universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Lab work n◦3

Teoria Algorítmica da Informação

Trabalho realizado por:

- Isaac dos Anjos Nº Mec. 78191
- Lucas Barros Nº Mec. 83895
- Pedro Cavadas Nº Mec. 85090

Introdução

Hoje em dia, algoritmos de machine learning são os algoritmos mais usados para fazer reconhecimento de imagem.

O objetivo deste trabalho é fazer o reconhecimento de imagem utilizando métodos de compressão de imagem, algo um pouco fora do habitual.

Neste relatório vamos primeiro fazer uma abordagem mais matemática e formulatória ao problema para explicar conceitos importantes e de seguida vamos explicar melhor a abordagem utilizada e o código. Por último vamos apresentar os resultados obtidos e fazer a análise de resultados.

Modelo Matemático

A abordagem utilizada para calcular a similaridade entre duas imagens é a de medir a similaridade encontrada na noção de informação algorítmica, isto é, comparar a complexidade de Kolmogorov.

A ideia principal por trás disto é a de *Normalized Information Distance* (*NID*) que calcula a complexidade de Kolmogorov e obtém uma distância entre duas *strings*. Esse valor de distância é dado pela fórmula:

$$NID(x,\ y) \ = \ \frac{max\{K(x|y),\ K(y|x)\}}{max\{K(x),\ K(y)\}}$$

onde o *K(x)* é a complexidade de Kolmogorov da *string x* e *K(x|y)* a complexidade de Kolmogorov de *x* dado *y*.

No entanto, a complexidade de Kolmogorov não é computável e por isso é necessário utilizar uma aproximação da mesma, mas computável. Para isto recorreu-se à *Normalized Compression Distance* (*NCD*) que é uma aproximação com base na compressão da informação. O cálculo desta aproximação é dado pela fórmula:

$$NCD(x,\ y) \ = \ \frac{C(x,y) - min\{C(x),\ C(y)\}}{max\{C(x),\ C(y)\}}$$

onde *C(x)* é o número de *bits* necessário para um certo compressor *C* representar *x* e *C(x, y)* é o número de *bits* necessário para comprimir *x* e *y* juntos. Uma distância com valor próximo de 1 indica pouca similaridade entre as *strings* (que no nosso caso serão os valores dos pixels das imagens).

A principal vantagem de utilizar esta técnica é que pode-se utilizar compressores de uso geral, como por exemplo **gzip**, **bzip2**, ou **ppmd**, entre outros. No entanto, é preciso que alguns requisitos sejam cumpridos, como por exemplo, *C(x, x)* deverá ser aproximadamente *C(x)*.

Por outro lado, uma melhor aproximação do *NID* requer compressores condicionais, o que nos leva à utilização de *Normalized Conditional Compression Distance*, que é dado pela fórmula:

$$NCCD(x, y) = \frac{max\{C(x|y), C(y|x)\}}{max\{C(x), C(y)\}} .$$

onde *C(x)* é o número de *bits* necessário para um certo compressor *C* representar *x* e *C(x, y)* é o número de *bits* necessário para comprimir *x* e *y* juntos.

Descrição da Solução

A nossa solução para classificar um uma imagem, possui 3 componentes principais:

- A primeira componente possui uma classe NCD.cpp que possui a implementação do método de NCD. O método vai ser explicado mais detalhadamente numa fase posterior.
- A segunda componente possui a classe NCCD.cpp que tal como a NCD, possui a implementação do método de NCCD cuja implementação vai também ser explicada numa fase posterior.
- A terceira componente é o script de teste que vai utilizar tanto o NCD como o NCCD para realizar os testes e escrever os resultados.

Para executar o projeto, é necessário ter a biblioteca de OpenCV instalada com todas as bibliotecas de c++, pois o projeto faz uso deste módulo no reconhecimento de imagem.

Para executar o projeto, recorremos ao auxílio de um ficheiro premake5.lua para ajudar a linkar as bibliotecas aos ficheiros correspondentes. Para executar o projeto basta fazer o comando “./build.sh” na pasta de origem, e de seguida fazer o comando “./run.sh test {directory}” em que directory é o diretório que contém as imagens (normalmente “ImgCondComp/orl_faces/”).

O projeto contém ainda um componente “resize.cpp” que tem como objetivo fazer um pré-processamento das imagens de maneira a reduzir a sua resolução. No entanto tal não foi utilizado no teste final pois concluímos que a Accuracy do algoritmo piorava com este pré-processamento.

src/NCD.cpp

Tal como explicado em cima, este ficheiro possui a classe NCD que calcula a medida de NCD entre 2 imagens. Esta classe possui um atributo opcional passado no construtor que é o nome do compressor, neste trabalho estão disponíveis 2 opções, o gzip que é o compressor usado por default e o lzma.

Possui 3 métodos, GetFileSize, compress e compute.

O método GetFileSize recebe como argumento uma string que é o nome (caminho) do ficheiro e realiza o comando "ls -l {filename} | cut -d ' ' -f5" e retorna o resultado deste comando que é o tamanho do ficheiro (em bytes).

O método compress recebe duas strings como parâmetros que são os nomes (caminhos) para os dois ficheiros que pretendemos medir o NCD. O segundo ficheiro é um argumento opcional pois como se vê na fórmula do NCD, é necessário comprimir os dois ficheiros em conjunto mas também é necessário comprimir cada um deles individualmente. Para comprimir, este método utiliza a função "system" para executar comandos com o compressor indicado e usa também o método getFileSize para extrair o tamanho dos ficheiros comprimidos. Como retorno, este método retorna o tamanho do ficheiro comprimido.

O método compute recebe também duas strings (os nomes dos ficheiros), neste caso os dois argumentos são obrigatórios e calcula o NCD entre os dois ficheiros recorrendo ao método compress e aplicando a fórmula.

src/NCCD.cpp

A classe *NCCD* foi projetada para calcular a distância da imagem *x* dentro do conjunto de dados fornecido. O objeto fornece valores padrão para os seus atributos, "directory" e "context model", após a inicialização. A classe contém uma função, "*compute*" onde calcula a distância de *x* no conjunto de dados e retorna o valor real.

Como a classe usa uma aplicação externo, o *popen*, que foi utilizado para capturar o resultado e converter para o valor real.

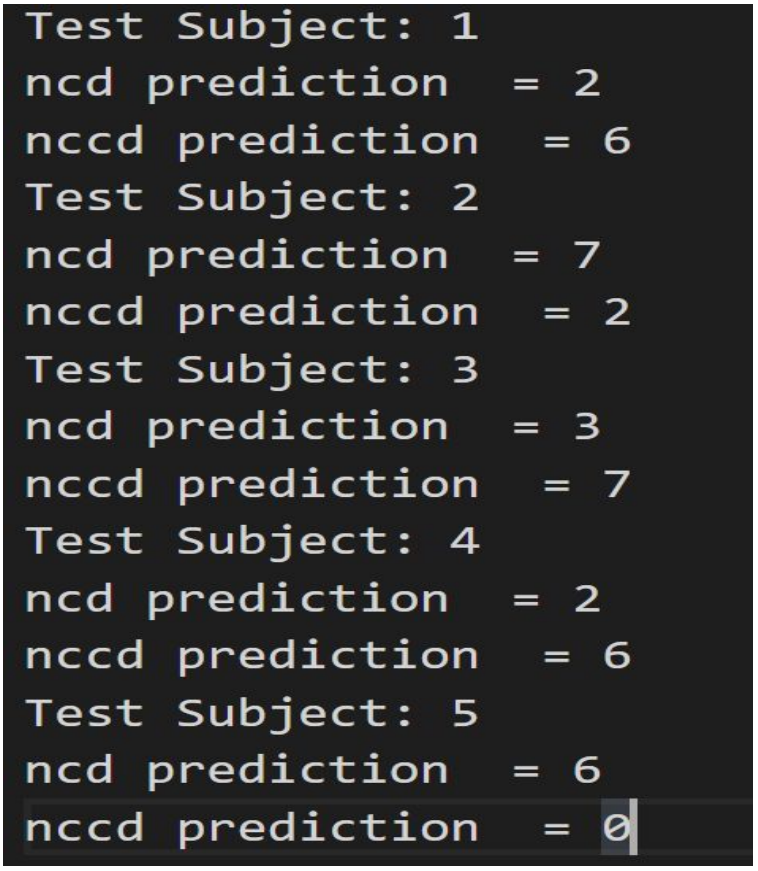
src/resize.cpp

O tempo de execução depende do tamanho do conjunto de dados e do tamanho de cada objeto. "*resize"* é uma aplicação da consola que pode redimensionar todos os dados por um valor real. Esta aplicação usa bibliotecas do *opencv2* para ler, redimensionar e guardar as imagens. Reduz a informação, preservando a aparência da imagem. As imagens de saída reduzidas são guardadas no caminho indicado.

src/test.cpp

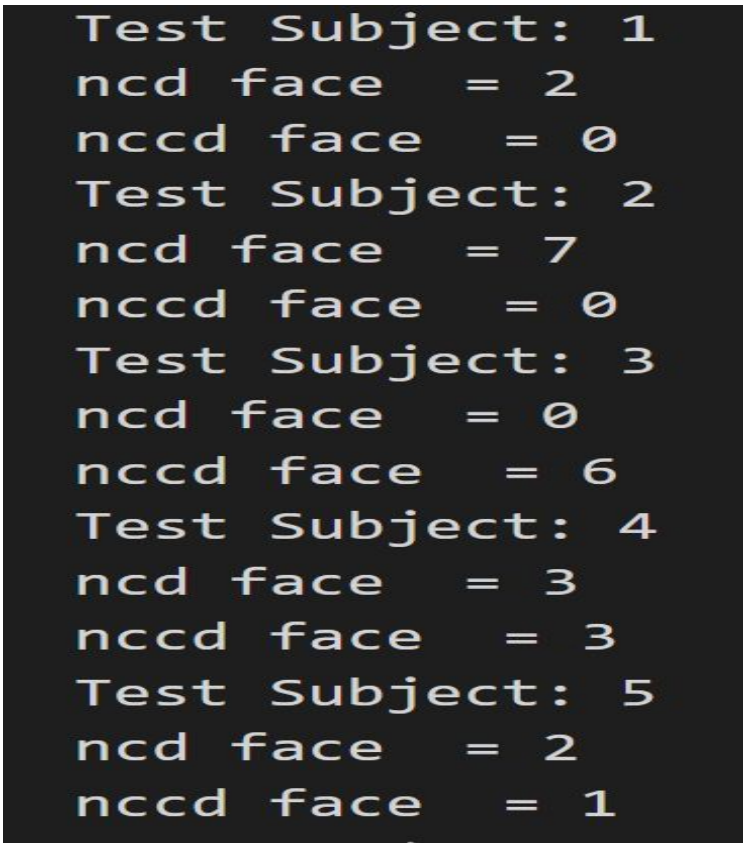
O ficheiro test.cpp é contém o script usado para realizar a experiência e calcular os resultados. Tal como explicado no guião, o dataset foi partido em 2 partes de 3/10 e 7/10 de imagens para cada pessoa em que as primeiras 3 imagens são o banco de dados e as restantes 7 são para realizar testes. A partir daqui foi realizada a experiência utilizando as classes acima, cujos resultados foram escritos no ficheiro.

Resultados



A imagem acima é um pequeno excerto do ficheiro “results.txt”. O ficheiro está organizado em Subject id, ncd prediction (0 - 7) em que 0 significa que aquela pessoa não foi identificada corretamente em nenhuma das 7 imagens de teste, e 7 significa que a pessoa foi identificada corretamente em todas as 7 imagens de teste, e nccd prediction cujo raciocínio é semelhante à linha de cima. Como se pode observar, não existe uma relação entre as previsões de NCD e NCCD, o que não permite retirar nenhuma conclusão. No entanto, mais abaixo no ficheiro, existem algumas pessoas a que ambas as previsões identificam corretamente no mesmo número de testes.

Podemos então concluir que devem ser usados os dois métodos para reconhecimento facial pois apenas um deles pode apresentar deformidades dependendo da pessoa e das imagens no banco de dados.



A imagem acima apresenta os mesmos resultados mas com as imagens reduzidas. Como se pode observar comparando as duas imagens, reduzir a resolução das imagens produz uma diminuição da Accuracy geral do algoritmo o que seria de se esperar já que há perda de informação.

Conclusão

Podemos concluir com este trabalho que NCD e NCCD são duas medidas que são uma alternativa a machine learning no reconhecimento facial e que dependendo das condições (número de imagens de treino, etc...) pode apresentar resultados satisfatórios. Podemos também concluir que embora ambas as medidas apresentem falhas facilmente observáveis pelos ficheiros, sendo combinadas possuem uma accuracy mais razoável.

Percentagem de trabalho de cada um dos membros

Isaac dos Anjos - 35%

Lucas Barros - 35%

Pedro Cavadas - 30%

Referências

https://github.com/premake/premake-core

https://opencv.org/

http://www.cplusplus.com

https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html