Duarte Dias, Rita Fonseca, Miguel Rabuge
Paulo de Carvalho
Teoria da Informação
19 de Outubro de 2019
Universidade de Coimbra

Relatório PL1

Entropia, Redundância e Informação Mútua

3. Referente ao Exercício, obtivemos os seguintes resultados:

Tabela 1:

Fonte	Entropia
MRIbin.bmp	0.66
MRI.bmp	6.86
Landscape.bmp	7.61
soundMono.wav	4.07
Lyrics.txt	4.41

3.1 **MRIbin.bmp**

No que toca às imagens podemos observar que o MRIbin é o que, de longe, apresenta menor entropia. Isto deve-se ao facto de ser codificado utilizando apenas 2 cores, preto (0) e branco (255). É esperado um valor de ordem de grandeza consideravelmente superior para a componente preta (0) em oposição à branca (255). É interessante reparar que apesar de existirem 2 valores distintos, a entropia é menor que o valor da menor unidade de divisão, 1 bit.Podemos verificar os resultados,experimentalmente:

Figura 1 - MRIbin.bmp

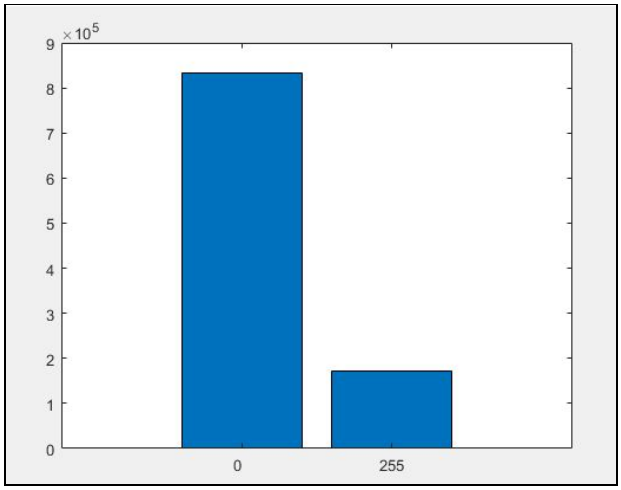


Figura 2 - Histograma do MRIbin.bmp

3.2 **MRI.bmp**

O MRI (normal) apresenta uma entropia bastante mais elevada, ainda que seja a mesma figura. Isto acontece porque a codificação utilizada incorpora 256 tons de cinza.

São esperados 3 picos, dois dos quais, mais relevantes, em torno da componente mais escura e da componente cinzenta (escura), devido ao facto de serem as duas cores mais predominantes na imagem. O último pico, menos relevante, é relativo à componente branca que, apesar de reduzida, tem a sua importância na imagem.

O mesmo verifica-se experimentalmente:

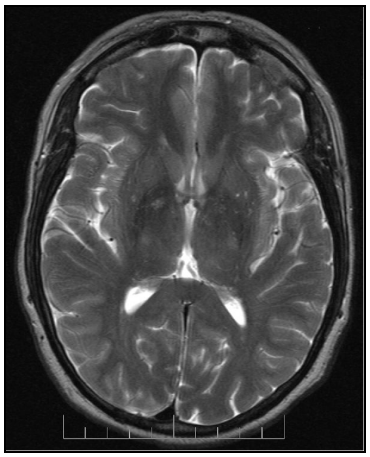


Figura 3 - MRI.bmp

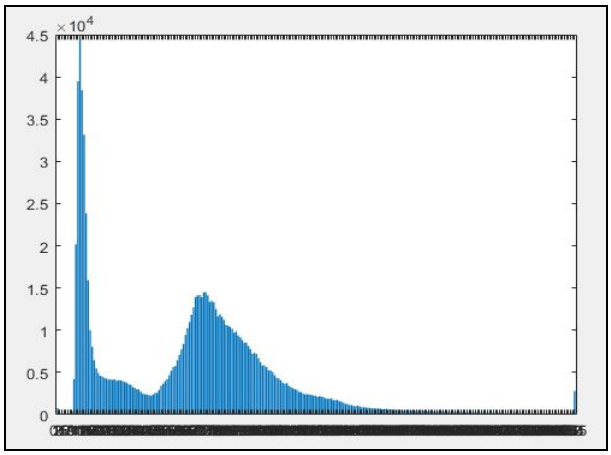


Figura 4 - Histograma do MRI.bmp

3.3 **Landscape.bmp**

Por fim, no que toca as imagens, temos o Landscape.bmp, que, apesar de ser também codificado utilizando os mesmos 256 tons de cinza que a imagem anterior apresenta uma maior dispersão dos seus tons,o que aumenta a sua entropia. É esperado, portanto, uma dispersão relativamente uniforme nos 256 tons com duas fortes afluências na componente escura e na componente clara, devido ao facto de observarmos, empiricamente, na imagem, estes dois constituintes, o que se traduz em dois possíveis pico, o que é verificado experimentalmente.

Figura 5 - Landscape.bmp

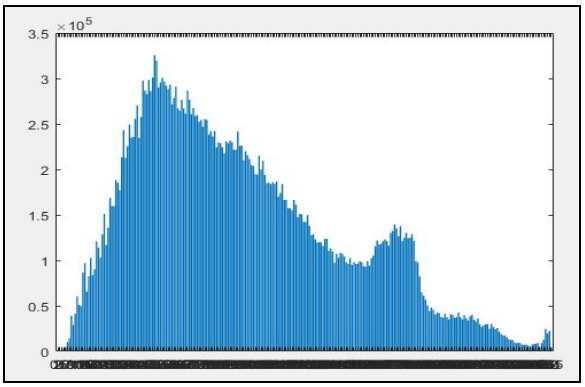


Figura 6 - Histograma do Landscape.bmp

3.4 **SoundMono.wav**

No caso do ficheiro SoundMono.wav temos uma fonte de dados proveniente de um ficheiro de áudio que possui uma entropia 4.07.

É esperado que exista uma concentração de valores no ponto zero, uma vez que um ficheiro de áudio oscila fortemente em torno do mesmo, como podemos constatar na figura 7.

Apesar de ser utilizada uma fonte com 8 bits de quantização, a alta concentração de valores em torno do valor 0 reduz significantemente o valor necessário para o codificar.

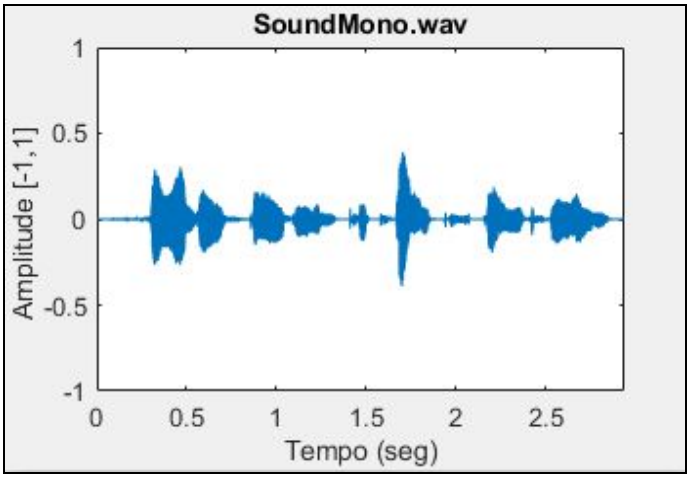


Figura 7 - Visualização gráfica

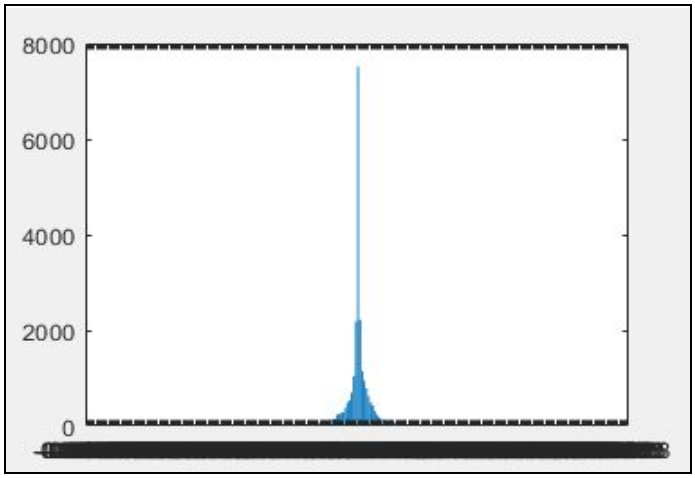
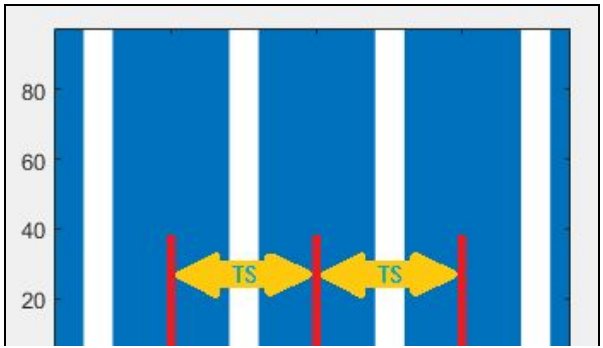


Figura 8 - Histograma do SoundMono.wav

Para determinar o número de bits de quantização foram utilizados os comandos:

```
info = audioinfo('soundMono.wav');
Info.bitspersample
```



Posteriormente, este número foi utilizado para calcular o tamanho dos intervalos no histograma, tal que:

```
Ts = 2/power(2,info.bitspersample);
alphabet = -1:Ts:1-Ts;
```

sendo Ts o tamanho do intervalo.

Figura 9 - Um olhar mais próximo

3.5 **Lyrics.txt**

No caso de Lyrics.txt podemos observar que apresenta uma entropia de 4.41 bits/símbolo.

Estudo sobre a entropia da lingua inglesa:

Zero-order model:	4.76 (=log 27) bits/letter
1st-order model:	4.03 bits/letter
2nd-order model:	3.32 bits/letter
3rd-order model:	3.01 bits/letter
Fourth-order model:	2.8 bits/letter
1st-order word model:	2.14 bits/letter

In Slides da Cadeira Cap. 2 Slide 45

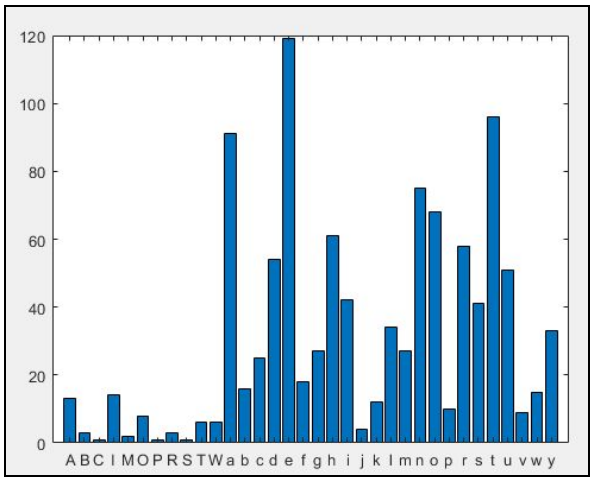


Figura 10 - Histograma de Lyrics.txt

Podemos observar que o valor da entropia é bastante semelhante ao valor da entropia de ordem 0, porque é, na verdade, o que estamos a fazer: um agrupamento de ordem 0, elemento a elemento. Portanto o resultado esperado seria algo semelhante - o que se verificou.

Em resposta à pergunta:

> 🕮 Será possível comprimir cada uma das fontes de forma não destrutiva? Se Sim, qual a compressão máxima que se consegue alcançar? Justifique.

→ Sim, qualquer fonte pode ser comprimida de forma não destrutiva, basta analisar a frequência e atribuir um código de prefixo adequado. Quanto à compressão máxima que se pode alcançar, podemos recorrer ao **Teorema de Shannon**.

Segundo o **Teorema de Shannon**, para qualquer alfabeto A de dada probabilidade P, o número de bits esperado é minorado por H(P). Mas em que medida e este resultado importante? Este resultado denota que qualquer código encontrado terá pior ou igual eficiência que H(P). É necessário, também, dizer que este resultado é apenas teórico, dado que nem todos os códigos atuais tem pior eficiência que H(P). Isto abre assim, caminho à procura do melhor código de compressão.

4. Referente ao exercício 4, os resultados foram:

Tendo como dado que o comprimento médio dos códigos de Huffman são minorados pela entropia da fonte (Source) e majorados pela entropia da fonte + 1, ou seja:

$$H(S) \leq \bar{l} < H(S)+1$$

In Slides da Cadeira Cap. 2 Slide 110
Esperamos que o comprimento médio seja superior à entropia calculada na Tabela 1 e menor que a mesma mais 1 bit - o que se verifica.

Tabela 2:

Fonte	H(X)	Comprimento Medio	H(X) +1
MRIbin.bmp	0.66	1	1.66
MRI.bmp	6.86	6.8909959	7.86
Landscape.bmp	7.61	7.6293	8.61
soundMono.wav	4.07	4.1107	5.07
Lyrics.txt	4.41	4.4435	5.41

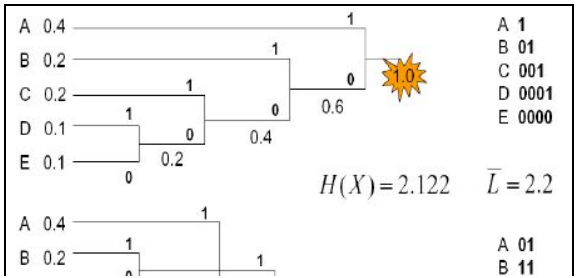
→ A variância entre os comprimentos máximo e mínimo foi, em todos os casos, de 1. Tal resultado indica que o agrupamento de símbolos, 2 a 2, foi o mais efetivo possível, ou seja, a variância foi mínima, mantendo a entropia mínima.

Em resposta à pergunta:

> 🕮 Será possível reduzir-se a variância? Se sim, como pode ser feito em que circunstância será útil?

Sim, dependendo da árvore gerada (supondo que a árvore não está num estado de menor variância possível).

Poderíamos obter tais resultados ao garantirmos que quando aglutinamos ramos da árvore, o fazemos maximizando a escolha de ramos com menor número de elementos, ou seja, "*colocando os símbolos na lista usando a ordem mais elevada possível*" (in Slides da Cadeira Cap.2 Slide 104).



Uma redução na variância poderá ser bastante útil se necessitarmos de transmitir os dados Over-the-Wire.
Uma menor variância representará uma menor Bandwidth (Capacidade de canal) máxima necessária, o que representará um uso mais eficiente da mesma.

Exemplo ilustrativo à direita.

In Slides da Cadeira Cap.2 Slide 103

5. Referente ao exercício 5, os resultados foram:

Tabela 3:

Fonte	$\frac{H(X^2)}{2}$	Comprimento Medio	$\frac{H(X^2)}{2} + \frac{1}{2}$
MRIbin.bmp	0.40	0.60	0.90
MRI.bmp	5.19	5.21	5.69
Landscape.bmp	6.21	6.22	6.71
soundMono.wav	3.31	3.32	3.81
Lyrics.txt	3.65	3.66	4.15

Analisando os dados, nota-se que de facto que o comprimento médio é minorado por $\frac{H(X^2)}{2}$ e majorado por $\frac{H(X^2)}{2} + \frac{1}{2}$ tal que:

$$\frac{H(S^n)}{n} \leq \bar{l} < \frac{H(S^n)}{n} + \frac{1}{n}$$

In Slides da cadeira Cap.2 Slide 110

Porém, uma vez que H(S^n) ≠ nH(S), **não** se verificam as seguintes desigualdades:

$$H(S) \leq \bar{l} < H(S) + \frac{1}{n}$$

In slides da cadeira Cap.2 Slide 110

Para a proposição acima ser verdadeira, as fontes teriam de ser independentes e a este facto equivale H(S^n) = nH(S). Mas como H(S^n) ≠ nH(S), então concluímos que as fontes não são independentes, ou, por outras palavras, **existe uma dependência estatística entre as fontes**.

Notas:

1) Para fazer os agrupamentos dos símbolos 2 a 2 foram usadas diferentes técnicas para codificar os símbolos compostos:

- **Nas imagens e texto:**

  Tendo em conta que a imagens selecionadas contêm valores entre 0 e 255, estas podem ser codificadas com 8 bits. Tomando proveito dessa propriedade, vamos fazer um shift left de 8 bits ao primeiro byte (colunas ímpares) e adicionar o segundo byte à ao mesmo (colunas par). Desta forma evitamos desperdiçar bits, garantindo a unicidade de cada grupo de elementos, uma vez que a informação das colunas ímpares fica no byte mais significativo e a das colunas pares no byte menos significativo.

  Importante também será dizer que o protocolo utilizado para para matrizes com o número de colunas ímpares será a remoção da última coluna.

  Denotar que no texto, cuja codificação se assemelha à acima referida (dado que se encontra entre 65 e 122), foi aplicada uma prévia transformação utilizando a função parsetext, que remove todos os caracteres desnecessários, deixando apenas uma string com todos os caracteres seguidos.

- **No áudio:**

  Tendo em conta que, os valores variam no intervalo [-1,1[com Ts = 0.0078125, somando 1, ficam [0,2[, todos positivos. Agora, se dividirmos cada valor presente naquele intervalo discreto por Ts, vamos obter um intervalo de [0,256[~~ [0,255], podendo este ser codificado com a função twobitsencoded, de forma semelhante à indicada acima, com um shift left de 8 bits nas colunas impares e soma com as pares.

2) Para descodificar tais códigos recorre-se à função getLabel que depende do tipo de label escolhido (dado por um número passado por parâmetro, opcode), retorna uma array com os labels corretos. Os labels são determinados ao reverter a função de codificação, utilizando a função bitand como máscara para eliminar os bis indesejados.

3) Para a codificação foi utilizada a função twobitsencoded, descrita abaixo:

Funcionamento da função *twobitsencoded*:

- **Parâmetros:**

  Recebe uma **fonte** previamente codificada com as técnicas acima descritas.

  Recebe o **expoente** da potência de base 2 que irá multiplicar os elementos das colunas ímpar.

- **Valor de retorno**

  Retorna uma matriz codificada 2 a 2, linha a linha da fonte original.

- **Descrição**

  Recebendo uma fonte, verifica se a esta (matriz) tem um número de colunas ímpar e, caso o tenha, elimina a última coluna. Seguidamente, multiplica todos os

elementos das colunas ímpar (i) por 2^OG e soma o valor das colunas par (i+1) à coluna ímpar imediatamente anterior.

4) Com vista a manter o código mais limpo e organizado foi criada a função getData. A essa função cabem os parâmetros de entrada : source, a matriz original; alpha, um alfabeto (podendo este ser opcional); e a ordem (esta ordem será necessária para calcular entropia, comprimento de huffman e variância de ordem n (representando, por exemplo, ordem 2 agrupamentos de 2 bits). O funcionamento da função é bastante simples, dado o alfabeto de entrada calcula o respetivo histograma, calculando ainda entropia, comprimento médio de codificação (utilizando códigos de Huffman) e a variância. Para o som há uma variação da mesma com o nome getSoundData.

6. Referente ao exercício 6, os resultados foram:

Tabela 4 (Sendo a Query o ficheiro 'query.bmp'):

Alinea	Target	IM Média	Maior valor de IM	Rectangle(x)	Rectangle(y)
b)	**Original**	0.805608	1.350032	421	316
	Noise	0.693957	1.196346	421	316
	Inverted	0.805608	1.350032	421	316
	Lightning Contrast	0.583521	1.224019	421	316
c)	**Target 1**	0.741896	1.700786	121	196
	Target 2	0.796450	1.200681	76	151
	Target 3	0.822688	1.170559	166	91
	Target 4	0.685719	1.112329	256	196

b) Relativamente aos 4 primeiros target's foi, de facto, possível localizar a imagem nas várias distorções do target:

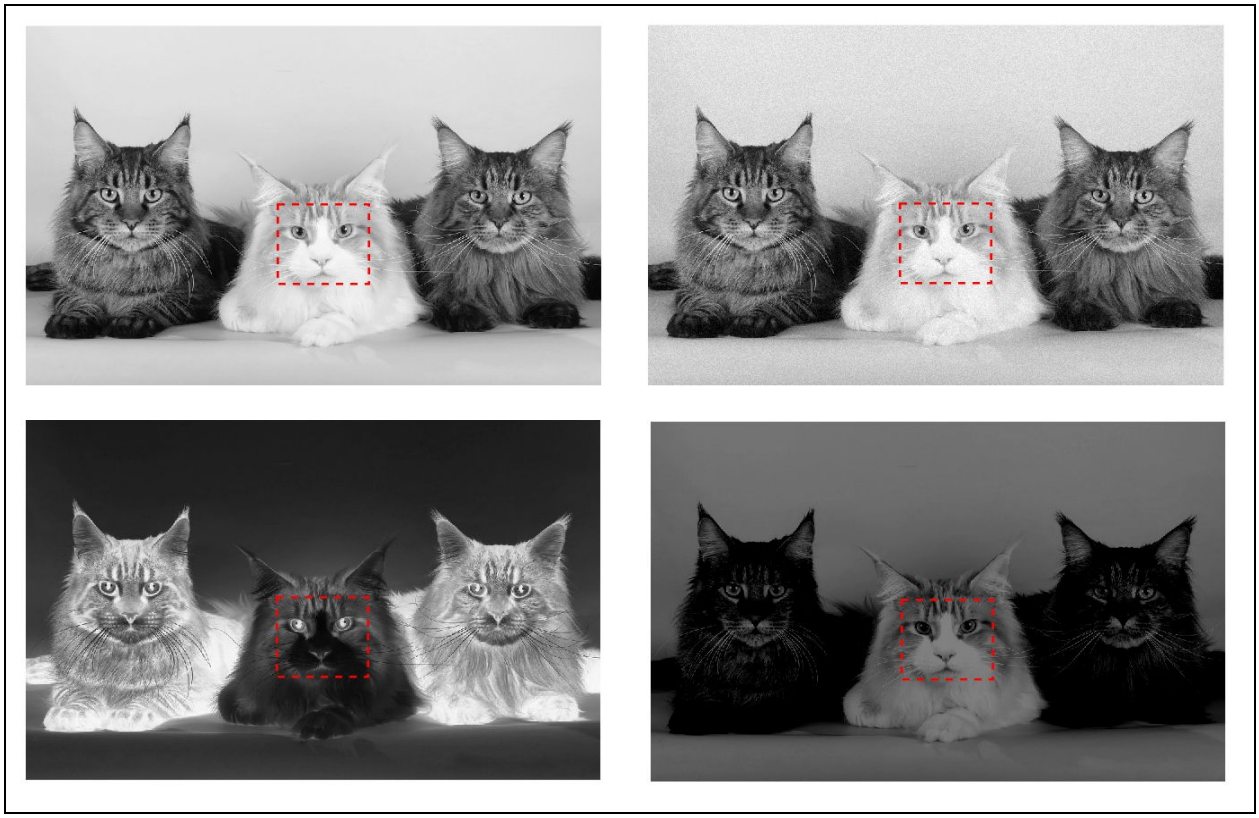


Figura 11 - Target original, noise , inverted e lightning contrast

Relativamente às distorções verifica-se que:

- **Target_inverted:** não altera em nada quer o valor da informação mútua média, quer no valor máximo.

  Tal acontecimento deve-se ao facto de uma imagem invertida, em nada mudar o valor da entropia em relação à imagem original, logo a informação mútua será exatamente igual.

- **Target_noise:** Diminui tanto o valor médio de informação mútua como o valor máximo da mesma.

  Adicionando ruído a uma imagem original, é de esperar que a informação mútua, ou seja, a informação entre ambas as fontes diminua, uma vez que a entropia do target é corrompida aleatoriamente em todas as posições pelo ruído. Logo a query, ao passar sobre cada janela, pode encontrar uma informação próxima da original, se a amplitude do ruído adicionada for baixa ou uma informação menor se for alta, havendo assim uma proporcionalidade entre o aumento do ruído e diminuição da informação mútua.

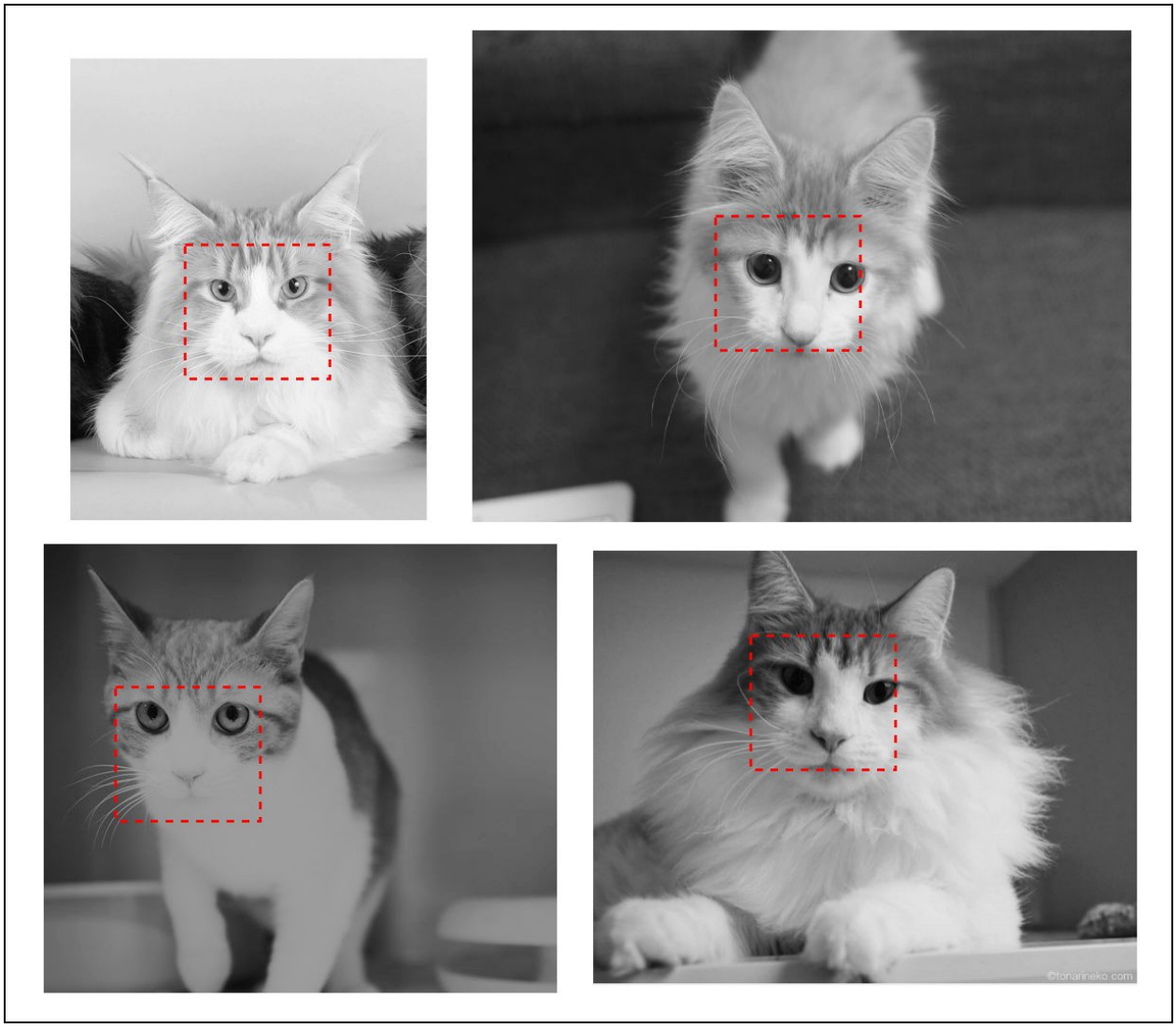
- **Target_lightning_contrast:** Diminui o valor médio de informação mútua, abaixo do noise, porém, apesar de diminuir relativamente ao original e inverted, aumenta em relação ao noise o valor máximo de informação mútua.

  Ao oscilar os valores luminosos, estamos, também, a corromper a nossa informação. Dito isto, é de esperar que o valor médio de informação mútua diminua, e, consoante a corrupção, poderá ser menor ou maior do que o noise. Assim, apesar de se verificar que o noise tem uma maior informação mútua média que este, não implica que todas as janelas deste tenham um valor de informação média também menor. Tal se verifica na janela da cara do gato central em que a corrupção feita pela oscilação da luminosidade foi menor, do que a corrupção feita pelo ruído, e portanto terá uma maior informação mútua que a janela da cara do gato central do noise.

Nota: 1) Para retirar a matriz de informação mútua é utilizada a função “mutual_info” que, resumidamente, utiliza a fórmula I(X;Y) = H(X) + H(Y) - H(X,Y). X é a query, Y é a janela atual, e X,Y é uma matriz codificada com um shift left de 8 bits, semelhante à codificação 2 a 2, mas entre duas fontes distintas.

2) Posteriormente, é chamada uma função “getCoords” que, com base no step e na matriz de informação mútua, retornada pela “mutual_info”, procura o máximo nesta e converte as coordenadas da mesma para coordenadas no target, x e y, que é usada na função rectangle para desenhar o retângulo correspondente à janela de maior informação mútua.

c) Relativamente aos targets*, foi possível localizar as faces dos gatos nas 4 imagens tal que



Tais resultados foram os esperados, uma vez que, sendo os targets* imagens semelhantes à query a nível de tons, é de esperar que a informação mútua máxima corresponda a uma janela ao redor da face do gato.
Estes resultados levam a uma conclusão importante: o algoritmo é, não só, capaz de detectar imagens com ruído (como vimos previamente) mas também é capaz de detectar imagens semelhantes. A natureza dos resultados indica que existem padrões na natureza que se repetem e que estes podem ser medidos através da informação mútua. As aplicações no "mundo real" são imensas, algoritmos mais elaborados e com bases de dados maiores podem reconhecer todo o tipo de padrões. O reconhecimento de padrões está também ligado a outras áreas da Engenharia Informática e não só, como nos caso Inteligência Artificial. Concluímos assim que, num mundo altamente padronizado, o reconhecimento de padrões representa uma base importantíssima para uma multiplicidade de ramos e que na natureza há necessariamente uma dependência estatística em toda a sua envolvente.