Reconhecimento de células em exames de Papanicolau

Thiago Guerra, Thiago Utsch, Uriel Andrade

June 18, 2024

1 Introdução

O exame de Papanicolau é um procedimento histológico utilizado para identificar alterações celulares no colo do útero. Ele é o método principal para a detecção precoce de lesões que podem indicar câncer cervical. Neste projeto, desenvolvemos um aplicativo na linguagem de programação "Python" que analise imagens de exames de Papanicolau e permita o reconhecimento automático de células cancerosas.

2 Conceitos

2.1 Haralick

Os descritores de Haralick são um conjunto de 14 medidas estatísticas utilizadas para calcular a textura de uma imagem digital. Eles foram propostos por Robert M. Haralick em 1973 e são baseados na matriz de co-ocorrência, que é uma representação da distribuição espacial dos pixels que compõem uma textura.

A matriz de co-ocorrência é definida a partir da probabilidade conjunta de ocorrência de pares de valores de pixels para cada distância e direção. Essa matriz é geralmente discretizada para ângulos de 0°, 45°, 90° e 135°, e distâncias não euclidianas (como chessboard ou cityblock) também são comuns.

A partir da matriz de co-ocorrência, obtém-se distribuições marginais, bem como suas médias e variâncias. Algumas das medidas mais comumente extraídas da matriz de co-ocorrência são: energia, contraste, correlação, variância, inverso do momento diferencial, soma média, soma da variância, soma da entropia, entropia, diferença da variância, diferença da entropia, medida de informação de correlação 1, medida de informação de correlação 2 e medida de correlação máxima.

Os descritores de Haralick são amplamente utilizados em diversas aplicações, como detecção de melanoma, reconhecimento de padrões e análise de texturas. Eles são capazes de quantificar propriedades de suavidade, rugosidade e regularidade das texturas, além de identificar características estatísticas ou propriedades estruturais locais constantes.

2.2 Invariantes de Hu

Os Invariantes de Hu, também conhecidos como Momentos Invariantes de Hu, são um conjunto de 7 números calculados usando momentos centrais que são invariantes a transformações de imagem. Eles foram definidos por Ming-Kuei Hu em 1962.

Os Invariantes de Hu são usados para reconhecer formas 2D em diferentes posições (traslações), orientações (rotações) e tamanhos (escala). Eles permitem identificar um determinado objeto mesmo que tenha sofrido mudança de tamanho ou mesmo que seja rotacionado.

2.3 Matriz de Co-ocorrência

Uma matriz de co-ocorrência é uma ferramenta amplamente utilizada em diversas áreas, como processamento de imagens, análise de textos e aprendizado de máquina, para capturar e representar a frequência de ocorrência conjunta de diferentes pares de elementos ou características em um conjunto de dados. A seguir, explicarei como a matriz de co-ocorrência é usada em dois contextos principais: processamento de imagens e análise de textos.

2.3.1 Matriz de Co-ocorrência em Processamento de Imagens

No processamento de imagens, a matriz de co-ocorrência é usada para analisar a textura da imagem. Ela é construída calculando a frequência com que pares de pixels com valores de intensidade específicos ocorrem em uma determinada relação espacial em uma imagem. Aqui está um exemplo de como construir essa matriz:

1- Definir a Relação Espacial: Pode ser definida em termos de deslocamento (dx, dy), como 1 pixel à direita (1,0) ou 1 pixel acima (0,1).

2- Construir a Matriz: Considere uma imagem em escala de cinza onde cada pixel tem um valor de intensidade, para cada pixel na imagem, observe o pixel na posição deslocada pela relação espacial. Conte quantas vezes cada par de valores de intensidade ocorre.

Suponha que temos uma imagem em escala de cinza com valores de intensidade de 0 a 255. A matriz de co-ocorrência seria uma matriz 256x256 onde a entrada (i, j) representa a frequência com que o valor de intensidade i ocorre ao lado do valor de intensidade j na relação espacial definida.

2.4 HSV

O espaço de cores HSV (Hue, Saturation, Value) é um sistema de cores que define o espaço de cor utilizando três parâmetros:

1. Matiz (Hue): Define o tipo de cor, abrangendo todas as cores do espectro, desde o vermelho até o violeta, mais o magenta. Atinge valores de 0 a 360, mas para algumas aplicações, esse valor é normalizado de 0 a 100 porcento.

2. Saturação (Saturation): Também chamado de "pureza". Quanto menor esse valor, mais com tom de cinza aparecerá a imagem. Quanto maior o valor, mais "pura" é a imagem. Atinge valores de 0 a 100 porcento.

3. Valor (Value) ou Brilho (Brightness): Define o brilho da cor. Atinge valores de 0 a 100 porcento.

Uma cor no espaço HSV é especificada informando um ângulo de matiz, o nível de croma e o nível de leveza. Um ângulo de matiz de zero é vermelho. O ângulo de matiz aumenta em uma direção no sentido anti-horário. As cores complementares têm 180 diferenças.

Este sistema foi inventado no ano de 1974, por Alvy Ray Smith. É caracterizado por ser uma transformação não-linear do sistema de cores RGB. Outros sistemas de cores relacionados incluem o HSL (L de luminosity ou luminosidade) e o HSI (I de intensity ou intensidade).

Os espaços de cor HSV são geralmente usados por artistas e podem ser dependentes do dispositivo ou independentes do dispositivo.

3 Técnicas Implementadas

3.1 SVM

O SVM, ou Máquina de Vetores de Suporte, é um poderoso algoritmo de aprendizado de máquina utilizado principalmente para tarefas de classificação, mas que também pode ser aplicado em problemas de regressão e detecção de anomalias. Seu princípio fundamental é encontrar um hiperplano que melhor separa as diferentes classes de dados em um espaço de alta dimensão. Este hiperplano é escolhido de modo a maximizar a margem, que é a distância entre ele e os pontos de dados mais próximos de qualquer classe, conhecidos como vetores de suporte. O objetivo é que este hiperplano tenha a maior margem possível para garantir uma melhor generalização dos dados.

A capacidade do SVM de lidar com dados não linearmente separáveis é uma de suas características mais impressionantes. Isso é possível graças ao uso de funções kernel, que transformam os dados originais em um espaço de maior dimensão onde um hiperplano linear pode efetivamente separar as classes. Alguns dos kernels mais comuns incluem o linear, polinomial, radial basis function (RBF) e sigmoide. O uso de kernels permite que o SVM capture relações complexas entre as características dos dados sem a necessidade de realizar transformações explícitas nos dados originais.

O desempenho do SVM é altamente dependente da escolha dos parâmetros, como o tipo de kernel e os parâmetros associados a ele. Por exemplo, no caso do kernel RBF, os parâmetros de regularização e gama são cruciais para definir o comportamento do classificador. A regularização controla a complexidade do modelo e evita overfitting, enquanto o parâmetro gama define a influência de um

único exemplo de treinamento. Ajustar esses parâmetros corretamente é essencial para obter um modelo robusto e preciso, muitas vezes necessitando de técnicas como validação cruzada para otimizar os resultados.

A aplicação do SVM é vasta e abrange diversas áreas, incluindo bioinformática, reconhecimento de imagem, e processamento de texto. Sua robustez e capacidade de generalização fazem dele uma escolha popular em cenários onde a precisão e a eficácia são fundamentais. No entanto, um dos desafios do SVM é seu custo computacional, especialmente em grandes conjuntos de dados, uma vez que a complexidade do algoritmo aumenta com o número de amostras. Apesar disso, avanços em técnicas de otimização e implementações eficientes têm mitigado esse problema, permitindo que o SVM continue sendo uma ferramenta valiosa no arsenal do aprendizado de máquina.

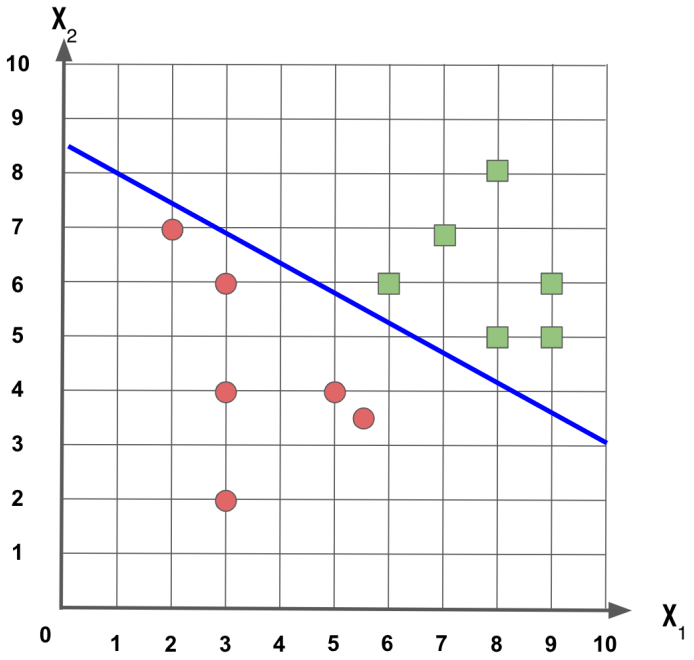

Figure 1: Gráfico Exemplo SVM.

3.2 ResNet50

A ResNet50 é uma das arquiteturas de redes neurais profundas mais influentes e amplamente utilizadas, especialmente em tarefas de reconhecimento de imagens. Desenvolvida por pesquisadores da Microsoft, a ResNet, abreviação de Residual Network, revolucionou a forma como as redes neurais são treinadas, especialmente quando se trata de redes muito profundas. Composta por 50 camadas, a ResNet50 consegue lidar com o problema do desaparecimento do gradiente, um desafio comum em redes neurais profundas onde os gradientes se tornam extremamente pequenos, dificultando a atualização dos pesos durante o treinamento.

O principal avanço da ResNet50 reside na introdução de blocos residuais. Esses blocos contêm conexões de atalhos que permitem que o gradiente flua diretamente através das camadas, facilitando o treinamento de redes muito profundas. Essencialmente, essas conexões de atalhos realizam uma soma elementar da entrada e da saída de uma camada convolucional, permitindo que a rede aprenda a identidade da função se isso for necessário. Essa abordagem não só ajuda a mitigar o problema do desaparecimento do gradiente, mas também permite que a rede se beneficie do aumento da profundidade, aprendendo representações mais complexas dos dados.

A arquitetura ResNet50 foi testada e validada em diversos benchmarks de reconhecimento de imagem, como o ImageNet, onde apresentou um desempenho superior em comparação com arquiteturas anteriores. Sua capacidade de alcançar alta precisão sem aumentar significativamente o número de parâmetros faz dela uma escolha eficiente e poderosa para muitas aplicações de visão computacional. Além disso, a ResNet50 tem sido amplamente adotada em transfer learning, onde modelos pré-treinados

são utilizados como ponto de partida para resolver novos problemas com conjuntos de dados específicos, economizando tempo e recursos computacionais.

O impacto da ResNet50 vai além do reconhecimento de imagens, influenciando a pesquisa em diversas áreas do aprendizado profundo. Sua arquitetura foi adaptada e estendida para outras tarefas, como segmentação de imagens, detecção de objetos e até em modelos de linguagem natural. A introdução dos blocos residuais não só melhorou o desempenho dos modelos, mas também abriu caminho para a exploração de redes ainda mais profundas, como a ResNet101 e ResNet152, demonstrando que, com as técnicas corretas, redes neurais profundas podem ser treinadas de maneira eficaz e eficiente.

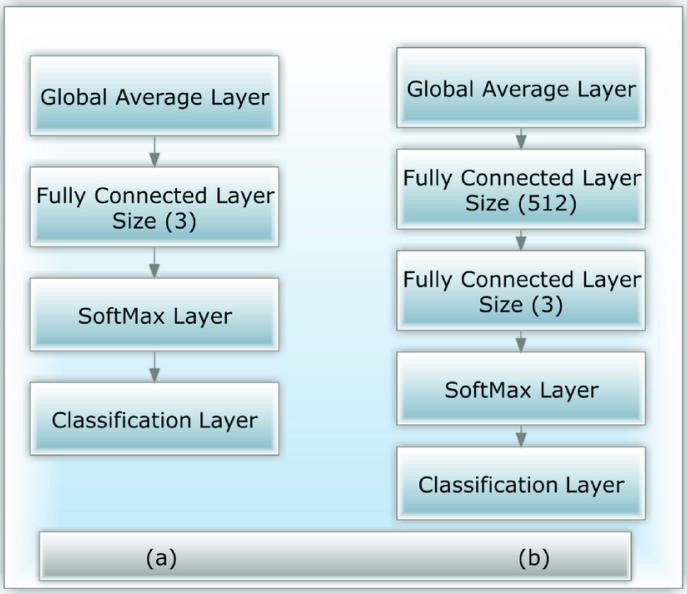

Figure 2: Versões do ResNet50.

4 Bibliotecas Utilizadas

4.1 Pandas

A biblioteca Pandas é uma poderosa ferramenta de código aberto para a manipulação e análise de dados em Python. Ela oferece estruturas de dados rápidas, flexíveis e expressivas, como DataFrames e Series, que facilitam a limpeza, transformação e análise de grandes conjuntos de dados. Utilizando Pandas, é possível realizar operações complexas com poucas linhas de código, como junções de tabelas, filtragem, agrupamento e agregação de dados. Além disso, a biblioteca integra-se perfeitamente com outras ferramentas do ecossistema científico de Python, como NumPy e Matplotlib, permitindo uma análise de dados mais eficiente e visualmente rica. A documentação abrangente e uma vasta comunidade de usuários tornam o Pandas uma escolha essencial para cientistas de dados, analistas e desenvolvedores.

4.2 Tkinter

A biblioteca Tkinter é a interface padrão do Python para a criação de aplicações gráficas de usuário (GUIs). Ela fornece um conjunto robusto e extensível de ferramentas que permitem o desenvolvimento de interfaces gráficas de forma simples e eficiente. Tkinter facilita a criação de janelas, botões, menus, caixas de diálogo, e outros elementos de interface, utilizando uma abordagem orientada a objetos. Como parte integrante da biblioteca padrão do Python, Tkinter não requer instalação adicional e é altamente acessível tanto para iniciantes quanto para desenvolvedores experientes. A flexibilidade e a simplicidade de uso do Tkinter, combinadas com sua capacidade de ser executada em diferentes plataformas, tornam-no uma escolha popular para o desenvolvimento de aplicações desktop interativas.

4.3 Pillow

A biblioteca Pillow é uma extensão moderna e amigável da Python Imaging Library (PIL), projetada para facilitar o processamento e a manipulação de imagens em Python. Pillow oferece uma ampla gama de funcionalidades, incluindo abertura, modificação e salvamento de vários formatos de imagem, bem como a aplicação de filtros, transformações geométricas, ajustes de cores e operações de desenho. Integrando-se perfeitamente com outras bibliotecas como Tkinter, Pillow permite a incorporação de imagens em interfaces gráficas de usuário, tornando-a uma ferramenta poderosa para desenvolvedores que trabalham com aplicações gráficas e processamento de imagens. Sua facilidade de uso e extensa documentação contribuem para sua popularidade entre desenvolvedores e cientistas de dados.

4.4 Matplotlib

A biblioteca Matplotlib é uma das ferramentas mais amplamente utilizadas em Python para a criação de visualizações gráficas de dados. Ela permite gerar uma vasta gama de gráficos, desde simples plots de linhas até complexas visualizações tridimensionais e animações. Com Matplotlib, os usuários podem personalizar quase todos os aspectos das suas visualizações, como cores, estilos de linha, marcações e rótulos, facilitando a criação de gráficos que são tanto informativos quanto visualmente atraentes. A biblioteca é altamente compatível com outras ferramentas do ecossistema científico de Python, como NumPy e Pandas, permitindo uma integração eficiente de dados e gráficos. Além disso, Matplotlib possui uma documentação abrangente e uma comunidade ativa de usuários, o que torna o processo de aprendizado e resolução de problemas mais acessível para desenvolvedores de todos os níveis.

4.5 Skimage

A biblioteca scikit-image (skimage) é uma poderosa e versátil ferramenta para processamento de imagens em Python, projetada para atender tanto a pesquisadores quanto a engenheiros de software. Ela oferece um amplo conjunto de algoritmos para tarefas como filtragem, segmentação, morfologia, transformações geométricas e análise de características. Desenvolvida como uma extensão da biblioteca SciPy, scikit-image integra-se facilmente com outras ferramentas do ecossistema científico de Python, como NumPy e Matplotlib, facilitando o desenvolvimento de pipelines de processamento de imagens robustos e eficientes. Além disso, a biblioteca é bem documentada e suportada por uma comunidade ativa, o que torna a implementação de técnicas avançadas de processamento de imagens mais acessível e prática para profissionais de diversas áreas.

4.6 Mahotas

A biblioteca Mahotas é uma ferramenta especializada para processamento de imagens em Python, conhecida por sua eficiência e abrangência de funcionalidades. Desenvolvida com foco em desempenho, Mahotas é escrita em C++ e oferece uma ampla gama de algoritmos rápidos para tarefas como filtragem, segmentação, morfologia, e extração de características. Destaca-se especialmente em aplicações que requerem processamento de alto desempenho em imagens grandes ou em lote. Além de sua velocidade, Mahotas é fácil de integrar com outras bibliotecas do ecossistema Python, como NumPy e SciPy, proporcionando uma experiência de desenvolvimento fluida e poderosa. A documentação detalhada e exemplos práticos ajudam os desenvolvedores a aproveitar ao máximo suas capacidades, tornando Mahotas uma escolha popular entre cientistas e engenheiros que precisam de soluções avançadas e rápidas para o processamento de imagens.

4.7 OS

A biblioteca os em Python é uma ferramenta essencial para a interação com o sistema operacional de forma eficiente e abrangente. Ela fornece uma interface portátil para a manipulação de arquivos e diretórios, permitindo tarefas como navegação pelo sistema de arquivos, criação e remoção de diretórios, e execução de comandos do sistema. A biblioteca os também facilita a manipulação de variáveis de ambiente e a obtenção de informações detalhadas sobre o sistema operacional e processos em execução. Por ser uma parte integrante da biblioteca padrão do Python, a os é amplamente utilizada e confiável, oferecendo funções críticas para o desenvolvimento de scripts de automação, gerenciamento de sistemas e aplicativos que precisam interagir diretamente com o ambiente do sistema operacional.

5 Análise dos resultados

5.1 SVM - Binário

Tempo de execução: 1:32

5.1.1 Matrix de confusão

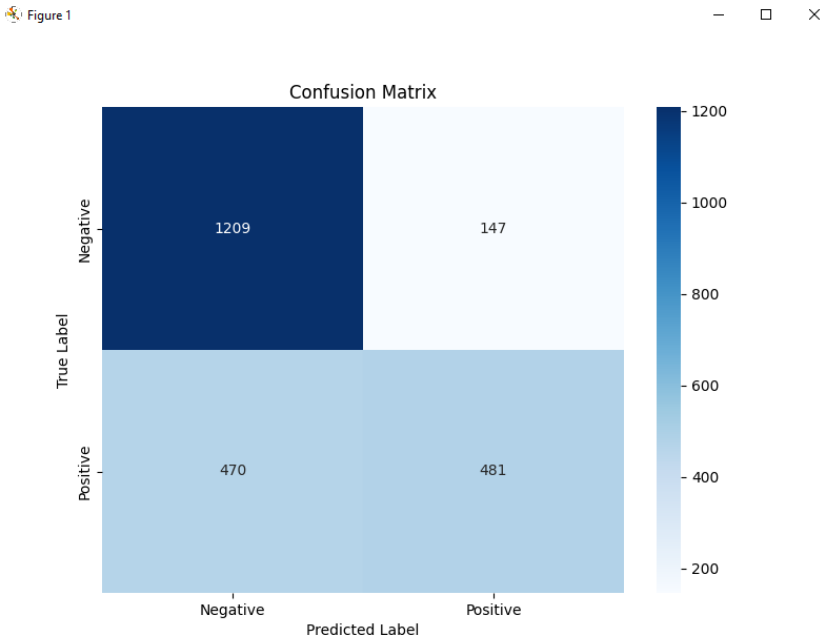

Figure 3: Matrix Confusão SVM - Binário.

5.1.2 Acurácia

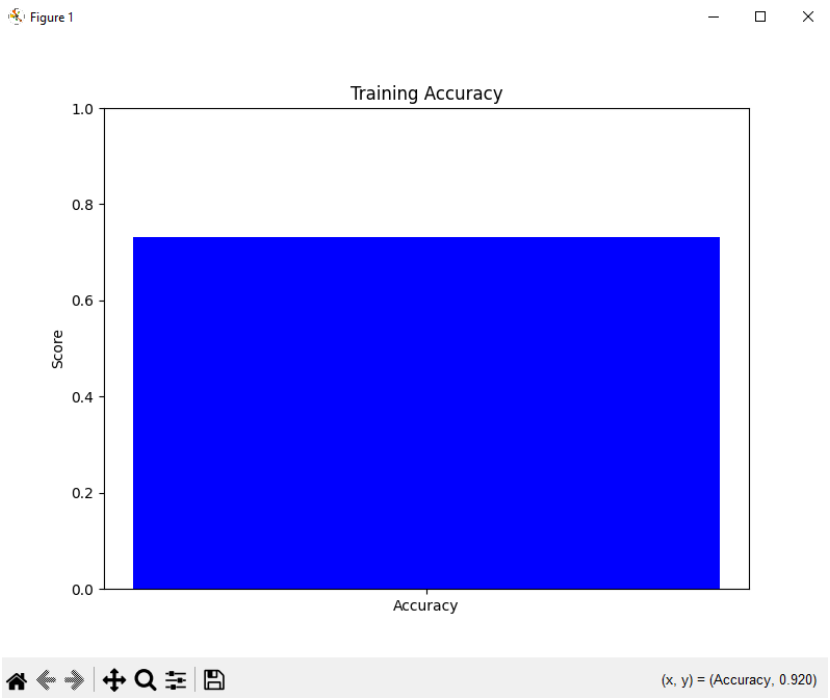

Figure 4: Acurácia SVM - Binário.

5.1.3 Classificação

```
[Running] python -u "c:\Users\WazX\Downloads\Papanicolau-main\teste3.py"
Accuracy: 0.7325530992631123
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.89      0.80      1356
           1       0.77      0.51      0.61       951

    accuracy                           0.73      2307
   macro avg       0.74      0.70      0.70      2307
weighted avg       0.74      0.73      0.72      2307


Training Results
Metric: ['Accuracy']
Value: [0.7325530992631123]
```

Figure 5: Classificação SVM - Binário.

5.2 SVM - 6 classes

Tempo de execução 1:53

5.2.1 Matrix de confusão

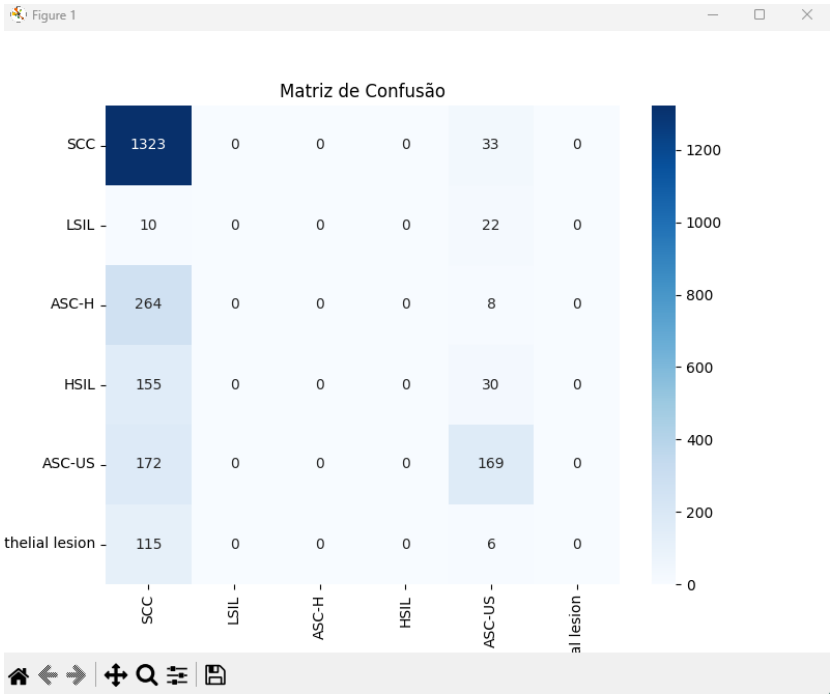

Figure 6: Matrix Confusão SVM - 6 classes.

5.2.2 Acurácia

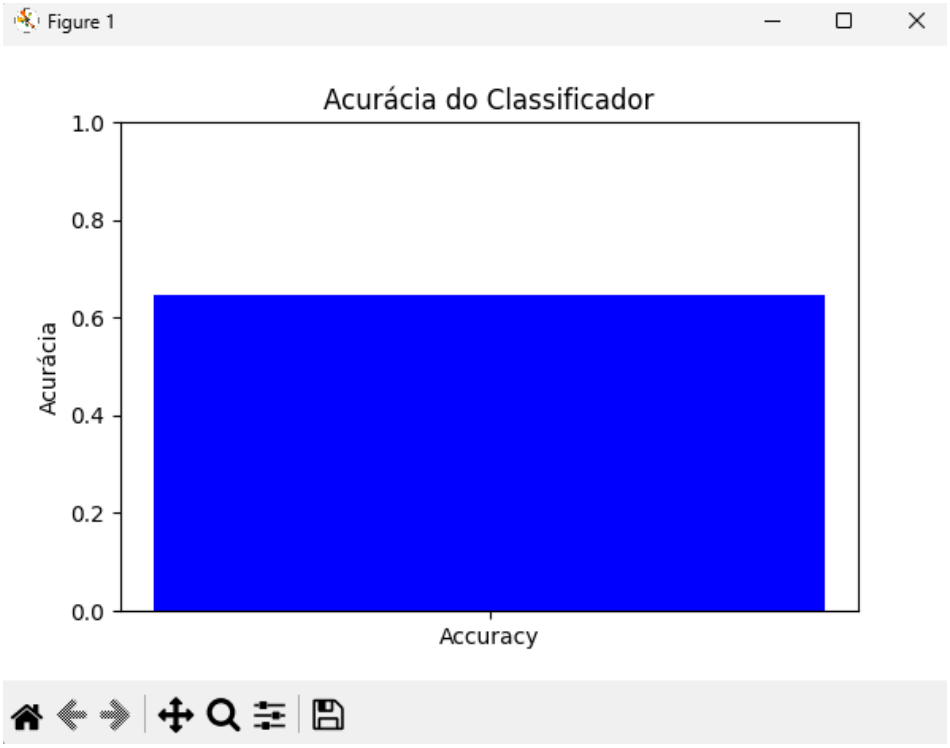

Figure 7: Acurácia SVM - 6 classes.

5.2.3 Classificação

```
Acurácia: 0.646727351538795
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.98      0.78      1356
           1       0.00      0.00      0.00        32
           2       0.00      0.00      0.00       272
           3       0.00      0.00      0.00       185
           4       0.63      0.50      0.56       341
           5       0.00      0.00      0.00       121

    accuracy                           0.65      2307
   macro avg       0.21      0.25      0.22      2307
weighted avg       0.47      0.65      0.54      2307
```

Figure 8: Classificação SVM - 6 classes.

5.3 ResNet50 - Binário

Tempo de execução: 3:08

5.3.1 Matrix de confusão

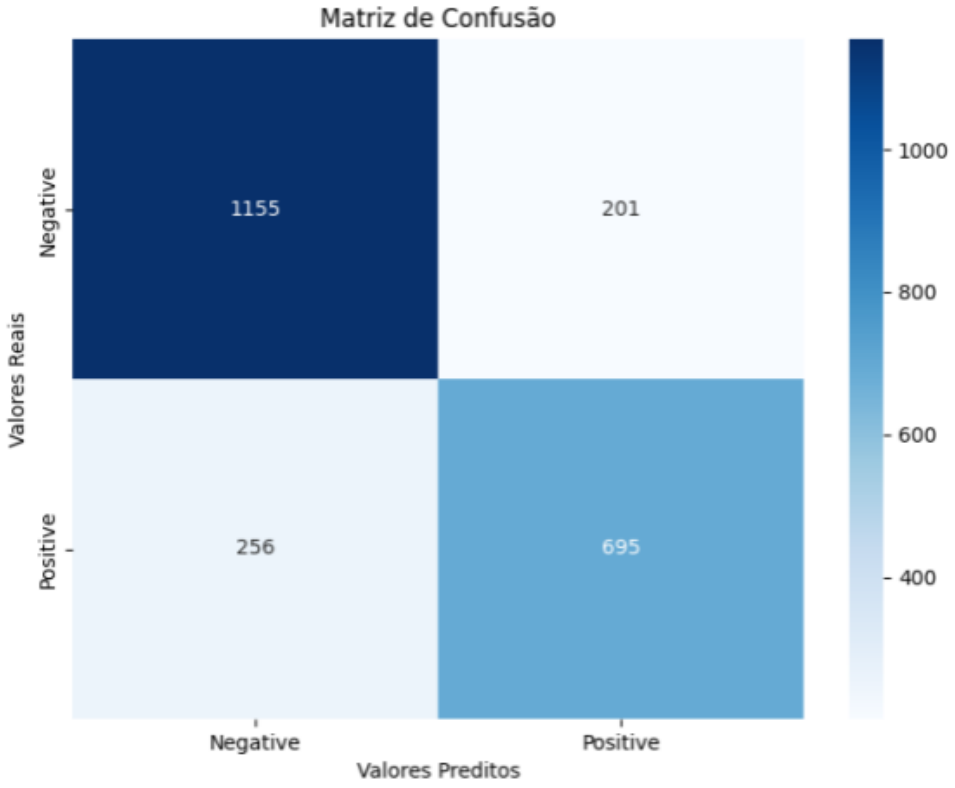

Figure 9: Matrix Confusão ResNet50 - Binário.

5.3.2 Acurácia

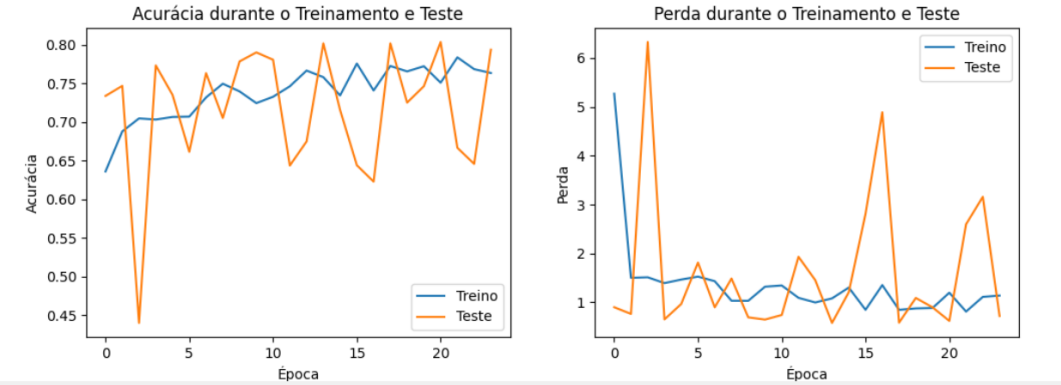

Figure 10: Acurácia ResNet50 - Binário.

5.3.3 Classificação

```
Epoch 1/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 233s 794ms/step - accuracy: 0.5852 - loss: 15.3195 - val_accuracy: 0.7339 - val_loss: 0.8991
Epoch 2/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 788ms/step - accuracy: 0.6863 - loss: 1.4852 - val_accuracy: 0.7469 - val_loss: 0.7625
Epoch 3/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 227s 786ms/step - accuracy: 0.6930 - loss: 1.8036 - val_accuracy: 0.4400 - val_loss: 6.3279
Epoch 4/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 230s 797ms/step - accuracy: 0.6833 - loss: 1.7620 - val_accuracy: 0.7733 - val_loss: 0.6500
Epoch 5/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 789ms/step - accuracy: 0.7010 - loss: 1.4508 - val_accuracy: 0.7352 - val_loss: 0.9653
Epoch 6/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 227s 786ms/step - accuracy: 0.7195 - loss: 1.2730 - val_accuracy: 0.6615 - val_loss: 1.8131
Epoch 7/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 227s 787ms/step - accuracy: 0.7412 - loss: 1.2600 - val_accuracy: 0.7633 - val_loss: 0.8970
Epoch 8/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 229s 792ms/step - accuracy: 0.7689 - loss: 0.9135 - val_accuracy: 0.7052 - val_loss: 1.4859
Epoch 9/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 789ms/step - accuracy: 0.7442 - loss: 0.9954 - val_accuracy: 0.7785 - val_loss: 0.6919
Epoch 10/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 789ms/step - accuracy: 0.7259 - loss: 1.3179 - val_accuracy: 0.7902 - val_loss: 0.6461
Epoch 11/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 790ms/step - accuracy: 0.7023 - loss: 1.8208 - val_accuracy: 0.7807 - val_loss: 0.7403
Epoch 12/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 229s 793ms/step - accuracy: 0.7634 - loss: 0.8745 - val_accuracy: 0.6437 - val_loss: 1.9313
Epoch 13/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 789ms/step - accuracy: 0.7557 - loss: 1.1677 - val_accuracy: 0.6749 - val_loss: 1.4531
Epoch 14/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 229s 793ms/step - accuracy: 0.7411 - loss: 1.1997 - val_accuracy: 0.8019 - val_loss: 0.5777
Epoch 15/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 227s 786ms/step - accuracy: 0.7172 - loss: 1.5592 - val_accuracy: 0.7156 - val_loss: 1.2070
Epoch 16/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 229s 791ms/step - accuracy: 0.7576 - loss: 0.9926 - val_accuracy: 0.6441 - val_loss: 2.8122
Epoch 17/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 227s 786ms/step - accuracy: 0.7297 - loss: 1.4581 - val_accuracy: 0.6229 - val_loss: 4.8893
Epoch 18/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 790ms/step - accuracy: 0.7652 - loss: 1.0358 - val_accuracy: 0.8019 - val_loss: 0.5828
Epoch 19/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 791ms/step - accuracy: 0.7680 - loss: 0.8614 - val_accuracy: 0.7252 - val_loss: 1.0890
Epoch 20/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 229s 792ms/step - accuracy: 0.7653 - loss: 1.0068 - val_accuracy: 0.7464 - val_loss: 0.8966
Epoch 21/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 228s 788ms/step - accuracy: 0.7511 - loss: 1.1403 - val_accuracy: 0.8036 - val_loss: 0.6191
Epoch 22/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 231s 799ms/step - accuracy: 0.7924 - loss: 0.7186 - val_accuracy: 0.6667 - val_loss: 2.5972
Epoch 23/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 227s 787ms/step - accuracy: 0.7319 - loss: 1.6747 - val_accuracy: 0.6459 - val_loss: 3.1597
Epoch 24/50
289/289 ━━━━━━━━━━━━━━━━━━━━ 231s 800ms/step - accuracy: 0.7601 - loss: 1.1985 - val_accuracy: 0.7937 - val_loss: 0.7190
73/73 ━━━━━━━━━━━━━━━━━━━━ 46s 627ms/step - accuracy: 0.8149 - loss: 0.5529
Acurácia: 0.8019072413444519
73/73 ━━━━━━━━━━━━━━━━━━━━ 47s 629ms/step
```

Figure 11: Classificação ResNet50 - Binário.

5.4 ResNet50 - 6 classes

Tempo de execução: 2:47

5.4.1 Matrix de confusão

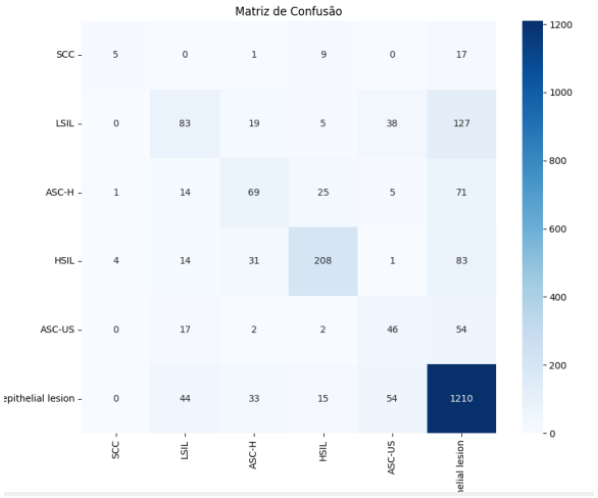

Figure 12: Matrix Confusão ResNet50 - 6 classes.

5.4.2 Acurácia

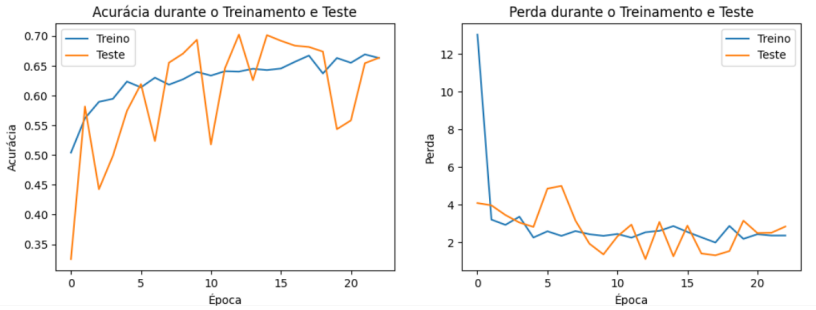

Figure 13: Acurácia ResNet50 - 6 classes.

5.4.3 Classificação

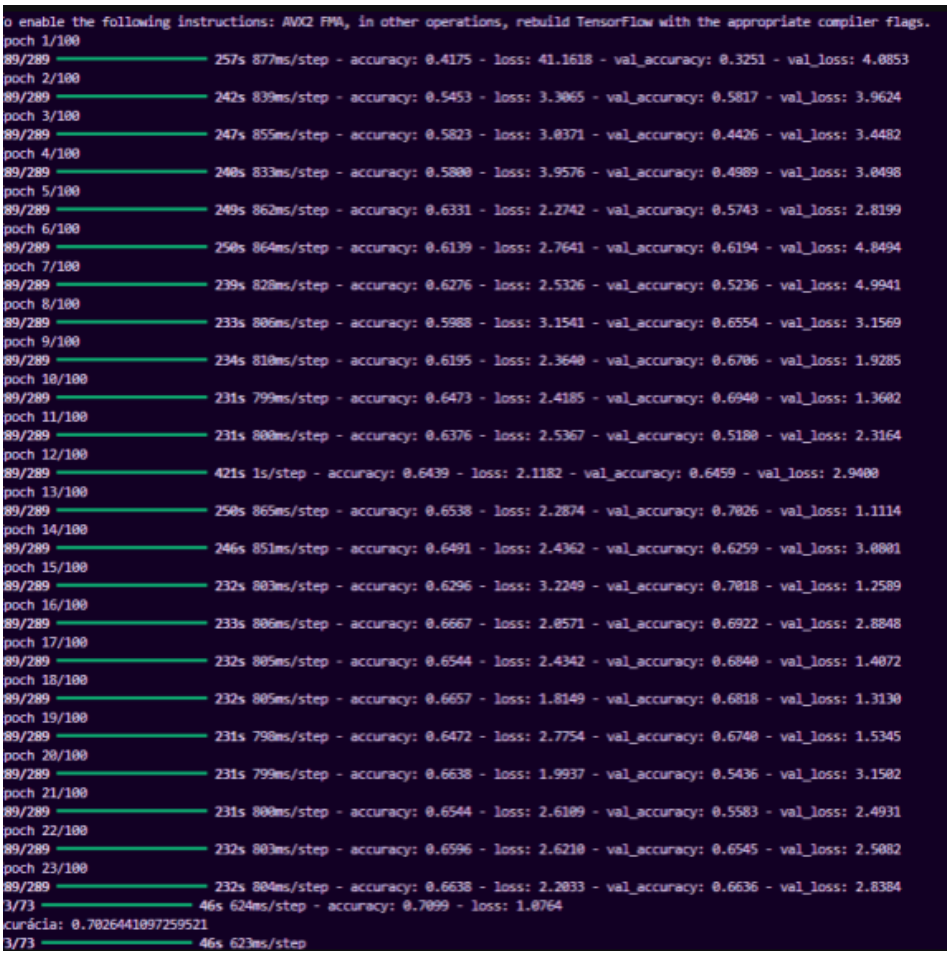

Figure 14: Classificação ResNet50 - 6 classes.

6 Conclusão

O exame de Papanicolau é uma ferramenta crucial na identificação de alterações celulares no colo do útero, sendo o principal método para a detecção precoce de lesões precursoras do câncer cervical. Neste projeto, desenvolvemos um programa utilizando a linguagem de programação Python para analisar imagens de exames de Papanicolau e permitir o reconhecimento automático de células cancerosas. Nesse trabalho pudemos consolidar conhecimento na area de processamento e análise de imagens, tais como a implementação prática de uma SVM e ResNet50, que agregou muito valor e conhecimento ao grupo, além de destacar nobreza deste trabalho na area de pesquisas e identificação do câncer, que é uma doença que a humanidade enfrenta a seculos, e precisa ser identificada quanto antes para o correto tratamento prévio, aumentando a chance de vida do paciente que começa a profilaxia antes da doença apresentar seus sintomas iniciais.

References

https://www.sciencedirect.com/science/article/abs/pii/S0925231220309723
https://ieeexplore.ieee.org/document/10425807
https://arxiv.org/html/2405.01600v1
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10028025/
http://www.decom.ufop.br/prof/marcone/projects/ppm676-17/Applsci-2021-CervicalCancerClassification.pdf