UNIVALI

Visão Computacional

Reconhecimento facial utilizando Redes Neurais Convolucionais

Discente: Daniel Tatsch
Docentes: Eros Comunello e Wemerson Parreira

Tópicos abordados

- Seleção da base de dados
  - rap3df-database (2ª versão)
- Detecção / extração dos rostos nas imagens
  - Cascade Classifier - OpenCV
- Estudo implementação de Redes Neurais Convolucionais (CNNs)
  - TensorFlow
  - Avaliação da melhor configuração
- Generalização do modelo selecionado para todas os conjuntos de dados
  - Resultados obtidos

Seleção da base de dados

rap3df_data_02 - https://github.com/Piemontez/rap3df-database

- 80 pessoas
- Posições: frontal, perfil (esquerda e direita), cima, baixo, aleatória
- 119 x 149 pixels - RGB, infravermelho e em profundidade
- Exemplos de imagens:

Frontal | Esquerda | Direita | Cima | Baixo | Aleatória

Detecção da região da face

Haar-Cascade Classifier

- Uso de arquivos de treinamento pré definidos
  - *haarcascade_frontalface_default.xml*
- 72 faces detectadas corretamente (48 x 48 pixels)
  - Uso da região encontrada nas imagens frontais para as demais

Frontal | Esquerda | Direita | Cima | Baixo

Geração dos dos dados de treinamento e teste

```
all_rgb_pictures = rgb_loader.get_all_rgb_pictures()
faces            = face_detector.detect2(all_rgb_pictures)

for i in range(len(faces.keys())):
    X, y = generate_training_data(faces, i)

    X_name = 'dados2/X-{}.pickle'.format(i)
    y_name = 'dados2/y-{}.pickle'.format(i)

    pickle_out = open(X_name, 'wb')
    pickle.dump(X, pickle_out)
    pickle_out.close()

    pickle_out = open(y_name, 'wb')
    pickle.dump(y, pickle_out)
    pickle_out.close()
```

```
def generate_training_data(faces, indice):
    training_data = []

    for key, value in faces.items():
        class_num = list(faces).index(key)
        class_num = 0 if class_num == indice else 1

        for i in range(len(value)):
            img_grayscale = cv2.cvtColor(value[i], cv2.COLOR_RGB2GRAY)
            training_data.append([img_grayscale, class_num])

    random.shuffle(training_data)

    X = []
    y = []

    for features, label, in training_data:
        X.append(features)
        y.append(label)

    X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)

    return [X,y]
```

Redes Neurais Convolucionais (CNN)

- Scikit-Learn
  - Separação de grupos de treinamento e validação (20%)
  - Cálculo do peso das classes
    - Total: 360 imagens
    - Classe 0: 5; Classe 1: 355
- TensorFlow
  - Keras - Biblioteca de Rede Neural
  - TensorBoard - Avaliação de métricas como acurácia e perda

Redes Neurais Convolucionais (CNN)

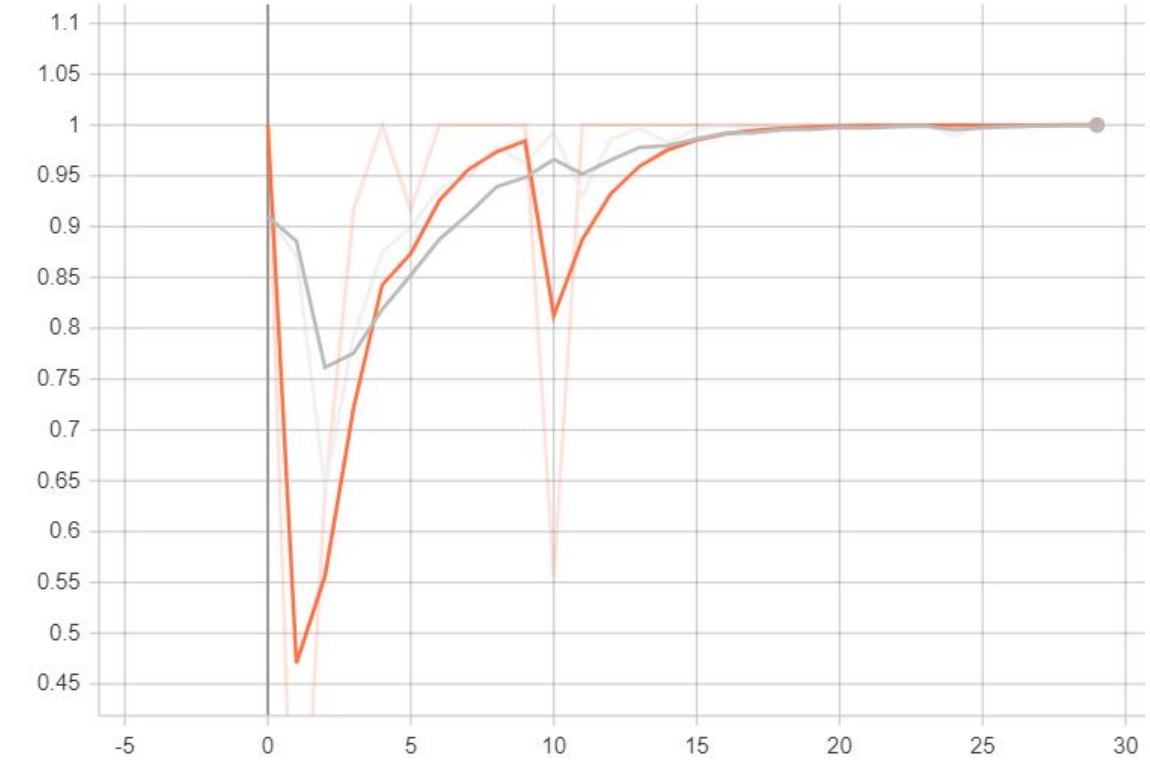
Avaliação da melhor configuração do modelo

- Camadas convolucionais = [1, 2, 3]
- Camadas densas = [1, 2, 3]
- Quantidade de funções de kernel e neurônios = [32, 64, 128]
- Treinamento por 30 épocas

Redes Neurais Convolucionais (CNN)

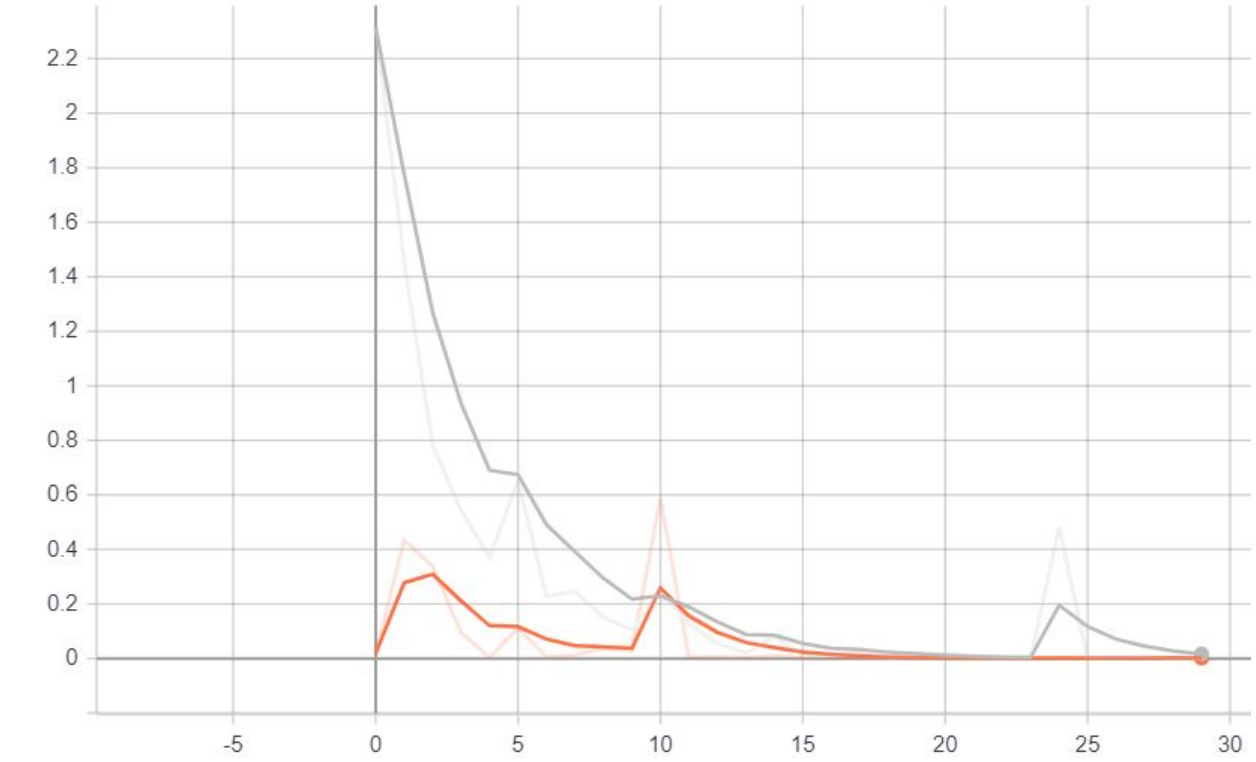
Análise via TensorBoard

epoch_accuracy



Name	Smoothed	Value	Step
1-conv-128-nodes-2-dense-1606001676\train	0.9996	1	29
1-conv-128-nodes-2-dense-1606001676\validation	1	1	29

epoch_loss



Name	Smoothed	Value	Step
1-conv-128-nodes-2-dense-1606001676\train	0.01651	2.3285e-4	29
1-conv-128-nodes-2-dense-1606001676\validation	1.6735e-3	7.8933e-4	29

Redes Neurais Convolucionais (CNN)

Generalização do modelo selecionado para todos os conjuntos de dados

```
model = Sequential()

model.add(Conv2D(64, (3,3), input_shape=X.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128, (3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dropout(0.3))

model.add(Dense(128))
model.add(Activation("relu"))
model.add(Dropout(0.3))

model.add(Dense(128))
model.add(Activation("relu"))
model.add(Dropout(0.3))

model.add(Dense(2))
model.add(Activation("softmax"))

model.compile(loss='sparse_categorical_crossentropy',
              optimzer='adam',
              metrics=['accuracy'])

history = model.fit(x=X_train, y=y_train, batch_size=32, epochs=30, class_weight=peso_classes, validation_data=(X_test, y_test))
```

Redes Neurais Convolucionais (CNN)

Resultados - Valores médios

- Acurácia: 98.73%
- Acurácia de validação: 98.74 %
- Perda: 0.05062
- Perda de validação: 0.1165
- Precisão: 98.84%
- Recall: 98.77%
- F1-score: 98.3%

Redes Neurais Convolucionais (CNN)

Resultados - Valores médios considerando as melhores épocas em termos de perda de validação

- Acurácia: 99.21%
- Acurácia de validação: 98.9%
- Perda: 0.03555
- Perda de validação: 0.04133

Considerações

- Mesmo com a aproximação de uma mesma região para todas as posições das faces e com a generalização de um mesmo formato de rede neural para todas os conjuntos gerados, o modelo apresentou resultados satisfatórios
- Porém, com testes utilizando as imagens em posições aleatórias, apenas 5 faces foram classificadas corretamente
    - Pode ser justificado pelo pouco número de amostras pertencentes à classe 0 nos dados de validação. Essa característica atribui um viés aos cálculos das métricas apresentadas

Considerações

Trabalhos complementares:

- Ajuste dos parâmetros do algoritmo de detecção de faces, localizando a região correta das faces em cada posição
- Variação da quantidade de dados para treinamento e validação
- Validação do modelo criado com bases de dados maiores

Códigos disponíveis em:

https://github.com/danieltatsch/mestrado/tree/visao_computacional/projeto_final

UNIVALI

Visão Computacional

Reconhecimento facial utilizando Redes Neurais Convolucionais

Discente: Daniel Tatsch
Docentes: Eros Comunello e Wemerson Parreira