Relatório 22 - Prática: Reconhecimento de Emoções com TensorFlow 2.0 e Python (III)

Lucas Augusto Nunes de Barros

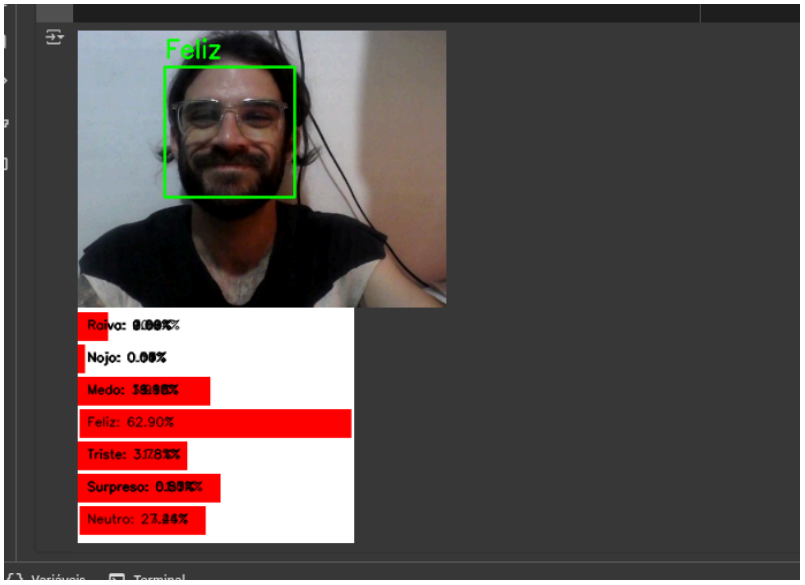
Descrição das atividades

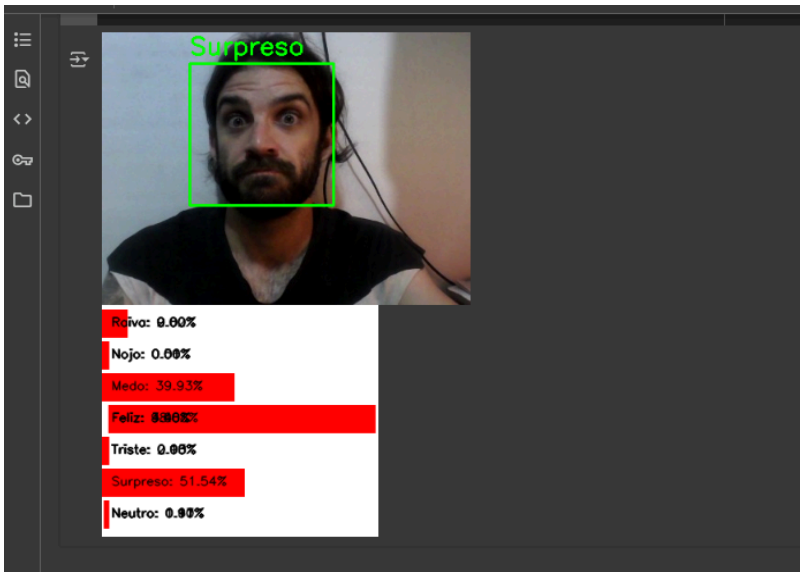
O reconhecimento de emoções, proposto no card, utiliza aprendizado profundo para classificar expressões faciais em categorias como raiva, felicidade, tristeza, medo e etc. O TensorFlow, como ja foi visto, é uma biblioteca amplamente utilizada para construir modelos de reconhecimento com imagens, graças ao suporte robusto a redes neurais convolucionais (CNNs), permitindo a criação, treinamento e o armazenamento dos modelos salvos em formatos como .h5, que contêm a arquitetura e os pesos aprendidos, possibilitando sua reutilização sem necessidade de um novo treinamento.

A utilização de um modelo pré-treinado trouxe um pequeno desafio de compatibilidade relacionado a versão do TensorFlow usada para salvar o modelo. No caso, o modelo de reconhecimento de emoções salvo em formato HDF5 (*Hierarchical Data Format version 5*) apresentou problemas devido a versão do TensorFlow instalado no Google Colab (TensorFlow 2.18.0) apresentar incompatibilidade em um parâmetro de função, resultado de uma alteração no formato de entrada. Além disso, erros de desserialização ocorreram devido a objetos inicializadores (GlorotUniform, Zeros, Ones) que incluíam um parâmetro dtype, tratado de forma diferente em versões antigas do TensorFlow (provavelmente anteriores a 2.17.0) em comparação com versões mais recentes.

Para garantir que o modelo pré-treinado funcionasse corretamente com o TensorFlow 2.18.0 no Google Colab, foram necessárias algumas modificações específicas no código proposto inicialmente pelo autor do curso. Primeiramente, novas classes personalizadas para inicializadores foram definidas, de forma à contornar os erros de desserialização, ignorando o parâmetro *dtype* incompatível e permitindo o carregamento de todas as camadas do modelo sem falhas. A seguir, para resolver a incompatibilidade do formato da entrada, a biblioteca h5py foi utilizada para acessar e corrigir a configuração do modelo armazenada no arquivo HDF5, ajustando o parâmetro *batch_input_shape* da primeira camada para [None, 48, 48, 1]. Isso envolveu a análise e a correção da estrutura aninhada de camadas Sequential e a reconstrução do modelo antes de carregar os pesos.

Em resumo, o modelo foi carregado e envolto por um novo modelo que possui uma camada de entrada redefinida que garante a compatibilidade do formato de entrada. Por fim, uma reconstrução da arquitetura foi necessária para assegurar que todos os parâmetros das camadas (como filtros, tamanhos de kernel e regularizadores) correspondessem ao original. Essas alterações resolveram os problemas de compatibilidade, permitindo que o modelo fosse carregado no ambiente do Google Colab mesmo utilizando da nova versão atualizada do TensorFlow.





Na seção 3 do curso é proposta a implementação de uma arquitetura que faça algo similar ao que foi mostrado na seção anterior, uma rede convolucional que faça o reconhecimento de emoções através do treinamento com o conjunto de dados *Fer2013,* que consistem em imagens de faces em escala de cinza com tamanho de 48x48 pixels. Os rostos foram registrados de forma que ficassem centralizados e ocupassem aproximadamente a mesma quantidade de espaço em cada imagem.

O resultado foi satisfatório e mais nenhum erro de compatibilidade surgiu no desenvolver desta etapa, uma vez que o erro causado anteriormente era devido ao conflito de versões entre o TensorFlow instalado na plataforma e o TensorFlow utilizado para salvar o modelo. O modelo foi implementado ficando ao fim com os seguintes valores estruturais.

```
Total params: 5,905,863 (22.53 MB)
Trainable params: 5,902,151 (22.51 MB)
Non-trainable params: 3,712 (14.50 KB)
```

Adicionalmente, ao fim da seção são disponibilizados alguns tópicos especiais como detecção de emoções em vídeos, uso da técnica de *Data Augmentation* para otimizar o treinamento de redes CNN, transferência de aprendizado utilizando o modelo VGG16. além de trazer a implementação de uma rede *Inception.* Esse último tipo de rede introduziu o conceito de módulo *Inception*, característica principal da arquitetura, projetada para melhorar a eficiência computacional e o desempenho em tarefas de visão computacional.

O módulo *Inception* traz benefícios ao trocar o empilhamento de camadas convolucionais de maneira linear, como em arquiteturas tradicionais, por processamento da entrada de forma paralela, com várias operações convolucionais e de *pooling*, combinando os resultados em uma única saída. Isso permite que a rede capture diferentes tipos de características e em diferentes escalas, sem aumentar excessivamente a complexidade computacional.

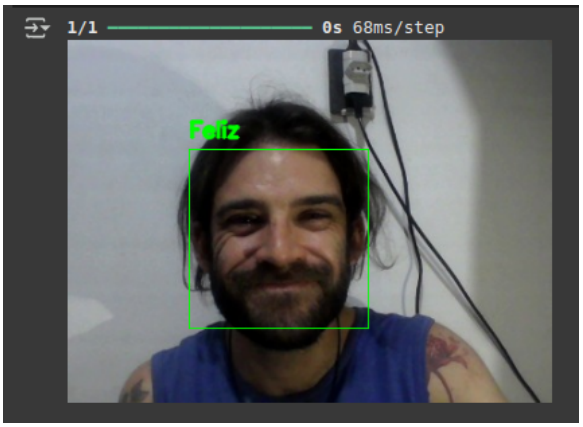
Por fim, a quarta seção do curso traz um apanhado de conceitos importantes e faz uma revisão em tópicos como redes *perceptron* de uma ou mais camadas, como funciona o cálculos dos pesos durante o treinamento do modelo, gradiente descendente e sua aplicação no processo de treinamento.

O *perceptron* é a unidade fundamental de uma rede neural, recebendo entradas, aplicando pesos, somando-os e passando por uma função de ativação. Uma rede *perceptron* de uma camada resolve apenas problemas lineares. Redes perceptron multicamadas já conseguem modelar relações não lineares.

O gradiente descendente minimiza a função ajustando pesos na direção oposta ao gradiente (taxa de maior crescimento). Enquanto a taxa de aprendizado controla o tamanho dos passos, o gradiente descendente guia esses passos na direção correta. O *backpropagation* é um algoritmo usado para calcular o erro da saída e fazer a propagação para trás, ajustando pesos com base no cálculo do gradiente descendente.

O *bias* desloca a função de ativação, aumentando a flexibilidade da rede. O erro é a diferença entre previsão e valor real, podendo ser calculado com diferentes bases dependendo do tipo de aplicação. O parâmetro delta quantifica a contribuição individual dos neurônios para o erro total. É calculado na camada de saída com base no erro e na derivada da função de ativação.

Finalizado o curso, foi desenvolvida uma prática para capturar uma imagem da webcam, processar e enviá-la ao modelo, obtendo então uma resposta relativa a emoção expressa pela pessoa na imagem.



Conclusão

A abordagem de reconhecimento de emoções utilizando redes neurais convolucionais e ferramentas como TensorFlow e OpenCV, representa um importante conhecimento sobre visão computacional, permitindo a implementação e entendimento sobre a estrutura de uma rede neural suficientemente complexa para reconhecer expressões faciais com alta precisão. O curso deixa claro que a combinação de ferramentas permite criar modelos robustos que podem ser utilizados em aplicações práticas.

Referências

[1] PICO.NET. What is the difference between 'SAME' and 'VALID' padding in tf.nn.max_pool of tensorflow? . Disponível em: <https://www.pico.net/kb/what-is-the-difference-between-same-and-valid-padding-in-tf-nn-max-pool-of-tensorflow/> . Acesso em 19 de maio de 2025

[2] KAGGLE.COM. FER2013. Disponível em: <https://www.kaggle.com/datasets/msambare/fer2013> . Acesso em 20 de maio de 2025

[3]DEEPAI.ORG. Compreendendo o Módulo Inception em Deep Learning. Disponível em:

<https://deepai.org/machine-learning-glossary-and-terms/inception-module> . Acesso em 20 de maio de 2025

[4] card 22 - Reconhecimento de Emoções com TensorFlow 2.0 e Python