UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
INTRODUÇÃO À TEORIA DA INFORMAÇÃO

Projeto Final – Reconhecedor de padrões utilizando LZW

ANTONIO JONAS GONÇALVES - 2016021023

JORDY ALLYSON DE SOUSA - 11426758

THIAGO ALVES DE ARAUJO - 2016019787

1 – INTRODUÇÃO

O projeto consistiu em desenvolver um reconhecedor de padrões baseado em LZW. Para isto, utilizamos o algoritmo LZW para gerar um modelo de dicionário para as várias categorias do base de dados. Para os testes foi utilizado a base de dados *ORL Database of faces.*

2 – DESENVOLVIMENTO

O projeto foi desenvolvimento utilizado a linguagem de programação python. Para a seleção das amostras de treino e classificação, foi utilizado a técnica de validação cruzada de forma aleatória. Por fim, para a etapa de classificação, foi utilizado o algoritmo K-NN (k-nearest neighbors) com o parâmetro k = 1. Abaixo podemos ver alguns trechos do código.

```python
def LZWCompression(self, image, k, dicionario):
    index = []
    table_size = len(dicionario)
    MAX = 2**k


    firstRound = True
    for pixel in image:
        if(firstRound):
            byte = bytes([ord(chr(pixel))])
            s = b''

        index = self.getKeysByValue(dicionario,s+byte)

        if index != -1:
            s += byte
        else:
            index.append(self.getKeysByValue(dicionario,s))
            if table_size < MAX and self.state == 0:
                dicionario[table_size] = s + byte
                table_size += 1
            s = byte

        byte = bytes([ord(chr(pixel))])

        firstRound = False

    return len(index)
```

Figura 1 Compressor LZW

```python
def main():

    tempo = []
    acc = []

    for i in range(9,17):
        start_time = time.time()

        #Inicia o classificador
        KNN = KNNClassificator()

        Imgs = AbrirImg()

        train = copy.deepcopy(Imgs)

        test, labels = KNN.crossValidation(train)

        KNN.Fit(train, i)

        n = KNN.predicao(test,labels,i)

        tempo.append(time.time() - start_time)
        acc.append(n)

    script(tempo,  acc) ##Escreve os resultados em um arquivo de saída
```

Figura 2 Main

3 – RESULTADOS

Abaixo podemos ver os gráficos com os resultados obtidos após realizarmos os testes variando o tamanho do dicionário de k = 9 até k = 16. É valido ressaltar que todos os testes foram realizados em uma mesma máquina para assegurarmos que o tempo de execução não seria comprometido. O primeiro gráfico representa a taxa de acerto x valor k.

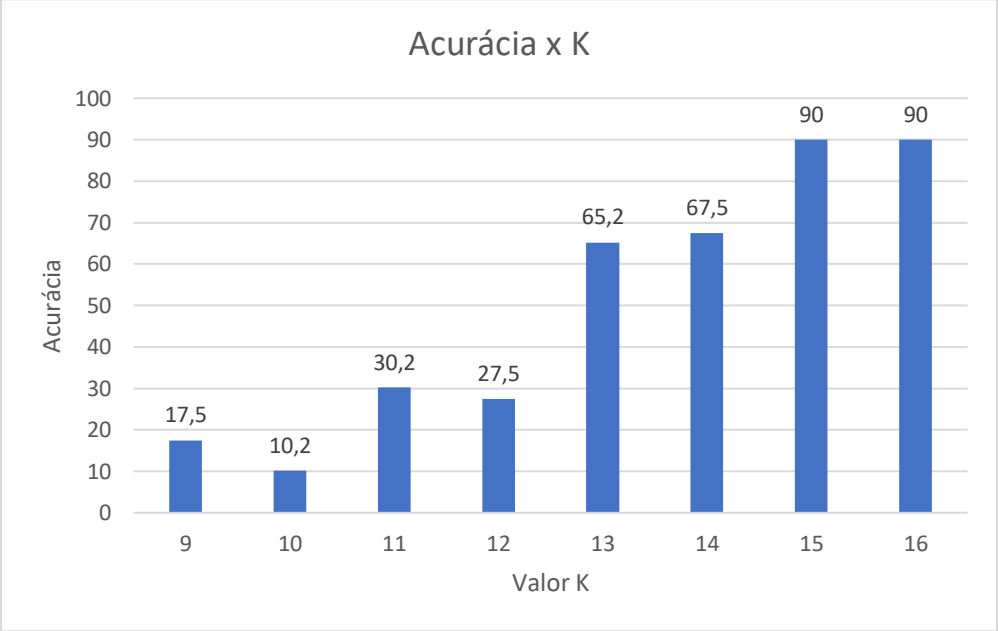

Figura 3 Acurácia em porcentagem x K

O segundo gráfico apresenta o tempo necessário para execução do algoritmo em função do valor k

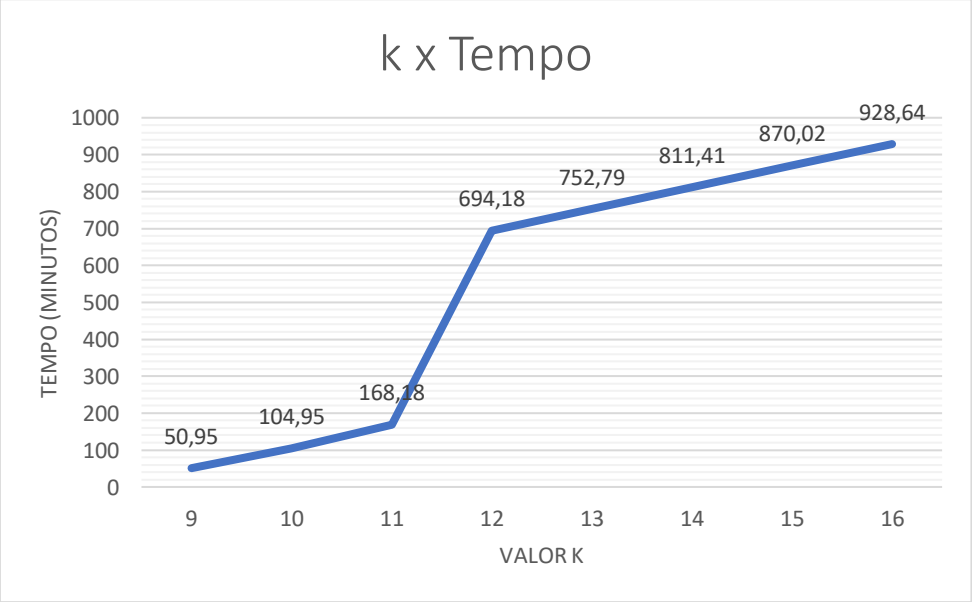

Figura 4 Tempo em minutos x K

Para uma melhor visualização do tempo, abaixo podemos ver um gráfico com a escala de tempo em horas.

Figura 5 Tempo em horas x k

4 – Conclusão

Como podemos observar, os resultados para k = 15 e k = 16 apresentaram uma acurácia igual, porém com uma diferença de tempo consideravelmente grande. Também é valido ressaltar que o aumento da acurácia não cresce de forma linear conforme aumentamos o valor de k (como é possível observar para k = 10 e k = 12).