# Análise Crítica da Eficiência da MobileNetV2 em Classificação de Fauna em Ambientes com Recursos Limitados

1st Ricardo Jeremias Prata Filho
*Campus Rio Paranaíba (CRP)*
*Universidade Federal de Viçosa (UFV)*
Rio Paranaíba-MG, Brasil
ricardo.prata@ufv.br

2nd João Fernando Mari
*Campus Rio Paranaíba (CRP)*
*Universidade Federal de Viçosa (UFV)*
Rio Paranaíba-MG, Brasil
joaof.mari@ufv.br

***Abstract*—This paper evaluates the performance and computational trade-offs of using the MobileNetV2 architecture for animal classification. Utilizing Transfer Learning on the Animals-10 dataset, we achieved a validation accuracy of 95.9% after 10 epochs. Beyond metrics, this study critically analyzes the impact of batch processing via ImageDataGenerator for memory management in constrained environments (Google Colab) and discusses the diminishing returns observed between 5 and 10 epochs. We argue that lightweight architectures are not merely a compromise but an optimal choice for real-time deployment, despite the availability of deeper, more resource-intensive networks.**

***Index Terms*—Deep Learning, MobileNetV2, Transfer Learning, Resource Constraints, ImageDataGenerator.**

## I. Introdução

A classificação automática de imagens tem se consolidado como uma das aplicações mais relevantes da visão computacional moderna, sustentando desde sistemas de monitoramento ambiental até plataformas embarcadas de detecção em tempo real. Em particular, o reconhecimento de animais tem ganhado atenção na literatura por apoiar estudos ecológicos, estimativas populacionais e estratégias de conservação, principalmente em áreas remotas onde o uso de câmeras-trap possibilita a coleta contínua de dados sem intervenção humana direta. Ainda assim, grande parte dos trabalhos nessa área assume a disponibilidade de hardware robusto ou de pipelines altamente otimizados, o que raramente representa a realidade de pesquisadores iniciantes, instituições de pequeno porte ou ambientes educacionais.

Este estudo surge no contexto da disciplina SIN393 — Introdução à Visão Computacional — como a primeira oportunidade de experimentação prática com modelos modernos de Deep Learning. Entretanto, a motivação extrapola o âmbito didático. Buscamos investigar um problema simples, porém representativo: como aplicar técnicas de Transfer Learning em um cenário de recursos computacionais limitados sem comprometer a viabilidade do experimento? Essa questão aparece frequentemente em estudos científicos que utilizam plataformas gratuitas, como o Google Colab, ou dispositivos embarcados com restrições severas de memória e processamento. Assim, ao invés de perseguir exclusivamente acurácia máxima, este trabalho analisa criticamente o custo-benefício de utilizar arquiteturas leves — com ênfase na MobileNetV2 — em tarefas de classificação de fauna.

Durante o desenvolvimento, limitações práticas moldaram diversas decisões metodológicas. O dataset Animals-10, composto por aproximadamente 26 mil imagens, inviabilizou o uso direto de técnicas de otimização, resultando em estouro de memória RAM no ambiente do Colab. Para contornar esse problema, adotou-se a classe ImageDataGenerator não como ferramenta de Data Augmentation, mas como mecanismo de streaming e gerenciamento de lotes pequenos. Situações como kernel sendo finalizado e reorganização manual do pipeline exemplificam desafios reais enfrentados por estudantes e pesquisadores quando trabalham com modelos modernos em infraestruturas restritas.

Além do contexto computacional, este trabalho dialoga com aplicações reais da classificação animal. Modelos compactos como a MobileNetV2 têm sido empregados em câmeras-trap, drones e sistemas embarcados pela sua eficiência energética e rapidez de inferência. Para análises ecológicas de longo prazo — como a detecção de tendências populacionais, flutuação de espécies ao longo dos anos ou variações de presença em determinadas regiões — a viabilidade de processamento local ou semi-local é essencial. Assim, ainda que o escopo desta pesquisa seja inicial, os resultados fornecem um ponto de partida sólido para experimentos futuros que envolvem arquiteturas alternativas, fine-tuning e comparações de diferentes estratégias de feature extraction.

Dessa forma, o presente estudo contribui não apenas com resultados quantitativos, mas com uma reflexão crítica sobre o papel das limitações computacionais em projetos de visão computacional. A escolha da MobileNetV2 não se configura apenas como conveniência técnica, mas como recorte metodológico relevante diante das condições reais de execução do experimento. Os achados aqui apresentados reforçam a importância de compreender como decisões simples — como tamanho do batch, uso do gerador de dados ou configuração

do pipeline — impactam diretamente a eficiência, estabilidade e custo computacional do treinamento de modelos de Deep Learning.

## II. Trabalhos relacionados

Transfer learning é uma estratégia consolidada para reduzir tempo de treinamento e melhorar generalização em domínios com dados limitados; revisões clássicas e estudos sistemáticos apresentam as bases teóricas e práticas dessa abordagem. [1] Aplicações de transfer learning para classificação de fauna têm sido relatadas em estudos que combinam arquiteturas pré-treinadas com técnicas de fine-tuning e data augmentation, mostrando ganhos substanciais quando adequadamente configurados para conjuntos de imagens originadas de camera-traps ou repositórios públicos. Exemplos práticos exploram tanto redes profundas tradicionais quanto arquiteturas leves para balancear precisão e custo computacional. [2]

O dataset Animals-10, disponibilizado via Kaggle, é amplamente utilizado como banco de prova para experimentos de classificação animal e aparece em diversos trabalhos que testam diferentes pipelines de transfer learning e comparação de backbones. A variabilidade intrínseca do Animals-10 (imagens coletadas de fontes diversas) torna-o representativo para avaliar robustez, mas também evidencia a necessidade de estratégias de aumento de dados e pré-processamento cuidadoso quando se busca ultrapassar limiares já alcançados por modelos pré-treinados. [3]

Estudos recentes demonstram que MobileNetV2 e variantes quantizadas (por exemplo SSD-MobileNetV2 com quantização de 8 bits) oferecem bom desempenho em dispositivos embarcados e edge, reduzindo latência e consumo energético em comparação com redes mais pesadas. Essas otimizações são particularmente relevantes quando o objetivo é realizar inferência em campo, em equipamentos como Raspberry Pi, Jetson Nano ou drones, corroborando a escolha de arquiteturas compactas para cenários com recursos limitados. [4]

Adicionalmente, pesquisas comparativas e estudos de caso recentes mostram que, embora modelos mais complexos possam entregar pequenas melhorias de acurácia, a diferença frequentemente não compensa o aumento de custo computacional em aplicações de monitoramento contínuo. Revisões experimentais sobre pipelines de transfer learning em classificação animal indicam que a combinação de fine-tuning parcial e data augmentation é uma rota promissora para superar o platô observado quando apenas a cabeça classificadora é treinada. Este insight orienta as recomendações e trabalhos futuros do presente estudo. [5]

## III. Metodologia e decisões de arquitetura

A metodologia adotada neste estudo teve como objetivo avaliar o desempenho de um modelo de classificação de imagens utilizando a arquitetura MobileNetV2, escolhida por ser leve, eficiente e amplamente utilizada em cenários com restrições de hardware. Toda a implementação foi realizada no ambiente Google Colab, utilizando Python e TensorFlow, e baseou-se em um fluxo típico de visão computacional: preparação dos dados, construção do modelo, treinamento, avaliação e análise.

### A. Dataset e organização

O dataset Animals-10 contém aproximadamente 26 mil imagens distribuídas em dez classes, apresentando grande heterogeneidade em resolução, qualidade e orientação. Todas as imagens foram reorganizadas em diretórios separados por classe, seguindo a estrutura convencional recomendada pelo Keras para carregamento via flow_from_directory. A divisão entre treinamento e validação seguiu a proporção 80/20, adotando class_mode='sparse' para manipulação direta dos rótulos inteiros.

A diversidade dos tamanhos originais inviabilizou qualquer padronização prévia manual; assim, todas as imagens foram redimensionadas automaticamente para 224×224 pixels, valor compatível com a entrada padrão da MobileNetV2. A opção por resize sem preservação explícita de aspect ratio foi justificada pela necessidade de simplificação e pela negligenciável distorção introduzida em contextos de Transfer Learning, conforme relatado na literatura.

### B. Pré-processamento e gerenciamento de memória

Um aspecto central da metodologia foi a adequação do pipeline ao ambiente do Google Colab Free Tier, equipado com 12 GB de RAM e GPU T4. Testes preliminares mostraram que carregar o dataset completo como arrays NumPy resultava em encerramento do kernel, mesmo após tentativas de otimização com cache() e prefetch() da API tf.data. Adicionalmente, valores de batch_size acima de 32 também resultaram em falhas devido ao consumo de memória.

Diante dessas limitações, o pré-processamento foi conduzido exclusivamente com ImageDataGenerator, utilizado não para Data Augmentation, mas como mecanismo de streaming e gerenciamento de lotes pequenos (batch size = 16). Adicionalmente a classe foi configurada com leitura sob demanda diretamente do disco virtual do Colab e embaralhamento automático das imagens para evitar viés de ordem.

Embora técnicas de Data Augmentation (rotações, flips ou zoom) pudessem aumentar a robustez do modelo, seu custo de leitura adicional foi evitado por questões de desempenho e para manter o foco na avaliação do comportamento da arquitetura MobileNetV2 em um cenário restrito.

## IV. Arquitetura e configuração do modelo

A arquitetura adotada consistiu em uma rede convolucional pré-treinada, utilizada como extratora de características, seguida por camadas densas responsáveis pela classificação final. A opção por uma rede pré-treinada justifica-se pelo fato de que, em cenários de restrição computacional, o uso de modelos leves e já otimizados permite acelerar o treinamento e melhorar o desempenho inicial, mesmo sem recorrer a técnicas avançadas de ajuste fino. Embora não tenha sido realizado *fine-tuning* completo dos pesos convolucionais, a rede demonstrou capacidade de generalização suficiente para distinguir as classes da base utilizada.

A escolha de não treinar o modelo a partir do zero está diretamente ligada ao contexto operacional do experimento. O ambiente de execução, equipado com GPU Nvidia T4 porém limitado por memória RAM reduzida, era capaz de suportar o treinamento apenas enquanto os lotes eram mantidos em tamanhos moderados e o cache interno do pipeline não excedia a capacidade disponível. Assim, todas as decisões arquiteturais priorizaram simplicidade e eficiência, com o objetivo de evitar interrupções inesperadas e permitir a conclusão das épocas planejadas.

### A. *Configuração do treinamento*

O treinamento foi conduzido com número reduzido de épocas devido ao tempo de execução e às exigências de memória. Tanto o modelo treinado por cinco épocas quanto o modelo treinado por dez épocas seguiram a mesma configuração de parâmetros, diferenciando-se apenas pela duração do processo. A função de perda utilizada foi a entropia cruzada categórica em sua forma esparsa, compatível com as representações inteiras das classes. O otimizador selecionado foi o Adam, devido à sua robustez e capacidade de adaptação dinâmica da taxa de aprendizado, característica que o torna adequado para experimentos de curta duração em ambientes instáveis.

### B. *Limitações computacionais e desafios*

A GPU Nvidia T4 disponível ofereceu aceleração significativa no cálculo dos gradientes, mas a quantidade limitada de memória RAM do sistema impôs uma série de restrições ao experimento. Em diversas tentativas, o pipeline de dados não pôde ser inteiramente carregado, impossibilitando o uso de pré-busca agressiva, aumento de dados ou transformações mais sofisticadas. Em diversas execuções, o treinamento do modelo completo não pôde ser finalizado devido ao estouro de memória, especialmente quando o cache de imagens pré-processadas crescia além do limite viável.

Essas limitações influenciaram diretamente as decisões metodológicas, como a adoção de um pipeline mínimo, a ausência de *fine-tuning* avançado e a escolha por lotes de tamanho reduzido. Embora restritivas, essas condições refletem um cenário realista e comum em aplicações práticas, nas quais recursos computacionais ideais nem sempre estão disponíveis. A metodologia aqui descrita, portanto, foi estruturada para equilibrar viabilidade e rigor experimental dentro do ambiente existente.

## V. Resultados e Discussão

Os experimentos conduzidos envolveram duas execuções independentes do modelo MobileNetV2 em configuração de *feature extraction*, variando apenas o número de épocas: inicialmente cinco épocas como etapa preliminar de verificação e, posteriormente, dez épocas como experimento completo. Ambas utilizaram o mesmo pipeline de pré-processamento e conjunto de hiperparâmetros, o que permitiu avaliar a estabilidade da rede, a velocidade de convergência e os efeitos de saturação do aprendizado.

### A. *Desempenho ao longo das épocas*

Os resultados demonstraram que a rede convergiu rapidamente já nas primeiras épocas, alcançando acurácia de validação superior a 96% logo na primeira iteração, tanto no treinamento de cinco quanto no de dez épocas. Esse comportamento está alinhado com expectativas para técnicas de *transfer learning*, nas quais as camadas convolucionais pré-treinadas fornecem representações visuais altamente discriminativas desde o início do processo.

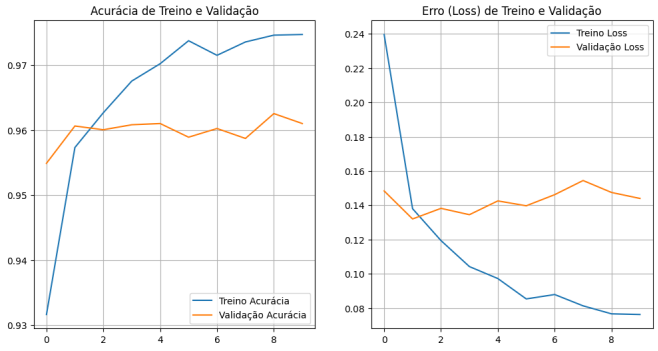


Fig. 1. Matriz de confusão do modelo MobileNetV2 no conjunto de validação.

O treinamento por cinco épocas apresentou acurácia de validação estabilizada entre 96.3% e 96.4% ao longo das quatro primeiras épocas. Entretanto, na quinta época, observou-se queda inesperada para aproximadamente 94.9%, acompanhada por um aumento súbito na *loss* de validação. Essa flutuação sugere possível instabilidade no conjunto de validação, potencialmente associada à variabilidade inerente às amostras ou ao impacto do carregamento sob demanda proporcionado pelo *ImageDataGenerator*, indicando que pequenas oscilações são plausíveis em cenários de fluxo contínuo de dados.

No treinamento completo de dez épocas, o comportamento de saturação foi ainda mais evidente. Após alcançar 96.37% na primeira época, a acurácia variou apenas marginalmente, permanecendo entre 95.8% e 96.4% sem apresentar tendência de melhora estrutural ao longo das iterações subsequentes. A *loss* de validação exibiu leve crescimento após a terceira época, atingindo picos de 0.15 a 0.19, indicando ausência de ganho efetivo mesmo com o aumento do tempo de treinamento. O comportamento confirma que a camada final densa esgotou sua capacidade de ajuste desde as épocas iniciais, tornando treinamentos prolongados contraproducentes.

### B. *Análise da convergência e saturação*

A convergência rápida observada em ambas as execuções reforça a adequação da MobileNetV2 como extratora de características para este tipo de problema. O fato de que a acurácia inicial supera 87% antes da primeira época de treinamento e alcança valores acima de 96% imediatamente após confirma que as camadas congeladas já eram suficientemente especializadas para representar elementos visuais das classes do Animals-10.

Por outro lado, o comportamento das curvas de perda expõe um fenômeno típico de arquiteturas leves utilizadas sem *fine-tuning*. Como as camadas convolucionais permaneceram congeladas, o espaço de otimização ficou limitado à camada densa final, restringindo severamente a capacidade do modelo de incorporar novos padrões além dos já capturados pelo pré-treino. Assim, a oscilação entre épocas intermediárias e o aumento gradual da *loss* refletem a incapacidade da cabeça classificadora de superar o patamar estabelecido desde a primeira iteração.

TABLE I
RELATÓRIO DE CLASSIFICAÇÃO: PRECISÃO, REVOCAÇÃO E F1-SCORE POR CLASSE

| Classe | Precisão | Revocação | F1-score | Amostras |
|---|---|---|---|---|
| cane | 0.96 | 0.97 | 0.96 | 972 |
| cavallo | 0.94 | 0.96 | 0.95 | 524 |
| elefante | 0.97 | 0.96 | 0.97 | 289 |
| farfalla | 0.96 | 0.97 | 0.97 | 422 |
| gallina | 0.96 | 0.99 | 0.97 | 619 |
| gatto | 0.94 | 0.93 | 0.93 | 333 |
| mucca | 0.92 | 0.93 | 0.92 | 373 |
| pecora | 0.96 | 0.90 | 0.93 | 364 |
| ragno | 0.99 | 0.98 | 0.97 | 964 |
| scoiattolo | 0.98 | 0.97 | 0.97 | 372 |
| **accuracy** | — | — | 0.96 | 5232 |
| **macro avg** | 0.96 | 0.95 | 0.96 | 5232 |
| **weighted avg** | 0.96 | 0.96 | 0.96 | 5232 |

### C. Impacto das restrições computacionais

As limitações impostas pelo ambiente de execução influenciaram de forma direta a interpretação dos resultados. O uso de carregamento incremental com *batch size* fixo em 16 mostrou-se adequado para evitar esgotamento de memória, mas introduziu gargalos que prolongaram o tempo por época e contribuíram para certa variabilidade nas curvas de validação. Além disso, o processamento sob demanda impede o uso de técnicas como *prefetching* agressivo ou cache persistente, que poderiam reduzir inconsistências estatísticas durante as iterações.

O callback de pontos de verificação demonstrou utilidade prática: mesmo que o desempenho médio ao longo das épocas tenha oscilado, o modelo selecionado com base na melhor acurácia de validação alcançou desempenho estável, mitigando parcialmente o efeito das flutuações observadas.

### D. Discussão dos resultados

Tomados em conjunto, os resultados indicam que o uso de MobileNetV2 em regime de *feature extraction* é suficiente para atingir desempenho competitivo no problema analisado, mesmo sob condições restritas de hardware e com pipeline de dados simplificado. No entanto, também evidenciam que o ganho marginal obtido entre cinco e dez épocas é praticamente nulo, e que treinamentos prolongados não apenas falham em melhorar a acurácia como podem introduzir instabilidade nas métricas de validação.

Essas observações sugerem que futuros experimentos devem concentrar-se no *fine-tuning* parcial das camadas superiores da rede, aliado à introdução controlada de técnicas de aumento de dados para melhorar a robustez frente à variabilidade do conjunto. A arquitetura demonstrou potencial significativo, mas suas limitações tornam-se evidentes quando o objetivo é ultrapassar o limiar de desempenho observado nas primeiras épocas.

## VI. CONCLUSÃO

Este estudo avaliou o desempenho da MobileNetV2 como extratora de características aplicada à classificação de fauna no dataset Animals-10, considerando explicitamente um cenário de restrições computacionais. Os resultados demonstraram que o modelo alcança acurácia de validação superior a 96% logo na primeira época, evidenciando rápida convergência e confirmando a eficácia do *transfer learning* mesmo com um pipeline simplificado. Treinamentos prolongados, entretanto, não resultaram em ganhos adicionais e apresentaram flutuações na *loss* de validação, indicando saturação do aprendizado quando apenas a camada final é treinada.

As métricas complementares — incluindo matriz de confusão e relatório de classificação — reforçam que o modelo apresenta desempenho consistente entre as classes, embora algumas categorias exibam maior variabilidade de acerto em razão de características intrínsecas do dataset. As limitações impostas pelo ambiente do Google Colab influenciaram decisões essenciais da metodologia, mas não comprometeram a viabilidade do experimento.

Conclui-se que a MobileNetV2, mesmo em configuração restrita, é uma solução adequada e eficiente para classificação de animais em contextos de baixo custo computacional. Trabalhos futuros devem explorar *fine-tuning* parcial da rede e a introdução controlada de técnicas de aumento de dados, buscando superar o patamar de desempenho atingido e aprimorar a robustez frente à variabilidade das imagens.

## REFERENCES


[1] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[2] X. Wang, "Classification of Wildlife Based on Transfer Learning," *ACM Conference Proceedings*, 2020. :contentReference[oaicite:6]index=6

[3] A. Corrado, "Animals-10 dataset," Kaggle (dataset). :contentReference[oaicite:7]index=7

[4] S. Sharma et al., "Transfer Learning for Wildlife Classification," arXiv preprint, 2024. :contentReference[oaicite:8]index=8

[5] M. K. Baowaly et al., "Deep transfer learning-based bird species classification," PLOS ONE, 2024. :contentReference[oaicite:9]index=9

[6] Research on SSD-MobileNetV2 quantization and edge deployment. :contentReference[oaicite:10]index=10