Reconhecimento de Emoções

Visão Computacional

Jorge Ribeiro

Introdução

Mais de 90% da comunicação humana é ***não-verbal***

Professor Albert Mehrabian, UCLA

O objetivo da inteligência artificial é compreender e se comunicar com o ser humano

- Saber o que o homem sente auxilia na comunicação
- A interação entre duas ou mais pessoas vai muito além das palavras pronunciadas
- Expressões faciais ajudam na contextualização

A popularização das CNNs e o aumento do poder computacional favoreceu o avanço do aprendizado de máquina

- Treinamento mais rápido
- Mais bases de dados disponíveis

Nos últimos anos, boas bases para reconhecimento de emoções surgiram

- Bons datasets públicos a partir de 2010
- Porém, alguns não eram grandes e generalizados o suficiente
- CNNs necessitam de um grande volume de exemplos para treinar de forma satisfatória

Desafios na tarefa de classificar emoções

- Erros de anotação no dataset
- Datasets gerados em laboratório (não generalizados)
- Classificar emoções é uma tarefa difícil, pois uma reação pode significar mais de uma emoção

Data set FER2013

Dataset disponibilizado em um challenge do Kaggle. Conta com mais de 35000 imagens.

Existem outros datasets famosos para emoções

Cohn-Kanade

Gerado em laboratório. Extenso, porém pouco generalizado.

RaFD

Também gerado em laboratório. Pouco generalizado e de difícil acesso.

O dataset FER2013 possui imagens “in the wild”

Preparação

Pré-processamento

Algoritmo de detecção de faces disponível no OpenCV (Haar-cascade).

Arquiteturas de Rede

Inicialmente uma versão modificada da AlexNet. Um treinamento também foi feito com uma versão levemente modificada da VGG.

Avaliação

Obtida pela visualização dos gráficos gerados pelo tensorboard (treinamento feito com tensorflow).

AlexNet modificada

Baseado no trabalho de Gudi, Recognizing Semantic Features in Faces using Deep Learning, 2016.

```
def build_alexnet(self):
    # Smaller Alexnet
    img_aug = tflearn.ImageAugmentation()
    img_aug.add_random_flip_leftright()
    img_aug.add_random_crop([SIZE_FACE, SIZE_FACE], padding = 4)
    print('[+] Building Alexnet')
    self.network = input_data(shape = [None, SIZE_FACE, SIZE_FACE, 1],
        data_augmentation = img_aug)
    self.network = conv_2d(self.network, 64, 5, activation = 'relu')
    self.network = max_pool_2d(self.network, 3, strides = 2)
    self.network = conv_2d(self.network, 64, 5, activation = 'relu')
    self.network = max_pool_2d(self.network, 3, strides = 2)
    self.network = conv_2d(self.network, 128, 4, activation = 'relu')
    self.network = dropout(self.network, 0.3)
    self.network = fully_connected(self.network, 3072, activation = 'relu')
    self.network = fully_connected(self.network, len(EMOTIONS),
        activation = 'softmax')
    self.network = regression(self.network, optimizer = 'momentum',
        loss = 'categorical_crossentropy')
    self.model = tflearn.DNN(self.network,
        checkpoint_path = SAVE_DIRECTORY + RUN_NAME,
        max_checkpoints = 1, tensorboard_verbose = 0)
    return self.model
```

Resultados

	AlexNet modificada		
Epochs	**50**	**100**	**400**
Acurácia	**62.4%**	**73.9%**	**70.7%**
Acurácia de Validação	**58.8%**	**62.18%**	**65.3%**

VGG modificada

Baseada na implementação padrão da VGG, feita em TFLearn. Disponível em https://goo.gl/4jHHX4

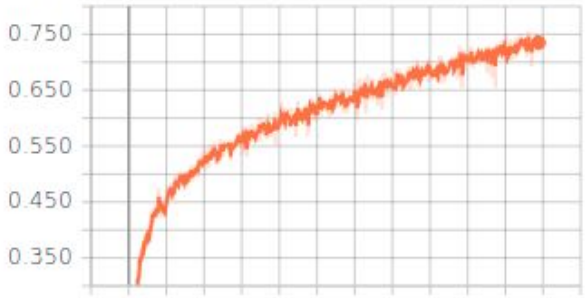
```
def build_vgg(self):
    # Smaller VGG
    img_aug = tflearn.ImageAugmentation()
    img_aug.add_random_flip_leftright()
    img_aug.add_random_crop([SIZE_FACE, SIZE_FACE], padding = 4)
    print('[+] Building VGG')
    self.network = input_data(shape = [None, SIZE_FACE, SIZE_FACE, 1],
        data_augmentation = img_aug)
    self.network = conv_2d(self.network, 64, 3, activation = 'relu')
    self.network = conv_2d(self.network, 64, 3, activation = 'relu')
    self.network = max_pool_2d(self.network, 2, strides = 2)

    self.network = conv_2d(self.network, 128, 3, activation = 'relu')
    self.network = conv_2d(self.network, 128, 3, activation = 'relu')
    self.network = max_pool_2d(self.network, 2, strides = 2)

    self.network = conv_2d(self.network, 256, 3, activation = 'relu')
    self.network = conv_2d(self.network, 256, 3, activation = 'relu')
    self.network = conv_2d(self.network, 256, 3, activation = 'relu')
    self.network = max_pool_2d(self.network, 2, strides = 2)

    self.network = conv_2d(self.network, 512, 3, activation = 'relu')
    self.network = conv_2d(self.network, 512, 3, activation = 'relu')
    self.network = conv_2d(self.network, 512, 3, activation = 'relu')
    self.network = max_pool_2d(self.network, 2, strides = 2)

    self.network = conv_2d(self.network, 512, 3, activation = 'relu')
    self.network = conv_2d(self.network, 512, 3, activation = 'relu')
    self.network = conv_2d(self.network, 512, 3, activation = 'relu')
    self.network = max_pool_2d(self.network, 2, strides = 2)

    self.network = dropout(self.network, 0.3)
    self.network = fully_connected(self.network, 3072, activation = 'relu')
    self.network = fully_connected(self.network, len(EMOTIONS),
        activation = 'softmax')
    self.network = regression(self.network, optimizer = 'rmsprop',
        loss = 'categorical_crossentropy', learning_rate = 0.0001)
    self.model = tflearn.DNN(self.network,
        checkpoint_path = SAVE_DIRECTORY + RUN_NAME,
        max_checkpoints = 1, tensorboard_verbose = 0)
    return self.model
```

Resultados

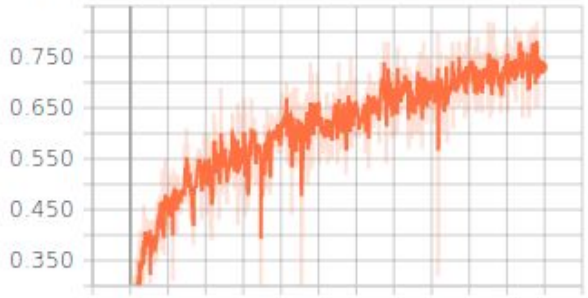
	VGG modificada
Epochs	**100**
Acurácia	**92.8%**
Acurácia de Validação	**65.9%**

Gráficos de acurácia e perda

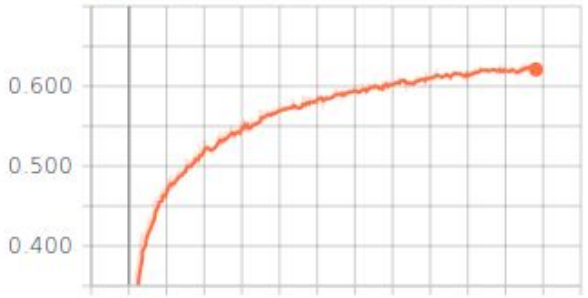
Accuracy



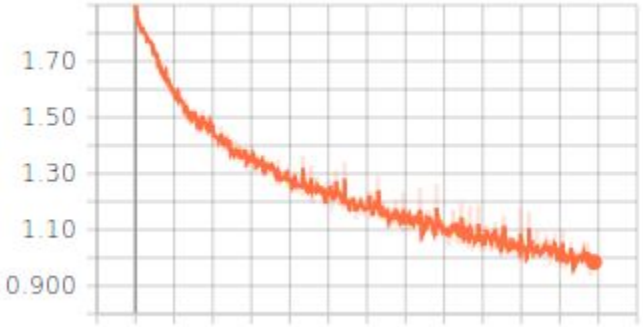
Accuracy/__raw_



Accuracy/Validation

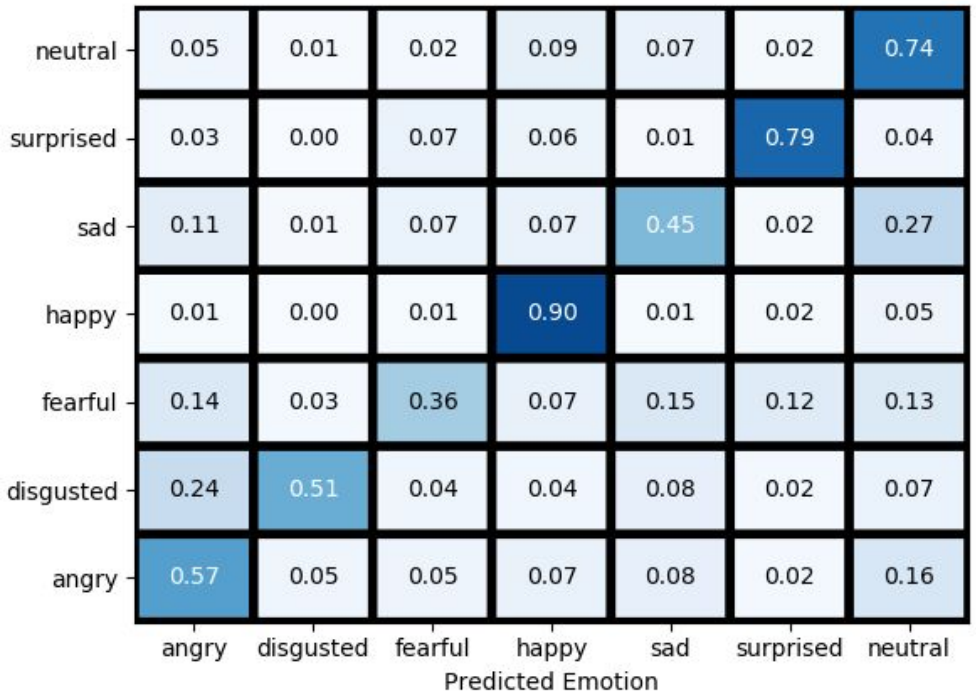


Loss

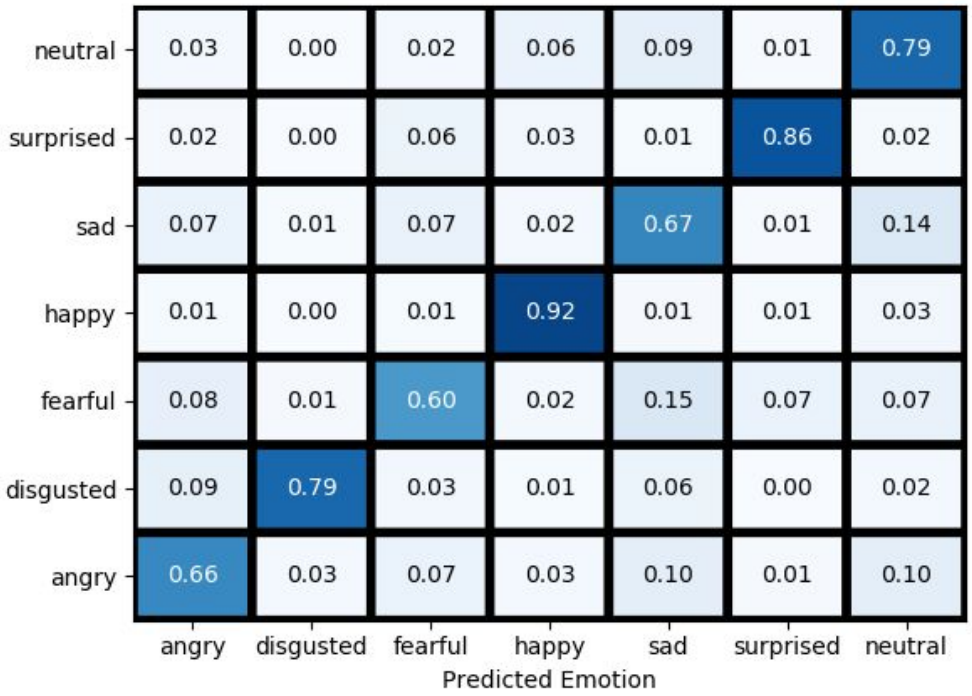


Loss/Validation

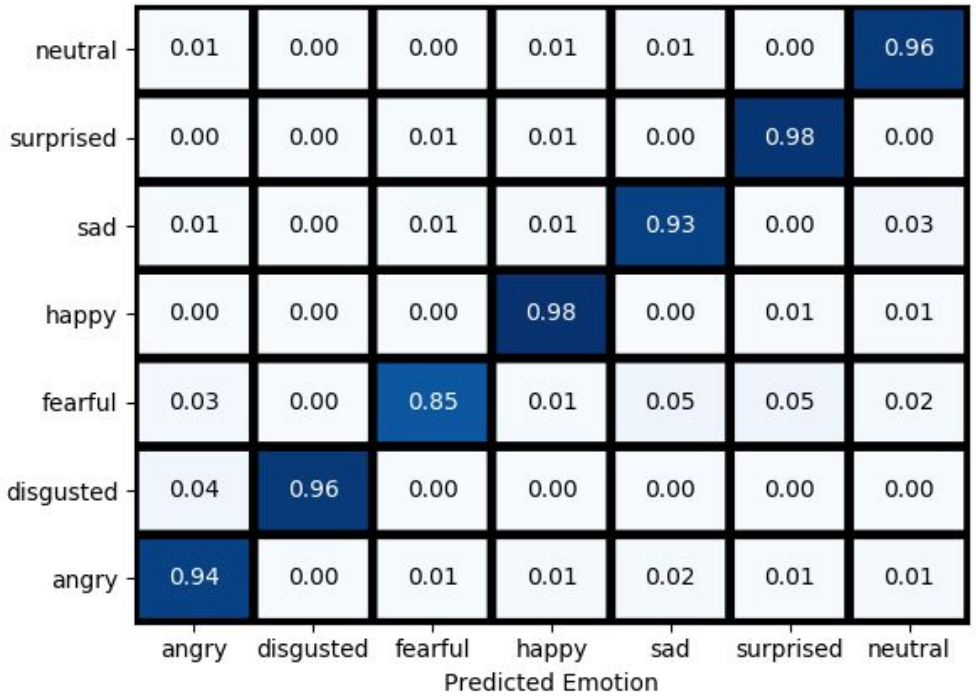
Matrizes de confusão - AlexNet com 100 epochs



Matrizes de confusão - AlexNet com 400 epochs



Matrizes de confusão - VGG com 100 epochs



Planos futuros

- Melhorar a acurácia de validação da VGG
- Testar diferentes arquiteturas
- Treinar com diferentes datasets

Muito obrigado!

Perguntas?

Referências

[1] Enrique Correa, Arnoud Jonker, Michael Ozo, Rob Stolk, Emotion Recognition using Deep Convolutional Neural Networks, 2016.

[2] Amogh Gudi, Recognizing Semantic Features in Faces using Deep Learning, 2016.

[3] Dan Duncan, Gautam Shine, Chris English, Facial Emotion Recognition in Real Time, 2016.

[4] Tanner Gilligan, Baris Akis, Emotio AI, Real-Time Emotion Detection using CNN, 2016.

[5] Ali Mollahosseini, David Chan, and Mohammad H. Mahoor, Going Deeper in Facial Expression Recognition using Deep Neural Networks, 2015.

[6] Alexandru Savoiu, James Wong, Recognizing Facial Expressions Using Deep Learning, 2017.

[7] Emotion Recognition With Python, OpenCV and a Face Dataset, http://www.paulvangent.com/2016/04/01/emotion-recognition-with-python-opencv-and-a-face-dataset/