Michel André Lima Vinagreiro

TUTORIAL FINAL DE TREINAMENTO DOS ALGORITMOS UTILIZADOS NO PROJETO DE RECONHECIMENTO FACIAL DE APLICAÇÃO VEICULAR

Faculdade de tecnologia de Santo André

Janeiro de 2020

O objetivo deste tutorial é permitir a reprodução dos experimentos, pesquisa e resultados por meio da explicação da concepção do projeto, métodos utilizados, ferramentas confeccionadas e aplicadas e efeitos observados.

1. Considerações iniciais.

Antes de iniciar a reprodução da pesquisa mencionada, é importante seguir os passos descritos neste tutorial com fidelidade. Toda pesquisa foi desenvolvida utilizando ferramentas e pacotes de softwares de determinadas versões, porém a utilização de outros pacotes e versões pode ser estudada, pois podem impactar positivamente na melhoria do projeto.

1.2. Ferramentas de *hardware* utilizadas.

Na etapa de aquisição das amostras(imagens) de treinamento e testes, o *hardware* utilizado consiste de uma câmera de resolução mínima de 640 x 480 *pixels*.

Para realizar os tratamentos iniciais nas imagens, foi utilizado um microcomputador pessoal de configurações *core I5 vpro*, *8GB RAM, SSD 240GB,* sistema operacional *Ubuntu 18.04 (Linux)*. No geral, o processo de tratamento e preparação dos conjuntos de imagens não demanda elevado poder computacional.

O treinamento da rede *CNN* é realizado utilizando-se o serviço de computação em nuvem da empresa *Google*, denominado *Colab Notebook*, que disponibiliza gratuitamente uma *GPU* (unidade de processamento gráfico) modelo *Nvidia Tesla K80*, responsável por acelerar o processo de treinamento.

O *hardware* utilizado para aplicação em modo de predição é uma plataforma de *hardware embarcado* que consiste em um mini microcomputador denominado *RaspiBerry Model 3B*, muito difundido no ramo de eletrônica embarcada

1.3. Ferramentas de *software* utilizadas.

O primeiro passo consiste em instalar a interface de aplicações denominada *Python*, em sua versão *3.6*. A instalação, no sistema operacional, é realizada utilizando o comando:

sudo apt-install python 3.6.8

Por padrão, o instalador de pacotes do *python*, *pip*, é instalado como componente durante o processo acima descrito.

A partir do ponto em que a instalação foi concluída, o *pip* é utilizado para realizar a instalação das bibliotecas necessárias.

A primeira instalação são das bibliotecas *numpy, scipy, matplotlib e opencv-contrib-python*:

pip install numpy, scipy
pip install matplotlib
pip install opencv-contrib-python

Uma vez que os componentes acima citados foram instalados corretamente, a extração dos *frames* a partir dos arquivos de vídeo pode ser realizada.

A etapa posterior, que consiste na localização das faces, somente poderá ser realizada após a instalação de duas bibliotecas adicionais:

pip install dlib
pip install face_recognition

A etapa final de instalação dos pacotes básicos utilizados no projeto se dá pela instalação dos pacotes *tensorflow* e *keras*:

pip install --upgrade tensorflow
pip install keras

2. Etapas do processo de treinamento e confecção de bases de dados.

2.1. Extração de imagens a partir de arquivos de vídeo.

Após instaladas todas as bibliotecas mencionadas, o ponto de partida de todo o trabalho é a gravação de vídeos, cuja duração deve ser de em média 3 minutos, capturando uma variada gama de expressões das faces dos usuários que se desejam reconhecer pelo sistema. É importante que os usuários, durante a gravação dos vídeos, se mantenham olhando diretamente para a câmera.

Após gravados os vídeos, o próximo passo consiste em extrair destes vídeos, as imagens contendo as faces dos usuários. As imagens então, são nomeadas de forma ordenada e padronizada e então são armazenadas em uma pasta.

A aplicação *lervideo.py* tem como função realizar a tarefa acima mencionada. Antes de executar o programa, a pasta que receberá as imagens deve ser criada e, dentro do programa, as linhas de programa que contenham os caminhos das pastas onde se encontram o vídeo e da pasta que receberá as imagens resultantes devem ser editadas.

2.2. Extração dos vetores de características.

A segunda tarefa a ser realizada consiste em encontrar, isolar e representar as características das faces de cada usuário. A aplicação *treina_face.py* recebe "*n*" imagens de cada usuário, extrai as faces e gera, para cada face, um vetor de características. Cada vetor é acompanhado de uma *string* que consiste, basicamente, do nome do respectivo usuário. Todos os vetores resultantes deste processamento são arranjados, dando origem a uma matriz que representa a base de dados, que será utilizada posteriormente pelo modelo de predição.

De forma semelhante a abordada na subseção anterior, os caminhos das pastas contendo as imagens referentes as faces de cada usuário devem ser alterados.

Durante os testes executados, a proporção do números de imagens pertencentes a classe de usuário "intruso" para as demais classes (usuário 1, usuário 2, ..etc) é de 100 para 1, isso garante que dentre o grande volume de amostras que representam as faces, a probabilidade de que uma face desconhecida seja classificada como pertencente a determinado usuário de forma incorreta é reduzida.

O *dataset* público *FEI database* é um conjunto de imagens de faces capturadas por pesquisadores da instituição *Centro Universitário FEI*. O *FEI dabase* foi utilizado como banco de imagens para o usuário "intruso".

O restante das imagens que não foram utilizadas na geração dos vetores integrantes da base de dados podem ser utilizadas para treinar e testar a rede *CNN* responsável pela classificação mais acurada, uma vez que a face já foi pré-classificada no modo de predição, utilizando a base de dados.

2.3. Montagem dos conjuntos de treinamento e teste da rede CNN utilizando *keras*.

A primeira tarefa consiste em preparar a estrutura de pastas utilizada pela biblioteca em que a rede *CNN* foi gerada. A biblioteca *keras* não necessita que as amostras sejam rotuladas, pois, já faz a separação das classes pela estrutura de pastas, sendo que o rótulo numérico atribuído a classe obedece a ordem alfabética da das pastas. A estrutura de pastas deve ser organizada como se segue:

/IMAGENS
- *TREINAMENTO*
  - *USUÁRIO1*
    - *1.jpg*
    - *2.jpg*
    - *3.jpg*
    - *…*
    - *n.jpg*
  - *USUÁRIO 2*
    - *1.jpg*
    - *2.jpg*
    - *3.jpg*
    - *…*
    - *n.jpg*
- *TESTE*
  - *USUÁRIO 1*
    - *1.jpg*
    - *2.jpg*
    - *3.jpg*
    - *…*
    - *n.jpg*
  - *USUÁRIO 2*
    - *1.jpg*
    - *2.jpg*
    - *3.jpg*
    - *…*
    - *n.jpg*

Uma vez montada a estrutura de pastas, o programa *keras_dataset_make.py* é utilizado para extrair as imagens das faces dos usuários das imagens anteriormente extraídas dos arquivos de vídeos, realizando o redimensionamento das imagens das faces para a resolução padrão de entrada da rede CNN e salvando-as no formato "*jpg*" dentro das pastas que representam os conjuntos de dados de treino ou testes, para um determinado usuário.

Dentro do programa devem ser editadas as linhas de programa que contenham o número de imagens para cada conjunto, o caminho das imagens a serem processadas e o caminho de destino das imagens das faces recortadas. Após os conjuntos serem montados, todos as imagens devem ser verificadas antes de iniciar o treinamento da rede CNN.

2.4. *Treinamento da rede CNN*.

Uma vez montados os conjuntos de treinamento e testes o programa *treina_keras.py* deve ser executado. As linhas de programa que informam os números de amostras(imagens) pertencentes aos conjuntos de treinamento e testes, bem como o caminho onde se encontram as amostras devem ser editadas. Se o número de amostras que se possui for diferente do número de amostras informadas nas linhas de programa, parâmetros adicionais no programa devem ser alterados, tais como o número de épocas e o tamanho do lote de treinamento. O ideal é que o tamanho do lote de treinamento seja uma subdivisão do número total de amostras de treinamento.

A menos que o computador que será utilizado para treinar a rede possua uma quantidade de memória *RAM* elevada e seja dotado de uma *GPU* suficientemente poderosa para a tarefa, o treinamento utilizando recursos de computação em nuvem deve ser descartado.

A técnica mais comum quando se deseja treinar uma rede programada em um *desktop* e que necessita ser treinada em nuvem é a compactação de todos os arquivos e pastas que serão utilizados no treinamento, mantendo a estrutura de pastas e garantindo que quando o pacote for descompactado, tal estrutura se mantenha. Após realizado o treinamento, o modelo treinado pode ser utilizado posteriormente para predição.

A arquitetura da rede CNN é um dos pontos que podem ser flexibilizados para melhorar a velocidade e acurácia da aplicação.