Reconhecimento de números usando aprendizado profundo

Introdução

Este experimento foi desenvolvido por Vilson Rodrigues Câmara Neto e coordenado pelo professor Orivaldo Vieira. Ele busca reconhecer números escritos à mão por estudantes do Bacharelado em Ciências e Tecnologia (ECT) da UFRN.

Foi utilizada uma rede neural convolucional (CNN, em inglês) para segmentar e extrair os caracteres e transformá-los em vetores de características.

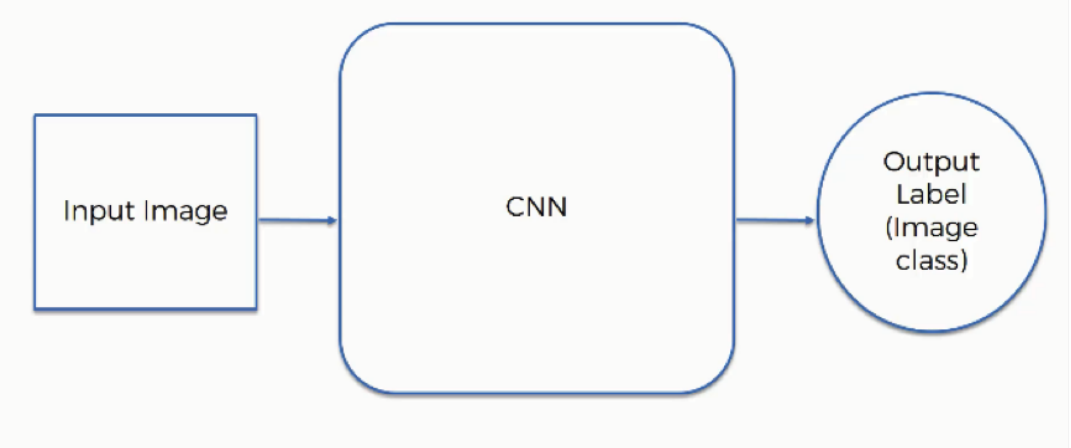

Figura 1: Funcionamento básico da CNN

Um computador compreende imagens através de 3 canais de cores: vermelho, verde e azul (RBG, em inglês), com os pixels variando entre 0 à 255.

Depois foi utilizado a rede neural supervisionada, Multi Layer Perceptron (MLP), da biblioteca Keras para realizar o treino e teste dos dados a fim de encontrar a melhor acurácia.

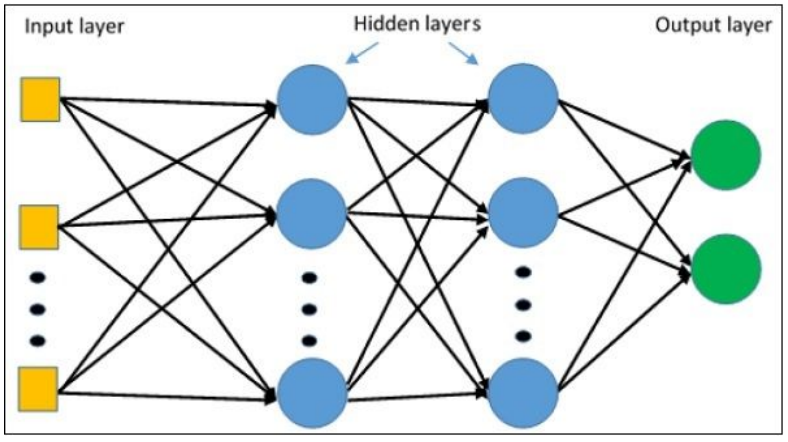

Figura 2: Imagem ilustrativa do funcionamento da MLP

Metodologia

Primeiro foi selecionada a base de dados de caracteres escritos à mão com 1170 exemplos. Depois extraídos do .zip e armazenados em 2 listas, uma com as imagens e outra com o alvo delas, que foi chamado de labels. Depois foi aplicada uma transformação para converter em tons de cinza a fim de melhorar a identificação de um caractere usando a biblioteca OpenCV.

Depois de ter a base de dados, foi realizado um split para separar em treino e teste, a divisão ficou 25% para teste e 75% para treino.

Para reduzir a dimensionalidade foi divido todos os valores do vetor de características por 255 (máximo como já citado).

Partimos para a etapa de modelagem da rede. A biblioteca Keras foi importada para usar a MLP. A primeira camada com 50 neurônios, a segunda com 100 e a saída com 10.

800 épocas foi o número escolhido em que a MLP iria treinar.

Códigos

```
import os
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import matplotlib.pyplot as plt
```

Figura 3: Bibliotecas utilizadas para extração de caracteres e exibição de gráficos

```
[ ]  images = []
     labels = []
     def traverse_dir(path):
         for file_or_dir in os.listdir(path):
             abs_path = os.path.abspath(os.path.join(path, file_or_dir))
             print(abs_path)
             if os.path.isdir(abs_path):  # dir
                 traverse_dir(abs_path)
             else:                        # file
                 if file_or_dir.endswith('.jpg'):
                     image = read_image(abs_path)
                     images.append(image)
                     labels.append(path[len(path)-1])

         return images, labels
```

Figura 4: Função para separar em as imagens das labels a partir da leitura da pasta

```
def read_image(file_path):
    image = cv2.imread(file_path)
    # converte para tons de cinza
    gray_scale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # inverte a cor
    image = cv2.bitwise_not(gray_scale)
    return image


def extract_data(path):
    images, labels = traverse_dir(path)
    images = np.array(images)

    return images, labels
```

Figura 5: Função para converter em tons de cinza e depois inverter a cor a fim de facilitar a identificação dos caracteres

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

Figura 6: Importando a biblioteca Keras

```
model = Sequential()
# Adding the input layer and the first hidden layer
model.add(Dense( activation = 'relu', input_dim = (28*32), units = 50, kernel_initializer = 'uniform'))

# Adding the second hidden layer
model.add(Dense( activation = 'relu', units = 100, kernel_initializer = 'uniform' ))

# Adding the output layer
model.add(Dense( activation = 'sigmoid', units = 10, kernel_initializer = 'uniform'))
```

Figura 7: Modelagem da MLP

```
history = model.fit(X_train, y_train,validation_split=0.1, batch_size = 10, epochs = 800)
```

Figura 8: Treinamento da MLP

Experimentos

Desde o começo a rede mostrou um alto nível de precisão ao classificar, depois de adicionar e remover camadas, diminuir e aumentar neurônios, a combinação que melhor gerou acertos foi a de 50-100-10. A acurácia desse modelo foi de 91,4%. Um ótimo número, dado que a base de dados tem alguns dígitos confusos, e também, não foi uma grande base de dados. Situações de classificação de imagens demandam milhares de exemplos, então podemos dizer que foi um sucesso.