



Parte 2: Tarefas de Classificação

Projeto de Trainee da área de Visão Computacional



Introdução



"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.

Agora que você já sabe tudo sobre as imagens, pode seguir para a próxima parte de sua missão: fazer um classificador! Nessa parte, você vai aprender quais são os modelos mais comuns em CV e como implementá-los.

Tarefa 1

Para nosso modelo aprender, precisamos dar uma série de imagens para treiná-lo, e algumas imagens para ele testar se está aprendendo. Assim, sua primeira 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 possui funções para fazer isso, qualquer dúvida entre em contato. Aqueles que querem se aventurar podem fazer uma função que mexe direto no diretório usando bibliotecas como Os, pathlib e glob. A ideia dessa função é acessar as pastas do diretório, criar três novas (treino,

validação 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 informado, ou seja, a porcentagem das imagens que serão usadas para cada grupo.

Tarefa 2

Agora que você dividiu seu diretório, você deve transformar suas imagens de forma que seu modelo consiga aprender. Sinta-se livre para usar o framework (PyTorch, Keras ou TensorFlow) que achar melhor ou mais simples se quiser.

Tanto no TensorFlow quanto no PyTorch, você precisa carregar suas imagens e transformá-las em tensor. Ainda, no TensorFlow é possível usar funções como `flow_from_directory` que fazem isso diretamente. Caso tenha dúvidas, não deixe de olhar as referências ou chamar um dos membros da área para te dar aquela ajuda ;).

Tarefa 3

Nesta terceira tarefa, você deve criar um modelo para classificar suas imagens. Primeiramente, transforme-as em um vetor $1 \times N$ e aplique uma regressão logística. Se quiser experimentar outros modelos aqui, sinta-se livre para isso, só não tente usar uma rede neural ainda rs. Também não se esqueça de verificar as métricas do modelo, como a acurácia!

Tarefa 4

Crie agora uma rede convolucional 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 comparar com o anterior.

Aqui vai uma listinha de redes canônicas em que você pode se inspirar (~~e copiar~~) para criar o seu modelo:

- <https://www.tensorflow.org/tutorials/images/cnn>
- https://keras.io/examples/vision/mnist_convnet/
- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html

Se quiser se aventurar um pouco mais, tente algumas coisas como:

- Usar data augmentation no seu modelo
- Testar pelo menos dois tipos de funções de ativação
- Usar MaxPooling no seu modelo
- Adicionar 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.

Tarefa 5



"Na vida nada se cria, tudo se copia." - Wesley Pereira de Almeida, quem algumas lendas dizem ser a reencarnação de Obi-Wan Kenobi.

Agora que você entendeu todo o poder das redes convolucionais, podemos tentar algo diferente que foi criado pelos deuses e fornecido para nós meros mortais: o transfer learning. Dessa forma, vamos experimentar uma rede extremamente profunda e previamente treinada, capaz de extrair as features das imagens, como bordas, texturas, demais padrões e gradientes. Como a rede é previamente treinada, aproveitamos esses pesos já treinados e adicionamos poucas camadas treináveis para torná-lo mais acurado para nossa tarefa.

Para isso, os frameworks padrões já possuem métodos que nos permitem carregar diversos modelos pré-treinados. Assim, podemos reaproveitar o loop de treinamento do item anterior apenas adicionando a nova rede ao fluxo. Ficou com alguma dúvida? Não se preocupe, aqui vai algumas referências fornecidas pelas documentações oficiais:

- https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
- https://www.tensorflow.org/tutorials/images/transfer_learning

Uma vez que você alterou a rede, verifique se será necessário fazer alguma alteração no formato de entrada das imagens (nesse momento, a documentação é sua maior aliada). Teste, reteste, tente outros modelos e já sabe, né? Qualquer dúvida, chame algum membro de CV.

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 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 indicamos alguns links que podem ser úteis caso você trave em alguma tarefa:

- 2.1
 - https://wizardforcel.gitbooks.io/tensorflow-examples-aymeric-damien/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/
 - https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
 - https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html
 - https://keras.io/examples/vision/image_classification_from_scratch/
 - https://keras.io/api/layers/regularization_layers/
- 2.5

- https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
- https://www.tensorflow.org/tutorials/images/transfer_learning