Paralelização e implementação na nuvem de algoritmos de detecção de lesões na substância branca do cérebro

Yan Vianna Sym, Mariana Pinheiro Bento, Letícia Rittner
School of Electrical and Computer Engineering, University of Campinas, São Paulo
Email: yan.vsym1@gmail.com, marianapbento@gmail.com, lrittner@dca.fee.unicamp.br

***Resumo*—A extração de dados partir de imagens é um processo demorado e exigente computacionalmente, muitas vezes necessitando de otimizações específicas e diversos processadores. Quando o conjunto de dados é muito grande e possui conteúdo sensível, surgem diversos desafios em relação ao armazenamento, transferência, consistência e privacidade das informações. Este trabalho propõe-se a estudar, reproduzir e melhorar métodos de identificação e segmentação de lesões na substância branca do cérebro humano. São utilizados algoritmos de processamento paralelo e computação na nuvem com o objetivo de reduzir o tempo necessário para processar, transferir e armazenar os dados sem prejudicar a segurança das informações.**

***Abstract*—Extracting data from images is a time-consuming and computationally demanding process, often requiring specific optimizations and multiple processors. When the dataset is very large and has sensitive content, several challenges arise regarding the storage, transfer, consistency, and privacy of information. This project proposes to study, reproduce and improve methods of identification and segmentation of lesions in the white matter of human brain. Parallel processing and cloud computing algorithms are used aiming at reducing the time needed to process, transfer and store the data without jeopardizing information security.**

I. Introdução

Lesões na substância branca do cérebro (*white matter lesions*, ou WMLs) podem causar deficit mentais significativos e estão frequentemente associadas a doenças neurológicas. Sua ocorrência aumenta conforme a idade, elevação sistólica na pressão sanguínea e aterosclerose. A análise quantitativa dessas lesões costuma ser realizada manualmente por médicos em imagens de ressonância magnética (MRI), e muitas vezes ela representa uma tarefa demorada e subjetiva, devido a variações de tamanho, forma e localização das lesões [1].

Para superar esses problemas, nos últimos anos foram propostos diversos métodos automatizados para estudar WMLs, tendo como objetivo auxiliar no diagnóstico e na monitorização das lesões ao reduzir a subjetividade do procedimento. O método proposto por Bento *et al* ([1], [2]) automaticamente detecta e segmenta WMLs utilizando algoritmos de classificação baseados em análise bidimensional de textura e aprendizagem supervisionada [3].

A extração de dados a partir de um conjunto significativo de imagens é um processo demorado e que demanda muitos recursos computacionais. Muitas vezes são necessárias otimizações específicas do contexto para que o objetivo seja alcançado em um tempo satisfatório. Quando o conjunto de dados envolvido é muito grande e possui conteúdo sigiloso, surgem diversos desafios com relação ao armazenamento, transferência, consistência e privacidade das informações.

Este trabalho propõe-se a estudar, reproduzir e aprimorar o método de identificação e segmentação de WMLs do cérebro humano proposto por Bento *et al* ([1], [2]). Serão estudados e aplicados algoritmos de processamento paralelo e computação na nuvem com o propósito de minimizar o tempo necessário para processar os dados utilizados sem prejudicar a segurança das informações [4].

O método estudado extrai das imagens um conjunto de atributos de interesse que são utilizados tanto para o treinamento dos classificadores quanto para a detecção de lesões. Esses atributos são utilizados no processo de aprendizagem para treinar classificadores responsáveis por identificar e segmentar WMLs no cérebro humano. Para avaliar o quanto as predições se aproximam das imagens originais são utilizados indicadores estatísticos e também o coeficiente Dice.

II. Metodologia

O desenvolvimento deste projeto foi dividido em cinco etapas sequenciais, tendo início com a preparação da base de imagens e finalizado com a otimização do método.

A. Preparação da base de imagens

Para compor o banco de dados do projeto foram utilizadas imagens do cérebro de 28 pacientes. Elas foram disponibilizadas no desafio ISLES (*Ischemic Stroke Lesion Segmentation Challenge*) [5] que aconteceu no MICCAI-15 (*Medical Image Computing and Computer Assisted Intervention*) [6]. Para cada paciente foram disponibilizadas duas imagens tridimensionais de ressonância magnética do cérebro de modalidades distintas; uma imagem FLAIR (*fluid-attenuated inversion recovery*) [7] e outra imagem de ressonância ponderada em T1, obtidas utilizando um scanner 3T (Magnetom Trio; Siemens). Todas as imagens disponibilizadas são compostas de 153 fatias de 230 por 230 pixels cada e foram segmentadas manualmente por um profissional da área (Fig. 1).

Também está disponibilizada no banco de dados do MICCAI-15 uma máscara binária demarcando as regiões com WMLs, gerada através da segmentação manual por profissionais. Além dos três tipos de imagens mencionados, foi gerada pelo programa Cerebra [8] um quarto tipo de imagem

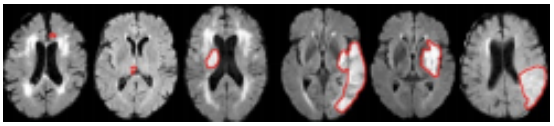

Figura 1. Fatias do cérebro de seis pacientes do MICCAI-15, onde as regiões com WMLs foram manualmente delineadas em vermelho.

fundamental para este trabalho, composto de máscaras binárias que demarcam as regiões com substância branca no cérebro do paciente (regiões *white matter*, ou WM) (Fig. 2).

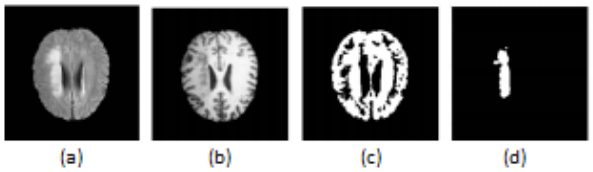

Figura 2. Fatia de um exame de MRI para um paciente do conjunto (a) modalidade FLAIR (b) modalidade T1 (c) Máscara binária demarcando (em branco) as regiões com WM (d) Máscara binária demarcando (em branco) regiões com WMLs.

B. Preparação do ambiente de desenvolvimento

Após obter todas as imagens do MICCAI-15, foi realizado um estudo para determinar qual é o melhor serviço para armazenar e acessar as imagens de forma segura. Levando em conta restrições de memória, velocidade de processamento e custo por tempo de utilização, optou-se por utilizar computação na nuvem para o armazenamento das imagens.

Dentre os serviços mais utilizados para computação na nuvem estão Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform e IBM Cloud. Para este trabalho foi adotado o Amazon Elastic Compute Cloud (Amazon EC2), um serviço web que disponibiliza capacidade computacional segura e redimensionável, além de um armazenamento altamente personalizável com custos acessíveis.

Foi utilizada a instância Ubuntu_anaconda de 64 bits, que já possui diversos módulos instalados. O modelo escolhido foi o c3.8xlarge, que possui 32 CPU (*central processing unit*), 60 GiB[1] de memória RAM e processadores Intel Xeon E5-2680 v2 (Ivy Bridge) de alta frequência, recomendado para aplicações que possuem grandes quantidades de dados e necessitam de um alto desempenho computacional.

A comunicação entre o cliente e o servidor da Amazon instanciado foi realizada utilizando SSH (*secure shell*), um protocolo de rede criptográfico para operação de serviços de rede de forma segura sobre uma rede insegura. As imagens e os arquivos executáveis foram transferidos para o servidor utilizando SFTP (*secure file transfer protocol*).

Por fim, foram importados no servidor todas as bibliotecas e módulos necessários para o projeto. Foi utilizado o pacote NumPy para programação científica e a biblioteca Scikit-learn para aprendizado de máquina.

[1]O GibiByte é uma unidade de medida para armazenamento eletrônico de informação, designando 2^{30} bytes de memória.

C. Reprodução do método

Nesta etapa do trabalho foram reproduzidas todas as seis etapas do método de detecção de WMLs (Fig. 3).

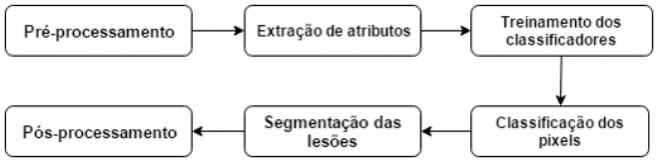

Figura 3. Fluxograma com as seis etapas do método de detecção de WMLs.

Durante a etapa de pré-processamento, todas as imagens são normalizadas e redimensionadas. Em seguida, para cada pixel da região WM, o método extrai sete atributos de textura para treinar os classificadores e conseguir identificar WMLs. Eles são: intensidade, intensidade do pixel imediatamente abaixo (fatia inferior), intensidade do pixel imediatamente acima (fatia superior), intensidade média da fatia, gradiente, gradiente morfológico e padrão binário local.

No pós-processamento das imagens são realizados filtros com base no conhecimento empírico sobre as lesões. São removidos grãos (isto é, componentes ligados em primeiro plano) e poros (isto é, componentes ligados em segundo plano) das imagens, de forma que o resultado desta etapa final sejam regiões WML segmentadas fatia a fatia.

D. Escolha dos classificadores

Na quarta etapa foram estudados quais são os melhores algoritmos de aprendizado indutivo para alcançar os objetivos do projeto. Os atributos extraídos das imagens são utilizados como entrada no processo de treinamento supervisionado para que, na etapa de testes, os classificadores utilizados sejam capazes de inferir funções que identificam e segmentam WMLs.

Na aprendizado supervisionado, cada exemplo tanto do conjunto de treinamento quanto do conjunto de teste é um par constituído por um objeto de entrada e um objeto de saída. Para este trabalho, o objeto de entrada é um vetor de sete atributos que foram extraídos na etapa de análise de textura, enquanto o objeto de saída é um valor booleano que indica se um determinado pixel é ou não WML.

A metodologia utiliza o modelo de classificação *Support Vector Machine* (SVM) com *kernel* linear. Neste modelo os dados são representados como vetores de n dimensões e as categorias são separadas por hiperplanos de n-1 dimensões. O aprendizado é baseado em encontrar o hiperplano de margens máximas que separa objetos de classes distintas.

Tendo como objetivo aprimorar o método original de detecção de WMLs, este trabalho também adotou como modelos de classificação, além do SVM, o *Random Forest* (RF) e o *k-Nearest Neighbors* (kNN). O RF faz parte do conjunto de algoritmos baseados em conjuntos de árvores, e ele opera construindo diversas árvores de decisão. A classe resultante é determinada através do cálculo da moda do resultado de todas as árvore geradas. Já o kNN é um modelo de classificação baseado em instâncias que classifica os objetos com base em um voto majoritário entre seus k vizinhos mais próximos.

E. Otimização do método

Nesta etapa do projeto foi realizado um estudo para identificar os trechos onde a execução do método é demasiadamente lenta devido ao processamento de grandes quantidades de dados. Em seguida foi realizada uma análise com o objetivo de encontrar as melhores abordagens para tornar esses trechos mais eficientes. Através de um estudo da dependência dos dados entre as etapas do método, concluiu-se que a melhor maneira de otimizar o tempo de execução do algoritmo seria através de paralelismo. A computação paralela é a utilização simultânea de múltiplos recursos computacionais para realizar cálculos ou execuções de processos simultaneamente [9].

Com o auxílio da biblioteca joblib e os 32 processadores disponíveis no servidor da Amazon, o algoritmo foi modificado com a introdução de paralelismo. A biblioteca joblib oferece uma classe chamada Parallel que divide o interpretador Python em vários processos, permitindo a escrita de laços em paralelo. Foi adotada a estratégia de paralelizar todas as etapas do método, atribuindo a cada processador somente uma responsabilidade. Como os cálculos são independentes, ao invés de um único processador realizar todas as tarefas, cada processador fica responsável pela manipulação e gerenciamento dos dados de um único paciente.

Na etapa de pós-processamento, ao invés de tratar os ruídos de classificação sequencialmente para cada fatia analisada, o trabalho foi paralelizado de forma que cada processador ocioso fica responsável apenas por uma parte dos cálculos. Desse modo foi possível dividir um problema envolvendo grandes quantidades de dados em problemas pequenos que podem ser resolvidos ao mesmo tempo, gerando ganhos tanto em tempo de execução quanto em escalabilidade da solução.

III. Resultados

Os experimentos tiveram início após a preparação do ambiente de desenvolvimento e a reprodução do método proposto. As modificações realizadas fizeram com que o tempo de execução do método reduzisse de 2920 segundos para 560 segundos, resultando em uma melhoria de 80,8% (Fig. 4). O ganho computacional gerado pela utilização de paralelismo permitiu que os experimentos com o método fossem executados mais rapidamente. Através dos experimentos foram encontrados os hiperparâmetros e *thresholds* de classificação que otimizam os resultados do método.

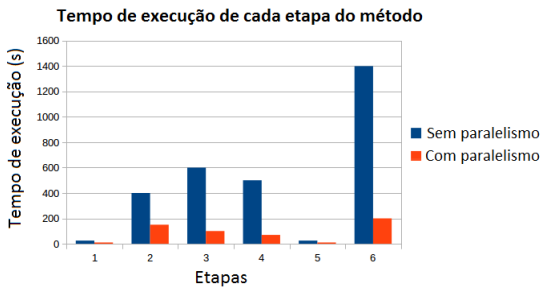

Figura 4. Tempo de execução de cada etapa do método antes (em azul) e depois (em laranja) das otimizações realizadas.

Para avaliar o quanto as predições do método se aproximam da realidade, foram utilizados a acurácia, precisão e o coeficiente Dice. A acurácia representa o grau de conformidade de um valor medido ou calculado em relação ao seu valor real ou a uma referência preestabelecida, e neste trabalho ela foi obtida através da matriz de confusão. Através da matriz de confusão foram extraídos a precisão, sensibilidade e especificidade, que também são medidas estatísticas.

O coeficiente Dice, por sua vez, é muito utilizado em estatística para comparar a similaridade entre duas amostras. Seja X uma fatia de um exame MRI contendo a região WML original segmentada por um profissional e Y uma fatia onda a região WML foi determinada através do método, então o coeficiente Dice QS entre X e Y pode ser calculado por:

$$QS = \frac{2|X \cap Y|}{|X| + |Y|}$$

Foram escolhidos aleatoriamente 18 pacientes para o conjunto de treinamento e 10 pacientes para o conjunto de teste. Ao todo foram avaliados 8518833 pixels de regiões WM, dos quais 907429 são WML. Para prevenir *overfitting* foi utilizada a técnica de validação de modelos *k-fold cross-validation* com k=10. O método foi executado utilizando cada modelo de classificação individualmente e também utilizando todos os modelos em conjunto (Tabela I). Essa técnica é denominada *bagging* [10], e ela consiste em um meta-algoritmo projetado para melhorar estabilidade do modelo de classificação.

Tabela I
Resultados obtidos quando os três modelos de classificação são utilizados individualmente e também simultaneamente através de bagging.

	SVM	RF	kNN	Bagging
Dice	0.681	0.668	0.673	0.692
Acurácia (%)	95.4	81.3	81.6	95.8
Precisão (%)	79.9	34.6	35.3	84.1
Sensibilidade (%)	75.4	85.5	87.7	74.1
Especificidade (%)	97.7	80.8	80.9	98.3

Os melhores resultados obtidos neste trabalho ocorreram quando os três classificadores foram utilizados em conjunto através do método de *bagging*, cuja predição apresentou acurácia de 95.8%, precisão de 84.1% e coeficiente Dice de 0.692 após a etapa de pós-processamento. Já os melhores resultados individuais foram apresentados pelo SVM, pois a sua acurácia, precisão e coeficiente Dice se sobressaíram consideravelmente em relação ao RF e ao kNN (Fig. 5).

Embora o RF e o kNN tenham apresentado bons resultados para o coeficiente Dice final, eles apresentaram uma acurácia muito inferior em relação ao SVM. Isso pode ser explicado pois tanto o RF quanto o kNN apresentam uma sensibilidade maior em relação ao SVM. Também é interessante notar que o pós-processamento foi muito mais eficiente para o RF e kNN do que para o SVM.

É possível comparar diretamente os resultados deste trabalho com os relacionados na literatura, pois foi utilizado o conjunto de dados público do MICCAI-15. O algoritmo que

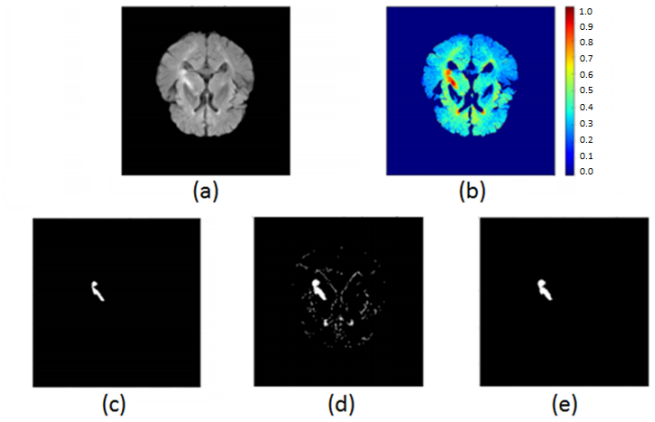

Figura 5. (a) Fatia no formato FLAIR obtida através de um exame de MRI para um determinado paciente do conjunto MICCAI-15. (b) Mapa de probabilidades de pixels WMLs para a fatia. (c) Máscara binária segmentada manualmente por um especialista, onde regiões brancas indicam lesões. (d) Predição de pixels WMLs realizada pelo classificador *SVM*. (e) Predição de pixels WMLs resultante da etapa de pós-processamento.

venceu a competição ISLES 2015 apresentou um coeficiente Dice de 0.69, que se iguala ao resultado obtido neste trabalho.

Com o propósito de observar a dispersão do coeficiente Dice entre as fatias de cada paciente do conjunto de testes, foi utilizado o Boxplot (Fig. 6). A diferença entre os resultados obtidos para cada paciente decorre principalmente da variação de tamanho e forma das lesões.

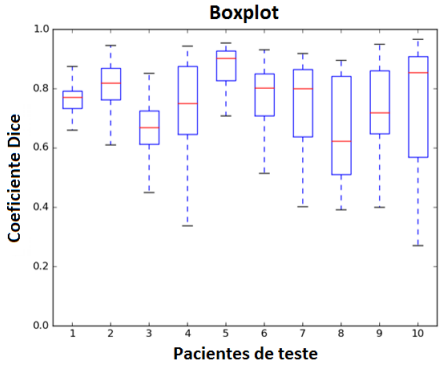

Figura 6. Boxplot do coeficiente Dice para os 10 pacientes utilizados nos experimentos deste trabalho.

IV. Conclusão

Este trabalho teve como objetivo principal estudar, reproduzir e otimizar métodos de identificação e segmentação de lesões na substância branca do cérebro humano, baseando-se no trabalho proposto por Bento *et al*([1], [2]). Foram estudados algoritmos de análise de textura, classificação de imagens, programação paralela e computação na nuvem, tendo como propósito minimizar o tempo necessário para o processamento de dados sem prejudicar a persistência e a segurança das informações.

Para o armazenamento das imagens e processamento dos dados foi utilizado o Amazon EC2, um serviço web que disponibiliza capacidade computacional segura e redimensionável na nuvem. O modelo escolhido foi o c3.8xlarge da instância Ubuntu_anaconda, que é otimizado para processamentos paralelos que necessitam de um alto desempenho computacional. Foram realizadas modificações na metodologia proposta com a adição de programação paralela, tendo como objetivo diminuir o tempo de execução do método. O resultado obtido foi um ganho de 80,8% em termos de desempenho computacional.

Quando os classificadores utilizados neste trabalho foram combinados através do método de *bagging*, foi obtida uma acurácia de 95.8%, precisão de 84.1% e coeficiente Dice de 0.692. Conforme o esperado, esse resultado foi superior ao encontrado quando os classificadores foram utilizados individualmente. A metodologia foi comparada com outros trabalhos relacionados na literatura, apresentando resultados finais melhores ou equivalentes.

Para trabalhos futuros nós planejamos realizar o processamentos de dados utilizando GPU (*graphics processing unit*), pois sua estrutura de processamento paralelo os torna muito mais eficientes do que as CPU de uso geral para problemas envolvendo grandes quantidades de dados. Nós também pretendemos utilizar algoritmos baseados em *deep learning* com o objetivo de aumentar a robustês do método.

O repositório contendo o trabalho desenvolvido foi disponibilizado em https://github.com/YanSym/TFCProject. Lá se encontram as imagens e algoritmos utilizados ao longo dos experimentos, logs de execução e arquivos de texto contendo todos os resultados obtidos.

Referências

[1] M. Bento, Y. Sym, R. Frayne, R. Lotufo, and L. Rittner, "Probabilistic segmentation of brain white matter lesions using texture-based classification," in *Image Analysis and Recognition: 14th International Conference, ICIAR 2017, Montreal, QC, Canada, July 5–7, 2017, Proceedings*, vol. 10317. Springer, 2017, p. 71.

[2] M. Leite, L. Rittner, S. Appenzeller, H. H. Ruocco, and R. Lotufo, "Etiology-based classification of brain white matter hyperintensity on magnetic resonance imaging," *Journal of Medical Imaging*, vol. 2, no. 1, pp. 014 002–014 002, 2015.

[3] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Gray scale and rotation invariant texture classification with local binary patterns," *Computer Vision-ECCV 2000*, pp. 404–420, 2000.

[4] J. Ekanayake and G. Fox, "High performance parallel computing with clouds and cloud technologies," in *International Conference on Cloud Computing*. Springer, 2009, pp. 20–38.

[5] O. Maier, B. H. Menze, J. von der Gablentz, L. Häni, M. P. Heinrich, M. Liebrand, S. Winzeck, A. Basit, P. Bentley, L. Chen *et al.*, "Isles 2015-a public evaluation benchmark for ischemic stroke lesion segmentation from multispectral mri," *Medical image analysis*, vol. 35, pp. 250–269, 2017.

[6] N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings*. Springer, 2015, vol. 9351.

[7] R. Ashikaga, Y. Araki, and O. Ishida, "Mri of head injury using flair," *Neuroradiology*, vol. 39, no. 4, pp. 239–242, 1997.

[8] Q. Lu, D. Gobbi, R. Frayne, and M. Salluzzi, "Cerebra-wml: A standalone application for quantification of white matter lesion," *Imaging Network Ontario Sym*, vol. 15, 2014.

[9] M. J. Quinn, *Parallel computing: theory and practice*. McGraw-Hill, Inc., 1994.

[10] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.