Descritor de textura
Local Binary Pattern

Carlos Eduardo Fontaneli, RA - 769949

23 de setembro de 2022

Resumo

O presente trabalho tem como objetivo implementar e analisar a técnica de descritor de textura Local Binary Pattern. Ademais, buscou-se utilizar os resultados obtidos com o método para classificar imagens através do algoritmo K-Nearest Neighbours.

Palavras-chaves: descrito, textura, binário, python.

Introdução

Implementou-se a técnica de descritor de textura Local Binary Pattern, para aplicação do resultados obtidos, em forma de histograma, em algoritmos de classificação de imagem, no caso o K-Nearest Neighbours.

O Local Binary Pattern é um descritor visual muito utilizado para classificação em visão computacional. O método se baseia em traçar um limiar na vizinhança de uma dado pixel e considera esse limiar como um número binário, que futuramente é convertido para decimal e caracteriza um atributo daquela imagem.

1 Objetivos

1.1 Implementação do Local Binary Pattern

Este trabalho buscou-se realizar a implementação manual do método de local binary pattern, considerando que o método recebe uma imagem em tom de cinza e retorna dela um histograma resultante da aplicação do método.

1.2 Classificação de imagens com os resultados obtidos

Após a implementação e aplicação do método buscou-se classificar imagens através da análise dos histograma obtidos para cada imagem analisada. Após isso buscou-se analisar os resultados obtidos na classificação através do cálculo de métricas e análise visual.

2 Metodologia

2.1 Imagens utilizadas

Para aplicação do método e futura classificação baseada na extração da descrição de textura foram utilizadas imagens binárias de números. Tais imagens se concentram em apenas uma imagem só que foi dividida gerando um vetor de imagens, onde cada imagem contêm apenas um número. Posteriormente tal vetor foi dividido em conjuntos de treino e teste que foram utilizados para classificação. Abaixo segue exemplos de 2 números de cada classe de algarismo

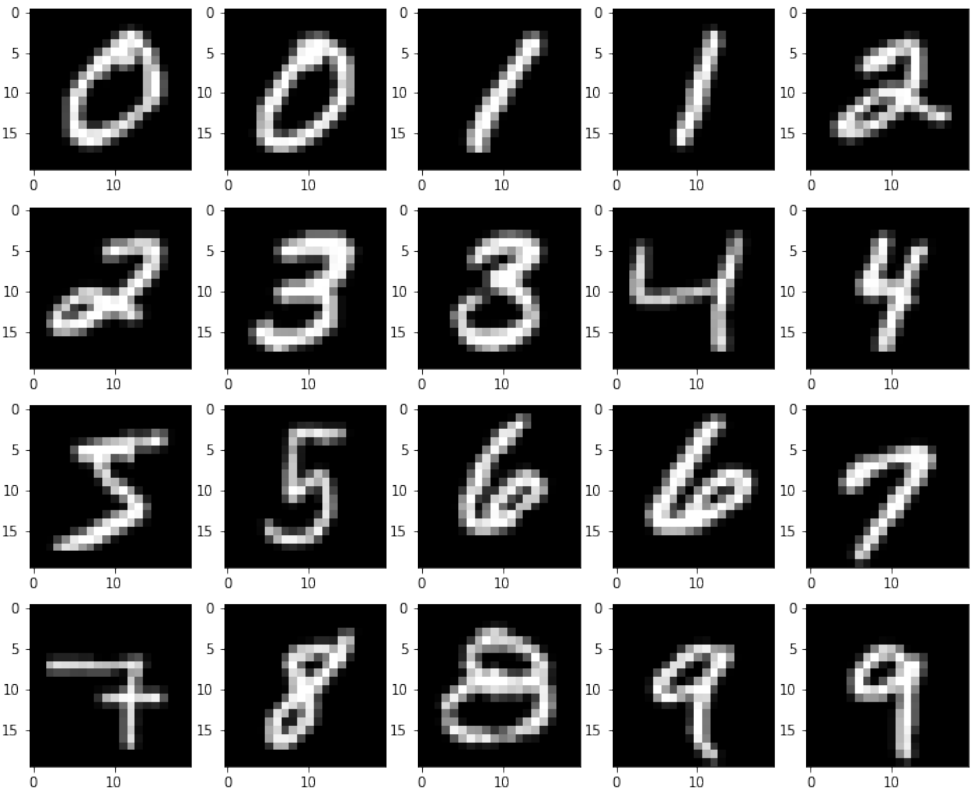

Figura 1 – Exemplo de números.

2.2 Implementação do método

O método se baseia no traçado de um limiar para cada pixel dado a análise de sua vizinhança. Dessa forma, dado uma imagem o método avalia a vizinhança de cada pixel que não seja parte da borda da imagem. Para cada pixel vizinho que é maior que o pixel central atribui-se o valor 0 aquele pixel e 1 caso seja menor. Assim gera-se um vetor final com 8 elemento de 0 e 1.

Depois de calcular tal vetor, considera-se ele como uma sequência de 8 bits que formam um número binário. Esse número é posteriormente convertido para um número inteiro no intervalo [0,255]. Dessa forma, aplica-se esses processos para pixel da imagem e salva-se o valor inteiro em um novo vetor do tamanho da imagem original decrescido de uma coluna e uma linha(dado a não utilização dos pixels da borda).

Com o novo vetor obtido, calcula-se o histograma dele utilizando 256 caixas. Onde cada caixa é um propriedade(atributo) associado a imagem em questão. Com isso se obtém material o suficiente para realizar a classificaçãod e uma imagem.

Abaixo segue o código em Python resultante de tal processo:

```
def lbp_histogram(img):
  num_rows, num_cols = img.shape
  nei_list = [(-1, 0), (-1, 1), (0, 1), (1, 1),
              (1, 0), (1, -1), (0, -1), (-1,-1)]

  img_lbp = np.zeros_like(img)
  for row in range(1, num_rows-1):
      for col in range(1, num_cols-1):

        # separando os vizinhos
          neighbours = []
          non_null_neighbours = 0
          for nei_index in nei_list:
            # contagem de vizinhos n o nulos
            if img[row + nei_index[0]][col + nei_index[1]] >= img[row][col]:
              neighbours.append(1)
            else:
              neighbours.append(0)

         #Convertendo o vetor bin rio para decimal
          zeros_ones = np.where(neighbours)[0]
          if len(zeros_ones) >= 1:
              num = np.sum(2**zeros_ones)
          else:
              num = 0
          img_lbp[row][col] = num

  # retorna o histograma da imagem resultante do m todo
  hist = np.histogram(img_lbp[1:-1, 1:-1].flatten(), bins=256)[0]
  return  hist
```

2.3 Classificador e métricas de avaliação

Como classificador utilizou-se o K-Nearest Neighbours com k=3. Portanto, a classificação foi feita baseada no cálculo da distância dos 3 vizinhos mais próximos do elemento sendo classificado. Para a aplicação do modelo utilizou-se a implementação da biblioteca sklearn.

Abaixo segue o código da aplicação do modelo:

```
# K-Nearest Neighbours
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(train_data, labels)
knn_result = knn.predict(test_data)
```

Após a execução do modelo extraiu-se diversas métricas de avaliação do resultado obtido através do método classification_report da biblioteca sklearn. Na sequência calculou-se a matriz de confusão e realizou-se sua plotagem.

Abaixo segue o código de tal processo:

```
# Calculando as m tricas da classificao realizada
metrics = classification_report(knn_result, labels, output_dict= False)

# Plotando a matriz de confus o resultante da classificacao
c_matrix = confusion_matrix(knn_result, labels)
class_names = range(10)
fig, ax = plot_confusion_matrix(conf_mat=c_matrix,
                                show_absolute=True,
                                colorbar=True,
                                class_names=class_names,
                                figsize=(10, 10));

plt.title('Confusion Matrix - KNN')
_ = plt.savefig('confusion_matrix.png');

plt.show();
```

3 Resultados

Após a aplicação do método e da classificação realizada com o KNN obteve-se as seguintes métricas:

	precision	recall	f1-score	support
0	0.82	0.55	0.66	376
1	0.95	0.95	0.95	251
2	0.47	0.35	0.40	333
3	0.67	0.57	0.61	294
4	0.65	0.69	0.67	234
5	0.52	0.57	0.54	228
6	0.44	0.57	0.50	193
7	0.58	0.81	0.68	180
8	0.54	0.71	0.62	191
9	0.58	0.66	0.62	220
accuracy			0.62	2500
macro avg	0.62	0.64	0.62	2500
weighted avg	0.64	0.62	0.62	2500

Figura 2 – Métricas da classificação.

E a seguinte matriz de confusão:

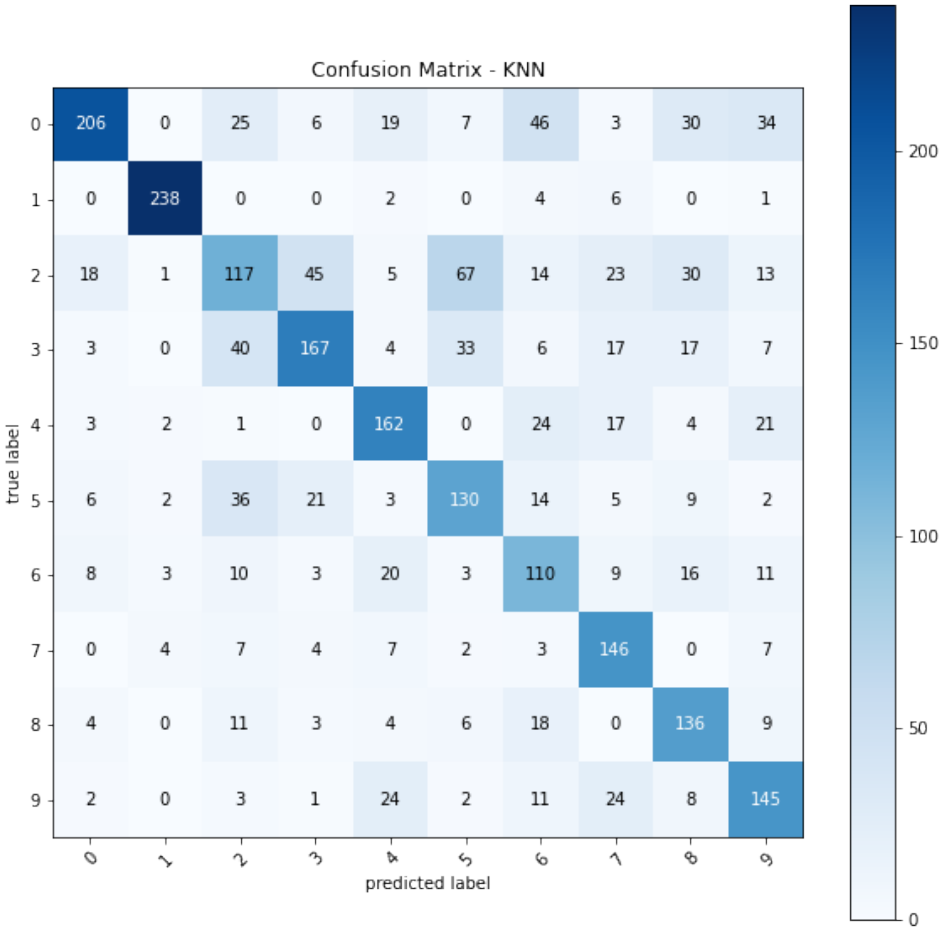

Figura 3 – Matriz de confusão da classificação.

Considerações finais

Através da análise dos resultados obtidos nas métricas e na análise visual da matriz de confusão, pode-se observar que a classificação não foi a mais precisa, ficando com cerca de 62% de acurácia. Apesar de nãos ser um valor baixo, está longe de uma classificação precisa.

Ademais, é possível perceber, por meio da análise da matriz de confusão, que o KNN errou muita classificações como falsos positivos, que estão representados na porção superior da matriz.

Vale a ressalva que para alguns números o método se mostrou mais preciso. Para

classificar 0 e 1 o método atingiu acurácias superiores a 80%, sendo que para números 1 o método atingiu altissímos 95% de acurácia. Contudo, houve números em que as classificações foram bastante insatisfatória, como por exemplo, o número 6 que obteve% uma acurácia de apenas 44% nas classificações de suas imagens.

Por fim vale ressaltar que, apesar dos resultados obtidos, não se pode afirmar com certeza que o resultado da classificação está totalmente atrelado ao método de Local Binary Pattern, as imagens utilizadas e o classificador também possuem grande contribuição no resultado obtido.