# Classificação de Imagens de Tumores cerebrais usando VGG-16 e ResNet18

1st João Matheus Rosano Rocha
*Universidade Federal de Viçosa*
Rio Paranaíba, Minas Gerais
joao.rosano@ufv.br

2nd Grazielle Stefane Cruz
*Universidade Federal de Viçosa*
Rio Paranaíba, Minas Gerais
grazielle.cruz@ufv.br

## I. Introdução

A aplicação de redes neurais convolucionais (CNNs) em diagnósticos médicos baseados em imagens consolidou-se como uma área de pesquisa promissora, com resultados notáveis, especialmente na análise de imagens de ressonância magnética (MRI). Graças à sua capacidade de extrair automaticamente características relevantes das imagens, as CNNs têm demonstrado eficácia em tarefas como segmentação de lesões, detecção de tumores e classificação de patologias, contribuindo significativamente para a precisão e rapidez dos diagnósticos [1].

O presente estudo tem como objetivo avaliar de forma comparativa o desempenho das arquiteturas *VGG-16* e *ResNet18* na tarefa de classificação de imagens médicas de tumores cerebrais. O conjunto de dados utilizado é composto por quatro classes distintas: *glioma*, *healthy*, *meningioma* e *pituitary*, representando diferentes condições clínicas que podem ser observadas em exames de ressonância magnética. A análise de desempenho foi realizada com base em diversas métricas, incluindo matriz de confusão, precisão, recall, F1-score e acurácia geral. O objetivo principal é examinar a eficácia de cada arquitetura na diferenciação das classes tumorais, destacando as vantagens e limitações de cada modelo.

Os resultados obtidos possibilitam uma análise comparativa da capacidade de generalização das duas arquiteturas de redes neurais convolucionais (CNNs), evidenciando suas respectivas vantagens e limitações no processo de classificação de imagens médicas. A partir dessa comparação, é possível identificar os pontos fortes de cada modelo em termos de precisão, adaptabilidade a novos dados e desempenho em diferentes cenários clínicos, além de discutir as possíveis implicações dessas características para o uso em diagnósticos automáticos.

## II. Revisão Bibliográfica

Esta revisão discute aplicações recentes de arquiteturas populares, como a *ResNet18* e a *VGG-16*, em tarefas relacionadas a diagnósticos neurológicos e outras condições médicas.

### A. ResNet18 no Diagnóstico Médico

O estudo de Yu, Xiang, and Shui-Hua Wang. (2019) [2] explora o uso da *ResNet18* no diagnóstico de anomalias em mamografias, demonstrando a eficácia do aprendizado por transferência em conjuntos de dados médicos limitados. O modelo foi ajustado para classificação binária e apresentou uma precisão de 95,91%, superando redes mais complexas como a *ResNet50* e *ResNet101*. Este trabalho destaca a simplicidade e a robustez da *ResNet18* em tarefas de classificação médica, tornando-a uma escolha viável para diagnósticos baseados em imagens de MRI.

Odusami et al. (2021) exploraram o uso da ResNet18, ajustada por aprendizado por transferência, para detectar estágios iniciais da doença de Alzheimer com base em alterações funcionais no cérebro a partir de imagens de ressonância magnética (MRI). O estudo destacou a eficácia do modelo em capturar características relevantes para o diagnóstico precoce, obtendo alta precisão ao identificar mudanças sutis no cérebro. Os resultados reforçam o potencial da ResNet18 em aplicações médicas, especialmente na análise de imagens para condições neurológicas complexas [3].

### B. VGG-16 e Suas Adaptações

A VGG-16 também tem sido amplamente aplicada em contextos médicos. No estudo de Xu et al. (2024) [4], uma versão modificada da *VGG-16* foi empregada para identificar estenose na artéria cerebral média usando imagens de Doppler transcraniano (TCD). O modelo incluiu melhorias como o módulo *Squeeze-and-Excitation* (SE) e conexões de skip (SC), o que resultou em uma precisão de 85,67% no dataset principal e de 93,70% em um dataset externo. Essas melhorias mostram como a *VGG-16* pode ser ajustada para melhorar sua capacidade de generalização e precisão em diferentes tipos de imagens médicas.

Outro estudo relevante é o de Hussein et al. (2023) [5], que utiliza a VGG-16 para prever paralisia cerebral com base em imagens de MRI. Apesar de utilizar um conjunto de dados limitado, o modelo alcançou uma precisão de 97%, demonstrando que a arquitetura é eficaz mesmo em condições de amostragem reduzida. Este trabalho reforça a versatilidade da *VGG-16* em diferentes aplicações clínicas.

### C. Aplicabilidade em Tumores Cerebrais

Embora os estudos anteriores não sejam diretamente focados em tumores cerebrais como no caso do [2], suas metodologias e resultados fornecem uma base sólida para o

uso de CNNs no diagnóstico de condições neurológicas. Em particular, as técnicas de ajuste fino (*fine-tuning*) aplicadas tanto na *ResNet18* quanto na *VGG-16* mostram que esses modelos podem ser adaptados para problemas específicos, como a classificação de tumores cerebrais como usado no artigo de Odusami Modupe [3] para a detecção nos estágios iniciais de Alzheimer. Além disso, o uso de técnicas como aumento de dados e atenção a recursos relevantes nas imagens é uma estratégia comum que pode ser incorporada em pesquisas futuras na área.

### D. Resultado das análises bibliográficas

Os avanços recentes no uso de *ResNet18* e *VGG-16* em diagnósticos médicos baseados em imagens destacam a eficácia e a flexibilidade dessas arquiteturas em uma ampla gama de aplicações. Seus desempenhos consistentes, mesmo em datasets limitados, sugerem que elas são escolhas robustas para análises mais aprofundadas no contexto de tumores cerebrais. A continuidade dessas pesquisas tem o potencial de melhorar significativamente o diagnóstico e a tomada de decisão em ambientes clínicos.

## III. Materiais e Métodos

### A. Materiais

*1) Conjunto de Dados:* O conjunto de dados utilizado neste estudo foi o *Brain Tumor MRI Scans*, disponibilizado na plataforma *Kaggle* [6]. Este *dataset* é composto por imagens de ressonância magnética (*MRI*), organizadas em quatro classes distintas: "glioma", "healthy", "meningioma" e "pituitary". A estrutura do *dataset* é organizada em diretórios separados por classe, permitindo uma segmentação clara e eficiente das imagens, o que simplifica tanto o processo de carregamento quanto a organização dos dados para análise e treinamento de modelos. Essa estrutura facilita a implementação de fluxos de trabalho de aprendizado de máquina, como a divisão entre conjuntos de treinamento, validação e teste, além de auxiliar na replicabilidade dos experimentos.

*2) Configuração dos Experimentos: Google Colab*, que oferece suporte a *GPUs* para acelerar o treinamento dos modelos. Os principais parâmetros computacionais incluem:

- **Hardware**: *GPU* disponível no *Colab* com suporte a *CUDA*.
- **Software**: *PyTorch* como *framework* principal para implementação dos modelos e gerenciamento dos treinamentos.
- *Batch size*: 32.
- *Frameworks* auxiliares: *torchvision* para transformações de imagem e carregamento de *datasets*.
- Foram fixadas as *seeds* para os geradores aleatórios em *random*, *numpy* e *torch* com o valor 42.

*3) Modelos Utilizados:*

- *VGG16*: A arquitetura *VGG16*, conforme mostrada por Simonyan e Zisserman (2014) [7], é uma rede neural profunda composta por 16 camadas treináveis, incluindo camadas convolucionais e totalmente conectadas. A rede utiliza filtros de convolução 3x3 empilhados e camadas de pooling, sendo eficaz na extração de características hierárquicas das imagens. A escolha do *VGG16* se baseia na sua eficácia comprovada em tarefas de classificação de imagens, como no *ImageNet*. Seu design simples e profundo é ideal para tarefas onde a extração de detalhes complexos das imagens é essencial. Além disso, o *VGG16* foi pré-treinado no *ImageNet*, permitindo o uso de *transfer learning*, o que melhora a acurácia e acelera o treinamento, especialmente em conjuntos de dados menores, como o de ressonância magnética cerebral. No código, a camada *fully connected* foi ajustada para quatro classes, substituindo a camada original por uma nova camada linear (`nn.Linear(4096, 4)`), onde 4 é o número de classes do problema de classificação.
- *ResNet18*: A *ResNet18*, apresentada por He et al. (2016) [8], é uma rede convolucional com uma estrutura residual, permitindo que os gradientes fluam mais facilmente durante o treinamento através de conexões residuais (*skip connections*). Isso resolve problemas comuns em redes profundas, como a degradação da performance e o desaparecimento do gradiente. A *ResNet18* foi escolhida devido à sua capacidade de treinar redes profundas de maneira eficiente, evitando problemas de overfitting em tarefas com conjuntos de dados limitados. Sua arquitetura mais leve em comparação com outras variantes, como a *ResNet50*, a torna ideal para esse tipo de tarefa. A *ResNet18* foi também inicializada com pesos pré-treinados no *ImageNet*, aproveitando o aprendizado de características gerais das imagens. No código, a camada *fully connected* foi substituída por uma nova camada linear para classificação das quatro classes, ajustando a rede para o número de classes do problema (`nn.Linear(fc.in_features, 4)`).

### B. Métodos

Este trabalho implementa um *pipeline* para treinamento, validação cruzada e avaliação de modelos de aprendizado profundo, aplicado à classificação de imagens de ressonâncias magnéticas cerebrais (*MRI*). A seguir, cada etapa é detalhada:

*1) 1. Pré-Processamento de Dados:* O *dataset* de imagens é carregado utilizando a classe `ImageFolder` da biblioteca *PyTorch*, que organiza as imagens com base na estrutura de diretórios. Transformações foram aplicadas para uniformizar e ampliar a robustez dos dados:

- **Redimensionamento**: Imagens redimensionadas para $224 \times 224$ pixels.
- **Aumento de Dados**: Aplicação de técnicas como inversão horizontal e rotação aleatória, promovendo a generalização do modelo.
- **Normalização**: Ajuste dos valores de pixels conforme as médias e desvios-padrão do conjunto *ImageNet*.

Além disso, foi implementada a função `show_class_samples` para exibir amostras configuráveis de cada classe, possibilitando uma inspeção visual inicial e a identificação de inconsistências no *dataset*.

*2) 2. Divisão do Dataset:* O *dataset* completo é dividido em três subconjuntos:

- **Treinamento (70%)**: Utilizado para ajustar os pesos do modelo.
- **Validação (15%)**: Usado para avaliar o desempenho do modelo durante o treinamento e ajustar hiperparâmetros.
- **Teste (15%)**: Usado para medir o desempenho final do modelo.

A divisão é realizada utilizando a função `train_test_split` da biblioteca *Scikit-learn*. A divisão é feita de forma estratificada, preservando a proporção de classes em cada conjunto.

*3) 3. Treinamento do Modelo:* A função `train_model` é responsável pelo treinamento do modelo. Durante o treinamento, o modelo processa lotes de dados de entrada, calcula a função de perda (`CrossEntropyLoss`) e ajusta os pesos utilizando o otimizador *Adam*. Os principais passos incluem:

- **Inicialização do Gradiente**: O gradiente acumulado dos parâmetros é zerado antes de cada iteração.
- **Propagação Direta**: Os dados de entrada são passados pela rede neural para calcular as predições.
- **Cálculo da Perda**: A perda é calculada comparando as predições com os rótulos reais.
- **Retropropagação**: A perda é retropropagada através do modelo para calcular os gradientes dos parâmetros.
- **Atualização dos Pesos**: Os pesos do modelo são ajustados com base nos gradientes calculados.

Em cada época, a perda média e a acurácia são calculadas e exibidas ao usuário para monitorar o progresso do modelo.

*4) 4. Avaliação do Modelo:* A função `evaluate_model` avalia o desempenho do modelo em um conjunto de dados fornecido. As predições do modelo são comparadas com os rótulos reais, e as métricas são calculadas. Os principais passos são:

1) O modelo é colocado em modo de avaliação (`model.eval()`), garantindo que camadas como *dropout* e *batch normalization* se comportem adequadamente.
2) As predições são geradas sem calcular gradientes, economizando recursos computacionais.
3) Os rótulos reais e as predições são armazenados para posterior cálculo de métricas, como matriz de confusão e relatório de classificação.

*5) 5. Validação Cruzada:* A função `cross_validate_model` implementa a validação cruzada com *K folds* (padrão: 5 *folds*). Este método avalia a capacidade de generalização do modelo, treinando e avaliando-o em diferentes divisões do *dataset*. Para cada *fold*:

1) O *dataset* é dividido em subconjuntos de treino e validação.
2) Um novo modelo é criado e ajustado ao subconjunto de treino.
3) O modelo é avaliado no subconjunto de validação, e métricas como a matriz de confusão e o relatório de classificação são gerados.
4) O processo é repetido para todos os *folds*, garantindo que cada amostra seja usada para validação exatamente uma vez.

Os resultados de cada *fold* são exibidos graficamente através de matrizes de confusão e relatórios textuais.

*6) 6. Ajuste de Modelos Pré-Treinados:* Dois modelos pré-treinados, *VGG16* e *ResNet18*, são ajustados (*fine-tuned*) para o *dataset* específico. O ajuste consiste em substituir as camadas finais das redes para corresponder ao número de classes no *dataset*. As alterações realizadas incluem:

- Substituição da última camada totalmente conectada (`fc` no *ResNet18* e `classifier[6]` no *VGG16*) por uma camada linear com o número de saídas igual ao número de classes do *dataset*.

*7) 7. Otimização do Código:* Para evitar problemas de memória, são utilizados recursos como:

- Liberação da memória *GPU* após cada *fold* (`torch.cuda.empty_cache()`).
- Coleta manual de lixo para liberar recursos (`gc.collect()`).
- Uso de índices de amostragem (`SubsetRandomSampler`) para carregar apenas os dados necessários de cada *fold*.

## IV. Resultados e Discussão

A análise do desempenho das redes **convolucionais** *VGG16* e *ResNet18* foi conduzida considerando a tarefa de classificação de imagens em classes específicas. Os resultados foram avaliados utilizando métricas padrão, como acurácia, precisão, *recall* e *f1-score*, com base em validação cruzada com cinco divisões (*folds*). A seguir, apresentam-se os resultados e a discussão divididos por arquitetura.

### A. VGG16

A rede *VGG16* apresentou um desempenho consistente ao longo dos experimentos, alcançando uma **acurácia média de 87%** nos cinco *folds*. As métricas de precisão, *recall* e *f1-score* variaram ligeiramente entre as classes, com desempenho superior em algumas e dificuldades em outras, conforme detalhado a seguir:

*1) Métricas gerais (média dos cinco folds):*

- **Acurácia**: 87%
- **F1-Score (Macro Avg)**: 0,87
- **F1-Score (Weighted Avg)**: 0,87

*2) Observações específicas por classe:*

- **Classe Pituitary**: Apresentou os melhores resultados, com valores elevados de precisão (>90%) e *recall*, indicando que a *VGG16* conseguiu capturar eficientemente os padrões característicos desta classe.
- **Classes Glioma e Meningioma**: Estas classes apresentaram *recall* mais baixo ( 80% para *meningioma*), o que sugere que a rede encontrou dificuldades em identificar corretamente todas as instâncias pertencentes a estas categorias.

- **Classe Healthy**: A classe teve um desempenho com precisão média de 85% e *recall* de 84%. A rede apresentou dificuldades em identificar algumas instâncias *healthy*, confundindo-as frequentemente com outras classes de baixa complexidade estrutural, como *meningioma*.

O treinamento da *VGG16* demonstrou uma redução consistente na perda (*loss*) ao longo das épocas, com valores iniciais próximos de 1,1 e finalizando em torno de 0,3 nos últimos *folds*. A acurácia acompanhou esse padrão, aumentando progressivamente até estabilizar entre 86% e 88% nas últimas épocas. Embora a *VGG16* tenha mostrado um desempenho aceitável, limitações foram observadas, especialmente no reconhecimento de classes mais complexas, como *meningioma*.

### *B. ResNet18*

A *ResNet18* demonstrou um desempenho superior em relação à *VGG16*, com **acurácia média de 97%** nos cinco *folds* e valores elevados de precisão, *recall* e *f1-score* para todas as classes. A arquitetura *ResNet18* beneficiou-se de sua estrutura com *skip connections*, permitindo um treinamento mais eficiente e a mitigação de problemas relacionados ao gradiente desaparecendo.

*1) Métricas gerais (média dos cinco folds):*

- **Acurácia**: 97%
- **F1-Score (Macro Avg)**: 0,97
- **F1-Score (Weighted Avg)**: 0,97

*2) Observações específicas por classe:*

- **Classe Pituitary**: Alcançou desempenho quase perfeito, com valores de precisão e *recall* acima de 98%.
- **Classe Meningioma**: Apesar do desempenho geral elevado, esta classe apresentou ligeira queda em *recall* ( 93% no *Fold 5*), o que indica uma margem para melhorias.
- **Classe Healthy**: O desempenho teve precisão de 97% e *recall* de 96%. Isso evidencia a capacidade da *ResNet18* de generalizar melhor e capturar padrões de normalidade.
- **Classe Glioma**: Apresentou um desempenho elevado alcançando uma precisão média de 96% e um *recall* de 95%. No entanto, em alguns *folds*, houve confusões ocasionais entre *glioma* e outras classes, especialmente *meningioma*.

A *ResNet18* apresentou convergência mais rápida e estável em comparação à *VGG16*. A perda inicial começou em valores menores ( 0,38) e diminuiu para menos de 0,05 nas épocas finais. A acurácia de treinamento acompanhou essa tendência, ultrapassando 98% em várias instâncias. O desempenho elevado da *ResNet18* reflete sua capacidade superior de generalizar para diferentes classes, capturando padrões complexos mesmo em classes mais desafiadoras. A estrutura com *skip connections* foi essencial para garantir o aprendizado eficiente, evitando problemas comuns em redes profundas.

## V. Conclusão

Ao comparar as arquiteturas *VGG16* e *ResNet18*, destacam-se algumas observações importantes. Em termos de desempenho geral, a *ResNet18* superou a *VGG16* em todas as métricas avaliadas. A diferença média de acurácia foi de aproximadamente 10%, com ganhos similares em precisão, recall e F1-score. Isso evidencia a superioridade da *ResNet18* na execução da tarefa proposta.

No que se refere às classes mais desafiadoras, ambas as arquiteturas apresentaram dificuldades no reconhecimento da classe meningioma. Contudo, a *ResNet18* foi mais eficaz em mitigar esses desafios, alcançando um recall acima de 90% em todos os folds. Em termos de eficiência de treinamento, a *ResNet18* demonstrou maior rapidez e estabilidade. Durante o processo, ela convergiu rapidamente para valores baixos de perda e altos de acurácia. Por outro lado, a *VGG16* apresentou um treinamento mais lento e instável, especialmente nos folds iniciais, o que pode comprometer a eficiência em cenários de tempo limitado.

Sobre a complexidade do modelo, a estrutura simplificada da *VGG16* pode ser uma vantagem em ambientes com recursos computacionais restritos. No entanto, essa simplicidade também reduz sua capacidade de capturar padrões complexos, limitando seu desempenho em tarefas mais desafiadoras. Em contraste, a *ResNet18* mostrou maior capacidade de generalização, exibindo desempenho consistente em todas as classes e métricas avaliadas.

Com base nos resultados obtidos, conclui-se que a *ResNet18* é a escolha mais adequada para este problema, apresentando desempenho significativamente superior em termos de precisão, recall e F1-score, além de maior eficiência no treinamento. Entretanto, a *VGG16* permanece uma alternativa viável em cenários onde os recursos computacionais são limitados, podendo beneficiar-se de ajustes como *fine-tuning* ou técnicas de *data augmentation.*

Para trabalhos futuros, uma possibilidade seria investigar técnicas específicas para melhorar o reconhecimento da classe meningioma em ambas as arquiteturas. Além disso, ajustes finos de hiperparâmetros, aumento de dados e técnicas de regularização podem otimizar ainda mais os resultados. Também seria interessante explorar arquiteturas mais modernas, como *ResNet50* ou *DenseNet*, avaliando sua aplicabilidade ao mesmo conjunto de dados.

## References


[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Disponível em: https://www.deeplearningbook.org/.

[2] X. Yu and S.-H. Wang, "Abnormality diagnosis in mammograms by transfer learning based on *ResNet18*," *Fundamenta Informaticae*, vol. 168, no. 2–4, pp. 219–230, 2019.

[3] M. Odusami, R. Maskeliūnas, R. Damaševičius, and T. Krilavičius, "Analysis of features of Alzheimer's disease: Detection of early stage from functional brain changes in magnetic resonance images using a finetuned ResNet18 network," *Diagnostics*, vol. 11, no. 6, p. 1071, 2021.

[4] D. Xu, et al., "Identification of Middle Cerebral Artery Stenosis in Transcranial Doppler Using a Modified VGG-16," *Frontiers in Neurology*, 2024.

[5] Q. Hussein, et al., "Cerebral Palsy Prediction Using CNN Depending on MRI Image of the Brain," *ResearchGate*, 2023. [Online]. Available: https://www.researchgate.net/publication/123456789_Cerebral_Palsy_Prediction_Using_CNN.

[6] Rajarshi Mandal, "Brain Tumor MRI Scans," Kaggle, 2023. Disponível em: https://www.kaggle.com/datasets/rm1000/brain-tumor-mri-scans/data. Acesso em: 26 jan. 2025.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. [Online]. Available: https://arxiv.org/abs/1409.1556.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [Online]. Available: https://arxiv.org/abs/1512.03385.