MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAIANO
CAMPUS-GUANAMBI

CAIO MAX, HIAGO COUTO, LAYNE CASTRO, MATHEUS MAGALHÃES,
MATHEUS NUNES

RELATÓRIO DE TÓPICOS AVANÇADOS EM ANÁLISE E PROJETOS DE SISTEMAS DE INFORMAÇÃO

GUANAMBI
2019

1. INTRODUÇÃO

O presente relatório apresenta uma descrição sucinta do processo de desenvolvimento de um sistema desktop para classificar as imagens, utilizando uma rede neural convolucional, intitulado SNG - *Classifier*. O mesmo possibilitará ao usuário o envio do database contendo as imagens para a classificação, as quais a rede deverá reconhecer e classificá-las em suas devidas categorias, retornando a qual categoria a imagem pertence.

Na primeira seção do trabalho é exposto um referencial teórico acerca dos principais conceitos sobre redes neurais convolucionais, para melhor entendimento do trabalho. Na seção dois, tem-se a descrição das ferramentas e linguagens utilizadas para desenvolvimento da interface do sistema e de toda codificação da rede neural. A próxima seção descreve todo o processo de desenvolvimento, e finalmente a seção que traz imagens do sistema em funcionamento.

2. REDES NEURAIS CONVOLUCIONAIS

Tendo como base o trabalho de Araújo et. al. (2017), as Redes Neurais Convolucionais (CNNs) são definidas como um grupo de algoritmos que são baseados nas redes neurais artificiais, empregando a convolução em alguma camada. Transformaram-se no novo modelo de visão computacional, e demonstram facilidade para seu treinamento, se apresentado grande quantidade de amostras rotuladas.

Nesse sentido, suas várias vantagens consistem principalmente na eficiência de extraírem propriedades importantes por meio de treinamento de transformações (*kernels*), além de necessitar de um número menor de preceitos de ajustes em relação a redes totalmente conectadas.

Ainda segundo o autor, as CNNs são construídas por meio de uma sequência de camadas, em que cada uma destas dispõe de uma função particular na disseminação do sinal de entrada. Sendo três principais camadas, as convolucionais, de pooling, e totalmente conectadas.

As camadas convolucionais, se apresentam como um grupo de filtros que obtêm como entrada um arranjo 3D, igualmente denominado de volume. Embora as dimensões de cada filtro dessa camada sejam reduzidas, ela se desdobra por toda a

profundidade do volume da entrada da camada.

Nesse segmento, os filtros são ajustados automaticamente durante o procedimento de treinamento, para que sejam ativos apenas diante a presença de atributos relevantes apresentados no volume. Cada filtro ocasiona uma estrutura ligada localmente que percorre toda extensão do volume da entrada.

Posterior a camada convolucional, apresenta a camada de pooling. Que tem como objetivo fazer a redução gradativamente da dimensão do volume de entrada. Com essa redução, consequentemente o custo computacional da rede diminui e evita *overfiting*, quando um modelo se configura muito bem ao conjunto de dados observado, mas se mostra ineficiente para prever novos resultados.

No pooling, a sua forma mais comum constitui-se na recolocação de valores de uma região pelo valor máximo, sendo denominada como max pooling. Esse processo reduz a dimensão da representação dos dados e acelera os procedimentos das próximas camadas.

A próxima camada denominada camada totalmente conectada, utiliza as saídas das camadas convolucionais e de pooling, no qual representa as características retiradas das imagens de entrada, e utiliza essas características para classificar a imagem.

Essa camada é formada por neurônios, sendo elementos de processamento, e o termo “totalmente conectado” quer dizer que todos os neurônios da camada antecedente estão conectados a todos os neurônios da camada consecutiva.

3. FERRAMENTAS E LINGUAGENS UTILIZADAS

3.1 Interface

3.1.1 *Qt Creator*

Com a IDE Qt *Creator* pode-se utilizar a biblioteca Qt em *Python*, a qual está permite a criação de diversos serviços em nossa aplicação em *Python*, como configuração de rede, banco de dados SQL, interface gráfica, etc. No nosso software a utilizamos para a criação da interface gráfica.

3.2 Rede Neural

3.2.1 Inception

A arquitetura de rede convolucional escolhida foi o Inception-v3 do *Google*. De acordo com Sundar et al. (2018), Inception-v3 é uma arquitetura de CNN desenvolvida pelo Google e, por sua vez, é uma evolução da arquitetura Inception-v1, também conhecido por GoogleNet, vencedora do concurso ILSVRC14 (*ImageNet Large Scale Visual Recognition Challenge* 2014).

Ainda segundo Sundar et al. (2018), a arquitetura consiste em uma série de redes de aprendizagem profunda com blocos repetidos.

Cada bloco corresponde a vários filtros convolucionais com tamanhos variando entre 5x5 e 1x1, o que cria um efeito de cascata a cada etapa. Este modelo garante que as informações destacadas e, por conseguinte, as características predominantes da imagem se perpetuem na arquitetura.

Sundar et al. (2018) também destaca a flexibilidade da arquitetura ligada ao uso de camadas convolucionais ou camadas aptas ao pooling. O Inception-v3 ainda

conta com um conjunto reduzido de parâmetros é serem utilizados se comparado com outras arquiteturas relevantes. A razão para tal, é a fatoração de camadas de convolução maior e sem camadas menores, entre outros métodos.

3.2.2 TensorFlow

Para auxiliar a construção do modelo, o *framework TensorFlow*, será utilizado. Em concordância com DEMIROVIĆ, SKEJIĆ; ŠERIFOVIĆ–TRBALIĆ (2018), o *TensorFlow* é uma base para o desenvolvimento de modelos em aprendizado de máquina, principalmente redes neurais. A comunidade em torno da plataforma vêm crescendo aceleradamente e tem se focado nos campos de treinamento e interferência em redes de aprendizagem profunda, além de áreas como processamento de imagens e otimização.

Essa ecleticidade no uso do framework só é possível devido a sua arquitetura extensível (DEMIROVIĆ; SKEJIĆ; ŠERIFOVIĆ–TRBALIĆ, 2018).

3.2.3 Keras

Para abstrair ainda mais o processo de construção do modelo, optamos por usar o Keras. O *Keras* é uma API de alto nível, para o desenvolvimento de redes neurais artificiais. A mesma é escrita na linguagem de programação *Python* e pode ser usada tendo o *TensorFlow* como base (TSAI; ZENG; CHANG, 2018).

3.2.4 Jupyter

Randles et al. (2017) define o *Jupyter Notebook* como um *software* de código aberto baseado em navegadores *web*, usado como um caderno para comportar dados, fluxos de trabalho, códigos de programação, rotinas e documentos, além de permitir a interoperabilidade entre membros de uma mesma equipe. A parte mais atraente dessa plataforma é a interação com múltiplos componentes como identificadores digitais, mecanismos de persistência, versionamento, agrupamento de dados, documentação, *software* e publicações.

3.2.5 *Google Colaboratory*

O *Google Colaboratory* é um projeto cuja finalidade se concentra na disseminação do conhecimento acerca do aprendizado de máquina e no apoio de pesquisas sobre o tema. Segundo Carneiro et al. (2018), os cadernos disponibilizados pelo *Colaboratory* são baseados nos do *Jupyter*, e atuam como um objeto do *Google* Docs. O resultado disto é um caderno virtual passível de ser compartilhado e alterado por diferentes colaboradores.

Ainda de acordo com Carneiro et al. (2018), a plataforma fornece ambiente sem diferentes versões do *Python*, já pré-configurados com as bibliotecas necessárias, por exemplo, o *TensorFlow* e o Keras. Por fim, essa ferramenta disponibiliza um serviço de processamento em tempo de execução acelerado por Unidade de Processamento Gráfico (GPU) (do inglês, *Graphics Processing Unit*).

3.2.6 Transferência de Aprendizagem

A Transferência de Aprendizagem (TL) (do inglês, *Transfer Learning*), será usada no treinamento, ou melhor, retreinamento do modelo. Salvador et al. (2018) traz a TL como um algoritmo que utiliza o conhecimento adquirido e os recurso de uma rede já treinada, para cumprir papel similar em um novo conjunto de dados. Salvador et al. (2018) ainda afirma serem várias as vantagens do uso da TL, além de ser uma abordagem ideal para conjuntos pequenos de dados, como é nosso caso. A saber, a CNN original utilizará a implementação do Inception-v3 disponível junto ao *Tensorflow*.

4. PROCESSO DE DESENVOLVIMENTO

O desenvolvimento do *software* foi dividido em três fases, sendo a primeira o desenvolvimento da rede neural convolucional, a segunda a criação da interface gráfica e por último a integração da rede e a interface, a primeira e segunda fase ocorreram simultaneamente.

Para o desenvolvimento da rede neural foi utilizado o *Keras* e o *TensorFlow*, para criar o modelo da rede, com x camadas ocultas, utilizando um dataset que continha diferentes imagens de flores para classificação. A rede foi testada com sucesso no *Google Colaboratory.*

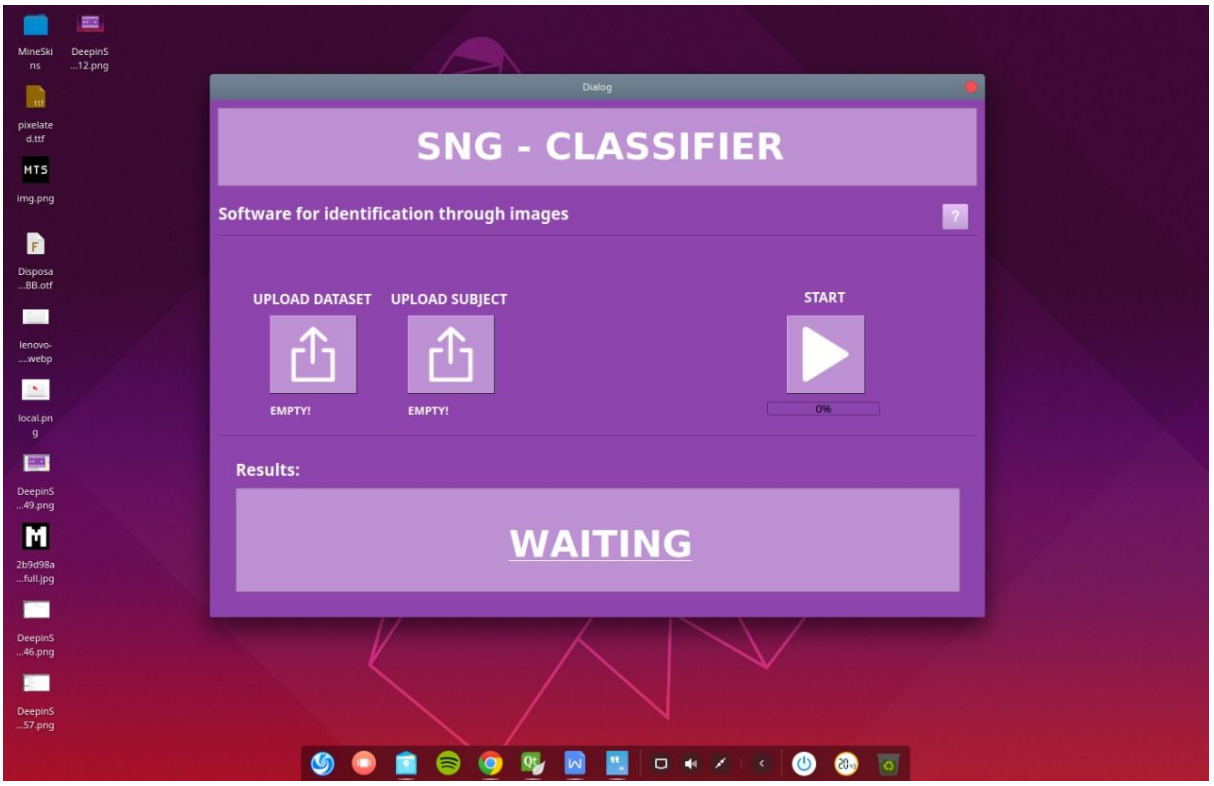
Com a IDE *Qt Creator*, pode-se utilizar a o *Qt Designer*, e criar a única e principal tela do sistema, com a junção de elementos como botões e labels, os quais já estão prontos para ser utilizados na tela, precisando apenas alterar alguns parâmetros para a personalização.

Na tela são atribuídas as funções de upload do dataset e subject e teste, com isso também é possível iniciar a análise, e posteriormente mostrar o resultado na tela para o usuário.

Seguindo para a última fase, foi realizada a integração da *CNN* com a interface gráfica, possibilitando assim o upload de diferentes datasets, no qual serão analisados pela rede desenvolvida, após o acionamento do botão *Start*.

5. DEMONSTRAÇÃO DO SISTEMA

Figura 1: Tela principal do sistema



Fonte: Elaboração própria (2019)

6. CONCLUSÃO

Para conclusão do trabalho, algumas considerações se fazem importantes, como as dificuldades encontradas durante o processo de desenvolvimento. Tais como, complicações em manipulação de diretórios do Python, visto que a equipe não dominava a linguagem, empecilhos para a escolha do modelo de classificação, uma vez que depende da correta instalação das bibliotecas do sistema.

Em consequência disso, alguns usuários podem não conseguir executar a aplicação pois as bibliotecas necessárias podem não ter as devidas configurações. Além disso, encontrou-se também dificuldades na escolha da melhor ferramenta para criação da interface.

Embora os problemas citados, conclui-se que o presente projeto colaborou de forma significativa para o aprendizado de todos da equipe executora, acerca dos métodos e procedimentos para a criação de um classificador de imagens utilizando redes neurais convolucionais.

7. REFERÊNCIAS

ARAÚJO, Flávio HD et al. Redes Neurais Convolucionais com Tensorflow: Teoria e Prática. SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos, v. 1, p. 382-406, 2017.

CARNEIRO, Tiago et al. Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. IEEE Access, v. 6, p. 61677-61685, 2018.

DEMIROVIĆ, Damir; SKEJIĆ, Emir; ŠERIFOVIĆ–TRBALIĆ, Amira. Performance of Some Image Processing Algorithms in Tensorflow. In: 2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP). IEEE, 2018. p. 1-4.

RANDLES, Bernadette M. et al. Using the Jupyter notebook as a tool for open science: An empirical study. In: 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL). IEEE, 2017. p. 1-2.

SALVADOR, Rodolfo C. et al. DeepTronic: An Electronic Device Classification Model using Deep Convolutional Neural Networks. In: 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM). IEEE, 2018. p. 1-5.

SUNDAR, KV Sai et al. Evaluating Training Time of Inception-v3 and Resnet-50,101 Models using TensorFlow across CPU and GPU. In: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2018. p. 1964-1968.

TSAI, Yi-Ting; ZENG, Yu-Ren; CHANG, Yue-Shan. Air Pollution Forecasting Using RNN with LSTM. In: 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, 2018. p. 1074-1079.