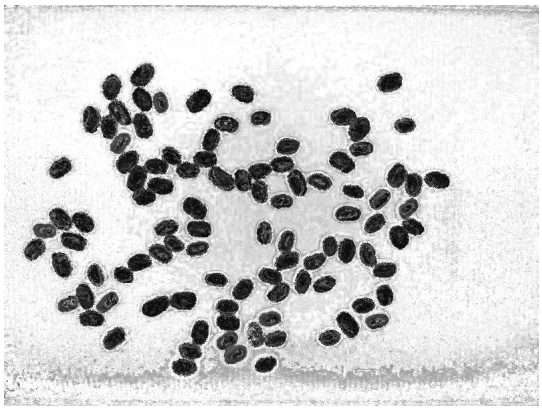
PSI3472 Concepção e Implementação de Sistemas Eletrônicos Inteligentes
Segundo Semestre de 2018 2º exercício-programa

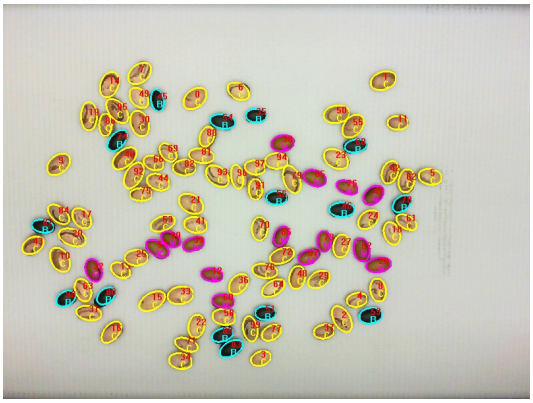
O objetivo deste exercício é exercitar-se no uso de tiny_dnn para implementar redes neurais.

O arquivo
www.lps.usp.br/hae/psi3472/ep2-2018/beans_pixelmap.zip
contém algumas das imagens utilizadas no artigo [1] para localizar e classificar os grãos em lotes de feijões. A figura 1a mostra a imagem original, a figura 1b mostra a imagem após passar pelo "mapeamento de pixels" usando FlaNN (vizinho mais próximo aproximado) e a imagem 1c mostra a localização e classificação final dos feijões em "carioca", "mulato" e "preto".

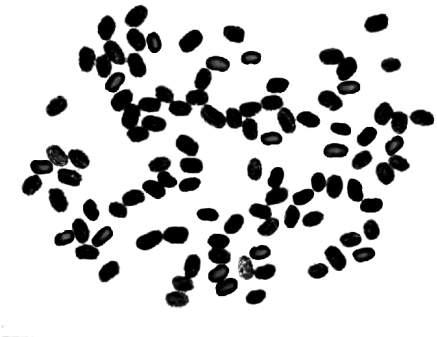
(a) img1.png: Imagem de entrada



(b) map1.png: Mapeamento de pixels usando FlaNN



(c) cla1.png: Imagem com feijões classificadas



(d) tin1.png: Mapeamento de pixels usando tiny-dnn

Figura 1: Exemplos de imagens.

Neste exercício, vamos resolver somente a primeira parte deste processo, isto é, o mapeamento de pixels em feijão (preto), fundo (branco) ou indeciso (diferentes níveis de cinza). Para isso, vamos utilizar as imagens de amostra "one?.ppm" e "zer??.ppm", respectivamente as cores do fundo e as cores do feijão. Usaremos rede neural construído usando tiny-dnn. A figura 1d mostra a saída obtida usando tiny-dnn.

Figura 2: Amostras de treinamento das cores de fundo (one*) e de feijão (zer*).

O seguinte comando:

```
>train-tiny 'zer*.ppm' 'one*.ppm' modelo.net
```

deve ler as imagens amostras de treinamento zer*.ppm e one*.ppm e, usando rede neural implementado usando tiny_dnn, construir um modelo de rede neural e salvar como modelo.net.

O seguinte comando:

```
>predict-tiny img1.png modelo.net tin1.png
```

deve ler img1.png e modelo.net e classificar cada pixel da imagem img1.png gerando a imagem tin1.png. Nessa imagem (tin1.png), os pixels dos feijões devem estar pintados de preto, os pixels de fundo devem estar pintados de branco e os pixels difíceis de classificar (olhando só a sua cor) devem estar pintados de diferentes tonalidades de cinza. Quanto mais a cor do pixel "parecer" a cor de um feijão, mais escuro deve ser a tonalidade de cinza do pixel.

Referências:

[1] S. A. Araújo, J. H. Pessota and H. Y. Kim, “Beans Quality Inspection Using Correlation-Based Granulometry,” *Engineering Applications of Artificial Intelligence*, vol. 40, pp. 84-94, April 2015.

Aula 6, exercício 3 (2 pontos): Faça um programa wildcard.cpp que imprime todos os nomes de arquivos que satisfazem um certo wildcard. Exemplo:

```
>wildcard 'zer*.ppm'
```

deve imprimir "zer0.ppm, zer1.ppm, zer2.ppm, zer3.ppm" se esses arquivos estiverem no diretório default.

Nota 1: Os apóstrofes (' ') impedem a expansão de wildcard pelo sistema.

Nota 2: Para ler todos os arquivos que satisfazem um certo wildcard (por exemplo, "zer*.ppm"), vocês podem usar o comando vsWildCard (do Cekeikon). O comando:

```
vector<string> nomes; vsWildCard("zer*.ppm",nomes);
```

irá colocar em "nomes" os nomes de todos os arquivos que casam com wildcard "zer*.ppm".

Aula 6, exercício 4 (4 pontos): Faça os programas train-tiny.cpp e predict-tiny.cpp de acordo com o enunciado. Deixei no site, como modelos, os programas train-tiny2.modelo.cpp e predict-tiny.modelo.cpp. Os programas estão quase prontos. Você só precisa completar o pouco que deixei em branco. Deixe gravado o modelo.net, para que possa mostrar rapidamente o seu programa funcionando ao professor/monitor.

Nota: O meu trein-tiny chegou ao erro 0.0170738.