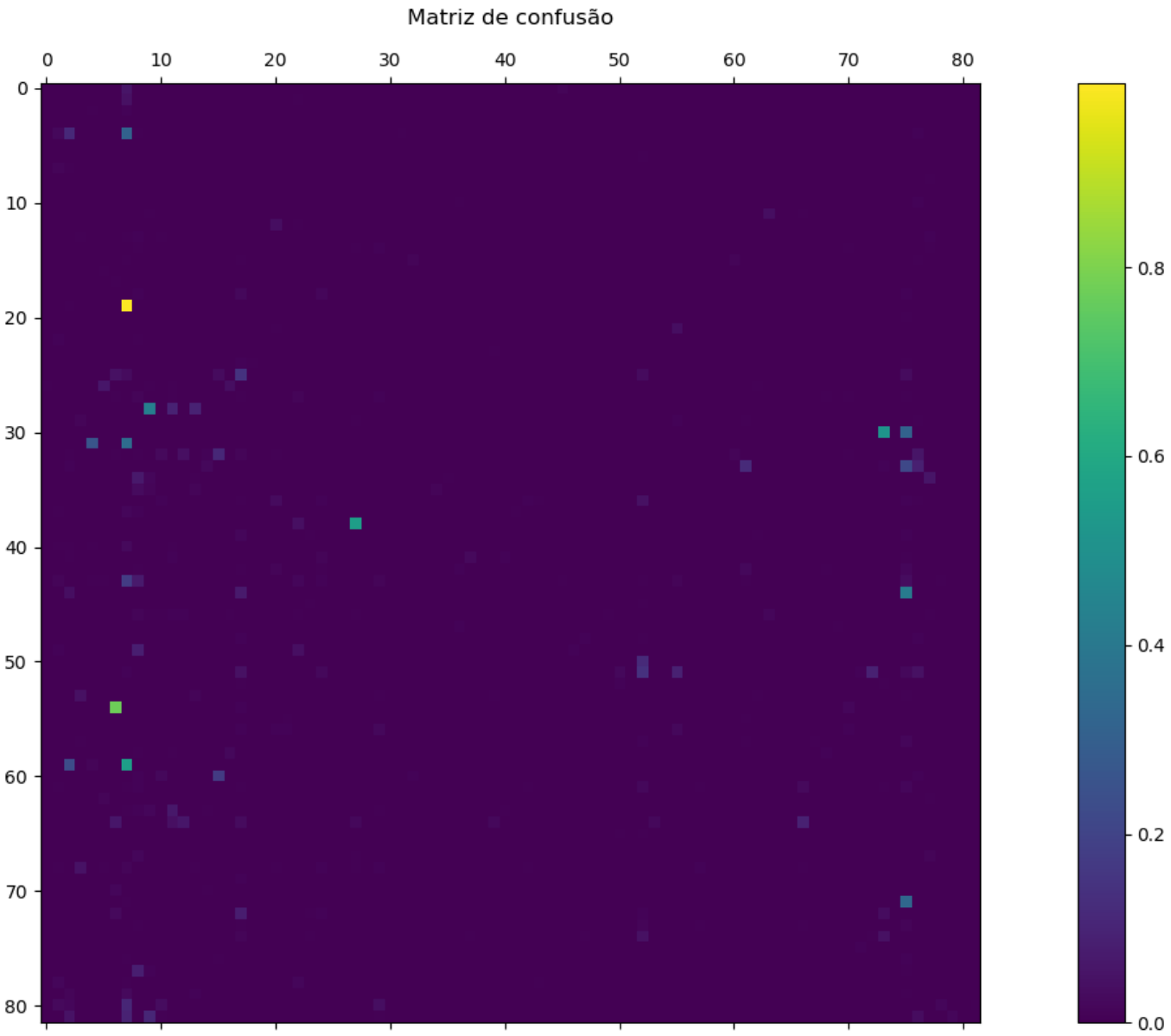
```
122 from keras.models import model_from_json
123
124 json_file = open('model.json', 'r')
125 loaded_model_json = json_file.read()
126 json_file.close()
127
128 loaded_model = model_from_json(loaded_model_json)
129 loaded_model.load_weights("model.h5")
130
131 print('Modelo importado.');
132 # Compilar modelo:
133 loaded_model.compile(loss=keras.losses.categorical_crossentropy, optimizer
        =keras.optimizers.Adadelta(),metrics=['accuracy'])
134
```

Resultado:

Test loss: 0.2169904357216396

Test accuracy: 0.9514195130511174

Matriz de confusão:



DoGrupo	Predição	Erro	NumdoGrupo	NumPredição	Razão
!	1	0.054615384615384614	1300	26520	0.049019607843137254
,	)	0.11647429171038824	1906	14355	0.13277603622431208
,	1	0.3111227701993704	1906	26520	0.07187028657616892
ascii_124	1	0.9955190440627334	1339	26520	0.050490196078431374
Delta	A	0.145985401459854	137	12367	0.011077868521064122
div	-	0.06060606060606061	792	33997	0.023296173191752215
exists	3	0.42857142857142855	21	10909	0.0019250160418003484
exists	5	0.09523809523809523	21	3545	0.005923836389280677
exists	7	0.09523809523809523	21	2909	0.007218975592987281
forall	v	0.5111111111111111	45	1558	0.028883183568677792
forall	X	0.3111111111111111	45	26594	0.0016921110024817629
forward_slash	,	0.26181818181818184	275	1906	0.14428121720881426
forward_slash	1	0.3527272727272727	275	26520	0.010369532428355957
G	9	0.11229314420803782	1692	3737	0.45276960128445276
G	y	0.057919621749408984	1692	9340	0.18115631691648823
gamma	R	0.1198044009779951	409	2671	0.15312616997379258
gamma	X	0.21515892420537897	409	26594	0.015379408889223133
gamma	y	0.08801955990220049	409	9340	0.043790149892933616
geq	2	0.06926406926406926	693	26141	0.026510079951034774
geq	z	0.05772005772005772	693	5870	0.11805792163543441
in	e	0.5531914893617021	47	3003	0.015651015651015652
l	1	0.176007866273353	1017	26520	0.03834841628959276

DoGrupo	Predição	Erro	NumdoGrupo	NumPredição	Razão
l	2	0.0727630285152409	1017	26141	0.03890440304502506
lambda	A	0.07339449541284404	109	12367	0.008813778604350286
lambda	X	0.41284403669724773	109	26594	0.004098668872678048
lt	2	0.08176100628930817	477	26141	0.018247197888374585
M	N	0.11954765751211632	2476	10862	0.22795065365494385
mu	N	0.14124293785310735	177	10862	0.016295341557724177
mu	p	0.0903954802259887	177	2680	0.06604477611940299
mu	u	0.0903954802259887	177	1269	0.13947990543735225
o	0	0.77728285077951	449	6914	0.06494070002892681
prime	)	0.23100303951367782	329	14355	0.02291884360849878
prime	1	0.5623100303951368	329	26520	0.012405731523378581
q	9	0.17479674796747968	1230	3737	0.32914102221032915
S	5	0.06510969568294409	1413	3545	0.398589562764457
sigma	0	0.05970149253731343	201	6914	0.0290714492334394
sigma	6	0.05970149253731343	201	3118	0.06446440025657472
sigma	sqrt	0.08955223880597014	201	8908	0.022563987427031883
times	X	0.3343586588741926	3251	26594	0.1222456193126269 1
u	A	0.07880220646178093	1269	12367	0.10261178943963775
z	2	0.08313458262350937	5870	26141	0.22455147086951532
{	1	0.10638297872340426	376	26520	0.014177978883861237
}	1	0.09549071618037135	377	26520	0.014215686274509804
}	3	0.10610079575596817	377	10909	0.03455862132184435

Agradecimentos

Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco

Classificação de caracteres (EMNIST) usando CNN
Aprendizado de Máquina

Marcelo Gervazoni Carbonera

12 de Julho de 2019