MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAIANO –
CAMPUS GUANAMBI

CALEBE GOMES PEREIRA
DIANE MARQUES PARDIM
FRANCE NÁDIA SANTOS MORAIS
JAQUELINE PEREIRA NOGUEIRA
MARCELO HENRIQUE MARTINS NEVES

RELATÓRIO DE IMPLEMENTAÇÃO DA INTERFACE DO SOFTWARE DE RECONHECIMENTO DE IMAGENS COM REDES NEURAIS CONVOLUCIONAIS

GUANAMBI
2019

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAIANO – *CAMPUS* GUANAMBI

CALEBE GOMES PEREIRA
DIANE MARQUES PARDIM
FRANCE NÁDIA SANTOS MORAIS
JAQUELINE PEREIRA NOGUEIRA
MARCELO HENRIQUE MARTINS NEVES

RELATÓRIO DE IMPLEMENTAÇÃO DA INTERFACE DO SOFTWARE DE RECONHECIMENTO DE IMAGENS COM REDES NEURAIS CONVOLUCIONAIS

Relatório apresentado a disciplina de Tópicos Avançados em Análise de Projetos de Informação do curso de Análise e Desenvolvimento de Sistemas, ofertado pelo IF Baiano - *campus* Guanambi.

Professor: Antônio Neto

GUANAMBI
2019

SUMÁRIO

RESUMO

O presente relatório retrata a criação de uma interface gráfica para usuário (GUI – *Graphic User Interface*) desenvolvida com a utilização da linguagem de programação *Python* e o *Framework Tkinter*, essa interface criada tem como objetivo a implementação de um código de reconhecimento de imagens de gatos e cachorros utilizando redes neurais convulsionais (CNN do inglês *Convolutional Neural network*). Através da presença de uma pasta de imagens para o treino e outra pasta de imagens para testes, a interface desenvolvida exibe o índice de acurácia da CNN. Este relatório está organizado da seguinte forma: o capítulo um apresenta a introdução ao projeto desenvolvido, o capítulo dois a descrição de como foi desenvolvida a implementação da interface gráfica no código, por fim, o capítulo três apresenta as considerações finais acerca deste trabalho.

INTRODUÇÃO

Atualmente o aprendizado de máquinas, mais especificamente o aprendizado com redes neurais convolucionais é um tema bastante relevante, devido ao seu grande sucesso principalmente na identificação de pessoas, animais ou objetos (VARGAS, PAES E VASCONCELOS, 2016).

Esse relatório apresenta o desenvolvimento de uma interface gráfica para um código CNN desenvolvido e disponibilizado pelo professor Antônio Queiroz da Silva Neto para os alunos da turma T4DS, disciplina Tópicos Avançados em Análise e Projetos de Sistemas de Informação do curso de Tecnologia em Análise e desenvolvimento de Sistemas do IF Baiano - Campus Guanambi no semestre 2019.1.

O objetivo do código desenvolvido foi executar o treinamento da CNN através de duas pastas de imagens (uma para treino e outra para teste) também disponibilizadas pelo professor, pastas essas que são identificadas através da interface gráfica, após a execução e treinamento da CNN a interface exibe na tela a acurácia obtida.

JUSTIFICATIVA

Atualmente nas organizações, a inteligência artificial vem tornando os processos mais eficientes. Um estudo da consultoria BCC Research mostra que os investimentos no setor devem somar US$ 15,2 bilhões em 2019, o que representa um crescimento médio anual de quase de 20%.

A importância do estudo dessa área de atuação é de suma importância para a vida acadêmica e profissional de qualquer acadêmico que planeje seguir na área de desenvolvimento. No mercado de trabalho em tecnologia da informação o tema tem imensa relevância, pois é responsável por várias das comodidades que representam o cotidiano das organizações.

OBJETIVOS

Objetivo geral

Este projeto teve como objetivo a implementação de uma interface gráfica para um código de reconhecimento de imagens de gatos e cachorros com redes neurais convolucionais.

Objetivos específicos

• Analise das discrepâncias da acurácia em relação a vários testes de reconhecimento de imagem de gatos e cachorros em relação a quantidade de imagens para treino e teste;
• Análise do tempo de obtenção das imagens de treino e teste pela interface criada em relação ao teste feito apenas através do código.

METODOLOGIA

Tecnologias utilizadas

A biblioteca utilizada para a implementação da rede neural foi a *Tensorflow,* que se trata de um sistema para criação e treinamento de redes neurais detectando e decifrando padrões e correlações, de maneira parecida à forma como humanos aprendem e raciocinam. Para permitir a criação da interface foi utilizado o *Keras* uma API de alto nível escrita em Python que roda como *frontend* em cima de *Tensorflow*.

A linguagem de programação utilizada foi o Python com o compilador Spyder 3.2.0. O software Spyder 3.2.0foi usado com a utilidade de edição dos códigos de programação.

Para a criação da interface gráfica foi utilizado o framework Tkinter que é uma biblioteca da linguagem Python que acompanha a instalação padrão e permite desenvolver interfaces gráficas. Isso significa que qualquer computador que tenha o interpretador Python instalado é capaz de criar interfaces gráficas usando o Tkinter, com exceção de algumas distribuições Linux, exigindo que seja feita o download do módulo separadamente.

Desenvolvimento do código

Abaixo o código disponibilizado pelo professor:

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from keras.preprocessing import image

classificador = Sequential()
classificador.add(Conv2D(32, (3,3), input_shape = (64, 64, 3),
activation = 'relu'))
classificador.add(BatchNormalization())
classificador.add(MaxPooling2D(pool_size = (2,2)))

classificador.add(Conv2D(32, (3,3), input_shape = (64, 64, 3),
activation = 'relu'))
classificador.add(BatchNormalization())
classificador.add(MaxPooling2D(pool_size = (2,2)))
```

```
classificador.add(Flatten())

classificador.add(Dense(units = 128, activation = 'relu'))
classificador.add(Dropout(0.2))
classificador.add(Dense(units = 128, activation = 'relu'))
classificador.add(Dropout(0.2))
classificador.add(Dense(units = 1, activation = 'sigmoid'))

classificador.compile(optimizer = 'adam', loss =
'binary_crossentropy',
                      metrics = ['accuracy'])

gerador_treinamento = ImageDataGenerator(rescale = 1./255,
                                         rotation_range = 7,
                                         horizontal_flip = True,
                                         shear_range = 0.2,
                                         height_shift_range = 0.07,
                                         zoom_range = 0.2)
gerador_teste = ImageDataGenerator(rescale = 1./255)

base_treinamento =
gerador_treinamento.flow_from_directory('dataset/training_set',
                                                              target_size
= (64, 64),
                                                              batch_size
= 32,
                                                              class_mode
= 'binary')
base_teste = gerador_teste.flow_from_directory('dataset/test_set',
                                               target_size = (64, 64),
                                               batch_size = 32,
                                               class_mode = 'binary')
classificador.fit_generator(base_treinamento, steps_per_epoch = 4000 /
32,
                            epochs = 10, validation_data = base_teste,
                            validation_steps = 1000 / 32)
imagem_teste =
image.load_img('dataset/test_set/cachorro/dog.3500.jpg',
                               target_size = (64,64))
imagem_teste = image.img_to_array(imagem_teste)
imagem_teste /= 255
imagem_teste = np.expand_dims(imagem_teste, axis = 0)
previsao = classificador.predict(imagem_teste)
previsao = (previsao > 0.5)

base_treinamento.class_indices
```

A parte em vermelha foi retirada no desenvolvimento do projeto da interface gráfica, pois o objetivo é apenas obter a acurácia após treino e teste.

Através desse código, foi criada uma função em um arquivo chamado GatoCachorroCode.py, essa função intitulada como **treaining_and_test** recebe o caminho da pasta de imagem de treino como parâmetro **training_set** e da pasta de teste como parâmetro **test_set** que são passadas como parâmetros nas funções existentes. `gerador_treinamento.flow_from_directory` e `gerador_teste.flow_from_directory`.

O código completo desse arquivo está apresentado abaixo.

```
"""
Created on Mon Jun 10 08:43:58 2019

"""

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from keras.preprocessing import image

acuracia = 0.999

def treaining_and_test (training_set, test_set):
    #try:
        if (training_set != '/' and test_set != '/'):
            classificador = Sequential()
            classificador.add(Conv2D(32, (3,3), input_shape = (64, 64,
3), activation = 'relu'))
            classificador.add(BatchNormalization())
            classificador.add(MaxPooling2D(pool_size = (2,2)))

            classificador.add(Conv2D(32, (3,3), input_shape = (64, 64,
3), activation = 'relu'))
            classificador.add(BatchNormalization())
            classificador.add(MaxPooling2D(pool_size = (2,2)))

            classificador.add(Flatten())

            classificador.add(Dense(units = 128, activation = 'relu'))
            classificador.add(Dropout(0.2))
            classificador.add(Dense(units = 128, activation = 'relu'))
            classificador.add(Dropout(0.2))
            classificador.add(Dense(units = 1, activation =
'sigmoid'))

            classificador.compile(optimizer = 'adam', loss =
'binary_crossentropy',
                                          metrics = ['accuracy'])

            gerador_treinamento = ImageDataGenerator(rescale = 1./255,
                                                     rotation_range =
7,
                                                     horizontal_flip =
True,
                                                     shear_range =
0.2,

height_shift_range = 0.07,
                                                     zoom_range = 0.2)
            gerador_teste = ImageDataGenerator(rescale = 1./255)

            base_treinamento =
gerador_treinamento.flow_from_directory(training_set,
```

```
target_size = (64, 64),

batch_size = 32,

class_mode = 'binary')
            base_teste = gerador_teste.flow_from_directory(test_set,
                                                           target_size
= (64, 64),
                                                           batch_size
= 32,
                                                           class_mode
= 'binary')

            classificador.fit_generator(base_treinamento,
steps_per_epoch = 4000 / 1000,
                                                    epochs = 1,
validation_data = base_teste,
                                                    validation_steps =
1000 / 32)


            #self.msg.config(text=classificador.metrics)

            return acuracia;
        else:
            return -1
    #except: return -2
```

Nesse código há ainda o tratamento de erro da especificação das pastas de imagens return -1 e o tratamento de erro do carregamento de arquivos return -2. Caso tudo ocorra bem a função retornará o valor da acurácia obtida.

Outro arquivo criado chamado de **gatoCachorro.py** possui os códigos referentes a interface criada. Esse código cria uma tela com uma etiqueta de nome da aplicação 1, uma etiqueta de informações da aplicação 2, uma etiqueta onde mostrar os avisos retornados da função **treaining_and_test 3**, um botão para abrir a pasta de treino 4, um botão para abrir a pasta de teste 5 e um botão para executar a função **treaining_and_test**6**,** passando os valores obtidos na função de obter pastas citadas anteriormente no mesmo código. Abaixo a imagem gráfica da aplicação e seu respectivo código.


```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 10 08:43:58 2019

"""

import os
from tkinter import filedialog
from tkinter import *
import gatoCachorroCode

training_set = '/'
test_set = '/'

def load_directory():
    dirName = filedialog.askdirectory()
    return dirName

def load_file():
    fileName = filedialog.askopenfilename()
    return fileName

class Janela:



    def __init__(self,toplevel):
        self.frame=Frame(toplevel)
        self.frame.pack()
        self.frame2=Frame(toplevel)
        self.frame2.pack()
        self.titulo=Label(self.frame,text='Interface IA',
                          font=('Verdana','22','bold'))
        self.titulo.pack(pady = 20)
        self.msg=Label(self.frame,
                       text = 'Este programa treina uma CNN para
reconhecimento de gatos e cachorros',
                       font=('Verdana','16'))
```

```
        self.infor=Label(self.frame,
                         text = 'O resultado da acurácia aparecerá
aqui.',
                         font=('Verdana','14','bold'))
        self.msg.focus_force()
        self.msg.pack(pady = 20)
        self.infor.pack(pady = 150)
        # Definindo o botão 1
        self.b01=Button(self.frame2,text='Abrir pasta de treino')
        self.b01['padx'],self.b01['pady'] = 10, 5
        self.b01['bg']='deepskyblue'
        self.b01.bind("<Any-Button>",self.button01)
        self.b01.bind("<FocusOut>",self.fout01)
        self.b01['relief']=RIDGE
        self.b01.pack(padx = 20, ipadx = 30, ipady = 30, side=LEFT)
        # Definindo o botão 2
        self.b02=Button(self.frame2,text='Abrir pasta de teste')
        self.b02['padx'],self.b02['pady'] = 10, 5
        self.b02['bg']='deepskyblue'
        self.b02.bind("<Any-Button>",self.button02)
        self.b02.bind("<FocusIn>",self.fin02)
        self.b02.bind("<FocusOut>",self.fout02)
        self.b02['relief']=RIDGE
        self.b02.pack(padx = 20, ipadx = 30, ipady = 30, side=LEFT)
        # Definindo o botão 3
        self.b03=Button(self.frame2,text='Treinar')
        self.b03['padx'],self.b03['pady'] = 10, 5
        self.b03['bg']='deepskyblue'
        self.b03.bind("<Any-Button>",self.button03)
        self.b03.bind("<FocusIn>",self.fin03)
        self.b03.bind("<FocusOut>",self.fout03)
        self.b03['relief']=RIDGE
        self.b03.pack(padx = 20, ipadx = 30, ipady = 30, side=LEFT)
        self.b03.config(bg = '#fc8888')



    def button01(self,event):
        global training_set
        training_set = load_directory()
    def button02(self,event):
        global test_set
        test_set = load_directory()
    def fin01(self,event): self.b01['relief']=FLAT
    def fout01(self,event): self.b01['relief']=RIDGE
    def fin02(self,event): self.b02['relief']=FLAT
    def fout02(self,event): self.b02['relief']=RIDGEE
    def fin03(self,event): self.b03['relief']=FLAT
    def fout03(self,event): self.b03['relief']=RIDGE



    def button03(self,event):

        global training_set
        global test_set
        start = gatoCachorroCode.treaining_and_test(training_set,
test_set)
        if (start == -1):
```

```
            self.infor.config(text='Selecione ambas as pastas de
treino e teste.', foreground = 'red')
        elif(start == -2):
            self.infor.config(text='Houve um erro ao carregar os
arquivos.', foreground = 'red')
        else:
            self.infor.config(text= 'Acurácia  = ' + str(start),
foreground = 'red')




raiz=Tk()
raiz.title("IA classificador de imagens")


window_height = 600
window_width = 900

screen_width = raiz.winfo_screenwidth()
screen_height = raiz.winfo_screenheight()

x_cordinate = int((screen_width/2) - (window_width/2))
y_cordinate = int((screen_height/2) - (window_height/2))

raiz.geometry("{}x{}+{}+{}".format(window_width, window_height,
x_cordinate, y_cordinate))


Janela(raiz)
raiz.mainloop()
```

Os códigos da interface e da função referente ao que foi descrito poderá ser encontrada em:https://github.com/calebegmsp/graphic-interface-dl.

Na execução do algoritmo trabalhado, há a compilação e execução dos arquivos de dados, escritos por um script em Python. São construídas extensões para que possibilite a injeção de dados e outras informações necessárias. A partir daí, ocorre um teste de cada parte da unidade gerando um compilado de informações a respeito deste processo de treino, com a finalidade da obtenção de uma correta identificação de imagens, oportunizando, inclusive, a verificação da acurácia/confiabilidade desta rede neural convolucional.

Descrevendo a interface gráfica

Nesta seção será apresentada as interfaces gráficas do usuário bem como, as instruções de como utilizá-las. A tela inicial do software é apresentada na figura seguinte.

O primeiro passo ao executar a aplicação é selecionar as pastas de treino e testes clicando nos botões “Abrir pasta de treino”. “Abrir pasta de teste”Após serem selecionados, o botão “Treinar”deverá requisitado.Clicando nele o nível de acurácia será gerado e mostrado na interface da aplicação.

Treinamento da RNA

Algoritmos de rede neurais artificiais possuem como entrada e saída valores numéricos, portanto é necessário converter os rótulos de cada classe de objetos de modo que a RNA consiga identificar na saída da rede. Procedendo de maneira que se reduza o número de parâmetros que deverão ser ajustados pela rede, e então melhorar o processo de treinamento.

CONSIDERAÇÕES FINAIS

Todos os objetivos específicos foram atendidos neste processo, permitindo, assim algumas constatações acerca da aplicação prática do conteúdo aprendido em sala de aula, como o fato de ser possível realizar projetos reais e implementar um *software* útil para uma determinada área/assunto sem a obrigatoriedade de se ter um conhecimento aprofundado sobre o mesmo.

O projeto apresentou uma introdução a inteligência artificial, classificação de imagem, aprendizagem de máquina e redes neurais artificiais convolucionais. Dentro de redes neurais artificiais foi mostrado o modelo com uma camada que trabalha da esquerda para a direita e também pode ter várias camadas ocultas, sendo chamado de multicamadas.

Foi possível constatar que a interface gráfica ajuda na rapidez de processos como abrir as pastas de imagem treino e teste além possuir um melhor visual para a análise da acurácia apresentada. A interface não possibilitou nem auxiliou em uma maior facilidade em observar possíveis discrepâncias da acurácia quando se faz grande quantidade de testes. Uma possível solução para isso seria armazenar e disponibilizar na interface o valor da acurácia realizadas em testes anteriores.

O processo de treino foi demorado, já que nem toda imagem que é fornecida pode ser um gato ou um cachorro e pode ter muita coisa na imagem que pode confundir no momento que os dados serão treinados e isso resulta em maior risco de erros. O treinamento realizado mostrou um bom resultado com uma quantidade pequena de dados para treinamento.

Assim sendo, com este projeto foi possível colocar em prática parte do conteúdo teórico aprendido em sala de aula.

REFERÊNCIAS

VARGAS, Ana Caroline Gomes; PAES, A. V. C. N.; VASCONCELOS, Cristina Nader. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: **Proceedings of the XXIX Conference on Graphics, Patterns and Images**. 2016. p. 1-4.

BCC RESEARCH. **Inteligência artificial**. [*S. l.*], 2019. Disponível em: https://www.bccresearch.com/. Acesso em: 11 jun. 2019.