Guia 1

Análise de modelos recomendados de *Transfer Learning* aplicado à Classificação de Imagens para Dataset WaRP – *Waste Recycling Plant Dataset*

1. Sobre o Dataset WaRP – *Waste Recycling Plant Dataset*

O **WaRP Dataset** foi desenvolvido para tarefas de visão computacional relacionadas à triagem de resíduos recicláveis em plantas industriais. Ele está organizado em três sub-conjuntos:

- **WaRP-D**: voltado à detecção de objetos em cenas industriais.
- **WaRP-C**: voltado à **classificação de imagens**, sendo o foco principal desta análise.
- **WaRP-S**: destinado à segmentação fracamente supervisionada.

1.1. Estrutura das classes do WaRP-C

O **WaRP-C** contém **8.823 imagens**, distribuídas em **6 classes** e **28 categorias (subclasses)** de resíduos recicláveis. Essas classes apresentam **alto desbalanceamento**, sendo que a classe mais rara possui apenas ~24 amostras. As imagens possuem **dimensões variadas**, com resoluções que vão de **35×40 até 668×703 pixels**, o que impõe desafios relevantes de **padronização, generalização e processamento**.

As classes estão agrupadas da seguinte forma:

- **Plastic bottles** (garrafas plásticas): 17 categorias (prefixo `bottle-`)
- **Glass bottles** (garrafas de vidro): 3 categorias (prefixo `glass-`)
- **Card boards** (papelão): 2 categorias
- **Detergents** (detergentes): 4 categorias
- **Canisters** (latas)
- **Cans** (recipientes)
- O sufixo -full indica que a garrafa está cheia de ar (ou seja, não está vazia).

2. Tipos de Aprendizado, Modelos e Algoritmos em Machine Learning

Em Machine Learning (ML), os tipos de aprendizado, os modelos e os algoritmos formam os três pilares essenciais que fundamentam o desenvolvimento de sistemas computacionais capazes de aprender a partir de dados e tomar decisões inteligentes. Esses elementos são amplamente aplicados em diferentes setores, desde recomendação de produtos, diagnósticos médicos, reconhecimento de imagens, até classificação de resíduos, como é o caso do dataset WaRP-C. O domínio dessas categorias permite construir soluções robustas, adaptáveis e eficientes para problemas do mundo real.

O **primeiro pilar**, os **tipos de aprendizado**, refere-se à maneira como o modelo aprende com os dados disponíveis. O mais comum é o **aprendizado supervisionado**, no qual os modelos são treinados com dados rotulados – ou seja, cada entrada possui uma saída conhecida. Esse paradigma é ideal para tarefas de **classificação** (por exemplo, identificar se uma imagem representa papel, plástico ou metal) ou **regressão** (como prever a temperatura para os próximos dias com base em séries históricas). Dentre os algoritmos supervisionados, destacam-se a **Regressão Linear**, a **Regressão Logística**, **Support Vector Machines (SVM)**, **k-Nearest Neighbors (k-NN)**, **Árvores de Decisão** e **Redes Neurais** (Mitchell, 1997; Géron, 2019). No caso do WaRP-C, por exemplo, é com aprendizado supervisionado que se realiza a classificação das imagens em categorias de resíduos recicláveis, pois os rótulos (como "papel", "plástico", "vidro") já são previamente definidos.

Já o **aprendizado não supervisionado** se aplica quando não há rótulos disponíveis. O objetivo é explorar padrões ocultos nos dados, revelando estruturas como agrupamentos, associações ou distribuições latentes. É comum em aplicações como **segmentação de clientes**, **compressão de imagens** e **redução de dimensionalidade** em dados genômicos ou de sensores. Entre os algoritmos mais utilizados estão **K-Means**, **DBSCAN**, **Hierarchical Clustering** e **PCA (Principal Component Analysis)** (Murphy, 2012).

O **aprendizado por reforço** representa outro paradigma, no qual um **agente** aprende a tomar decisões em um **ambiente dinâmico** com base em **recompensas** ou **punições** recebidas após cada ação. Essa abordagem é amplamente empregada em **jogos (como o AlphaGo)**, **robótica (navegação autônoma)** e **sistemas de controle (como veículos autônomos)**. Os algoritmos mais conhecidos incluem **Q-Learning**, **Deep Q-Networks (DQN)** e **Proximal Policy Optimization (PPO)** (Sutton & Barto, 2018).

Outros tipos relevantes incluem o **aprendizado semi-supervisionado**, útil quando apenas uma pequena parte dos dados está rotulada, e o **aprendizado auto-supervisionado**, onde o modelo aprende gerando suas próprias tarefas preditivas a partir dos dados brutos. Este último tem sido amplamente explorado em modelos de linguagem natural como o BERT e no campo da visão computacional com técnicas de contraste, como o SimCLR (LeCun et al., 2021).

O **segundo pilar**, os **modelos de Machine Learning**, corresponde à estrutura matemática responsável por transformar os dados de entrada em previsões ou classificações. Os **modelos lineares**, como a Regressão Linear e Logística, são úteis em problemas onde a relação entre as variáveis é simples e facilmente interpretável. Já os **modelos baseados em árvores**, como **Árvores de Decisão**, **Random Forests** e **XGBoost**, são eficazes em capturar

relações não-lineares e **interações entre variáveis**, sendo amplamente usados em **sistemas de crédito**, **diagnósticos médicos**, e **prognósticos industriais** (Hastie, Tibshirani & Friedman, 2009).

Os **modelos de vizinhança**, como o **k-NN**, comparam diretamente os dados de entrada com exemplos anteriores, sendo úteis para **recomendações baseadas em similaridade** (como em e-commerce). Os **modelos probabilísticos**, como o **Naive Bayes** e os **Modelos Ocultos de Markov (HMMs)**, são indicados para tarefas que envolvem **incerteza**, como **detecção de spam** ou **análise de séries temporais**.

Os **modelos de conjunto** (ensemble), como **Bagging**, **Boosting** e **Stacking**, combinam vários modelos para aumentar a acurácia e reduzir o viés ou a variância. São bastante eficazes em **competições de ciência de dados**, como no Kaggle. Por fim, os **modelos baseados em redes neurais profundas (Deep Learning)**, como **CNNs**, **RNNs** e **Transformers**, são os mais poderosos para lidar com **dados de alta dimensionalidade** e **representações complexas**, como em **processamento de imagens**, **áudio**, e **linguagem natural** (Goodfellow, Bengio & Courville, 2016).

O **terceiro pilar** são os **algoritmos de aprendizado**, que constituem os procedimentos computacionais responsáveis por ajustar os parâmetros dos modelos com base nos dados. Diferentes tarefas requerem diferentes algoritmos. Para **classificação**, os mais comuns são **SVM**, **Decision Trees**, **Random Forests** e **Redes Neurais**. Para **regressão**, usa-se **Regressão Linear**, **SVR (Support Vector Regression)** e **Árvores de Regressão**. Para **agrupamento**, algoritmos como **K-Means**, **DBSCAN** e **Gaussian Mixture Models** são populares. Para **redução de dimensionalidade**, destacam-se **PCA**, **t-SNE** e **UMAP**. Em **Deep Learning**, os algoritmos incluem **CNNs** (como para classificação de resíduos), **Transformers** (para modelos como ChatGPT) e **RNNs** (para séries temporais). Já no **aprendizado por reforço**, são utilizados algoritmos como **Q-Learning**, **DDPG** e **A3C** (Raschka & Mirjalili, 2022).

A escolha entre esses aprendizados, modelos e algoritmos depende de fatores como o tamanho e a natureza do dataset, os objetivos da tarefa, a disponibilidade de rótulos e os recursos computacionais. No projeto com o **dataset WaRP-C**, por exemplo, optou-se por uma **abordagem supervisionada**, com uso de **CNNs pré-treinadas (Transfer Learning)**, uma vez que as classes já estão rotuladas, há desbalanceamento de dados e restrição computacional. Com essa estratégia, aproveita-se o aprendizado prévio de grandes bases (como o ImageNet), ajustando o modelo para uma tarefa específica, mesmo com número reduzido de amostras por classe.

3. Por que Redes Neurais em detrimento de outros algoritmos clássicos?

A escolha por Redes Neurais Convolucionais (CNNs) em detrimento de algoritmos tradicionais como Máquinas de Vetores de Suporte (SVM), Random Forests ou K-Nearest Neighbors (KNN) se justifica, sobretudo, pela natureza dos dados que compõem o dataset WaRP-C. Este conjunto de dados é formado por imagens de resíduos recicláveis com variação significativa de tamanho (de 35x40 até 668x703 pixels), diferentes resoluções e distribuição altamente desbalanceada entre as classes (a menor classe com apenas 24 imagens).

Algoritmos clássicos como SVMs ou Random Forests apresentam desempenho limitado quando aplicados diretamente em dados brutos de imagem. Em geral, esses modelos exigem uma etapa de extração manual de características (features), a qual pode ser subjetiva, limitada e sensível à variabilidade das imagens. Além disso, não capturam bem padrões espaciais complexos, como contornos, texturas, gradações de cor e estruturas hierárquicas. Por outro lado, as CNNs são projetadas especificamente para lidar com dados visuais. Elas operam aprendendo automaticamente representações hierárquicas a partir das imagens, começando com a detecção de bordas e formas simples e progredindo para padrões mais complexos. Essa capacidade de aprendizado de características espaciais diretamente dos dados as torna particularmente eficazes em contextos como o do WaRP-C.

Adicionalmente, as CNNs se beneficiam de avanços em hardware (como GPUs) e frameworks (TensorFlow, PyTorch, Keras), facilitando sua aplicação mesmo para pesquisadores iniciantes. Elas também são compatíveis com técnicas modernas de regularização e ajuste fino (fine-tuning), o que permite controlar o overfitting e melhorar a generalização, mesmo em conjuntos de dados pequenos.

Outro fator crítico é a escalabilidade. Uma vez treinada ou ajustada, uma CNN pode ser facilmente transferida para novos domínios ou problemas semelhantes. Isso se alinha ao conceito de reutilização de conhecimento, que é crucial em projetos com recursos limitados de dados e processamento.

4. Por que Transfer Learning?

Transfer Learning, ou Aprendizado por Transferência, tem se consolidado como uma estratégia fundamental em tarefas de classificação de imagens, especialmente em contextos em que há limitações de dados disponíveis para treinamento. O princípio básico do Transfer Learning é reutilizar o conhecimento aprendido por um modelo em uma tarefa base (por exemplo, classificação de imagens no ImageNet) para resolver uma tarefa diferente, porém relacionada — como a classificação de resíduos recicláveis no dataset WaRP-C.

No cenário do WaRP-C, com aproximadamente 8.823 imagens e 28 classes altamente desbalanceadas, onde a classe mais rara contém cerca de 24 amostras, o uso de Transfer Learning é particularmente recomendado. Treinar uma rede neural convolucional (CNN) do zero nesse contexto acarretaria em graves riscos de overfitting e baixa generalização, uma vez que os modelos modernos como ResNet, DenseNet ou EfficientNet possuem dezenas de

milhões de parâmetros. Transfer Learning permite aproveitar modelos previamente treinados em grandes corpora de dados como o ImageNet (com mais de 14 milhões de imagens rotuladas) e aplicar esse conhecimento prévio ao novo domínio, com poucas amostras por classe.

Segundo Pan e Yang (2010), as vantagens do Transfer Learning incluem:
- Melhor desempenho com dados escassos;
- Aceleração do processo de treinamento;
- Redução do custo computacional;
- Melhoria da generalização em contextos ruidosos ou desbalanceados.

Em modelos como EfficientNet, ResNet ou VGG, os primeiros blocos convolucionais aprendem padrões universais como bordas, texturas e formas básicas, o que os torna transferíveis entre domínios com imagens visualmente distintas. Isso significa que mesmo em uma base industrial como a do WaRP-C, que difere bastante do domínio natural do ImageNet, o reaproveitamento dessas representações iniciais tende a ser eficiente.

Adicionalmente, em experimentos publicados por Tan & Le (2021), o uso de Transfer Learning com EfficientNet-V2 em datasets menores resultou em saltos significativos de acurácia, saltando de 86% para mais de 92% com a aplicação de técnicas de data augmentation e fine-tuning. Esses ganhos demonstram a aplicabilidade e a eficácia do Transfer Learning em cenários como o deste projeto.

5. Modelos recomendados de *Transfer Learning*

5.1. EfficientNet (B0–B2 ou V2 para ganho adicional)

- **Escalabilidade eficiente**: usa *compound scaling* para balancear profundidade, largura e resolução, otimizando desempenho com baixo custo computacional.
  Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. ICML.
  https://arxiv.org/abs/1905.11946

- **Robustez em transferência**: apresenta alta acurácia mesmo em conjuntos de dados pequenos e desbalanceados.
  Tan, M., & Le, Q. V. (2021). *EfficientNetV2: Smaller models and faster training*. ICML.
  https://arxiv.org/abs/2104.00298

- **Custo-benefício**: treinável com GPUs comuns (224×224), ideal para ambientes como Google Colab.

5.2. ResNet50

- **Confiabilidade e estabilidade**: é uma arquitetura amplamente adotada, com excelente desempenho em tarefas de classificação com *Transfer Learning*.
  He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*. CVPR.
  https://arxiv.org/abs/1512.03385

- **Ampla documentação e suporte**: existem muitos tutoriais e implementações otimizadas disponíveis.

- **Desempenho competitivo**: apesar de mais pesada, ainda é um excelente benchmark.

6. Modelos menos recomendados para este cenário

Modelo	Prós	Contras	Referência
VGG16/VGG19	Arquitetura simples e fácil de implementar	Muito pesada, propensa a *overfitting*	Simonyan & Zisserman (2014) arXiv:1409.1556
InceptionV3	Bons resultados com múltiplas escalas	Complexidade maior e inferior ao EfficientNet em custo	Szegedy et al. (2016) arXiv:1512.00567
MobileNetV2	Leve, ideal para dispositivos móveis (*edge*)	Menor precisão em tarefas exigentes	Sandler et al. (2018) arXiv:1801.04381
DenseNet	Excelente fluxo de gradiente	Elevado consumo de memória	Huang et al. (2017) arXiv:1608.06993
ViT/Swin	Avanços recentes com Transformers	Alta demanda computacional, requer muitos dados	Dosovitskiy et al. (2021) arXiv:2010.11929

7. Resumo comparativo

Modelo	Vantagens principais	Limitações	Referência

EfficientNet-B0/B2	Leve, preciso, ideal para datasets pequenos e desbalanceados	Menos flexível que ResNet em alguns cenários	Tan & Le (2019, 2021)
ResNet50	Confiável, robusto, muito documentado	Mais pesado, eficiência inferior ao EfficientNet	He et al. (2016)
InceptionV3	Boa acurácia, uso de múltiplas escalas	Complexidade elevada e custo computacional	Szegedy et al. (2016)
VGG16/19	Simples de implementar	Propenso a *overfitting*, pesado	Simonyan & Zisserman (2014)
MobileNetV2	Ágil, otimizado para dispositivos móveis	Baixa acurácia em tarefas mais complexas	Sandler et al. (2018)
DenseNet	Fluxo eficiente de gradientes	Alto consumo de memória	Huang et al. (2017)
ViT/Swin	Alto potencial em grandes bases de dados	Setup sofisticado, risco de *underfitting*	Dosovitskiy et al. (2021)

8. Qual tipo de Transfer Learning?

Dentro do Transfer Learning, existem diferentes estratégias para adaptação de modelos pré-treinados. A abordagem mais recomendada para o WaRP-C é o Fine-Tuning Parcial (Partial Fine-Tuning), que consiste em congelar parte das camadas convolucionais de um modelo pré-treinado e ajustar apenas as últimas camadas densas, normalmente responsáveis pela classificação final. Isso permite preservar os conhecimentos aprendidos nas camadas iniciais, ao mesmo tempo em que se adapta a arquitetura para a tarefa específica em questão.

Em um primeiro momento, as camadas convolucionais devem permanecer congeladas, permitindo que apenas os pesos da nova camada fully connected (FC) sejam atualizados. Após algumas épocas de treinamento, é possível liberar camadas adicionais para um fine-tuning gradual, melhorando a capacidade de especialização do modelo à nova tarefa. Essa abordagem progressiva reduz o risco de overfitting, comum em bases pequenas, como o WaRP-C.

O uso de EfficientNet-B0 e EfficientNet-B2 é particularmente indicado nesse contexto. Esses modelos possuem menos parâmetros que versões maiores como EfficientNet-B7, mas ainda assim oferecem excelente desempenho com baixo custo computacional. A

arquitetura baseada em compound scaling permite um balanceamento eficaz entre profundidade, largura e resolução, tornando-o ideal para execução em ambientes com GPU limitada (como o Google Colab).

Segundo Tan & Le (2019), EfficientNet-B0 é capaz de superar ResNet-50 e DenseNet em vários benchmarks com menor número de parâmetros. Em datasets desbalanceados, esse tipo de arquitetura, aliado a técnicas como data augmentation e oversampling de classes minoritárias, pode produzir resultados expressivos com menos dados rotulados.

9. Métricas e Desempenho Esperados

A escolha das métricas corretas é essencial para avaliar o desempenho de modelos de classificação, especialmente em cenários com desbalanceamento entre classes, como é o caso do dataset WaRP-C. Utilizar apenas a métrica de acurácia pode ser enganoso, pois ela tende a privilegiar as classes majoritárias. Assim, outras métricas devem ser utilizadas em conjunto para uma análise mais robusta do modelo.

Entre as métricas recomendadas, destacam-se:

• **Accuracy**: Representa a proporção total de acertos. É útil como visão geral, mas não é suficiente sozinha em datasets com classes desbalanceadas.

• **Precision**: Mede a proporção de verdadeiros positivos entre os exemplos classificados como positivos. Relevante para minimizar falsos positivos.

• **Recall**: Mede a proporção de verdadeiros positivos entre todos os exemplos da classe. Fundamental para garantir que o modelo reconhece classes minoritárias.

• **F1-Score (macro)**: Média harmônica entre Precision e Recall, calculada igualmente entre as classes. Ideal para cenários multiclasse e desbalanceados.

• **Matriz de Confusão**: Ferramenta visual que permite identificar quais classes estão sendo confundidas com outras. Útil para visualizar onde o modelo erra e quais classes são confundidas.

• **AUC-ROC (por classe):** Útil para avaliar separabilidade classe-a-classe com abordagem one-vs-all.

• **Cohen's Kappa:** Mede a concordância entre previsões e rótulos verdadeiros, ajustada ao acaso. Importante em contextos com chance elevada de acertos aleatórios.

Com base em benchmarks da literatura e experimentos prévios em datasets similares ao WaRP-C, pode-se esperar o seguinte desempenho com alguns modelos:

• EfficientNet-B0: Accuracy ~88–92%; F1-Score Macro ~70–80%

• ResNet50: Accuracy ~86–90%; F1-Score Macro ~68–78%

• VGG16: Accuracy ~80–84%; F1-Score Macro ~60–70%

• MobileNetV2: Accuracy ~82–88%; F1-Score Macro ~65–75%

O uso de técnicas de **data augmentation, oversampling de classes minoritárias, ajuste de pesos** e uso de **callbacks** durante o treinamento pode melhorar o desempenho nessas métricas.

10. Pipeline recomendado para o WaRP-C

1. **Pré-processamento**:
    - Redimensionar imagens: **Resize + padding** para 224×224, mantendo proporção.
    - **Normalização de pixels**: *rescale(1./255)* (necessária se estiver utilizando Keras/TensorFlow).

2. **Data Augmentation** (essencial para minoritárias):
    - Foco em classes minoritárias:
        - Horizontal flip
        - Rotação aleatória
        - Zoom
        - Ajuste de brilho

3. **Treinamento**:
    - Começar com **EfficientNet-B0** com camadas *congeladas* e *top layers* afináveis:
        - **Congelar camadas convolucionais iniciais** (pesos pré-treinados).
        - **Treinar apenas os top layers** (camadas finais: fully connected + classifier).

- Utilizar **fine-tuning parcial** após alguns epochs, se necessário.

- Avaliar métricas como *F1-score por classe* e *recall*

- Experimentar **EfficientNet-V2** e comparar com **ResNet50**

4. **Callbacks recomendados:**

    - **EarlyStopping**: para evitar overfitting, monitorando o F1 macro ou val_loss.

    - **ReduceLROnPlateau**: para ajuste dinâmico da taxa de aprendizado ao detectar platôs.

5. **Ajustes para classes minoritárias**:

    - **Oversampling seletivo** das classes raras.

    - **Ajuste de pesos: Reponderação de classes** (class weights) na função de perda.

    - **SMOTE em embeddings** (opcional, para balanceamento sintético).

6. **Avaliação**

    - Monitoramento por **F1-score macro**, **recall por classe** e análise da **matriz de confusão**.

11. Opção Final para o Projeto

Após considerar as características específicas do dataset WaRP-C — como o tamanho relativamente pequeno, o desbalanceamento severo entre classes e a variação na resolução das imagens —, o modelo mais indicado é o EfficientNet-B0 ou EfficientNet-B2 com fine-tuning parcial.

Esses modelos foram projetados com foco em eficiência de escala e mostram ótimo desempenho em tarefas de classificação com recursos computacionais limitados. A versão B0, por exemplo, apresenta excelente acurácia com tamanhos de entrada reduzidos (224×224), o que se alinha bem ao contexto do projeto.

Motivos principais para a escolha:

• Arquitetura leve, adequada para ambientes com recursos limitados (como Google Colab);
• Bom desempenho mesmo em imagens com resolução variável;
• Robusto a desbalanceamentos, quando combinado com técnicas de data augmentation;
• Menor propensão ao overfitting do que modelos mais antigos como VGG16.

Pipeline sugerido:

1. Pré-processamento: redimensionar todas as imagens para 224×224, aplicando padding quando necessário.
2. Augmentation: aplicar técnicas de data augmentation (flip horizontal, rotação, brilho, zoom) com foco nas classes minoritárias.
3. Treinamento: utilizar EfficientNet-B0 congelando as camadas iniciais e treinando apenas as camadas finais (classifier + FC head).
4. Ajustes: monitorar o F1-score por classe, realizar oversampling seletivo ou ajuste de pesos conforme necessário.

Como benchmark secundário, o ResNet50 pode ser utilizado para comparação de desempenho e robustez.

12. Conclusão

Para o cenário — **classificação de resíduos recicláveis com base em imagens** de um dataset com ~9k amostras, **alto desbalanceamento, resolução variada das imagens e volume moderado de dados — o Transfer Learning com EfficientNet-B0 ou B2** é a **melhor escolha**: eficiente, leve, altamente adaptável a *fine-tuning*, e ideal para ambientes computacionais moderados. A **ResNet-50** pode ser usada como benchmark ou segunda opção robusta.

Modelos mais antigos ou pesados (como VGG ou DenseNet) e *Transformers* como ViT devem ser evitados neste momento, pois exigem mais recursos e não oferecem ganho proporcional para a escala do projeto.

Essa escolha **não é apenas técnica, mas estratégica**: permite **maximizar desempenho mesmo com recursos computacionais limitados e poucos dados rotulados**.

Ao combinar esse modelo com boas práticas de pré-processamento, data augmentation, fine-tuning parcial e callbacks inteligentes, é possível obter resultados robustos, com excelente generalização e equilíbrio entre acurácia e sensibilidade por classe.

13. Referências Bibliográficas

DOSOVITSKIY, A. et al. *An image is worth 16x16 words: Transformers for image recognition at scale*. 2021. Disponível em: https://arxiv.org/abs/2010.11929.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media, 2019.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning*. Springer, 2009.

HE, K. et al. *Deep residual learning for image recognition*. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. Disponível em: https://arxiv.org/abs/1512.03385.

HUANG, G. et al. *Densely Connected Convolutional Networks*. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. Disponível em: https://arxiv.org/abs/1608.06993.

KRIŽEVA, B. et al. *Comparison of image classification methods: A case study*. *Journal of Electrical Engineering*, 2018.

LECUN, Y.; HADSELL, R.; CHOPRA, S. *A Path Towards Autonomous Machine Intelligence*. *Communications of the ACM*, 2021.

LECUN, Y. et al. *Deep learning*. *Nature*, v. 521, p. 436–444, 2015.

MITCHELL, T. M. *Machine Learning*. McGraw-Hill, 1997.

MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

PAN, S. J.; YANG, Q. *A Survey on Transfer Learning*. *IEEE Transactions on Knowledge and Data Engineering*, v. 22, n. 10, p. 1345–1359, 2010.

RASCHKA, S.; MIRJALILI, V. *Python Machine Learning* (3rd ed.). Packt, 2022.

SANDLER, M. et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. Disponível em: https://arxiv.org/abs/1801.04381.

SIMONYAN, K.; ZISSERMAN, A. *Very deep convolutional networks for large-scale image recognition*. 2014. Disponível em: https://arxiv.org/abs/1409.1556.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press, 2018.

SZEGEDY, C. et al. *Rethinking the Inception Architecture for Computer Vision*. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. Disponível em: https://arxiv.org/abs/1512.00567.

TAN, M.; LE, Q. V. *EfficientNet: Rethinking model scaling for convolutional neural networks*. In: International Conference on Machine Learning (ICML), 2019. Disponível em: https://arxiv.org/abs/1905.11946.

TAN, M.; LE, Q. V. *EfficientNetV2: Smaller models and faster training*. In: International Conference on Machine Learning (ICML), 2021. Disponível em: https://arxiv.org/abs/2104.00298.