

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO  
INSTITUTO DE COMPUTAÇÃO  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS OLIVEIRA SILVA

PREENCHIMENTO DE IMAGENS USANDO REDE NEURAL PROFUNDA (UNET)

RIO DE JANEIRO  
2023

MATHEUS OLIVEIRA SILVA

PREENCHIMENTO DE IMAGENS USANDO REDE NEURAL PROFUNDA (UNET)

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profa. Luziane Ferreira de Mendonça, D.Sc.  
Co-orientador: Prof. João Carlos Pereira da Silva, D.Sc.

RIO DE JANEIRO

2023

esperandominhaficha.png

MATHEUS OLIVEIRA SILVA

PREENCHIMENTO DE IMAGENS USANDO REDE NEURAL PROFUNDA (UNET)

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em \_\_\_\_ de \_\_\_\_\_ de \_\_\_\_\_

BANCA EXAMINADORA:

---

Luziane Ferreira de Mendonça  
D.Sc. (UFRJ)

---

João Carlos Pereira da Silva  
D.Sc. (UFRJ)

---

Daniel Sadoc Menasché  
D.Sc. (UFRJ)

---

Cláudio Miceli  
D.Sc. (UFRJ)

Dedico este trabalho primeiramente aos meus pais, que sempre estiveram ao meu lado, apoiando cada decisão que tomei, inclusive a de seguir na área de computação, mesmo quando eu não conseguia expressar claramente minha paixão por ela. À memória de meu avô e minha avó, que, embora não estejam mais entre nós, carrego a certeza de que estariam imensamente orgulhosos deste momento. Agradeço também a todos os amigos que conquistei ao longo desta jornada, em especial ao grupo "Winx", pois, sem dúvida, sem o apoio de cada um de vocês, esta conquista não teria sido possível.

## AGRADECIMENTOS

Antes de tudo, gostaria de expressar minha profunda gratidão aos meus orientadores, que demonstraram uma abertura e entusiasmo notáveis em relação ao tema escolhido. A dedicação deles, manifestada nas inúmeras horas de discussões em chamadas e reuniões, foi fundamental para a realização deste trabalho. Sem a orientação, paciência e expertise deles, certamente não teria sido possível alcançar o nível de detalhamento e profundidade que este trabalho possui.

Agradeço ao professor Daniel Sadoc pela oportunidade de atuar como monitor na disciplina de Avaliação e Desempenho por um ano. Esta experiência não apenas enriqueceu meus conhecimentos em simulações e análises estatísticas, mas também teve um impacto significativo em minha vida pessoal, graças à bolsa de monitoria que auxiliou em minha renda e em casa.

Também sou grato à equipe da COGG.AI, que se mostrou sempre disposta a acompanhar o progresso do meu projeto de TCC, oferecendo sugestões valiosas para seu aprimoramento. Um agradecimento especial ao meu supervisor, Flávio Franco Vaz, cujos insights sobre melhorias em minha rede neural foram cruciais para a qualidade do trabalho apresentado e que se tornaram fundamentais para futuras pesquisas e projetos.

*“All models are wrong, but some are useful.”*

George E. P. Box

## RESUMO

Este trabalho teve como principal objetivo desenvolver um modelo capaz de preencher imagens com uma precisão próxima da perfeição, uma tarefa conhecida como *inpainting*. Para alcançar esse objetivo, foi realizada uma revisão abrangente da literatura sobre o tema. O documento detalha a escolha da arquitetura U-Net para a tarefa, fornecendo uma explicação aprofundada de seu funcionamento específico para preenchimento de imagens. Além disso, são apresentados os métodos adotados para o preprocessamento das imagens e a criação de máscaras, seguindo uma metodologia de geração de dados em tempo real (*on the fly*). Os resultados obtidos pelo modelo são discutidos, com percepções particulares sobre suas limitações em determinados tipos de imagem, proporcionando uma compreensão clara das áreas em que o modelo se destaca e onde ainda há espaço para melhorias.

**Palavras-chave:** preenchimento de imagens. U-Net. preprocessamento de imagens. geração de dados customizados.

## ABSTRACT

This study primarily aimed to develop a model capable of inpainting images with near-perfect accuracy. To achieve this goal, a comprehensive literature review on the subject was conducted. The document elaborates on the choice of the U-Net architecture for the task, providing an in-depth explanation of its specific operation for image inpainting. Additionally, the methods adopted for image preprocessing and mask creation are presented, following an "on the fly" data generation methodology. The results obtained by the model are discussed, offering particular insights into its limitations on certain image types, providing a clear understanding of the areas where the model excels and where there is room for improvement.

**Keywords:** image inpainting. U-Net. image preprocessing. custom data generation.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Resultados de reconstrução via inpainting . . . . .	13
Figura 2 – Arquitetura original da U-NET . . . . .	15
Figura 3 – Caminho de codificação e decodificação da rede . . . . .	16
Figura 4 – Operação de Conv2D em uma imagem 5x5 com 2 filtros, gerando uma imagem 3x3 com 3 filtros. . . . .	20
Figura 5 – Operação de MaxPooling . . . . .	21
Figura 6 – U-Net do TCC . . . . .	21
Figura 7 – Convolução transposta . . . . .	24
Figura 8 – Função Sigmoide . . . . .	25
Figura 9 – Imagem mascarada aleatoriamente . . . . .	27
Figura 10 – Modelo evoluindo ao longo de 25 épocas com mil imagens de treino. . .	31
Figura 11 – Modelo evoluindo ao longo de 20 épocas com cem mil imagens de treino.	32
Figura 12 – Gráfico do coeficiente de Dice ao longo do treinamento usando cem mil imagens. . . . .	33
Figura 13 – Gráfico da função de perda ao longo do treinamento usando cem mil imagens. . . . .	34
Figura 14 – Evolução da imagem 1 . . . . .	41
Figura 15 – Evolução da imagem 2 . . . . .	42
Figura 16 – Evolução da imagem 3 . . . . .	43
Figura 17 – Evolução da imagem 4 . . . . .	44
Figura 18 – Evolução da imagem 5 . . . . .	45
Figura 19 – Evolução da imagem 6 . . . . .	46
Figura 20 – Evolução da imagem 7 . . . . .	47
Figura 21 – Evolução da imagem 8 . . . . .	48
Figura 22 – Evolução da imagem 9 . . . . .	49
Figura 23 – Evolução da imagem 10 . . . . .	50

## **LISTA DE ABREVIATURAS E SIGLAS**

GPU	Unidade de processamento gráfico ou placa de vídeo
RGB	Sistema de cores Vermelho, Verde e Azul (do inglês "Red, Green, Blue")

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>12</b>
1.1	CONTEXTUALIZAÇÃO DO PROBLEMA . . . . .	12
1.2	OBJETIVOS DO TRABALHO . . . . .	12
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA . . . . .</b>	<b>13</b>
2.1	DEEP LEARNING: UMA VISÃO GERAL . . . . .	13
2.2	INPAINTING: DESAFIOS E APLICAÇÕES . . . . .	13
2.3	REVISÃO DA LITERATURA . . . . .	14
<b>3</b>	<b>A ESCOLHA DA U-NET PARA INPAINTING . . . . .</b>	<b>15</b>
3.1	ORIGENS DA U-NET: SEGMENTAÇÃO DE IMAGENS . . . . .	15
3.2	ADAPTAÇÃO DA U-NET PARA INPAINTING . . . . .	17
<b>4</b>	<b>METODOLOGIA . . . . .</b>	<b>19</b>
4.1	IMPLEMENTAÇÃO DA U-NET . . . . .	19
4.1.1	Implementando bloco de convolução direta . . . . .	20
4.1.2	Bloco de deconvolução (Convolução Transposta) . . . . .	22
4.1.3	Entrada, Saída e Função de Ativação . . . . .	24
4.2	CONJUNTO DE DADOS . . . . .	25
4.3	GERAÇÃO DE DADOS "ON THE FLY" E DATA AUGMENTATION	26
4.4	MÉTRICAS UTILIZADAS . . . . .	27
4.5	CONFIGURAÇÃO DE HIPERPARÂMETROS . . . . .	28
<b>5</b>	<b>ANÁLISE DOS RESULTADOS . . . . .</b>	<b>31</b>
5.1	EVOLUÇÃO DOS RESULTADOS . . . . .	31
5.1.1	Modelo com mil imagens . . . . .	31
5.1.2	Modelo com cem mil imagens . . . . .	32
5.2	COMPORTAMENTO DAS MÉTRICAS DURANTE O TREINAMENTO	33
<b>6</b>	<b>DISCUSSÃO . . . . .</b>	<b>35</b>
6.1	LIMITAÇÕES DO MODELO . . . . .	35
6.2	SUGESTÕES DE MELHORIAS E TRABALHOS FUTUROS . . . . .	36
<b>7</b>	<b>CONCLUSÃO . . . . .</b>	<b>37</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>38</b>

<b>APÊNDICE A – EVOLUÇÃO DO MODELO AO LONGO DAS ÉPOCAS . . . . .</b>	<b>41</b>
--	-----------

## 1 INTRODUÇÃO

### 1.1 CONTEXTUALIZAÇÃO DO PROBLEMA

O *inpainting*, técnica que visa restaurar ou preencher regiões corrompidas ou ausentes em imagens, tem evoluído significativamente com o avanço da tecnologia (WEI; WU, 2022). Tradicionalmente, essa tarefa era realizada por meio de técnicas manuais em restaurações artísticas (BERTALMÍO et al., 2000). No entanto, com a era digital, o *inpainting* ganhou novas abordagens, passando de métodos determinísticos e heurísticas para soluções mais sofisticadas baseadas em aprendizado de máquina, e por ser um tipo de abordagem de modelo gerativo, muitos trabalhos na literatura usam arquiteturas GAN (GOODFELLOW et al., 2014). Recentemente, o deep learning (GOODFELLOW; BENGIO; COURVILLE, 2016) emergiu como uma abordagem promissora para essa tarefa, dada sua capacidade de aprender representações complexas de dados e generalizar a partir de grandes conjuntos de treinamento (HAYS; EFROS, 2007). Nesse contexto, a U-Net, originalmente projetada para segmentação de imagens, tem sido explorada para o *inpainting*, levantando questões sobre sua eficácia e adaptabilidade para essa nova aplicação (IIZUKA; SIMO-SERRA; ISHIKAWA, 2017).

### 1.2 OBJETIVOS DO TRABALHO

Este trabalho tem como foco principal explorar a aplicabilidade e eficácia da arquitetura U-Net para a tarefa de *inpainting*, usando uma arquitetura mais simples do que as que se encontram quando pesquisamos sobre o tema na literatura, mas almejando bons resultados. Inicialmente, busca-se compreender as particularidades da U-Net, sua origem na segmentação de imagens (RONNEBERGER; FISCHER; BROX, 2015) e como essa arquitetura pode ser adaptada para o preenchimento de imagens. A partir dessa base teórica, o objetivo é desenvolver um modelo de *inpainting* que utilize a U-Net, considerando as especificidades da tarefa e os desafios associados. Uma vez implementado, o modelo será avaliado em diferentes cenários e tipos de imagem, utilizando métricas apropriadas para quantificar sua eficácia. Finalmente, o trabalho visa identificar e discutir as limitações do modelo, propondo direções para pesquisas e melhorias futuras.

## 2 FUNDAMENTAÇÃO TEÓRICA

### 2.1 DEEP LEARNING: UMA VISÃO GERAL

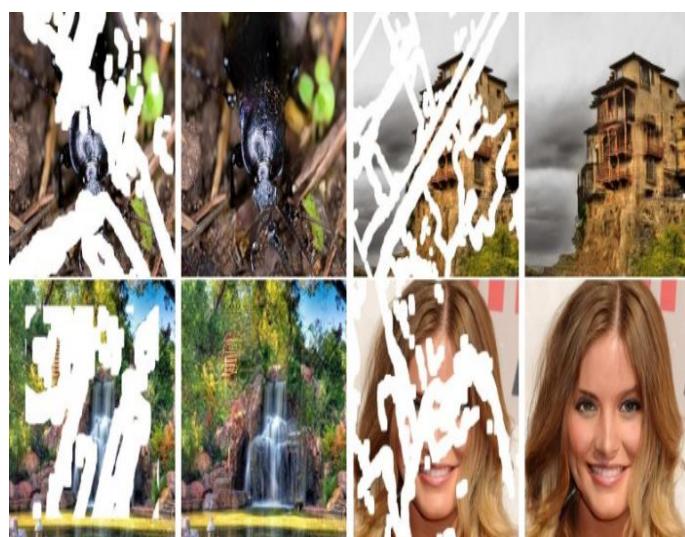
*Deep Learning*, ou aprendizado profundo, é uma subárea da inteligência artificial que se concentra em algoritmos inspirados pela estrutura e função do cérebro, chamados redes neurais artificiais. Enquanto as redes neurais têm suas raízes nas primeiras décadas da computação, o termo “profundo” refere-se ao número de camadas que essas redes podem ter. As redes neurais profundas, com muitas camadas, permitem a modelagem de funções complexas e a captura de padrões intrincados em grandes conjuntos de dados.

O ressurgimento do interesse pelo deep learning nas últimas décadas pode ser atribuído a vários fatores. Primeiramente, a disponibilidade de grandes volumes de dados, muitas vezes chamados de *big data*, forneceu o terreno fértil necessário para treinar modelos profundos. Em segundo lugar, os avanços em *hardware*, especialmente as unidades de processamento gráfico (GPUs), tornaram viável o treinamento de redes neurais com milhões, ou até bilhões, de parâmetros. Finalmente, inovações em técnicas de otimização e arquiteturas de rede permitiram que os modelos aprendessem mais eficientemente e evitassem armadilhas comuns, como o *overfitting* (VALDENEGRO-TORO; SABATELLI, 2022).

### 2.2 INPAINTING: DESAFIOS E APLICAÇÕES

*Inpainting* refere-se à técnica de restaurar ou preencher partes corrompidas ou ausentes de uma imagem de forma que a reconstrução seja visualmente coerente com as áreas circundantes da imagem.

Figura 1 – Resultados de reconstrução via inpainting



Historicamente, essa tarefa era predominantemente manual, com artistas restaurando fisicamente obras de arte danificadas. No entanto, com a digitalização, o *inpainting* ganhou uma abordagem computacional, onde algoritmos foram desenvolvidos para preencher automaticamente as regiões ausentes de imagens digitais.

Os desafios do *inpainting* digital são vastos. A restauração precisa ser feita de forma que as regiões preenchidas se harmonizem perfeitamente com o restante da imagem, preservando texturas, cores e padrões. Além disso, em muitos casos, o *inpainting* precisa inferir informações que não estão presentes na imagem original, o que exige uma combinação de técnicas determinísticas e heurísticas.

As aplicações do *inpainting* são variadas. Desde a restauração de fotos antigas e danificadas, passando pela remoção de objetos indesejados em imagens, até aplicações em vídeos, como a correção de falhas de transmissão. Com o advento do *deep learning*, o *inpainting* digital tem visto avanços significativos, com modelos capazes de realizar restaurações com um grau de realismo anteriormente inatingível.

### 2.3 REVISÃO DA LITERATURA

Ao explorar a literatura existente sobre *inpainting* utilizando técnicas de *deep learning*, dois trabalhos em particular chamaram a atenção devido à qualidade quase perfeita de seus resultados. O estudo (YU et al., 2019) introduziu o uso de convoluções controladas para *inpainting* e demonstrou uma capacidade notável de preencher áreas de imagens com formas livres de maneira precisa e realista. E o estudo (YU et al., 2018) também destacou-se pela implementação de mecanismos de atenção contextual no processo de *inpainting*. A técnica proposta permitiu que a rede neural “buscassem” texturas ou padrões em outras áreas da imagem para preencher regiões ausentes, resultando em restaurações de alta qualidade e visualmente indistinguíveis das partes originais da imagem.

Além desses trabalhos focados especificamente em *inpainting*, dois outros artigos foram fundamentais para a compreensão e implementação das redes neurais convolucionais utilizadas neste estudo. O artigo (RONNEBERGER; FISCHER; BROX, 2015) serviu como guia principal para a compreensão da arquitetura U-Net, enquanto (LONG; SHELHAMER; DARRELL, 2015) proporcionou uma base teórica sólida sobre redes neurais convolucionais conectadas. Ambos os artigos foram essenciais para estabelecer a fundação teórica sobre a qual este trabalho foi construído.

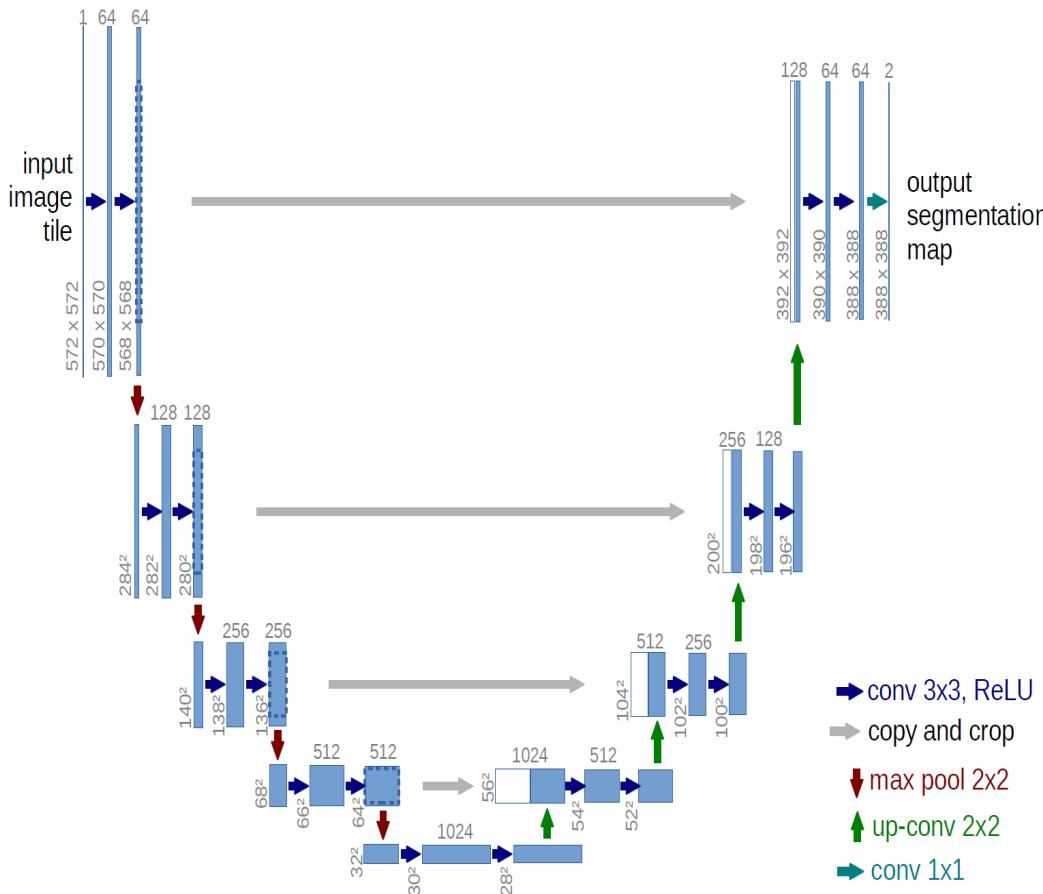
A excelência dos resultados apresentados por esses trabalhos e a profundidade teórica fornecida pelos artigos sobre U-Net e redes convolucionais conectadas reforçaram a decisão de investigar e desenvolver um modelo de *inpainting* baseado em *deep learning*, buscando alcançar um nível similar de perfeição.

### 3 A ESCOLHA DA U-NET PARA INPAINTING

#### 3.1 ORIGENS DA U-NET: SEGMENTAÇÃO DE IMAGENS

A U-Net (RONNEBERGER; FISCHER; BROX, 2015), uma arquitetura de rede neural convolucional, foi originalmente proposta para tarefas de segmentação de imagens biomédicas. A segmentação de imagens é um processo que envolve a classificação de cada pixel de uma imagem em uma categoria específica, como, por exemplo, distinguir células de tecido em imagens microscópicas.

Figura 2 – Arquitetura original da U-NET

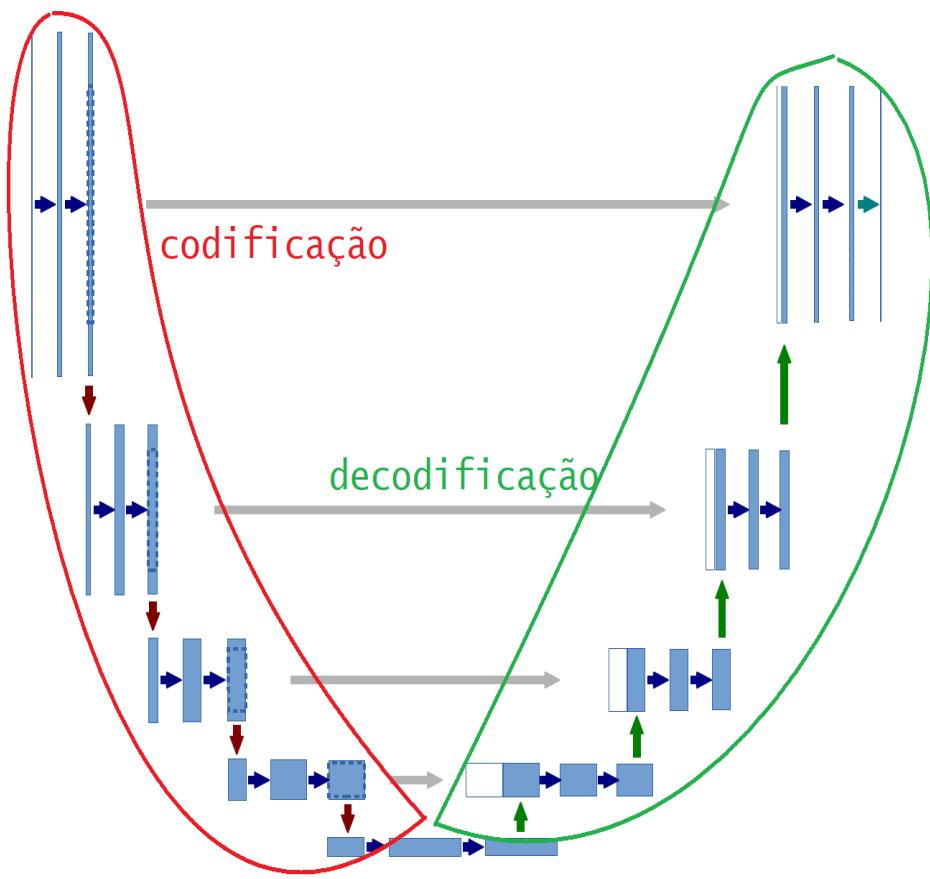


Fonte: (RONNEBERGER; FISCHER; BROX, 2015)

O que torna a U-Net particularmente eficaz para segmentação é sua arquitetura simétrica e em forma de “U”. Ela é composta por duas partes principais: a parte de contração (ou codificação) e a parte de expansão (ou decodificação).

Na parte de contração, a imagem é progressivamente reduzida em dimensão espacial enquanto aumenta em profundidade. Isso permite que a rede capture características mais

Figura 3 – Caminho de codificação e decodificação da rede



abstratas e de alto nível sobre a entrada, como a presença de formas complexas ou objetos inteiros. No entanto, com essa abstração, perde-se detalhes espaciais finos.

Na parte de expansão, a imagem é progressivamente reconstruída para sua dimensão espacial original. Aqui, as conexões de salto desempenham um papel crucial. Elas reintroduzem os detalhes espaciais perdidos durante a codificação, transferindo informações diretamente de uma camada da parte de contração para sua camada correspondente na parte de expansão. Isso permite que a rede combine informações contextuais de alto nível com detalhes espaciais de baixo nível, tornando-a extremamente eficaz para segmentação. Em outras palavras, a U-Net é capaz de “lembrar” os detalhes finos enquanto utiliza o contexto mais amplo, garantindo uma segmentação precisa.

Um aspecto distintivo da U-Net é, de fato, o uso dessas conexões de salto. Elas garantem que detalhes finos sejam preservados ao longo do processo, melhorando a precisão da segmentação.

A eficácia da U-Net em segmentação de imagens biomédicas levou a comunidade de pesquisa a explorar sua aplicabilidade em outras tarefas de visão computacional, incluindo *inpainting*, o que será discutido na próxima subseção.

Antes de falar das adaptações específicas da U-Net para *inpainting*, é essencial entender o conceito de “função de perda” em redes neurais. A função de perda, também conhecida

como função de custo, é um componente crucial no treinamento de redes neurais. Ela quantifica a diferença entre a saída prevista pela rede e o valor verdadeiro ou desejado. Essa medida de erro orienta o ajuste dos pesos da rede durante o processo de aprendizado, buscando minimizar a perda e, consequentemente, aprimorar o desempenho do modelo. Existem várias funções de perda, cada uma adequada a diferentes tipos de tarefas e objetivos de aprendizado. O entendimento correto dessa função é vital para a eficácia de qualquer modelo de aprendizado profundo.

### 3.2 ADAPTAÇÃO DA U-NET PARA INPAINTING

Embora a U-Net tenha sido originalmente projetada para segmentação de imagens, sua arquitetura flexível e capacidade de capturar informações contextuais e detalhes locais a tornaram uma escolha atraente para outras tarefas, incluindo *inpainting*. Na adaptação da U-Net para *inpainting*, a entrada da rede foi modificada para receber a imagem mascarada, que indica as regiões a serem preenchidas, geralmente representadas por desenhos pretos aleatórios ao longo da imagem. Essa modificação permite que a rede identifique com precisão onde a imagem precisa ser restaurada.

Além disso, a função de perda foi adaptada para enfatizar a fidelidade na região preenchida. Uma das funções de perda comumente utilizadas é o erro quadrático médio (MSE). O MSE calcula a média dos quadrados das diferenças entre os valores de pixel da imagem reconstruída e da imagem original, enfocando na redução da discrepância numérica entre elas, pixel a pixel. Isso é crucial para garantir uma reconstrução precisa em termos de intensidade e cor dos pixels.

Juntamente com o MSE, uma perda perceptual ou qualquer outro tipo de métrica é frequentemente empregada, no caso do trabalho proposto foi o coeficiente de Dice. Diferente do MSE, que se concentra na precisão numérica, o coeficiente de Dice busca assegurar que a imagem reconstruída e a imagem original sejam visualmente parecidas. A combinação dessas duas métricas de perda garante que o preenchimento seja não apenas numericamente preciso, mas também visualmente harmonioso e natural.

As conexões de salto da U-Net, características distintas da arquitetura, desempenham um papel crucial na tarefa de *inpainting*. Transferindo informações de camadas de contração para camadas de expansão, essas conexões garantem que detalhes locais da imagem original sejam utilizados para informar o preenchimento, resultando em restaurações mais naturais. Além disso, dada a natureza gerativa do *inpainting*, é vital garantir que a rede não caia em uma situação de overfitting, aprendendo a fazer isso apenas para o conjunto de dados. Para mitigar esse risco, são frequentemente empregadas técnicas que introduzem um elemento de aleatoriedade ou restrição no processo de aprendizado, reduzindo assim a probabilidade de que a rede memorize os dados de treinamento em vez de aprender a generalizar a partir deles. Dentre as técnicas mais utilizadas para este fim, podemos citar

o dropout e a regularização.

O dropout funciona desativando aleatoriamente alguns neurônios durante o treinamento, o que ajuda a rede a se tornar menos sensível a variações específicas nos dados de treinamento, promovendo assim uma melhor generalização. Por outro lado, a regularização, como a L1 ou L2, adiciona um termo de penalidade à função de perda para limitar a magnitude dos pesos da rede, evitando que ela se torne excessivamente complexa e específica para os dados de treinamento. Ambas as técnicas são eficazes em aumentar a robustez da rede ao sobreajuste, garantindo que ela possa realizar *inpainting* de maneira confiável em uma ampla variedade de imagens.

Ao adaptar a U-Net para *inpainting*, o objetivo principal é garantir que a rede possa preencher regiões ausentes de uma imagem de forma que o resultado seja indistinguível da imagem original para um observador humano. A combinação da arquitetura robusta da U-Net com modificações específicas para a tarefa de *inpainting* permite alcançar resultados de alta qualidade, tornando-a uma excelente escolha para essa aplicação.

## 4 METODOLOGIA

### 4.1 IMPLEMENTAÇÃO DA U-NET

A implementação foi realizada na linguagem Python utilizando a biblioteca TensorFlow, que oferece operações otimizadas de convolução e uma ampla gama de ferramentas para construção e treinamento de modelos de aprendizado profundo. É importante citar o guia (DUMOULIN; VISIN, 2016) que é uma leitura essencial para qualquer leitor que não domine bem a intuição de como são feitas as operações de convolução e deconvolução (convolução transposta).

Para entender melhor conceito de convolução, um pilar central em muitas tarefas de processamento de imagens, incluindo o *inpainting*, vamos formalizar como funciona. A convolução é um processo matemático aplicado a duas funções, gerando uma terceira função que expressa como o formato de uma é modificado pela outra. No contexto de processamento de imagens, a convolução envolve a aplicação de uma máscara (também conhecida como núcleo ou kernel) sobre a imagem.

A máscara é uma matriz pequena que percorre toda a imagem, centrada em cada pixel, para aplicar uma operação específica. Em termos matemáticos, a convolução discreta pode ser expressa pela seguinte fórmula:

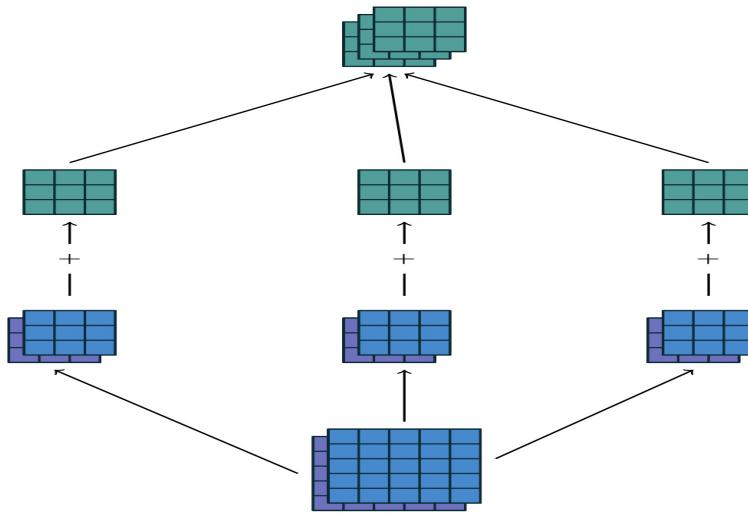
$$(f * g)(i, j) = \sum_m \sum_n f(m, n) \times g(i - m, j - n) \quad (4.1)$$

Nesta fórmula,  $f$  representa a imagem original,  $g$  é a máscara, e  $(i, j)$  são as coordenadas do pixel na imagem resultante. A convolução é essencialmente um somatório, calculando o produto ponto a ponto entre a máscara e a região da imagem sobre a qual ela se sobrepõe, e somando os resultados para produzir um único valor de pixel na imagem convoluída.

Um aspecto importante da convolução é o tratamento das bordas da imagem. Existem várias abordagens para lidar com pixels de borda, como extensão de borda, espelhamento ou ignorar os pixels de borda, cada uma impactando de maneira diferente o resultado da convolução.

Introduzir o conceito de convolução aqui prepara o terreno para discutir, na subseção seguinte, sobre convolução direta e outras técnicas avançadas, pois os conceitos básicos já foram estabelecidos. Entender a convolução é crucial para apreciar plenamente as nuances do inpainting e como as redes neurais, como a U-Net, aplicam esse processo para restaurar imagens de maneira eficaz. Na Figura 4 está um exemplo de como funciona exatamente o processo de convolução aplicado a imagens com  $n$  canais (filtros), onde no exemplo  $n = 2$ .

Figura 4 – Operação de Conv2D em uma imagem 5x5 com 2 filtros, gerando uma imagem 3x3 com 3 filtros.



Fonte: (DUMOULIN; VISIN, 2016)

#### 4.1.1 Implementando bloco de convolução direta

A operação de convolução direta<sup>1</sup> é a pedra angular das redes neurais convolucionais. No código apresentado, a função privada ConvBlock é responsável por aplicar duas camadas consecutivas de convolução direta à entrada fornecida, usando a operação Conv2D do TensorFlow. Cada camada de convolução utiliza um conjunto de filtros para extrair características da imagem de entrada. A quantidade de filtros e o tamanho do *kernel* são parâmetros definidos para cada bloco.

Após a aplicação da convolução, uma etapa opcional frequentemente utilizada é a camada de pooling, que tem o objetivo de reduzir as dimensões espaciais da saída ao mesmo tempo em que preserva as características mais importantes detectadas pela convolução. A operação MaxPooling2D é uma técnica comum de pooling, caracterizada por sua capacidade de selecionar o valor máximo dentro de uma janela definida, geralmente de tamanho 2x2, e deslocá-la ao longo da entrada.

A operação MaxPooling2D pode ser visualizada como uma janela deslizante que percorre a imagem: em cada posição da janela, apenas o valor máximo dentro da janela é retido. Matematicamente, se considerarmos uma matriz de entrada  $A$  e uma janela de pooling 2x2, a operação de MaxPooling2D em cada janela é dada por:

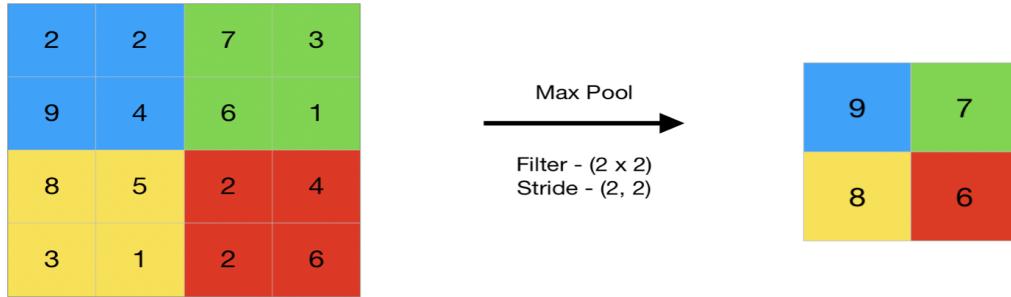
$$M(i, j) = \max(A[2i : 2i + 2, 2j : 2j + 2]) \quad (4.2)$$

---

<sup>1</sup> Convolução direta é um processo no qual o núcleo de convolução é aplicado diretamente sobre a imagem original, sem girá-lo ou invertê-lo, preservando assim a orientação espacial original do núcleo em relação à imagem.

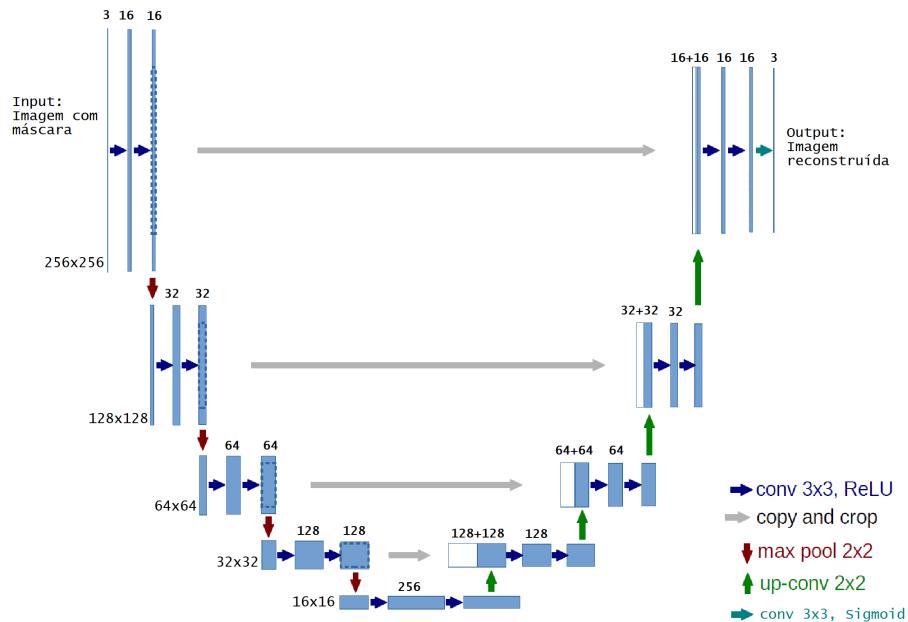
onde  $M$  é a matriz resultante da operação de pooling, e  $i, j$  são os índices da matriz  $M$ . Isso resulta em uma matriz  $M$  com metade das dimensões da matriz original  $A$ , pois a cada passo, a janela  $2 \times 2$  condensa quatro valores em um único valor máximo. Como pode ser visto na Figura 5 um exemplo com kernel  $2 \times 2$  e stride  $2,2$  ( stride determina o passo ou o deslocamento da janela do kernel ao longo da entrada ).

Figura 5 – Operação de MaxPooling



Na implementação específica da U-Net para este trabalho ( Figura 6 ), optou-se por uma estrutura profunda, composta por cinco blocos de convolução na parte de codificação. A escolha de um número elevado de blocos permite que a rede capture uma gama mais ampla de características, desde as mais superficiais até as mais complexas e abstratas. A progressão dos blocos foimeticulosamente projetada para lidar com diferentes níveis de abstracão da imagem.

Figura 6 – U-Net do TCC



No início da arquitetura U-Net, conforme ilustrado no primeiro bloco da Figura 6, a imagem de entrada, com dimensões de 256x256 pixels, é submetida a um processo de convolução inicial. Este processo envolve a aplicação de 16 kernels diferentes. Cada kernel, operando sobre os 3 canais de cores (RGB) da imagem, é responsável por extrair um tipo específico de característica ou padrão. Os 16 kernels geram, portanto, 16 mapas de características distintos após a convolução, que segue o padrão de cálculo demonstrado pela Figura 5, mas por exemplo, na Figura 4 é feito operação de convolução em algo com dimensão 5x5x2 ( 5x5 com 2 mapas de características ) gerando uma saída 3x3x2 e no caso da imagem de entrada da U-Net, estamos operando em dimensão 256x256x3, e gerando uma saída de dimensão 256x256x16. A razão para a aplicação dupla desses kernels, conforme indicado pelas duas setas na figura, é intensificar a extração de características, permitindo que a rede capture informações mais refinadas e complexas da imagem. Cada conjunto de convoluções é seguido por uma função de ativação, como a ReLU, para introduzir não-linearidade no processo de aprendizado. Esta etapa inicial é crucial para estabelecer uma base sólida de características que serão utilizadas nas etapas subsequentes de processamento pela U-Net.

A primeira fase da arquitetura U-Net é composta por cinco blocos de convolução, cada um seguido por uma operação de pooling. Em cada um desses blocos, a convolução é realizada utilizando filtros específicos, cuja quantidade e características evoluem progressivamente ao longo dos blocos. Inicialmente, o primeiro bloco utiliza 16 kernels, seguido pelo aumento progressivo para 32, 64, 128 e, finalmente, 256 kernels no quinto bloco, onde temos imagens de dimensão 16x16, e nesse ponto terminamos o caminho de codificação ( contração representada pela zona vermelha na Figura 3 ). Esse aumento sistemático no número de filtros permite que a rede refine sua capacidade de detecção de características à medida que avança nas camadas. Em termos práticos, enquanto os primeiros blocos se concentram em características mais genéricas, como bordas e gradientes, os blocos mais profundos são capazes de identificar padrões mais complexos, como texturas e formas específicas.

A decisão de dobrar o número de filtros em cada bloco, enquanto simultaneamente reduzimos pela metade a dimensão espacial através do *pooling*, é estratégica. Isso garante que, mesmo com a redução espacial, a capacidade da rede de capturar informações não é comprometida. Pelo contrário, essa abordagem permite que a rede mantenha um equilíbrio entre a resolução espacial e a profundidade de características, otimizando sua capacidade de representação em cada estágio da codificação.

#### 4.1.2 Bloco de deconvolução (Convolução Transposta)

Na segunda metade da arquitetura U-Net, representando a parte ascendente do 'U', utilizamos cinco blocos de convolução transposta, comumente referidos como 'deconvolução', para restaurar as dimensões espaciais da imagem. Este processo inicia com a imagem

de saída do quinto bloco da fase de convolução. Cada bloco de deconvolução utiliza um conjunto específico de kernels, começando com 256 e reduzindo progressivamente para 128, 64, 32 e, finalmente, 16, em um processo inverso ao observado na fase descendente (parte verde na Figura 3).

A operação Conv2DTranspose do TensorFlow é empregada para 'descomprimir' as características condensadas, efetivamente dobrando suas dimensões espaciais a cada bloco. Esse processo é o oposto do pooling realizado na fase descendente, onde as dimensões eram reduzidas pela metade. A convolução transposta consegue isso ao aplicar uma espécie de 'stride' inverso, onde a saída de cada operação é espacialmente maior do que a entrada.

Além da expansão espacial, um passo crucial nesta fase é a concatenação das características expandidas com as características correspondentes da fase de convolução. Essa junção é feita por sobreposição, onde os mapas de características da fase de deconvolução são alinhados e unidos com os mapas de características da fase de convolução, garantindo que a rede tenha acesso simultâneo às informações granulares (detalhes finos) e às informações mais abstratas (contexto mais amplo) durante a reconstrução da imagem. Este processo de concatenação é vital para a eficácia da U-Net, pois permite que a rede recupere detalhes finos da imagem enquanto também incorpora o contexto aprendido nas camadas mais profundas.

A convolução transposta, também conhecida como 'deconvolução', é uma operação fundamental na fase de expansão da arquitetura U-Net. Seu principal objetivo é aumentar as dimensões espaciais das características extraídas durante a fase de convolução. Em termos técnicos, a convolução transposta pode ser vista como o processo inverso da convolução regular.

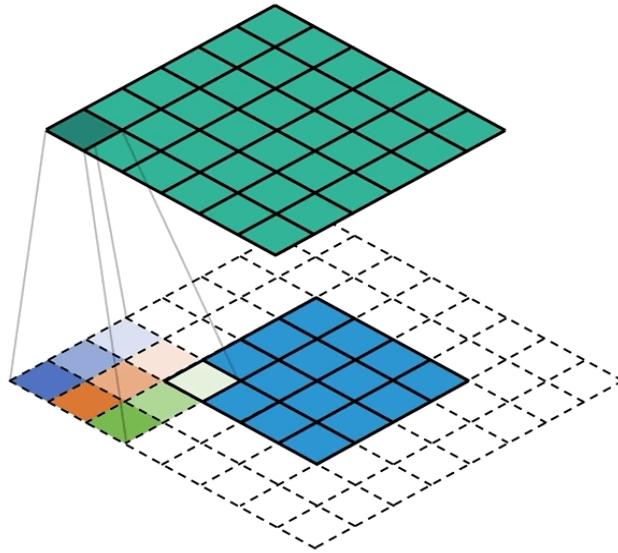
Enquanto a convolução regular aplica um kernel sobre a entrada, reduzindo suas dimensões espaciais, a convolução transposta opera de maneira a expandir estas dimensões. Isso é conseguido através da introdução de espaços 'vazios' entre os pixels da entrada. Em seguida, um kernel é aplicado de forma similar à convolução regular, mas ao invés de reduzir, ele expande a área de cobertura, 'preenchendo' esses espaços vazios.

Matematicamente, se considerarmos um kernel  $K$  e uma imagem de entrada  $I$ , a operação de convolução transposta pode ser expressa como:

$$O = K * I \quad (4.3)$$

onde  $*$  representa a operação de convolução transposta e  $O$  é a saída com dimensões espaciais ampliadas. Ao aplicar esta operação, os detalhes e as informações da entrada são 'espalhados' por uma área maior, permitindo a reconstrução de uma imagem com dimensões maiores que a entrada.

Figura 7 – Convolução transposta



#### 4.1.3 Entrada, Saída e Função de Ativação

A U-Net desse trabalho foi projetada para aceitar imagens coloridas como entrada, especificamente no formato de 256x256 pixels com valores normalizados e 3 canais de cor (RGB). A saída da rede é uma imagem reconstruída com as mesmas dimensões da entrada, onde as regiões corrompidas foram preenchidas.

Após a conclusão do último processo de deconvolução, a arquitetura construída produz um conjunto de mapas de características com dimensões (256, 256, 16). Para transformar esses mapas de características em uma imagem RGB reconstruída, é aplicada uma camada de convolução 2D com três filtros de tamanho 3x3. Esta camada de convolução produz um volume de saída com dimensões (256, 256, 3), correspondendo a uma imagem RGB.

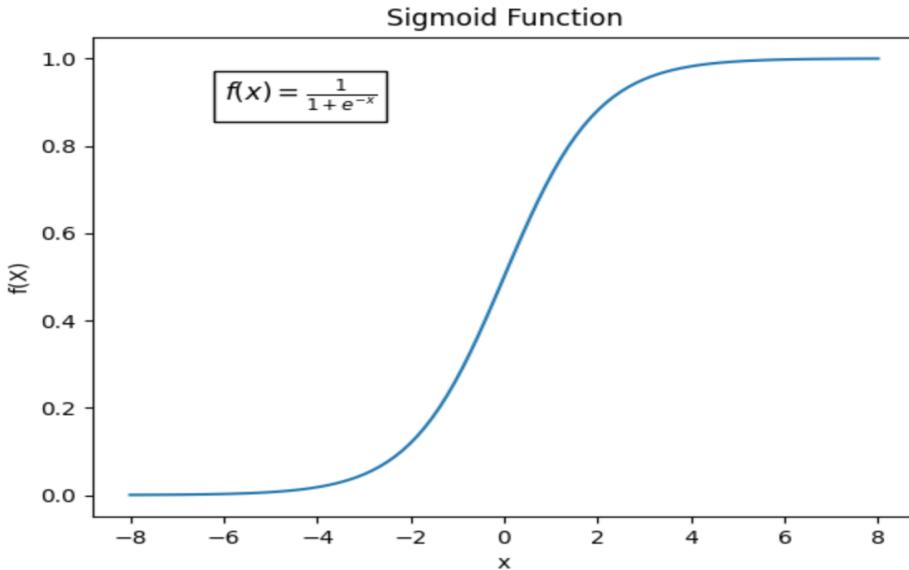
Funções de ativação são operações matemáticas aplicadas a cada nó de uma rede neural. Elas introduzem não-linearidade ao processo de aprendizado, permitindo que a rede aprenda padrões complexos e intrincados nos dados.

Sem a não-linearidade proporcionada pelas funções de ativação, uma rede neural, independentemente de sua profundidade, seria equivalente a um único perceptron linear, limitando severamente sua capacidade de modelar relações complexas nos dados. Portanto, as funções de ativação são essenciais para a eficácia das redes neurais em tarefas de aprendizado profundo.

No contexto da U-Net, especificamente na fase de reconstrução da imagem, a função de ativação sigmoide é empregada. Esta escolha é estratégica para tarefas de reconstrução de imagem. A função sigmoide mapeia os valores de entrada para um intervalo entre 0 e 1. Este mapeamento é particularmente útil para normalizar os valores do volume de saída, pois os valores dos pixels em imagens RGB normalizadas também variam entre 0

e 1. Assim, ao aplicar a sigmoide, asseguramos que a imagem reconstruída tenha valores de pixel dentro de um intervalo aceitável. Essa normalização é crucial para garantir que a imagem reconstruída possa ser comparada diretamente com a imagem original durante o treinamento e a avaliação da rede.

Figura 8 – Função Sigmoide



## 4.2 CONJUNTO DE DADOS

O conjunto de dados utilizado para treinar e avaliar a U-Net adaptada para inpainting foi o ImageNet (DENG et al., 2009), uma das bases de dados mais extensas e reconhecidas na área de visão computacional. O ImageNet é composto por milhões de imagens coloridas, variando em tamanho, mas muitas delas são redimensionadas para dimensões comuns, como 256x256 pixels, para facilitar o processamento. A diversidade no tamanho e na natureza colorida das imagens torna este conjunto de dados ideal para treinar redes neurais em tarefas complexas de processamento de imagem.

Além disso, o ImageNet é organizado conforme a hierarquia do WordNet, que é um grande banco de dados lexical da língua inglesa. No WordNet, as palavras são organizadas em conjuntos de sinônimos (synsets), cada um representando um conceito semântico. Essa organização hierárquica é benéfica para a visão computacional, pois fornece uma estrutura rica e detalhada para categorizar e relacionar as imagens, o que é crucial para tarefas que envolvem reconhecimento, classificação e agora, inpainting. Cada nó da hierarquia do WordNet no ImageNet é representado por centenas a milhares de imagens, refletindo uma ampla variedade de categorias e conceitos. Este arranjo facilita o treinamento de modelos de aprendizado profundo, permitindo que eles aprendam a distinguir e processar uma vasta gama de características visuais.

Neste trabalho, optou-se por uma amostragem do ImageNet, consistindo em 100.000 imagens selecionadas aleatoriamente. Esta seleção foi realizada de forma equitativa, garantindo que todas as classes do conjunto de dados tivessem a mesma chance de serem representadas. Dado o grande volume de imagens e a equidade na seleção, espera-se que a amostra ofereça uma representação balanceada das diversas classes, sem disparidades significativas na presença de qualquer categoria específica. A escolha por uma amostra em vez do conjunto de dados completo foi feita para tornar o treinamento mais gerenciável e rápido, sem comprometer significativamente a qualidade dos resultados. Muitos trabalhos na literatura também optam por amostras do ImageNet devido a suas dimensões massivas, garantindo assim que a amostra utilizada ainda é representativa e desafiadora o suficiente para validar a eficácia da U-Net para *inpainting*.

### 4.3 GERAÇÃO DE DADOS "ON THE FLY" E DATA AUGMENTATION

O preprocessamento de dados é uma etapa crucial em qualquer *pipeline* de aprendizado de máquina, e no contexto deste trabalho, foi ainda mais essencial devido à natureza do problema de *inpainting*. As 100.000 imagens selecionadas do ImageNet foram processadas de forma aleatória e normalizadas, garantindo que os valores dos pixels estivessem no intervalo [0,1], facilitando a convergência da rede durante o treinamento.

Uma das principais vantagens de se trabalhar com redes neurais convolucionais é a capacidade de aprender características hierárquicas dos dados. No entanto, para que a rede possa aprender características robustas e generalizáveis, é essencial fornecer uma variedade de dados durante o treinamento. No entanto, devido à limitação de memória e ao tamanho massivo do conjunto de dados, armazenar todas as variações possíveis de imagens em uma única variável seria inviável. Para contornar esse desafio, foi implementada uma técnica conhecida como *Data Augmentation*.

A técnica de Data Augmentation, essencial para enriquecer o conjunto de treinamento, foi adaptada especificamente para o contexto de inpainting. Em vez das transformações comuns, como rotação ou escala, a adaptação aqui envolve a criação de máscaras aleatórias nas imagens. Estas máscaras simulam áreas corrompidas ou faltantes, desafiando a rede a aprender a reconstruir a imagem original a partir de uma versão incompleta. Tal abordagem é crucial para aprimorar a capacidade da rede de lidar com cenários reais de inpainting, onde as omissões podem ser variadas e imprevisíveis.

Para implementar essa técnica de forma eficiente, foi desenvolvida a classe `createAugment`, baseada em uma referência de Stanford (AMIDI, 2018). Essa classe funciona como um gerador de dados dinâmico, produzindo *batches* de imagens mascaradas e suas correspondentes originais durante o treinamento. Em termos práticos, um *batch* refere-se a um conjunto de imagens processadas em grupo durante uma etapa do treinamento da rede. O tamanho do batch, neste caso definido para 32, determina quantas imagens são

processadas juntas, otimizando o uso da memória e acelerando o treinamento.

A função `random_mask2` é usada dentro da classe para aplicar linhas aleatórias nas imagens, atuando como máscaras. Estas linhas são geradas em posições e espessuras aleatórias, criando um amplo espectro de cenários de inpainting. O método `__getitem__` da classe `createAugment` é responsável por gerar cada 'batch' de imagens mascaradas, enquanto `__data_generation` aplica as máscaras e retorna o par de imagens mascaradas e originais. Assim, a classe `createAugment` não apenas facilita a implementação eficiente em termos de memória, mas também assegura a diversidade e a riqueza do conjunto de treinamento, tornando o modelo mais robusto e preparado para diferentes tipos de inpainting.

Figura 9 – Imagem mascarada aleatoriamente



#### 4.4 MÉTRICAS UTILIZADAS

A avaliação de modelos de aprendizado de máquina é fundamental para entender o desempenho de um algoritmo em tarefas específicas. No contexto do inpainting, a escolha de métricas adequadas é crucial para quantificar a qualidade das reconstruções produzidas pelo modelo. Neste trabalho, foram adotadas duas métricas principais: o Erro Médio Absoluto (MAE) (FILHO, 2022) e o coeficiente Dice (SØRENSEN-DICE..., 2022).

O Erro Médio Absoluto (MAE) é uma métrica de regressão que quantifica a média das diferenças absolutas entre as previsões e os valores reais. No domínio do inpainting, o MAE mede a discrepância média entre os pixels da imagem reconstruída e os pixels da imagem original. A métrica é expressa pela fórmula:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (4.4)$$

onde  $y_i$  representa o valor real,  $\hat{y}_i$  denota o valor previsto e  $n$  é o número total de observações.

Por outro lado, o coeficiente Dice, também conhecido como índice de similaridade de Sørensen-Dice, é uma métrica que quantifica a similaridade entre dois conjuntos de dados. No contexto do inpainting de imagens, ele mede o quão próximas são as áreas preenchidas na imagem reconstruída em comparação com a imagem original. O coeficiente varia de 0, que indica ausência de sobreposição, a 1, indicando sobreposição perfeita. A fórmula para calcular o coeficiente Dice é:

$$\text{Dice} = \frac{2|X \cap Y|}{|X| + |Y|} \quad (4.5)$$

onde  $X \cap Y$  representa a interseção entre os conjuntos de pixels, ou seja, os pixels que são similares em coloração e posição nas imagens prevista e real. O símbolo  $|\cdot|$  denota a contagem dos pixels, referindo-se à quantidade de pixels que correspondem na interseção ou nos conjuntos individuais. Portanto, o numerador  $2|X \cap Y|$  é o dobro da contagem de pixels similares entre a previsão e a realidade, e o denominador  $|X| + |Y|$  é a soma total dos pixels nas duas imagens, considerando suas respectivas áreas de preenchimento ou segmentação.

No desenvolvimento do modelo de inpainting, escolheu-se o otimizador Adam, uma técnica amplamente adotada em tarefas de aprendizado profundo devido à sua eficácia e eficiência. O otimizador Adam, introduzido por Kingma e Ba em seu trabalho seminal (KINGMA; BA, 2014), combina as vantagens dos métodos adaptativos de gradiente, como o AdaGrad, com os conceitos de momentum, oferecendo uma atualização eficiente dos pesos da rede neural. Ele ajusta a taxa de aprendizado de maneira adaptativa para diferentes parâmetros, o que facilita a convergência mais rápida e estável em comparação com otimizadores tradicionais, como o SGD (Stochastic Gradient Descent).

O uso do otimizador Adam no treinamento do modelo de inpainting foi motivado por sua capacidade de lidar de forma eficiente com grandes volumes de dados e sua robustez em navegar por superfícies de erro complexas. Essas características são particularmente benéficas para o contexto de inpainting, onde o modelo precisa aprender a reconstruir com precisão as partes faltantes ou corrompidas das imagens. Ao minimizar o Erro Médio Absoluto (MAE) como função de perda, o otimizador Adam ajuda a garantir que o modelo seja treinado de maneira eficaz, reduzindo o erro de reconstrução e melhorando a precisão geral das previsões.

## 4.5 CONFIGURAÇÃO DE HIPERPARÂMETROS

O ajuste de hiperparâmetros é uma etapa crucial no desenvolvimento de modelos de aprendizado de máquina. Estes hiperparâmetros, diferentemente dos parâmetros aprendidos durante o treinamento, são definidos antes deste e têm um impacto significativo no

desempenho e eficácia do modelo. A escolha apropriada dos hiperparâmetros pode não apenas acelerar o treinamento, mas também melhorar a generalização do modelo, ou seja, sua capacidade de performar bem em dados novos e não vistos durante o treinamento.

É essencial evitar problemas comuns como o treinamento de um modelo que é excessivamente complexo e específico para os dados de treinamento, conhecido como '*overfitting*'. Este fenômeno resulta em um modelo que, apesar de ter um desempenho excelente nos dados de treinamento, falha em generalizar para dados novos. Por outro lado, existe o '*underfitting*', que ocorre quando o modelo é demasiadamente simples, não conseguindo capturar a complexidade ou os padrões nos dados de treinamento, resultando em um desempenho deficiente tanto nos dados de treinamento quanto nos novos. A importância do ajuste de hiperparâmetros reside em sua capacidade de encontrar o equilíbrio ideal entre esses dois extremos, otimizando o desempenho do modelo. Hiperparâmetros mal ajustados podem levar a um treinamento ineficiente, enquanto uma configuração ideal pode melhorar significativamente a precisão e a eficiência do modelo.

No contexto deste trabalho, o ajuste de hiperparâmetros foi realizado de forma intuitiva e iterativa. Em vez de usar técnicas sistemáticas como busca em grade ou otimização bayesiana, optou-se por uma abordagem mais experimental. O número de épocas foi ajustado manualmente, observando-se o desempenho do modelo até que se alcançasse um resultado visualmente satisfatório, isto é, imagens reconstruídas que não estejam borradadas ou que deixem artefatos de borrão. Esta abordagem, embora menos sistemática, permite uma adaptação mais flexível às especificidades do conjunto de dados e do problema em questão, é pretendido em trabalhos futuros explorar melhor os hiperparâmetros se for buscar um modelo estado da arte.

O número de imagens selecionadas para treinamento foi baseado em intuição de outros projetos de visão computacional já realizados como por exemplo (OLIVEIRA; PITA, 2023), dessa vez foram usadas cem mil imagens, garantindo um equilíbrio entre a quantidade de dados e a capacidade computacional disponível. A decisão de aumentar o número de camadas na arquitetura da rede foi tomada com o objetivo de capturar detalhes mais refinados em diferentes escalas da imagem. A ideia é que camadas adicionais podem ajudar a rede a aprender representações mais complexas e detalhadas, beneficiando a tarefa de *inpainting*.

Em muitos cenários, a adição de camadas pode melhorar a capacidade da rede de capturar detalhes em diferentes níveis de abstração. Esta decisão está alinhada com estudos que mostram que redes mais profundas tendem a ter melhor desempenho em tarefas complexas de visão computacional, devido à sua capacidade de aprender hierarquias mais ricas de características (HE et al., 2016).

Apesar da abordagem adotada para o ajuste de hiperparâmetros ser mais experimental e intuitiva, ela foi fundamentada na observação direta da evolução do modelo e na análise de seu desempenho em relação ao conjunto de dados. A escolha de hiperparâme-

tros não é uma ciência exata e, muitas vezes, requer uma combinação de conhecimento teórico, experiência prática e experimentação. No contexto deste trabalho, as decisões tomadas buscaram otimizar o desempenho do modelo para a tarefa específica de *inpainting*, considerando as características únicas do conjunto de dados utilizado.

Na próxima seção, onde analisaremos o desempenho do modelo tanto visualmente quanto por meio de métricas, teremos a oportunidade de avaliar e entender melhor as implicações e os resultados das escolhas de hiperparâmetros que foram feitas. Esta análise nos permitirá refletir sobre a eficácia das decisões tomadas e, se necessário, considerar ajustes futuros para aprimorar ainda mais o desempenho do modelo.

## 5 ANÁLISE DOS RESULTADOS

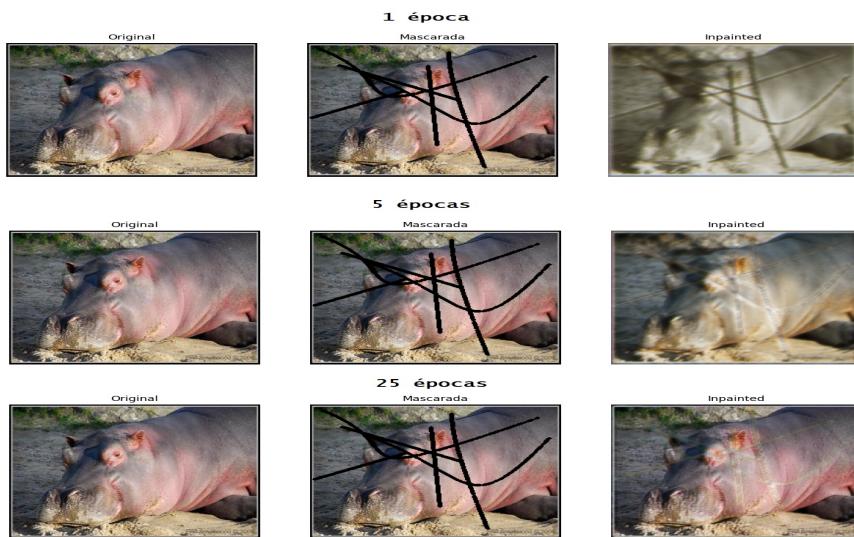
### 5.1 EVOLUÇÃO DOS RESULTADOS.

Com base em um entendimento prévio de visão computacional e aprendizado profundo por meio de convoluções, havia uma expectativa inicial sobre o comportamento do modelo durante as fases iniciais de treinamento. Era esperado que, nas primeiras tentativas de reconstrução, o modelo apresentasse imprecisões tanto na cor quanto na forma das imagens. Isso ocorre porque, nesse estágio, o modelo ainda está ajustando os pesos de cada filtro e aprendendo a reconstruir a imagem, compreendendo, por exemplo, a coloração da máscara, que representa a região que buscamos restaurar.

#### 5.1.1 Modelo com mil imagens

A metodologia adotada para a análise dos resultados envolveu testes controlados com um conjunto de mil imagens. Cada época de treinamento levava aproximadamente 15 segundos. Esse procedimento não só permitiu testar o funcionamento adequado das *callbacks* para salvamento de imagens e coleta de métricas, mas também ofereceu uma visão clara da evolução gradual do aprendizado da rede. Vale ressaltar que uma única época de treinamento com mil imagens implica em mil ajustes na rede neural. Assim, foi possível observar o modelo operando de maneira ainda rudimentar nas primeiras épocas, fornecendo insights valiosos sobre sua evolução.

Figura 10 – Modelo evoluindo ao longo de 25 épocas com mil imagens de treino.



A Figura 2 ilustra claramente as etapas iniciais de aprendizado do modelo. Como antecipado, nas primeiras tentativas, o modelo demonstra dificuldades em identificar a

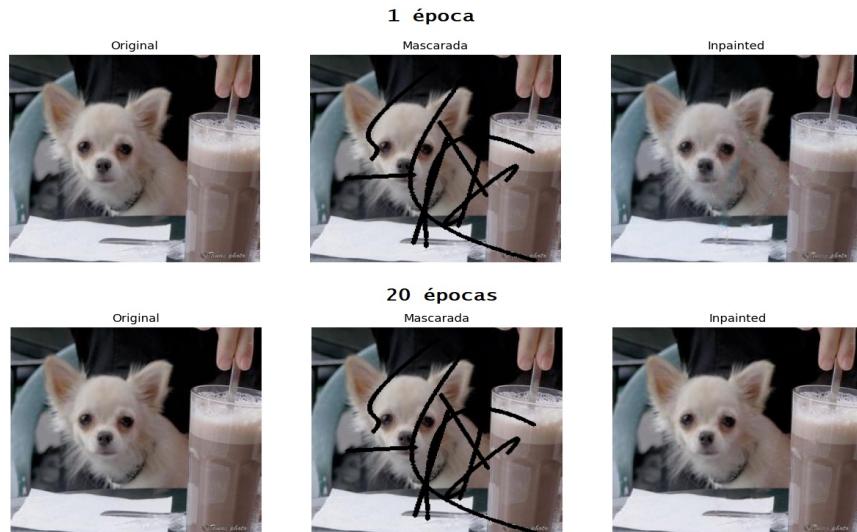
máscara, que representa a área a ser restaurada. A imagem resultante apresenta uma coloração significativamente distinta da original. Esse comportamento pode ser atribuído à estrutura da U-Net, que se baseia em um processo de codificação (*encoding*) seguido de decodificação (*decoding*). Até que o modelo ajuste adequadamente os pesos dos *kernels* das operações de convolução e deconvolução, é natural que ele não consiga reproduzir a imagem com precisão ou identificar corretamente a máscara.

Com o avanço das épocas de treinamento, observa-se uma melhoria notável na capacidade do modelo. Ele começa a recuperar a coloração original da imagem e reconhece que sua principal tarefa é restaurar as áreas completamente pretas, correspondentes à tripla RGB (0,0,0). Esse entendimento leva a um aumento na penalização nas regiões incompletas, otimizando o processo de reconstrução ao longo do tempo.

### 5.1.2 Modelo com cem mil imagens

Após a análise inicial com o conjunto reduzido de imagens, que teve um valor didático inestimável, procedeu-se com o treino utilizando um conjunto mais amplo de 100.000 imagens. Nesta fase, o modelo demonstrou resultados finais mais robustos e consistentes, validando as expectativas iniciais e consolidando a eficácia da arquitetura proposta.

Figura 11 – Modelo evoluindo ao longo de 20 épocas com cem mil imagens de treino.



Na Figura 3, observamos que, já na primeira época, o modelo apresenta um desempenho consideravelmente superior ao obtido após 25 épocas de treinamento com apenas mil imagens. Esse avanço pode ser atribuído ao volume significativamente maior de dados processados. Em uma única época com cem mil imagens, o modelo é exposto a mais exemplos do que em 20 épocas com mil imagens. Além disso, a diversidade ampliada de imagens contribui para um ajuste mais refinado dos pesos da rede.

Com essa perspectiva, não é surpreendente que, após 20 épocas de treinamento com cem mil imagens, a Figura 2 mostre uma reconstrução quase impecável. No entanto, nota-se uma imperfeição na restauração da marca d'água no canto inferior direito. Esse desafio pode ser atribuído à natureza do conjunto de dados ImageNet, que não é caracterizado por ter muitos textos. Além disso, como as máscaras são geradas aleatoriamente durante o treinamento, não há garantia de que elas sejam aplicadas sobre textos quando estes estiverem presentes na imagem. Na próxima seção, discutiremos como essa é uma das dificuldades comuns ao se buscar a perfeição em modelos de *inpainting*.

## 5.2 COMPORTAMENTO DAS MÉTRICAS DURANTE O TREINAMENTO

O monitoramento das métricas durante o treinamento de um modelo de aprendizado de máquina é essencial para entender sua evolução e identificar possíveis problemas ou oportunidades de otimização. No contexto deste trabalho, duas métricas foram consideradas: o coeficiente de Dice e a função de perda (*loss*).

O coeficiente de Dice, frequentemente usado em tarefas de segmentação, quantifica a similaridade entre duas amostras. Valores mais próximos de 1 indicam uma maior concordância entre a previsão do modelo e a verdadeira máscara da imagem. Por outro lado, a função de perda representa a diferença entre as previsões do modelo e os verdadeiros valores. Valores menores de perda indicam um melhor desempenho do modelo.

Figura 12 – Gráfico do coeficiente de Dice ao longo do treinamento usando cem mil imagens.

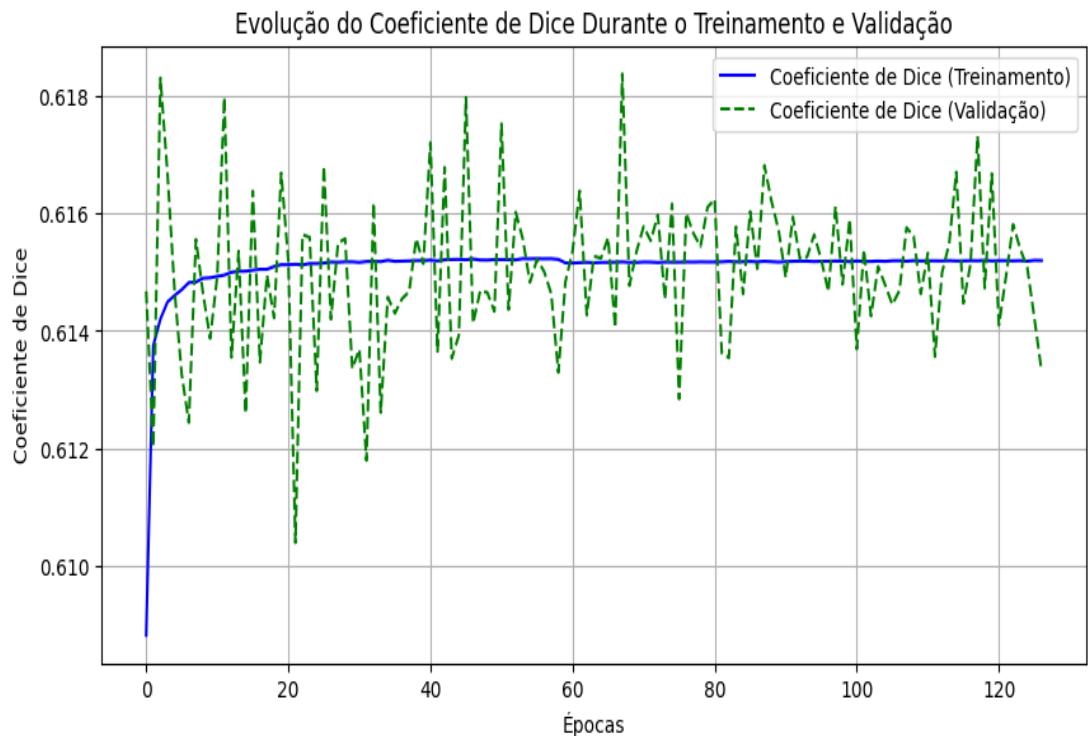
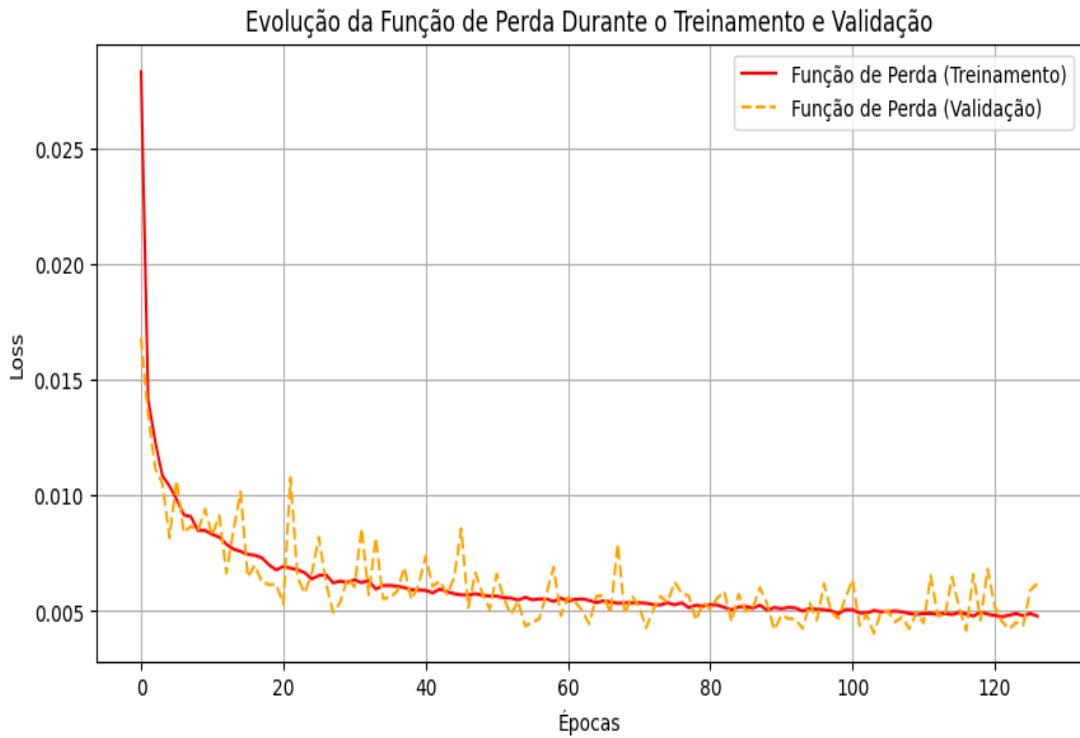


Figura 13 – Gráfico da função de perda ao longo do treinamento usando cem mil imagens.



Ao analisar o gráfico da Figura 4, nota-se que o coeficiente de Dice apresenta um aumento sutil ao longo das épocas. O maior avanço ocorre no início, período em que o modelo está ajustando os pesos dos filtros para reconstruir as imagens. No entanto, após um determinado número de épocas, o coeficiente estabiliza em torno de 0.615. Posteriormente, observa-se uma leve queda, seguida de uma nova estabilização em relação ao conjunto de treino. No que tange ao conjunto de validação, que representa a performance do modelo frente a imagens inéditas, os valores do coeficiente oscilam entre 0.618 e 0.612, sem ultrapassar esses limites. Desse modo, sob a perspectiva do coeficiente de Dice, não se justifica prosseguir além da época 50 ou 55, visto que, a partir desse ponto, não são percebidas melhorias significativas no desempenho do modelo, seja no conjunto de treino ou validação.

Ao observar a Figura 5, que ilustra a função de perda (*loss*), percebe-se que a perda cresce de forma repetitiva em relação ao conjunto de validação até aproximadamente a época 80. A partir daí, a ascensão brusca da perda diminui e, por volta da época 90, atinge seu valor mínimo no gráfico. Combinando essa análise com a Figura 4, onde se identifica que em torno da época 85 o modelo alcança um bom valor tanto para o coeficiente de Dice quanto para a função de perda, conclui-se que 85 épocas é o número máximo recomendado para o treinamento. Treinamentos adicionais após esse ponto podem acarretar em *overfitting*, situação na qual o modelo se especializa excessivamente nos dados de treino, comprometendo sua capacidade de generalização para novos dados.

## 6 DISCUSSÃO

### 6.1 LIMITAÇÕES DO MODELO

Em qualquer projeto de aprendizado de máquina, é crucial reconhecer e entender as limitações do modelo desenvolvido. Embora nosso modelo tenha demonstrado um desempenho notável em diversas situações, existem cenários específicos em que ele enfrenta desafios.

Um dos cenários mais evidentes é a reconstrução de textos. O modelo tem um desempenho admirável quando se trata de rabiscos ou omissões parciais em caracteres. Por exemplo, se partes de uma letra são obscurecidas ou removidas, o modelo pode, com razoável precisão, prever e reconstruir a letra original. No entanto, quando uma letra é completamente apagada ou omitida, o modelo enfrenta dificuldades em inferir sua presença com base no contexto das letras adjacentes. Isso sugere que, enquanto o modelo é sensível a pequenas perturbações, ele pode não ter a capacidade de inferência contextual necessária para preencher lacunas maiores em sequências de texto.

Além disso, quando se trata de objetos mais complexos e uma porção significativa deles é apagada ou obscurecida, o modelo também enfrenta desafios. A capacidade de reconstruir um objeto com base em uma pequena fração de sua representação original é uma tarefa complexa e exige uma compreensão profunda e contextual do objeto em questão.

No entanto, é importante destacar que, em geral, o modelo exibe um desempenho impressionante na reconstrução da maioria das classes disponíveis no ImageNet. Esta ampla gama de classes abrange uma variedade de objetos e cenários, indicando que o modelo é robusto e versátil em muitos aspectos.

A incapacidade do modelo de lidar perfeitamente com todos os cenários pode ser atribuída a várias razões. Primeiramente, a natureza intrínseca do aprendizado de máquina significa que os modelos são tão bons quanto os dados em que foram treinados. Se o conjunto de treinamento não contiver exemplos suficientes de letras ou objetos sendo completamente apagados, o modelo pode não aprender a inferir sua presença adequadamente. Além disso, a arquitetura do modelo e a capacidade de capturar contextos mais amplos podem ser outras áreas potenciais de melhoria.

Em resumo, enquanto nosso modelo representa um avanço significativo na tarefa de reconstrução de imagens, ainda existem áreas que exigem investigação e otimização adicionais.

## 6.2 SUGESTÕES DE MELHORIAS E TRABALHOS FUTUROS

No vasto campo do aprendizado profundo, a evolução contínua das técnicas e abordagens nos oferece uma gama de oportunidades para otimizar e refinar nossos modelos. No contexto deste trabalho, várias áreas de melhoria emergem como promissoras para aprimorar ainda mais a performance do nosso modelo.

A substituição da convolução transposta por uma combinação de *upsampling* seguido de uma convolução pode ser uma abordagem valiosa. Essa mudança tem o potencial de resultar em uma reconstrução de imagem mais suave, minimizando artefatos indesejados que podem surgir com a convolução transposta. Paralelamente, no domínio dos modelos gerativos, técnicas de normalização como LayerNormalization e InstanceNormalization têm se destacado por sua eficácia em comparação com a tradicional BatchNormalization. Essas abordagens consideram características específicas de cada amostra ou feature, promovendo uma qualidade superior nas gerações.

Outra técnica que merece atenção é o uso do *average pooling* em detrimento do *max pooling*. Em diversos cenários, o *average pooling* tem demonstrado ser mais representativo, capturando nuances que podem ser perdidas com o *max pooling*. No que diz respeito à estruturação do código, a adoção de subclassing da classe Layer do TensorFlow pode trazer benefícios significativos. Ao criar módulos personalizados e reutilizáveis, não só tornamos nosso código mais elegante, mas também facilitamos a experimentação e a modularidade.

A metodologia de treinamento também é um campo fértil para otimizações. A busca pela combinação ideal de hiperparâmetros pode ser a chave para desbloquear performances ainda melhores. No entanto, é importante mencionar que, devido a limitações de poder computacional, nem sempre foi possível investir tempo em otimizações intensivas. Esse desafio computacional também impactou nossa implementação de leitura e *augmentation*. Sabemos que paralelizar a leitura e vetorizar operações pode acelerar o processo, mas tais melhorias exigem recursos computacionais adicionais.

Em resumo, nosso modelo, embora promissor, ainda tem espaço para refinamentos. As áreas de melhoria identificadas, aliadas às inovações mais recentes da literatura, podem guiar futuras pesquisas e iterações neste domínio. A constante evolução da capacidade computacional certamente desempenhará um papel crucial nesse progresso.

## 7 CONCLUSÃO

Ao longo deste trabalho, exploramos a capacidade e versatilidade da arquitetura U-Net, tradicionalmente empregada em tarefas de segmentação, para abordar o desafio do *inpainting*. Através de adaptações criteriosas no pré-processamento dos dados, na formação das máscaras e na apresentação das labels, bem como ajustes na saída do modelo, conseguimos moldar a U-Net para atender às especificidades da nossa tarefa.

O *inpainting*, que envolve a reconstrução de áreas ausentes ou corrompidas em imagens, é uma tarefa complexa e desafiadora. No entanto, os resultados obtidos demonstram que, com as modificações apropriadas, a U-Net pode ser uma ferramenta valiosa nesse contexto. As adaptações realizadas não apenas permitiram que o modelo funcionasse eficazmente, mas também destacaram a flexibilidade e adaptabilidade desta arquitetura.

Além disso, o trabalho também lançou luz sobre as limitações do modelo, bem como áreas potenciais de melhoria. Estas observações são cruciais, pois fornecem um caminho claro para futuras pesquisas e otimizações. A capacidade de identificar e abordar essas limitações é um testemunho da natureza iterativa e evolutiva da pesquisa em aprendizado de máquina.

Em suma, este trabalho reforça a ideia de que, com as adaptações corretas, arquiteturas consagradas podem ser repensadas e reutilizadas para tarefas distintas das quais foram originalmente concebidas. A U-Net, em particular, demonstrou ser uma escolha robusta e eficaz para o *inpainting*, abrindo portas para futuras explorações e inovações nesta área.

Ao refletir sobre a jornada deste projeto final, fica evidente que a interseção entre conhecimento teórico, experimentação prática e inovação contínua é o cerne do progresso em ciência de dados e aprendizado de máquina. E é com essa perspectiva que concluímos nosso estudo, esperançosos e animados para as futuras possibilidades e descobertas que essa área tem a oferecer.

## REFERÊNCIAS

- AMIDI, S. **Keras - How to generate data on-the-fly?** 2018. Disponível em: <https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly>.
- BERTALMÍO, M. et al. Image inpainting. **Proceedings of the 27th annual conference on Computer graphics and interactive techniques**, 2000. Disponível em: <https://api.semanticscholar.org/CorpusID:308278>.
- DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2009. p. 248–255.
- DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. **arXiv preprint arXiv:1603.07285**, 2016.
- FILHO, M. **MAE (Erro Médio Absoluto) Em Machine Learning**. 2022. Disponível em: <https://mariofilho.com/mae-erro-medio-absoluto-em-machine-learning/>.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.
- GOODFELLOW, I. J. et al. **Generative Adversarial Networks**. 2014.
- HAYS, J.; EFROS, A. A. Scene completion using millions of photographs. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 26, n. 3, p. 4–es, jul 2007. ISSN 0730-0301. Disponível em: <https://doi.org/10.1145/1276377.1276382>.
- HE, K. et al. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778.
- IIZUKA, S.; SIMO-SERRA, E.; ISHIKAWA, H. Globally and locally consistent image completion. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 36, n. 4, jul 2017. ISSN 0730-0301. Disponível em: <https://doi.org/10.1145/3072959.3073659>.
- KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.
- LONG, J.; SHELHAMER, E.; DARRELL, T. **Fully Convolutional Networks for Semantic Segmentation**. 2015.
- OLIVEIRA, M.; PITA, R. **Usando GAN's para geração automática de rostos humanos**. 2023. Disponível em: <https://github.com/MatheusOliveiraSilva/Intro-Aprendizado-de-Maquina/blob/main/Trabalho%20Final%20-%20ML/relatorio%20final%20-%20introML.pdf>.
- RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. 2015.

SØRENSEN–DICE coefficient. 2022. Disponível em: [https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice\\_coefficient](https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient).

VALDENEGRO-TORO, M.; SABATELLI, M. Machine learning students overfit to overfitting. 2022. Disponível em: <https://arxiv.org/abs/2209.03032>.

WEI, R.; WU, Y. **Image Inpainting via Context Discriminator and U-Net**. 2022. Disponível em: <https://doi.org/10.1155/2022/7328045>.

YU, J. et al. **Generative Image Inpainting with Contextual Attention**. 2018.

YU, J. et al. **Free-Form Image Inpainting with Gated Convolution**. 2019.

## APÊNDICES

## APÊNDICE A – EVOLUÇÃO DO MODELO AO LONGO DAS ÉPOCAS

O presente apêndice tem como propósito apresentar, de forma detalhada, a evolução das imagens processadas pelo modelo ao longo das épocas de treinamento, evitando, assim, sobrecarregar o corpo principal do trabalho.

A metodologia adotada para esta análise consistiu no seguinte procedimento: foi implementada uma função de retorno (callback) que, ao iniciar o treinamento, seleciona 10 imagens de um diretório previamente estabelecido e cria 10 respectivas imagens mascaradas. A preservação destas imagens mascaradas é de suma importância, pois permite uma observação consistente da capacidade evolutiva do modelo em reconstruir a mesma categoria de imagem. Posteriormente, a cada intervalo de 5 épocas, o modelo é submetido a um teste para gerar 10 reconstruções. Estas imagens são armazenadas para possibilitar uma análise contínua da progressão do modelo.

A seguir, será apresentada a evolução do modelo em três momentos distintos do treinamento: após a conclusão da 1<sup>a</sup> época, da 10<sup>a</sup> época e da 50<sup>a</sup> época.

Figura 14 – Evolução da imagem 1

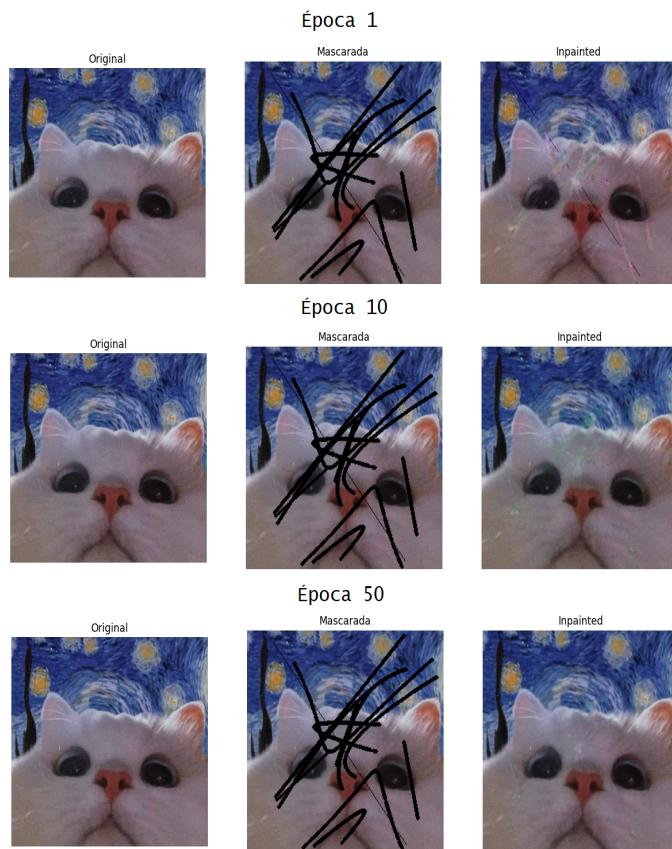


Figura 15 – Evolução da imagem 2



Figura 16 – Evolução da imagem 3

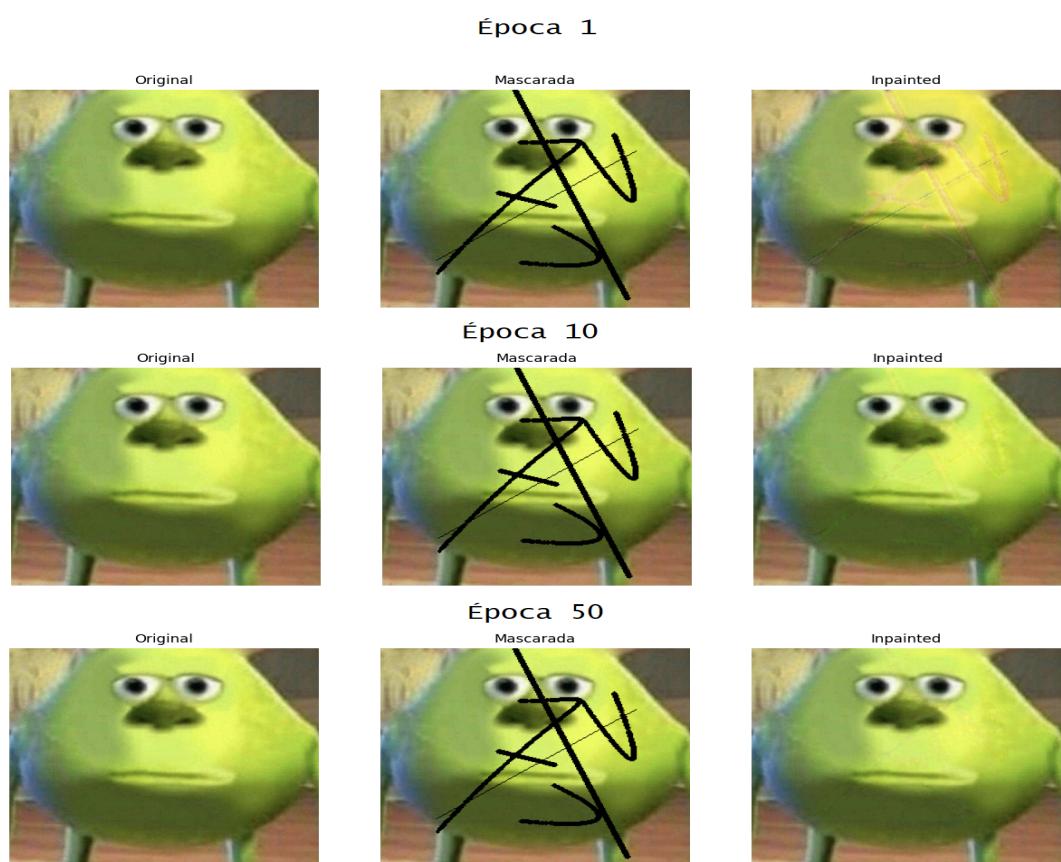


Figura 17 – Evolução da imagem 4

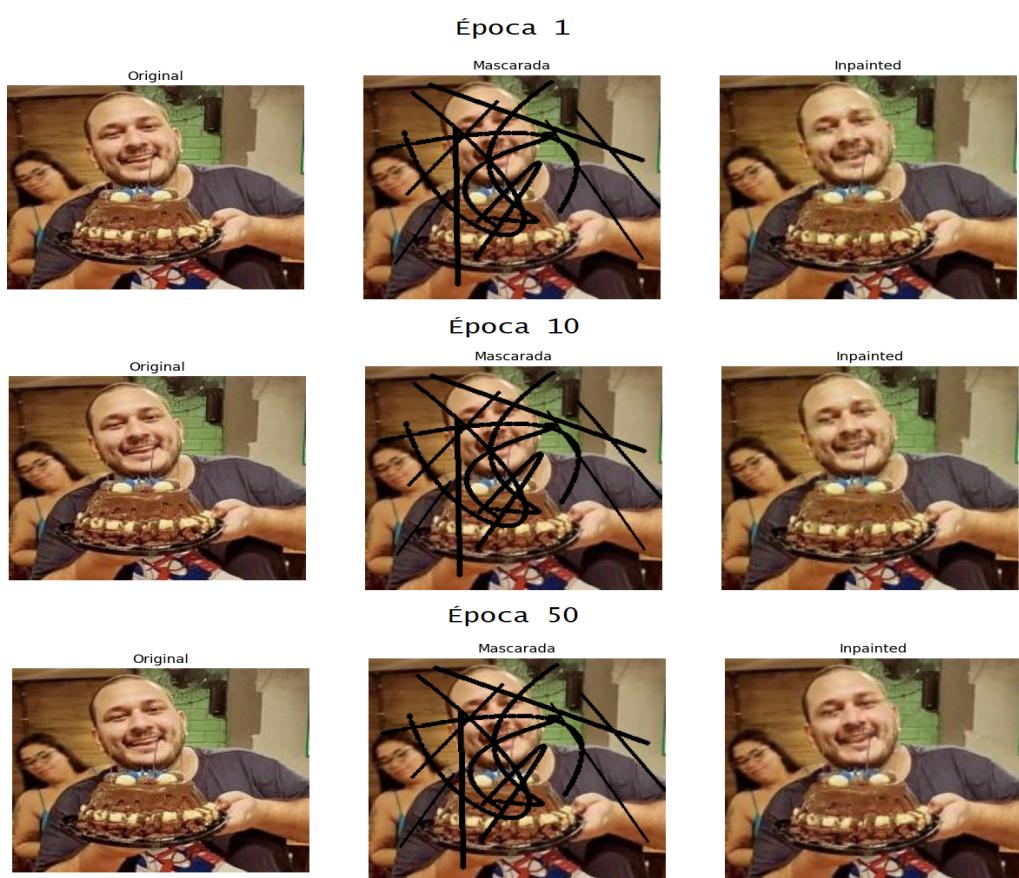


Figura 18 – Evolução da imagem 5

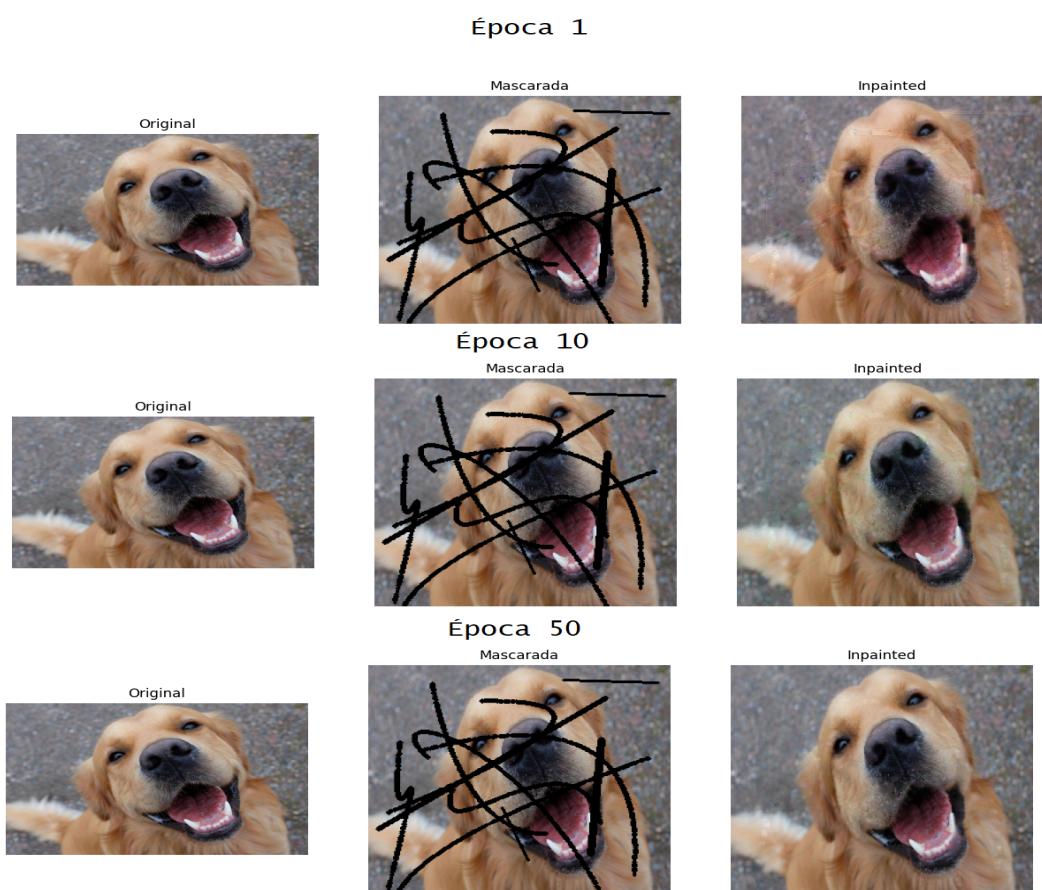


Figura 19 – Evolução da imagem 6

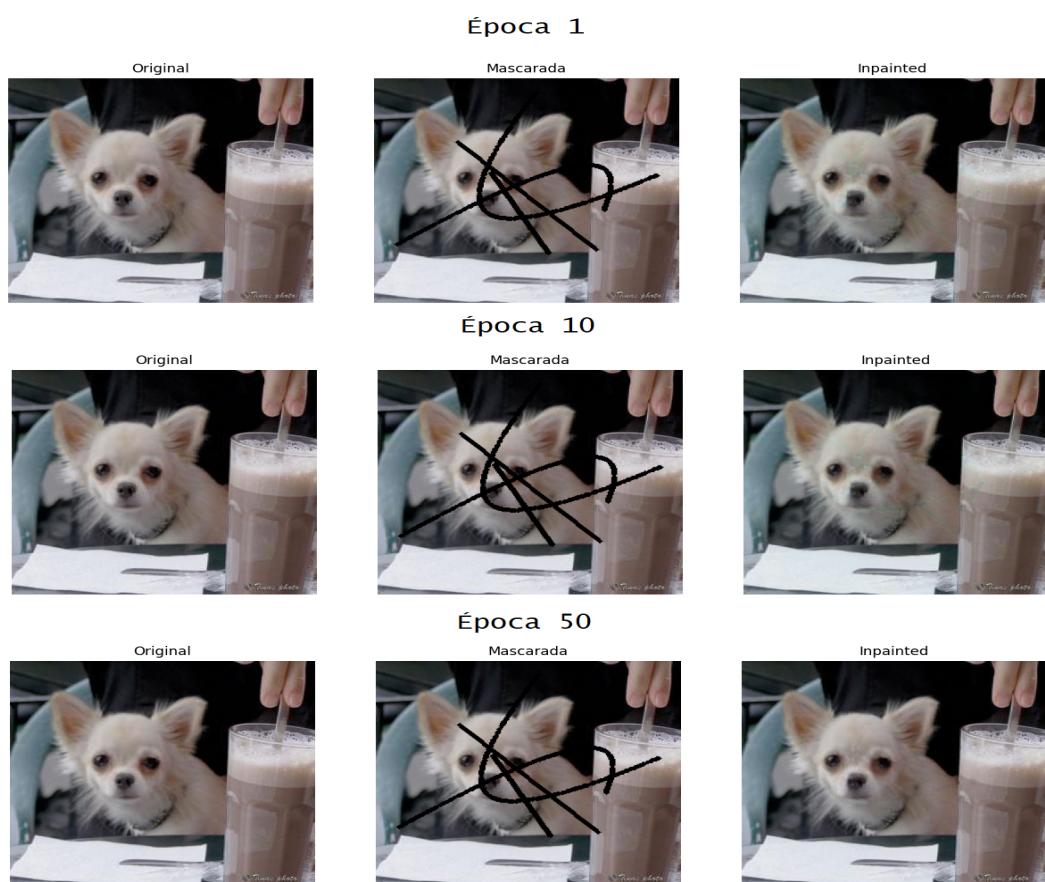


Figura 20 – Evolução da imagem 7



Figura 21 – Evolução da imagem 8

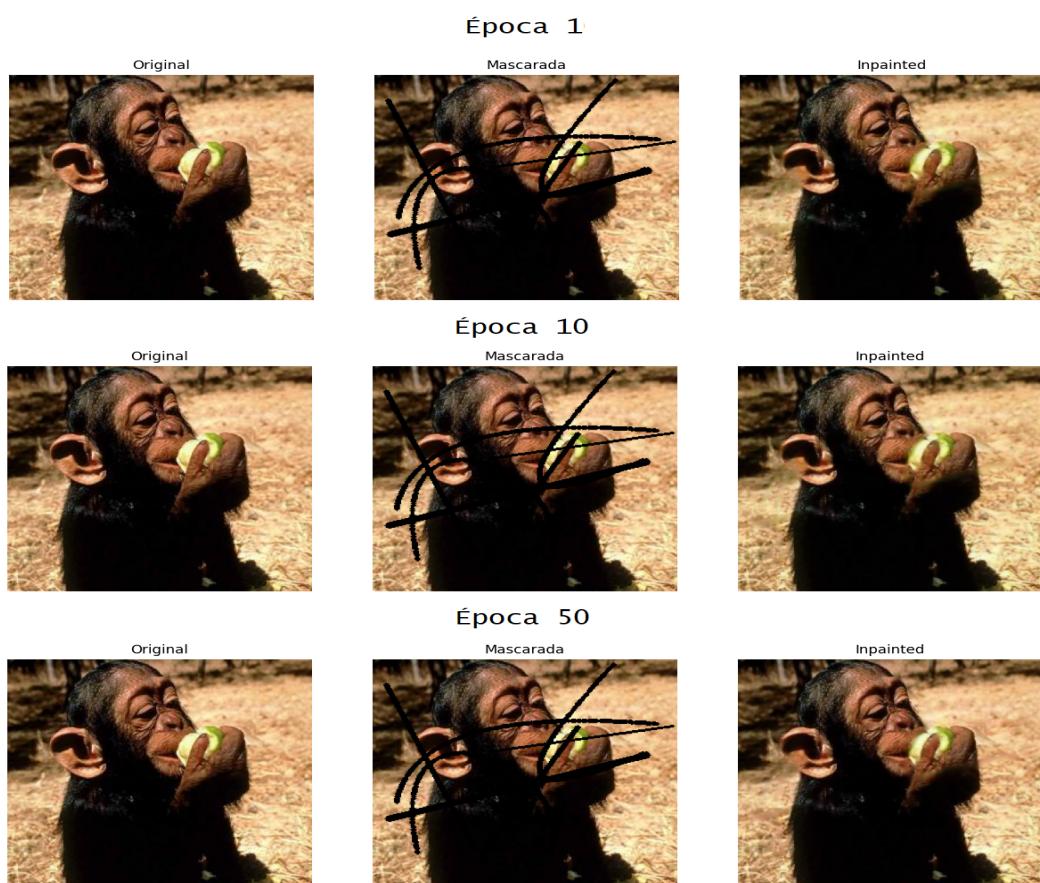


Figura 22 – Evolução da imagem 9

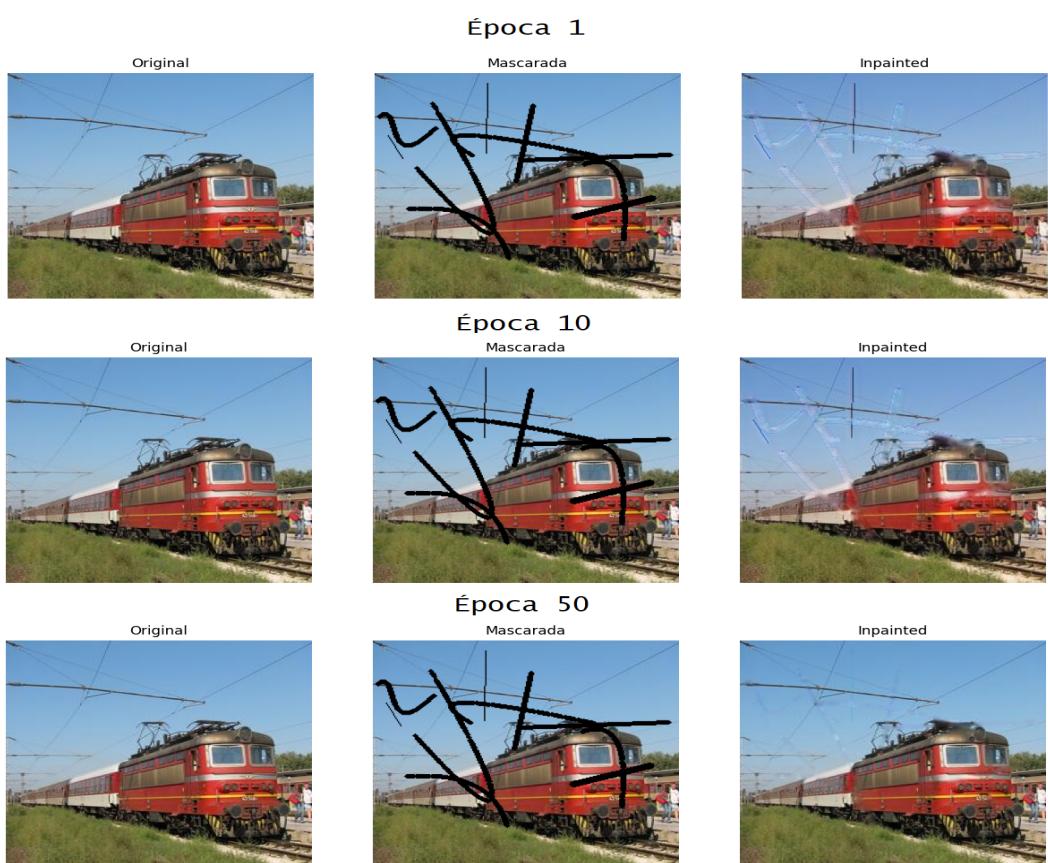


Figura 23 – Evolução da imagem 10

