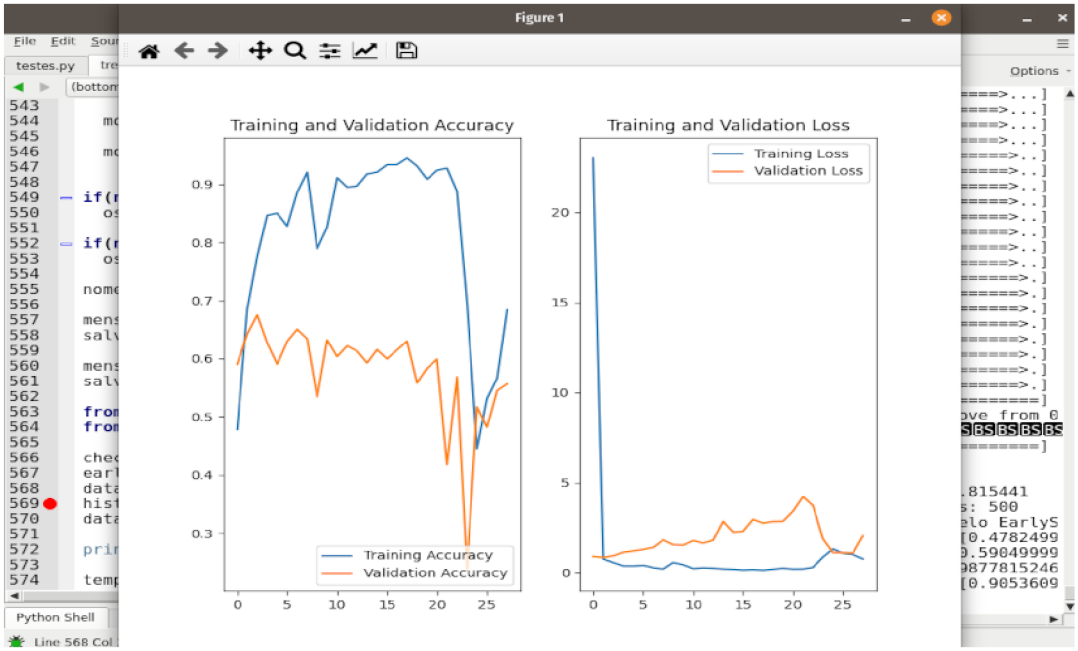
Sistema Comparador de *Convolutional Neural Networks* (CNNs)

Dizem que: -"Quanto mais preguiçoso for um programador, melhor ele será. Pois maior será seu empenho para automatizar as tarefas e assim reduzir seu trabalho diário".

Bom.. Apresento aqui um projeto para programadores bem preguiçosos.



Um problema recorrente em todo inicio de projeto na área de análise de imagens assistida por inteligência artificial (principalmente se for preciso embarcar, como ocorre em AIOT) é escolher a melhor topologia para a tarefa com base no limite de processamento disponível e a precisão oferecida pela topologia escolhida.

É muito importante atentar quanto ao trade of do projeto. Em alguns projetos será necessário obter a maior precisão possível, sem importar os custos de hardware. Já em outros a precisão não é tão importante. Também é possível que se deseje por a maior precisão possível dentro do limite de hardware disponível (embarcado).

Em todos os casos, é bom poder comparar de forma relativamente rápida algumas topologias que cobrem desde as mais leves (com menor custo de processamento, porém com menor capacidade de análise), até as mais pesadas (maior capacidade de análise mas com maior custo de processamento).

E como programador preguiçoso que sou, montei um script que é o foco deste artigo.

O script permite comparar rapidamente 11 topologias diferentes de redes, configuradas para a tarefa de classificação de imagens. Todas treinadas com o mesmo dataset e assim poder comparar os resultados das classificações e seus respectivos consumos de recursos.

O dataset utilizado para o treinamento e comparação das topologias poderá ser o dataset completo do projeto final de forma a permitir uma melhor base de comparações ou mesmo uma versão parcial do dataset apenas dar dar um panorama geral nos casos em que não se tem muito recurso disponível.

Este script também foi focado no aspecto didático, pois cada rede foi implementada de uma forma diferente. Justamente para aproveitar a oportunidade e abordar diferentes técnicas de construção. E como programadores preguiçosos, ainda poderemos aproveitar a implementação total ou parcial destes testes em nosso projeto final.

As redes escolhidas foram (em ordem crescente de capacidade de análise e consumo de recursos de processamento): LeNet, KarpathyNet, SmallVGG, VGG16, VGG19, ResNet50, U-Net (Não operacional com os datasets de exemplo citados), ResNet101, ResNet152, Inception V e a Xception.

Na data em que este artigo foi escrito já existem topologias mais inovadoras como a DALL-E 2, Visual Transformers, GANs... Mas como geralmente quem se preocupa com limites de hardware está lidando com sistemas embarcados e estas topologias mais novas requerem muito mais poder de processamento, não me preocupei em abordá-las neste momento.

O projeto foi dividido em dois scripts. Um primeiro script de treinamento das topologias e um segundo script que vai testar as topologias treinadas.

Muito importante é que estes são scripts que eu uso no meu dia a dia. Todas as redes implementadas funcionam. Porém é preciso observar que como são redes já construídas, requerem datasets com imagens em dimensões pré-definidas. Nenhuma dos datasets neste artigo tem o padrão de imagem para conseguir treinar a U-NET, esta portanto irá gerar erro no treinamento com estes datasets de exemplo (A menos que se redimensione as imagens).

É possível comparar os tempos de treinamento e execução de cada topologia, a acurácia calculada pelo algoritmo de treinamento, a acurácia real em percentual de acertos em testes realizado com um dataset separado, gráficos de treinamentos...

Para demonstrar o funcionamento dos scripts foi incluído no projeto dois datasets open sourse:

Dataset 01: Texturas de Tumores cancerígenos.

Dataset 02: Folhas de Oliveiras. Para a classificação de raças quanto ao formato das folhas.

Topologias configuradas como classificadoras podem tanto classificar a imagem em geral reconhecendo objetos (mas sem delimitar a região gráfica do objeto na imagem) como podem fazer a classificação de texturas.

A exata mesma rede pode executar uma ou outra tarefa (ou até as duas ao mesmo tempo) sem precisar de alterações em suas configurações. O que muda na realidade é a forma de ajustar os objetos na cena da imagem.

O que é óbvio.. Para classificar objetos nas imagens se espera ver o objeto inteiro na cena. Já a textura, se espera ver a textura em toda a área da imagem.

Ambos os datasets representam tarefas de classificação de imagens. Porém o dataset de texturas de tumores representa uma "tarefa mais leve" aonde a maioria das topologias oferece uma boa margem de precisão.

Já o dataset das folhas de oliveiras representa uma tarefa de classificação muito mais árdua, pois oferecem poucos dados para a rede pois o que diferencia cada subespécie da planta são sutis variações no formato das folhas. Somente as topologias mais pesadas obtiveram uma margem razoável.

Justamente por isto achei importante incluir este dataset.. Pelos fins didáticos..

Se fosse um projeto comercial que exigisse algo acima de 90% de precisão eu apelaria para uma Dall-E 2 ou uma DCGAN. Pois a precisão não foi satisfatória com as topologias testadas..

Existe também um outro aspecto no treinamento destes algoritmos... Estou assumindo que o dataset está perfeito.. Pois não detenho o conhecimento necessário para realizar a classificação das folhas por conta própria. Por isto pretendo realizar o teste de classificação das folhas com topologias mais poderosas... Mas pela minha experiência, metade das topologias abordadas deveriam ter dado resultados satisfatórios.. Da VGG16 pra cima..

Existem fatores ligados ao dataset que podem causar esta precisão anêmica mesmo com a mais poderosa de todas as topologias.. Por exemplo.. As imagens podem estar pequenas demais, oferecendo assim pouquíssimas informações para as redes. Eu testei e não é isto... Contribui sim..

Mas tem mais problemas ai.. Pode ocorrer também da quantidade de imagens estar muito abaixo do ideal.

Quanto maior a topologia, maior deverá ser o dataset. Sem dúvidas, estamos com poucas imagens..

E pode ocorrer ainda algo mais "gritante"... A qualidade do dataset não ser boa.

Tem um ditado que diz: Se alimentar a sua rede com lixo, ela somente irá cuspir lixo...

Não domino o conhecimento técnico de classificação das folhas... Logo não tenho como conferir.

Se fosse um projeto comercial e precisasse de mais de 90% de precisão, eu focaria na qualidade do dataset. No mínimo triplicando a quantidade de amostras.

E também submeteria este dataset aos cuidados de 3 ou mais especialistas do ramo de forma a garantir a veracidade da separação das amostras por classes.

Ter um bom dataset representa uns 95% do trabalho, chuto eu (é uma estimativa obtida mediante um chute técnico)…

Código fonte e material complementar do projeto podem ser encontrados em:

https://github.com/AndreFCosta608/ComparativoClassificadores

Descrição de arquivos:

- *resnet50.py* → Arquivo auxiliar não executável diretamente que implementa a topologia Resnet 50.
- *uNet.py* → Arquivo auxiliar não executável diretamente que implementa a topologia U-Net.
- *treinamento.py* → Arquivo principal executável diretamente. Realiza o treinamento das topologias (algumas configurações são realizadas no início do código). * Requer uso de GPU.
- *testes.py* → Arquivo principal executável diretamente. Realiza alguns testes extras, além dos testes de durante o treinamento, de desempenho. * Uso de GPU opcional.

DATASETs:

1. KAGGLE. Colerectal Histology MNIST. 2022. Disponível em: https://www.kaggle.com/datasets/kmader/colorectal-histology-mnist. Acesso em: 04 Jan. 2023.
2. MURATKOKLU. Grapevine Leaves. 2022. Disponível em: https://www.muratkoklu.com/datasets/vtdhnd10.php. Acesso em: 03 Jan. 2023.

REFERÊNCIAS:

1. COLAB. **Unet Pet Segmentation**. 2022. Disponível em: https://colab.research.google.com/github/margaretmz/image-segmentation/blob/main/unet_pet_segmentation.ipynb#scrollTo=uckkCSj5TpxD. Acesso em: 01 Fev. 2023.
2. DATAHACKER. **TF Implementing a VGG-19 network in TensorFlow 2.0**. 2020 Disponível em: https://datahacker.rs/vgg16-vgg19-network-implemented-tensorflow-2-0/. Acesso em: 12 Jan. 2023.
3. GITHUB. **Resnet_Model_TF2.0**. 2021. Disponível em: https://github.com/indiantechwarrior/Resnet_Model_TF2.0. Acesso em: 22 Jan. 2023.
4. KAGGLE. **Keras U-Net starter – LB 0.277**. 2018 Disponível em: https://www.kaggle.com/code/keegil/keras-u-net-starter-lb-0-277/notebook. Acesso em: 16 Jan. 2023.
5. MEDIUM. **GoogLeNet (InceptionV1) with TensorFlow**. Artificial Intelligence in Plain English. 2021. Disponível em: https://ai.plainenglish.io/googlenet-inceptionv1-with-tensorflow-9e7f3a161e87?gi=249240d2abe4. Acesso em: 12 Jan. 2023.
6. MEDIUM. **Step by step VGG16 implementation in Keras for beginners**. Towards Data Science. 2019. Disponível em: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c?gi=63d35a1f40ae. Acesso em: 27 Jan. 2023.
7. MEDIUM. **Xception: Implementing from Scratch using Tensorflow**. Towards Data Science. 2020. Disponível em: https://towardsdatascience.com/xception-from-scratch-using-tensorflow-even-better-than-inception-940fb231ced9?gi=fc3b45a24f1e. Acesso em: 16 Jan. 2023.
8. PYIMAGESEARCH. **How to (quickly) build a deep learning image dataset**. 2018 Disponível em: https://pyimagesearch.com/2018/04/09/how-to-quickly-build-a-deep-learning-image-dataset/. Acesso em: 01 Fev. 2023.
9. PYIMAGESEARCH. **Multi-label classification with Keras**. 2018. Disponível em: https://pyimagesearch.com/2018/05/07/multi-label-classification-with-keras/. Acesso em: 30 Jan. 2023
10. PYIMAGESEARCH. **U-Net Image Segmentation in Keras**. 2022. Disponível em: https://pyimagesearch.com/2022/02/21/u-net-image-segmentation-in-keras/. Acesso em: 22 Jan. 2023.
11. PYLESSONS. **TensorFlow Keras ResNet tutorial**. 2019. Disponível em: https://pylessons.com/Keras-ResNet-tutorial. Acesso em: 16 Jan. 2023.
12. PYPI.ORG. **Keras-unet-collection**. 2022. Disponível em: https://pypi.org/project/keras-unet-collection/. Acesso em: 22 Jan. 2023.