

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO**

Leonam Teixeira de Vasconcelos

**ESTUDO DE CASO: UTILIZAÇÃO DE REDES ADVERSÁRIAS
GERADORAS PARA SUPER-RESOLUÇÃO DE IMAGENS
CIENTÍFICAS**

Timóteo

2025

Leonam Teixeira de Vasconcelos

**ESTUDO DE CASO: UTILIZAÇÃO DE REDES ADVERSÁRIAS
GERADORAS PARA SUPER-RESOLUÇÃO DE IMAGENS
CIENTÍFICAS**

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Douglas Nunes de Oliveira

Timóteo

2025

Dedico aos meus pais, Léia e Manoel,
e ao meu irmão, Luan.

Agradecimentos

"O homem não é nada além daquilo

que a educação faz dele."

Immanuel Kant

A conclusão deste trabalho é um evento que, quando o comecei, aparentava estar muito distante no horizonte. Conciliar seu desenvolvimento com meu trabalho, e minha carreira em seus altos e baixos, foi uma tarefa árdua – quase tão difícil quanto o desenvolvimento do trabalho em si. Mas aqui estou, redigindo a última seção do documento, algo que antes parecia tão longínquo.

Tal conclusão, deste trabalho e do curso, jamais seriam possíveis sem o fiel e consistente apoio de minha família, portanto devo sem mais delongas deixar meus mais sinceros agradecimentos à minha mãe Léia, meu pai Manoel e ao meu melhor amigo: meu irmão, Luan, que daqui a alguns anos, estará escrevendo um trabalho para este mesmo curso, nesta mesma instituição, neste mesmo molde. Não posso deixar de também agradecer, minha tia Néia, que sempre esteve ao meu lado e ao lado dos meus pais.

Seria injusto eu não dedicar, mesmo que brevemente os agradecimentos a todos os meus amigos, junto com os quais trilhei esse caminho e também sou grato a todo o apoio que me foi concedido pelos meus queridos amigos de trabalho, com quem passo a maior parte do meu tempo, mesmo que de forma virtual. Meus profundos agradecimentos, a todos meus amigos que acompanharam direta, ou indiretamente o esforço e dedicação por trás da conclusão deste curso.

E por último, nada mais justo que deixar meus agradecimentos aos professores e professoras que fizeram parte de minha educação e treinamento como engenheiro de computação. Alguns de seus ensinamentos, ainda uso diariamente no meu dia a dia.

Resumo

O presente trabalho procura validar a viabilidade de treinamentos específicos sob algoritmos de aprendizado de máquina para aumentar a resolução de imagens. A área de inteligência artificial explorada são as redes adversárias geradoras (RAGs), uma tecnologia que durante o início da escrita deste trabalho, era relativamente nova no mercado. Para concentrar os esforços em um sub conjunto palpável de imagens, a super resolução é focada em bases de imagens médicas e astronômicas, ambas sendo obtidas de fontes públicas na internet. Apesar de facilmente encontradas, as imagens requerem tratamento para se adaptarem à estrutura dos modelos. Este tratamento também faz parte dos esforços deste trabalho. Um desafio inevitável e indispensável para alcançar os objetivos, é a integração de diversos softwares de terceiros, necessários para que o treinamento das redes adversárias geradoras aconteça. O trabalho explora diversas fases contendo tarefas mecânicas, tornando automação, uma alternativa pertinente. Os resultados do trabalho mostram que os benefícios de um treinamento específico não são perceptíveis em relação a um treinamento genérico, quando a infraestrutura utilizada para o treinamento é limitada.

Palavras-chave: aprendizado de máquina, super resolução de imagens, redes adversárias geradoras.

Abstract

This work, seeks to validate the viability of specific training for machine learning algorithms, in order to increase images resolutions. The artificial intelligence branch explored in this work are the generative adversarial networks (GANs), a technology that, by the time of this writing, is relatively new to the industry. To concentrate the effort in a tangible image subset, the super resolution is focused in datasets containing medical and astronomical images, both being publicly available on the internet. Although the images can be easily found, they require extra treatment to adapt to the model's structures. This treatment is also accounted for in the efforts of this work. An inevitable and crucial challenge to reach the goals, is to integrate the myriad of third part softwares required for the machine learning training. This work explores several phases containing mechanical tasks, making automation a pertinent alternative. The results show that the benefits of a specific training are not perceptible in comparison with a generic training, when the infrastructure used for the training is limited.

Keywords: machine learning, image super resolution, generative adversarial networks.

Listas de ilustrações

Figura 1 – Diagrama de diferentes áreas da super-resolução de imagens.	20
Figura 2 – Diferentes métodos para super-resolução de imagens.	20
Figura 3 – Diagrama de um neurônio artificial.	21
Figura 4 – Diagrama de uma rede neural ou de um perceptron de múltiplas camadas.	22
Figura 5 – Diagrama de uma rede neural ou de um perceptron de múltiplas camadas com fluxo de dados.	22
Figura 6 – Exemplo de operação de convolução. Os números foram propositalmente colocados de forma repetitiva para facilitar a explicação e os cálculos.	24
Figura 7 – Exemplo de operação de <i>max pooling</i> .	25
Figura 8 – Exemplo de operação de <i>average pooling</i> . Da esquerda para direita, e cima para baixo a operação é realizada sobre toda a matriz.	26
Figura 9 – Diagrama de uma RAG.	28
Figura 10 – Diagrama da SRGAN.	29
Figura 11 – Diagrama do bloco básico da SRGAN.	30
Figura 12 – Diagrama do novo bloco básico da ESRGAN.	30
Figura 13 – Exemplo de rede neural.	33
Figura 14 – Exemplo de rede neural definida em linguagem de alto nível.	33
Figura 15 – Diagrama de fluxo dos procedimentos metodológicos planejados para a realização e avaliação do trabalho e de seus resultados.	36
Figura 16 – Captura de tela da página principal do <i>Cancer Imaging Archive</i> .	43
Figura 17 – Captura de tela do <i>NBIA data retriever</i> .	44
Figura 18 – Código para imprimir informações de um arquivo <i>DICOM</i> .	47
Figura 19 – Saída obtida com a execução do código contido na imagem 18.	47
Figura 20 – Saída obtida com a execução do código contido na imagem 18, destacando o campo de interesse.	48
Figura 21 – Código para converter uma imagem <i>DICOM</i> para o formato <i>jpg</i> .	48
Figura 22 – Estrutura de imagens <i>DICOM</i> .	49
Figura 23 – Destacando o número de frames – ou número de imagens – nos metadados de um arquivo <i>DICOM</i> .	49
Figura 24 – Todas as imagens contidas no arquivo <i>DICOM</i> .	50
Figura 25 – Captura de tela do software <i>Magick-Utils</i> .	51
Figura 26 – Diagrama demonstrando diferença estrutural entre imagem monocromática e colorida.	52
Figura 27 – Diagrama de interação entre as partes envolvidas no modelo.	56
Figura 28 – Captura de tela do buscador de drivers da <i>NVIDIA</i> .	57
Figura 29 – Captura de tela do buscador avançado de drivers da <i>NVIDIA</i> .	58
Figura 30 – Captura de tela do site da <i>NVIDIA Developers</i> .	59
Figura 31 – Captura de tela do filtro do buscador de instaladores CUDA.	59
Figura 32 – Captura de tela da busca de instaladores CUDA.	60

Figura 33 – Captura de tela de instalação CUDA.	60
Figura 34 – Trecho do arquivo de configuração para treinamento e execução da implementação do modelo.	63
Figura 35 – Comando para executar o treinamento.	64
Figura 36 – Diagrama de sintaxe do programa.	65
Figura 37 – Imagem A e imagem B utilizadas no teste.	66
Figura 38 – Amostra aleatoriamente capturada das imagens de ressonância.	70
Figura 39 – Amostra aleatoriamente capturada das imagens de astronomia.	70
Figura 40 – Cálculo de erro MSE para base de dados de ressonância magnética.	72
Figura 41 – Cálculo de erro RMSE para base de dados de ressonância magnética.	73
Figura 42 – Cálculo de erro PSNR para base de dados de ressonância magnética.	74
Figura 43 – Cálculo de erro ERGAS para base de dados de ressonância magnética.	75
Figura 44 – Cálculo de erro MSE para base de dados astronômica.	76
Figura 45 – Cálculo de erro RMSE para base de dados astronômica.	77
Figura 46 – Cálculo de erro PSNR para base de dados astronômica.	77
Figura 47 – Cálculo de erro ERGAS para base de dados astronômica.	78

Lista de tabelas

Tabela 1 – Tabela sumarizando as imagens processadas.	52
Tabela 2 – Tabela de descrição do hardware	54
Tabela 3 – Tabela de resultados da execução do programa.	66
Tabela 4 – Tabela de descrição dos resultados da execução do programa.	66
Tabela 5 – Tabela de resultados da execução do programa com apenas a imagem A. .	67

List of abbreviations and acronyms

BN	<i>Batch Normalization</i> (Normalização em lote, do inglês)
CNN	<i>Convolutional Neural Network</i> (Redes neurais convolucionais, do inglês)
CPU	<i>Central Processing Unit</i> (Unidade de processamento central, do inglês)
CUDA	<i>Compute Unified Device Architecture</i> (Arquitetura de dispositivo de computação unificada, do inglês)
DICOM	Sigla para <i>Digital Imaging and Communications in Medicine</i> , um padrão de comunicação e gerenciamento de imagens para medicina
DVR	<i>Digital Video Recorder</i> (Gravadores de vídeo digital, do inglês)
ERGAS	<i>Erreur Relative Globale Adimensionnelle de Synthèse</i> (Erro adimensional de síntese global relativa, do francês)
ESRGAN	<i>Enhanced Super Resolution Generative Adversarial Networks</i> (Redes adversárias geradoras aprimoradas, para super resolução, do inglês)
ETE	Erro do treinamento específico
ETG	Erro do treinamento genérico
GAN	<i>Generative Adversarial Network</i> (Redes adversárias geradoras, do inglês)
GPU	<i>Graphics Processing Unit</i> (Unidade de processamento de gráficos, do inglês)
JPG	Sigla para <i>Joint Photographic Experts Group</i> , um formato de arquivos
JSON	<i>Javascript Object Notation</i> (Notação de objetos do Javascript, do inglês)
MLP	<i>Multilayer Perceptron</i> (Perceptron de múltiplas camadas, do inglês)
MRI	<i>Magnetic Resonance Imaging</i> (Imagen por ressonância magnética, do inglês)
MSE	<i>Mean Squared Error</i> (Erro quadrático médio, do inglês)
NBIA	<i>National Biomedic Imaging Archive</i> (Acervo nacional de imagens biomédicas, do inglês)
PNG	Sigla para <i>Portable Network Graphics</i> , um formato de arquivos
PSNR	<i>Peak Signal-to-Noise Ratio</i> (Relação sinal-ruído de pico, do inglês)
RAG	Redes Adversárias Geradoras

RMSE	<i>Root Mean Squared Error</i> (Raiz do erro quadrático médio, do inglês)
SRGAN	<i>Super Resolution Generative Adversarial Networks</i> (Redes adversárias geradoras para super resolução, do inglês)
TE	Treinamento Específico
TG	Treinamento Genérico
TPU	<i>Tensor Processing Unit</i> (Unidade de processamento de tensores, do inglês)

Lista de símbolos

α	Letra grega minúscula Alfa
β	Letra grega maiúscula Beta
γ	Letra grega minúscula Gama
μ	Letra grega minúscula Mu ou Mi
Π	Letra grega maiúscula Pi, símbolo matemático de produtório
σ	Letra grega minúscula Sigma
Σ	Letra grega maiúscula Sigma, símbolo matemático de somatório
ω	Letra grega minúscula Ômega
∞	Símbolo matemático para infinito

Sumário

1	INTRODUÇÃO	15
1.1	Justificativa	16
1.2	Objetivos	17
2	REVISÃO BIBLIOGRÁFICA	18
2.1	Trabalhos correlatos	19
2.2	Super resolução de imagens	19
2.3	Redes neurais artificiais	20
2.4	Redes neurais convolucionais	23
2.4.1	Max Pooling	25
2.4.2	Average Pooling	25
2.4.3	A arquitetura da rede	26
2.5	Treinamento da rede	27
2.6	Redes Adversárias Geradoras	28
2.6.1	SRGAN	29
2.6.2	ESRGAN	29
2.7	Equilíbrio de Nash	30
2.8	Métodos de verificação de qualidade de imagens	31
2.8.1	<i>Mean Squared Error (MSE)</i>	31
2.8.2	<i>Root Mean Squared Error (RMSE)</i>	32
2.8.3	<i>Peak Signal-to-Noise Ratio (PSNR)</i>	32
2.8.4	<i>Erreur Relative Globale Adimensionnelle de Synthèse (ERGAS)</i>	32
2.9	Bibliotecas de aprendizado de máquina	32
2.9.1	Pytorch	34
2.9.2	Tensorflow	34
3	PROCEDIMENTOS METODOLÓGICOS	35
3.1	Pesquisa sobre super resolução de imagens (etapa nº 1)	37
3.2	Pesquisa, organização e preparação das imagens (etapa nº 2)	37
3.3	Treinamento e avaliação (etapas nº 3, 4 e 5)	37
3.3.1	Instalação, preparação e configuração do ambiente (etapa nº 3)	38
3.3.2	Execução do treinamento das RAGs (etapa nº 4)	38
3.3.3	Análise da execução do treinamento (etapa nº 5)	38
3.4	Execução dos modelos treinados (etapa nº 6)	38
3.5	Coleta de dados (etapa nº 7)	38
3.6	Construção de gráficos e análise destes (etapa nº 8)	39
4	DESENVOLVIMENTO	40
4.1	Busca e captura de dados para treinamento	41

4.1.0.1	Obtenção de imagens médicas	42
4.1.0.2	Obtenção de imagens astronômicas	44
4.2	Preparação das imagens para treinamento	45
4.2.1	Conversão de imagens médicas para um formato comum	46
4.2.2	Uniformização e redução da resolução	50
4.3	Considerações gerais sobre as imagens pós processadas	52
4.4	Preparação do ambiente para treinamento	53
4.4.1	Descrevendo o hardware disponível	54
4.4.2	Descrição do software necessário para treinar o modelo	55
4.4.2.1	Experimentos preliminares com o ambiente	55
4.4.2.2	Breve descrição sobre versões	56
4.4.2.3	Instalação do <i>driver</i> da NVIDIA	57
4.4.2.4	Instalação do CUDA	58
4.4.3	Apresentação do modelo	61
4.4.3.1	Descrevendo a implementação	61
4.4.3.2	Detalhando a preparação do projeto para treinamento	62
4.4.4	Descrição dos experimentos práticos de treinamento	64
4.4.4.1	Executando o modelo	64
4.5	Coleta de dados	64
4.5.1	Experimentos de similaridade entre imagens	64
4.5.1.1	Exemplo de execução do programa acima	65
4.5.2	Coleta das imagens para avaliação dos resultados	67
4.5.2.1	Bases de dados utilizadas para a coleta de dados	67
4.5.2.2	Programa para extração de estatísticas das imagens	67
4.5.2.3	Análise dos resultados	68
5	RESULTADOS	69
5.1	Amostras das imagens resultantes	70
5.2	Visualização dos resultados	70
5.2.1	Considerações gerais sobre os gráficos de resultados	71
5.2.2	Estudo de resultados envolvendo a base de dados de ressonância magnética	71
5.2.2.1	Erro quadrático médio (MSE)	71
5.2.2.2	Raiz do erro quadrático médio (RMSE)	73
5.2.2.3	Relação sinal-ruído de pico (PSNR)	74
5.2.2.4	Erro adimensional de síntese global relativa (ERGAS)	75
5.2.3	Estudo de resultados envolvendo a base de dados de astronomia	76
5.2.3.1	Erro quadrático médio (MSE)	76
5.2.3.2	Raiz do erro quadrático médio (RMSE)	77
5.2.3.3	Relação sinal-ruído de pico (PSNR)	77
5.2.3.4	Erro adimensional de síntese global relativa (ERGAS)	78
5.3	Considerações gerais sobre o resultado	78
6	CONCLUSÃO	79

7	PROPOSTAS DE TRABALHOS FUTUROS	83
7.1	Uso de RAGs com imagens geográficas	84
7.1.1	Super resolução de imagens de satélites	84
7.1.2	RAGs e drones de baixo custo para mapeamento geográfico	84
7.2	Uso de RAGs para compressão e descompressão de imagens e vídeos	84
7.3	Uso de RAGs para microscopia	84
7.3.1	Super resolução de imagens microscópicas	84
7.3.2	RAGs e microscópios caseiros de baixo custo	85
7.4	Uso de RAGs em monitoramento	85
7.4.1	RAGs e Câmeras de segurança	85
7.4.1.1	RAGs para redução de armazenamento (compressão e descompressão)	85
7.4.1.2	RAGs para captura de detalhes refinados	85
7.4.2	Monitoramento de trânsito	85
7.5	Super resolução para entretenimento	86
7.5.1	RAGs para jogos	86
7.5.2	RAGs para filmes e séries	86
7.6	Super resolução de áudio	86
7.7	Portabilidade do trabalho atual	86
7.8	Estudo sobre a viabilidade de treinamentos pausados	86
	REFERÊNCIAS	88
	Índice	93

1 Introdução

*“As coisas nas quais você pensa
determinam a qualidade da tua mente.”*
Marco Aurélio

1.1 Justificativa

Na era da informação, uma quantidade avassaladora de dados sem precedentes é trafegada de dispositivo a dispositivo a cada segundo. Consumimos e produzimos um volume muito grande de informação. Em média o ser humano produziu cerca de 1,7MB de dados por minuto em 2020 (Vish, 2020) e esse número só tende a aumentar. Uma parte significativa do conteúdo produzido e consumido é em forma de imagens (fotos e vídeos, se pensarmos que os vídeos são apenas imagens sendo reproduzidas em determinada frequência). Se avaliarmos que a cada iteração tecnológica, celulares e câmeras ficam melhores, capturando imagens maiores, com mais qualidade, resolução, alcance dinâmico etc. a variação da quantidade de dados produzidos está em ascensão.

Outro tipo de imagem pouco utilizada pelo público geral e de grande importância para o público científico são imagens de registros históricos, astronômicas e médicas. A razão pela qual estes três gêneros de imagens foram listados em conjunto é simples. Existe um conjunto grande dessas imagens em acervos de fotografia que possuem baixa qualidade, seja pela tecnologia obsoleta utilizada na época ou pela complexidade e custos para capturar uma imagem de mais qualidade. Algumas imagens médicas em particular como Ressonância Magnética, não podem, em alguns casos ser obtidas com uma qualidade maior devido à consequências à saúde do paciente exposto ao equipamento (GUPTA; SHARMA; KUMAR, 2020).

Para ambos os casos citados (número de imagens e vídeos crescendo com o avanço tecnológico e imagens científicas de baixa qualidade) poderíamos utilizar alguma forma de descompressão para aliviar as consequências do aumento absurdo da quantidade de imagens criadas e consumidas e da baixa qualidade das imagens históricas, médicas e astronômicas (SUN; LI, 2019; A..., 2020). No primeiro caso, poderíamos reduzir a resolução das imagens propositalmente para que essas ocupem menos espaço de armazenamento nos dispositivos e menos banda durante as transferências, aumentando inclusive, a velocidade destas. Para visualizar a imagem em uma qualidade mais alta (tanto as imagens propositalmente reduzidas quanto as imagens já com qualidade baixa) se faz necessário então uma forma viável de aumentar a resolução dessas imagens.

O processo de aumentar a resolução de uma imagem é cientificamente conhecido como Super-resolução ou SISR (Single Image Super Resolution, Super-resolução de imagem única, do inglês) e é conhecido como um problema altamente dependente dos dados de entrada e que possui múltiplas soluções visto que para uma imagem de baixa resolução existem várias imagens que podem tê-la gerado (ZHU et al., 2020). Métodos algorítmicos baseados em conceitos matemáticos e sistemas baseados em aprendizado de máquina já existem para esse tipo de tarefa (TAKEMURA, 2013; KHALEDYAN et al., 2020). No entanto, recuperar detalhes finos, como texturas, ao aumentar a resolução de imagens em um fator alto. ainda é um problema remanescente (LEDIG et al., 2017).

Uma técnica relativamente nova introduzida em 2016 por Ian Goodfellow (GOODFELLOW et al., 2014; GOODFELLOW, 2017), faz o uso de várias redes neurais, competindo entre si para que em vez de manipularmos uma entrada e obter um resultado, possamos gerar um resultado novo a partir de aprendizado prévio. Essas redes são chamadas de Redes Gera-

doras Adversárias e foram desenvolvidas para que possamos gerar bases de dados novas e inéditas a partir de dados existentes (GOODFELLOW et al., 2014; MOREIRA, 2019) e já foram utilizadas para super-resolução de imagens, como mostra (MOREIRA, 2019). O trabalho, no entanto, deixa em aberto algumas propostas futuras para que possamos investigar novas formas de avaliar o desempenho deste método, assim como sugestões de novas formas com as quais podemos mensurar e julgar. Seria possível combinar estes trabalhos com o propósito de desenvolver um modelo viável para esse tipo de problema?

Este é o objetivo deste trabalho. Utilizar redes geradoras adversárias, reproduzindo e experimentando os trabalhos de (LEDIG et al., 2017) e (WANG et al., 2018), tendo como base de conhecimento a grande contribuição de (GOODFELLOW et al., 2014) para a ciência da computação e o trabalho de (MOREIRA, 2019) a fim de treinar um modelo onde possamos recuperar qualidade de uma imagem de baixa qualidade. Para seguir a linha de pesquisa dos autores mencionados, o trabalho foca em super-resolução de imagens científicas, mais especificamente, imagens astronômicas e médicas de aparelhos utilizados para exames de imagem, como ressonância magnética.

Dessa forma, é possível verificar o desempenho dessa metodologia no âmbito científico, recuperando ou aprimorando fotos precárias, ou de qualidade indesejável (como mencionado em (SUN; LI, 2019; A..., 2020; GUPTA; SHARMA; KUMAR, 2020)).

1.2 Objetivos

Experimentar os trabalhos de (LEDIG et al., 2017; MOREIRA, 2019; WANG et al., 2018) para a elaboração de uma rede geradora adversária treinada especificamente para a super-resolução de imagens científicas a fim de validarmos os custos e benefícios deste treinamento específico em relação à modelos genericamente treinados.

Mais especificamente, os objetivos do trabalho podem ser summarizados nos tópicos abaixo:

1. Definir um sub-conjunto de imagens, especificamente de uma ou duas áreas para especializarmos o trabalho
2. Definir um modelo de redes geradoras adversárias para o contexto apresentado
3. Integrar todo software e dependência necessários para treinar e utilizar o modelo escolhido no objetivo anterior
4. Utilizar as bases de imagens para treinar o modelo escolhido de forma específica
5. Utilizar-se de técnicas de cálculo de similaridade de imagens para avaliar o desempenho do treinamento realizado

2 Revisão bibliográfica

“Ler, é pensar com uma cabeça alheia”.

Arthur Schopenhauer

2.1 Trabalhos correlatos

As principais fontes para este trabalho se concentram principalmente em livros e artigos recentes, assim como artigos de sites online em alguns pontos específicos. Serão utilizados também alguns materiais mais antigos, especialmente na parte onde estarei fundamentando o conhecimento base (neurônios artificiais, redes neurais etc.). Estes trabalhos e técnicas já estão estabelecidos no meio acadêmico e suas citações são necessárias. Os artigos científicos foram obtidos de três fontes principais: Google Acadêmico, Portal de Periódico CAPES e Arxiv.

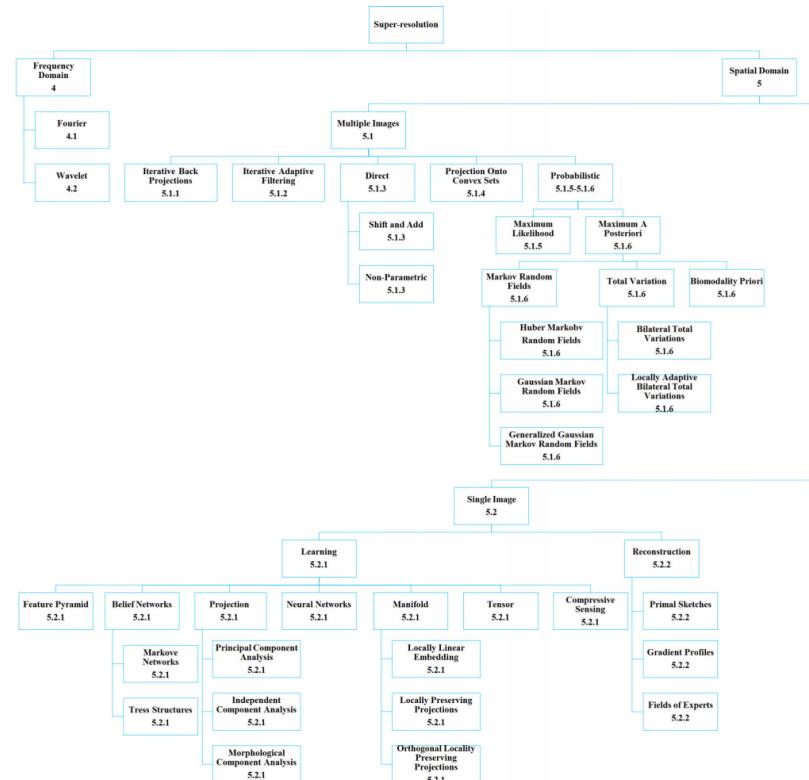
Ledig, et al. (LEDIG et al., 2017) desenvolveram uma arquitetura de redes geradoras adversárias (2.6) que possui desempenho superior à outras técnicas no âmbito de super-resolução de imagens. O trabalho publicado, está gratuitamente disponível no repositório *Arxiv* e será de suma importância para este trabalho. Pode-se dizer que este artigo foi a principal inspiração para o tema. Posteriormente, em 2018, (WANG et al., 2018) resolveu otimizar o modelo gerador proposto por (LEDIG et al., 2016). O autor identificou que para a forma com a qual o artigo inicial calcula os erros da rede geradora, a arquitetura não está otimizada. Foi proposta e implementada então, uma otimização para o modelo, que traz melhor desempenho no treinamento e resultados mais satisfatórios.

2.2 Super resolução de imagens

Super-resolução de imagens, é o processo de, artificialmente, produzir uma imagem de alta resolução a partir de uma imagem de baixa resolução. De acordo com (NASROLLAHI; MOESLUND, 2014), o objetivo da super-resolução de imagens, é aumentar a quantidade de *pixels* por unidade de área em uma imagem, aumentando assim o nível de detalhes se compararmos o resultado, com a imagem original. Ainda de acordo com os autores citados anteriormente, inúmeras aplicações de super-resolução de imagens foram e são exploradas, e diversas técnicas para se performar a super-resolução de imagens são conhecidas hoje em dia. A figura 1 abaixo, mostra as diferentes abordagens disponíveis para realizar super-resolução de imagens.

Como a imagem 1 indica, a super-resolução de imagens, pode ser realizada utilizando diversos artifícios e algoritmos diferentes. Algumas das estratégias utilizadas para tal são rápidas e computacionalmente simples, como filtragem linear, bicúbica e Lanczos, porém produzem resultados de baixa qualidade (LEDIG et al., 2016). Estes métodos simplificam demasiadamente o problema de ampliar a resolução de uma imagem. Métodos envolvendo redes neurais convolucionais têm mostrado grande desempenho para a tarefa de aumentar a resolução de imagens (LEDIG et al., 2016; DONG et al., 2015). Estes métodos aprendem padrões espaciais das imagens de entrada, sendo capazes de extrair detalhes e características da imagem e aplicar nesta, o modelo. A figura 2, traz três exemplos de super-resolução: interpolação bicúbica, rede profunda residual e redes adversárias generativas, da esquerda para direita. Na extrema direita, está a imagem original, para fins de comparação.

Figura 1 – Diagrama de diferentes áreas da super-resolução de imagens.



Fonte: (NASROLLAHI; MOESLUND, 2014)

Figura 2 – Diferentes métodos para super-resolução de imagens.



Fonte: (LEDIG et al., 2016)

2.3 Redes neurais artificiais

Este trabalho é fortemente apoiado em redes neurais. Toda a fundamentação posterior possui-as como base para algo um pouco mais complexo. Portanto, um conhecimento sólido deste tópico, é um requisito indispensável. Redes neurais, de acordo com (HAYKIN, 2007), são definidas como "[...] um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso."(HAYKIN, 2007, p.28). De forma simplória, uma rede neural consiste em componentes simples, que apesar de isolados não se-

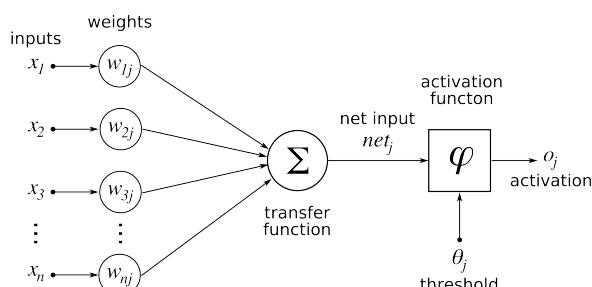
rem capazes de generalizar ou aprender complexidade exorbitante, quando organizados em conjuntos estruturados, contribuem individualmente para que a rede como um todo seja capaz de realizar tarefas mais complicadas.

Estas unidades simples de processamentos são conhecidas como *neurônios artificiais* e a rede neural, é um conjunto destes neurônios interligados de uma maneira específica. Um neurônio pode ser definido como uma estrutura que possui três características básicas: um conjunto de conexões (conhecidas tecnicamente como *sinapses*), uma forma de combinar sinais de entrada e uma função de ativação (HAYKIN, 2007).

As sinapses representam as conexões feitas entre neurônios e compreendem sinais de entradas sendo calibrados pelo peso da conexão. No modelo mais simples de redes neurais, esta calibração é feita em forma de multiplicação, ou seja, um sinal de entrada x_i é multiplicado pelo peso sináptico w_i para obtermos a sinapse i . Como dito anteriormente, um neurônio possui um conjunto de sinapses e os valores computados por estas precisam ser combinados de alguma forma. Uma das formas de calcular tal combinação é fazendo o simples somatório do valor das sinapses, obtendo assim uma saída para aquele neurônio.

A função de ativação modula esta saída de acordo com a necessidade. Talvez o problema a ser resolvido requeira limitar a saída deste neurônio para um intervalo arbitrário, ou talvez converter a saída em um conjunto limitado de valores. A função de ativação é a responsável por esta tarefa. A figura 3, descreve visualmente como um neurônio artificial é representado matemática e computacionalmente. Na imagem, os *inputs* são os sinais de entrada; *weights* são os pesos sinápticos; *transfer function* é o combinador interno (o símbolo grego Σ [sigma] indica um somatório); e *activation function*, representada por φ é a função de ativação. Para implementar esta arquitetura programaticamente, utiliza-se de operações matriciais para calcular a saída do neurônio, em relação às suas entradas ponderadas por seus respectivos pesos.

Figura 3 – Diagrama de um neurônio artificial.



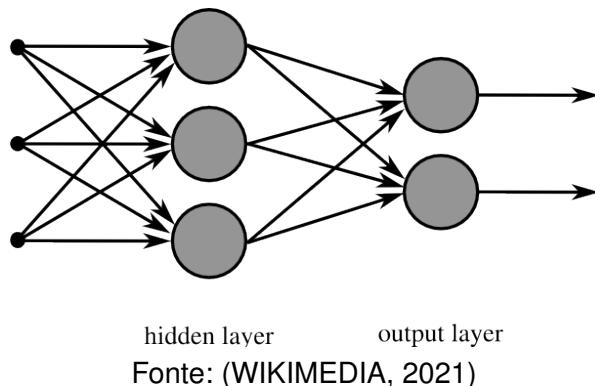
Fonte: (Chrislb, 2005)

Inspirado pela forma como nosso cérebro funciona e computa, o neurônio artificial foi primeiramente desenvolvido em 1943 por McCulloch e Pitts e ficou conhecido como *modelo de McCulloch-Pitts* (MCCULLOCH; PITTS, 1943). Apesar de inovador, o neurônio tem suas limitações. Um único neurônio está limitado a problemas linearmente separáveis (HAYKIN, 2007). Esta limitação trouxe uma estagnação na área de inteligência artificial em torno do

neurônio por alguns anos.

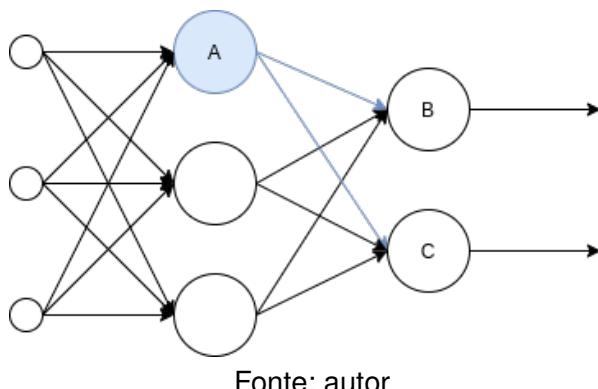
Posteriormente, o neurônio foi organizado em camadas, formando assim, uma rede neural onde agora, os problemas aplicáveis não precisam ser necessariamente linearmente separáveis. A imagem 4 mostra como tal estrutura de neurônios é organizada, onde *hidden layer* é uma camada intermediária, ou uma camada escondida e *output layer* é a camada de saída. Cada círculo representa um neurônio e as setas, a ligação entre estes.

Figura 4 – Diagrama de uma rede neural ou de um perceptron de múltiplas camadas.



O sinal de entrada desta rede, é propagado da entrada para a saída, em forma de uma série de operações matemáticas. O princípio de funcionamento do neurônio continua o mesmo: os sinais de entrada são balanceados pelo peso das ligações, combinados internamente e calibrados pela função de ativação. Agora no entanto, a saída deste neurônio é propagada para todo neurônio que recebe seu sinal de saída como entrada. Vide figura 5. A saída do neurônio *A* é propagada para todos os neurônios que possuem-a como entrada: neurônios *B* e *C*.

Figura 5 – Diagrama de uma rede neural ou de um perceptron de múltiplas camadas com fluxo de dados.



Estas redes foram e são amplamente utilizadas para os mais diversos tipos de classificação e modelagem de dados, porém, por maior que seja sua capacidade de generalização e aprendizado, este modelo em específico é insuficiente para se capturar padrões espaciais

em determinados tipos de dados. Isto é, é possível encontrar padrões entre números mas estes modelos não dependem da ordem da entrada. Caso a entrada seja uma imagem, em um perceptron de múltiplas camadas (considere uma imagem preto e branca representada como uma matriz de $N \times M$ onde cada posição corresponde ao valor do pixel da imagem naquele local), não faz diferença se a imagem em forma de matriz for remodelada em um vetor, retirando assim uma dimensão da estrutura (ZHANG et al., 2021). De forma mais simples, este modelo tem dificuldades em aprender padrões que representam o relacionamento de um dado com seus adjacentes, como um pixel e a forma como este interfere com os pixels ao seu redor e por estes é afetado.

2.4 Redes neurais convolucionais

Redes neurais convolucionais foram desenvolvidas para resolver o problema de aprender o relacionamento entre dados adjacentes e são amplamente utilizadas para o processamento de imagens justamente por terem a capacidade de extrair características úteis para reconhecimento de objetos e classificação (ZHANG et al., 2021). Este modelo possui inspiração na biologia, teoria de grupos entre outros (ZHANG et al., 2021). Nesta seção as redes neurais convolucionais serão introduzidas.

Convolução é um conceito matemático que é calculado entre duas funções f e g e mede o quanto a função g se sobrepõe à função f se g se desloca através de f e é calculada formalmente como (WIKIPEDIA, 2020; WEISSTEIN, 2003; ZHANG et al., 2021):

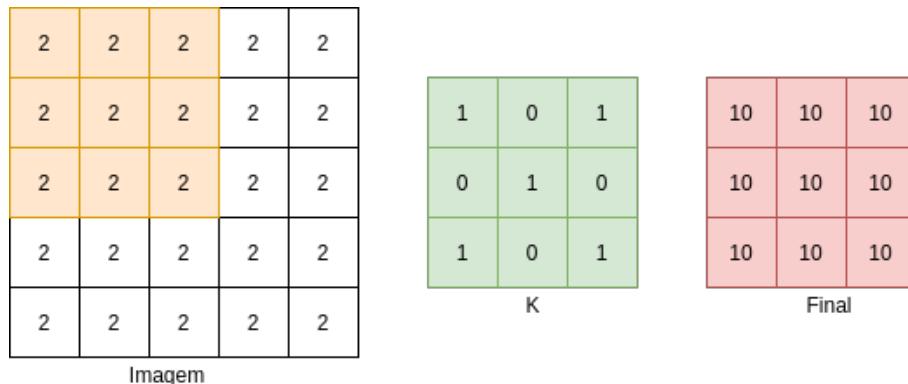
$$(f * g)(x) = \int f(z)g(x - z)dz \quad (2.1)$$

A equação 2.1 pode ser generalizada para múltiplas dimensões, como é o caso de imagens. Imagens em preto e branco podem ser definidas com duas dimensões (largura e altura) enquanto imagens coloridas podem ser definidas com três dimensões (largura, altura e cor).

Para detectar um objeto em específico em uma imagem, considerando que o modelo matemático desta esteja disponível, pode-se calcular a convolução da primeira imagem em relação ao objeto. Assim, os pontos mais propensos de se encontrar o objeto terão um pico na convolução (ZHANG et al., 2021). No caso de imagens coloridas, essas convoluções devem ser adaptadas para trabalhar com canais de cores diferentes (vermelho, verde e azul): os três canais que compõem uma imagem colorida. As convoluções extraem características da imagem e reduzem a complexidade do processamento. De acordo com (ZHANG et al., 2021), camadas de convoluções geralmente requerem menos parâmetros que uma rede completamente conectada, como um MLP.

Para capturar detalhes espaciais de uma imagem, as redes convolucionais conectam um conjunto de pixels adjacentes em um único neurônio da próxima camada. Este neurônio tem um nome especial. É conhecido como *campo receptivo local*. Esta operação que acaba de ser descrita é uma convolução (GULLI; KAPOOR; PAL, 2019). Veja a imagem abaixo:

Figura 6 – Exemplo de operação de convolução. Os números foram propositalmente colocados de forma repetitiva para facilitar a explicação e os cálculos.



Fonte: autor, baseado em (ZHANG et al., 2021)

Na figura 6, a matriz de entrada representa uma imagem e a matriz K é a matriz com a qual iremos fazer a convolução da imagem. Em redes convolucionais (chamaremos de CNN a partir de agora para sermos mais breve. CNN vem do inglês, *convolutional neural networks* que traduz diretamente para redes neurais convolucionais), essa matriz K é chamada de *kernel* ou *núcleo*, em português. Esta matriz é responsável por extrair informações da imagem. No exemplo anterior, da figura 6, durante cada iteração da convolução, sobrepõe-se um subconjunto de pixels da imagem. Na primeira iteração, ela sobrepõe os pixels pintados em amarelo e o resultado desta convolução é 10 ($2*1 + 2*0 + 2*1 + 2*0 + 2*1 + 2*0 + 2*1 + 2*0 + 2*1 = 10$), portanto, na próxima camada, temos uma saída no valor de 10. Na próxima iteração, a matriz núcleo se desloca em um pixel para a direita e faz a mesma operação. Quando ela atinge o final da matriz de imagem, volta ao início e desloca um pixel abaixo, e assim por diante até cobrir toda a imagem. Por meio dessas operações, somos capazes de aproveitar e aprender o relacionamento entre pixels adjacentes, algo que não é tão simples de se fazer com uma rede MLP.

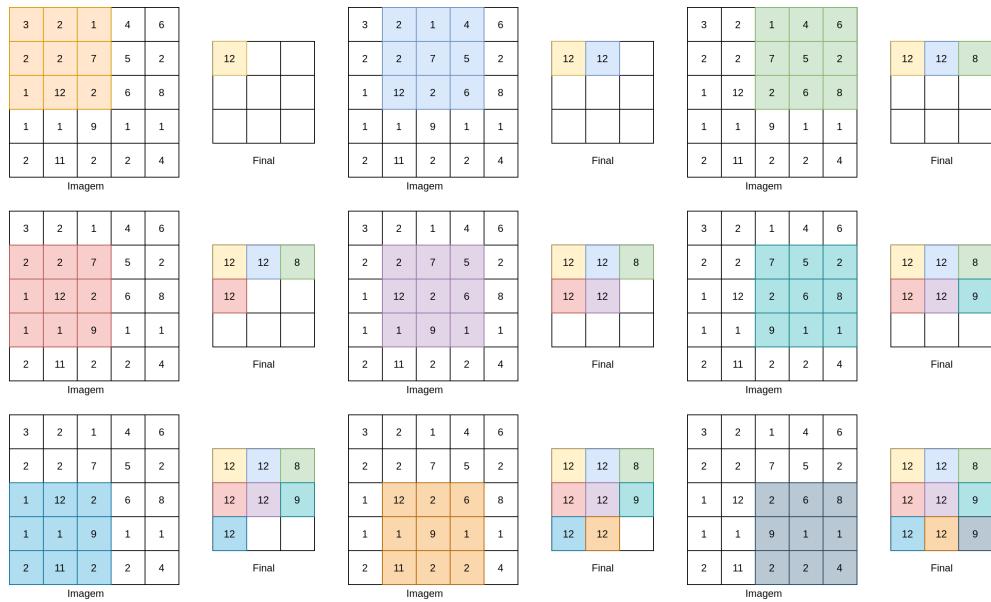
Além de convoluções, uma funcionalidade importante em redes convolucionais, é o *pooling*. Esta operação reduz a complexidade e a resolução das características agregadas e aprendendidas de uma imagem a medida que o processamento da imagem acontece. No final das contas, o objetivo é capturar detalhes específicos da imagem. Estes detalhes, devem ser suficientes para se extrair a informação necessária. Identificar um objeto em uma imagem como um cachorro, uma pessoa, um carro etc. por exemplo. As ultimas camadas da rede devem ser bastante sensíveis à imagem de entrada, de forma que se a alterarmos minimamente, a alteração trará consequências significativas ao resultado final. A saída de uma camada de convolução é chamada de *feature map* (mapa de atributos, do inglês) (ZHANG et al., 2021; BROWNLEE,).

Existem várias formas de se fazer o *pooling*, entre elas temos *max pooling* e *average pooling*.

2.4.1 Max Pooling

Para fazer o *max pooling* de uma matriz, seleciona-se um subconjunto de seus pixels, representando-os como um único dado para a próxima camada: o pixel com o maior valor (ZHANG et al., 2021; BROWNLEE,). O diagrama abaixo (figura 7) ilustra de forma simplória a técnica de *max pooling*. Da esquerda para direita, e cima para baixo a operação opera sobre toda a matriz. Percebe-se que alguns valores aparecem mais vezes. Talvez reduzir o tamanho da matriz de *pooling* (conhecida também como janela de *pooling*) ou normalizar os *pixels* da imagem antes do processamento, seja uma forma de evitar tais repetições excessivas.

Figura 7 – Exemplo de operação de *max pooling*.

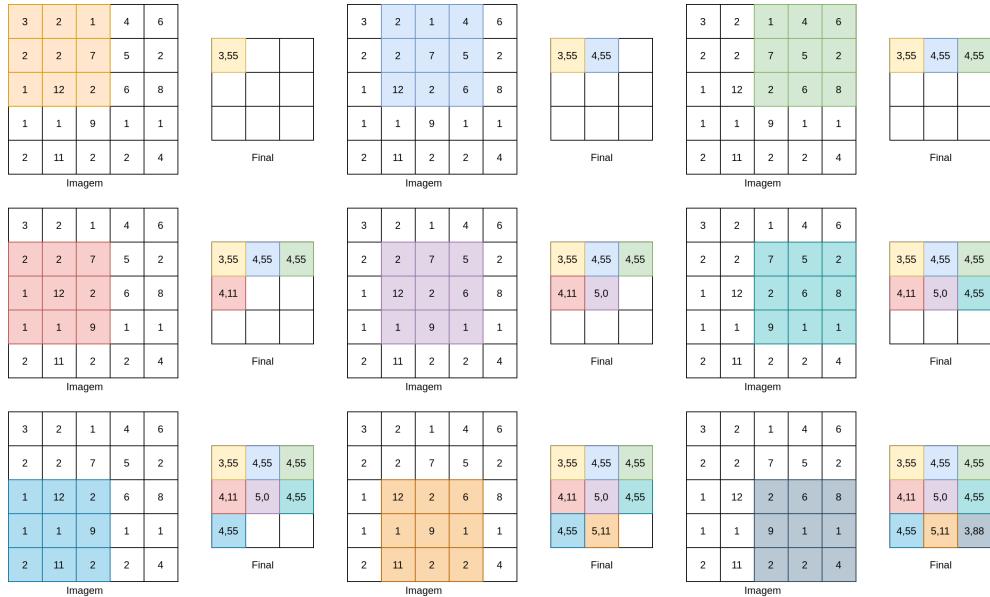


Fonte: autor, baseado em (ZHANG et al., 2021; GULLI; KAPOOR; PAL, 2019)

2.4.2 Average Pooling

Esta operação é bastante parecida com o *max pooling* na forma como é realizada, porém o cálculo é diferente. Em vez de se calcular o valor máximo dos pixels da matriz de entrada, calcula-se a média aritmética (ZHANG et al., 2021; BROWNLEE,). O diagrama abaixo ilustra o que foi dito:

Figura 8 – Exemplo de operação de *average pooling*. Da esquerda para direita, e cima para baixo a operação é realizada sobre toda a matriz.



Fonte: autor, baseado em (ZHANG et al., 2021; GULLI; KAPOOR; PAL, 2019).

2.4.3 A arquitetura da rede

Redes convolucionais podem ser organizadas das mais variadas maneiras existentes para as mais variadas tarefas. Duas formas famosas na literatura, são as redes convolucionais para classificação que terminam completamente conectadas e as completamente convolucionais. A primeira arquitetura consiste em uma CNN tradicional, com as camadas de convolução e *pooling*, porém no final ela se conecta a uma rede completamente conectada (i.e. cada neurônio da camada anterior se conecta a todos neurônios da próxima camada) como um MLP e a partir dele pode classificar o conteúdo de uma imagem, como o exemplo famoso de detectar se o objeto da foto é um cachorro ou um gato. Nesse caso, a CNN irá extrair das imagens as características necessárias para avaliar o que o objeto é. Com as características extraídas, estas são alimentadas na rede completamente conectada e esta por sua vez faz a classificação.

A segunda arquitetura não envolve camadas completamente conectadas. Neste modelo, a rede mantém a estrutura da CNN até a última camada. Um exemplo de aplicação desta arquitetura, é a segmentação de imagens. Aproveitando o caso de uso da arquitetura anterior, suponha que em vez de identificar se na imagem há um gato ou um cachorro, o objetivo seja descobrir onde especificamente, naquela imagem está o cachorro. Para esta funcionalidade, a rede precisaria de não apenas detectar as características da imagem e identificar o cachorro, como também precisaria demarcar o animal na imagem. Para isso, essas redes utilizam as características extraídas nas camadas iniciais para reconstruir a imagem nas camadas finais, destacando o objeto em questão (KUMAR, 2020).

Para este trabalho, ambas as arquiteturas serão utilizadas. Nas próximas seções, ficará mais claro a aplicação de cada uma destas.

2.5 Treinamento da rede

Para que redes neurais aprendam, estas precisam ser submetidas a um processo de treinamento, que no caso de redes neurais, representa uma técnica de ajuste dos pesos sinápticos, a fim de reduzirmos o erro da rede. Existem várias formas de treinar uma rede neural, e uma das mais famosas é a retro propagação do erro, conhecida popularmente pelo termo em inglês *backpropagation* (RUMELHART; HINTON; WILLIAMS, 1986). Como o nome sugere, o treinamento é realizado perante à disseminação do erro obtido pela rede, de trás pra frente. Isto é, para ajustar os pesos da rede utilizando este algoritmo, insere-se na rede uma entrada da qual a saída esperada é conhecida. Calcula-se então, a diferença entre a saída esperada e obtida. Os pesos em seguida, são atualizados proporcionalmente a este erro, levando também em consideração o quanto cada peso interfere no erro, da camada mais profunda, para a camada menos profunda (RUMELHART; HINTON; WILLIAMS, 1986). A fim de se otimizar este treinamento, existem ferramentas matemáticas para atualizarmos os pesos de formas mais eficientes.

Em 1944, um matemático chamado Haskell B. Curry, descreveu a aplicação de um método já conhecido para a otimização de problemas, que posteriormente ficou conhecido como *Gradient Descent* (CURRY, 1944). Este método utiliza um conceito matemático chamado gradiente, para encontrar a direção mais eficiente para o mínimo de uma função. O gradiente é um vetor de n componentes, onde n é a dimensão da função a ser otimizada, que no caso do treinamento, é a função que representa o erro da rede neural, em função de todos seus parâmetros treináveis, como seus pesos. Quando o gradiente desta função é calculado, sabe-se, exatamente em qual direção os erros devem ser atualizado para se alcançar o mínimo, o mais rápido possível (com o menor número de iterações de treinamento), diminuindo assim, a complexidade e o tempo necessários para o treinamento.

Retro-propagação do erro em conjunto com o *Gradient Descent* compõem o algoritmo ótimo de treinamento. Com a retro-propagação, altera-se o peso individual, proporcionalmente ao quanto este, interfere no erro. O gradiente então, fornece o caminho mais rápido para minimizar este erro. Para que estas operações sejam realizadas, uma série de cálculos matemáticos são realizadas a cada etapa de treinamento, fazendo com que a complexidade desta fase crucial no desenvolvimento de redes neurais seja uma das mais demoradas e computacionalmente exaustivas.

Apesar de o tópico de treinamento de redes neurais ser de suma importância nos estudos de inteligência artificial, é externo ao objetivo deste trabalho, descrever em fino detalhe tal atividade. As bibliotecas de inteligência artificial amplamente utilizadas na indústria, que serão utilizadas na parte prática do presente trabalho, abstraem grande parte da complexidade matemática e algebraica do treinamento. Portanto, maiores detalhes sobre treinamento de redes neurais serão omitidos.

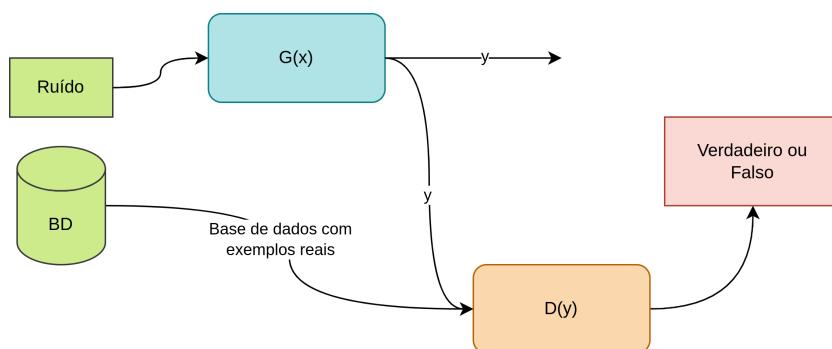
2.6 Redes Adversárias Geradoras

Uma forma relativamente nova na indústria para gerar dados, é a utilização das *redes geradoras adversárias*. O conceito foi introduzido em 2014 por Ian Goodfellow (GOODFELLOW et al., 2014). As redes geradoras utilizam o princípio da competição para produzir os resultados e para disso, utiliza (pelo menos na arquitetura inicial proposta) duas redes neurais, competindo entre si para gerar quaisquer forma de dados que sejam aplicáveis à esse tipo de modelo.

O modelo consiste de duas redes neurais denominadas *geradora* ($G(x)$) e *discriminadora* ($D(y)$). A geradora, como o próprio nome diz gera o dado em questão: imagens, texto, música entre outros (C. Han et al., 2018; XU et al., 2018; OZA; VAGHELA; SRIVASTAVA, 2019).

A rede discriminadora, tem uma função simples: validar o que a geradora produz. Esta rede detecta se o que foi produzido pela geradora é falso ou não. Seja o problema de se implementar uma RAG (sigla para redes geradoras adversárias) capaz de gerar imagens de pôr do sol. A rede discriminadora será treinada para identificar o que é e o que não é uma imagem de pôr do sol enquanto a rede geradora será treinada para a enganar. Taí a competição: enquanto a rede D fica melhor em detectar se uma imagem é falsa ou não, a rede G fica melhor em enganar a rede D a achar se que o que ela está produzindo, é uma imagem verdadeira de pôr do sol.

Figura 9 – Diagrama de uma RAG.



Fonte: autor, baseado em (GULLI; KAPOOR; PAL, 2019; ZHANG et al., 2021; MOREIRA, 2019).

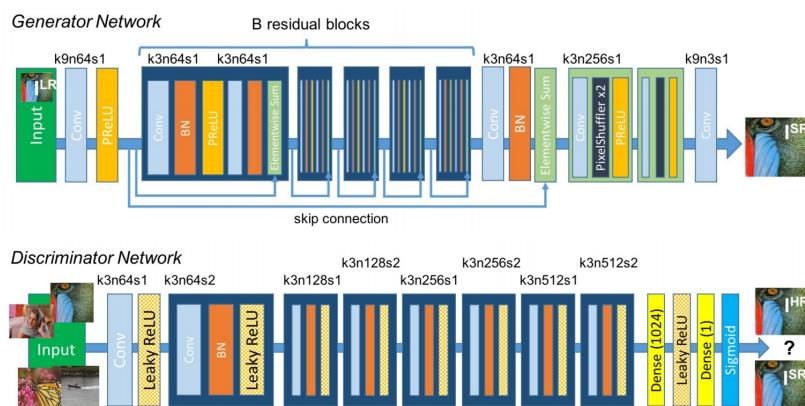
De acordo com a imagem 9, a saída da rede geradora alimenta a rede discriminadora, assim como dados reais também entram nesta. Estes são utilizados para treiná-la a diferenciar com cada vez mais precisão um dado falso de um verdadeiro (ou, gerado de real). No caso de super-resolução de imagens, estes dados são em forma de imagens. A geradora é treinada para converter ruído em imagens. Imagens estas, que por sua vez sejam capazes de enganar a discriminadora a classificar se a imagem gerada é uma imagem verdadeira. Uma vez que as imagens artificialmente geradas são capazes de se passarem por imagens verdadeiras (entenda *se passarem por imagens verdadeiras* como serem identificadas como imagens reais), a rede geradora estará produzindo imagens semelhantes às imagens nas quais a rede discrimi-

nadora foi treinada para identificar. O termo "semelhante", utilizado da forma como foi utilizado, pode ser subjetivo. Para os olhos humanos, é natural e intuitivo avaliar se duas imagens são parecidas ou não. No contexto de modelos de redes neurais, a semelhança é avaliada até onde o modelo consegue capturar e reproduzir padrões.

2.6.1 SRGAN

SRGAN é a arquitetura de GAN proposta para fazer super-resolução de imagens. Ela utiliza duas redes (discriminadora e geradora), montadas individualmente em arquiteturas específicas para obter o máximo das redes na super-resolução. O diagrama abaixo (10) mostra como a rede é organizada. O nome das camadas são dados por $k< x >n< y >s< z >$, onde x é o tamanho do kernel, y é o número de *feature maps* (i.e. o tamanho da saída da camada convolucional) e s é a sigla para *stride*, que representa o passo que a janela de pooling vai se deslocar em cada iteração (e.g. com $s=1$, a janela de *pooling* vai se deslocar de pixel a pixel)

Figura 10 – Diagrama da SRGAN.



Fonte: (LEDIG et al., 2017)

Neste modelo, a discriminadora é treinada para avaliar se a imagem é uma imagem original de alta resolução ou uma imagem super-resolvida artificialmente (LEDIG et al., 2017) enquanto a geradora (essa parte difere um pouco da arquitetura original proposta por (GOODFELLOW et al., 2014)) é treinada para processar uma imagem de baixa resolução (anteriormente, no lugar da imagem em baixa resolução, a entrada era ruído), processá-la, e obter uma imagem de maior resolução na saída.

2.6.2 ESRGAN

A SRGAN, obteve grande desempenho em relação aos outros modelos testados porém, (WANG et al., 2018) detectou alguns pontos na arquitetura que podem ser otimizados. Perceba, que na figura 10, a rede geradora é formada por um conjunto de blocos básicos (*B residual blocks*). Esses blocos, internamente são formados por uma série de camadas e entre elas temos uma camada representada pela cor laranja, identificada como BN. Vide figura 11.

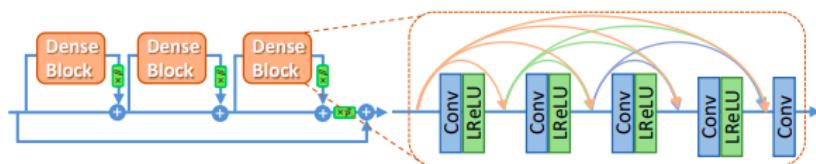
Figura 11 – Diagrama do bloco básico da SRGAN.



Fonte: (LEDIG et al., 2017)

A camada *BN* da imagem 11 performa uma normalização na saída da camada anterior chamada *Batch Normalization*. Esta normalização reduz alguns problemas existentes no treinamento de redes neurais, performando uma normalização não a nível de amostra, mas a nível de *batches*, que são um conjunto de dados treinados em sequência numa rede. De acordo com (WANG et al., 2018), esta normalização não é eficiente quando o modelo está lidando com otimização de dados utilizando relação sinal-ruído. Para tal, (WANG et al., 2018) propõe um novo modelo de bloco básico e um novo modelo de rede discriminadora. A figura 12 ilustra o novo bloco básico proposto, que elimina completamente as camadas de normalização de *batch*.

Figura 12 – Diagrama do novo bloco básico da ESRGAN.



Fonte: (WANG et al., 2018)

2.7 Equilíbrio de Nash

Equilíbrio de Nash é um caso estudado em teoria dos jogos, onde os oponentes (em um jogo não cooperativo), não se sentem mais incentivados em continuar com suas jogadas (CHEN, 2022; ELDRIDGE, 2022).

Quando redes geradoras adversárias são treinadas, como explicado anteriormente, as redes neurais do modelo competem entre si. O objetivo do treinamento é encontrar o equilíbrio de Nash entre as redes competidoras. De acordo com Goodfellow (GOODFELLOW, 2017), isso torna as RAGs mais desafiadoras de se trabalhar do que uma rede neural clássica, onde geralmente procura-se, apenas otimizar uma função.

2.8 Métodos de verificação de qualidade de imagens

Apesar da qualidade de uma imagem poder ser tratada como algo subjetivo do ponto de vista da fotografia como arte, existem formas de a avaliarmos quantitativamente. Neste trabalho há um interesse particular em avaliar a proximidade, ou similaridade entre imagens. Considere as seguintes assunções:

1. Seja **A** uma imagem de alta qualidade
2. Seja **B** a versão comprimida e reduzida da imagem **A**
3. Seja **C** o resultado da super resolução da imagem **B** por uma **ESRGAN**

Tenha em mente, que as assunções acima se aplicam somente ao escopo deste trabalho. Os métodos estatísticos e quantitativos de calcular a similaridade entre imagens devem, em sua maioria calculá-la independentemente das imagens em questão, sejam versões comprimidas uma da outra ou sejam imagens completamente distintas e não relacionadas.

Para de verificar a eficiência e confiabilidade da super resolução obtida na imagem **C**, precisa-se de avaliar a similitude, ou disparidade entre a imagem **A**, original e descomprimida, e a imagem **C** super resolvida, após a compressão.

Existem vários métodos estatísticos que podem ser utilizados para se calcular a semelhança matemática entre duas imagens. Dessa forma, é possível medir numericamente o quanto próximas são as imagens **A** e **C**.

Abaixo, alguns destes métodos estão descritos. Leve em consideração, que os nomes e siglas serão deixados como são mais conhecidos, e em consequência mais facilmente encontrados. Em sua maioria, os termos originais são em inglês, porém uma tradução está apresentada na descrição.

Existem diversas bibliotecas disponíveis para calcular estes índices de similaridade, como a biblioteca de *Python*, *sewar* (KHALEL, 2023). Uma biblioteca de código aberto que implementa diversos índices de maneira simplória e minimalista.

2.8.1 Mean Squared Error (MSE)

O *Mean Squared Error* (Erro quadrático médio, do inglês), pode ser definido da seguinte forma entre duas imagens (É bastante comum encontrar o termo *sinal* no lugar de *imagem*) **x** e **y**:

$$MSE(x, y) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.2)$$

No MSE, quanto menor o valor, menor o erro entre as duas imagens e em consequência, maior a similaridade.

2.8.2 Root Mean Squared Error (RMSE)

O *Root Mean Squared Error* (Raiz do erro quadrático médio, do inglês) é uma variação do MSE(2.8.1) e pode ser definido da seguinte forma entre duas imagens (Assim como o MSE(2.8.1), é bastante comum encontrar o termo *sinal* no lugar de imagem) \mathbf{x} e \mathbf{y} :

$$RMSE(x, y) = \sqrt{\sum_{i=1}^n \frac{(y_i - \hat{y}_i)^2}{n}} \quad (2.3)$$

No RMSE, assim como no MSE(2.8.1) quanto menor o valor, menor o erro entre as duas imagens e em consequência, maior a similaridade.

2.8.3 Peak Signal-to-Noise Ratio (PSNR)

O *Peak Signal-to-Noise Ratio* (WANG et al., 2004) (Relação sinal-ruído de pico, do inglês) pode ser definido da seguinte forma entre os sinais ou imagens \mathbf{x} e \mathbf{y} :

$$PSNR(x, y) = 10 \cdot \log_{10} \left(\frac{MAX_x^2}{MSE(x, y)} \right) \quad (2.4)$$

No PSNR, quanto maior o valor, maior a similaridade entre as imagens.

2.8.4 Erreur Relative Globale Adimensionnelle de Synthèse (ERGAS)

O *Erreur Relative Globale Adimensionnelle de Synthèse* (WALD, 2000) (Erro adimensional de síntese global relativa, do francês) pode ser definido da seguinte forma entre os sinais ou imagens \mathbf{x} e \mathbf{y} :

$$ERGAS(x, y) = 100 \cdot \frac{h}{l} \sqrt{\frac{1}{N} \sum_{i=1}^N \frac{RMSE(x, y)^2}{M(x, y)^2}} \quad (2.5)$$

No ERGAS, quanto menor o valor – mais próximo de zero – maior a similaridade.

2.9 Bibliotecas de aprendizado de máquina

Considerando apenas a parte matemática por trás de toda esta teoria que fundamenta todo o meio de inteligência artificial e aprendizado de máquina, existe por si só, uma complexidade gigantesca. Implementar modelos e algoritmos populares do zero, tomaria um tempo grande e está sujeito a diversos problemas caso testes específicos não sejam propriamente realizados.

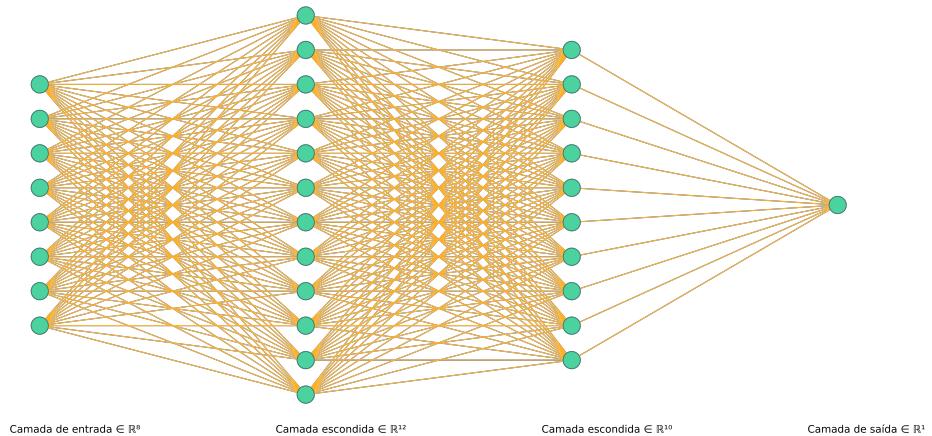
Dada tamanha dificuldade, bibliotecas e *frameworks* de matemática computacional voltadas para aprendizado de máquina foram desenvolvidos por pesquisadores(as) e engenheiros(as) e disponibilizados ao público geral. Estas ferramentas, abstraem algoritmos e estruturas de uma maneira a eliminar esta complexidade do desenvolvimento de modelos. Isto

possibilita que as pessoas foquem seus esforços em desenvolver o modelo mais apropriado e otimizado para o contexto, ao invés de se preocuparem com problemas já resolvidos e bastante testados.

Diversas bibliotecas estão disponíveis para tais aplicações, muitas delas são de código aberto e têm a visibilidade de uma grande comunidade. Isso permite com que problemas sejam encontrados com antecedência e sugestões de melhoria sejam constantes. As duas ferramentas mais populares durante a escrita deste trabalho para aprendizado de máquina e inteligência artificial são *Pytorch* e *Tensorflow*.

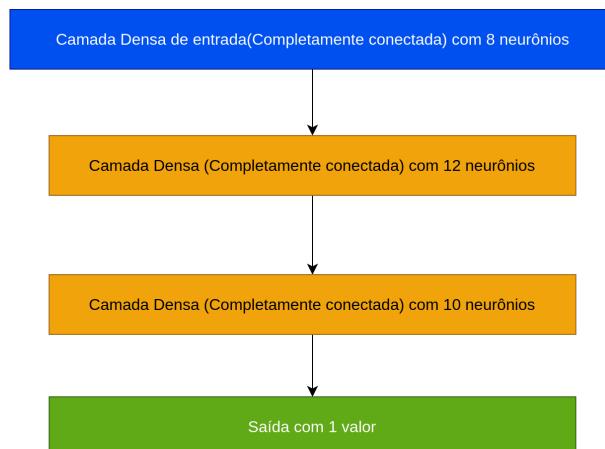
Ambas bibliotecas permitem criar uma rede neural como exemplificado na imagem 13 em uma linguagem de alto nível, como o diagrama 14 mostra.

Figura 13 – Exemplo de rede neural.



Fonte: Autor, gerada por (LENAIL, 2023).

Figura 14 – Exemplo de rede neural definida em linguagem de alto nível.



Fonte: Autor.

Além de dar acesso à uma maneira relativamente simples de desenvolver modelos menos sofisticados, como demonstrado nas figuras 13 e 14, o *Tensorflow* também fornece acesso à outros níveis de abstração. Desde o mais baixo, onde pessoas que realmente entendem a biblioteca a fundo têm controle total, até o mais alto, onde utilizamos de objetos (como uma camada) previamente desenvolvido e testado, como blocos para construir os modelos finais.

2.9.1 Pytorch

Pytorch, é uma biblioteca de computação de tensores otimizada, baseada na biblioteca *Torch*, capaz de executar em CPUs e em GPUs (PYTORCH, 2023). Foi originalmente desenvolvido pela atual empresa Meta, porém hoje, em código aberto, faz parte da fundação Linux (ZEMLIN, 2022).

Alguns produtos famosos que fazem uso do *Pytorch* em seu software são o auto-piloto dos carros da *Tesla* (KARPATHY, 2019); *Pyro*, uma linguagem de programação probabilística desenvolvida pela empresa *Uber* (GOODMAN, 2017), entre outros.

2.9.2 Tensorflow

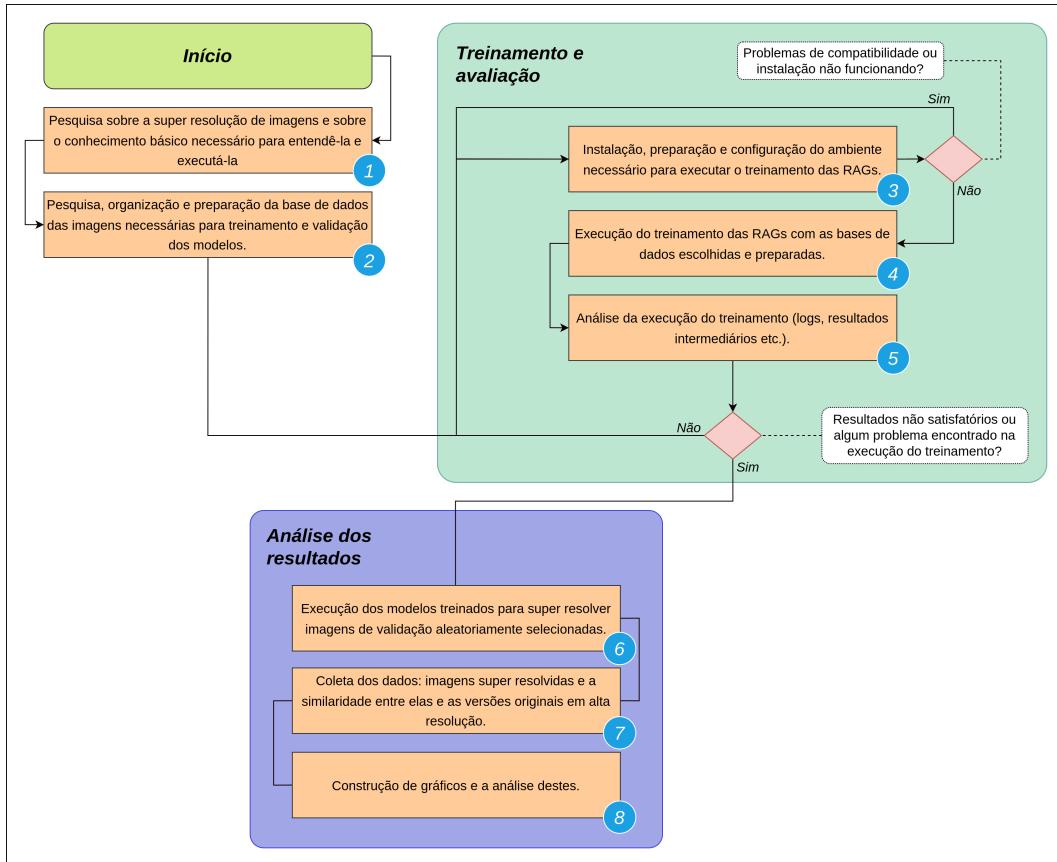
Tensorflow é uma biblioteca inicialmente desenvolvida pelo *Google* como parte do projeto *Google Brain*. Foi disponibilizada como código aberto em 2015 (JEFF; MONGA, 2015; ABADI et al., 2015). Diversas aplicações utilizam o *Tensorflow*, como o *Google RankBrain* (DAVIES et al., 2020), um motor de busca baseado em aprendizado de máquina. O *Spotify*, utiliza *Tensorflow* para recomendações e busca de música para seus usuários (NGAHANE; BAER, 2019).

3 Procedimentos metodológicos

“Sem música, a vida seria um erro.”

Friedrich Nietzsche

Figura 15 – Diagrama de fluxo dos procedimentos metodológicos planejados para a realização e avaliação do trabalho e de seus resultados.



Fonte: autor.

O presente trabalho pode ser classificado quanto a diversos aspectos. Em termos de objetivos, a pesquisa se classifica como *pesquisa exploratória*, pois conceitos e modelos existentes são utilizados e experimentados a fim de se elaborar uma aplicação na qual poderemos desfrutar dos modelos para um subconjunto de imagens. Nenhum novo modelo foi proposto. Quanto aos procedimentos técnicos, esta pesquisa se classifica como *pesquisa experimental*. Apesar de a coleta e análise de resultados estatísticos minuciosos e rebuscados não serem o foco principal do trabalho, algumas métricas são adotadas, medidas e analisadas. Portanto, com a característica principal do trabalho sendo a de experimentar sobre o trabalho anterior de terceiros, a pesquisa experimental se encaixa mais confortavelmente com o escopo do trabalho.

Nesta parte do trabalho é definido o fluxo no qual a proposta foi implementada, desde o princípio majoritariamente teórico, até a parte mais prática, onde é testado de fato, modelos de redes geradoras adversárias. O diagrama da figura 15 ilustra as fases e as subseções em sequência, detalham um pouco mais a fundo a intenção e objetivos de cada uma das fases.

3.1 Pesquisa sobre super resolução de imagens (etapa nº 1)

Esta fase representa toda a pesquisa não necessariamente planejada que ocorreu anteriormente e durante as fases práticas de experimentação. Ela envolveu desde questões teóricas sobre as tecnologias desenvolvidas como resoluções de dúvidas e problemas que foram encontrados no caminho de pesquisa e desenvolvimento. Durante as fases finais, uma quantidade grande de dúvidas, naturalmente surgiu. Esta etapa da metodologia engloba também as pesquisas para sanar tais dúvidas de implementação, implantação, testes e validação, não se restringindo apenas à pesquisa preliminar do trabalho.

3.2 Pesquisa, organização e preparação das imagens (etapa nº 2)

Dados são requisitos básicos para se treinar modelos de redes neurais. No caso deste trabalho, especificamente imagens científicas de áreas médica e astronômica são utilizadas. Existem bases de dados famosas e acessíveis para ambas as aplicações como as disponíveis nos sites *OpenfMRI* e no *Catálogo de dados do governo dos Estados Unidos*. O trabalho desta etapa consistiu na classificação e segregação das imagens obtidas nestas fontes e preparação para treinar e alimentar os modelos.

Esta preparação consistiu em algumas etapas:

1. Dimensionar as imagens de acordo com a necessidade e recursos disponíveis. Imagens maiores requerem mais poder computacional para serem processadas e mais tempo de treinamento dos modelos. No caso deste trabalho, ambos são limitados, portanto a resolução das imagens precisa ser reduzida.
2. Recortar as imagens para uma resolução consistente, i.e., todas imagens com mesma altura e mesma largura. Esta consistência é importante, pois a dimensão dos parâmetros da rede convolucional não se adapta ao tamanho da imagem de entrada.
3. Gerar imagens de baixa resolução a partir das imagens obtidas nas fases 1 e 2. O conjunto das imagens dimensionadas e recortadas com as imagens de baixa resolução formou a base de dados para o treinamento da RAG. Estas imagens com resolução reduzida, devem ser alimentadas ao modelo para que este produza uma imagem de resolução alta. Estas imagens são, posteriormente comparadas com as imagens originais para tomar-se conhecimento da eficiência e precisão do método utilizado.

No fim da preparação destes dados, que não são dedicados especificamente ao uso em redes adversárias geradoras, uma nova base de dados nasce, preparada para este estilo de modelo.

3.3 Treinamento e avaliação (etapas nº 3, 4 e 5)

Repare que no diagrama (15) esta fase consiste em duas sub-etapas. Estas sub-etapas são repetidas caso o resultado não seja satisfatório. Eventualmente algum parâmetro de trei-

namento pode ser alterado a fim de obtermos um resultado idôneo.

3.3.1 Instalação, preparação e configuração do ambiente (etapa nº 3)

Nesta etapa, todo software necessário para o treinamento e execução dos modelos são instalados e configurados, atentando-se aos diferentes níveis de compatibilidades entre as variadas partes envolvidas. Esta fase é pruna a muitos problemas de equivalência entre versões de softwares e justamente por isso, erros são comuns nas execuções. Estas dificuldades são as razões por trás do laço superior nesta fase, retornando para ela mesma: caso a instalação dê errado, uma nova tentativa, com configurações diferentes é realizada.

3.3.2 Execução do treinamento das RAGs (etapa nº 4)

Nesta etapa, executa-se e experimenta-se com as redes generativas adversárias escolhida. É colocado em prática o que foi definido até então, somente na teoria. Em alguns trechos desta etapa é necessário deixar o modelo treinando e esta etapa consome bastante tempo. Por este motivo, é preciso prestar bastante atenção nas limitações de recursos durante a etapa de preparação das imagens (seção 3.2).

Modelos existentes foram utilizados. Tais modelos estão disponíveis em repositórios de aprendizado de máquina e são treinados em bases de imagens genéricas, como fotos cotidianas, imagens de animais, paisagens etc. Estes modelos existentes foram treinados, neste trabalho, com a mesma técnica dos modelos especificamente treinados para o objetivo deste trabalho: verificar a qualidade com a qual ele produz imagens médicas e astronômicas de alta resolução, mesmo não sendo treinado especificamente para este contexto e escopo.

3.3.3 Análise da execução do treinamento (etapa nº 5)

Aqui, são julgados, os treinamentos obtidos pelas experimentações, e de acordo com estes, a decisão de prosseguir ou regredir alguns passos para refazer experimentos, é tomada. Com um treinamento satisfatório, pula-se para a próxima fase. Caso contrário, a fase anterior é executada novamente, calibrando os parâmetros de treinamento. Todo o processo de treinamento e seus dados colhidos são documentados, independentes se satisfazem ou não os objetivos. Esta fase pode também retornar à fase de número 3, caso os resultados não estejam de acordo com o esperado.

3.4 Execução dos modelos treinados (etapa nº 6)

Nesta etapa, os modelos que foram treinados com sucesso nas etapas anteriores, são executados com as imagens obtidas na etapa de organização (seção 3.2).

3.5 Coleta de dados (etapa nº 7)

Aqui, os dados são coletados: imagens que foram super resolvidas e também a similaridade entre estas e suas respectivas versões originais em alta resolução. Para tornar esta fase

mais eficiente, scripts são utilizados para não só coletar os dados, como também analisá-los e organizá-los.

3.6 Construção de gráficos e análise destes (etapa nº 8)

Com um volume grande de imagens, sem uma maneira visual de compreender os resultados, pouco pode-se dizer sobre os resultados. Então nesta fase, os dados coletados na etapa anterior (seção 3.5) são plotados, visualizados e comentados.

4 Desenvolvimento

“O início é a parte mais importante do trabalho.”

Platão

Esta parte do trabalho é dedicada à descrição da execução do procedimento metodológico. Aqui, descreve-se todo o desenvolvimento do trabalho, com o máximo de detalhes possível, desde a obtenção dos dados, até a coleta e análise dos resultados. A intenção final desta seção é descrever uma receita, que se seguida passo a passo, o resultado obtido será equivalente aos resultado deste trabalho.

4.1 Busca e captura de dados para treinamento

A primeira parte do desenvolvimento da proposta envolve todo o esforço de pesquisa, verificação e download de dados para que o modelo possa ser treinado. O modelo em questão será treinado e trabalhará com imagens portanto devemos encontrar e analisar bases de dados de imagens que se encaixem no escopo proposto. As imagens precisam ser imagens médicas e astronômicas, os dois grupos de imagens para os quais o modelo irá se especializar.

Encontrar imagens que se encaixem nas descrições acima, não é nada trabalhoso com os motores de busca disponíveis na internet. Contudo, encontrar uma quantidade grande de imagens consistentes não é uma tarefa tão simples. Por imagens consistentes, refiro a um conjunto de imagens:

1. que seja composto por imagens de resoluções próximas ou uniformes. Os modelos são preparados para treinarem e serem alimentados com imagens de uma dimensão constante, ou seja, para os treinar com imagens de uma resolução de 256 *pixels* de altura por 256 *pixels* de largura, deve ser garantido que todo o restante da base de imagens esteja nesta resolução. Alguns modelos aceitam treinar com resoluções heterogêneas (i. e. as resoluções diferem de imagem pra imagem) desde que a largura e a altura sejam divisíveis por um número definido N .
2. que possua um número de imagens suficiente para um treinamento significativo. Este requisito apesar de não ser exato, é extremamente importante para a qualidade dos resultados. Com poucas imagens de treinamento, o modelo não será capaz de aprender os padrões que descrevem aquele grupo de imagem e o resultado é pobre. No caso de redes adversárias geradoras, imagens visualmente e matematicamente distintas do objetivo podem ser geradas, se porventura os modelos forem treinados com dados insuficientes.

Caso o primeiro requisito não seja cumprido, é possível, ainda que manualmente, uniformizar a resolução das imagens. O segundo tópico no entanto, é mais crítico, e caso não seja possível cumpri-lo, técnicas avançadas de gerações de bases de dados a partir de um conjunto de dados podem ser utilizadas, como foi mencionado na seção 1.1. Apesar desta possibilidade existir, esta técnica foge do escopo do trabalho, portanto irei me ater aos objetivos e prosseguir com o desenvolvimento.

Como os objetivos do trabalho envolvem treinamento com imagens de áreas específicas, médicas e astronômicas, alguns recursos específicos para estas áreas podem ser úteis.

Existem fontes formais de imagens médicas e imagens astronômicas. Alguns exemplos de fontes para imagens médicas são:

- OpenNeuro (OpenNeuro, 2022), antigo OpenfMRI (OpenfMRI, 2022). Site com diversas bases de dados contendo imagens médicas. Entre elas, bases de imagens de ressonância magnética.
- FastMRI (FastMRI, 2022). Site com imagens de ressonância magnética.
- *Cancer Image Archive* (Cancer Imaging Archive, 2022). Serviço de banco de imagens médicas sobre câncer.

E também, alguns exemplos de bases de imagens astronômicas:

- Kaggle (SRIVASTAVA, 2024). O Kaggle em si, é uma comunidade de ciência de dados e possui diversos dados e recursos para treinamento de modelos de aprendizado de máquina.
- SDSS (SDSS, 2024). O SDSS (Sloan Digital Sky Survey) regularmente publica e atualiza bases de dados de imagens de astros e objetos profundos no espaço.

Nas subseções abaixo, está descrito o procedimento utilizado para acessar as bases de dados médicas e astronômicas, começando pelas imagens médicas, as mais trabalhosas de se encontrar.

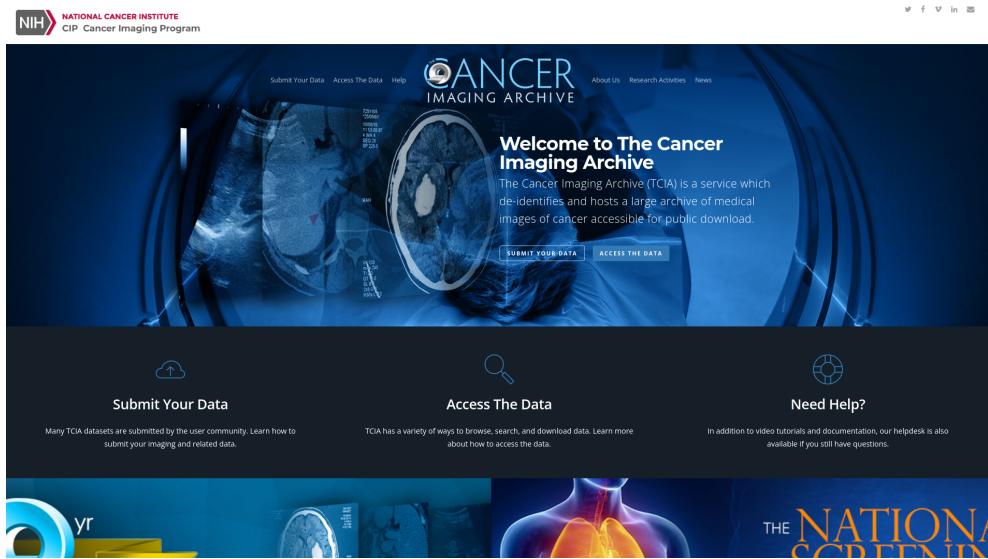
4.1.0.1 Obtenção de imagens médicas

Encontrar imagens médicas individuais ou até grupos pequenos, com até 300 exemplares, foi uma tarefa simples. Como salientado anteriormente, os mecanismos de busca modernos nos permitem fazer tal busca com precisão e velocidade. Todavia, encontrar uma base de dados com alguns milhares de imagens não foi simples nem rápido.

Algumas bases de dados com quantidades grandes de exemplares foram encontradas nos serviços citados. No entanto uma prioridade baixa foi atribuída a estas, devido principalmente aos problemas de resoluções de imagens. Ou as resoluções eram muito baixas ou eram muito heterogêneas, e apesar de ser possível ajustar isso posteriormente, é um trabalho extra a ser evitado.

Um serviço particularmente interessante para o caso de uso estudado, é o *Cancer Image Archive* (Cancer Imaging Archive, 2022) (figura 16). Neste serviço, é possível buscar as bases de imagens separadas por coleções classificadas por diversos aspectos como tipo de equipamento que produziu a imagem, tipo de câncer do qual as imagens se tratam, data de atualização dos dados, espécie da qual as imagens se tratam, entre outros. Grandes bases de dados foram encontradas utilizando este serviço, algumas com centenas de gigabytes de imagens.

Figura 16 – Captura de tela da página principal do *Cancer Imaging Archive*.

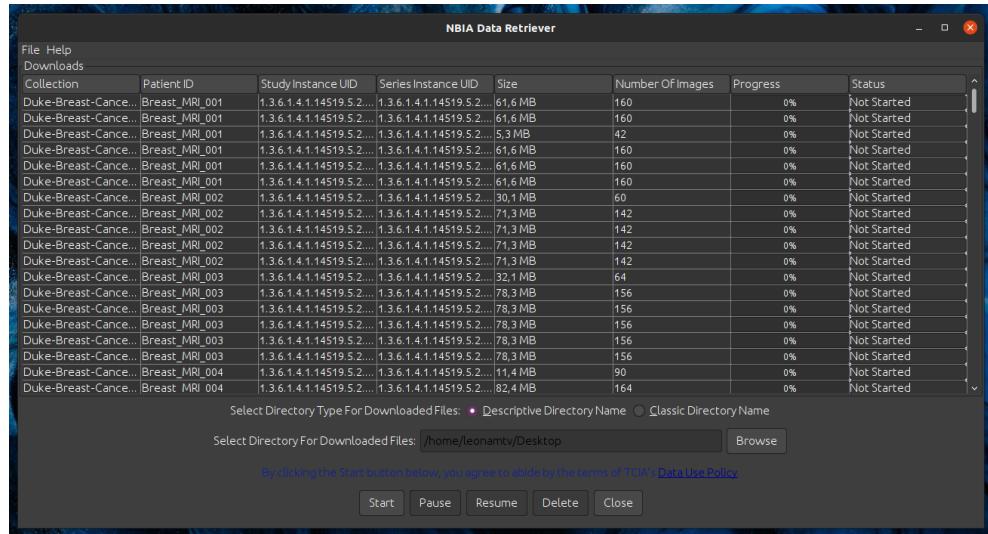


Fonte: (Cancer Imaging Archive, 2022)

A base de dados escolhida foi a base de imagens de câncer de mama *Duke Breast Cancer MRI* (SAHA et al., 2018) por mera tentativa e erro. Após procurar por algumas coleções, este conjunto de imagens é interessante por alguns motivos:

- Possui uma quantia significativa de imagens: 773126. Por motivos de armazenamento, apenas uma fração da base de imagens foi utilizada;
- As imagens escolhidas da coleção possuem resoluções uniformes, ou pelo menos aproximadas. Apenas parte da base foi utilizada, por isso, o termo "escolhidas". Isso não é um problema pois a base de dados contém tantas imagens, que a quantia total é impraticável para o escopo do trabalho. As imagens ocuparam cerca de 370GB em disco após o download.

Uma peculiaridade deste serviço utilizado, é a existência de uma ferramenta oficial para baixar as imagens chamada *NBIA Data Retriever* (Cancer Imaging Archive, 2022)(Vide figura 17). Nem todas as bases de dados requerem o download pela ferramenta mas a base de dados escolhida só estava disponível por este método.

Figura 17 – Captura de tela do *NBIA data retriever*.

Fonte: Captura de tela tirada pelo autor.

O procedimento para fazer o download segue os passos abaixo:

1. Deve-se criar uma conta no site (Cancer Imaging Archive, 2022) (figura 16) para ter acesso ao *carrinho*;
2. Adicione as bases de dados de interesse no carrinho e faça o download;
3. Um arquivo com o nome no formato **manifest-xxx.tcia** será baixado. Este arquivo contém todas as informações que o software *NBIA Data Retriever* (figura 17) precisa para fazer o download das imagens;
4. Instale o programa *NBIA Data Retriever* (figura 17) e abra o arquivo baixado com ele;
5. Faça o download das imagens.

4.1.0.2 Obtenção de imagens astronômicas

Diferente das imagens médicas, encontrar a base de dados de astronomia foi uma tarefa objetiva. Logo nas primeiras tentativas, é possível encontrar uma base de dados que se encaixe nos requisitos, com pouco esforço de pesquisa.

A base de dados astronômica encontrada, possui imagens de baixa resolução no geral. Por volta de 258 pixels de largura e altura. Apesar de a qualidade visual das imagens serem pobres, a baixa resolução traz menos estresse para os recursos computacionais disponíveis para treinamento.

Diferente da base de dados anterior que é monocromática, esta é colorida. Tal complexidade afeta o desempenho do treinamento dos modelos. Vide seção 4.3

4.2 Preparação das imagens para treinamento

As imagens encontradas não estavam integralmente prontas para o treinamento. Processamento anterior à esta fase é necessário para poder realizá-la com sucesso. Os modelos possuem entradas com formatos e resoluções específicos e treiná-los requer atenção à tais especificações.

O modelo em questão, na forma como foi implementado possui alguns requisitos ou recomendações sobre como deve ser treinado:

- As imagens devem possuir uma resolução uniforme. Mesma largura e mesma altura para todo o conjunto de imagens;
- A resolução das imagens deve ser divisível por 4. Este é um requisito específico para a ESRGAN de super-resolução em 4 vezes;
- As imagens devem estar em um formato capaz de ser lido pelas camadas de entradas do modelo (qualquer formato amplamente utilizado é aceitável e.g. png, jpg etc.).

A resolução das imagens estava de acordo com os requerimentos. Os bancos de imagens já proveram as imagens com uma resolução uniforme entre todas os exemplares. Contudo, por limitações de recursos, há possibilidades de as imagens serem muito grandes para o treinamento. O modelo é treinado em grupos de imagens. Estes grupos possuem uma quantidade fixa de exemplares escolhidos aleatoriamente pelo modelo a partir de uma base de treinamentos e esta quantidade é definida através de um parâmetro chamado *batch size*. Por exemplo, se a base de dados contém 6000 imagens e um *batch size* de 60 é utilizado, o treinamento do modelo irá ocorrer de 60 em 60 imagens escolhidas aleatoriamente dentro das 6000 imagens disponíveis. Este parâmetro é particularmente interessante para-se regular a quantidade de imagens em memória simultaneamente durante toda a fase de treinamentos. Quanto mais imagens na memória ao mesmo tempo, mais rápido será o treinamento pois menos leituras de disco serão realizadas.

Antes do treinamento definitivo, uma série de escolhas devem ser tomadas, para que o tempo e os recursos para treinar o modelo caibam dentro do disponível. Pode-se reduzir a resolução das imagens para que seja possível colocar mais imagens na memória mas isso pode reduzir a qualidade do modelo final, já que reduzir a resolução nada mais é que reduzir a qualidade (e quantidade de detalhes) dos dados utilizados para treinamento. Menos qualidade (quantidade de detalhes), menos padrões para o modelo aprender e em consequência este produzirá resultados inferiores. Há também a possibilidade de se reduzir o tamanho do *batch size*. Essa ação no entanto, faz com que durante o treinamento, mais leituras em disco ocorram e isso prolonga consideravelmente o tempo de duração desta fase.

Como os recursos disponíveis são baixos e limitados, devo adotar uma combinação das alternativas apresentadas para treinar o modelo. É então necessário, reduzir a resolução das imagens até uma resolução onde detalhes ainda são suficientes para um aprendizado

significativo. A redução do *batch size*, para a não violação do limite de memória, é também uma alternativa adotada.

4.2.1 Conversão de imagens médicas para um formato comum

Para treinar o modelo, as imagens utilizadas precisam estar em um formato comum, como mencionado anteriormente. As imagens médicas no entanto, estão em um formato especial utilizado internacionalmente em contexto médico: o formato *DICOM* (arquivos com extensão ".dcm"). *DICOM* é um padrão internacional de imagens médicas. É o padrão para armazenar, transmitir e utilizar imagens médicas de forma digital. O nome é sigla para *Digital Imaging and Communications in Medicine* (Imagem e comunicação digital em medicina, do inglês). O formato revolucionou a forma como equipamentos de imagem funcionavam desde que foi desenvolvido. (DICOM Standard, 2019; Medical Connections, 2011; DICOM Standard, 2024; Medical Connections, 2007; WESTON, 2020)

O padrão *DICOM* é complexo e longo e não cabe ao escopo deste trabalho descrever em detalhes o seu funcionamento e implementações. Felizmente, este padrão é amplamente utilizado na medicina e há bastante documentação esclarecendo tudo aquilo que o progresso do trabalho exige.

As imagens médicas que serão utilizadas para o treinamento estão todas neste formato e precisam ser convertidas para um formato aceito pelo modelo. Primeiramente, tal conversão deve ser realizada de forma automatizada, dada tamanha a quantidade de imagens.

Encontrar uma biblioteca capaz de extrair informações de imagens *DICOM* é uma tarefa simples, dada a popularidade deste padrão na comunidade técnica. Imagens deste formato armazenam muito mais que apenas informações sobre a localização e cores dos pixels que formam a imagem. Informações sobre o paciente e sobre o local onde a imagem foi capturada ficam contidos também no arquivo, desta forma as imagens sempre carregam informações sobre suas origens: onde foram geradas e de quem é o corpo descrito por aquela imagem.

É possível acessar estas informações com a biblioteca *pydicom* (Pydicom, 2022) para *python*. Vide código abaixo:

Figura 18 – Código para imprimir informações de um arquivo *DICOM*.



```

1 import pydicom
2
3 filename='imagem.dcm'
4
5 ds = pydicom.read_file(filename)
6
7 print(ds)
8

```

Fonte: Autor.

O código da imagem 18 acima, produz a seguinte saída:

Figura 19 – Saída obtida com a execução do código contido na imagem 18.

```

Dataset.file_meta -----
(0002, 0000) File Meta Information Group Length UL: 116
(0002, 0001) File Meta Information Version OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID UI: Ultrasound Multi-frame Image Storage
(0002, 0003) Media Storage SOP Instance UID UI: 999.999.999
(0002, 0010) Transfer Syntax UID UI: Implicit VR LittleEndian
(0002, 0012) Implementation Class UID UI: 999.999.999

-----
(0008, 0008) Image Type CS: ''
(0008, 0016) SOP Class UID UI: Ultrasound Multi-frame Image Storage
(0008, 0018) SOP Instance UID UI: 999.999.999
(0008, 0020) Study Date DA: ''
(0008, 0030) Study Time TM: ''
(0008, 0050) Accession Number SH: ''
(0008, 0060) Modality CS: 'MR'
(0008, 0070) Manufacturer LO: ''
(0008, 1030) Study Description LO: ''
(0010, 0010) Patient's Name PN: ''
(0010, 0020) Patient ID LO: ''
(0010, 0030) Patient's Birth Date DA: ''
(0010, 0040) Patient's Sex CS: ''
(0018, 1063) Frame Time DS: '100.0'
(0020, 000d) Study Instance UID UI: 999.999.999
(0020, 000e) Series Instance UID UI: 999.999.999
(0020, 0010) Study ID SH: ''
(0020, 0011) Series Number IS: None
(0020, 0013) Instance Number IS: None
(0028, 0002) Samples per Pixel US: 1
(0028, 0004) Photometric Interpretation CS: 'MONOCHROME2'
(0028, 0008) Number of Frames IS: '76'
(0028, 0009) Frame Increment Pointer AT: (0018, 1063)
(0028, 0010) Rows US: 512
(0028, 0011) Columns US: 512
(0028, 0100) Bits Allocated US: 16
(0028, 0101) Bits Stored US: 16
(0028, 0102) High Bit US: 0
(0028, 0103) Pixel Representation US: 0
(7fe0, 0010) Pixel Data OW: Array of 39845888 elements

```

Fonte: Autor.

De acordo com a saída, o arquivo *DICOM* utilizado traz uma série de informações relevantes para o contexto medicinal. Para o objetivo deste trabalho, o interesse é voltado apenas ao conteúdo do campo *Pixel Data* (vide imagem 20).

Figura 20 – Saída obtida com a execução do código contido na imagem 18, destacando o campo de interesse.

```
(0010, 0010) Patient ID          DS: '100.0'
(0018, 1063) Frame Time         UI: 999.999.999
(0020, 000d) Study Instance UID UI: 999.999.999
(0020, 000e) Series Instance UID UI: ''
(0020, 0010) Study ID           SH: ''
(0020, 0011) Series Number      IS: None
(0020, 0013) Instance Number    IS: None
(0028, 0002) Samples per Pixel   US: 1
(0028, 0004) Photometric Interpretation CS: 'MONOCHROME2'
(0028, 0008) Number of Frames    IS: '76'
(0028, 0009) Frame Increment Pointer AT: (0018, 1063)
(0028, 0010) Rows               US: 512
(0028, 0011) Columns             US: 512
(0028, 0100) Bits Allocated     US: 16
(0028, 0101) Bits Stored         US: 16
(0028, 0102) High Bit            US: 0
(0028, 0103) Pixel Representation US: 0
(7fe0, 0010) Pixel Data          OW: Array of 39845888 elements
```

Fonte: Autor.

Este atributo contém as informações da imagem em si. Neste objeto há tudo que é necessário para converter as imagens em um formato comum. Com algumas bibliotecas auxiliares, estes dados podem ser inseridos em uma estrutura de dados auxiliar, através das quais as imagens são escritas em um formato específico, como *jpg* por exemplo. O código abaixo, faz justamente isso.

Figura 21 – Código para converter uma imagem *DICOM* para o formato *jpg*.

```
1 import pydicom
2 import numpy as np
3
4 from PIL import Image
5
6 filename='imagem.dcm'
7
8 ds = pydicom.dcmread(filename)
9 image = ds.pixel_array.astype(float)[1]
10
11 scaled_image = (np.maximum(image, 0) / image.max()) * 255.0
12 scaled_image = np.uint8(scaled_image)
13
14 final_image = Image.fromarray(scaled_image)
15
16 final_image.save("imagem.jpg")
```

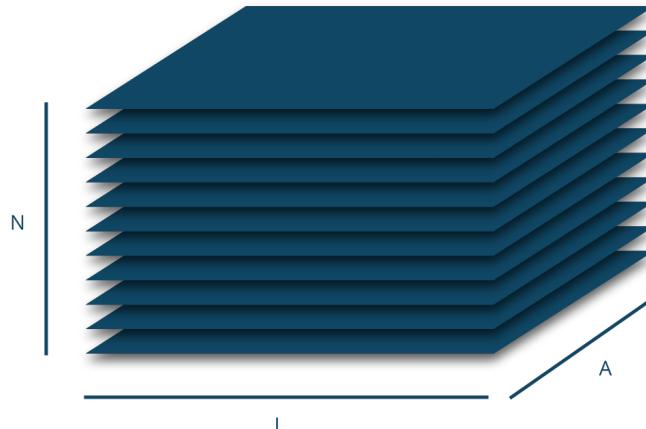
Fonte: Autor.

Alguns detalhes sobre o código contido na imagem 21 valem uma descrição e explicação, dada sua importância para que a conversão seja feita com sucesso.

Na linha 9, o programa lê o arquivo de imagem como números de ponto flutuante e no final da linha, captura o item na segunda posição (posição 1, já que o índice em *python* se inicia

em zero). Este detalhe é de extrema importância pois os arquivos *DICOM*, podem conter mais de uma imagem por arquivo. As imagens contidas em um arquivo *DICOM* possuem mesma largura e altura, e são indexadas por imagem individual, como mostra a imagem abaixo:

Figura 22 – Estrutura de imagens *DICOM*.



Fonte: Autor.

Na imagem 22, *L* representa a largura – uniforme para todas as imagens; *A* representa a altura – também uniforme para todas as imagens – e *N*, o número de imagens contidas no arquivo. Este valor *N* pode ser encontrado nos metadados do arquivo *DICOM* em um atributo chamado *Number of frames* como destacado na imagem 23:

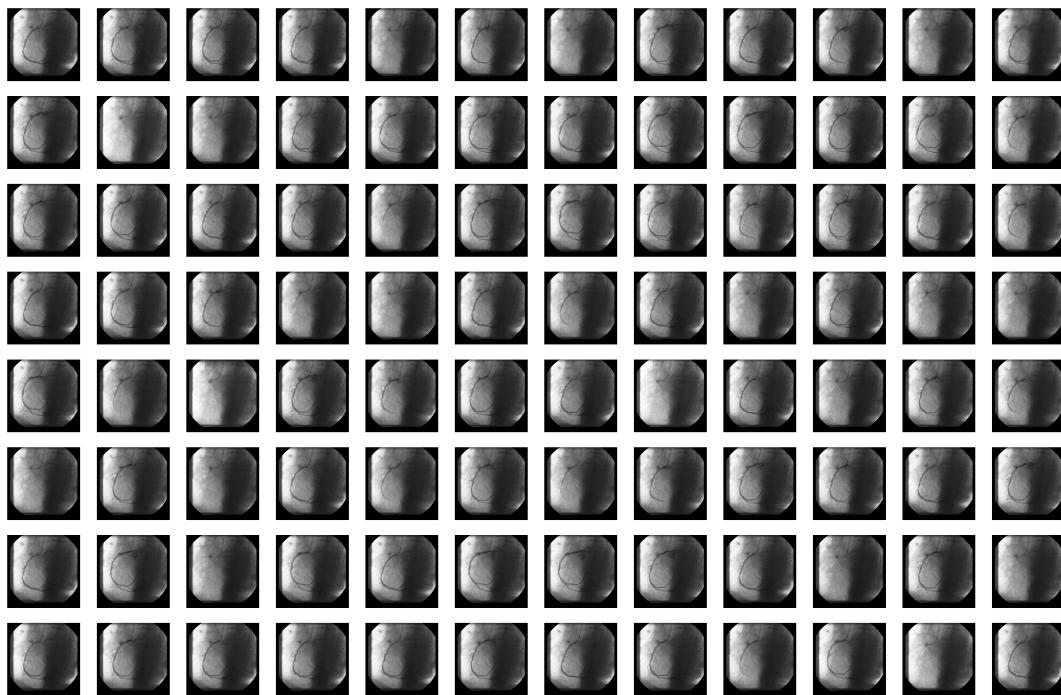
Figura 23 – Destacando o número de frames – ou número de imagens – nos metadados de um arquivo *DICOM*.

(0018, 1311) Positioning Secondary Angle	DS: '2.0'
(0018, 0010) Private Create Date	LO: 'CARDIO-D.R. 1.0'
(0018, 1030) [Maximum Frame Size]	UI: '2048x444'
(0020, 000d) Study Instance UID	UI: '1.3.12.2.1107.5.4.3.123456789012345.19950922.121803.6'
(0020, 000e) Series Instance UID	UI: '1.3.12.2.1107.5.4.3.123456789012345.19950922.121803.8'
(0020, 0010) Study ID	SH: '..'
(0020, 0011) Series Number	IS: '1'
(0020, 0013) Instance Number	IS: 'None'
(0020, 0020) Patient Orientation	CS: ''
(0021, 1010) Private Creator	LO: 'CARDIO-D.R. 1.0'
(0021, 1030) [Maximum Frame Number]	UI: '15'
(0028, 0092) Samples per Pixel	US: 1
(0028, 0084) Photometric Interpretation	CS: 'MONOCHROME2'
(0028, 0008) Number of Frames	IS: '96'
(0028, 0009) Frame Increment Pointer	AI: '(0018, 1063)
(0028, 0110) Rows	US: 512
(0028, 0111) Columns	US: 512
(0028, 0100) Bits Allocated	US: 8
(0028, 0101) Bits Stored	US: 8
(0028, 0102) Pixel Depth	US: 7
(0028, 0103) Pixel Representation	US: 0
(0028, 1040) Pixel Intensity Relationship	CS: 'LIN'
(0028, 1030) Recommended Viewing Mode	CS: 'NAT'
(0028, 6040) R Wave Pointer	US: [20, 53, 77]
(0028, 6100) Mask Subtraction Sequence	1 item(s) ----
(0028, 6101) Mask Operation	CS: 'NONE'
(0028, 6100) Mask Frame Numbers	US: 0

Fonte: Autor.

Para uma conversão em massa de uma base com muitas imagens é preciso tratar situações como esta, pois o processo de conversão, apesar de ser rápido para um único arquivo, pode demorar horas para grandes quantidades de imagens. Caso tais detalhes não sejam levados em consideração, um erro pode interromper a execução do programa de conversão e mais tempo terá de ser gasto para resolver o problema. O único arquivo *DICOM* mostrado na imagem 23 possui 96 imagens e gerou todas as imagens mostradas na figura abaixo:

Figura 24 – Todas as imagens contidas no arquivo *DICOM*.



Fonte: Autor com imagens do (Cancer Imaging Archive, 2022).

Outro ponto do código contido na imagem 21 que deve ser explicado, são as linhas 11 e 12. Para salvar as imagens em um formato comum, é necessário converter seus pixels para o valor correto. Por padrão as imagens contém pixels com valores entre 0 e 255 e é isso que a linha 11 faz: normaliza os pixels para um valor entre 0 e 255. E por fim, a linha 12 converte cada pixel para um inteiro de 8 bits.

Agora, a imagem está em um formato utilizável para treinar o modelo. Para fazer este procedimento em todas as imagens da base, o programa precisa ser alterado para ler os arquivos *DICOM* do disco de forma automática e escrever os novos arquivos *jpg* também de forma automatizada.

4.2.2 Uniformização e redução da resolução

Um dos requisitos necessários para se treinar o modelo é uma resolução uniforme em todas as imagens. Uniforme em alguns aspectos: largura e altura consistentes e ambas as dimensões devem ser múltiplas de **N**, onde **N** é a escala na qual o modelo será treinado para performar. **N** vale 4 para o contexto deste trabalho, já que o modelo treinado, será capaz de super resolver imagens em quatro vezes o tamanho original.

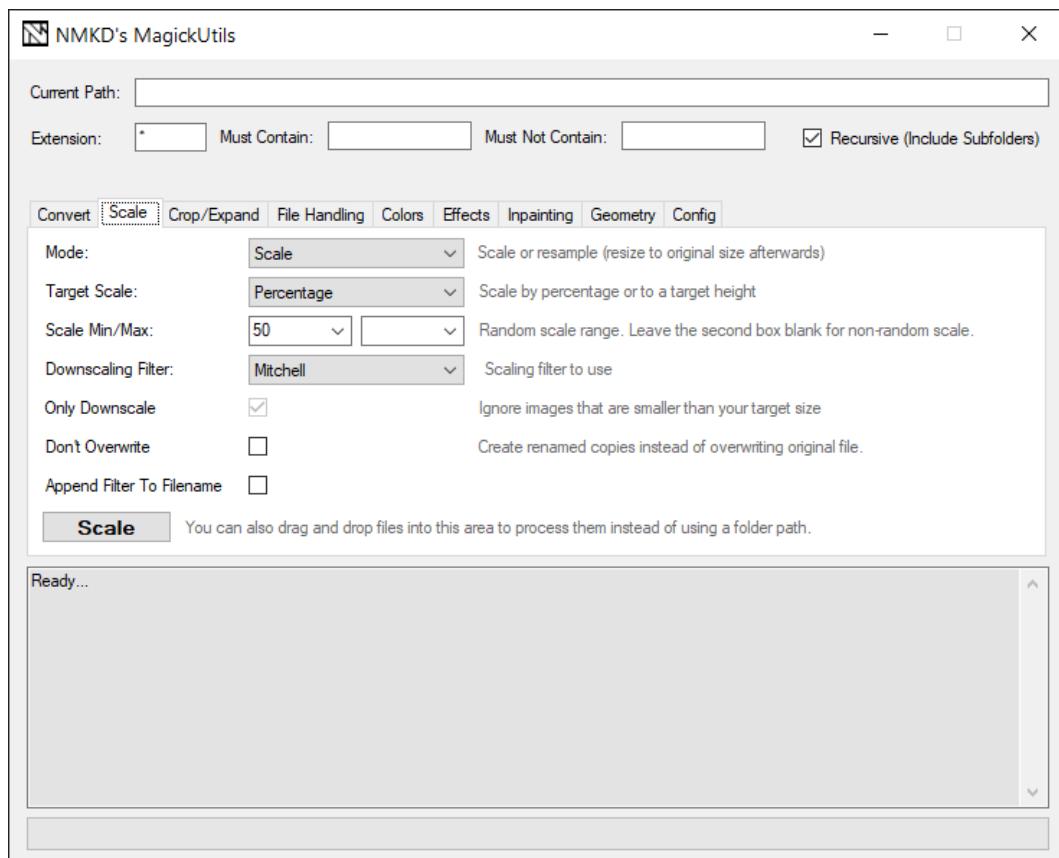
Não é necessário preocupar com a consistência entre largura e altura já que as imagens originalmente possuem altura e largura uniformes em todas as imagens selecionadas para treinamento. A segunda parte contudo, há de ser feita.

Transformar todas as milhares de imagens em imagens com resoluções múltiplas de 4 manualmente é uma tarefa completamente infactível. Alguma forma de automação deste

processo é indispensável para a continuação do trabalho. Foi considerado o desenvolvimento de um software para tal, que fosse capaz de ler imagens do disco, encontrar sua resolução e redimensioná-la para o valor mais próximo que também fosse múltiplo de 4. O desenvolvimento e testes de tal ferramenta demandaria bastante tempo para ser implementado e testado. Se levada em consideração, a necessidade de que este software seja flexível à novas configurações (como novos valores para **N**, por exemplo), novas operações (como cortes em imagens, por exemplo), é razoável concluir com boa precisão, que ainda mais tempo seria despendido para trabalhar na criação de uma ferramenta capaz de realizar todas estas tarefas.

Este desenvolvimento foi dispensado, pois um software de código aberto chamado *Magick-Utils* não muito famoso, já faz a conversão da resolução de forma automatizada. Este software foi encontrado de forma inesperada, na descrição de um vídeo tutorial para o treinamento de uma implementação do modelo ESRGAN, e seu código é completamente aberto

Figura 25 – Captura de tela do software *Magick-Utils*.



Fonte: *Github* do criador do software (N00MKRAD, 2022).

O processo é relativamente lento, mas é possível fazer a conversão de todas as imagens em algumas horas.

Como descrito na seção 4.2, uma das formas de se controlar tempo e consumo de recursos do treinamento é reduzindo a resolução das imagens. Caso tal redução se faça necessária, o software *Magick-Utils* (N00MKRAD, 2022) é perfeito para a tarefa. O programa

suporta o usuário selecionar uma pasta inteira contendo dezenas de milhares de imagens. Este, irá alterar todas as imagens para uma resolução menor, até que as imagens estejam em um tamanho praticável para os recursos computacionais disponíveis para o treinamento.

4.3 Considerações gerais sobre as imagens pós processadas

Para sumarizar o estado das bases de imagens após o processamento descrito nas seções anteriores, segue a tabela abaixo:

Tabela 1 – Tabela sumarizando as imagens processadas.

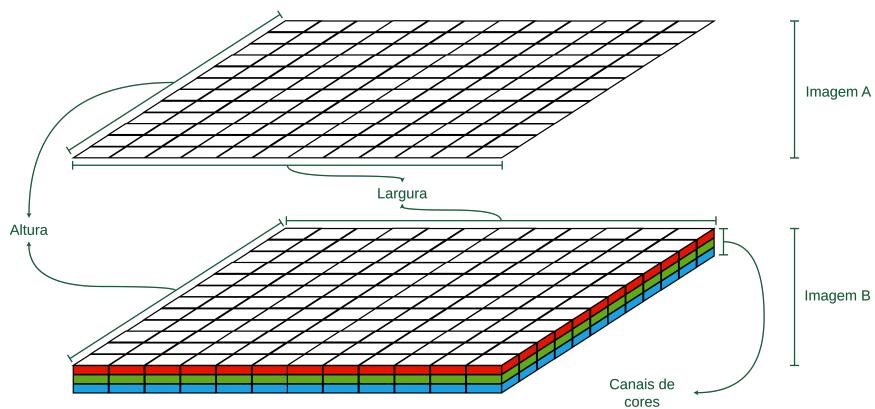
Estatística	Base de imagens	
	Ressonância magnética	Astronomia
Nº de imagens	6031	243435
Resolução	108x108	424x424
Tamanho médio (bytes)	4288.04	12674.51
Tamanho 90º percentil (bytes)	5314.6	17674.51
Cor	Monocromática (Tons de cinza)	Colorida

Fonte: Autor.

Algumas das diferenças entre as bases de dados podem impactar no treinamento. A resolução das imagens astronômicas é quase quatro vezes maior que a resolução das imagens médica e em consequência, também medem mais de três vezes mais em disco.

Como se a diferença de resolução não bastasse, as imagens astronômicas são coloridas. A presença de cor aumenta, em algumas dimensões a estrutura de dados, neste caso, o tensor que irá trafegar nas redes. Enquanto uma imagem monocromática como as da base médica, pode ser representada com uma estrutura bidimensional. A imagem colorida requer mais uma dimensão para os canais de cores.

Figura 26 – Diagrama demonstrando diferença estrutural entre imagem monocromática e colorida.



Fonte: Autor

Na figura 26, a imagem A representa uma imagem monocromática, com apenas duas dimensões: altura e largura. A imagem B, representa uma imagem colorida com cores RGB. Uma dimensão a mais é necessária para armazenamento e processamento: os canais de cores.

4.4 Preparação do ambiente para treinamento

Nesta parte do trabalho, é descrita toda a configuração, e tentativas de configuração, necessárias para o treinamento correto do modelo ESRGAN. Detalhes do hardware disponível e utilizado são apresentados, assim como também especificações de software como sistema operacional, bibliotecas utilizadas, forma de executar o treinamento etc.

O modelo utilizado, consiste em uma implementação de terceiros da especificação da ESRGAN já que entender os pormenores deste modelo é um esforço muito além do escopo do trabalho. Como utilizar tal software, envolve lidar com softwares e implementações externos, é preciso ater-se aos detalhes disponibilizados para que seja possível configurar o ambiente e preparar as entradas de forma correta.

Antes da descrição mais acurada do processo, é fundamental deixar claro os requisitos necessários para utilizar uma ESRGAN. O modelo, como explicado anteriormente consiste de várias camadas de processamento. Cada uma destas, executa uma quantidade significativa de cálculos cada vez que o modelo é alimentado com dados de entrada, seja em fase de treinamento, testes ou na utilização real.

Modelos como este, assim como diversos outros tipos de modelos de inteligência artificial, conseguem tirar bastante proveito de processadores com vários núcleos. Estes processadores conseguem, quando implementado de tal forma, realizar uma grande quantidade de cálculos simultaneamente. Cálculos que individualmente são simples operações matemáticas, mas quando combinados da forma correta, representam todo o modelo que este trabalho analisa. Um tipo de processador que se encaixa nesta categoria, são as GPUs (Graphics Processing Unit, unidade de processamento de gráficos, do inglês). Vários dispositivos domésticos vêm de fábrica com uma GPU dedicada que possui centenas, senão milhares de núcleos.

Com isso em mente, diversos modelos e bibliotecas de computação numérica e inteligência artificial são desenvolvidos com a ideia de que serão treinadas, testadas e utilizadas em GPUs, em mente. Vários testes de performance demonstram a superioridade das GPUs em comparação com os processadores tradicionais (CPUs) para tarefas de treinamento de modelos de inteligência artificial. De acordo com (VELASQUEZ; LIND, 2019), para modelos mais complexos é notável a diferença de desempenho quando se utiliza de GPUs para o treinamento. Esta diferença no entanto, decresce com modelos mais simples.

Para usufruir deste hardware, é necessário no entanto de uma forma de interface para que o modelo, implementado utilizando uma tecnologia X, representando qualquer variação do modelo feito com tecnologias distintas, possa acessar os recursos da GPU, independente do modelo. É esse o papel do CUDA. Uma ferramenta desenvolvida pela NVIDIA para fazer esse interfaceamento. CUDA, de acordo com a própria empresa por trás da ferramenta, é

uma plataforma de computação paralela que engloba diversos recursos. A ferramenta é capaz de compilar código para executar em GPUs, usar interfaces em diferentes linguagens para acessar recursos do hardware entre outras coisas. Os desenvolvedores que utilizam CUDA, determinam quais partes do código devem ser executadas de forma paralela e demarcam este trecho com anotações. O compilador ou interpretador então, detecta estes trechos e os executa em núcleos paralelos na GPU.

Várias destas bibliotecas utilizadas para desenvolver modelos de inteligência artificial e redes neurais fazem uso de recursos matemáticos como tensores, que generalizam várias estruturas básicas: valores escalares, vetores, matrizes etc. Existem hardwares específicos para processamento de tensores, os assim chamados TPUs (Tensor Processing Unit, unidade de processamento de tensores, do inglês) e inclusive, há formas de acessá-los de forma gratuita. Como o modelo utilizado neste trabalho foi desenvolvido e otimizado para GPUs, é supérfluo entrar em mais detalhes sobre TPUs. Vale no entanto, a menção e talvez um trabalho futuro envolvendo o assunto.

4.4.1 Descrevendo o hardware disponível

Para treinar o modelo, um computador esteve disponível. A especificação de seu hardware está contida na tabela abaixo:

Tabela 2 – Tabela de descrição do hardware

Item	Descrição
CPU	Intel i7 Octa Core
Memória RAM	16GB
Memória de Vídeo	2GB
Modelo da placa de vídeo	GeForce MX350
Disco	512GB (apenas 90GB disponível)
Sistema operacional	Windows 10 e Ubuntu 24.04 disponível

Fonte: Autor.

A placa de vídeo apresentada (GeForce GT920M) é um modelo de computadores portáteis voltada para jogos. Uma informação importante de se documentar, é sua quantidade de Núcleos *CUDA* conhecidos popularmente pelo termo em inglês *Cuda Core*. Os núcleos *CUDA* são os núcleos de processamento de instruções das *GPUs* da *NVIDIA*, conceito bem parecido com o conceito de núcleos de processadores (RYLES, 2022). A placa em questão possui compatibilidade com *CUDA*, um requisito indispensável, e possui 384 núcleos *CUDA* (NVIDIA, 2022; Technical City, 2022). Esta placa de vídeo de entrada possui 96 vezes mais núcleos que a CPU disponível.

O hardware descrito, está distante de ser o ideal para o treinamento de um modelo complexo e exigente em termos de recursos, como é o caso do modelo descrito e utilizado neste trabalho. Treinamentos do tipo costumam ser feitos em servidores dedicados, com processadores poderosos e grandes quantidades de memória RAM e memória de vídeo (VRAM).

Por causa desta limitação de recursos, é preciso recorrer à alternativas a nível de software, como descrito na seção 4.2.

4.4.2 Descrição do software necessário para treinar o modelo

Para treinar, executar e utilizar o modelo de forma geral, as dependências da implementação utilizada devem ser satisfeitas. Ou seja, é preciso antes, instalar as bibliotecas, *frameworks* etc. necessários para a execução do modelo.

Nesta seção será descrito, com o maior nível de detalhes possível, como este processo é realizado. Da preparação do ambiente até à execução do treinamento e do modelo em si. Tenha em mente que esta etapa, apesar de parecer simples (afinal, consiste apenas em instalar softwares e no máximo fazer uma eventual configuração), pode ser bastante complicada por alguns motivos.

O primeiro passo nesta tarefa, é reproduzir o ambiente onde o modelo foi desenvolvido e testado. Reproduzir um ambiente qualquer, requer manter todas as bibliotecas e softwares em geral, na mesma versão que o ambiente desejado, ou pelo menos manter tudo em uma versão compatível. Caso o(s) desenvolvedor(es) do modelo forneçam estas informações, grande parte do esforço está adiantado. Caso estas informações não estejam disponíveis, estas precisam, de forma não negociável, ser encontradas, seja buscando em fóruns ou outras fontes elaboradas por pessoas que passaram pelo mesmo problema, ou por tentativa e erro.

O segundo motivo complicador desta fase do trabalho, é a compatibilidade entre software e hardware. Como explicado anteriormente, reproduzir o ambiente a nível de software requer encontrar uma interseção de compatibilidade entre todas as dependências necessárias para executar o modelo. O mesmo acontece com hardware. Alguns dispositivos, como *GPUs*, que são extremamente necessárias para este trabalho, possuem *drivers* específicos em versões específicas, que integram com versões particulares de bibliotecas necessárias. Até aí, nenhuma novidade em relação aos problemas de software. Quando tratamos de hardware no entanto, não se pode contar com versões atualizadas de determinado *driver* para sempre. Os desenvolvedores podem interromper o suporte de uma *GPU* A, mais antiga e obsoleta, para concentrarem os esforços num modelo B, mais atualizado. Isto pode ser um problema comprometedor, especialmente quando há uma limitação tão grande de hardware como é o caso deste trabalho.

4.4.2.1 Experimentos preliminares com o ambiente

Os primeiros experimentos de preparação do ambiente foram feitos em um computador com uma instalação do sistema operacional Ubuntu versão 18.04LTS. Várias fontes informais como vídeos e comentários em fóruns e redes sociais sobre o assunto recomendaram o uso de uma distribuição Linux qualquer para o treinamento de modelos complexos e pesados como o modelo atual. As recomendações sempre levavam em consideração a forma como os sistemas operacionais Linux gerenciam os recursos de forma mais minimalista, deixando mais memória, processador e disco disponíveis para o treinamento em si. Além disso, Linux pareceu uma boa primeira alternativa, devido à familiaridade de uso.

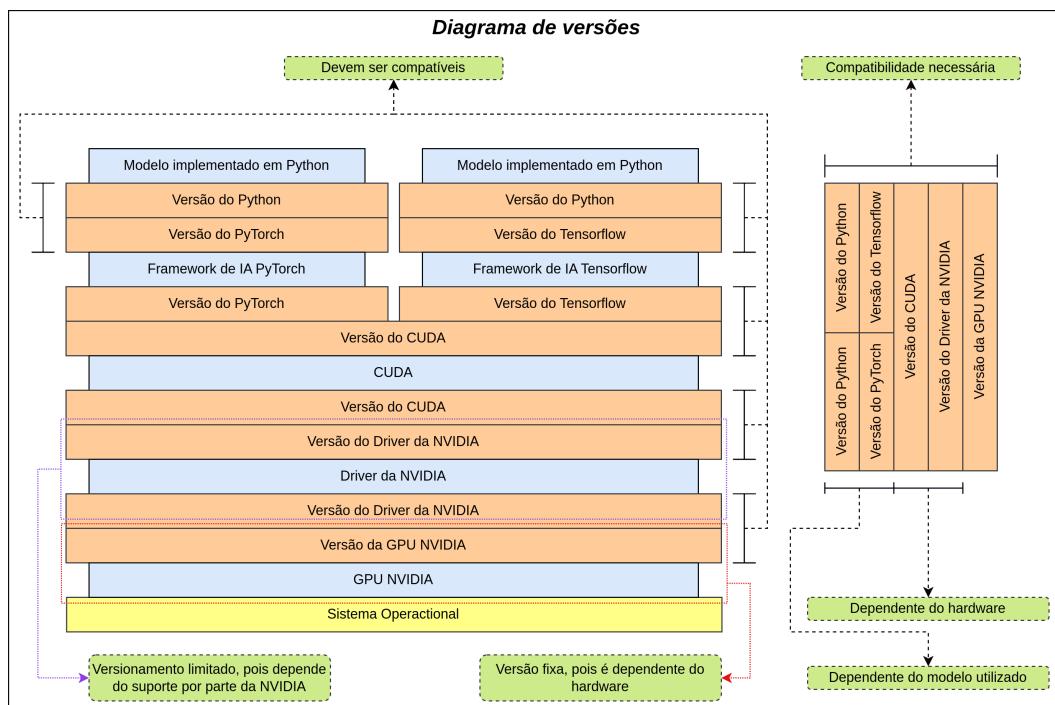
Contudo, os experimentos no Linux foram um insucesso. Como parte das dependências são fornecidas pela empresa *NVIDIA*, suas bibliotecas proprietárias precisam ser instaladas no Linux e isso nem sempre é uma tarefa simples. No experimento realizado, ao instalar o *driver* necessário para utilizar a *GPU*, várias funcionalidades básicas do sistema operacional pararam de funcionar propriamente.

Como uma instalação do Windows 10 estava disponível, não valeria o esforço de tentar resolver os diversos problemas de compatibilidade entre dependências da *NVIDIA* e o sistema operacional, o desenvolvimento prosseguiu utilizando como sistema operacional, o Windows 10.

4.4.2.2 Breve descrição sobre versões

Para fazer qualquer procedimento com este modelo, seja treinamento, teste ou execução, vários níveis de compatibilidade de software com software e de software com hardware precisam ser garantidos. O diagrama abaixo ilustra a interação entre as partes:

Figura 27 – Diagrama de interação entre as partes envolvidas no modelo.



Fonte: Autor.

Na figura 27, os blocos laranja fazendo contato entre si representam compatibilidade de comunicação entre as partes. Ao todo são quatro níveis de compatibilidade. As versões do *python* e dos frameworks (*PyTorch* e *Tensorflow*) já são predeterminadas pela implementação do modelo. A dificuldade vem em encontrar uma versão do *CUDA* que se comunica com a versão dos frameworks e do *driver* e uma versão do *driver* que se comunica com o hardware em si.

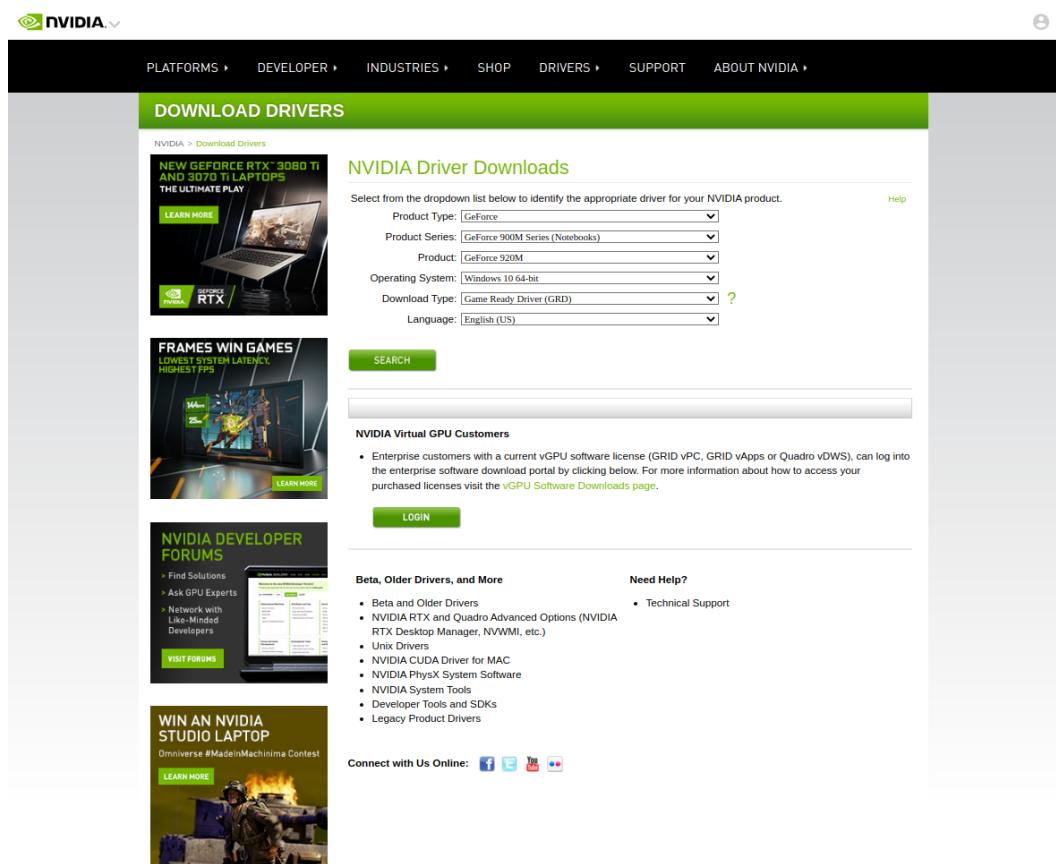
4.4.2.3 Instalação do *driver* da NVIDIA

O passo base para prosseguir com a instalação das dependências, é instalar um *driver* compatível com ambos, a placa de vídeo disponível, e a versão das bibliotecas utilizadas.

Esta seção, apesar de ter sido colocada imediatamente anterior à seção sobre *CUDA* (vide seção 4.4.2.4) devido à ordem cronológica correta e esperada dos passos, foi desenvolvida em repetição antes e após a instalação do *CUDA*. Isto aconteceu por inúmeros problemas de compatibilidade que apareceram ao tentar integrar o *driver* com a biblioteca e estes dois com o hardware em si. O processo é extremamente demorado e mecânico. Como a instalação e desinstalação tomam bastante tempo por si só, é de se esperar que fazê-los várias vezes até obtermos uma combinação compatível entre si, tomaria muito tempo.

Para encontrar o *driver* específico ao *hardware* (GPU) disponível, basta buscar no mecanismo oficial de busca de *drivers* atualizados do site da *NVIDIA* (figura 28).

Figura 28 – Captura de tela do buscador de drivers da *NVIDIA*.

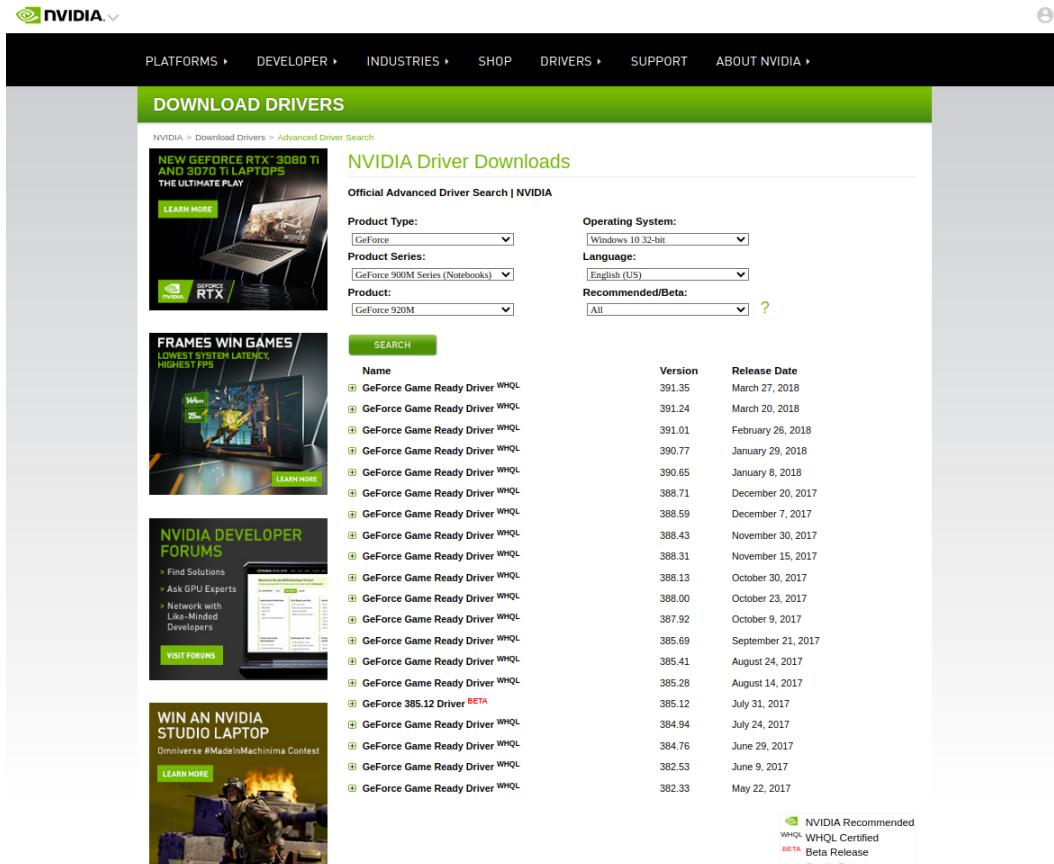


Fonte: Buscador de *drivers* atualizados da *NVIDIA* (NVIDIA, 2023b)

Caso, por motivos de compatibilidade o *driver* atualizado não integre com as bibliotecas necessárias para o desenvolvimento, existe ainda a alternativa de procurar por versões anteriores ou versões de teste do *driver* para o hardware. A figura 27 mostra o buscador avançado de *drivers* que a *NVIDIA* disponibiliza. Nele, dispõe-se várias opções com as quais é possí-

vel experimentar. Basta buscar pelo modelo do seu hardware, filtrando pelas várias opções disponíveis.

Figura 29 – Captura de tela do buscador avançado de drivers da *NVIDIA*.



Fonte: Buscador avançado de *drivers* da *NVIDIA* (NVIDIA, 2023a)

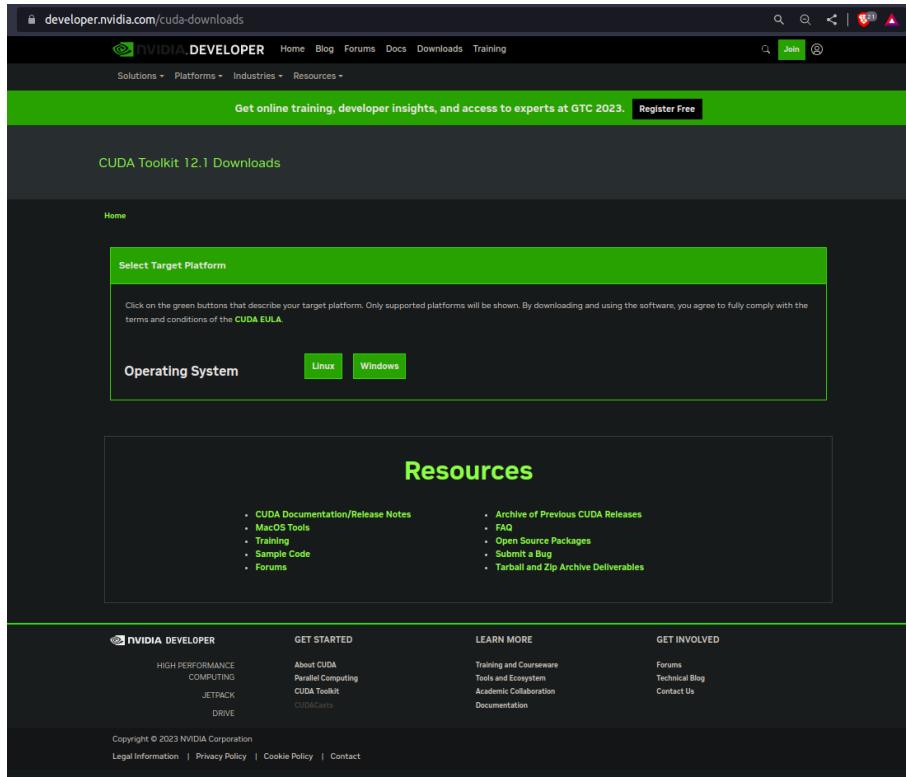
Após encontrar o *driver* específico, basta fazer o *download*, executar o arquivo e seguir os passos para instalação. A instalação contém várias opções e componentes mais voltadas para cenários específicos. As opções padrão geralmente são suficientes para a maior parte dos casos, de acordo com os testes realizados. Além da instalação do *driver*, algumas atualizações podem ser necessárias. O instalador é capaz de resolvê-las por conta própria.

4.4.2.4 Instalação do CUDA

Esta parte precisa de ser realizada em perfeita harmonia com a seção 4.4.2.3. Qualquer versão incorreta pode impedir o funcionamento do projeto como um todo.

A primeira parte da instalação, como esperado, é baixar o instalador. Isso pode ser feito através do site da *NVIDIA Developers*. Um site voltado para usuários mais técnicos que desenvolvem para as plataformas da *NVIDIA*.

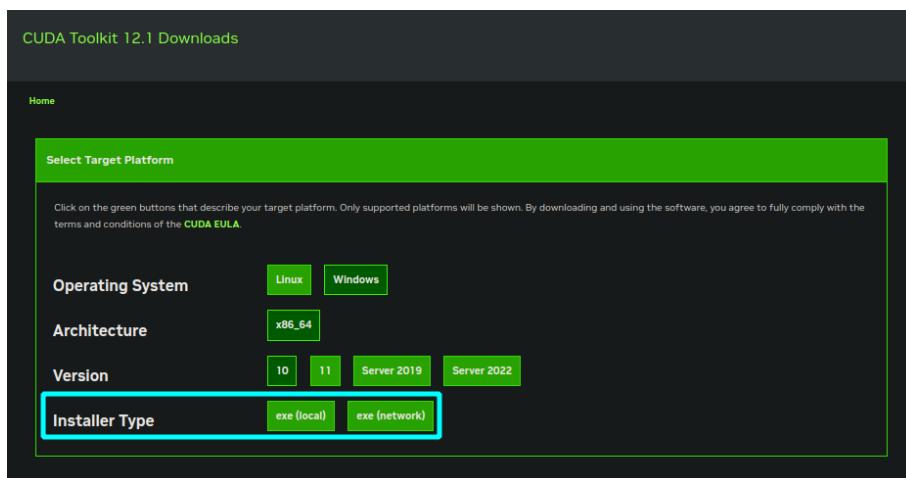
Figura 30 – Captura de tela do site da *NVIDIA Developers*.



Fonte: Buscador de instaladores CUDA (NVIDIA, 2024).

A figura 30 mostra a página inicial do site. Para encontrar o instalador correto para cada situação, preencha o filtro com os detalhes desejados.

Figura 31 – Captura de tela do filtro do buscador de instaladores CUDA.

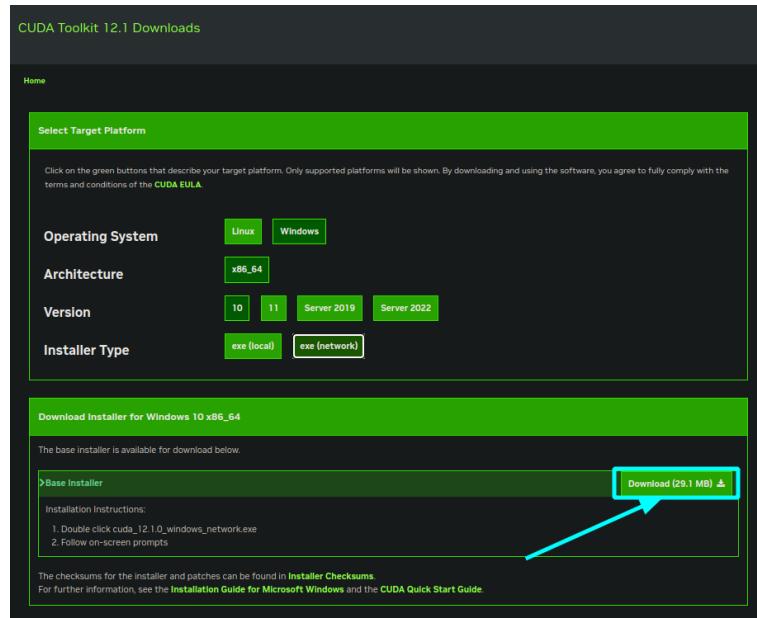


Fonte: Filtros do buscador de instaladores CUDA (NVIDIA, 2024).

Com os filtros da figura 31 preenchidos, serão disponibilizadas duas versões para download (*Installer type*, tipo do instalador, do inglês): o instalador completo (opção *local*) e o instalador pela rede (opção *network*). O instalador completo faz um download lento e

grande mas uma vez baixado, nenhum download extra será necessário. O segundo, pela rede, baixa um instalador mais compacto e, durante a instalação faz o download do restante. Ambas opções trarão o mesmo resultado. Após a escolha do tipo de instalador, um botão para o download será exibido.

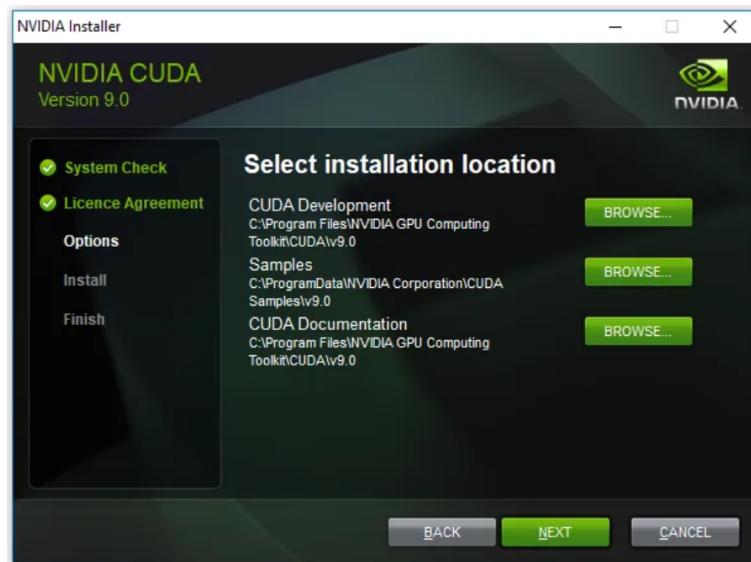
Figura 32 – Captura de tela da busca de instaladores CUDA.



Fonte: Busca completa de instaladores CUDA (NVIDIA, 2024).

Faça o download, clicando no botão destacado na figura 32 e execute o arquivo baixado.

Figura 33 – Captura de tela de instalação CUDA.



Fonte: Tela do instalador CUDA. (KITSON, 2022)

Nas opções do instalador acima, serão disponibilizadas a versão expressa e a versão avançada. A versão expressa, de instalação mais rápida, é suficiente para o propósito desta instalação. Siga os passos normalmente até finalizar. Esta instalação pode tomar um tempo considerável, dependendo do tipo de instalador escolhido. Caso a opção rede (*network*) tenha sido a escolhida, a maior parte do software será baixada durante a instalação, justificando assim a demora extra.

4.4.3 Apresentação do modelo

Nesta subseção, detalhes sobre a implementação do modelo, assim como características de configurações serão descritos.

4.4.3.1 Descrevendo a implementação

Algumas implementações foram utilizadas durante os experimentos. Cada uma delas, fornece uma abordagem diferente para a mesma arquitetura, apesar de compartilharem o mesmo objetivo final. Uma das principais diferenças encontradas entre implementações distintas, são as ferramentas utilizadas para desenvolvê-las. Sejam versões diferentes ou ferramentas completamente diferentes.

O requisito necessário para que uma destas implementações sejam adequadas para o trabalho é interoperabilidade com os drivers e hardwares disponíveis. No final das contas, as implementações utilizadas podem ser divididas em dois grupos:

- Modelos implementados utilizando a biblioteca *Tensorflow*
- Modelos implementados utilizando a biblioteca *PyTorch*

Ambas as bibliotecas são extremamente populares no meio científico de aprendizado de máquinas e computação numérica em geral. Estas bibliotecas abstraem parte da complexidade matemática por trás das construções e execuções dos modelos, dando à pessoa que está desenvolvendo, uma linguagem mais declarativa e de alto nível para expressar suas intenções. Toda a parte algébrica por trás do modelo, é então trabalhada pela biblioteca. Otimizações e transformações são feitas, para maximizar o aproveito do hardware, livrando assim, o(a) desenvolvedor(a) final de tais aperfeiçoamentos.

Após várias tentativas com modelos baseados no Tensorflow, foi concluído que, dado o recurso computacional disponível, estes modelos não iriam trazer resultados satisfatórios em tempo hábil. Os treinamentos com os modelos em Tensorflow testados, produziam erros de memória insuficiente para treinamento, horas após o início do processo.

Treinar as redes só foi possível com os modelos em Tensorflow utilizados, quando os parâmetros de treinamentos eram absurdamente baixos (e.g. uma imagem carregada por vez, i.e. um *batch_size* de 1). Isso por si só iria deteriorar consideravelmente a qualidade dos resultados, dentro do tempo disponível para treinamento que inicialmente, compreendia o período noturno, quando a máquina não estava sendo utilizada. Ou seja, os modelos baseados no Tensorflow iriam tomar um tempo impraticável para atingir um número **n** de épocas.

Além disso, nem todos os modelos em Tensorflow eram executáveis na máquina utilizada, seja por incompatibilidade de versões das bibliotecas, seja por erros de configuração.

No final das contas, um modelo baseado em PyTorch foi o escolhido. Este modelo possui certa complexidade para se configurar, mas diferente dos modelos em Tensorflow testados anteriormente, as falhas acontecem cedo, permitindo assim melhor adaptabilidade no treinamento. O consumo de memória deste modelo, também foi significativamente menor, com parâmetros pequenos (mas não tão pequenos a ponto de delongar muito o treinamento), é possível utilizar a máquina para outras tarefas paralelas, algo infactível nos modelos anteriores.

O modelo de código aberto *BasicSR* (BLUEAMULET, 2023), foi o modelo eleito. O projeto em si, contém outras implementações e capacidades, mas para o escopo desse trabalho, o foco principal foi sob o módulo de ESRGAN com super resolução de quatro vezes das imagens de entrada. A implementação é baseada no modelo de ESRGAN anteriormente citado neste trabalho (WANG et al., 2018).

4.4.3.2 Detalhando a preparação do projeto para treinamento

Para executar a implementação, seja treinamento ou execução, esta antes, deve ser configurada. O projeto utilizado requer duas configurações básicas além obviamente, das dependências descritas anteriormente (driver da NVIDIA e CUDA). Primeiramente um modelo pré-treinado inicial deve ser encontrado assim como um arquivo de configurações contendo os dados que o modelo precisa para funcionar propriamente, como caminho para o diretório com os dados de treinamento, qual algoritmo de treinamento utilizar, etc.

Os modelos pré-treinados podem ser encontrados juntos com o projeto da implementação do modelo. Uma característica muito importante que deve ser levada em consideração é a arquitetura do modelo pré-treinado. Essa arquitetura deve ser idêntica à arquitetura para a qual o modelo irá ser treinado, ou seja, se uma ESRGAN para fazer super resolução de quatro vezes será treinada, obrigatoriamente, um modelo pré-treinado de ESRGAN com super resolução de quatro vezes deve ser utilizado. Qualquer outra variação não irá funcionar, devido à falta de compatibilidade da estrutura interna das camadas e pesos entre uma arquitetura e outra.

Após obtido o modelo pré-treinado, deve se organizar os dados de treinamento de uma maneira específica para que a implementação possa entender e configurar o projeto. Como o objetivo é obter um modelo treinado capaz de super resolver imagens em quatro vezes sua resolução inicial, as bases de dados devem estar dispostas com as seguintes características:

1. Uma base de dados de treinamento, contendo:

- Imagens em baixa resolução
- A versão de alta resolução (quatro vezes a resolução) das imagens de baixa resolução

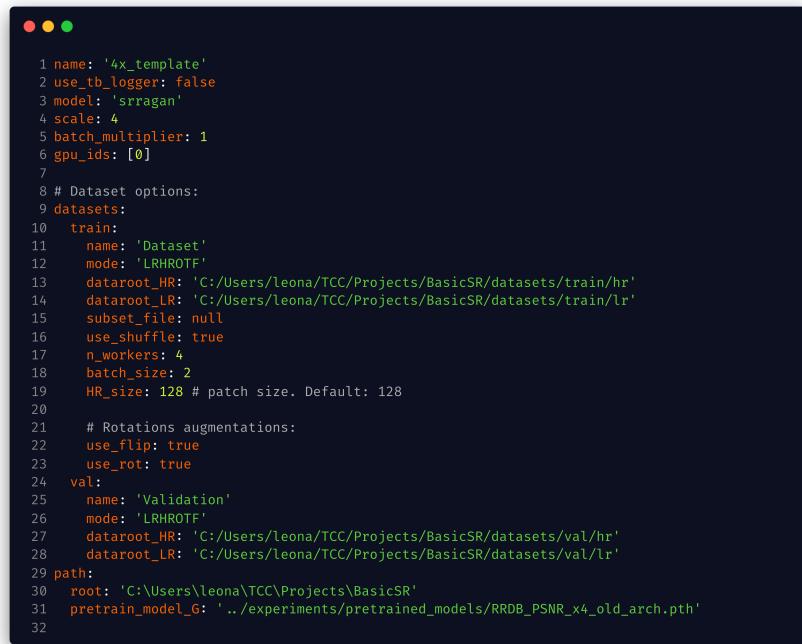
2. Uma base de dados de validação, contendo:

- Imagens em baixa resolução
- A versão de alta resolução (quatro vezes a resolução) das imagens de baixa resolução

A proporção de dados separados para cada uma das duas bases de dados foi de 80:20: 80% das imagens dedicadas à treinamento, e 20%, dedicadas à validação. Para gerar as imagens de baixa resolução, a partir das imagens de alta resolução, o processo descrito na seção 4.2.2 foi utilizado.

Com as imagens preparadas e distribuídas em diretórios específicos, de acordo com a descrição anterior, os caminhos onde cada uma das divisões da base de dados se encontra podem ser adicionados no arquivo de configuração do projeto.

Figura 34 – Trecho do arquivo de configuração para treinamento e execução da implementação do modelo.



```

1 name: '4x_template'
2 use_tb_logger: false
3 model: 'srragan'
4 scale: 4
5 batch_multiplier: 1
6 gpu_ids: [0]
7
8 # Dataset options:
9 datasets:
10 train:
11   name: 'Dataset'
12   mode: 'LHRHOTF'
13   dataroot_HR: 'C:/Users/leona/TCC/Projects/BasicSR/datasets/train/hr'
14   dataroot_LR: 'C:/Users/leona/TCC/Projects/BasicSR/datasets/train/lr'
15   subset_file: null
16   use_shuffle: true
17   n_workers: 4
18   batch_size: 2
19   HR_size: 128 # patch size. Default: 128
20
21 # Rotations augmentations:
22   use_flip: true
23   use_rot: true
24 val:
25   name: 'Validation'
26   mode: 'LHRHOTF'
27   dataroot_HR: 'C:/Users/leona/TCC/Projects/BasicSR/datasets/val/hr'
28   dataroot_LR: 'C:/Users/leona/TCC/Projects/BasicSR/datasets/val/lr'
29 path:
30   root: 'C:\Users\leona\TCC\Projects\BasicSR'
31   pretrain_model_G: '../experiments/pretrained_models/RRDB_PSNR_x4_old_arch.pth'
32

```

Fonte: Autor.

No arquivo de configuração mostrado anteriormente, alguns parâmetros devem ter atenção especial. O parâmetro *scale* representa a proporção na qual deseja-se treinar o modelo, quatro nesse caso. Os parâmetros *dataroot_HR* e *dataroot_LR* representam o diretório com as imagens de alta e baixa resolução, respectivamente, para treino e validação. E por último, mas não menos importante, o atributo *pretrain_model_G* é o caminho onde está armazenado o modelo pré-treinado.

4.4.4 Descrição dos experimentos práticos de treinamento

O processo anterior foi feito a partir do zero para duas bases de imagens diferentes, criando uma cópia do projeto no final do processo por segurança. Desta forma, os resultados do modelo terão uma maior reproduzibilidade.

4.4.4.1 Executando o modelo

Com tudo configurado e preparado, basta executar o modelo, com o comando abaixo, seguindo as opções recomendadas na documentação do projeto:

Figura 35 – Comando para executar o treinamento.



```
1 python train.py --data_dir <caminho do projeto> --batch_size 2 --num_iter 1000 --crop
2
```

Fonte: Autor.

Estes parâmetros utilizados acima, foram os que funcionaram melhor após experimentar com o modelo. Perceba, que o *batch_size* é 2, assim como no projeto em Tensorflow mencionado anteriormente. No caso anterior, treinar com apenas duas imagens por vez era um impedimento, dada a extensão no tempo total do treinamento. Com este modelo atual no entanto, em cerca de dez horas o treinamento cobria próximo de 100 épocas com 1000 iterações cada. Um número razoável para o propósito do trabalho.

4.5 Coleta de dados

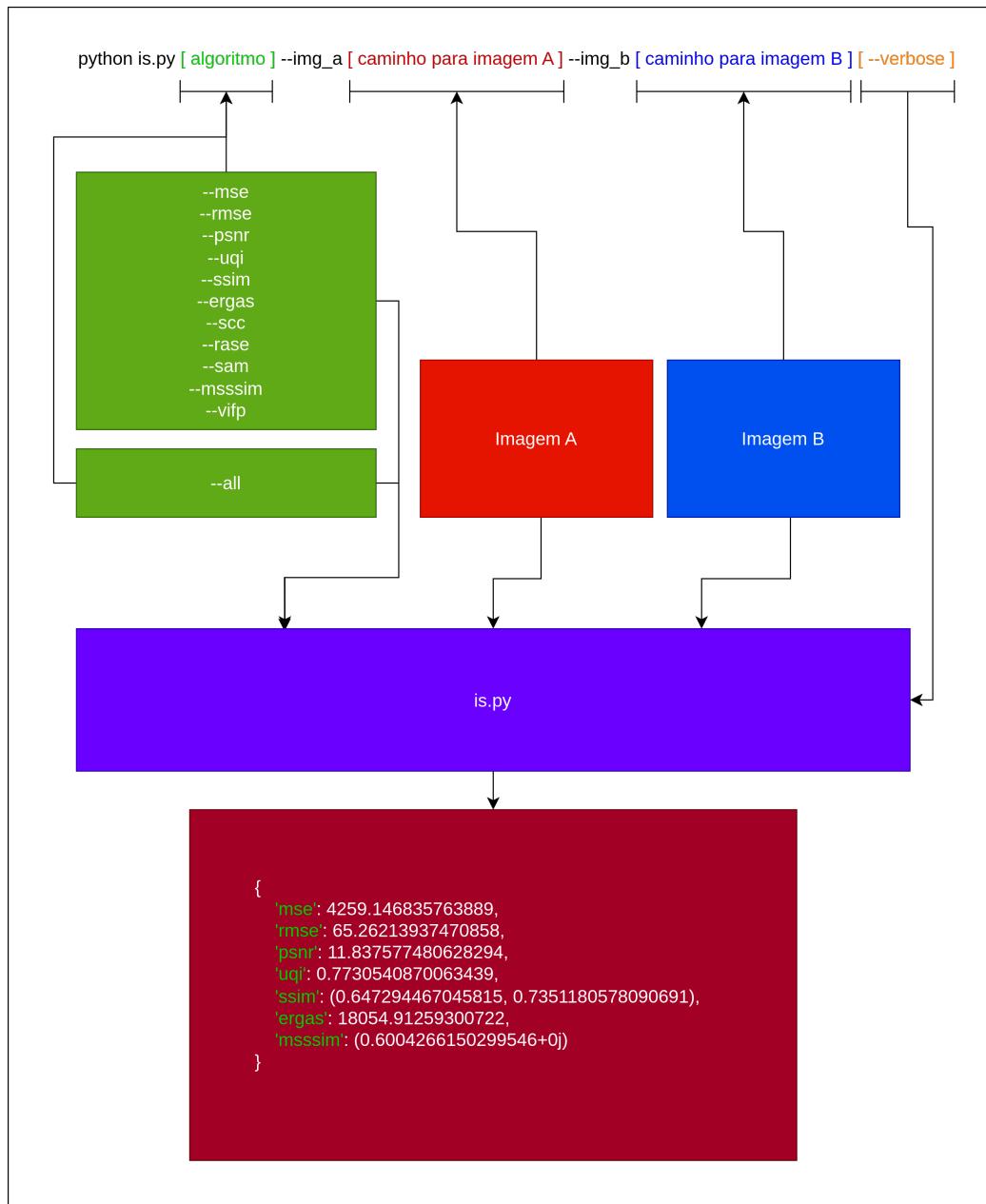
Nesta seção, será descrito o procedimento pelo qual a colheita de dados foi feita. Diversas automações foram feitas para extrair o máximo de dados com o mínimo de tempo. Quanto mais dados coletados, mais rica é a análise.

4.5.1 Experimentos de similaridade entre imagens

Como dito na seção 2.8, existem formas quantitativas de avaliar a qualidade de imagens. Um pequeno programa foi desenvolvido para lidar com essa tarefa de forma automatizada (VASCONCELOS, 2023). O programa executa em linha de comando, recebendo como parâmetros as duas imagens e os algoritmos de similaridade que será utilizado. O software tem os seguintes algoritmos de similaridade disponíveis: MSE, RMSE, PSNR, UQI, SSIM, ERGAS, SCC, RASE, SAM, MSSSIM e VIFFP.

Para o escopo deste trabalho, serão utilizados os algoritmos MSE, RMSE, PSNR e ERGAS. A sintaxe utilizada segue o seguinte padrão

Figura 36 – Diagrama de sintaxe do programa.



Fonte: Autor.

Na figura 36, o parâmetro *algoritmo* representa quais algoritmos serão utilizados para calcular a similaridade entre as imagens. Para executar todos os algoritmos, deve-se passar uma lista com todos os desejados, ou passar o parâmetro *-all*.

Para alimentar o programa com as imagens, os parâmetros *-img_a* e *-img_b* são utilizados. Estes recebem os caminhos das imagens. O parâmetro *-verbose*, caso passado, imprime algumas informações a mais.

4.5.1.1 Exemplo de execução do programa acima

Para o exemplo, as duas imagens abaixo serão utilizadas:

Figura 37 – Imagem A e imagem B utilizadas no teste.



Fonte: Autor (imagens e montagem).

Ao se executar o programa nas imagens da figura 37, o resultado demonstrado na tabela 3 é obtido.

Tabela 3 – Tabela de resultados da execução do programa.

Algoritmo	Resultado
MSE	4259,146835763889
RMSE	65,26213937470858
PSNR	11,837577480628294
ERGAS	18054,91259300722

Fonte: Autor.

Fazendo uma breve interpretação dos resultados, podemos concluir que a similaridade entre as imagens, como é de se esperar é baixa. A tabela 4, descreve os resultados.

Tabela 4 – Tabela de descrição dos resultados da execução do programa.

Algoritmo	Resultado	Análise
MSE	4259,146835763889	Quanto mais próximo de 0, mais similar.
RMSE	65,26213937470858	Quanto mais próximo de 0, mais similar.
PSNR	11,837577480628294	Quanto maior o valor, mais similar.
ERGAS	18054,91259300722	Quanto mais próximo de 0, mais similar.

Fonte: Autor.

Agora, apenas como meio de comparação, o programa será executado comparando agora, a imagem A com ela mesma. Os resultados estão na tabela 5.

Tabela 5 – Tabela de resultados da execução do programa com apenas a imagem A.

Algoritmo	Resultado
MSE	0,0
RMSE	0,0
PSNR	∞
ERGAS	0,0

Fonte: Autor.

4.5.2 Coleta das imagens para avaliação dos resultados

O treinamento das redes adversárias geradoras, produz algumas imagens com as quais o próprio programa de treinamento avalia a performance do modelo. No entanto, a quantidade de imagens produzidas durante o treinamento não é significativa para propriamente se avaliar como o modelo evoluiu após o treinamento.

Para coletar uma quantidade considerável de imagens, basta alimentar o modelo já treinado e armazenar a imagem resultante. No modelo utilizado, é necessário organizar as imagens desejadas em um diretório específico, obedecendo obviamente a estrutura requerida e preencher, com o caminho do diretório, o arquivo de configuração.

4.5.2.1 Bases de dados utilizadas para a coleta de dados

- Base de dados médica 4.1.0.1
- Base de dados astronômica 4.1.0.2

4.5.2.2 Programa para extração de estatísticas das imagens

O processo de alimentar uma imagem em resolução reduzida à rede já treinada para em seguida formalmente compará-la, utilizando-se dos métodos de similaridade apresentados anteriormente (2.8) à sua versão original é tedioso e mecânico: características comuns em tarefas automatizáveis.

Para tornar o processo mais eficiente, um programa codificado em *Python* foi implementado. O programa, de forma superficial realiza as seguintes tarefas:

1. De um arquivo de configuração, lê uma lista de bases de dados contendo as seguintes informações sobre cada:
 - Nome
 - Caminho do diretório contendo os arquivos originais (imagens em alta resolução)
 - Caminho do diretório contendo os arquivos extraídos da RAG (as mesmas imagens acima, com resoluções reduzidas e alimentadas à RAG)
 - Extensão dos arquivos

2. Para cada uma das bases de dados, o programa:

- Lê as imagens do diretório com as imagens já passadas pela RAG
- Encontra a imagem equivalente no diretório contendo as imagens originais
- Como as imagens originais podem ter dimensões levemente diferentes das imagens que saem da RAG, o programa redimensiona a maior entre as duas imagens, para as dimensões da menor.
- Utilizando a biblioteca mencionada na seção 4.5.1, produz um objeto contendo as estatísticas para todas as imagens contidas nos diretórios da configuração.
- Salva o objeto como um arquivo *JSON*

4.5.2.3 Análise dos resultados

Uma ferramenta renomada no meio de análise e processamento de dados, é o *Jupyter Notebook*. A ferramenta, consiste em um servidor, provedor de um interpretador interativo no navegador. Este interpretador, fornece de forma direta e eficaz um método para executar várias vezes seguidas, trechos de código, sem precisar de executar novamente o programa inteiro.

A ferramenta é perfeita para a análise dos dados produzidos na seção 4.5.2.2. Com os arquivos *JSON* em mãos, um para a base de dados de ressonância magnética e outro para a base de dados astronômica, é possível extrair destes os dados e assim criar visualizações para análise dos resultados.

5 Resultados

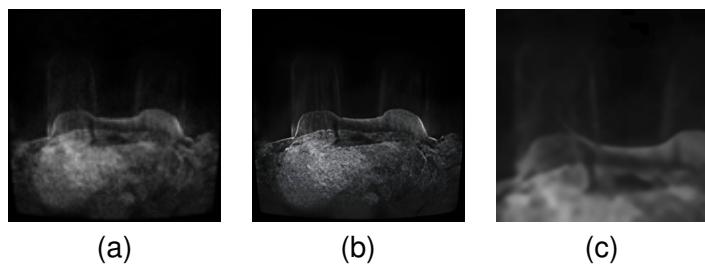
*“A vida é longa
se você sabe usá-la.”.
Lúcio Sêneca*

Nesta parte, serão descritos os resultados obtidos, após todo o desenvolvimento elaborado anteriormente. Abaixo, gráficos e seus respectivos comentários serão detalhados. Os objetivos desta seção são visualizar e compreender como o ETE (Erro do treinamento específico) se compara com ETG (Erro do treinamento genérico). Os indicadores de desempenho, serão os métodos matemáticos para verificar similaridade entre imagens, descritos na seção 2.8.

5.1 Amostras das imagens resultantes

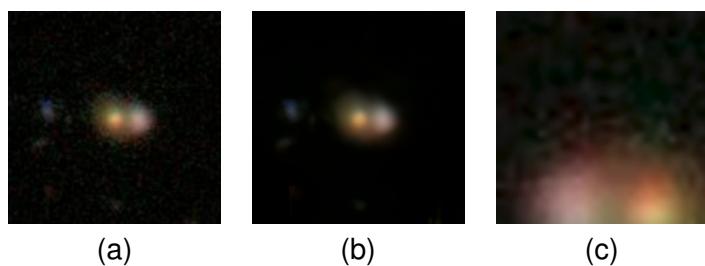
As figuras 38 e 39 são amostras aleatoriamente extraídas das bases de dados de ressonância magnética e astronômica, respectivamente. Em cada uma das figuras, a imagem **a** representa a imagem original assim como foi obtida da fonte. A imagem **b** representa a imagem **a**, após ser comprimida e passada pelo modelo treinado com o TG (Treinamento genérico). A imagem **c**, representa a imagem **a**, comprimida e alimentada ao modelo treinado pelo TE (Treinamento específico).

Figura 38 – Amostra aleatoriamente capturada das imagens de ressonância.



Fonte: Autor

Figura 39 – Amostra aleatoriamente capturada das imagens de astronomia.



Fonte: Autor

5.2 Visualização dos resultados

Todas as imagens abaixo, possuem um gráfico superior, com os valores reais dos erros calculados, assim como uma regressão polinomial de primeiro grau e uma regressão polinomial de maior grau. Este gráfico está representado em escala logarítmica para os erros MSE e RMSE, devido à grande dispersão observada na visualização. Dessa forma, a imagem é mais compacta e concisa.

A regressão de primeiro grau permite visualizar uma tendência geral dos dados e a regressão de maior grau aproximará ainda mais dos pontos distribuídos. As regressões fornecem uma melhor ideia de qual dos casos obteve um erro maior.

As imagens também possuem um histograma no gráfico inferior, representando a distribuição dos erros para ambos os cenários estudados, treinamento genérico e treinamento específico.

5.2.1 Considerações gerais sobre os gráficos de resultados

Todas as imagens abaixo contém, no gráfico geral, os seguintes componentes:

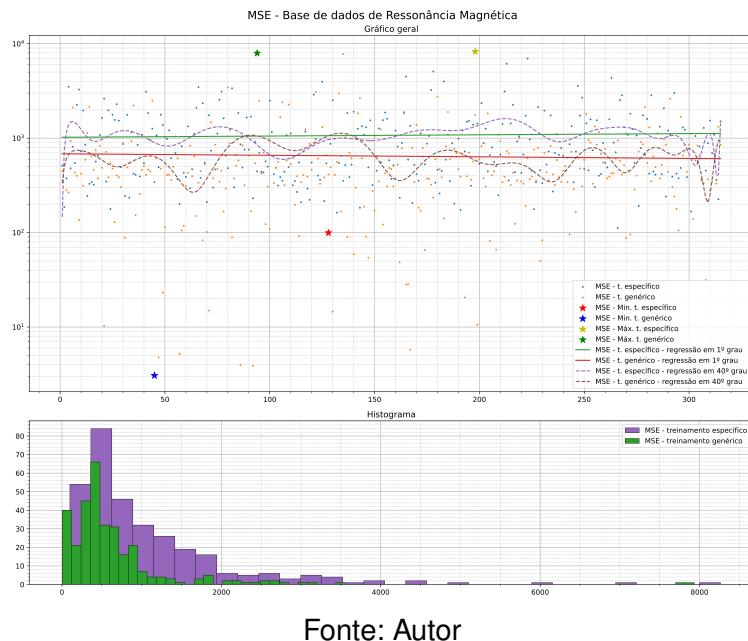
- Os pontos em azul, representando o valor do ETE.
- Os pontos em laranja, representando o valor do ETG.
- A estrela em vermelho, representando o valor mínimo do ETE.
- A estrela em azul, representando o valor mínimo do ETG.
- A estrela em amarelo, representando o valor máximo do ETE.
- A estrela em verde, representando o valor máximo do ETG.
- A curva em verde, representando o polinômio de primeiro grau, originado da regressão dos valores do ETE.
- A curva em vermelho escuro, representando o polinômio de primeiro grau, originado da regressão dos valores do ETG.
- A curva pontilhada em roxo, representando o polinômio de 40º grau, originado da regressão dos valores do ETE.
- A curva pontilhada em marrom, representando o polinômio de 40º grau, originado da regressão dos valores do ETG.
- Um histograma com a distribuição dos erros na parte inferior das imagens

5.2.2 Estudo de resultados envolvendo a base de dados de ressonância magnética

5.2.2.1 Erro quadrático médio (MSE)

Começando com o erro quadrático médio, ao comparar a similaridade das imagens super resolvidas pela rede genericamente treinada com a similaridade das imagens super resolvidas pela rede treinada especificamente para lidar com imagens de ressonância magnética, são obtidos os resultados abaixo:

Figura 40 – Cálculo de erro MSE para base de dados de ressonância magnética.



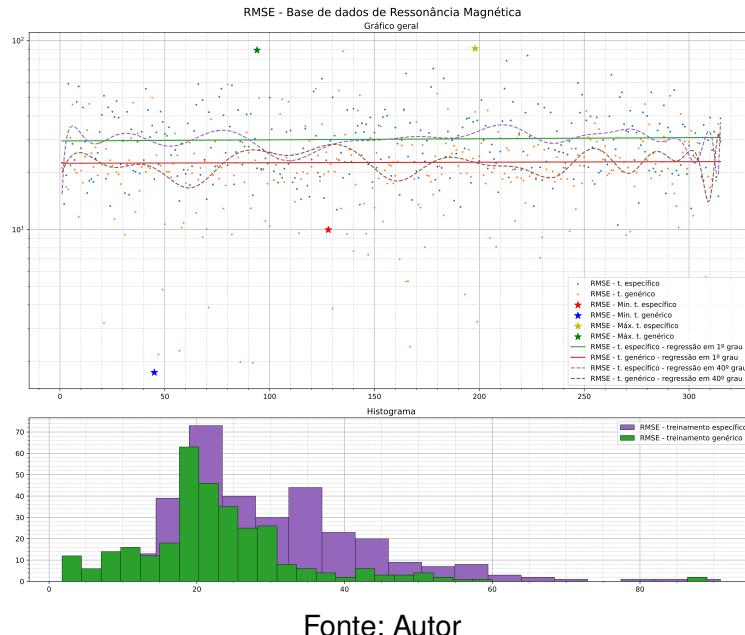
Fonte: Autor

Os erros da figura 40 foram maiores nas imagens treinadas de forma específica. É notável que os polinômios relativos ao treinamento específico estão, em sua maior parte, acima dos polinômios do treinamento genérico.

A mesma conclusão pode ser tirada, ao analisar a distribuição dos valores no histograma. A distribuição dos valores do ETE estão deslocadas mais à direita da distribuição do ETG.

5.2.2.2 Raiz do erro quadrático médio (RMSE)

Figura 41 – Cálculo de erro RMSE para base de dados de ressonância magnética.



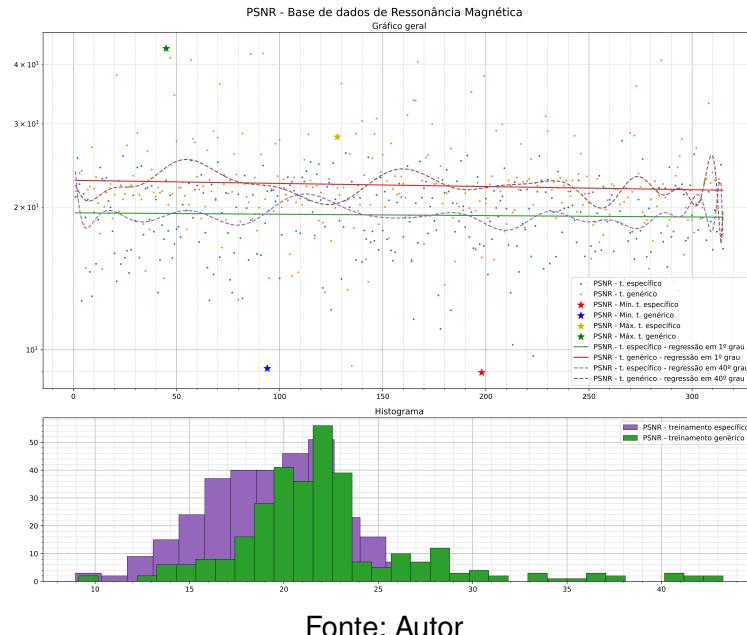
Fonte: Autor

O gráfico da figura 41 trás resultados muito similares aos da figura 40, porém a variação do erro é menor. Enquanto no caso da figura 40 os valores se concentram entre 10 e 10000 para todas as amostras, para o erro RMSE, o intervalo é mais restrito: entre 0 e 100. Isso se deve à raiz quadrada, inexistente no cálculo do erro MSE.

Os resultados que podem ser extraídos do gráfico e histograma são praticamente idênticos aos extraídos da figura 40. O treinamento específico da rede, produziu imagens com erros maiores que o treinamento genérico.

5.2.2.3 Relação sinal-ruído de pico (PSNR)

Figura 42 – Cálculo de erro PSNR para base de dados de ressonância magnética.



Fonte: Autor

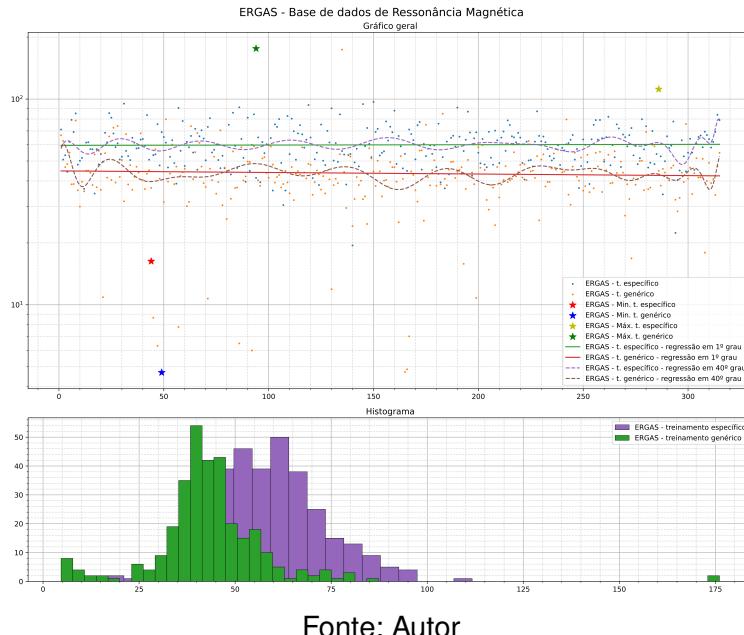
Para esse erro em específico, a análise se inverte. Como descrito na seção 2.8, a similaridade entre as imagens é diretamente proporcional ao erro PSNR, enquanto nos casos anteriores a proporção era inversa.

Na figura 42, mesmo utilizando uma escala logarítmica, observa-se bastante variação nas amostras. Seria difícil extrair significado destes dados sem as regressões polinomiais, dada tamanha dispersão.

Com as regressões, nota-se que o treinamento específico produziu erros PSNR mais próximos de zero em relação ao treinamento genérico. O histograma confirma esta análise, demonstrando o deslocamento à esquerda das amostras do treinamento específico.

5.2.2.4 Erro adimensional de síntese global relativa (ERGAS)

Figura 43 – Cálculo de erro ERGAS para base de dados de ressonância magnética.



Fonte: Autor

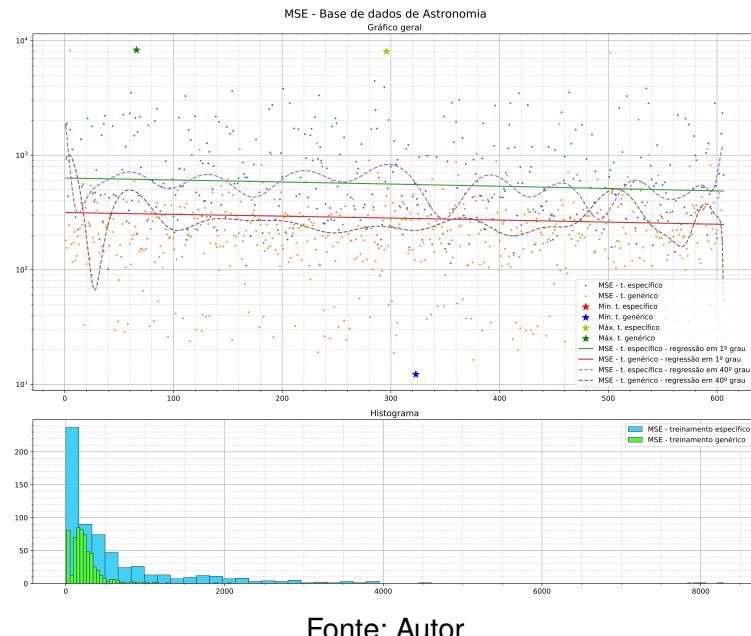
O erro ERGAS da figura 43, também produziu agrupamentos de amostras, mas dentro dos grupos, houve bastante variação entre os valores. Com as curvas de regressão no entanto, podemos claramente observar que o treinamento específico, novamente produziu erros maiores que o treinamento genérico.

Assim como nas figuras 40 e 41, a distribuição das amostras do ETE ficaram deslocadas à direita, evidenciando ainda mais o resultado antes descrito.

5.2.3 Estudo de resultados envolvendo a base de dados de astronomia

5.2.3.1 Erro quadrático médio (MSE)

Figura 44 – Cálculo de erro MSE para base de dados astronômica.

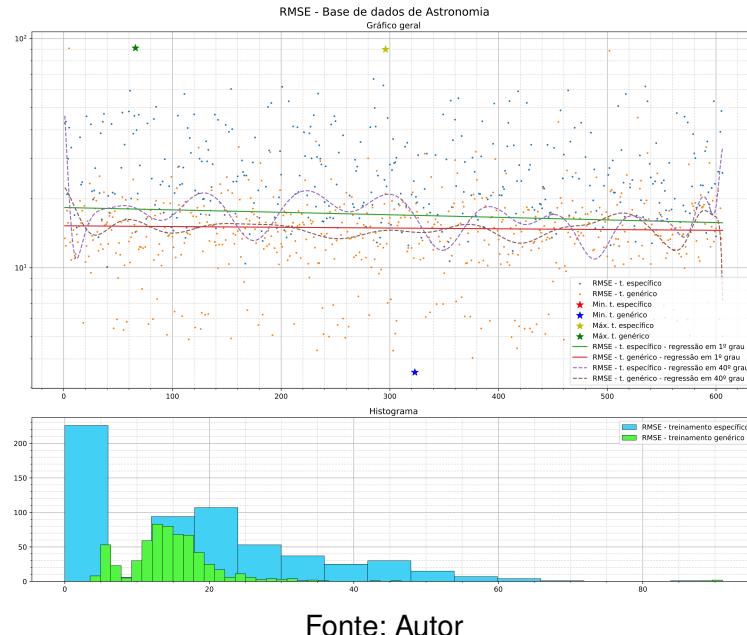


Fonte: Autor

Na figura 44, percebe-se com as regressões e o histograma que o ETE foi em sua maior parte, maior que o ETG. A distribuição do ETE se desloca à direita da distribuição do ETG.

5.2.3.2 Raiz do erro quadrático médio (RMSE)

Figura 45 – Cálculo de erro RMSE para base de dados astronômica.

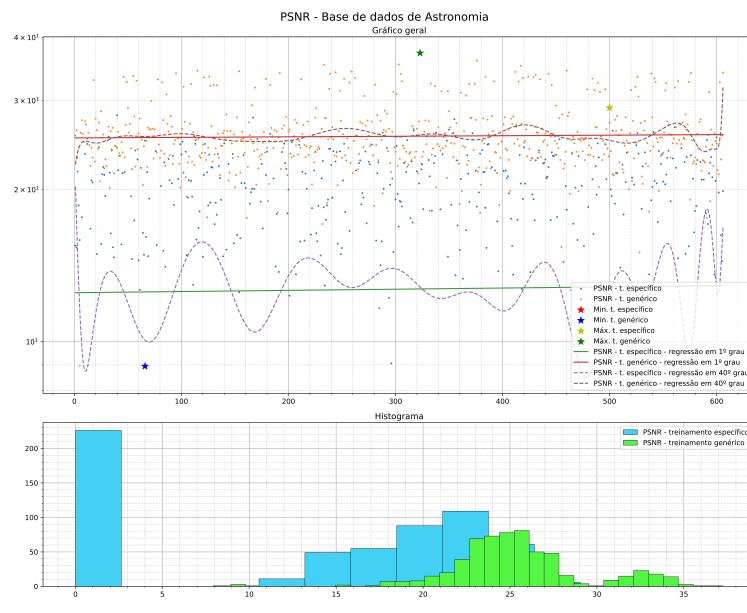


Fonte: Autor

A figura 45 traz resultados similares aos da figura 44: o ETE é maior que o ETG. A principal diferença, como explicado anteriormente é o intervalo de distribuição reduzido de ambos os erros em relação à figura 44.

5.2.3.3 Relação sinal-ruído de pico (PSNR)

Figura 46 – Cálculo de erro PSNR para base de dados astronômica.



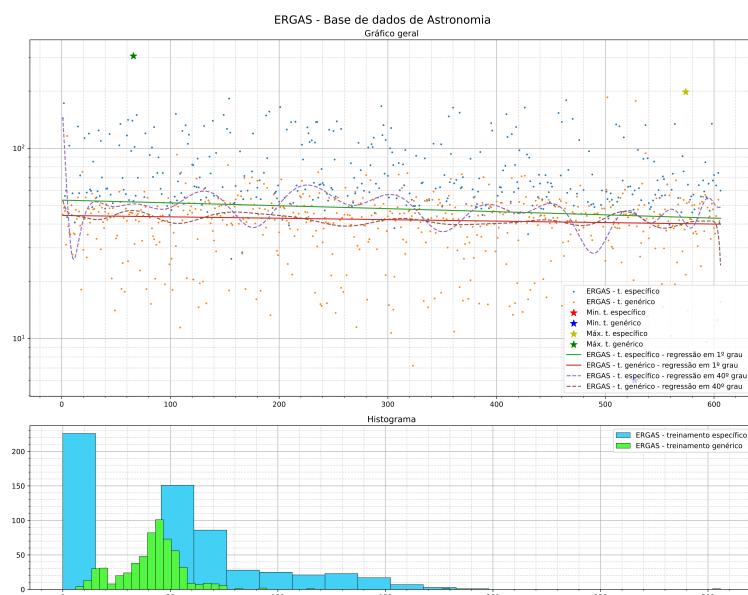
Fonte: Autor

No gráfico da figura 46 é possível observar que os dois cenários formam agrupamentos. O ETE está deslocado para baixo do gráfico. O resultado extraído disso são condizentes com o que foi visto nos erros anteriores. O treinamento específico gerou erros PSNR mais próximos de zero e estão distribuídos mais à esquerda dos erros do treinamento genérico.

Note também que o distanciamento entre as regressões do ETE e as regressões do ETG foi maior, em relação ao erro PSNR calculado para a base de dados de ressonância magnética na seção 5.2.2.3.

5.2.3.4 Erro adimensional de síntese global relativa (ERGAS)

Figura 47 – Cálculo de erro ERGAS para base de dados astronômica.



Fonte: Autor

Outra vez, na figura 47 o ETE é maior que o ETG e assim como no erro ERGAS da seção 5.2.2.4, a dispersão dificultaria a análise sem as regressões. Apesar disso, o resultado se mantém. O treinamento específico produziu imagens menos similares às originais.

5.3 Considerações gerais sobre o resultado

No geral, o TE produziu resultados inferiores ao TG e isso foi fortemente evidenciado em cada um dos gráficos descritos anteriormente. Constante e consistentemente o ETG produziu erros menores que o ETE, excluindo naturalmente o PSNR em que a análise se inverte, e em consequência, o maior valor é o mais atraente, do ponto de vista de similaridade. Com os ajustes necessários, o ETE sempre apontou na direção de menor similaridade com as imagens originais e isso é notável até mesmo de forma subjetiva e visual, como demonstrado nas imagens 38 e 39 na seção 5.1.

6 Conclusão

“Nem homem nem nação podem existir sem uma ideia sublime.”
Fiódor Dostoiévski

Com os resultados descritos de forma técnica, uma conclusão sob estes é possibilitada. A proposta do trabalho, de verificar se treinar redes adversárias gerativas especificamente para um determinado cenário produziria melhores resultados do que uma RAG treinada de forma genérica, foi fortemente desafiada pelos resultados, no quesito viabilidade. O trabalho documenta uma estratégia pouco otimizada de se realizar super resolução de imagens, especialmente se comparado ao estado da arte. No ponto de vista científico no entanto, este texto pode vir a conduzir futuros leitores para fora de um caminho pouco frutífero e que não trará resultados positivos.

Os resultados obtidos pelo TE (treinamento específico) foram constantemente inferiores aos resultados obtidos pelo TG (Treinamento genérico). Em todos os métodos de avaliação utilizados (MSE, RMSE, PSNR e ERGAS) o TE produziu um desempenho pior. Algumas hipóteses podem ser levantadas sobre as razões pelas quais tais resultados foram obtidos.

A primeira das hipóteses é a limitação de recursos computacionais para o treinamento específico. Para executar o treinamento na máquina disponível, apenas duas imagens eram colocadas em memória por iteração, ou seja, um *batch size* de dois. No contrário, problemas de memória impediriam o treinamento. Para fins de comparação, o modelo treinado por (WANG et al., 2018), utilizou um *batch size* de 16 e o modelo de (WANG et al., 2021), uma evolução do anterior, utiliza um *batch size* de 48.

Além desta limitação de memória, há também a limitação de processamento e, conectada a isso, as restrições de tempo. Redes neurais complexas como as que compõem a ESRGAN são, geralmente, treinadas em servidores de alto desempenho, contendo múltiplos chips gráficos com dezenas de milhares de núcleo. Os treinamentos realizados para este trabalho, foram feitas em uma máquina pessoal, ordens de grandeza menos potente que as máquinas utilizadas pela indústria. Enquanto treinamentos profissionais acontecem por dias, o dispositivo utilizado neste trabalho não estava disponível vinte e quatro horas por dia para o treinamento. Tudo isso acarretou na redução de mais um parâmetro de configuração: o número total de épocas.

Para ambas as bases de dados, o treinamento específico foi feito por 100 épocas com 1000 iterações cada, um número baixo em comparação à treinamentos mais profissionais. Apesar disso, o modelo pré-treinado, obrigatório para especializar as redes, pode ter ajudado o desempenho do TE não ter sido ainda mais inferior ao desempenho do TG. TG que por sua vez, foi feito por terceiros em um ambiente com recursos muito mais abundantes. Apesar de não ser conhecido o número de épocas exato utilizado no modelo treinado genericamente, é possível ter uma ideia aproximada deste valor. O artigo de (WANG et al., 2018) menciona treinamentos de 50000, 100000, 200000 e 300000 iterações. Um segundo artigo feito em cima de uma melhoria ao modelo ESRGAN (WANG et al., 2021) detalha treinamentos de 400000 e 1000000 de iterações. Valores totalmente impraticáveis para os recursos disponíveis para a execução deste trabalho. Um treinamento com esta duração envolveria um comprometimento temporal muito grande. Estes valores se traduziriam em semanas, quiçá meses de uso intenso e “ininterrupto” do dispositivo utilizado para o treinamento, algo que foge do escopo do presente trabalho. A palavra “ininterrupto” foi escrita entre aspas por um motivo. O treinamento de fato,

não pode ser interrompido, porém, existem formas de configurar o projeto para salvar pontos de checagem do modelo. Isso abre as portas para um trabalho futuro: explorar treinamentos longos, porém pausados e analisar os resultados.

Ambos modelos treinados nos artigos citados anteriormente, contaram com recursos computacionais de alto nível para o treinamento das redes. Tais dispositivos reduzem consideravelmente o tempo de treinamento, como mencionado na seção 4.4.

Outra hipótese diz respeito à disparidade entre os erros da base de ressonância magnética e de astronomia. Comparando as seções 5.2.2 e 5.2.3, especialmente os erros PSNR (seções 5.2.2.3 e 5.2.3.3) nota-se que o TE sob a base astronômica, apesar de ter sido realizado no mesmo dispositivo, com os mesmos parâmetros, produziu resultados piores que o TE da base de ressonância magnética. Existem duas diferenças críticas entre as duas bases de dados. Primeiro, a base de dados astronômica possui imagens coloridas. Segundo, como visto na seção 4.3, as imagens de ressonância magnética são menores em disco que as imagens astronômicas. Ambas as diferenças impõem obstáculos para o TE das imagens astronômicas.

Imagens coloridas, são mais complexas estruturalmente que imagens monocromáticas (vide diagrama da figura 26). Considerando os mesmos recursos e parâmetros utilizados para um treinamento, é seguro dizer que o treinamento realizado sobre imagens coloridas produzirá resultados inferiores. Quando se trata de imagens coloridas, as dimensões das estruturas internas de dados aumentam em algumas vezes. Uma imagem de largura L, altura A em tons de cinza possui uma estrutura bidimensional L por A, cada posição contendo a intensidade do respectivo pixel. A mesma imagem, agora com três canais de cores (vermelho, verde e azul), possui a mesma estrutura, porém repetida três vezes, uma para cada canal. Logo, há mais informação para ser analisada e mais informação para as redes se especializarem. Um trabalho futuro pode ser extraído destas disparidades: refazer os treinamentos com bases de imagens mais uniformes.

O desempenho do TE, mesmo que inferior ao TG, poderia ter sido significativamente pior. Na seção de desenvolvimento 4.4.3.2, foi descrito que para treinar as redes, um modelo pré-treinado seria necessário, para que o treinamento em si não precisasse aprender do zero. Dessa forma, a redes puderam se especializar em cima de algo já treinado, mesmo que minimamente.

O maior desafio deste trabalho não foi a revisão bibliográfica, não foi compreender conceitos complexos de aprendizado de máquina e também não foi o esforço para encontrar e processar bases de imagens acessíveis de forma pública e gratuita. Apesar de todas estas fases citadas terem requerido um esforço significativo, o maior obstáculo, e em consequência, onde o maior aprendizado foi concretizado, foram as integrações de softwares de terceiros um com o outro, podendo-se utilizar apenas da documentação provida pelos desenvolvedores. Documentação que embora essencial, não foi suficiente para garantir o funcionamento de ponta a ponta. Bastante tentativa e erro foi empregada neste processo de integração.

O trabalho desenvolvido, aponta que dado recursos reduzidos ou disputados por outras tarefas, utilizar um modelo de super resolução baseado em redes geradoras adversárias do

tipo ESRGAN treinado de forma genérica, trará resultados mais satisfatórios.

7 Propostas de trabalhos futuros

*“Feliz é aquele que transfere o
que sabe e aprende o que ensina.”
Cora Coralina*

Durante todo o processo de pesquisa e execução deste trabalho, várias outras possíveis aplicações de super-resolução foram detectadas. Seja como uma inteira nova utilidade ou como uma variação deste trabalho. A ideia aqui é documentar algumas possíveis ideias extraídas do trabalho atual.

7.1 Uso de RAGs com imagens geográficas

7.1.1 Super resolução de imagens de satélites

Existe na internet um acervo bem completo e extenso de imagens aéreas e de satélites, algumas destas bem antigas. Ferramentas como *Google Earth* permitem viajar no passado, para visualizar como diversas regiões eram anos atrás. Estas imagens, apesar de impressionantes, têm baixa resolução. Geralmente, quanto mais antigas as imagens, menor a resolução. O trabalho proposto é de utilizar a super resolução com GAN para restaurar tais imagens e talvez até produzir um mapa mais refinado.

7.1.2 RAGs e drones de baixo custo para mapeamento geográfico

Fazer um mapeamento geográfico de uma região com detalhes suficientes para extrair detalhes de estruturas como prédios, casas etc. utilizando de técnicas de fotogrametria pode ser um processo caro. Equipamentos que possuem os recursos necessários e boa resolução têm preços consideráveis. Com uma forma confiável de fazer super resolução de imagens, equipamentos inferiores, e em consequência mais baratos, podem ser utilizados para fazer o escaneamento aéreo, e no pós processamento, a ideia é super resolver as imagens produzidas para prosseguir então com o mapeamento.

7.2 Uso de RAGs para compressão e descompressão de imagens e vídeos

Como dito no final da seção 1.1, o uso de RAGs para super resolução de imagens pode trazer uma forma de descompressão de imagens como subproduto. Todo mundo atualmente possui uma câmera poderosa em seus bolsos e armazenar grandes quantidades de fotos e vídeos, produzidos em grande escala diariamente, é um problema comum enfrentado pelo usuário médio. Caso seja possível comprimir estas imagens e vídeos – que não passam de uma sequência de imagens – sob demanda, descomprimi-los com fidelidade suficiente para que não seja um problema para o usuário, assume-se então que um método eficiente de descompressão de imagens foi desenvolvido.

7.3 Uso de RAGs para microscopia

7.3.1 Super resolução de imagens microscópicas

Seguindo a mesma linha deste trabalho, como trabalho futuro a sugestão seria de usar bases de dados existentes para aprimorar a qualidade das imagens de microscopia disponíveis

na internet. Aumentar a quantidade de detalhes em uma imagem de microscópio pode ser de grande ajuda para o público alvo deste tipo de conteúdo.

7.3.2 RAGs e microscópios caseiros de baixo custo

Existem diversas formas de se fabricar um microscópio com tecnologias acessíveis como *WebCams* de entrada. Apesar da facilidade para se produzir tal equipamento, como é de se esperar, a qualidade das imagens geradas não possui grande qualidade. A proposta, seria de utilizar tal ferramenta em conjunto com uma RAG para super resolver as imagens – talvez em tempo real, mostrando diretamente em um display a imagem já super resolvida. Este trabalho produziria uma forma barata e eficaz de se observar e analisar objetos microscópicos.

7.4 Uso de RAGs em monitoramento

7.4.1 RAGs e Câmeras de segurança

7.4.1.1 RAGs para redução de armazenamento (compressão e descompressão)

Um dos problemas existentes com sistemas de segurança é o armazenamento dos vídeos capturados. Até quando manter um vídeo armazenado? O máximo possível é a melhor alternativa. Contudo, o tempo máximo possível é obviamente limitado pelo armazenamento disponível. Os gravadores de vídeo digital (conhecidos como DVR) utilizados em sistemas de segurança costumam utilizar a gravação contínua até encher os discos rígidos. Uma vez completamente cheios, o armazenamento mais antigo vai sendo sobreescrito à medida que vídeo subsequente é gravado. A proposta seria comprimir estes vídeos, e posteriormente descomprimi-los com boa qualidade e fidelidade ao material original, para aumentar significativamente este período máximo de retenção de imagens.

7.4.1.2 RAGs para captura de detalhes refinados

Identificar objetos em imagens de câmeras de segurança não é uma tarefa simples. A resolução baixa muitas vezes não permite extrair informações importantes como identidade de pessoas ou identificação de veículos. Com uma forma eficiente de super resolver estas imagens, talvez seja possível mudar a forma de analisar vídeos e imagens de sistemas de segurança.

7.4.2 Monitoramento de trânsito

Assim como no caso acima (seção 7.4.1.2), o objetivo nesta proposta é aprimorar o detalhe de imagens produzidas por câmeras de baixa resolução. Obter identificação de veículos pode ser positivamente afetado por uma ferramenta de super resolução de imagens de câmeras de trânsito.

7.5 Super resolução para entretenimento

7.5.1 RAGs para jogos

O objetivo seria a utilização de RAGs para remasterizar jogos antigos originalmente produzidos para dispositivos de baixa resolução.

7.5.2 RAGs para filmes e séries

O intuito desta proposta é a utilização de RAGs para, assim como o caso acima (7.5.1), aumentar a resolução e definição da imagem. Existem diversas ferramentas que já fazem este tipo de tarefa. Este trabalho seria um estudo de caso. Neste tópico, um adendo poderia ser construir um aplicativo ou ferramenta para facilitar a execução destas super resoluções. Desta forma, o público não técnico poderia tirar proveito dos resultados.

7.6 Super resolução de áudio

Áudio digital possui uma frequência de gravação e uma taxa de amostragem limitadas pela capacidade dos equipamentos e software utilizados na gravação e mixagem. A proposta deste trabalho seria aumentar esta frequência sem reduzir significativamente a fidelidade do áudio. Existem formas tradicionais de ampliar esta frequência (ou resolução) como interpolação, mas estes mecanismos não são inteligentes o suficiente para detectar padrões que outrora podem ser identificados por um modelo de aprendizado de máquina como RAGs.

Uma RAG, composta por redes capazes de extrair contexto espacial dos dados de entrada, talvez seja capaz de interpolar estatisticamente amostras de áudio entre amostras existentes, criando assim um áudio digital mais encorpado e próximo à sua contraparte analógica.

7.7 Portabilidade do trabalho atual

O presente trabalho necessita de um desktop ou servidor rodando o modelo para executá-lo. Para utilizar a versão móvel, ainda assim é necessário um *backend* processando os dados e os enviando para o dispositivo móvel. Algumas bibliotecas de inteligência artificial como o *Tensorflow* possuem versões dedicadas para executar em dispositivos móveis. Alguns dispositivos possuem até mesmo hardware dedicado para este tipo de processamento, como a *Neural Engine* utilizada pela *Apple* e alguns de seus produtos. A proposta deste trabalho futuro seria portar o modelo atual ou um modelo similar para dispositivos móveis de forma a eliminar a necessidade de uma infraestrutura inteira para o funcionamento da ferramenta. Esta proposta pode ainda transformar o trabalho em um produto.

7.8 Estudo sobre a viabilidade de treinamentos pausados

Como mencionado na seção 6, existe a possibilidade de se realizar um treinamento pausado em ciclos. A proposta para este trabalho futuro seria experimentar sobre esta téc-

nica, coletando dados para elaborar sobre a viabilidade de se treinar um modelo por esse método. Planejamento e agendamento das atividades será essencial para realizar um treinamento longo. Diferentemente do trabalho atual, uma atividade dessas requer pontualidade para executar os ciclos de treinamento: enquanto neste trabalho, todos treinamentos do modelo foram feitos em uma única execução, na proposta, o treinamento seria feito por um número de épocas por execução, até atingir-se um resultado satisfatório.

Referências

- A Review of Image Super-Resolution. 2020. Disponível em: <<https://blog.paperspace.com/image-super-resolution/>>. Citado nas páginas 16 e 17.
- ABADI, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. nov. 2015. Citado na página 34.
- BLUEAMULET. *BlueAmulet/BasicSR*. 2023. Original-date: 2019-07-09T16:00:14Z. Disponível em: <<https://github.com/BlueAmulet/BasicSR>>. Citado na página 62.
- BROWNLEE, J. *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*. Disponível em: <<https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/>>. Citado nas páginas 24 e 25.
- C. Han et al. GAN-based synthetic brain MR image generation. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. [S.I.: s.n.], 2018. p. 734–738. ISBN 1945-8452. Journal Abbreviation: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). Citado na página 28.
- Cancer Imaging Archive. *Cancer Imaging Archive*. 2022. Disponível em: <<https://wiki.cancerimagingarchive.net/display/NBIA/Downloading+Images+Using+the+NBIA+Data+Retriever>>. Citado nas páginas 42, 43, 44 e 50.
- CHEN, J. *Nash Equilibrium: How It Works in Game Theory, Examples, Plus Prisoner's Dilemma*. 2022. Disponível em: <<https://www.investopedia.com/terms/n/nash-equilibrium.asp>>. Citado na página 30.
- Chrislb. *English: Diagram of an artificial neuron*. 2005. Disponível em: <https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png>. Citado na página 21.
- CURRY, H. B. The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, v. 2, n. 3, p. 258–261, out. 1944. ISSN 0033-569X, 1552-4485. Disponível em: <<http://www.ams.org/qam/1944-02-03/S0033-569X-1944-10667-3/>>. Citado na página 27.
- DAVIES, D. et al. *A Complete Guide to the Google RankBrain Algorithm*. 2020. Disponível em: <<https://www.searchenginejournal.com/google-algorithm-history/rankbrain/>>. Citado na página 34.
- DICOM Standard. *History*. 2019. Disponível em: <<https://www.dicomstandard.org/history>>. Citado na página 46.
- DICOM Standard. *Current Edition*. 2024. Disponível em: <<https://www.dicomstandard.org/current>>. Citado na página 46.
- DONG, C. et al. Image Super-Resolution Using Deep Convolutional Networks. *arXiv:1501.00092 [cs]*, jul. 2015. ArXiv: 1501.00092. Disponível em: <<http://arxiv.org/abs/1501.00092>>. Citado na página 19.
- ELDRIDGE, S. *Nash equilibrium / Definition, Examples, & Facts / Britannica*. 2022. Disponível em: <<https://www.britannica.com/science/Nash-equilibrium>>. Citado na página 30.

- FastMRI. *FastMRI*. 2022. Disponível em: <<https://fastmri.org/>>. Citado na página 42.
- GOODFELLOW, I. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160 [cs]*, abr. 2017. ArXiv: 1701.00160. Disponível em: <<http://arxiv.org/abs/1701.00160>>. Citado nas páginas 16 e 30.
- GOODFELLOW, I. J. et al. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, jun. 2014. ArXiv: 1406.2661. Disponível em: <<http://arxiv.org/abs/1406.2661>>. Citado nas páginas 16, 17, 28 e 29.
- GOODMAN, N. *Uber AI Labs Open Sources Pyro, a Deep Probabilistic Programming Language*. 2017. Disponível em: <<https://www.uber.com/en-GR/blog/pyro/>>. Citado na página 34.
- GULLI, A.; KAPOOR, A.; PAL, S. *Deep Learning with TensorFlow 2 and Keras - Google Books*. 2019. Disponível em: <https://www.google.com.br/books/edition/Deep_Learning_with_TensorFlow_2_and_Kera/hebZzAEACAAJ?hl=en>. Citado nas páginas 23, 25, 26 e 28.
- GUPTA, R.; SHARMA, A.; KUMAR, A. Super-Resolution using GANs for Medical Imaging. *Procedia Computer Science*, v. 173, p. 28–35, jan. 2020. ISSN 1877-0509. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1877050920315076>>. Citado nas páginas 16 e 17.
- HAYKIN, S. *Redes Neurais: Princípios e Prática*. [S.I.]: Bookman Editora, 2007. Google-Books-ID: bhMwDwAAQBAJ. ISBN 978-85-7780-086-5. Citado nas páginas 20 e 21.
- JEFF, D.; MONGA, R. *TensorFlow - Google's latest machine learning system, open sourced for everyone*. 2015. Disponível em: <<https://ai.googleblog.com/2015/11/tensorflow-googles-latest-machine.html>>. Citado na página 34.
- KARPATY, A. *PyTorch at Tesla - Andrej Karpathy, Tesla*. 2019. Disponível em: <<https://www.youtube.com/watch?v=oBklltKXtDE>>. Citado na página 34.
- KHALEDYAN, D. et al. Low-Cost Implementation of Bilinear and Bicubic Image Interpolation for Real-Time Image Super-Resolution. In: *2020 IEEE Global Humanitarian Technology Conference (GHTC)*. Seattle, WA, USA: IEEE, 2020. p. 1–5. ISBN 978-1-72817-388-7. Disponível em: <<https://ieeexplore.ieee.org/document/9342625/>>. Citado na página 16.
- KHALEL, A. *Sewar*. 2023. Original-date: 2018-08-23T09:11:28Z. Disponível em: <<https://github.com/andrewekhalel/sewar>>. Citado na página 31.
- KITSON, J. *Installing Tensorflow with CUDA, cuDNN and GPU support on Windows 10*. 2022. Disponível em: <<https://towardsdatascience.com/installing-tensorflow-with-cuda-cudnn-and-gpu-support-on-windows-10-60693e46e781>>. Citado na página 60.
- KUMAR, A. *Semantic Image Segmentation using Fully Convolutional Networks*. 2020. Disponível em: <<https://towardsdatascience.com/semantic-image-segmentation-using-fully-convolutional-networks-bf0189fa3eb8>>. Citado na página 26.
- LEDIG, C. et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. set. 2016. Disponível em: <<https://arxiv.org/abs/1609.04802v5>>. Citado nas páginas 19 e 20.

- LEDIG, C. et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *arXiv:1609.04802 [cs, stat]*, maio 2017. ArXiv: 1609.04802. Disponível em: <<http://arxiv.org/abs/1609.04802>>. Citado nas páginas 16, 17, 19, 29 e 30.
- LENAIL, A. *NN SVG*. 2023. Disponível em: <<https://alexlenail.me/NN-SVG/>>. Citado na página 33.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, v. 5, n. 4, p. 115–133, 1943. ISSN 1522-9602. Disponível em: <<https://doi.org/10.1007/BF02478259>>. Citado na página 21.
- Medical Connections. *DICOM DateTime Format*. 2007. Section: kb. Disponível em: <<https://www.medicalconnections.co.uk/kb/DICOM-DateTime-Format/>>. Citado na página 46.
- Medical Connections. *DicomObjects / DICOM API / SDK Toolkit*. 2011. Disponível em: <<https://www.medicalconnections.co.uk/DicomObjects/>>. Citado na página 46.
- MOREIRA, F. *GERAÇÃO DE IMAGENS EM SUPER RESOLUÇÃO COM REDES GERADORAS ADVERSÁRIAS*. 2019. Disponível em: <<https://repositorio.ufsc.br/handle/123456789/202494>>. Citado nas páginas 17 e 28.
- N00MKRAD. *MagickUtils*. 2022. Original-date: 2020-07-20T15:21:16Z. Disponível em: <<https://github.com/n00mkrad/magick-utils>>. Citado na página 51.
- NASROLLAHI, K.; MOESLUND, T. B. Super-resolution: a comprehensive survey. *Machine Vision and Applications*, v. 25, n. 6, p. 1423–1468, ago. 2014. ISSN 0932-8092, 1432-1769. Disponível em: <<http://link.springer.com/10.1007/s00138-014-0623-4>>. Citado nas páginas 19 e 20.
- NGAHANE, S.; BAER, J. *The Winding Road to Better Machine Learning Infrastructure Through Tensorflow Extended and Kubeflow*. 2019. Disponível em: <<https://engineering.spotify.com/2019/12/the-winding-road-to-better-machine-learning-infrastructure-through-tensorflow-extended-and-kubeflow/>>. Citado na página 34.
- NVIDIA. *GeForce 920M Dedicated Graphics for Laptops*. 2022. Disponível em: <<https://www.nvidia.com/en-us/geforce/gaming-laptops/geforce-920m/specifications/>>. Citado na página 54.
- NVIDIA. *CUDA Toolkit 12.1 Downloads*. 2024. Disponível em: <<https://developer.nvidia.com/cuda-downloads>>. Citado nas páginas 59 e 60.
- NVIDIA, N. *Advanced Driver Search official NVIDIA drivers*. 2023. Disponível em: <<https://www.nvidia.com/Download/Find.aspx>>. Citado na página 58.
- NVIDIA, N. *Download the latest official NVIDIA drivers*. 2023. Disponível em: <<https://www.nvidia.com/download/index.aspx>>. Citado na página 57.
- OpenfMRI. *OpenfMRI*. 2022. Disponível em: <<https://openfmri.org/>>. Citado na página 42.
- OpenNeuro. *OpenNeuro*. 2022. Disponível em: <<https://openneuro.org/>>. Citado na página 42.
- OZA, M.; VAGHELA, H.; SRIVASTAVA, K. Progressive Generative Adversarial Binary Networks for Music Generation. *arXiv:1903.04722 [cs, eess]*, mar. 2019. ArXiv: 1903.04722. Disponível em: <<http://arxiv.org/abs/1903.04722>>. Citado na página 28.

- Pydicom. *Pydicom*. 2022. Disponível em: <<https://pydicom.github.io/>>. Citado na página 46.
- PYTORCH, P. *PyTorch documentation — PyTorch 2.0 documentation*. 2023. Disponível em: <<https://pytorch.org/docs/stable/index.html>>. Citado na página 34.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, out. 1986. ISSN 0028-0836, 1476-4687. Disponível em: <<http://www.nature.com/articles/323533a0>>. Citado na página 27.
- RYLES, G. *What are CUDA Cores?* 2022. Disponível em: <<https://www.trustedreviews.com/explainer/what-are-cuda-cores-4226433>>. Citado na página 54.
- SAHA, A. et al. A machine learning approach to radiogenomics of breast cancer: a study of 922 subjects and 529 DCE-MRI features. *British Journal of Cancer*, v. 119, n. 4, p. 508–516, ago. 2018. ISSN 0007-0920, 1532-1827. Disponível em: <<http://www.nature.com/articles/s41416-018-0185-8>>. Citado na página 43.
- SDSS. *Sloan Digital Sky Survey*. 2024. Disponível em: <<https://www.sdss4.org/>>. Citado na página 42.
- SRIVASTAVA, A. *Astronomy Image Classification Dataset*. 2024. Disponível em: <<https://www.kaggle.com/datasets/abhikalpsrivastava15/space-images-category>>. Citado na página 42.
- SUN, N.; LI, H. Super Resolution Reconstruction of Images Based on Interpolation and Full Convolutional Neural Network and Application in Medical Fields. *IEEE Access*, v. 7, p. 186470–186479, 2019. ISSN 2169-3536. Conference Name: IEEE Access. Citado nas páginas 16 e 17.
- TAKEMURA, E. S. Algoritmos para Super-Resolução de Imagens Baseados nas Filtragens de Wiener e Adaptativa Usando a Transformada Wavelet. p. 82, 2013. Disponível em: <<http://www.pee.ufrj.br/index.php/pt/producao-academica/dissertacoes-de-mestrado/2013-1/2013121601-2013121601/file>>. Citado na página 16.
- Technical City. *NVIDIA GeForce 920M*. 2022. Disponível em: <<https://technical.city/en/video/GeForce-920M>>. Citado na página 54.
- VASCONCELOS, L. T. d. *leonamtv/image-similarity-scripts*. 2023. Original-date: 2023-03-21T01:44:08Z. Disponível em: <<https://github.com/leonamtv/image-similarity-scripts>>. Citado na página 64.
- VELASQUEZ, A. P.; LIND, E. *A performance comparison between CPU and GPU in TensorFlow*. 2019. Disponível em: <<http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1354858&dswid=8907>>. Citado na página 53.
- Vish. *How Much Data Is Created Every Day in 2021? [You'll be shocked!]*. 2020. Disponível em: <<https://techjury.net/blog/how-much-data-is-created-every-day/>>. Citado na página 16.
- WALD, L. Quality of high resolution synthesised images: Is there a simple criterion ? In: . SEE/URISCA, 2000. p. 99. Disponível em: <<https://hal.science/hal-00395027>>. Citado na página 32.
- WANG, X. et al. *Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data*. arXiv, 2021. ArXiv:2107.10833 [cs, eess]. Disponível em: <<http://arxiv.org/abs/2107.10833>>. Citado na página 80.

- WANG, X. et al. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. *arXiv:1809.00219 [cs]*, set. 2018. ArXiv: 1809.00219. Disponível em: <<http://arxiv.org/abs/1809.00219>>. Citado nas páginas 17, 19, 29, 30, 62 e 80.
- WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, v. 13, n. 4, p. 600–612, abr. 2004. ISSN 1941-0042. Conference Name: IEEE Transactions on Image Processing. Citado na página 32.
- WEISSTEIN, E. W. Text, *Convolution*. 2003. Publisher: Wolfram Research, Inc. Disponível em: <<https://mathworld.wolfram.com/Convolution.html>>. Citado na página 23.
- WESTON, A. *Understanding DICOM*. 2020. Disponível em: <<https://towardsdatascience.com/understanding-dicom-bce665e62b72>>. Citado na página 46.
- WIKIMEDIA. *MultiLayerNeuralNetwork*. 2021. Disponível em: <https://upload.wikimedia.org/wikipedia/commons/4/47/MultiLayerNeuralNetwork_english.png>. Citado na página 22.
- WIKIPEDIA. *Convolução*. 2020. Page Version ID: 59483262. Disponível em: <<https://pt.wikipedia.org/w/index.php?title=Convolu%C3%A7%C3%A3o&oldid=59483262>>. Citado na página 23.
- XU, J. et al. Diversity-Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 3940–3949. Disponível em: <<https://aclanthology.org/D18-1428>>. Citado na página 28.
- ZEMLIN, J. *Welcoming PyTorch to the Linux Foundation - Linux Foundation*. 2022. Disponível em: <<https://www.linuxfoundation.org/blog/blog/welcoming-pytorch-to-the-linux-foundation>>. Citado na página 34.
- ZHANG, A. et al. Dive into Deep Learning. *arXiv:2106.11342 [cs]*, jul. 2021. ArXiv: 2106.11342. Disponível em: <<http://arxiv.org/abs/2106.11342>>. Citado nas páginas 23, 24, 25, 26 e 28.
- ZHU, X. et al. GAN-Based Image Super-Resolution with a Novel Quality Loss. *Mathematical Problems in Engineering*, v. 2020, p. e5217429, fev. 2020. ISSN 1024-123X. Publisher: Hindawi. Disponível em: <<https://www.hindawi.com/journals/mpe/2020/5217429/>>. Citado na página 16.

Índice

- Aprendizado de máquina, 32
- Backpropagation, 27
- CNN, 23
- CPU, 53
- CUDA, 54, 57, 58, 60
- DICOM, 46
- Enhanced Super Resolution Generative Adversarial Networks, 29
- ERGAS, 32
- Erreur Relative Globale Adimensionnelle de Synthèse, 32
- ESRGAN, 29
- GAN, 28, 29
- GPU, 53, 57
- Inteligência artificial, 32
- Mean Squared Error, 31
- MLP, 20, 22, 23
- MSE, 31
- Multilayer Perceptron, 22, 23
- NVIDIA, 57, 58
- Peak Signal-to-Noise Ratio, 32
- PSNR, 32
- Pytorch, 34
- RAG, 28
- Redes Adversárias Geradoras, 28
- Redes Neurais Artificiais, 20
- Redes neurais convolucionais, 23
- RMSE, 32
- RNA, 20
- RNC, 23
- Root Mean Squared Error, 32
- Similaridade de imagens, 31
- SRGAN, 29
- Super Resolution Generative Adversarial Networks, 29
- Tensorflow, 34
- TPU, 54
- Trabalhos futuros, 84
- Treinamento, 53, 54
- Treinamento da rede, 27