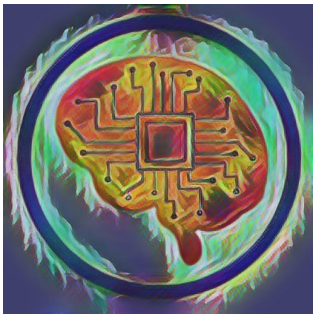


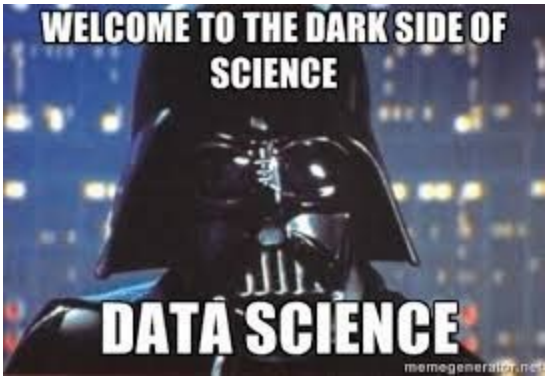
Área de Visão Computacional

Projeto de Trainee

Fala entusiasmad@ da área de visão computacional! Primeiramente, gostaríamos de reiterar os parabéns por terem entrado no Grupo Turing, esperamos que você tenha uma ótima experiência, aprenda muito e faça a diferença com os conhecimentos de IA.

Bom, chega de enrolação e vamos ao que importa! A ideia aqui é fazer você botar a mão na massa (não literalmente a não ser que você seja engenheiro civil) num projetinho de Visão Computacional em que você aprenda os principais conceitos conforme avance no projeto. Junto a este documento, deixaremos diversos links e materiais para você estudar o necessário para a realização do projeto, mas qualquer dúvida não hesite em falar com seu Mentor, algum membro da área de CV, ou membro do Turing em geral. A verdade é que qualquer pessoa vai estar disposta a resolver seu problema.

Enunciado



Você é um dos Jedis sobreviventes após a Ordem 66 e está numa missão secreta com

Obi Wan Kenobi para ajudar a manter os outros Jedis em segurança. Para isso, você criará um sistema de segurança que consegue classificar se uma imagem contém o Mestre Yoda, o Lord Sith Darth Vader, ou se contém algum Stormtrooper.

(se não gosta de star wars não é minha culpa que vc tenha mau gosto, mas se quiser falar mal do 9 me chama rs).

Você vai receber um arquivo contendo pastas com imagens dos personagens. Muitos bothanianos morreram para trazer esses dados, então trate-a com importância e respeito! Vamos te apresentar um pipeline (palavra não tão chique para uma série de tarefinhas que vão te guiar na tarefa) de aprendizado de CV enquanto você aplica no projeto de classificação.

Tarefas de Manipulação

"Muito a fazer com imagens tem você antes de classificar", Mestre Yoda para Luke Skywalker em seu treinamento em Dagobah.

Dificilmente um dataset virá com imagens boas e suficientes para um bom resultado num projeto de classificação, assim, é extremamente importante sabermos tratar imagens corretamente, e aqui você terá que aprender algumas das técnicas.

Antes de começar com as tarefas, sugiro que crie um novo ambiente virtual em Python com as seguintes bibliotecas:
Numpy, Pandas, Matplotlib, OpenCV, Pillow, Tensorflow, PyTorch, Sklearn, scikit-image, glob, os.

Tensorflow e PyTorch são bibliotecas que nos ajudam a criar redes neurais. Sinta-se livre para usar a que quiser, entretanto, recomendamos usar Keras, que é uma biblioteca dentro de Tensorflow que facilita todo o processo de criação das camadas.

Tarefa 1

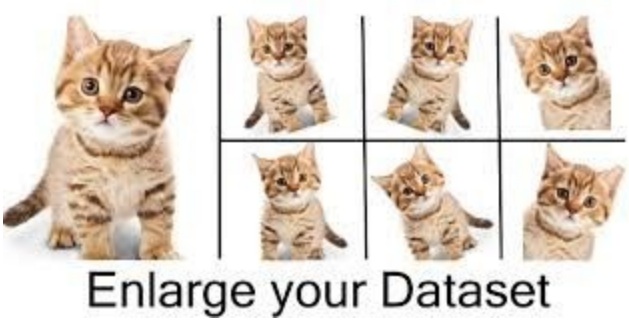
Seu primeiro desafio consiste na simples (porém talvez não tão simples na primeira vez) tarefa de visualizar as imagens do seu dataset. Crie uma função que recebe o endereço de uma imagem e retorna sua visualização.
Por aprendizado, faça isso de três formas diferentes: uma com OpenCV, outra com Pillow e outra com Matplotlib.

Tarefa 2

Sua segunda tarefa é aprender a lidar com os espaços de cores. Descubra como converter uma imagem colorida em uma imagem preto e branco e escreva uma função usando numpy que o faz. Depois veja como converter espaços de cores usando OpenCV. Leia também sobre a diferença entre espaço RGB e BGR. Qual é usado pela biblioteca OpenCV?

Tarefa 3

Como a maioria dos datasets não possuem um número suficiente de imagens para nosso modelo aprender a distinguir com grande precisão as diversas classes, costumamos aplicar a técnica de Augmentation no nosso dataset, que consiste em fazer algumas modificações em imagens existentes para criar imagens que contenham informações novinhas para nosso modelo.



Assim, crie uma função que recebe uma imagem e aplique o processo de augmentation nela. Dentre as técnicas, use:

- Inverter a imagem (flip)
- Rotacionar de um ângulo aleatório
- Zoom aleatório na imagem

- Contraste aleatório
- Brilho aleatório
- Sinta-se livre para adicionar outro processo se quiser

Escolha duas técnicas acima para implementar na mão usando numpy, e depois descubra como fazer utilizando bibliotecas.
Aqui você terá que criar a função, mas quando for treinar seu modelo, há formas de fazer isso direto, procure na documentação de tensorflow.

Tarefa 4

Outro processo comum é o de aplicar filtros para melhorar a qualidade da imagem dentre outros resultados que se pode obter. Para isso, aplique em uma imagem de sua escolha um filtro de *blur, gaussian blur* e *median blur* (pode usar OpenCV, mas se quiser aprender como fazer na mão vai ser um grande aprendizado). Para entender mais como os filtros funcionam, **crie um kernel de sua escolha** e aplique na sua imagem utilizando OpenCV. Agora pode brincar um pouquinho tentando imaginar o que o filtro vai fazer com sua imagem.

Há diversos tipos de filtro que podem ser usados em uma imagem, é uma técnica bem conhecida para detecção de bordas em imagens.

Os filtros de *blur* costumam ser bons quando queremos um modelo mais robusto, que detecta nossa imagem mesmo com “noise”. Os filtros de média são bons quando queremos reduzir o “noise” da nossa imagem, e torná-la mais uniforme.

Tarefas de Classificação

“Como os robôs vão dominar o mundo se meu modelo não sabe a diferença entre um chihuahua e um bolinho?”, Luke para Yoda após seu modelo dar 27% de acurácia.

Tarefa 1

Para nosso modelo aprender, precisamos dar uma série de imagens pra ele treinar, e algumas imagens pra ele testar se está aprendendo. Assim, sua tarefa é dividir suas imagens em um diretório com imagens de treino, teste e validação (não vale fazer isso na mão kkkkkk).

Mais uma vez, tensorflow e keras possuem funções de fazer isso, qualquer dúvida entre em contato, aqueles que querem se aventurar podem fazer uma função que mexe direto no diretório usando Os, pathlib e glob. A ideia dessa função seria acessar as pastas do diretório, criar duas novas, de treino e teste, e selecionar fotos aleatórias para inserir no diretório de treino até que se tenha o número suficiente de acordo com o argumento de split dado, ou seja, a porcentagem das imagens que serão usadas para treino.

Tarefa 2

Agora que você dividiu seu diretório, você deve transformar suas imagens em um forma que seu modelo consegue aprender, recomendamos que você use o tensorflow.Keras, ou keras para os mais íntimos, para isso. Essa biblioteca é amplamente usada em machine learning e dispõe diversas funções para transformar seu diretório em um dataset, além de que é *user friendly.* Sinta se livre para usar o PyTorch se quiser, mas acreditamos que keras é mais fácil de usar para iniciantes.

Aqui você deve transformar sua imagem em um tensor, ou então aprender a usar funções como flow_from_directory que fazem isso diretamente.

Tarefa 3

Nesta terceira tarefa, você deve criar um modelo qualquer para classificar suas imagens. Experimente transformá-las em um vetor 1xN e aplicar uma regressão logística por exemplo.

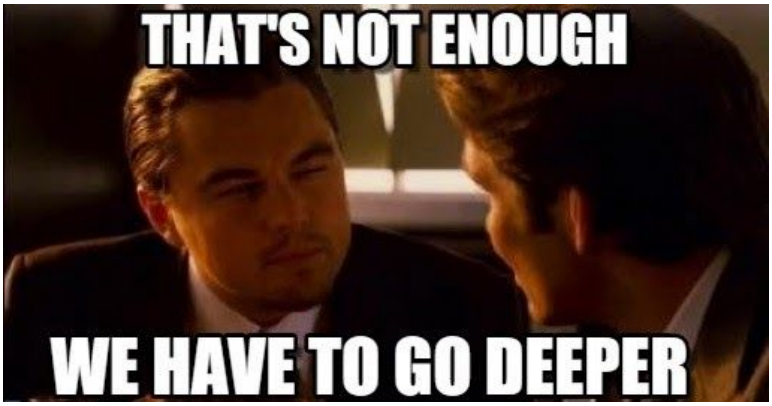
criar uma rede convolucional usando Tensorflow, Keras ou PyTorch (recomendamos Keras) para treinar seu modelo. Sinta-se livre para criar uma rede inspirada em algum tutorial ou rede famosa de classificação de imagens.
Não se esqueça de testar a acurácia do seu modelo com essa primeira rede.

Tarefa 4

Crie agora uma rede convolucional usando Tensorflow, Keras ou PyTorch (recomendamos Keras) para treinar seu modelo e sinta o poder do lado negro, quer dizer, da Convolução. Sinta-se livre para criar uma rede inspirada em algum tutorial ou

rede famosa de classificação de imagens.
Não se esqueça de testar a acurácia do seu modelo com essa rede e compare com a anterior.

Tarefa 5



Chegou a hora de botar fogo no parquinho.

- Use data augmentation no seu modelo
- Teste pelo menos dois tipos de funções de ativação
- Use MaxPooling no seu modelo
- Adicione mais camadas de convolução

Faça seu melhor, mas lembre que a ideia aqui é aprender os caminhos da força, não precisa gastar todo o tempo do mundo para conseguir um modelo com 90% de acurácia.

Você é um verdadeiro Jedi!

Se você chegou vivo até aqui, você já sabe mais que o necessário para nos ajudar e contribuir nos nossos projetos de Visão Computacional, junte-se ao nosso lado, acreditamos que temos outras coisas muito legais para te ensinar, e se não, vamos aprender, aplicar e disseminar juntos!

Links úteis para as tarefas

Recomendamos sempre também se aventurar nas documentações, tutoriais no medium, toward data science e pyimagesearch, mas no nosso drive de CV também temos alguns vídeos rápidos de ver que dão uma ótima introduzida em visão computacional. Também estamos introduzindo alguns links que podem ser úteis caso você trave em alguma tarefa.

- 1.1
  - https://www.geeksforgeeks.org/python-pil-image-show-method/
  - https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.imshow.html
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html
  - https://learn.datacamp.com/courses/image-processing-in-python
- 1.2
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html
  - https://medium.com/tulisan-ibe/image-conversion-using-python-and-opencv-8cd7c0ef7051
- 1.3
  - https://towardsdatascience.com/data-augmentation-for-deep-learning-4fe21d1a4eb9
  - https://sigmoidal.ai/reduzindo-overfitting-com-data-augmentation/
  - https://www.tensorflow.org/tutorials/images/data_augmentation (mais legal :))
- 1.4
  - https://towardsdatascience.com/image-filters-in-python-26ee938e57d2
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html
  -
- 2.1
  - https://wizardforcel.gitbooks.io/tensorflow-examples-aymericdamien/5.1_build_an_image_dataset.html
  - https://www.kaggle.com/freeman89/create-dataset-with-tensorflow
- 2.2
  - https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
  - Pode olhar exemplos no site do tensorflow em alguns tutoriais para entender melhor
- 2.3
  - https://www.kaggle.com/gulsahdemiryurek/image-classification-with-logistic-regression
- 2.4

    - https://www.tensorflow.org/tutorials/images/cnn
    - https://keras.io/examples/vision/mnist_convnet/
- 2.5
    - https://keras.io/examples/vision/image_classification_from_scratch/
    - https://keras.io/api/layers/regularization_layers/