Revista Ilha Digital

Endereço eletrônico:
http://ilhadigital.florianopolis.ifsc.edu.br/

INSTITUTO FEDERAL
SANTA CATARINA

DEEP LEARNING COM MÓDULO OPENCV DNN

Jonathan C. C. Bonette [1]

1 INTRODUÇÃO

O campo da visão computacional existe desde o final dos anos 1960. A classificação de imagens e a detecção de objetos são alguns dos problemas mais antigos no campo da visão computacional que os pesquisadores tentaram resolver por muitas décadas. Usando redes neurais e aprendizado profundo, chegamos a um estágio em que os computadores podem começar a realmente entender e reconhecer um objeto com alta precisão, superando até os humanos em muitos casos. E para aprender sobre redes neurais e aprendizado profundo com visão computacional, esse trabalho focará no uso de um módulo do **OpenCV** conhecido por **DNN**.

2 OBJETIVO

Durante o semestre vimos diversas ferramentas usando o módulo **OpenCV**, uma das melhores bibliotecas de visão computacional existentes. Além disso, ele também possui funcionalidades para executar a inferência de aprendizado profundo. A melhor parte é suportar o carregamento de diferentes modelos de diferentes estruturas, com as quais podemos realizar várias funcionalidades de aprendizado, assim como reconhecimento de imagens, padrões, formas e recursos.

O objetivo dessa análise era usar o módulo OpenCV DNN para analisar o funcionamento de padrões de imagens, mais conhecidos como *deep learning*, as últimas atividades que foram desenvolvidas focaram nessa análise de forma diferente, usando a webcam podemos filtrar cores de elementos gráficos e fazer uma análise quantitativa dos elementos filtrados, dessa forma foi interessante usar os conceitos adquiridos durante as últimas atividades com o foco do reconhecimento de padrões e como foco o *deep learning*, ou inteligência artificial.

3 DESENVOLVIMENTO

Foi usado um modelo de rede neural treinado em um conjunto de dados, um pacote chamado **ImageNet**, usando o framework **Caffe**. A tarefa de identificação fica para um módulo chamado de **DensNet121**. Interessante ressaltar que esse banco de dados já treinado tem mais de 1.000 classes do conjunto de dados, então há uma boa chance de os padrões usados serem reconhecidos facilmente.

3.1 Metodologia

i. As classes de conjunto de dados são carregadas e lidos item por item.
ii. O módulo do reconhecimento de imagens é carregado pelo código.
iii. A imagem, vídeo ou imagem em tempo real é carregada e analisada pelo módulo de inteligência usado.
iv. O código é rodado, analisado os padrões e conjuntos de dados e gerado a saída, seja ela, imagem, vídeo ou tempo real.

3.2 Explicação do código

3.2.1. Reconhecendo padrão fixo

O padrão fixo foi estabelecido pra uma análise inicial da ideia. As primeiras linhas do código são os importes dos módulos que são necessários para o carregamento das funcionalidades, é legal comentar que o conjunto de dados treinado já existe para download. Um arquivo de texto é carregado pelo python, usamos linhas de código para quebrar as linhas, as frases e ler esses dados item por item para uma análise mais precisa do item dado para a análise. Além disso é interessante ver que os módulos já vêm com funções pré-

[1] Aluno de Graduação em Engenharia Eletrônica, IFSC/Florianópolis, <jonathan.ccb@aluno.ifsc.edu.br >

programadas para a análise dos itens dados pelo usuário. Ou seja, o código acaba por ficar muito enxuto, isso é bom, pois o as linhas de código ficam menos verbosas e mais compreensíveis.

Logo após existe a leitura dos dados, seja ela uma imagem, vídeo ou streaming e a conversão ou preparação dessa fonte de leitura de dados para um framework já definido pelos módulos. Também foi preciso fazer uma análise dos outputs que contém todas as previsões, essas funções remodelam os dados gerados pelas imagens, o arquivo de texto e assim analisar os padrões e definir Ids para esses rótulos. Como os Ids ainda não são exatos, é preciso passar pelas linhas de código que fazem um tratamento de pontuação, e rotulam por probabilidade os padrões analisados.

O módulo por fim faz a análise dos padrões, categoriza e define um Id para aquela imagem.

Figura 1 – Análise de uma Imagem

Fonte: Aluno (2022).

Para a primeira análise foi usado um reconhecimento fixo de padrões de imagem, mais simples, e pode-se ver que a ferramenta trabalhou de forma esperada, categorizando o gato acima.

3.2.1. Reconhecendo padrões fixo

Já vendo o funcionamento da ferramenta, vamos ao principal, o módulo OpenCV DNN, podemos começar facilmente com a detecção de objetos em deep learning, com uma visão computacional propriamente dita. Assim como a classificação, carregaremos as imagens, os modelos apropriados e propagaremos a entrada através do modelo. As etapas de pré-processamento para visualização adequada na detecção de objetos serão um pouco diferentes.

Da mesma maneira usaremos pacotes já criados e treinados de pacotes de dados para a análise da ferramenta, como os módulos **MS COCO**, **MobileNetSSD** e o módulo de *deep learning* **TensorFlow.**

As primeiras linhas do código são os importes dos módulos que são necessários para o carregamento das funcionalidades, é legal comentar que o conjunto de dados treinado já existe para download. Um arquivo de texto é carregado pelo python, usamos linhas de código para quebrar as linhas, as frases e ler esses dados item por item para uma análise mais precisa do item dado para a análise. O padrão segue o mesmo, mas agora no meio do código vemos uma estrutura de loop fazendo várias detecções de padrões e categorizando elas nos Ids encontrados.

E por fim umas linhas para separar em caixas os itens encontrados e categorizados pelo módulo.

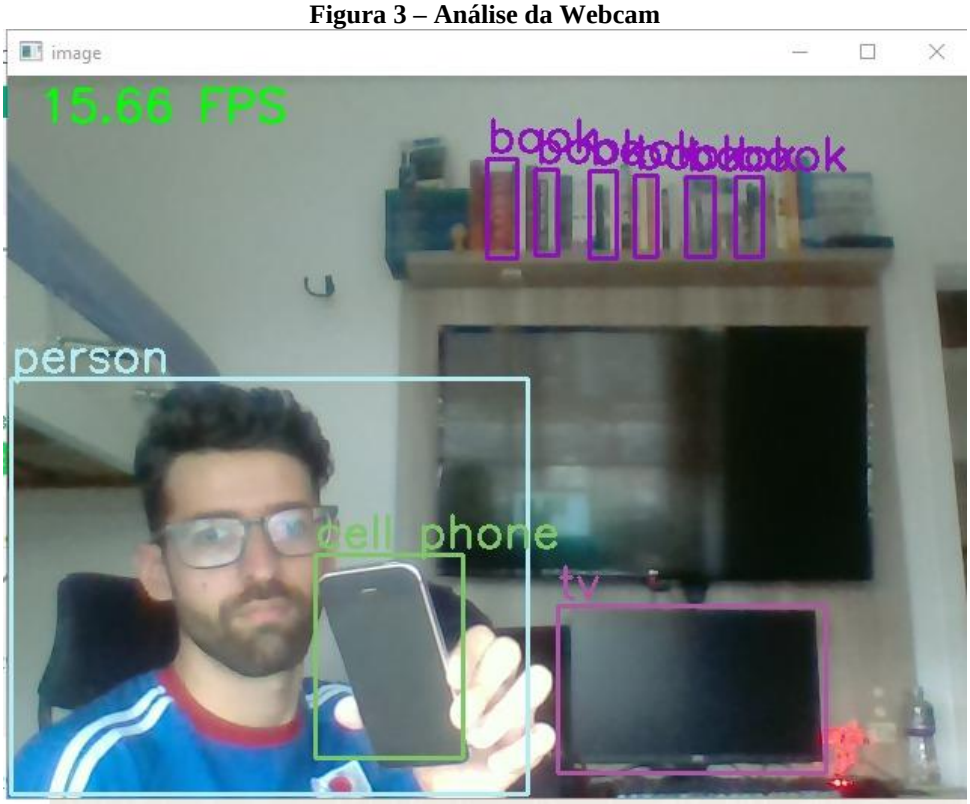
Figura 2 – Análise de uma Imagem

Fonte: Aluno (2022).

3.2.1. Reconhecendo padrões móveis

Da mesma maneira, é possível reconhecer padrões móveis, como um vídeo por exemplo, o código trabalha da mesma maneira apenas o input que vai ser de modo vídeo. Além de usar vídeos prontos, podemos também usar a versão em tempo real, uma análise feita via webcam e vendo o funcionamento das ferramentas descritas acima. Abaixo está o código completo usando a última versão, via webcam.

Figura 3 – Análise da Webcam



Fonte: Aluno (2022).

```
# imports
import cv2
import time
import numpy as np

# carrega o modelo DNN
```

```
with open('../../input/object_detection_classes_coco.txt', 'r') as f:
    class_names = f.read().split('\n')

# categoriza por cores os objetos analisados
COLORS = np.random.uniform(0, 255, size=(len(class_names), 3))

# carrega o modelo DNN
model = cv2.dnn.readNet(model='../../input/frozen_inference_graph.pb',
                        config='../../input/ssd_mobile-
net_v2_coco_2018_03_29.pbtxt.txt', framework='TensorFlow')

# captura a imagem da webcam em tempo real
cap = cv2.VideoCapture(0)

# faz um tratamento do tamano do vídeo
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

# cria um objeto do tipo `VideoWriter()`
out = cv2.VideoWriter('../../outputs/vid_result.mp4', cv2.VideoWri-
ter_fourcc(*'mp4v'), 30,
                      (frame_width, frame_height))

# faz o loop sobre cada detecção de cada frame do vídeo
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        image = frame
        image_height, image_width, _ = image.shape

        # criar o blob, para trabalhar com armazenamentos de dados binários
        blob = cv2.dnn.blobFromImage(image=image, size=(300, 300), mean=(104,
117, 123), swapRB=True)

        # tratamento do começo do tempo do vídeo
        start = time.time()
        model.setInput(blob)
        output = model.forward()

        # tratamento do final do tempo do vídeo
        end = time.time()

        # calcula o FPS atual de cada detecção
        fps = 1 / (end-start)

        # faz o loop sobre cada detecção
        for detection in output[0, 0, :, :]:

            # extrai o grau de confiabilidade da detecção
            confidence = detection[2]

            # colocar quadrados em volta dos padrões se o grau de confiabili-
dade for maior que um certo limite, ou pula a detecção
            if confidence > .4:

                # obtem o ID do objeto
                class_id = detection[1]

                # mapeia a classe
                class_name = class_names[int(class_id)-1]
                color = COLORS[int(class_id)]

                # obtem as coordenadas da caixa que ficará em volta dos obje-
tos observados
```

```
                box_x = detection[3] * image_width
                box_y = detection[4] * image_height

                # pega o limite da caixa e define uma altura e largura
                box_width = detection[5] * image_width
                box_height = detection[6] * image_height

                # desenha a caixa em volta de cada objeto detectado
                cv2.rectangle(image, (int(box_x), int(box_y)),
(int(box_width), int(box_height)), color, thickness=2)

                # coloca o nome da classe na caixa criada
                cv2.putText(image, class_name, (int(box_x), int(box_y - 5)),
cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)

                # faz a contagem do FPS
                cv2.putText(image, f"{fps:.2f} FPS", (20, 30), cv2.FONT_HERS-
HEY_SIMPLEX, 1, (0, 255, 0), 2)

        # gera a imagem, coloca o texto e salva a imagem no diretório output
(q == sair)
        cv2.imshow('image', image)
        out.write(image)
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    else:
        break


# libera a vídeo
cap.release()
cv2.destroyAllWindows()
```

4 CONCLUSÕES

Usando deep learning, podemos fazer muitas análises, desde as mais simples as mais complexas. As ferramentas usadas nas últimas aulas mostraram pra gente o que elas são capazes de fazer, esse trabalho tem uma finalidade de mostrar mais uma dessas formas de utilizar a ferramenta OpenCV e suas diversas capacidades.

Com um pouco mais de tempo poderíamos desenvolver o código para a análise de mascaras e cores e detecção de quantidades e formas em tempo real, é uma outra forma de poder aplicar a ferramenta de acordo com o conteúdo programático da disciplina.

Foi escolhido o trabalho com essa fermenta pela facilidade de uso, e pela extrema usabilidade nos tempos atuais.

Os assuntos aqui abordados já estão sendo discutidos de forma acadêmica há um bom tempo, é legal ver projetos utilizando deep learning e processamento digitais de sinais, como por exemplo a análise clássica de séries temporais, análises do domínio da frequência, modelos de memória a longo prazo (LTSM), detecção de frangos de corte doentes por processamento digital de imagens e aprendizado profundo, análise de imagem digital em patologia da mama - das técnicas de processamento de imagem à inteligência artificial, investigação experimental de deep learning para processamento digital de sinais em comunicações de fibra óptica de curto alcance, entre outros.

REFERÊNCIAS

OpenCV DNN e Deep Learning. Disponível em: https://learnopencv.com/getting-started-with-opencv/. Acesso em 07 dez. 2022.

Artigos Científicos citados:

https://www.sciencedirect.com/science/article/abs/pii/S1537511018307785
https://www.sciencedirect.com/science/article/abs/pii/S1931524417302955
https://ieeexplore.ieee.org/abstract/document/9195215