Escola Politécnica da Universidade de São Paulo

PSI3471 - Fundamentos de Sistemas Eletrônicos Inteligentes

Classificação de grãos de arroz com rede convolucional

Davi Giordano Valério 11805273

São Paulo
2023

1. Introdução

O presente exercício programa aborda o problema de classificação utilizando redes convolucionais, em especial uma estrutura de rede inspirada no modelo LeNet-5. Para aplicar a rede, será utilizado um conjunto de dados equilibrado de grãos de arroz que foi reduzido para apenas 250 imagens com grãos de cinco tipos diferentes (Karacadag, Jasmine, Basmati, Ipsala e Arborio).

Dessa maneira, o presente exercício possui como objetivo classificar as diferentes variedades de arroz. Além disso, buscou-se analisar a pertinência de aplicar transformações geométricas, em especial, alinhar os grãos horiontalmente, para aumentar a performance da rede. Para esse projeto, será utilizada a linguagem Python na versão 3.9.

2. Métodos e Materiais

2.1 Alinhamento horizontal dos grãos

Para realizar o alinhamento horizontal dos grãos, foi desenvolvido um script em python 'alinha.py', em anexeo. Esse script identifica o centro do grão e encontra seu ângulo de orientação utilizando os momentos da imagem, como pode ser visualizado na seção de código abaixo:

```
# Calcula os momentos da imagem
M = cv2.moments(binary)
# Calcula centro
cX = int(M["m10"] / M["m00"])
cY = int(M["m01"] / M["m00"])
# Calcula angulo de orientacao
angle = 0.5 * np.arctan2(2*M["mu11"],(M["mu20"]-M["mu02"]))
```

Em seguida, o script desenha na imagem o centro e o ângulo encontrado, como pode ser observado na imagem abaixo, correspondente ao arquivo 'Jasmine2.jpg'.

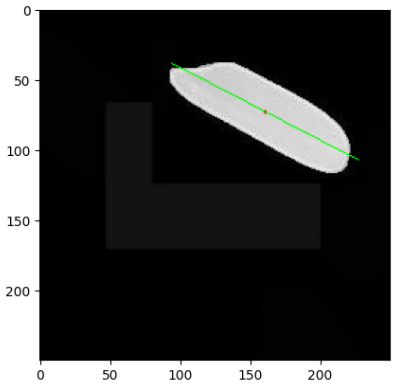

Figura 2.1: Centro e ângulo para Jasmine1

Em seguida, utilizando o método warpAffine() da biblioteca openCV, como pode ser visualizado no código a seguir, o script centraliza e alinha horizontalmente a imagem, cujo resultado pode ser observado na imagem em sequência.

```
# Matriz de rotacao
M_rotate = cv2.getRotationMatrix2D((cX, cY), np.degrees(angle), 1)
M_rotate = np.vstack([M_rotate, [0, 0, 1]])  # extend to a 3x3 matrix
# Matriz de translacao
dx = cols//2 - cX
dy = rows//2 - cY
M_translate = np.float32([[1, 0, dx], [0, 1, dy], [0, 0, 1]])  # 3x3 matrix
# Combina rotacao com translacao
M_affine = np.matmul(M_translate, M_rotate)
# Slice da matriz para inserir no warpAffine
M_affine = M_affine[:2, :]
# Aplica a rotacao e translacao
img_transformed = cv2.warpAffine(img_clean, M_affine, (cols, rows))
```

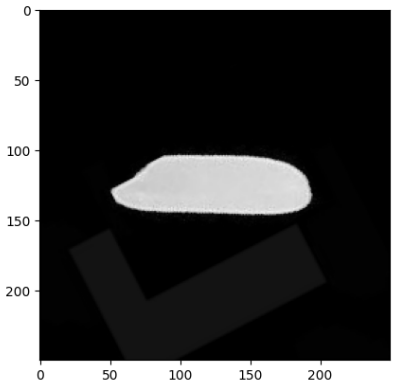

Figura 2.2: Grão centralizado e alinhado, Jasmine2.png

2.2 Classificação dos grãos

Para classificar os grãos de arroz, foi utilizada uma abordagem de *5-fold cross validation*, ou seja, o conjunto de dados foi separado em cinco subconjuntos e o treinamento foi aplicado para quatro deles, enquanto o teste foi realizado no restante. Conforme especificado no enunciado, cada subconjunto é composto por dez imagens ordenadas sequencialmente, de forma que o primeiro subconjunto é formado pelas imagens 1 a 10 de cada tipo de grão e o último é formado pelas imagens 41 a 50 de cada classe. A estrutura de modelo construída seguiu o modelo Lenet-5, e seu código pode ser visualizado abaixo:

```
1   model = Sequential()
2   # Camada 1: Convolucao + Average Pooling
3   model.add(Conv2D(filters=6, kernel_size=(5, 5), activation='tanh',
    input_shape=(32, 32, 3)))
4   model.add(AveragePooling2D())
5   # Camada 2: Convolucao + Average Pooling
6   model.add(Conv2D(filters=16, kernel_size=(5, 5), activation='tanh'))
7   model.add(AveragePooling2D())
8   # Transformando em linha de valores
9   model.add(Flatten())
10  # Camada 3: Camada densa
11  model.add(Dense(units=120, activation='tanh'))
12  # Camada 4: Camada densa
13  model.add(Dense(units=84, activation='tanh'))
14  # Camada 5: Cama de saida
15  model.add(Dense(units=5, activation = 'softmax'))
```

Com isso, foram realizados cinco treinamentos distintos, conforme o método especificado, tanto para os dados alinhados quanto para não alinhados. Em seguida, os resultados foram compilados e foi possível determinar se houve melhoria significativa do resultado após alinhá-los

3. Resultados e Discussão

Para os dados não alinhados, as acurácias obtidas podem ser visualizadas na tabela 3.1. Com isso, a acurácia média vale $93,6\%$ e o seu desvio padrão vale $3,44\%$.

Fold	Acurácia
1	98%
2	96%
3	92%
4	94%
5	88%

Tabela 3.1: Acurácias para dados não alinhados

De maneira análoga, os resultados obtidos para os dados alinhados podem ser observados na tabela 3.2. Com isso, a acurácia média é de $94,8\%$ e o desvio padrão igual a $2,04\%$

Fold	Acurácia
1	98%
2	92%
3	94%
4	96%
5	94%

Tabela 3.2: Acurácias para dados alinhados

Portanto, pode-se concluir que apenas alinhar as imagens não apresentou resultados melhores na classificação, visto que a média obtida para os dados não alinhados está contida em $0,948 \pm 0,0204$, intervalo consistido pela média somada ou subtraída do desvio padrão dos dados alinhados.