Projeto Demonstrativo 6: Reconhecimento de Faces

Pedro Henrique Luz de Araujo
pedrohluzaraujo@gmail.com
Raphael Soares Ramos
raphael.soares.1996@gmail.com

Departamento de Ciência da Comptutação
Universidade de Brasília
Campus Darcy Ribeiro, Asa Norte
Brasília-DF, CEP 70910-900, Brazil,

Abstract

O reconhecimento de faces tem como objetivo identificar faces de indivíduos específicos ou saber distinguir diferentes pessoas por meio de imagens das faces delas. No presente trabalho, desenvolvemos modelos treinados e avaliados em conjunto de dados de imagens de faces. O primeiro método implementado foi baseado em camadas completamente conectadas que têm como entrada vetores de imagens obtidos por rede convolucional profunda pré-treinada. Alcançamos resultados de acurácia no conjunto de validação superiores a 70%. Já a segunda parte do trabalho visa a detectar e reconhecer faces a partir de *face embeddings* obtidas do modelo OpenFace. [1]

1 Introdução

O reconhecimento de faces é uma tarefa de visão computacional com variadas aplicações práticas, entre elas, sistemas de segurança, sistemas biométricos, interações humano e computador e indexação automática de imagens. Tal tarefa pode se referir a alguns subproblemas, como: dadas duas imagens de rostos, decidir se pertencem ao mesmo indivíduo; dada uma imagem de rosto, decidir se tal rosto pertence a um indivíduo específico; e dada uma imagem de rosto, verificar se o rosto pertence a algum indivíduo de um conjunto específico de pessoas [7].

Em 1991, Turk e Pentland [15] propuseram um método de reconhecimento de faces que se tornou extremamente popular. A técnica consistia em projetar as imagens de face em um espaço de características contendo as variações significantes entre as imagens. As características obtidas foram denominadas *eigenfaces*, por serem os autovetores (*eigenvectors*) do conjunto de faces. Uma vez obtidas as *eigenfaces*, elas podem ser usadas para o reconhecimento de faces por meio da medição da distância entre imagens no espaço de características.

Mais recentemente, as técnicas estado-da-arte em reconhecimento de faces usam redes neurais convolucionais [11, 13]. Além de obterem resultados muito melhores que outras técnicas de visão computacional, as redes neurais convolucionais têm a vantagem de dispensar a engenharia de características (*feature engineering*).

[1] Pedro contribuiu com a implementação, experimentos e testes da parte 1, além da escrita da introdução e partes da metodologia, resultados e conclusões referentes à parte 1. Raphael contribuiu com a implementação, experimentos e testes da parte 2, além da escrita das partes da metodologia, resultados e conclusões referentes à parte 2.

No presente trabalho treinaremos e avaliaremos diferentes modelos treinados no *dataset* de reconhecimento de faces *Labeled Faces in the Wild* [7] (LFW). Ele contém 13233 imagens de 5749 indivíduos, entre os quais 1680 têm mais de uma imagem no *dataset*. As imagens medem 250 por 250 pixeis e são resultantes de operações de re-escala e recorte em faces detectadas pelo detector Viola-Jones [16]. O objetivo do modelo é, dadas duas imagens, classificá-las como pertencentes a mesma pessoa ou não.

2 Metodologia

2.1 Ferramentas

Para realizar operações sobre matrizes e vetores, utilizamos a biblioteca de computação numérica em Python, NumPy. Usamos ainda, como linguagem, Python 3.5.2, e o gerenciador de bibliotecas Anaconda 3. Por fim, foram usadas as bibliotecas Keras [3] e Tensorflow [1] para o treinamento e construção de redes neurais. O treinamento foi realizado com o uso de uma placa de vídeo GeForce GTX 750 Ti.

2.2 Parte 1

2.2.1 O modelo

A primeira parte do trabalho tem como objetivo analisar a viabilidade do uso de uma rede convolucional pré-treinada como extrator de características de imagens para reconhecimento facial. Foi usada para esse propósito uma rede Xception [2] pré-treinada no ImageNet [5], base de dados com mais de 14 milhões de imagens.

A rede Xception é baseada em unidades de convoluções separáveis em profundidade (*depthwise separable convolutions*), em que a convolução é separada em duas operações: uma convolução em profundidade, que é espacial e independente entre os canais de entrada; e uma por pontos, que projeta os canais obtidos da operação anterior em um novo espaço de canais.

Uma vez que inicializamos a rede com parâmetros aprendidos no ImageNet, a intuição é que ela possa servir como um extrator de características de imagens. Isto é, dada uma imagem, a rede deverá retornar um vetor capaz de descrevê-la, que servirá de entrada para um classificador. Para tanto, retiramos as ultimas camadas (totalmente conectadas) da Xception e substituímos por uma camada de *average pooling* global em duas dimensões que retorna um vetor de 2048 dimensões.

Cada uma das imagens do LFW, com as faces já alinhadas pela técnica proposta por Huang [8], foi dada como entrada para a rede descrita, sendo salvos os vetores de saída em um dicionário. Dessa forma, não é necessário processar as imagens repetidas vezes durante o treino.

Para classificar as faces como pertencentes a mesma pessoa ou não, construímos um modelo inspirado no DeepFace [13]. Ele recebe como entrada um mapa de característica para cada um das duas imagens que serão comparadas, pré-calculados na etapa descrita acima. Em seguida ambos os vetores são combinados de alguma forma [2] e passam por uma camada completamente conectada e uma camada de Dropout [12] seguida por uma unidade com ativação sigmóide que gera a classificação.

[2] Mais sobre isso na seção de experimentos.

2.2.2 Experimentos

Realizamos experimentos variando três hiper-parâmetros e avaliando os resultados no conjunto de validação do *dataset*. A separação entre treino e validação já foi feita pelos criadores do dataset e consiste em 2200 pares de imagem para treinamento e 1000 pares para validação. Em cada um dos conjuntos, metade dos pares corresponde a faces do mesmo indivíduo (exemplos positivos) e a outra metade a faces de indivíduos diferentes (exemplos negativos).

Variamos conjuntamente o número de unidades da camada completamente conectada (128, 512 ou 1024) e o valor de probabilidade de Dropout (0.2, 0.5, 0.8). Uma vez obtidos os melhores valores para esses hiperparâmetros, experimentamos diferentes maneiras de combinar os vetores de características das duas imagens de entrada (soma, concatenação, subtração, produto interno e produto elemento a elemento). Dessa forma, ao todo foram treinados 14 modelos diferentes.

Como otimizador usamos *rmsprop* [14] para minimizar a função de entropia cruzada binária. Sejam p a previsão da rede e y o rótulo que se deseja prever, a função de entropia cruzada binária é

$$\textit{cross-entropy} = -(y\log(p) + (1-y)\log(1-p)). \tag{1}$$

O tamanho de *batch* usado foi 128 e o modelo foi treinado por 60 épocas, sendo que o salvamos apenas quando houvesse melhora na acurácia do conjunto de validação.

2.3 Parte 2

2.3.1 O modelo

Na segunda do parte do trabalho foi realizado: detecção de faces; cálculo das *face embeddings* para quantificar a face; treino de uma *Support Vector Machine* [6] (SVM) nas embeddings; reconhecimento de faces em imagens e vídeos. Para construir esse pipeline foi usado modelos de redes neurais profundas em dois pontos chaves: detecção de faces e extração dos vetores de 128 características (*embeddings*) que quantificam cada face na imagem. Uma *embedding* é representado por $f(x) \in \mathbb{R}^d$. Ela incorpora uma imagem x em um espaço euclidiano de d-dimensões.

O modelo responsável por quantificar cada face na imagem pertence ao projeto *OpenFace* [2], que é uma implementação em Python e Torch [10] de reconhecimento de faces com redes neurais profundas. Esta implementação veio da publicação do paper [11].

A rede do modelo *OpenFace* (*FaceNet*) é treinada para calcular as face embeddings de uma forma diferente. Cada *batch* dos dados inclui três imagens: a âncora (imagem atual), a imagem positiva e a imagem negativa. A âncora e a imagem positiva pertencem à mesma pessoa/face, enquanto as imagens negativas não. A rede neural computa as embeddings para cada face e então atualiza os pesos da rede usando a função *triplet loss* como parâmetro, conforme mostra a figura 1. É computada uma função $f(x)$, de uma imagem x em um espaço de características $\mathbb{R}^d$ de modo que a distância euclidiana entre todas as faces, independente das condições da imagem, de mesma identidade é pequena, enquanto a distância entre pares de imagens de diferentes identidades é grande. A rede consiste de uma *batch input layer* e uma rede neural convolucional profunda seguida por normalização L_2, que resulta na face embedding. Esse processo é seguido pela *triplet loss* durante o treinamento.

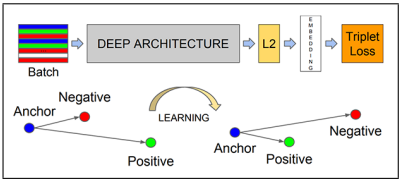

Figure 1: Estrutura do modelo utilizado.

2.3.2 Experimentos

Foram feitos experimentos no dataset LFW sem alinhamento e em um dataset gerado a partir dos alunos da turma. Em ambos os datasets é realizada a detecção e reconhecimento de faces. Foi selecionado um subconjunto do LFW para treino e um para teste. O mesmo foi feito com dataset gerado.

Primeiro é aplicado um detector de faces da OpenCV [9] para localizar e detectar faces nas imagens de entrada. Então, assumindo que cada imagem possui apenas um rosto, é proucrada uma região de interesse com maior probabilidade de conter um rosto e é verificado se essa probabilidade é maior do que uma constante estabelecida. Depois, esta região de interesse é passada pela rede convolucional profunda para gerar um vetor que descreve a face (*embedding*). Por fim, o modelo *SVM* é treinado para reconhecer novas faces.

3 Resultados

3.1 Parte 1

A figura 2 exibe a acurácia de treino e validação para diferentes valores dos hiperparâmetros dropout e número de unidades ao longo do treinamento. A figura 3 é análoga, mas corresponde aos modelos treinados com diferentes métodos de combinação de característica das imagens.

A Tabela 1 exibe os valores de acurácia na validação para cada um dos modelos treinados variando número de unidades e dropout. A Tabela 2 exibe a mesma métrica para modelos treinados com 512 unidades e 20% de dropout e diferentes métodos de combinação.

Modelo	Acurácia
Units_128_Drop_0.2	0.718
Units_128_Drop_0.5	0.721
Units_128_Drop_0.8	0.719
Units_512_Drop_0.2	**0.735**
Units_512_Drop_0.5	0.718
Units_512_Drop_0.8	0.726
Units_1024_Drop_0.2	0.72
Units_1024_Drop_0.5	0.728
Units_1024_Drop_0.8	0.722

Table 1: Resultados no conjunto de validação para o primeiro cenário de experimento de hiper-parâmetro.

O treinamento do modelo mostrou-se extremamente rápido, uma vez que as características das imagens fora previamente computada, levando menos de um minuto para concluir

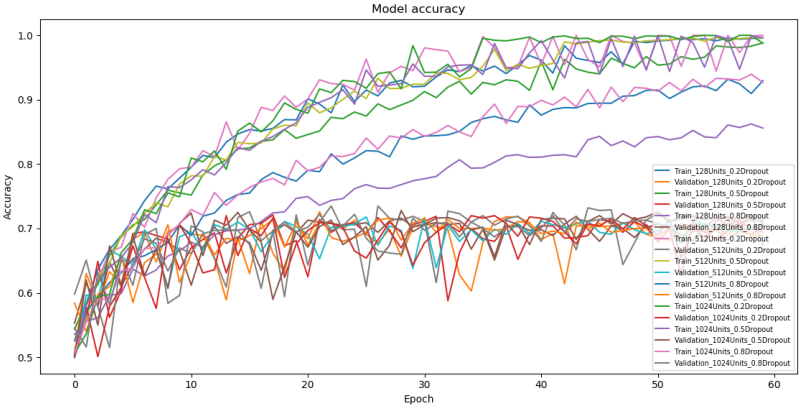

Figure 2: Acurácia no treino e na validação ao longo do treinamento para diferentes valores de dropout e número de unidades.

Modelo	Acurácia
concat	**0.729**
add	0.711
subtract	0.684
dotProduct	0.678
multiply	0.708

Table 2: Resultados no conjunto de validação para o segundo cenário de experimento de hiper-parâmetro.

60 épocas.

3.2 Parte 2

Nas figuras 4, 6 e 5 é testado o modelo construído de detecção e reconhecimentos de faces com imagens não vistas anteriormente pelo modelo.

4 Discussão e Conclusões

O modelo apresentado na Parte 1 conseguiu superar a *baseline* do conjunto de validação, alcançando 73,5% de acurácia, contra os 50% de *baseline*. O método de combinação das entradas que obteve melhor resultado foi a concatenação. Isso pode ser explicado pelo fato de a concatenação permitir que o modelo aprenda operações arbitrárias sobre as características das imagens em vez de se restringir a uma operação escolhida pelo usuário.

Por outro lado, nota-se das figuras 2 e 3 que ocorre *overfitting*. Isso porque há grande distância entre a acurácia na validação e no treino, que chega por vezes a 100%. Isso sugere que o modelo está aprendendo a classificar imagens não por meio do que realmente importa, isto é, as propriedades das faces, mas sim por outros elementos da imagem (iluminação, expressão facial, *background*). Esse problema de *overfitting* também foi encontrado por Taigman et al. [13] por causa do reduzido tamanho do conjunto de treino.

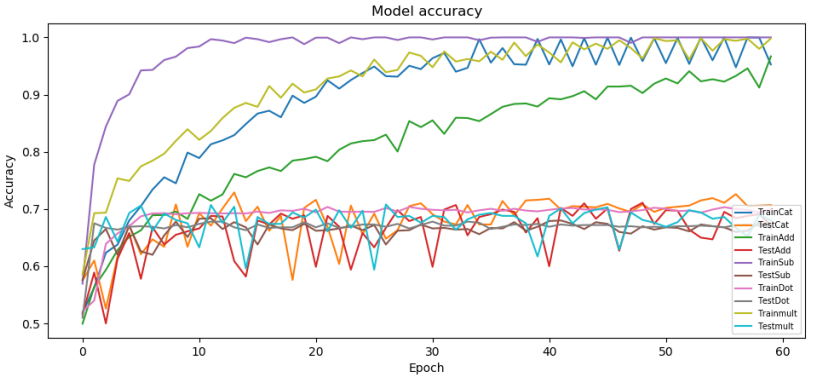

Figure 3: Acurácia no treino e na validação ao longo do treinamento para diferentes combinações de características de imagens.

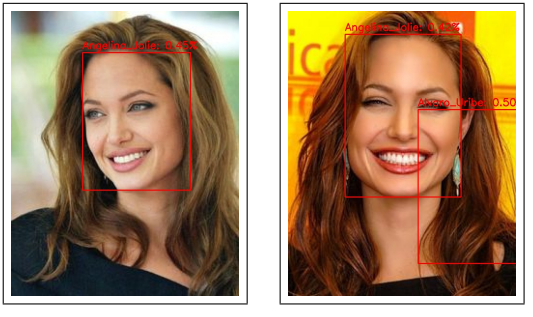

Figure 4: Fotos da Angelina Jolie testadas no pipeline construído de detecção e reconhecimento de faces.

Assim, seria interessante avaliar a performance do modelo em *datasets* maiores, a fim de reduzir o *overfitting*. Outra possível via de melhoria seria o uso de pesos do extrator pré-treinados em um conjunto de imagens do domínio de reconhecimento facial. Afinal, parte do erro de generalização do modelo pode se dar ao fato de o extrator ter "aprendido" a reconhecer objetos e não rostos. Por fim, uma maneira de aumentar o tamanho do *dataset* seria por meio da função custo *triplet loss* [1]. Dessa forma, o treinamento não ficaria restrito aos pares pré-selecionados de faces.

Dito isso, a técnica apresenta como vantagem a grande rapidez de treinamento, a simplicidade do modelo e desnecessidade de gerar características manualmente. Mostra-se necessário fazer as melhorias elencadas acima para avaliar a viabilidade do modelo para usos práticos.

O modelo apresentado na parte 2, por outro lado, mostrou-se muito bom para a detecção de faces, embora não tenha uma performance tão boa quanto ao reconhecimento de pessoa. Isso não é surpreendente, visto que há grande número de classes, muitas delas com poucos exemplos de treinamento. Seria interessante em um trabalho futuro avaliar o modelo no mesmo *dataset* usado na parte 1 a fim de comparar os resultados.

Figure 5: Fotos da Laura Bush testadas no pipeline construído de detecção e reconhecimento de faces.

Figure 6: Fotos do Robert Downey Jr testadas no pipeline construído de detecção e reconhecimento de faces. A classificação dada aqui foi errada.

References

[1] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[3] François Chollet et al. Keras. https://keras.io, 2015.

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL http://arxiv.org/abs/1610.02357.

[5] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[6] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[7] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[8] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In *NIPS*, 2012.

[9] *The OpenCV Reference Manual*. Itseez, 3.2.0 edition, Dezembro 2016.

[10] Koray Kavukcuoglu Ronan Collobert, Clement Farabet and Soumith Chintala. Torch - a scientific computing framework for luajit, 2017. URL http://torch.ch. [Online; accessed 30-Novembro-2018].

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

[13] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[14] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[15] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[16] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.