UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA

INTRODUÇÃO À TEORIA DA INFORMAÇÃO

Reconhecedor de Padrões Baseado em LZW

Giovanni Bruno Travassos de Carvalho - 11506849
Jordan Elias Rodrigues - 11516379
Kaio Moura dos Santos - 11506860

Professor: Derzu Omaia

1. Introdução

Este trabalho tem como objetivo implementar um reconhecedor de padrões baseado no algoritmo LZW de compressão com dicionário de tamanho variável utilizando uma base de dados contendo imagens de faces humanas, ORL Database of Faces. Essa base contém imagens da face de 40 pessoas, cada pessoa possuindo 10 imagens da face. Destas 10 imagens, 9 foram separadas para construção e treinamento de um dicionário utilizando o algoritmo LZW. O processo de teste utiliza uma imagem de cada pessoa e passa pela compressão de todos os dicionários gerados pelas 40 pessoas.

2. Desenvolvimento

A implementação deste projeto foi feita utilizando a linguagem Python e a IDE *Replit*, que nos permitiu desenvolver em conjunto pelo navegador. Os testes foram rodados na IDE *Spyder Anaconda*, pois o *Replit* não comportava arquivos tão grandes como os utilizados e nem o processamento do algoritmo para a base.

As imagens da base de dados devem estar em escala de cinza para melhor desempenho, logo as imagens estão em formato PGM. Um arquivo desse tipo traz as imagens em tons de cinza salvo no formato mapa cinza portátil (PGM) e codificado com um ou dois bytes (8 ou 16 bits) por pixel. Ele contém informações de cabeçalho e uma grade de números que representam diferentes tons de cinza, de preto (0) a branco (até 65.536). Os arquivos PGM são normalmente armazenados em formato de texto ASCII, mas também possuem uma representação binária.

O programa primeiramente escolhe de maneira aleatória uma imagem de cada pessoa para teste e armazena seu índice em uma variável. Essa variável escolhida para teste é retirada dos dados. Os dados restantes serão de treinamento. Como o algoritmo foi testado para 40 pessoas com 10 imagens de cada, foram retiradas 40 imagens para teste e sobraram 360 para treino. Na leitura dessa imagem, ela passa por uma função '*dataToString()*' que transforma a imagem em string através da função '*getpixel()*' que transforma cada pixel em uma imagem de banda única. Não foi feito o descarte do cabeçalho. Para execução do algoritmo é preciso mudar o caminho para encontrar os dados na função '*readFile()*' do *classes.py*.

Com cada imagem sendo agora uma string, é gerado um dicionário por pessoa para cada K, aplicando o algoritmo do LZW. O Dicionário se mantém estático durante a classificação. A classificação se dá com base na quantidade de índices após a aplicação do LZW. A maior compressão corresponde a maior semelhança entre a imagem teste e as outras da base de dados. A menor quantidade de índices significa a melhor compressão.

Para cada imagem escolhida como teste é feita a compressão utilizando os dicionários criados com as imagens de treino. A saída são listas de índices codificados, a lista com o menor tamanho é a que possui a menor quantidade de índices, ou seja, a melhor compressão, portanto é como será classificada. Por fim é feita uma comparação entre o índice da pessoa que representa a imagem e o índice da pessoa que o classificador atribui.

3. Resultados

Os testes foram feitos com diferentes tamanhos de contexto *K bits*, de 9 a 16, esse parâmetro define o tamanho máximo do dicionário com LZW, onde um contexto $K = 9$ tem dicionário de tamanho $2^9 = 512$.

O tempo de processamento aumentou conforme o crescimento do tamanho de K. A taxa de acertos da classificação também seguiu o mesmo formato, com exceção do K =10, onde teve um decaimento em relação ao K anterior.

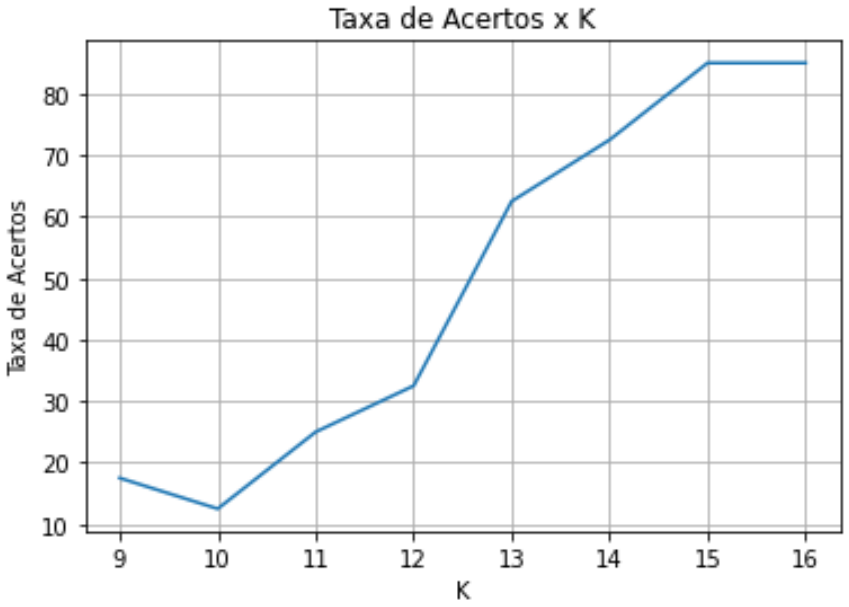

Figura 1: Gráfico de Taxa de Acerto

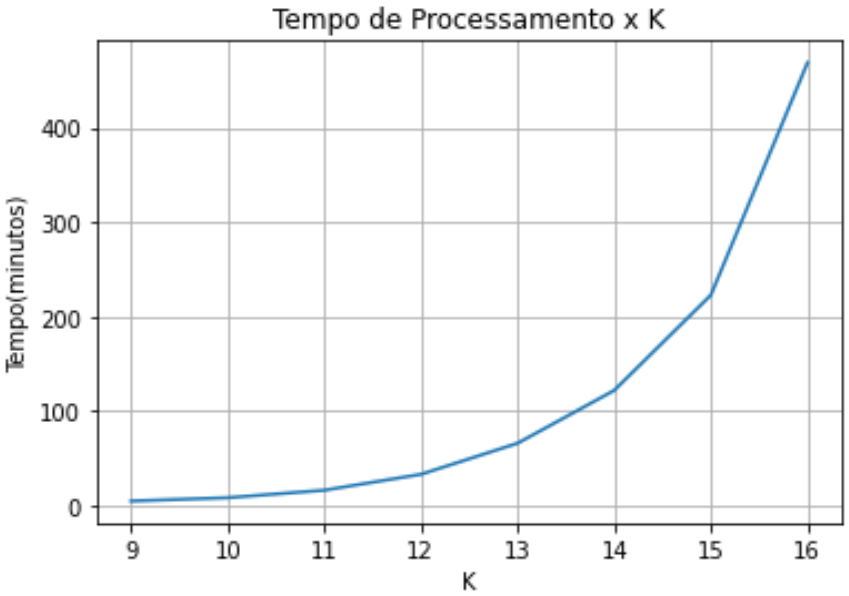

Figura 2: Gráfico de Tempo Processamento

4. Conclusão

O algoritmo desenvolvido apresentou-se com tempo de execução muito elevado para essa base de dados de 360 imagens de treinamento levando, em seu contexto de maior tamanho de dicionário, quase oito horas de processamento. Porém, foi também nesse contexto onde se obteve a maior taxa de acertos na classificação, que com uma acurácia de 85% se mostrou muito bom. Uma possível melhoria futura seria paralelizar algumas funções, dividindo o trabalho do processador para otimizar o desempenho.

Referências

1 - FILEMEMO.INFO. **.pgm Extensão de arquivo**. Disponível em: https://filememo.info/extension/pgm. Acesso em: 20 Jun. 2021.

2 - WIKIPEDIA. **Reconhecimento de padrões**. Disponível em: https://pt.wikipedia.org/wiki/Reconhecimento_de_padr%C3%B5es/. Acesso em: 27 Jun. 2021.

3 - DROPBOX. **orl_faces**. Disponível em: https://www.dropbox.com/s/mnhfhb1i51loknk/orl_faces.zip?dl=0. Acesso em: 20 Jun. 2021.

4 - GETHOWSTUFF. **Python PIL getpixel() method to get pixel value.** Disponível em: https://gethowstuff.com/python-pil-getpixel-method-pixel-value/ .Acesso em: 20 Jun. 2021.