Trabalho prático 01 - Optical Character Recognizer (OCR)

SSC0715 - Sensores inteligentes

Augusto Ribeiro Castro - 9771421

1 de Novembro de 2019

1 Introdução

Como primeiro trabalho prático da disciplina SSC0715 - Sensores inteligentes, foi proposta a elaboração de um reconhecedor de caracteres (do inglês *Optical Character Recognizer - OCR*) a partir do uso de alguma rede neural de aprendizado profundo (*Deep Learning*) adaptada às novas classes por meio de *transfer learning*. A aplicação desenvolvida utiliza a biblioteca de código aberto TensorFlow, na versão 2.0, conforme o tutorial mais novo disponível no *site* [1] da biblioteca.

A técnica de *transfer learning* consiste em utilizar um modelo desenvolvido para uma primeira tarefa como ponto de partida para um novo modelo que será aplicado na resolução de outro problema. No contexto do aprendizado profundo aplicado ao processamento de imagens, a técnica é muito usada a partir de redes premiadas em desafios de reconhecimentos de imagens para extensos conjuntos de treino e de validação.

As redes premiadas de aprendizado profundo são disponibilizadas e reutilizadas em diversas outras aplicações, partindo do pressuposto que a camada de extração de características desses sistemas foi tão extensamente treinada e testada que, mesmo com seus pesos congelados, ela possui um desempenho extremamente satisfatório. Sendo assim, a técnica de *transfer learning* conforme foi explorada neste trabalho consiste em fazer uso da camada de extração de características de uma dessas redes conhecidas de aprendizado profundo e criar uma nova camada final de classificação para adaptar o sistema às classes de letras e números a serem reconhecidas.

Para alimentar os treinamentos necessários, bem como realizar a validação, foi padronizado o uso do conjunto de dados *The Chars74k* [2]. Ele é composto por 74 mil imagens de letras maiúsculas, minúsculas e dos dez algarismos numéricos, totalizando então 62 classes. O conjunto de dados é constituído de caracteres escritos à mão, de imagens de caracteres extraídas de fotos e de caracteres de fontes de computador.

Mais de 50 mil imagens do total são referentes às fontes de computador, de modo que elas acabam sendo muito influentes sobre a acurácia obtida. Além disso, os caracteres extraídos de fotos são divididos em fotos boas e fotos ruins. Para não prejudicar o desempenho do OCR, apenas as fotos identificadas como boas foram utilizadas.

2 Solução desenvolvida

Nas seções abaixo são discutidos aspectos da implementação feita do OCR, explicando a plataforma utilizada e como executá-lo, o pré-processamento feito e a topologia de rede utilizada. Detalhes maiores de implementação estão comentadas dentro do código entregue, de modo que os comentários próximos de cada bloco de teste tendem a explicar melhor as decisões tomadas.

2.1 Recursos utilizados

O trabalho foi desenvolvido utilizando a ferramenta Google Colab, integrada ao Google Drive. Pelo fato de os caminhos relativos de diretórios descritos em código serem referentes à ferramenta, é recomendado que o arquivo *transfer_learning_OCR.ipynb* entregue junto com este relatório seja colocado em um Google Drive e executado dentro do ambiente do Google Colab para garantir o funcionamento correto.

O Google Colab disponibiliza 12 GB de memória RAM ao usuário, além da possibilidade de aceleração por GPU, facilmente ativada a partir do menu *Runtime*, que possui a opção *Change runtime type*. É altamente

recomendado ativar a aceleração por GPU para a execução do código. Outra característica da ferramenta é que os dados são persistentes apenas dentro da seção, sendo apagados a cada 12h. Sendo assim, o código feito faz o *download* dos dados quando é modificado, faz a extração dos arquivos e os move para uma pasta específica que pode ser acessada por chamadas posteriores de código.

2.2 Processamento dos dados

Ao longo do desenvolvimento do projeto, antes da adição das imagens de fontes de computador ao conjunto de dados, a maior parte das imagens eram compostas de fotos. Sendo assim, para melhorar o desempenho, foi tentada a estratégia de transformar todo o *dataset* utilizado em imagens em escala de cinza. Para isso, como foi utilizada a classe *ImageDataGenerator* [3], foi definida uma função de conversão de RGB para escala de cinza identificada em código e que é executada pelo Keras em cada imagem depois das operações de reescala.

A fim de não mudar a topologia da rede escolhida, que necessita que cada imagem tenha três canais de cores, o resultado da conversão de RGB para escala de cinza é então replicada para cada um dos canais da imagem, a fim de manter o formato desejado. A documentação do Keras realizaria a conversão de RGB para escala de cinza automaticamente, mas a opção geraria imagens de apenas um canal, o que inviabiliza o uso da topologia escolhida.

Com relação ao formato das imagens, foi escolhido do formado de 224 por 224 pixels para permitir o treinamento da rede escolhida. As imagens foram dividias em lotes (*batches*) de 32 imagens, conforme o padrão do Keras. Além disso, o conjunto total de imagens foi dividido em 70% de imagens para treinamento e 30% das imagens para validação, totalizando assim 51905 imagens das 62 classes para treinamento e 22202 imagens das 62 classes para validação.

2.3 Características da rede

Foi utilizada a camada de extração de características da rede MobileNetv2, conforme feito pelo tutorial utilizado [1]. Acima da camada de extração de características, foi adicionada uma camada densa de 512 neurônios com função de ativação *Rectified Linear Unit*. Acima dessa camada, foi adicionada uma camada de *Dropout* com probabilidade de 20% de abandonar um neurônio, a fim de reduzir o risco de *overfitting* na rede desenvolvida. Por fim, foi adicionada uma última camada densa de 62 neurônios, a fim de realizar a classificação das classes envolvidas no problema. A última camada densa possui como função de ativação a *Softmax* para calcular a probabilidade de cada uma das classes para uma dada imagem. a topologia da rede pode ser visualizada na figura 1.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer_1 (KerasLayer)   (None, 1280)              2257984
_________________________________________________________________
dense_2 (Dense)              (None, 512)               655872
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 62)                31806
=================================================================
Total params: 2,945,662
Trainable params: 687,678
Non-trainable params: 2,257,984
_________________________________________________________________
```

Figura 1: Modelo completo das camadas da rede de aprendizado profundo utilizado no OCR, conforme mostrado pelo TensorFlow.

Para treinamento da rede, foram escolhidas 10 épocas, cada uma delas com número de passos igual ao número de lotes do conjunto de treino. Ao final de cada época, é realizada a validação do modelo treinado no conjunto de validação. A cada época é calculada também entropia e todas as estatísticas são armazenadas para serem analisadas posteriormente.

3 Resultados

Nesta seção são mostrados gráficos e imagens que exemplificam o desempenho do OCR desenvolvido. Primeiramente, com relação à acurácia do classificador, o gráfico da figura 2 mostra que a aplicação desenvolvida chega a ter cerca de 83% de acurácia no conjunto de validação e mais de 90% de acurácia no conjunto de treinamento. Com relação à entropia, o gráfico da grandeza da figura 2 mostra ainda que a entropia para o treino e para a validação tendem a cair com o avanço das épocas, indicando que o classificador melhorou ao longo do treinamento e que o número de épocas empregado está adequado.

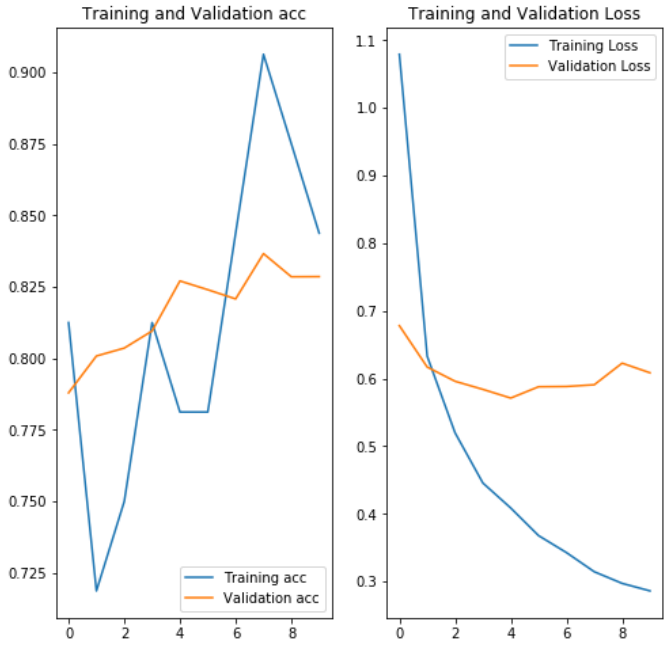

Figura 2: Gráficos da acurácia e da entropia em função nas épocas para a validação e para o treino.

As imagens mostradas nas figuras 3 e 4 mostram ainda exemplos de imagens dos conjuntos de treino e de teste com as respectivas classificações. O resultado parece bastante satisfatório, dado que poucos erros são mostrados e ainda assim, dentro dos erros, alguns são facilmente explicados pelo fato de algumas letras minúsculas serem muito parecidas com a versão maiúscula, sendo diferenciadas muitas vezes apenas pelo tamanho.

4 Conclusão

Os resultados apresentados na seção anterior permitem afirmar que o OCR desenvolvido possui funcionamento bastante satisfatório. Uma acurácia de validação de 83% num sistema que possui 62 classes é um valor extremamente mais alto que a acurácia possível com tentativas aleatórias. Além disso, o comportamento das métricas extraídas mostra que não aconteceu o *overfitting*, garantindo que o sistema possui capacidade de generalização.

Com relação à técnicas que poderiam ainda ser aplicadas para melhorar o desempenho do OCR, poderiam ser feitas as seguintes tentativas: *data augmentation* e o ajuste fino de parâmetros por meio do treinamento de parâmetros congelados da camada de extração de características. A primeira opção foi descartada pelo fato de algumas classes serem muito parecidas com outras após alguma operação de rotação, como por exemplo as letras M e W. Sendo assim, a aplicação da técnica poderia prejudicar o desempenho do classificador.

Já para o ajuste fino, seria necessário muito emprego computacional para treinar a rede, além do fato de na breve pesquisa realizada, não ter sido encontrada uma maneira de descongelar algumas camadas da MobileNetV2 ao invés de todas as camadas, conforme é possível realizar por exemplo com a rede InceptionV3. Sendo assim, a tentativa foi abortada pelo fato de o desempenho já estar satisfatório.

Figura 3: Conjunto de imagens de treino com respectivas classificações feitas pela rede neural.

Figura 4: Conjunto de imagens de validação com respectivas classificações feitas pela rede neural.

Referências

[1] Tutorial mais novo de *transfer learning* do TensorFlow. Disponível em: <https://www.tensorflow.org/tutorials/images/transfer_learning_with_hub>. Acesso em 18 de out. de 2019.

[2] Conjunto de dados utilizado. Disponível em: <http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/>. Acesso em 19 de out. de 2019.

[3] Documentação do Keras. Disponível em: <https://keras.io/preprocessing/image/>. Acesso em 19 de out. de 2019.