Projeto em Identificação de Indivíduos por Impressão Digital
Professor: Alexandre Xavier Falcão
E-mail: afalcao@ic.unicamp.br
MO445/MC940 - Primeiro Semestre de 2022

1 Enunciado

Este projeto tem por objetivos exercitar e avaliar vocês sobre os conceitos aprendidos na disciplina MO445 / MC940 no contexto da identificação de invivíduos a partir de impressões digitais. O projeto está dividido em duas fases, cada fase utiliza uma base de imagens diferente: images_01 e images_02 para as fases 1 e 2, respectivamente.

As imagens para a fase 1 estão sobrescritas com anotações que podem dificultar a identificação dos indivíduos (Figura 1a). O código project01.c, que depende da libmo445 [1], contém funções que deverão ser codificadas para gerar uma região de interesse (ROI) contendo, o quanto for possível, a impressão digital apenas (Figura 1b). Vocês podem avaliar também a possibilidade de perder parte da impressão digital, extraindo uma ROI centrada de menor tamanho. Isso evitaria os riscos que sobraram na Figura 1b.

As imagens para a fase 2 já não apresentam este tipo de problema (Figuras 1c e 1d). Neste caso, porém, partes comuns das imagens de um mesmo indivíduo precisam ser alinhadas para que uma rede neural possa reconhecê-las como sendo do mesmo indivíduo. O código project02.c, que também depende da libmo445, recebe um arquivo com pares de imagens, denominadas fonte e destino, para serem comparadas; extrai uma ROI centrada na impressão digital da imagem fonte; alinha esta ROI com uma ROI correspondente na imagem destino; e prepara os dados para que os pares de ROIs correspondentes sejam analisados por uma rede neural (scripts em ./scripts), a fim de decidir se as imagens fonte e destino são ou não são do mesmo indivíduo.

2 Como desenvolver o projeto?

Para desenvolver a primeira fase do projeto, vocês devem preencher os códigos das funções indicadas em project01.c. Essas funções são necessárias em três operações do programa principal: uma filtragem alternada sequencial envolvendo fechamento seguido de abertura morfológica, uma filtragem de fechamento de buracos (bacias) em objetos e uma erosão morfológica, respectivamente. A dilatação e a erosão morfológicas para imagens binárias deverão seguir os algoritmos 1 e 2, respectivamente. Já o fechamento de buracos deverá seguir o algoritmo 3. **Esses algoritmos assumem que pixels de objeto possuem valor diferente de 0 (e.g., 1 ou 255) e pixels de fundo valor 0. Note que no caso do fechamento de buracos, ao conquistar um pixel q, este nunca estará na fila Q.** Além desses algoritmos, outras funções mais simples e auxiliares são indicadas para preenchimento no código do programa project01.c. Para avaliar se sua implementação está correta, vocês podem trocar sua função pela que está comentada no código do programa. Avaliem os resultados variando os hiperparâmetros dessas e

[1] Baixe a última versão da página do curso e altere o Makefile da pasta fingerprint do projeto para o endereço no qual instalou a libmo445 na sua máquina. Para compilar o programa, basta digitar make project01.

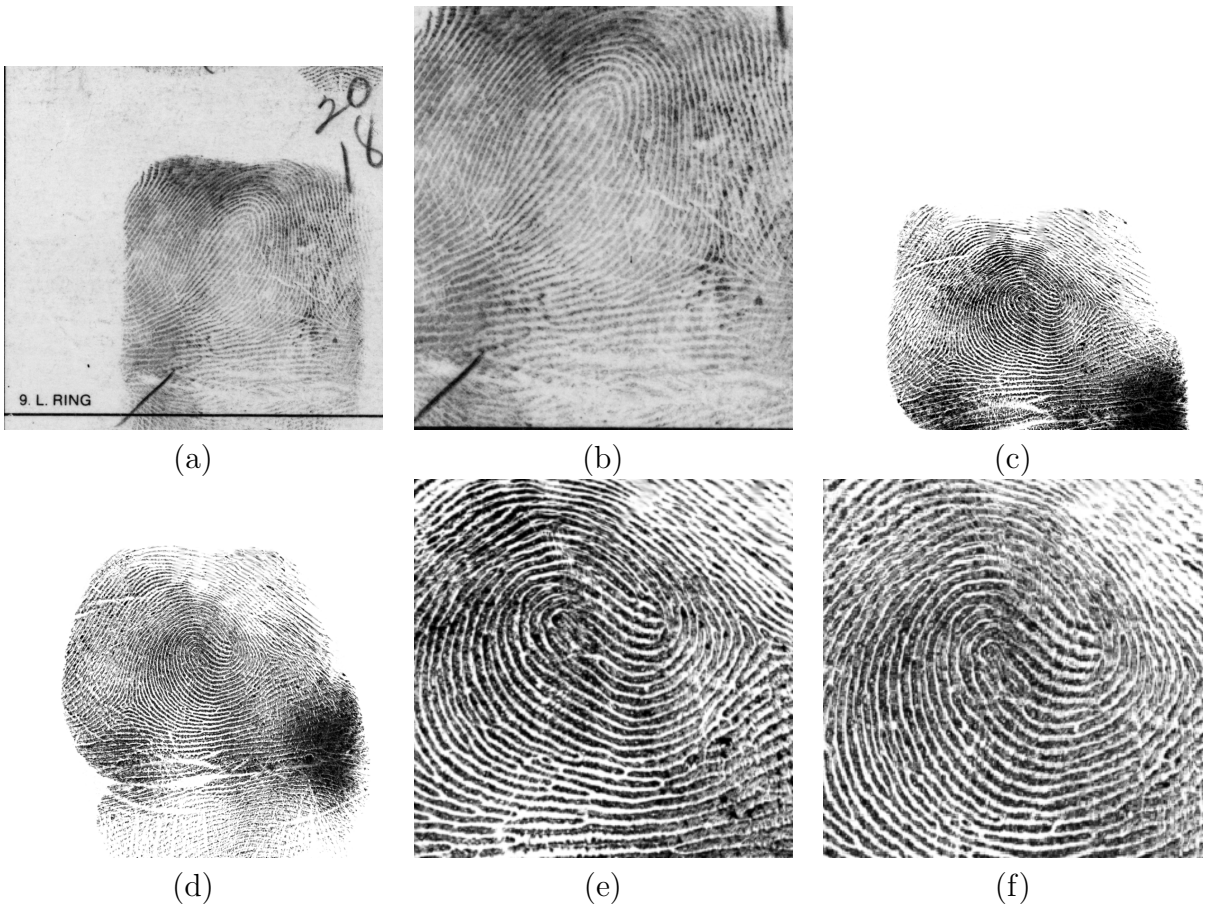

Figura 1: (a) Exemplo de imagem da base 1. (b) Região de interesse contendo a impressão digital em (a). (c) e (d) Exemplos de imagens fonte e destino, respectivamente, da base 2. (e) e (f) Exemplos de ROIs alinhadas das imagens (c) e (d), respectivamente.

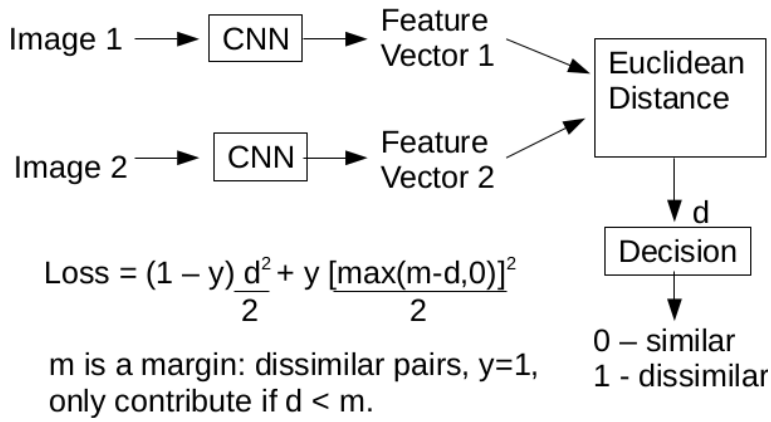

Figura 2: Rede siamesa cujo o objetivo é aprender um espaço de características reduzido em que imagens de uma mesma classe são bem mais similares do que imagens de classes distintas.

outras funções do código, visando sempre isolar a maior porção possível contendo apenas a impressão digital.

Para a fase 2 do projeto, vocês devem utilizar o programa project02.c para gerar primeiro um arquivo csv com pares de ROIs alinhados para comparação e uma pasta com essas ROIs a partir de um arquivo csv de comparação e da pasta images_02. O artigo do algoritmo de alinhamento está na pasta fingerprint. Vocês podem criar arquivos de comparação para treino, validação e teste, evitando repetir comparações entre eles. Um modelo pré-treinado em outra base está disponível na pasta models. As comparações poderão envolver pares de imagens do mesmo indivíduo (genuínas) ou pares de imagens de indivíduos distintos (impostoras). Procurem misturar de forma balanceada os casos de comparações genuínas e impostoras. O arquivo deltas.txt contém os hiperparâmetros para alinhamento entre as imagens fonte e destino do arquivo de comparações. Além dos deltas, as dimensões das ROIs podem ser modificadas para avaliar a qualidade do alinhamento. O programa project02.c gera também um arquivo texto com os scores (distâncias) obtidos, mas este arquivo não é utilizado no projeto. Na pasta scripts, vocês irão encontrar scripts Python para treinar, avaliar e executar a rede neural em pares de ROIs gerados por project02.c. Além dos hiperparâmetros do alinhamento, vocês devem tentar melhorar a arquitetura da rede neural e sua função de perda em siameseNN.py. Para facilitar a tarefa, vejam o relatório e os códigos na pasta fingerprint sobre aprendizado contrastivo.

A Figura 2 mostra como treinar um modelo de rede siamesa por aprendizado contrastivo. A função de perda utiliza a distância d entre os vetores de características das imagens de entrada. A variável y deve ser 0 quando as imagens são consideradas similares (e.g., mesma classe) e 1 no caso contrário. Então a função de perda só utilizará distâncias entre imagens dissimilares (de classes distintas) se $d < m$ (a distância for menor do que a margem). Para facilitar esta parte da tarefa, vocês podem trabalhar também com o notebook contrastive_learning.ipynb. Ele contém explicações sobre os módulos e exemplos de possíveis alterações na rede para avaliação.

Algoritmo 1 – Dilatação Morfológica

ENTRADA: Máscara binária $\hat{I} = (D_I, I)$, conjunto S vazio ou com pixels de bordas **internas**, e raio de dilatação γ.
SAÍDA: Máscara dilatada $\hat{D} = (D_I, D)$ e conjunto S com pixels de bordas **externas**.
AUXILIARES: Relações de Adjacência $A_{\sqrt{2}}$ e A_1, fila de prioridades Q, mapa de custos C, mapa de raizes R, variável *tmp*.

1. **Para Cada** $p \in D_I$ **Faça** $C(p) \leftarrow +\infty$ *e* $D(p) \leftarrow I(p)$.
2. **Se** $S = \emptyset$ **Então** $S \leftarrow \{p \in D_I \mid I(p) \neq 0$ *e* $\exists q \in A_1(p), I(q) = 0\}$.
3. **Enquanto** $S \neq \emptyset$ **Faça**
4. | *Remova p de S.*
5. | *Atribua* $C(p) \leftarrow 0$, $R(p) \leftarrow p$, *e insira p em Q.*
6. **Enquanto** $Q \neq \emptyset$ **Faça**
7. | *Remova p de Q tal que* $p = arg\min_{\forall q \in Q}\{C(q)\}$.
8. | **Se** $C(p) \leq \gamma^2$ **Então**
9. | | *Atribua* $J(p) \leftarrow I(R(p))$.
10. | | **Para Cada** $q \in A_{\sqrt{2}}(p) \mid q \in D_I$ *e* $C(q) > C(p)$
11. | | *e* $I(q) = 0$ **Faça**
12. | | | *Atribua* $tmp \leftarrow \|q - R(p)\|^2$.
13. | | | **Se** $tmp < C(q)$ **Então**
14. | | | | **Se** $q \in Q$ **Então** *Remova q de Q.*
15. | | | | $C(q) \leftarrow tmp$ *e* $R(q) \leftarrow R(p)$.
16. | | | | *Insira q em Q.*
17. | **Senão** *Atribua* $S \leftarrow S \cup \{p\}$.
18. **Retorne** $\hat{D}$ *e* S.

Algoritmo 2 – EROSÃO MORFOLÓGICA

ENTRADA: Máscara binária $\hat{I} = (D_I, I)$, conjunto S vazio ou com pixels de bordas **externas**, e raio de erosão γ.
SAÍDA: Máscara erodida $\hat{E} = (D_I, E)$ e conjunto S com pixels de bordas **internas**.
AUXILIARES: Relações de Adjacência $A_{\sqrt{2}}$ e A_1, fila de prioridades Q, mapa de custos C, mapa de raizes R, variável *tmp*.

1. **Para Cada** $p \in D_I$ **Faça** $C(p) \leftarrow +\infty$ *e* $E(p) \leftarrow I(p)$.
2. **Se** $S = \emptyset$ **Então** $S \leftarrow \{p \in D_I \mid I(p) = 0$ *e* $\exists q \in A_1(p), I(q) \neq 0\}$.
3. **Enquanto** $S \neq \emptyset$ **Faça**
4. | *Remova p de S.*
5. | *Atribua* $C(p) \leftarrow 0$, $R(p) \leftarrow p$, *e insira p em Q.*
6. **Enquanto** $Q \neq \emptyset$ **Faça**
7. | *Remova p de Q tal que* $p = arg\min_{\forall q \in Q}\{C(q)\}$.
8. | **Se** $C(p) \leq \gamma^2$ **Então**
9. | | *Atribua* $J(p) \leftarrow I(R(p))$.
10. | | **Para Cada** $q \in A_{\sqrt{2}}(p) \mid q \in D_I, C(q) > C(p)$

```
11.          e I(q) ≠ 0 Faça
12.              Atribua tmp ← ||q − R(p)||².
13.              Se tmp < C(q) Então
14.                  Se q ∈ Q Então Remova q de Q.
15.                  C(q) ← tmp e R(q) ← R(p).
16.                  Insira q em Q.
17.      Senão Atribua S ← S ∪ {p}.
18. Retorne Ê e S.
```

Algoritmo 3 – Fechamento de Bacias

Entrada:	Imagem $\hat{I} = (D_I, I)$.
Saída:	Imagem com bacias fechadas $\hat{C} = (D_I, C)$.
Auxiliares:	Relação de Adjacência A_1, fila de prioridades Q, variável tmp.

```
1.  Para Cada p ∈ D_I Faça
2.      Faça C(p) ← +∞
3.      Para Cada q ∈ A_1(p) Faça
4.          Se q ∉ D_I Então
5.          Faça C(p) ← I(p), insira p em Q e break.
6.  Enquanto Q ≠ ∅ Faça
7.      Remova p de Q tal que p = arg min_{∀q∈Q}{C(q)}.
8.      Para Cada q ∈ A_1(p) | q ∈ D_I e C(q) > C(p) Faça
9.      Atribua tmp ← max{C(p), I(q)}.
10.     Se tmp < C(q) Então
11.         C(q) ← tmp.
12.         Insira q em Q.
13. Retorne Ĉ.
```

3 Como escrever os relatórios?

O formato dos relatórios segue o padrão descrito na página do curso. Lembrem-se de documentar os códigos e explicar no relatório todos os experimentos realizados, acrescentar tabelas com os resultados obtidos, discutir os resultados, e ilustrá-los com figuras.

4 Prazos.

1. Fase 1: 12/05/2022 (extração da ROI).
2. Fase 2: 26/05/2022 (alinhamento) e 12/07/2022 (rede neural).