Comparação de desempenho entre algoritmos de classificação na extração de informação a partir de datasets sintéticos randomizados

Felipe N.C. Carvalho[1]

[1] Centro de Matemática Computação e Cognição (CMCC)
Universidade Federal do abc (UFABC) – Santo André, SP – Brazil

felipe.nunes@aluno.ufabc.edu.br

Resumo. *O treinamento de modelos de classificação de imagens em muitos casos pode ser de alta complexidade devido à dificuldade de adquirir dados de treinamento suficientes em determinados domínios. Por essa razão, o uso da técnica de randomização de domínio se apresenta como uma forma de contornar este problema. Este artigo visa realizar uma análise quantitativa comparativa acerca da eficiência de redes VGG e Resnet na implementação de classificadores treinados apenas a partir de bases de dados sintéticos gerados através da técnica de randomização de domínio, ao reimplementar o trabalho realizado em [Valtchev and Wu 2021] com o objetivo de maximizar a acurácia média obtida por classificadores treinados via datasets randômicos através de uma arquitetura mais moderna.*

1. Introdução

Nos últimos anos, a realização de pesquisas na área de Inteligência Artificial (IA) tem se tornado cada vez mais relevante e frequente, e o tema passou a estar no centro de muitas das discussões não apenas da academia, mas também na população em geral. Em meio a esse cenário, os modelos de IA baseados em aprendizado supervisionado tem se destacado, sendo utilizados em infindas aplicações, tais como a criação de ferramentas extremamente complexas, como o *Chat GPT* e outras *IAs* generativas, modelos mais simples como classificadores e diversos outros.

Ainda que muito eficientes, os modelos de aprendizado supervisionado exigem uma grande quantidade de dados para seu treinamento e apesar do fato de que em alguns casos esses dados serem de fácil obtenção e livre consulta, em diversos outros, este não é o caso. Um exemplo prático pode ser dado para [Valan et al. 2021], que cita em seu trabalho o desafio de coletar amostras em quantidades suficientes de espécimes de insetos e a identificação de suas características, preservando a integridade das mesmas, uma vez que o processo de obtenção e preparação das amostras destes espécimes demanda muito tempo e recursos. Para além disso, é importante frisar que mesmo quando há a disponibilidade de um número satisfatório de amostras para uma determinada aplicação, outro problema recorrente é a atribuição de rótulos a essas amostras, que por muitas vezes é extremamente complexa, chegando a necessitar até mesmo de informações específicas para cada um dos pixels de uma imagem, por exemplo. Devido a essa complexidade, a criação de bases de dados (*datasets*) úteis para treinamento e de simples obtenção tem sido um desafio constante na área.

Dentro deste contexto, quando falamos de problemas envolvendo imagens, a criação de *datasets* pode ser feita de forma artificial por meio da utilização de ferramentas de computação gráfica, tais como os casos apresentados por [Roberts et al. 2021, Ros et al. 2016, Tremblay et al. 2018b, Varol et al. 2017]. Estes trabalhos entretanto, realizam a síntese de cenas buscando a fidelidade com a realidade, o que gera a necessidade de um grande esforço humano na elaboração de modelos 3D e texturas, o que torna o processo demorado e custoso.

Para solucionar o problema da elaboração de *datasets*, surgiu a proposta da Randomização de Domínio (RD). Essa abordagem utiliza distribuições aleatórias para gerar ambientes, texturas e transformações geométricas, com o objetivo de alcançar uma invariância de domínio nas imagens sintéticas geradas [Yue et al. 2019]. Essa extensa variação de parâmetros possibilita que modelos de inteligência artificial generalizem o problema, extraindo apenas as informações mais importantes das múltiplas variações possíveis [Loquercio et al. 2020].

A técnica de RD tem se mostrado extremamente eficaz em sua tarefa de gerar datasets úteis para treinamento de Redes Neurais convolucionais de classificação, tais como Redes Neurais Residuais (resnets), Redes VGG, e outras. Dado que o objetivo da existência destes datasets gerados via RD é permitir que os classificadores treinados com ele sejam capazes de generalizar para dados reais nunca antes vistos, a comparação entre modelos de arquitetura para os classificadores pode trazer informações relevantes sobre quais tipos de algoritmos consegue atingir melhores acurácias na extrapolação para dados reais ao ser treinada com este tipo de algoritmo. Portanto, a partir dessa dúvida este trabalho se faz relevante, ao reimplementar o trabalho realizado por [Valtchev and Wu 2021] com uma arquitetura de rede neural distinta, visando realizar um comparativo da acurácia atingida pelas redes *Visual Geometry Group* (VGG) e *residual neural network* (Resnet) neste caso em específico.

2. Fundamentação teórica

As redes neurais convolucionais (CNN) inicialmente propostas por [LeCun et al. 1989] tem sido extensivamente estudadas nas ultimas décadas devido à sua grande eficácia em criar modelos precisos de classificação de imagens. Entre as diferentes propostas de redes neurais convolucionais que surgiram ao longo dos anos, duas de grande destaque são as redes VGG [Simonyan and Zisserman 2014] e Resnet [He et al. 2016]. Apesar do fato de ambas, Resnet e VGG serem modelos de aprendizado profundo baseados em CNNs, elas não funcionam da mesma forma, possuindo distinções muito importantes entre si e que devem ser entendidas para que se compreenda corretamente o objetivo deste trabalho.

2.1. CNNs

As redes Convolucionais mais simples, tais como a *LeNet* proposta por [LeCun et al. 1989] são compostas por três tipos de camadas: Convolucionais, *pooling* e de neurônios completamente ligados. De forma simplificada, elas funcionam intercalando camadas de convolução que utilizam *kernels* em conjunto com uma função de ativação tal como a *ReLu* com camadas de *pooling*. Por fim a rede adiciona ao menos duas camadas de neurônios, que funcionam como um perceptron multi camadas (MLP).

As camadas convolucionais garantem ao algoritmo um aumento das características aprendidas ao viabilizar a análise de características cada vez mais específicas e complexas

a cada convolução. Camadas de pooling, por sua vez, atuam na diminuição de dimensionalidade do tensor que representa a imagem, além de propiciar ao algoritmo uma maior invariância a pequenas alterações nos valores de entrada. De maneira similar aos MLPs, as CNNs podem ser treinadas com uma variação do algoritmo *backpropagation*, sendo que o modelo aprende não apenas os pesos das camadas completamente ligadas, mas também os *kernels* das convoluções.

2.2. VGGs

As redes VGG apresentadas por [Simonyan and Zisserman 2014] não se distanciam muito da CNN original no que diz respeito ao pipeline geral. Entretanto, as mudanças na arquitetura propostas por ela tornaram possível a construção de redes ordens de grandeza mais profundas, ao propor o uso de *kernels* menores (3x3), que obviamente permitem que o *patern matching* resultante da operação de convolução tenha dimensões maiores do que as da CNN de Lecun, que trabalhava com *kernels* maiores (5x5 ou 7x7), viabilizando um maior número de camadas de convolução.

Outra mudança importante apresentada pela VGG é o uso da ReLu como função de ativação ao invés da tangente hiperbólica utilizada por *Lecun*.

2.3. Resnets

A pesar de as redes VGG terem provado que a profundidade de uma CNN tem grande impacto na eficiência dos modelos gerados, números altos de camadas convolucionais passavam a gerar uma queda na acurácia dos modelos devido ao problema da dissipação do gradiente no algoritmo do *backpropagation*, que fazia com que as camadas iniciais da rede parassem de aprender. Foi nesse contexto que a Resnet apareceu, resolvendo este problema com a introdução das conexões residuais, que consistem em enviar a saída do *pooling* para frente no pipeline das convoluções, ”pulando”algumas delas. Simultaneamente a isso as convoluções tradicionais também eram feitas, e após o número de convoluções definidas no ciclo, ambos os resultados são somados, causando a propagação dos dados da imagem inicial para camadas mais profundas, com pouca perda de informação e ainda assim mantendo o aprendizado oferecido pelas convoluções adicionais.

3. Reimplementação

3.1. Recapitulação do trabalho original

O artigo de [Valtchev and Wu 2021], que este trabalho busca reimplementar, utiliza uma rede VGG para criar um modelo de classificação de imagens de cães e gatos. Nesse artigo, foi empregada a técnica de *transfer learning*, na qual se utiliza um modelo pré-treinado em uma base de dados genérica como ponto de partida. Ao aplicar essa técnica, o treinamento da nova rede torna-se muito mais rápido, pois as camadas convolucionais são mantidas intactas e apenas os pesos do MLP das camadas finais são reaprendidos.

Os autores não deixaram claro qual modelo serviu de base para o *transfer learning*; portanto, neste trabalho optou-se por utilizar a linguagem Python, implementando os modelos VGG-16 e ResNet-50, ambos pré-treinados na base de dados *ImageNet*, disponíveis na biblioteca *Keras* do framework *TensorFlow*.

O *dataset* utilizado para o treinamento específico a partir do modelo base foi gerado pelos autores via RD a partir de um gerador randômico de autoria própria

[Valtchev 2021] e o modelo de rede neural escolhido por eles foi uma VGG-16 com a adição de uma camada de 128 neurônios no final. Quanto à divisão dos dados de treinamento foi utilizada a proporção 80-20 nos dados sintéticos de treinamento. Infelizmente a função de ativação utilizada nas camadas adicionadas para o *transfer learning* não foi informada e por essa razão assumiremos que utilizaram a função *ReLu*, que é a padrão para o modelo original de VGG [Simonyan and Zisserman 2014]. O dataset de imagens reais utilizado no trabalho original infelizmente não pôde ser encontrado devido à indisponibilidade no site indicado [Kaggle Inc. 2024], por essa razão foi utilizado o dataset disponível em [Wu 2018]

Como os próprios autores afirmam no trabalho original, redes VGG necessitam de muito poder computacional para serem treinadas. Por essa razão, a *resnet-50* foi escolhida como uma boa substituição a ela, uma vez que, sendo um modelo mais atual e sofisticado, permite um treinamento menos custoso ao mesmo passo que viabiliza o uso de diversas camadas convolucionais. A hipótese desta pesquisa é que a *resnet-50* terá um desempenho similar ou superior à *VGG-16* com um gasto menor de recursos computacionais.

3.2. reimplementando a VGG-16

O objetivo inicial da pesquisa foi a busca pela reimplementação da rede VGG-16 apresentada. Para tal, o primeiro passo foi a obtenção dos datasets de treinamento e teste. Portanto, sempre que for citado o "dataset sintético"este será uma referência ao dataset com 20.000 imagens sintéticas criadas pelos pesquisadores do trabalho original. Como o dataset real utilizado pelos autores não pode ser obtido, utilizou-se do dataset [Wu 2018] com 10.000 imagens, separando 2.000 imagens para teste e 8.000 para treinamento, este será referenciado como "dataset real". A divisão 80-20 de treinamento-teste foi usada em todos os modelos gerados, sendo que os dados de treinamento também foram separados em 80-20 (treinamento e validação, respectivamente) em todos eles.

Este trabalho assumirá como modelo "padrão"uma rede treinada com os mesmos hyper-parâmetros da VGG do artigo original, sendo eles: Uso de *dropout* (30%), transformações nas imagens (rotação, zoom, translação e flip) e *max pooling*. Logo, sempre que uma análise for feita para avaliar a ausência ou não de um desses parâmetros, os outros serão mantidos estáticos.

O modelo de VGG-16 padrão foi reimplementado com sucesso e os dados de comparação com os resultados obtidos no artigo original podem ser visualizados na figura 1. É importante frisar que para cada ponto no gráfico, um modelo distinto foi treinado, para garantir uma análise mais robusta. Nota-se que nosso modelo foi capaz de atingir resultados muito próximos aos originais tanto em treinamento com dados sintéticos quanto com reais. Ainda assim, o desempenho do nosso modelo foi ligeiramente superior ao original quando comparamos o treinamento com dados sintéticos. Essa diferença pode se dar devido a alguma distinção de implementação não identificada. Como o esperado, o desempenho dos modelos treinados com dados sintéticos foi inferior ao treinados com dados reais em cerca de 10%. Essa diferença se dá pela natureza dos dados sintéticos, que de fato possuem menos informação. Ainda assim, o resultado próximo a 90% do melhor caso é considerado satisfatório em comparação aos trabalhos correlatos usando treinamento puramente sintético [Tremblay et al. 2018a].

Também foi realizada uma pequena análise acerca do impacto do uso do *dropout*

e das transformações nas imagens nos modelos VGG treinados com dados sintéticos. Os resultados podem ser visualizados na figura 1. Devido ao fato de essa análise ter sido realizada com apenas um modelo para cada categoria, foi utilizada uma *seed* fixa para tirar a aleatoriedade do modelo e viabilizar o impacto direto do uso desses parâmetros em modelos exatamente iguais. Ainda assim, devido a isso, os resultados não são muito precisos e mais testes poderiam ser feitos futuramente. O artigo original reporta uma distinção da ordem de 0,5% entre o uso ou não do *dropout*, enquanto nosso resultado apontou um valor ligeiramente superior a 1% para o *dropout* e da ordem de 2% para o uso das transformações (o artigo original não analisou o impacto do uso das transformações). Como ambos os parâmetros causaram uma piora da acurácia, o que é o contrário do esperado, há um indicativo de que isso possa ser oriundo de um erro de implementação ou má utilização dos parâmetros que será tratado na seção 4.2. Logo, uma análise mais extensa para descobrir os parâmetros ideais deve ser realizada em um trabalho futuro.

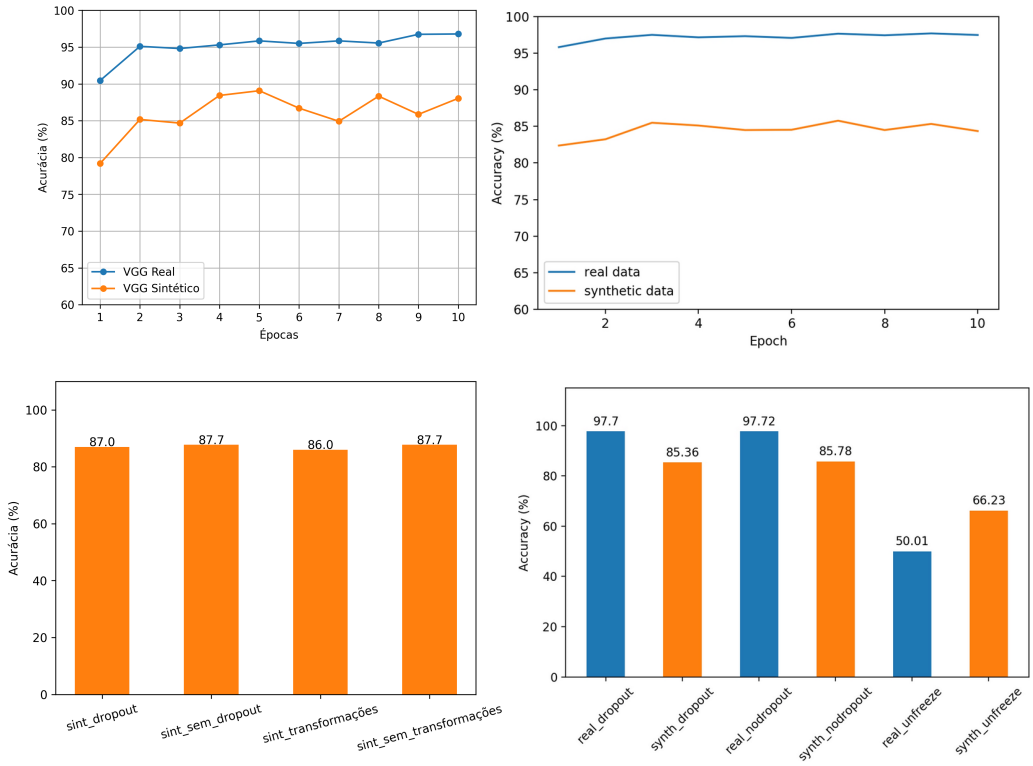

Figura 1. Comparação da nossa VGG_16 padrão com a de Valtchev. Coluna esquerda: superior mostra a acurácia da reimplementação em 10 épocas; inferior, o efeito de dropout e transformações. Coluna direita: superior mostra o desempenho do modelo original; inferior, impacto do uso dos parâmetros de randomização. Fonte: Imagens à direita [Valtchev and Wu 2021]

4. modificações

4.1. Implementando a Resnet-50 padrão

Tendo atingido os resultados esperados ao reimplementar o modelo padrão para a VGG-16, o próximo passo foi implementa-lo para a Resnet-50 e investigar formas de maximizar

sua performance na generalização para dados reais. A comparação da acurácia dos modelos Resnet-50 com os do modelo padrão do artigo original pode ser conferida na figura 2. Devido à alta demanda de recursos computacionais que CNNs demandam para treinamento, todos os testes feitos com a Resnet-50 foram realizados utilizando um número par de épocas de treinamento, totalizando 5 amostras para cada teste, contra 10 amostras utilizadas no artigo original e na reimplementação da VGG-16 padrão.

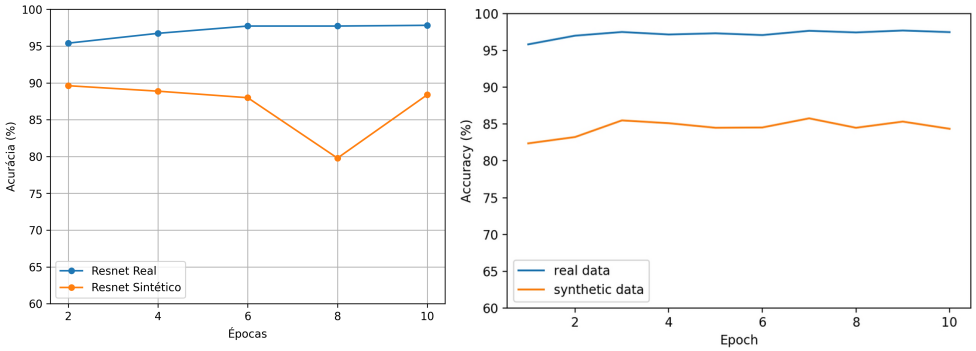

Figura 2. Comparação da nossa Resnet-50 padrão (à esquerda) com a VGG-16 de Valtchev (à direita). Fonte: Imagem à direita [Valtchev and Wu 2021]

Nota-se que nossa *Resnet* padrão atingiu sua melhor acurácia com apenas duas épocas de treinamento quando treinada com dados sintéticos, com uma acurácia de 89% contra os 86% do melhor caso na VGG original, o que indica um bom início, pois a Resnet se mostrou ligeiramente superior antes mesmo da implementação de qualquer mudança significativa da implementação padrão.

4.2. Otimizando a acurácia da Resnet-50

Com o modelo padrão da Resnet implementado, o objetivo foi melhorar seu desempenho ao extrapolar para dados reais. O primeiro objeto de observação foi o *pooling*. Todos os modelos treinados até então utilizaram o *max-pooling* em suas camadas convolucionais, sendo que a última delas foi adicionada após a saída da rede pré-treinada, como uma última camada de *pooling* antes do MLP. Então, em busca de aumentar a acurácia total, foi testado o uso do *average pooling* apenas nesse *pooling* anterior à camada completamente conectada de 128 neurônios criada para o *transfer learning*.

Como pode-se observar na figura 3, o desempenho não obteve um aumento significativo de acurácia ao trocar o *max-pooling* pelo *average-pooling*, visto que nos melhores casos foram obtidas acurácias de 89,6% para o max e 89,4% para o *average*, indicando que a troca do tipo de *pooling* não é um fator relevante quando aplicado apenas nas camadas posteriores à rede pré-treinada.

Outro teste realizado foi a troca da função de ativação no MLP. Como citado anteriormente, os modelos padrão utilizam a ReLu como função de ativação no MLP. Como alternativa a isso, foi realizada a troca pela função Gelu, muito utilizada em arquiteturas mais modernas por consistir em uma curva mais suave do que a ReLu original, apesar de funcionar de maneira similar.

Ao analisar os dados da figura 3, nota-se que, assim como na análise dos tipos de *pooling*, a troca da função de ativação no MLP também não atingiu resultados significativos, considerando o ponto destoante na época 8 do modelo padrão, que provavelmente

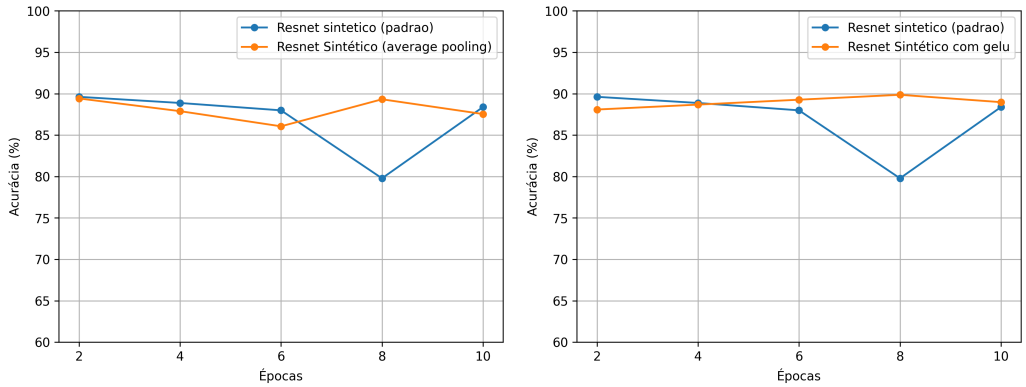

Figura 3. Comparação da Resnet-50 com average e max pooling (à esquerda) e comparção da Resnet-50 com ReLu GeLu (à direita).

advêm do baixo n amostral de 5 utilizado na análise. Portanto, desconsiderando o ponto destoante ambas as análises apontam para o fato de que nem o tipo de *pooling* nem a função de ativação do MLP tem impactos significativos na acurácia final ao extrapolar dados sintéticos para reais.

Após a análise dos dados obtidos pelos modelos padrão e pelas mudanças feitas até aqui, o aumento em quase 2% da acurácia do modelo padrão VGG ao retirar as transformações de pré processamento bem como a queda ao adicionar o *dropout* de 30% se destacaram. Por essa razão foi realizada mais uma bateria de testes com a Resnet-50, mas dessa vez sem o uso das transformações e do *dropout*, para tentar eliminar esse aparente limitador da nossa modelagem.

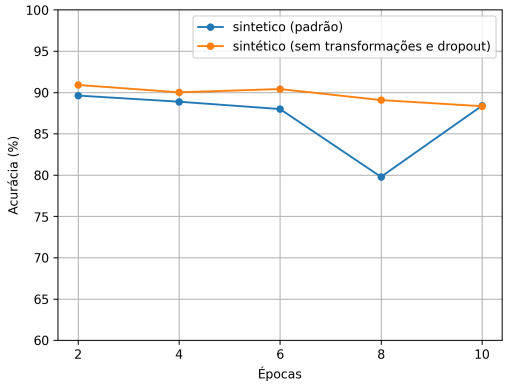

Figura 4. Comparação da Resnet-50 padrão com a sem transformações e dropout.

Os resultados, mostram claramente que este era de fato um limitador para os modelos, visto que, como a figura 4 demonstra, a acurácia no melhor caso para o modelo sintético treinado sem *dropout* e sem transformações atingiu uma acurácia de 90,9% no seu melhor caso, contra os 89,6% do melhor caso do modelo padrão de nossa reimplementação. É importante notar que a diferença de ao menos 1% se mantém consistente com o aumento das épocas, indicando que houve uma melhora real, ainda que pequena, na acurácia do modelo.

5. conclusões

Dado que o objetivo deste trabalho era Reimplementar e melhorar os resultados do artigo original ao implementar um modelo mais recente de CNN, pode-se afirmar que ele foi alcançado, uma vez que, ao analisar o melhor modelo sintético de Resnet-50 obtido neste trabalho, foi atingida uma acurácia máxima de 90,9%, em contrapartida aos 86% do modelo VGG-16 do artigo original. Ainda assim, mesmo considerando apenas os modelos VGG-16, nossa reimplementação atingiu uma acurácia máxima de 89.1%, superando a implementação do artigo original em aproximadamente 3%.

Um fator importante é que foi confirmada a superioridade de eficiência do modelo Resnet sobre o VGG em termos de velocidade, visto que ambas tiveram um desempenho próximo, em termos de tempo de treinamento, apesar do fato de a Resnet-50 possuir aproximadamente 8,8 milhões de parâmetros a mais do que a VGG-16. Para além disso, ao ser treinado com dados reais, nosso melhor modelo Resnet atingiu uma acurácia de 97,8%, que é condizente com o estado da arte para este tipo de classificador, como esperado.

Apesar de os resultados terem sido satisfatórios, ainda se faz necessária a realização de uma análise mais robusta acerca do impacto individual de cada parâmetro de randomização para ambos os modelos, visando uma otimização ainda mais eficiente.

6. Apêndice

O código fonte, assim como todos os modelos treinados e seus respectivos *logs* estão disponíveis no repositório do github: https://github.com/felipnunes/Artigo-Redes-Neurais-Convolucionais.git

Referências

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Kaggle Inc. (2024). Kaggle: Your home for data science. Disponível em: `https://www.kaggle.com`. Acesso em: 31 jul. 2025.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Loquercio, A., Kaufmann, E., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D. (2020). Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 36(1):1–14.

Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M. (2021). Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10912–10922.

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018a). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977.

Tremblay, J., To, T., and Birchfield, S. (2018b). Falling things: A synthetic dataset for 3d object detection and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2038–2041.

Valan, M., Vondráček, D., and Ronquist, F. (2021). Awakening a taxonomist's third eye: exploring the utility of computer vision and deep learning in insect systematics. *Systematic Entomology*, 46(4):757–766.

Valtchev, S. Z. (2021). Synthetic image dataset (cats, dogs, bikes, cars). Disponível em: `https://www.kaggle.com/datasets/zarkonium/synthetic-image-dataset-cats-dogs-bikes-cars`. Acesso em: 24 de agosto de 2025.

Valtchev, S. Z. and Wu, J. (2021). Domain randomization for neural network classification. *Journal of big Data*, 8(1):94.

Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, T. (2018). Cat and dog dataset. Disponível em: https://www.kaggle.com/datasets/tongpython/cat-and-dog. Acessado em: 24 de agosto de 2025.

Yue, X., Zhang, Y., Zhao, S., Sangiovanni-Vincentelli, A., Keutzer, K., and Gong, B. (2019). Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2100–2110.