

UNIVERSIDADE FEDERAL FLUMINENSE

André Fernandes Gonçalves

**Classificação de Alagamento em Vias com Tráfego
na Cidade do Rio de Janeiro através de Visão
Computacional**

NITERÓI

2025

André Fernandes Gonçalves

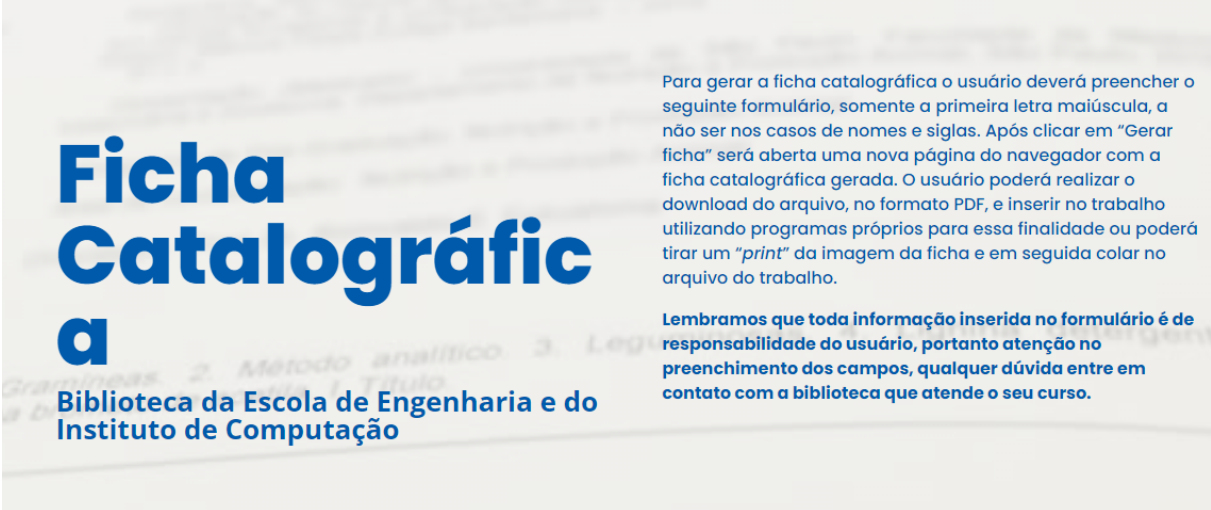
Classificação de Alagamento em Vias com Tráfego na Cidade do Rio de Janeiro através de Visão Computacional

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação

Orientador:
Aura Conci

NITERÓI

2025



Ficha Catalográfica

Biblioteca da Escola de Engenharia e do Instituto de Computação

Para gerar a ficha catalográfica o usuário deverá preencher o seguinte formulário, somente a primeira letra maiúscula, a não ser nos casos de nomes e siglas. Após clicar em “Gerar ficha” será aberta uma nova página do navegador com a ficha catalográfica gerada. O usuário poderá realizar o download do arquivo, no formato PDF, e inserir no trabalho utilizando programas próprios para essa finalidade ou poderá tirar um “*print*” da imagem da ficha e em seguida colar no arquivo do trabalho.

Lembramos que toda informação inserida no formulário é de responsabilidade do usuário, portanto atenção no preenchimento dos campos, qualquer dúvida entre em contato com a biblioteca que atende o seu curso.

Figura 1: Local da ficha catalográfica

ANDRÉ FERNANDES GONÇALVES

CLASSIFICAÇÃO DE ALAGAMENTO EM VIAS COM TRÁFEGO NA CIDADE DO
RIO DE JANEIRO ATRAVÉS DE VISÃO COMPUTACIONAL

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação

Aprovada em <MES> de 2025.

BANCA EXAMINADORA

Prof. Aura Conci - Orientadora, UFF

Prof. Leandro Augusto Frata Fernandes, UFF

Prof. Flávia Cristina Bernardini, UFF

Prof. Aristófanês Corrêa Silva, UFMA

Prof. Raul Queiroz Feitosa, PUC

Niterói

2025

Aos meus pais, que me apoiaram ao longo da minha jornada.

Agradecimentos

A minha orientadora, Aura Conci, que me ajudou em todo o caminho e sempre confiou em mim.

A UFF e a CAPES, pela bolsa que me ajudou durante meus estudos do mestrado.

A Luis Rezende e Otávio Flaeschen pela ajuda na criação do conjunto de dados.

Ao Centro de Operações do Rio pelo acesso as câmeras da cidade do Rio de Janeiro.

Resumo

Este trabalho teve como objetivo desenvolver uma abordagem inicial para a classificação automática de alagamentos da cidade do Rio de Janeiro. Para isso, este trabalho primeiramente criou um conjunto de dados original composto de 4620 imagens separadas em 78% para treino e 22% para validação. Este conjunto de dados foi montado com imagens de câmeras de ruas da cidade do Rio ao longo de diferentes meses, em diferentes horas do dia e em diferentes níveis de alagamento. Após o conjunto de dados ser devidamente analisado e montado de forma equilibrada, cinco arquiteturas de redes neurais foram avaliadas nesse conjunto de dados. Essas arquiteturas foram escolhidas de acordo com trabalhos da literatura com temática relacionada ao problema abordado. Após o treinamento das arquiteturas *VGG-19*, *Inception*, *DenseNet*, *MobileNetV2*, e *Visual Transformer* para o problema de classificação de imagem, o *Visual Transformer* obteve melhor resultado e também foi analisado com diferentes níveis de iluminação e três diferentes otimizadores. Também foi investigado se o aumento na quantidade de imagens de treino traria ganhos compensadores. O desempenho dos resultados foi comparado com outros dois conjuntos de dados. Ao final, o *Visual Transformer* foi melhor modelo com com acurácia de 82,6% no conjunto de dados original. O código usado para treino e avaliação dos modelos, assim como o conjunto de dados original criado estão disponíveis [neste repositório de GitHub](#).

Palavras-chave: Visão Computacional, Rede Neural Convolucional, *Visual Transformer*, Classificação, Alagamento.

Abstract

This work aimed to develop an initial approach for the automatic classification of floods in the city of Rio de Janeiro. To this end, this work first created an original dataset composed of 4620 images separated into 78% for training and 22% for validation. This dataset was assembled with images from street cameras in the city of Rio over different months, at different times of the day and at different flood levels. After the dataset was properly analyzed and assembled in a balanced way, five neural network architectures were evaluated on this dataset. These architectures were chosen according to works in the literature with a theme related to the problem addressed. After training the *VGG-19*, *Inception*, *DenseNet*, *MobileNetV2*, and *Visual Transformer* architectures for the image classification problem, the *Visual Transformer* obtained the best result and was also analyzed with different lighting levels and three different optimizers. It was also investigated whether increasing the number of training images would bring compensating gains. The performance of the results was compared with two other datasets. In the end, the *Visual Transformer* was the best model with an accuracy of 82.6% on the original dataset. The code used for training and evaluating the models, as well as the original dataset created are available [in this GitHub repository](#).

Keywords: Computer Vision, Convolutional Neural Network, *Visual Transformer*, Classification, Flooding

Lista de Figuras

1	Local da ficha catalográfica	
2	Sala central do Centro de Operações do Rio	13
3	Arquitetura de uma <i>CNN</i> genérica.	18
4	Arquitetura de uma <i>DenseNet</i> genérica[36].	21
5	Arquitetura de um <i>Visual Transformer</i> [45].	23
6	Via completamente seca, rotulada como classe 0	33
7	Rua molhada, ainda rotulada como classe 0	33
8	Rua parcialmente alagada (classe 1).	34
9	Avenida totalmente alagada (classe 1).	34
10	Quantidade de imagens de normalidade por câmera disponível.	35
11	Quantidade de imagens de alagamento por câmera disponível.	36
12	Histograma de quantidade de imagens de alagamento por câmera disponível.	37
13	Quantidade de câmeras com um número mínimo de imagens de cada classe.	38
14	Exemplo de região sem alteração no canal Y	39
15	Exemplo de mesma região com 50% do valor original de Y	40
16	Acurácia do <i>ViT</i> durante treino e validação.	45
17	Perda do <i>ViT</i> durante treino e validação.	46
18	Acurácia do <i>ViT</i> com diferentes otimizadores.	47
19	Acurácia do <i>ViT</i> com NAdam.	48
20	Perda do <i>ViT</i> com NAdam.	49
21	Acurácia do <i>ViT</i> com NAdam e menor taxa de aprendizagem.	49
22	Perda do <i>ViT</i> com NAdam e menor taxa de aprendizagem.	50

Lista de Tabelas

1	Resumo dos trabalhos relacionados	30
2	Acurácia dos cinco modelos com diferentes taxas de aprendizagem.	43
3	Redução de áreas de brancos e desempenho do <i>ViT</i>	45
4	Acurácia \times tempo de treino \times número de imagens usadas	46
5	Acurácia do <i>ViT</i> com diferentes otimizadores.	47
6	Métricas do <i>ViT</i> de dia e à noite.	50
7	Performance do <i>ViT</i> em outros conjuntos de dados	51

Sumário

1	Introdução	12
1.1	Objetivos	12
1.2	Organização do Texto	13
2	Fundamentação Teórica	15
2.1	Técnicas de Processamento de Imagens	15
2.1.1	Espaço de Cores	15
2.1.2	Conversões entre RGB e YCbCr	16
2.1.3	Transformações de Intensidade no Domínio Espacial	16
2.1.4	Filtragem por Convolução	17
2.2	Redes Neurais Convolucionais	17
2.2.1	Arquitetura de uma Rede Neural Convolucional	17
2.2.1.1	Camada Convolucional	18
2.2.1.2	Camada de Pooling	19
2.2.1.3	Camada Totalmente Conectada	19
2.2.2	Processo de Treinamento de uma CNN	19
2.2.3	Aprendizado por Transferência	20
2.2.4	Visual Geometry Group (VGG)	20
2.2.5	Inception	20
2.2.6	DenseNet	21
2.2.7	MobileNet	21
2.2.8	Visual Transformer (ViT)	22

3	Trabalhos Relacionados	24
3.1	Aprendizado de Máquina	24
3.2	Imagens de Satélite	25
3.2.1	U-Net	26
3.3	Alagamento em áreas urbanas	26
4	Materiais e Métodos	31
4.1	Conjunto de dados	31
4.1.1	Captação de imagens	32
4.1.2	Análise de imagens	32
4.1.3	Pré Processamento	37
4.2	Modelos de classificação	39
4.3	Avaliação dos modelos	40
5	Experimentos e Resultados	43
5.1	Resultados dos Modelos	43
5.2	Avaliação do melhor modelo	44
5.2.1	Diminuição da intensidade luminosa	44
5.2.2	Curvas de acurácia e perda do melhor modelo	45
5.2.3	Melhor modelo com diferentes quantidades de câmeras	46
5.3	Comparação entre diferentes otimizadores	47
5.4	Comparações entre cena diurna e noturna	49
5.5	Outros conjuntos de dados	50
6	Conclusão	52
6.1	Trabalhos futuros	53
	REFERÊNCIAS	54

1 Introdução

As inundações em áreas urbanas apresentam riscos significativos à infraestrutura, saúde pública e meio ambiente. A água inundada pode levar a muitos problemas, incluindo a interrupção de transportes e serviços essenciais (energia elétrica, internet, etc.) . Para os órgãos governamentais responsáveis pelo monitoramento de áreas urbanas, como o Centro de Operações do Rio (COR), a observação constante dessas áreas urbanas é crucial para a emissão de alertas precoces e respostas eficazes. A detecção e avaliação oportuna dos riscos de inundação permitem que as autoridades implementem planos de evacuação, desloquem serviços de emergência e iniciem medidas de controle de inundação para minimizar danos e proteger vidas.

O monitoramento pelo COR é realizado na própria sala de controle, onde dezenas de funcionários monitoram uma quantidade ainda maior de telas que exibem as milhares de câmeras da cidade do Rio de Janeiro, como pode ser visto na Figura 2, tornando isto um trabalho árduo.

1.1 Objetivos

O monitoramento atualmente realizado pelo COR não é automatizado. Este monitoramento não só é dependente da mão-de-obra dos funcionários frente a diversas telas, como também depende em parte de notificações enviadas por terceiros, utilizando o aplicativo do COR para reportar problemas na cidade de forma direta.

Com base em diversas pesquisas sobre o problema das inundações urbanas utilizando diferentes Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Network*), apresentadas na Seção 3.3, a presente dissertação visa desenvolver uma abordagem inicial utilizando *CNNs* com aprendizado por transferência (*transfer learning*), treinadas em um conjunto de dados original, para não apenas facilitar o monitoramento das milhares de câmeras na cidade do Rio de Janeiro pelo COR, mas também notificar automaticamente quando uma rua estiver em uma situação que necessite de ação do COR.



Figura 2: Sala central do Centro de Operações do Rio

Para automatizar o monitoramento da situação de alagamento da cidade do Rio de Janeiro, cinco arquiteturas de redes neurais foram treinadas e avaliadas. Todos os modelos foram treinados em conjunto de dados original formado por imagens do circuito de câmeras do Centro de Operações do Rio. Todas as câmeras neste conjunto de dados são fixas, de forma que não há nenhuma modificação na paisagem ao longo do dia.

1.2 Organização do Texto

O capítulo 2 explica brevemente os conceitos mais importantes para o desenvolvimento deste trabalho, começando por técnicas de processamento de imagem e finalizando na abordagem dos fundamentos de redes neurais.

O capítulo 3 considera alguns trabalhos relevantes para o problema de classificação do estado de alagamento em regiões urbanas nos últimos 5 anos. Ao final, algumas características das tecnologias utilizadas em cada artigo, assim como as deste trabalho, são comparadas em uma tabela.

O capítulo 4 descreve a metodologia usada na execução deste trabalho. Explicando a criação do conjunto de dados, a escolha das arquiteturas de *CNNs* inicialmente empregadas e as métricas utilizadas para a comparação dos resultados.

O capítulo 5 apresenta e discute os resultados do treinamento das diferentes arquiteturas comparadas. O modelo com melhor performance é mais detalhadamente avaliado

após a inclusão de alterações no pré-processamento e uso de diferentes conjuntos de dados de entrada.

O capítulo 6 sumariza as contribuições e limitações deste trabalho, apresentando conclusões relacionadas aos seus resultados. Também Discute possíveis pontos para melhoria futuras em prosseguimentos da linha de pesquisa .

2 Fundamentação Teórica

Este capítulo explica de forma breve os conceitos necessários para entender todas as ferramentas usadas no capítulo 4. Para isso, o capítulo é dividido em duas sessões, onde a primeira se baseia em conceitos de processamento de imagem. Já a segunda sessão é dedicada exclusivamente para redes neurais convolucionais e as arquiteturas utilizadas.

2.1 Técnicas de Processamento de Imagens

Este trabalho foi desenvolvido em Python, utilizando a biblioteca *Pillow* para manipular imagens. As equações que definem as conversões entre espaços são descritas de acordo com a documentação do *Pillow*[16].

2.1.1 Espaço de Cores

Uma imagem colorida é composta por uma ou mais bandas. Usualmente, uma imagem digital em cores é composta por três bandas monocromáticas, onde cada banda armazena uma informação diferente da imagem colorida no pixel $p(m,n)$ da banda.

Cada banda (ou canal), de largura M pixels e altura N pixels, pode ser considerada como um plano onde um pixel está relacionado a um par de coordenadas inteiras $p(m,n)$ pertencente à banda. O valor armazenado na posição m,n no plano indica a intensidade da banda na imagem naquele ponto, usualmente com 256 possíveis valores diferentes de serem armazenados em um byte. Onde 0 é a intensidade mais escura e 255 a intensidade mais clara [49, 72].

Existem espaços diferentes de representação de cores, sendo o mais comum o RGB CIE (sigla para *Commission internationale de l'éclairage*) de 1931, ou simplesmente RGB (do inglês *Red, Green and Blue*). Neste espaço cada cor aparece em seu componente espectral primário correspondente a essas cores, de modo que cada banda armazena unicamente a informação da intensidade do vermelho, verde e azul respectivamente [29, 49]. O espaço

de cores XYZ-CIE 1931 (XYZ) foi projetado para representar todas as cores visíveis, mesmo as que não são representáveis no RGB. As componentes X e Z são elementos sem visibilidade aos olhos humanos, portanto, não existem na práticas como cores [57]. O componente Y representa a luminância ou a melhor representação monocromática possível da cena originalmente colorida.

2.1.2 Conversões entre RGB e YCbCr

Nestas conversões não há nenhum tipo de alteração nos valores dos pixels no espaço de cor original antes de convertê-los para pixels no espaço de cor desejado. As equações 2.1, 2.2 e 2.3 mostram a conversão do espaço RGB para YCbCr. Já as equações 2.4, 2.5 e 2.6 mostram a conversão do espaço YCbCr para RGB.

$$Y \leftarrow R \cdot 0,29900 + G \cdot 0,58700 + B \cdot 0,11400 \quad (2.1)$$

$$Cb \leftarrow R \cdot -0,16874 + G \cdot -0,33126 + B \cdot 0,50000 + 128 \quad (2.2)$$

$$Cr \leftarrow R \cdot 0,50000 + G \cdot -0,41869 + B \cdot -0,08131 + 128 \quad (2.3)$$

$$R \leftarrow Y + (Cr - 128) \cdot 1,40200 \quad (2.4)$$

$$G \leftarrow Y + (Cb - 128) \cdot -0,34414 + (Cr - 128) \cdot -0,71414 \quad (2.5)$$

$$B \leftarrow Y + (Cb - 128) \cdot 1,77200 \quad (2.6)$$

2.1.3 Transformações de Intensidade no Domínio Espacial

As transformações de intensidade são funções matemáticas no domínio espacial da imagem, que operam diretamente no valor dos pixels da imagem de entrada [29]. Essas funções podem ser representadas pela expressão:

$$g(x,y) = T[f(x,y)] \quad (2.7)$$

onde $f(x,y)$ é a imagem de entrada, $g(x,y)$ é a imagem de saída e T é uma função definida sobre um ponto x,y ou sua vizinhança.

Uma das transformações de intensidade mais simples é a transformação linear, onde os valores de intensidade de um canal são multiplicados por uma constante, como repre-

sentado pela equação:

$$i' = \alpha \times i \quad (2.8)$$

Através da equação 2.8, a intensidade i de um pixel é multiplicada pelo fator de escala α , resultando na nova intensidade i' .

2.1.4 Filtragem por Convolução

Também chamados de máscaras ou kernels, filtros são matrizes de tamanho $m \times n$ aplicadas sobre uma imagem através de multiplicações pelos valores das máscaras e somas, cujos resultados serão os valores da imagem processada. Suas dimensões e valores que a compõem variam dependendo da característica a ser extraída de cada vizinhança $m \times n$ da imagem, como detecção de bordas ou texturas [72].

2.2 Redes Neurais Convolucionais

As redes neurais convolucionais, ou *CNNs*, são uma classe especial de redes neurais artificiais que se especializam em processar dados com uma topologia em matrizes [30].

Inspiradas pela organização do córtex visual dos mamíferos [37], as *CNNs* são projetadas para extrair características locais que dependem de uma pequena vizinhança na imagem, onde estas pequenas características podem ser usadas por outras camadas para detectar características de maior ordem e extrair informações sobre a imagem [7].

CNNs são usadas principalmente para tarefas que trabalham em imagens, como reconhecimento de objetos, segmentação de imagens e detecção de padrões complexos de dados visuais.

O aprendizado de características locais permite que a *CNN* reconheça esses padrões em qualquer outro local da imagem, tornando-a invariante a translação. A utilização de características de uma camada por camadas sucessivas permite que a *CNN* aprenda padrões espaciais hierárquicos [15].

2.2.1 Arquitetura de uma Rede Neural Convolutacional

A arquitetura básica de uma *CNN* inclui três tipos principais de camadas: convolutacional, de pooling e totalmente conectada. Uma *CNN* típica é formada por pares de camadas de

convolução e de pooling empilhadas, que ao final são ligadas a uma camada totalmente conectada. A Figura 3 mostra uma arquitetura genérica de *CNN*.

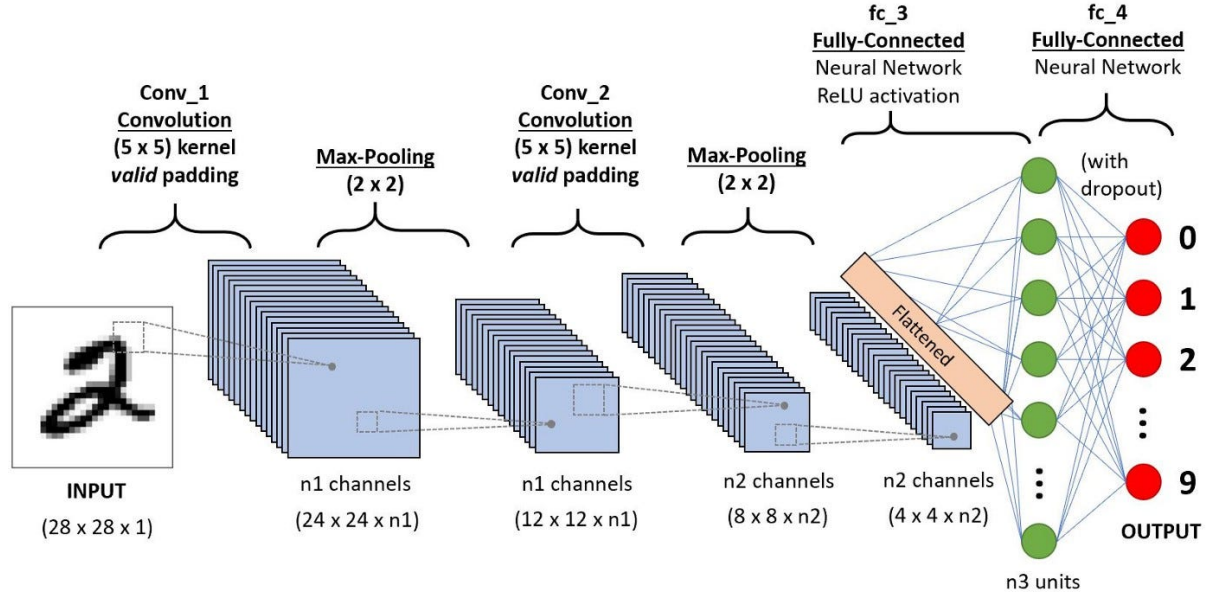


Figura 3: Arquitetura de uma *CNN* genérica.

2.2.1.1 Camada Convolutacional

A camada convolutacional é o núcleo de uma *CNN* e é responsável pela extração de características. Para isso pesos na forma de kernels, ou filtros da Análise de Imagem tradicional, são aplicados sobre a imagem, onde são escorregados e multiplicados pelos valores dos pixels da imagem de entrada e somados resultando em um pixel para cada kernel, usando uma quantidade definida de passos. Esse processo gera os mapas de características, onde cada mapa representa a resposta de um filtro específico a uma região da imagem de entrada. Essa camada detecta padrões locais como bordas, texturas ou formas, (de acordo com o kernel usado) que podem ser usados para extrair informações de padrões mais complexos conforme a rede se aprofunda.

A operação de convolução S entre um filtro I e uma entrada K pode ser representada como:

$$S(x,y) = \sum_m \sum_n I(x+m,y+n)K(m,n) \quad (2.9)$$

onde $S(x,y)$ é o mapa de características gerado (saida), $I(x,y)$ representa a entrada local (imagem original) e $K(m,n)$ o kernel aplicado.

2.2.1.2 Camada de Pooling

A camada de pooling altera a saída da camada convolucional com um resumo estatístico de uma vizinhança, melhorando a eficiência computacional da rede pela diminuição da dimensionalidade da saída e ajudando a tornar a saída menos variante a translações da entrada. Este resumo estatístico pode ser feito com diferentes operações. Por exemplo a arquitetura VGG19 [74] utiliza *max pooling*, onde a saída é o maior valor na vizinhança analisada. A operação *average pooling* é utilizada no InceptionV3 [77], onde a média da vizinhança é calculada.

2.2.1.3 Camada Totalmente Conectada

Estas são geralmente as últimas camadas em uma *CNN*, sendo responsáveis pela combinação das características extraídas nas camadas anteriores para realizar a classificação final. Nesta camada, todos os neurônios estão conectados a cada unidade da camada anterior, como um perceptron multicamadas comum [55]. O seu aprendizado é feito por um algoritmo de retro-propagação de erro [66].

2.2.2 Processo de Treinamento de uma CNN

O treinamento de uma *CNN* é realizado por meio de aprendizado supervisionado, onde o modelo ajusta os pesos e vieses das camadas convolucionais (totalmente conectadas) para minimizar a diferença entre as previsões e os rótulos verdadeiros.

A função de perda, que calcula o erro da rede, é derivada e propagada para ajustar os pesos de forma iterativa. Este trabalho utilizou a Entropia Cruzada como função de perda, que é definida por:

$$H(P,Q) = -E_P[\log Q] \quad (2.10)$$

onde $H(P,Q)$ é a entropia cruzada da distribuição Q em relação a distribuição P , sendo a distribuição Q o resultado da *CNN* e a distribuição P a distribuição de treinamento [51]. $E_P[.]$ é o valor esperado em relação a distribuição P .

Para minimizar a função de perda, foi usada a descida de gradiente estocástica (*SGD*, do inglês *Stochastic Gradient Descent*) [65]. Enquanto a descida de gradiente padrão calcula o gradiente utilizando todo o conjunto de dados, o *SGD* utiliza amostras de dados para o cálculo do gradiente na iteração atual. Dessa forma, a *SGD* altera o valor dos

parâmetros da seguinte forma:

$$\omega^{\tau+1} = \omega\tau - \eta\Delta E \quad (2.11)$$

onde ω é o peso sendo usado, τ é o número da iteração, η é a taxa de aprendizado e ΔE é o gradiente do erro.

2.2.3 Aprendizado por Transferência

Uma possibilidade importante nas *CNNs* é o uso de redes pré-treinadas e aprendizado por transferência (*transfer learning*), onde os pesos dos modelos pré-treinados em grandes conjuntos de dados são aproveitados e ajustados para problemas específicos com menos dados, reduzindo o tempo de treinamento e melhorando a precisão em aplicações com conjuntos de dados menores. Neste trabalho, o aprendizado por transferência baseado no IMAGENET [21] foi aplicado em todos os modelos avaliados, alterando sua última camada para uma classificação binária.

2.2.4 Visual Geometry Group (VGG)

A *VGG* é uma família de *CNNs* que possuem arquiteturas simples. Propostas por [74], as *VGGs* são compostas por blocos uniformes empilhados com duas camadas convolucionais e uma camada de *pooling*. As camadas de *pooling* têm tamanho 2x2 reduzindo a dimensionalidade da convolução pela metade.

Já as camadas convolucionais compartilham do mesmo tamanho de filtro, sendo estes 3x3. Além disso, cada bloco dobra a quantidade de filtros de suas camadas convolucionais em relação ao bloco anterior. Por exemplo, o primeiro bloco de uma rede *VGG* possui 64 filtros 3x3 em suas camadas convolucionais, enquanto o segundo bloco possui 128 filtros 3x3.

Este trabalho utilizou a *VGG-19*, que indica a existência de 19 camadas de aprendizado, contando não só as camadas convolucionais como as camadas completamente conectadas.

2.2.5 Inception

Diferente do *VGG*, a família *Inception*[76] de *CNNs* utiliza o empilhamento de módulos Inception, onde convoluções de diferentes tamanhos são realizadas no mesmo módulo, especificamente convoluções de 1x1, 3x3 e 5x5. Ao final, tais convoluções são concatenadas

junto ao *pooling* da própria entrada, gerando a saída do módulo.

O *InceptionV3*[77] fatoriza convoluções maiores em uma sequência de convoluções menores. A convolução 5x5, por exemplo, é substituída por duas convoluções 3x3, diminuindo a quantidade de parâmetros mas mantendo a expressividade das características extraídas.

Outra modificação implementada no *InceptionV3* foi a utilização do *label smoothing*, onde a confiança do modelo é reduzida para evitar *overfitting*. Isso é feito alterando a probabilidade esperada de cada classe:

$$q(k) = (1 - \epsilon)\delta_{k,j} + \frac{\epsilon}{K} \quad (2.12)$$

Onde j é a classe correta, k é a classe sendo avaliada, $\delta_{k,j}$ é o *Delta de Dirac*, que se torna 1 quando $k = j$ e 0 em qualquer outro caso, ϵ é uma probabilidade de incerteza do modelo, e K é o número de classes. Portanto, o *label smoothing* retira da probabilidade 1 da classe correta uma incerteza ϵ e a distribui igualmente entre as outras K classes do problema.

2.2.6 DenseNet

A característica de maior importância da *DenseNet*[36] é que sua arquitetura é baseada em blocos densos, que são compostos por camadas convolucionais onde sua entrada recebe a saída de todas as camadas convolucionais anteriores a ela. Os blocos densos são ligados por camadas de transição, compostas por uma camada de convolução 1x1 e uma camada de *average pooling* 2x2. A arquitetura de uma *DenseNet* simples pode ser vista na Figura 4 [36].

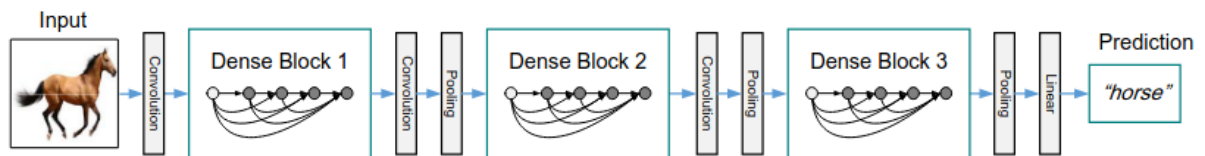


Figura 4: Arquitetura de uma *DenseNet* genérica[36].

2.2.7 MobileNet

A família *MobileNet* de redes neurais foi desenvolvida para utilização por aplicações móveis ou embarcadas [34], possuindo arquiteturas mais simples e leves em comparação a

outras redes neurais profundas, mas continuam a oferecer boa performance. Sua maior distinção de outras *CNNs* é a introdução de convoluções separáveis em profundidade, que substituem as convoluções tradicionais.

As convoluções separáveis em profundidades fatorizam uma convolução comum em uma convolução em profundidade e uma convolução 1×1 , também chamada de convolução pontual. Isso significa que na convolução em profundidade, em vez de aplicar um filtro que abrange todos os canais da janela a qual está realizando a convolução, será aplicada um filtro por canal, onde a saída de cada canal é combinada pela convolução pontual, reduzindo o custo computacional envolvido na convolução.

Além disso, a arquitetura também possui dois hiperparâmetros globais, um multiplicador de largura α e um multiplicador de resolução ρ , que servem para reduzir o custo computacional da rede. O hiperparâmetro α serve para afinar a rede neural em cada camada por um fator α , aplicado tanto no número de canais de entrada quanto de canais de saída. Já o hiperparâmetro ρ é aplicado na imagem de entrada e nas representações internas de cada camada, multiplicando-os por ρ .

A arquitetura *MobileNetV2*[67] introduz blocos inversos residuais na arquitetura da rede. Este bloco aumenta a dimensionalidade da entrada através de um conjunto de convoluções 1×1 e aplica uma convolução em profundidade em seguida, para capturar informações em cada canal expandido. Ao final, uma convolução 1×1 é aplicada para reduzir os canal de volta ao tamanho original.

2.2.8 Visual Transformer (ViT)

A arquitetura *Visual Transformer*[45] aplica o conceito de *transformers*[81] utilizados em processamento de linguagem natural para tarefas de classificação de imagens, como alternativa às *CNNs*. A arquitetura em si é relativamente simples, composta um *transformer encoder*, sem a utilização de um *transformer decoder*, que fornece sua saída a um perceptron multicamadas simples (*MLP*, do inglês *Multilayer Perceptron*), sendo este o responsável pela classificação. Apesar de simples, nos processos necessários para enviar a imagem para o *transformer* é onde a complexidade da arquitetura aparece.

A Figura 5 [45], mostra a arquitetura de um *ViT*. Diferente de uma *CNN* comum, a imagem é dividida em blocos de tamanho fixo denominados *patches*. Esses *patches* são transformados em vetores e linearizados. Para convertê-los ao espaço de características do modelo, os *patches* passam por um *embedding*, através de uma projeção em um espaço

de dimensão fixa. Antes de enviar a sequência de *embeddings* ao *transformer*, é necessário adicionar um *token* de posicionamento para informar à rede sobre a posição de cada *patch* na imagem original. Além desses *tokens*, um *token* adicional *class* é colocado no início da sequência, que receberá a saída do processamento realizado pelo *transformer*.

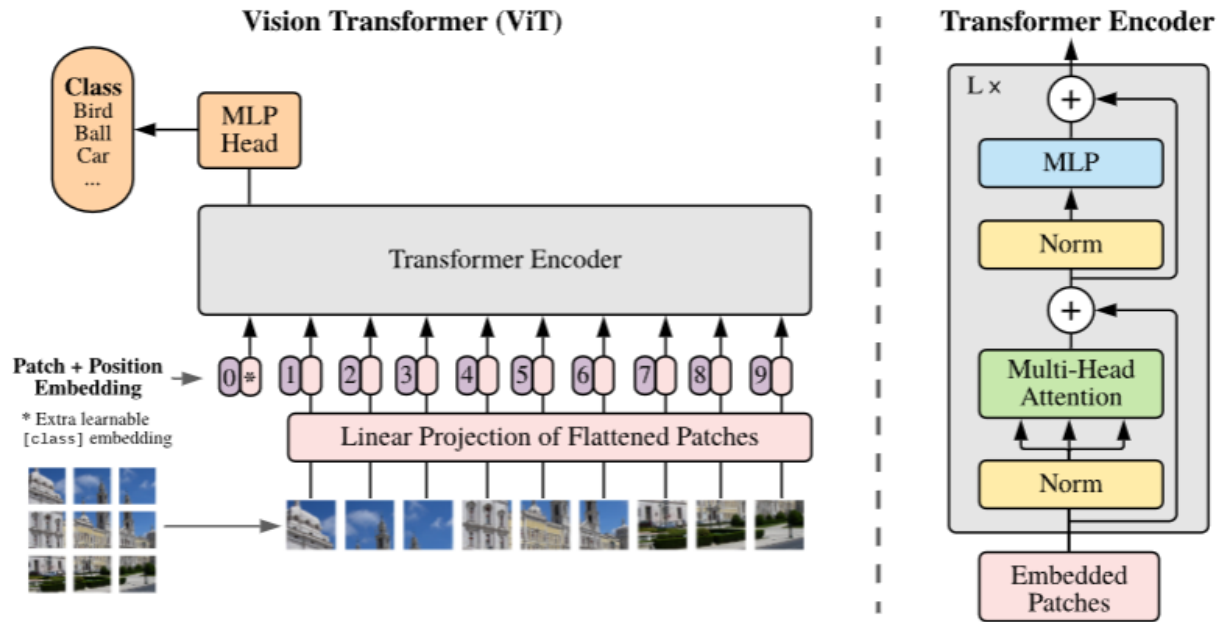


Figura 5: Arquitetura de um *Visual Transformer*[45].

O *Transformer* é uma sequência de pares de camadas alternadas, o tamanho dessa sequência de camadas define a profundidade do modelo. O primeiro par de camadas consiste em uma camada de normalização, transformando os valores de todos os elementos da sequência para limites predefinidos, e uma camada de *Multi-Headed Self-Attention*, onde é computada uma soma ponderada de entre cada par de elemento do *patch*. O segundo par de camadas possui uma camada de normalização e uma camada de *MLP*, para processar cada elemento do *patch*.

O processamento final do *transformer* é enviado para um *MLP* final denominado *MLP Head*, responsável por receber o *token class* que representa as informações da imagem aprendidas no *transformer*, e utilizá-lo para gerar a classificação final da imagem.

3 Trabalhos Relacionados

Diversos estudos foram realizados sobre classificação ou detecção de inundações, seja realizando a tarefa com algoritmos de aprendizado de máquina ou redes neurais e utilizando diferentes fontes de dados: imagens de satélites, imagens de câmeras ou até mesmo valores numéricos retirados de ferramentas de medição, como um pluviômetro por exemplo.

Neste capítulo, primeiramente serão apresentados trabalhos que utilizaram algoritmos de aprendizado de máquina para realizar previsões sobre alagamentos. Em seguida, trabalhos voltados ao uso de imagens de satélite sobre áreas alagadas serão brevemente descritos, com uma pequena seção sobre utilização de *U-Net* para segmentação e classificação dessas imagens. A última seção é reservada para trabalhos onde a classificação de imagens de alagamentos urbanos desempenha papel principal. Portanto, devido ao foco desta pesquisa ser a classificação dessas imagens, principalmente voltado a sistema de câmeras urbanas, a última seção possui seus trabalhos discutidos com maior detalhe em comparação às seções anteriores. Todos os estudos em cada seção são apresentados em ordem cronológica, começando dos mais antigos.

3.1 Aprendizado de Máquina

Utilizando dados do Departamento Meteorológico de Bangladesh (*Bangladesh Meteorological Department*) e do Conselho de Desenvolvimento Hídrico de Bangladesh (*Bangladesh Water Development Board*), Toufique *et al.* [79] treinaram quatro algoritmos de aprendizado de máquina para realizar a predição de inundações em Bangladesh, onde o Random Forest [9] foi o melhor, com acurácia de 85,6%.

Jamunadevi *et al.* [41] propuseram uma abordagem utilizando algoritmo de memória de curto longo prazo [33] e *CNN* para prever alagamentos em um período de um ano no estado de Tamil Nadu, na Índia, com dados de temperatura, humidade, precipitação, velocidade do vento, luz solar e evapotranspiração, obtendo acurácia de 97,4% em relação ao alagamentos de 2024.

Vimala *et al.* [82] utilizaram 22 indicadores para realizar predição de alagamento com 6 algoritmos de aprendizado supervisionado, incluindo o *Random Forest*. Nos resultados apresentados, a Regressão Logística [20] foi o melhor modelo, com acurácia de 99,85%, enquanto o *Random Forest* obteve acurácia de 89,37%.

Shah *et al.* [71] usaram *Random Forest* e *XGBoost* [14] para criar um mapa de suscetibilidade a inundações da cidade de Chennai, no estado de Tamil Nadu. Os autores usaram 6 métricas de três fontes diferentes para treinar os algoritmos, onde o *XGBoost* a melhor acurácia de 70%.

3.2 Imagens de Satélite

Kool *et al.* [47] classificaram o estado de inundação sazonal de um pântano em Mara na Tanzânia. Para isso, treinaram um *Random Forest* sazonalmente em conjuntos de imagens do satélite Sentinel-2 [18], obtendo acurácia de 98,6%.

Siddique *et al.* [73] propuseram uma abordagem integrada utilizando *Random Forest*, *k-Nearest Neighbours (k-NN)* [19] e *k-means* [38] para classificar imagens do satélite Sentinel-1 [17] do estado da Índia de Uttar Pradesh. Obtendo acurácia de 97,0% para a cidade de Basti e 93,8% para a cidade de Ayodhya, ambas no estado de Uttar Pradesh.

Reddy e Vimal [62] classificaram 400 imagens de satélites, obtendo resultados de 92,72% de acurácia com análise discriminante linear [78] e 95,05% com AlexNet [48].

Cao *et al.* [11] utilizaram uma *CNN* simples, proposta pelos autores, onde a saída da camada completamente conectada alimenta um classificador de Máquina de Vetores de Suporte (*SVM*, do inglês *Support Vector Machine*) [26] para classificar o pântano de Qilihai, na China. Obtiveram acurácia de 0,903 no conjunto de dados, que foi montado através de imagens do Sentinel-1 e Sentinel-2.

Myin e Thein [56] aplicaram um *Random Forest* em 1000 imagens do Sentinel-1 da região de Bago, em Myanmar, com acurácia de 93,18%.

Huang *et al.* [35] projetaram um modelo chamado *WaterDetectionNet* (WDNet) para mapear alagamentos, usando imagens de radar de abertura sintética [22] do Sentinel-1 sobre lago de Poyang, na China. O modelo WDNet com módulo de atenção própria atingiu acurácia de 0,987.

Bouchelkia *et al.* [8] propuseram um abordagem para dectectar diferenças em terrenos pós-inundações, com pares de imagens do Sentinel-2 antes e depois da ocorrência de

inundações na cidade de Derna na Líbia. A arquitetura de rede neural profunda [69] proposta pelos autores obteve área sobre união de 95,8%.

Sudiana *et al.* [75] classificaram imagens do Sentinel-1 de áreas urbanas em Jacarta, capital da Indonésia, utilizando uma *3D-CNN* [61]. Conseguiram acurácia de 71,8% com 20 imagens de satélite.

3.2.1 U-Net

Com o objetivo de segmentar e classificar imagens de alagamento de diferentes fontes, Ahmed *et al.* [2] compararam diferentes arquiteturas para esses trabalhos: *DeepLabv3* [13]; *MaskRCNN* [31]; e *U-Net* [64]. O *DeepLabv3* obteve interseção sobre união [63] médio de 0,95, em comparação com 0,93 do *MaskRCNN* e 0,94 do *U-Net*.

Alhady *et al.* [4] compararam a performance do *U-Net* em imagens do Sentinel-2 com diferentes métodos de índice de água [50], onde a utilização do *Modified Normalized Difference Water Index* [84] obteve a melhor performance, com área sobre união de 97,71%/.

Chandram *et al.* [12] propuseram um sistema de detecção de alagamentos baseado no *U-Net* com área sobre união de 0,85.

Pech-May *et al.* [59] classificaram imagens do Sentinel-1 do estado mexicano de Tabasco, utilizando o *U-Net*. Os autores compararam a acurácia de área sobre união treinando em diferentes épocas, onde o modelo treinado em 200 épocas foi o melhor, com acurácia de 94,31% e área sobre união de 73,02%.

3.3 Alagamento em áreas urbanas

Akshya e Priyadarsini [3] utilizaram imagens aéreas de áreas alagadas na Índia para classificar o grau de severidade de alagamentos. Inicialmente, é realizada a extração de características utilizando *Bag of Visual Words (BoVW)*. Depois uma clusterização com *k-means*. Na classificação eles usaram uma abordagem híbrida, feita através de uma *SVM*. Usando um conjunto de dados de 200 imagens, sendo 100 para cada classe, a abordagem conseguiu acurácia média de 92%. Apesar da abordagem interessante de utilizar *BoVW* para classificação com aprendizado de máquina clássico, os autores somente avaliaram *SVM* com diferentes *kernels*, sem comparar com outros algoritmos de aprendizado de máquina. Além disso, o modelo não só foi testado em somente um conjunto de dados,

como o conjunto de dados usado não foi disponibilizado.

Sazara *et al.* [68] abordaram o problema de classificação de inundação (ou não em imagens) utilizando a saída de um extrator de características para treinar modelos de aprendizado de máquina diferentes. Para extração de características foram avaliados Padrões Locais Binários (LBP) , Histograma de Gradientes Orientados (HOG) e uma rede neural *VGG-16*. Os modelos de aprendizado de máquina treinados nas características extraídas foram a Regressão Logística, k-NN, e uma Árvore de Decisão. A combinação que forneceu melhores resultados, com precisão de 0,94 e revocação de 0,97 , foi o uso da *VGG-16* como extrator de características e a Regressão Logística como classificador. O conjunto de dados utilizado consistiu de 253 imagens de alagamento e 238 sem alagamento foi disponibilizado e foi usado para comparação do melhor modelo desta pesquisa, na Seção 5.5. Entretanto, os autores não avaliaram o modelo em outros conjuntos de dados.

Jony *et al.* [42] classificaram binariamente alagamentos em imagens retiradas de redes sociais. Para isso, utilizaram três diferentes *CNNs* para extração de características com uma simples rede neural para classificação binária do problema, usando duas abordagens, sobre o conjunto de dados do *Disaster Image Retrieval from Social Media 2017*[53]. A primeira abordagem utilizou InceptionV3 e Xception como extratores, enquanto a segunda abordagem utilizou *VGG-16* como extrator. Os resultados mostraram que realizar uma média dos resultados de cada classe entre as duas abordagens gerou melhor acurácia que usar cada abordagem unicamente. As abordagens em conjunto, a primeira abordagem e segunda abordagem tiveram acurácia de 93% , 92,8% e 90,6% respectivamente. Este trabalho utilizou o conjunto de dados do desafio *MediaEval 2017* e comparou seus resultados com outros métodos apresentados no mesmo desafio, obtendo resultados competitivos com os melhores deste desafio.

Vineeth e Neeba [83] desenvolveram um método para calcular a profundidade de alagamentos e classificar se há alagamento ou não em um conjunto de dados de aproximadamente 5000 imagens. Para classificação, foi utilizada uma MobileNet, alcançando acurácia de 0,94. A utilização de *VGG-16* para o cálculo de profundidade, separada em 4 classes, resultou em acurácia de 0,78. Este trabalho não disponibilizou o conjunto de dados e também não comparou seus resultados com outros modelos ou em outros conjuntos de dados.

Piedad *et al.*[60] propuseram um sistema de detecção de alagamento utilizando imagens de **circuito fechado** de câmeras. O conjunto de dados utilizado consiste de 30000 imagens separadas em dia e noite. Primeiramente, as imagens passam por análise de cena

utilizando o DeepLabv3, um modelo de aprendizado profundo e segmentação semântica, onde objetos são detectados e são aplicadas cores diferentes em tais objetos para facilitar contagem e visualização. As imagens pré-processadas com essas cores foram utilizadas para o treino e teste de uma *CNN*, que obteve 80,67% de acurácia e 81% de revocação. Em termos de acurácia entre dia e noite, enquanto a acurácia para imagens de dia ficou com média de 87,08%, a acurácia para imagens de noite ficou com média de 70,66%. O modelo não foi testado em outros conjuntos de dados ou comparados com outras arquiteturas. O conjunto de dados também são disponibilizado.

Pally e Samadi[58] desenvolveram um pacote em *Python* chamado *Flood Image Classifier* para classificar e detectar objetos em imagens de inundações. O pacote consiste em diversas arquiteturas de *CNNs* (Mask RCNN, Fast RCNN, SSD MobileNet, EfficientDet, YOLOv3) treinadas em mais de 9000 imagens. Também utilizam detecção de borda através do algoritmo de Canny[10] para estimar o nível de água e **técnicas de segmentação** para identificar e medir a inundação, sendo possível avaliar a profundidade e gravidade dos danos. Os autores disponibilizaram os modelos de detecção [no repositório de GitHub](#).

Com o objetivo de detectar se ocorreu um desastre, e se há fogo ou inundação, Sghaier *et al.*[70] utilizaram uma simples *CNN* de três camadas de convolução para classificar a imagem em 4 classes: se há fogo, se não há fogo, se há inundação; e se não há inundação. Com um conjunto de dados de 1045 imagens contendo fogo, 231 imagens sem fogo, 1034 imagens com inundação e 326 imagens sem inundação, a *CNN* obteve acurácia de 0,99. Vale apontar que ambas as categorias de não haver fogo e de não haver inundação representam situações de normalidade, portanto, são redundantes e poderiam ser uma única classe, resumindo o estudo para um problema de três classes. Também não houve nenhum tipo de comparação com outras arquiteturas ou conjuntos de dados, além do conjunto utilizado não ser disponibilizado.

Em um conjunto de dados de 2000 imagens, Agung *et al.*[1] utilizaram uma MobileNet com três novas camadas antes da camada completamente conectada para realizar a classificação binária de alagamento em ruas. Os autores obtiveram acurácia de 0,96, precisão de 0,95 e revocação de 0,97. Entretanto, estes resultados não foram comparados com nenhuma outra arquitetura, inclusive com a própria MobileNet sem as alterações realizadas. Assim como também não houve comparação com outros conjuntos de dados.

Islam *et al.*[39] propuseram uma abordagem combinando *CNN* e algoritmos de ordenação para classificar imagens de drones em três diferentes graus de severidade. O sistema criado detecta o nível de alagamento e calcula a distância do drone até a área afetada,

gerando então um valor ponderado representando a prioridade de atenção do drone à cada área detectada, estes valores são ordenados pelo algoritmo de ordenação. Com 1011 imagens divididas entre treino, teste e validação, os modelos testados DensetNet e InceptionV3 conseguiram acurácia de 0,81 e 0,83, respectivamente. Como reconhecido pelos próprios autores, mais estudos sobre a viabilidade do método descrito precisam ser feitos, visto que não houve comparação com outros conjuntos de dados.

Hidayat *et al.* [32] compararam a performance do *VGG-16* e *MobileNet* para classificar o alagamento em rodovias. Com um conjunto de 2000 imagens da cidade de Macáçar, na Indonésia, os autores obtiveram acurácia de 99% e tempo de processamento de 9 segundos com o *MobileNet*. Já o *VGG-16* conseguiu acurácia de 96% e tempo de processamento de incríveis 69 segundos. Os autores não especificaram a configuração da máquina onde os modelos foram treinados, mas justificaram a diferença discrepante entre o processamento dos modelos através da simplicidade do *MobileNet* em comparação ao *VGG-16*. Não houveram avaliações em outros conjuntos de dados, o conjunto de dados utilizado não foi disponibilizado ou descrita a sua quantidade de imagens.

Arora e Bhavadharini [5] compararam as arquiteturas *VGG-19* e *VGG-16* para classificar cenas de desastres naturais, mais especificamente ciclones, terremotos, inundações e incêndios florestais. Treinando os modelos em 48 épocas, o modelo *VGG-19* conseguiu acurácia de 94,0% e o *VGG-16* conseguiu 92,1%, com o total de aproximadamente 4000 imagens. Apesar da alta acurácia, os autores não especificaram a acurácia por classe. Essa é uma análise importante pois o conjunto de treino possui 3321 imagens, onde a classe de terremoto possui 1012 imagens, enquanto a classe de ciclone possui somente 696 imagens, mostrando um claro desequilíbrio. Outras arquiteturas não foram avaliadas, assim não houve testes em outros conjuntos de dados ou a disponibilização do conjunto de dados utilizado.

A Tabela 1 mostra um resumo da literatura sobre classificação de estados de inundações apresentada neste capítulo.

Tabela 1: Resumo dos trabalhos relacionados

Trabalho	Extração de Característica	Classificação	Quantidade de Imagens
Esta pesquisa	Não	VGG-19, InceptionV3, DenseNet, MobileNetV2, ViT	4620
Akshya	BoVW	k-means e SVM	200
Sazara et al.	VGG16	Regressão Logística	491
Jony et al.	InceptionV3, Xception, VGG16	CNN própria	Não Informado
Vineeth e Neeba	Não	MobileNet	5000
Piedad et al.	Não	CNN própria	30000
Pally e Samadi	Não	Mask RCNN, Fast RCNN, SSD MobileNet, EfficientDet, YOLOv3	9000
Sghaier et al.	Não	CNN própria	1045
Agung et al.	Não	MobileNet	2000
Islam et al.	Não	DenseNet e InceptionV3	1011
Hidayat et al.	Não	VGG16 e MobileNet	Não Informado
Arora e Bhavadharini	Não	VGG16 e VGG19	4000

4 Materiais e Métodos

Este capítulo descreverá todo o processo para a criação da metodologia usada. Desde a descrição e adaptação do conjunto de dados utilizado, cobrindo as escolhas dos modelos de classificação até a forma de avaliação dos resultados, além das tecnologias utilizadas em cada passo.

Apesar de ser possível classificar a situação de alagamento de uma rua em diferentes níveis, para o problema de classificação descrito, as diferentes situações foram simplificadas para um problema binário, onde o objetivo é definir se a quantidade de água na rua permite o trânsito de veículos normalmente (classe 0), ou a quantidade é o suficiente para impactar o trânsito (classe 1).

4.1 Conjunto de dados

O conjunto de dados consiste em imagens de 77 câmeras diferentes da cidade do Rio de Janeiro, capturadas pelo sistema de câmeras do Centro de Operações do Rio (COR). Neste conjunto, cada câmera possui 30 imagens para cada uma das classes deste problema de classificação binário. Ou seja, o conjunto de dados é formado por 60 imagens de cada câmera, totalizando 4620 imagens. Do total de 77 câmeras, 60 foram escolhidas para compor o conjunto de treino e 17 câmeras para validação, ou seja 78% das câmeras compõem o conjunto de treino e 22% o conjunto de validação.

Este trabalho frequentemente se refere ao conjunto de dados usado pelo número de câmeras, em vez de descrever diretamente o número de imagens. Isso é feito propositalmente, visto que foi decidido não utilizar imagens da mesma câmera em treino e em validação, para evitar qualquer tipo de viés em sua validação. Portanto, cada câmera do conjunto de dados pertence somente ao conjunto de treino ou o conjunto de validação.

Como as imagens foram capturadas em uma cidade movimentada durante o período de chuvas, uma variedade de imagens foi utilizada para o treinamento, desde ruas vazias

em um dia ensolarado até tráfego intenso em dias chuvosos durante a noite. Devido a chuvas extremamente fortes, algumas lentes de câmeras estavam muito molhadas para distinguir o que era exibido. Essas imagens não foram incluídas no conjunto de dados.

Ao longo do ano, foram captadas imagens de 472 câmeras distintas, com diferentes proporções de imagens de cada classe para cada câmera. Dessa forma, para criar um conjunto de dados balanceado, foram escolhidas as câmeras com quantidade suficiente de imagens de ambas as classes para que todas as câmeras do conjunto de dados tenham a mesma quantidade de imagens.

4.1.1 Captação de imagens

As imagens foram captadas ao longo de 2023 em diferentes condições de iluminação e épocas de chuva, com diferentes intensidades de chuva. Foi necessário captar tais imagens de maneira eficiente e inteligente, visto que o sistema de câmeras do COR é composto por milhares de câmeras, tornando custoso o monitoramento de todas elas, e que as chuvas, apesar de intensas, são imprevisíveis: poderiam cair a qualquer hora, durar intervalos de tempo muito variados.

O sistema de captação foi desenvolvido em Python [80], utilizando a API do Waze para receber notificações de possíveis situações de alagamento. A biblioteca OpenCV [40] foi utilizada para captar as imagens e a API do Google Cloud para upload e armazenamento. Foram utilizados alertas do aplicativo Waze sobre alagamento de ruas e notificações do próprio sistema do COR para definir quais ruas da cidade do Rio de Janeiro estavam alagadas e assim, começar a captação de imagens dessas ruas.

4.1.2 Análise de imagens

Para a classificação das imagens, foi criada uma aplicação web baseada em Flask e utilizando MongoDB como banco de dados. As imagens já estavam separadas em seis (6) níveis definidos pelo próprio COR. Estes níveis, em ordem crescente de severidade, são: Normal; Poça; Lâmina; Alagamento; Transbordo; e Bolsão.

Para simplificar, tendo em vista que não há um limite claramente definido entre cada uma dessas classificações, as classes 'Normal' e 'Poça' foram definidas como classe 0, visto que não há impacto no trânsito de veículos. À partir do surgimento de lâminas d'água, o trânsito de veículos começa a ser afetado visto que estes devem diminuir sua velocidade para evitar derrapagem [52] então, a partir do nível3 'Lâmina', as classes mais severas

foram definidas como classe 1.

A Figura 6, mostra uma importante avenida na zona sul do Rio de Janeiro, no entorno da Lagoa Rodrigo de Freitas. A Av. Epitácio Pessoa na altura da Rua Maria Quitéria, no lado desta lagoa próximo ao bairro de Ipanema. A Figura 7, mostra a rua Jardim Botânico, no cruzamento com a rua Pacheco Leão, no bairro Jardim Botânico. Estas figuras são exemplos de vias classificadas como **não alagadas**, ou classe 0. Quando há água na via, mas ela não impede o tráfego de veículos, decidiu-se que tais situações não seriam classificadas como *alagamento*.

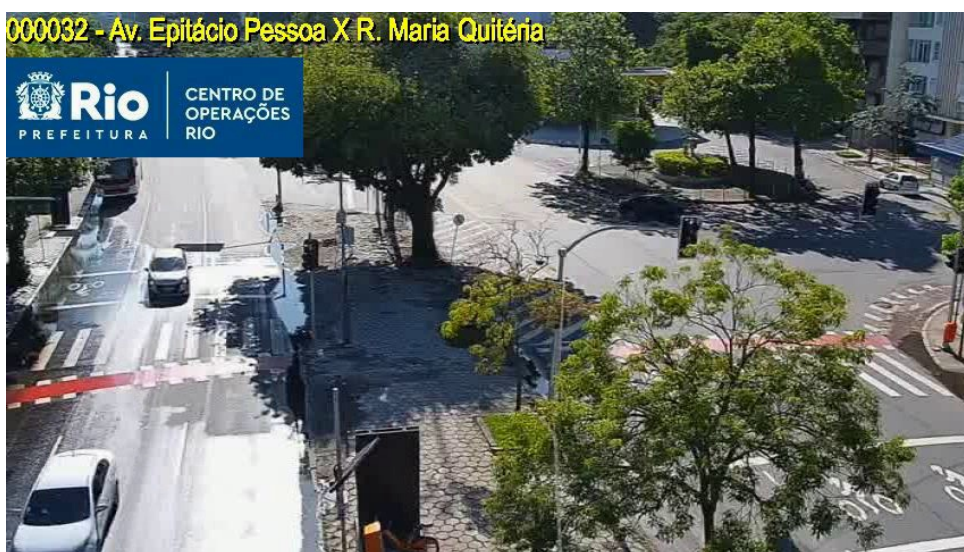


Figura 6: Via completamente seca, rotulada como classe 0



Figura 7: Rua molhada, ainda rotulada como classe 0

Por outro lado, a Fig. 8 e a Fig. 9 mostram capturas de vias classificadas como **alagadas**, ou classe 1. Em ambas as imagens, é possível ver água cobrindo bastante

a rua, dificultando ou impossibilitando a passagem de veículos. A Fig. 8 é uma cena capturada pela câmera localizada na zona sul do Rio de Janeiro, no entorno da Lagoa Rodrigo de Freitas. A Av. Borges de Medeiros em seu cruzamento com a Rua Saturnino de Brito, no lado desta lagoa próximo ao bairro Jardim Botânico. A Fig. 8 é o exemplo de uma rua (na zona norte da cidade) parcialmente alagada, dificultante o transito dos veículos onde as calçadas ainda podem ser identificadas. A Fig. 9 é o exemplo de uma rua (da zona sul), sem distinção de calçada ou rua, completamente alagada. Essas situações constituem a classe *alagamento*.



Figura 8: Rua parcialmente alagada (classe 1).



Figura 9: Avenida totalmente alagada (classe 1).

As Figuras 10 e 11 mostram a quantidade de imagens (eixo vertical) de cada câmera (eixo horizontal) para a classe 0 e classe 1, respectivamente. Essas imagens consistem

no conjunto de dados original capturado de 472 câmeras do sistema do COR, sem balanceamento das classes. As imagens de cada classe correspondem as cores, ou seja se identificam em **alagadas** ou **não alagadas** de acordo com as cores (azul ou vermelho) de parte das linhas das câmeras mostradas no eixo horizontal do histograma de distribuição. Pode-se observar que algumas câmeras possuem muito mais imagens da classe 1 que a outras, provavelmente por ter ocorrido mais chuvas nas regiões onde estas estão localizadas. Assim, a criação de um conjunto de dados sem uma análise prévia da representatividade de cada câmera no conjunto possivelmente criaria um viés em qualquer modelo treinado neste conjunto.

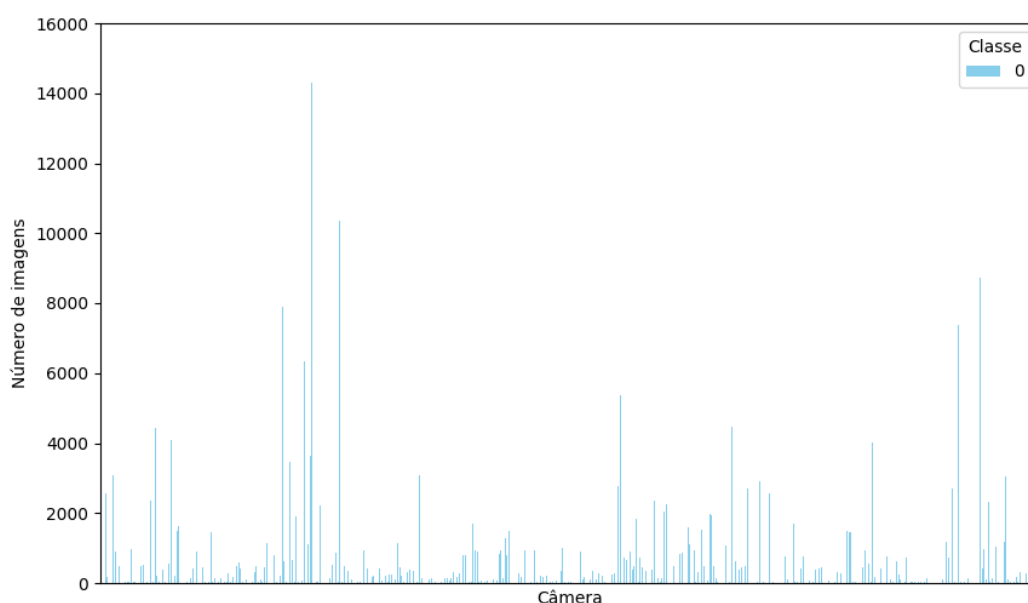


Figura 10: Quantidade de imagens de normalidade por câmera disponível.

A Figura 12 mostra uma distribuição de frequência de câmeras por quantidade de imagens, separadas em intervalos de 100 imagens. O intervalo com maior ocorrência de câmeras indica o desbalanceamento do conjunto de imagens captadas onde 206 câmeras, ou 43,6%, possuem até 100 imagens ao todo.

Analisando o total de 472 câmeras, 330 delas possuem menos de 500 imagens registradas de ambas as classes, ou 69,9% das câmeras. Vale ressaltar que esses dados não levam em conta a proporção de imagens de cada classe no total em cada câmera. Diante do exposto, foi necessário considerar qual a quantidade de imagens de cada classe por câmera deve ser considerada para criar um conjunto de dados balanceado e diverso em sua representatividade de câmeras de lugares diferentes. Para isso, foi analisada a

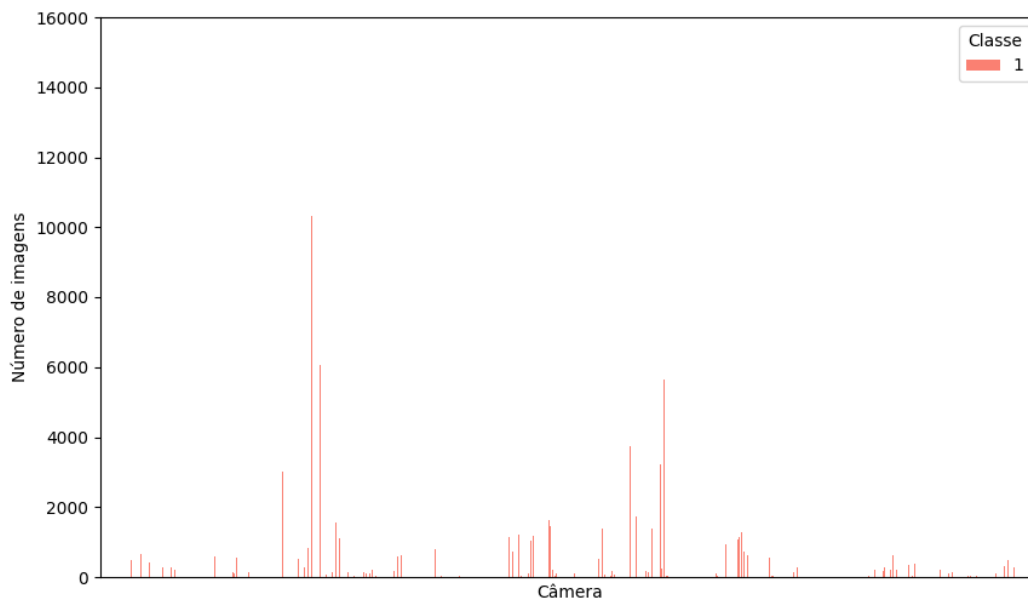


Figura 11: Quantidade de imagens de alagamento por câmera disponível.

quantidade de câmeras que possui um determinado número mínimo de imagens para cada classe, permitindo saber quantas imagens e qual seria a variedade de um conjunto de dados balanceado formado com as imagens captadas e classificadas.

A Figura 13 mostra a quantidade de câmeras para a condição descrita. Inicialmente, foi julgado que 10 imagens de cada classe por câmeras formaria um conjunto de dados com relativamente poucas imagens levando em consideração as outras possíveis quantidades de imagens por câmera. Essa escolha geraria um conjunto de dados de 1800 imagens, visto que há 90 câmeras com pelo menos 10 imagens de cada classe. Uma possibilidade é um conjunto de dados com 50 imagens de cada classe que forneceria 60 câmeras, para um total de 6000 imagens. Entretanto, o conjunto de dados ficaria muito maior e com menor variedade de câmeras que usar, por exemplo, o valor escolhido de 30 imagens por câmeras, que teria 77 câmeras.

Portanto, as arquiteturas serão treinados no conjunto de dados de 77 câmeras, com 4620 imagens divididas em 30 imagens de cada classe para cada câmera. Esse conjunto foi dividido em 60 câmeras para treino e 17 para validação. Ao final da avaliação dos modelos de *CNNs*, uma reavaliação do número de imagens por câmeras é feito para tomar uma decisão certa sobre a melhor quantidade de imagens por câmeras, e se essa mudança acarretaria melhora nos índices usados para avaliação do modelo, como é apresentado na

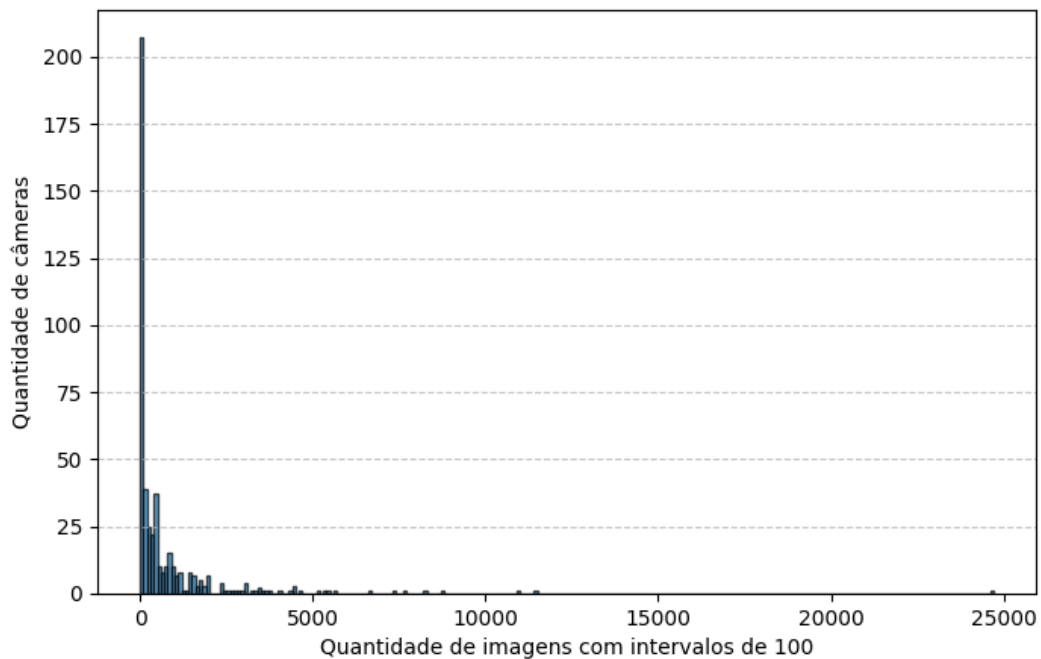


Figura 12: Histograma de quantidade de imagens de armazenamento por câmera disponível.

seção [5.2.3](#).

4.1.3 Pré Processamento

Diversas etapas de pré-processamento foram aplicadas para melhorar a qualidade das imagens no conjunto de dados.

Primeiramente, uma máscara foi aplicada para remover o logotipo do COR (ou seja, a região superior esquerda com os brasão da cidade e palavras brancas sobre um fundo azul) das imagens para evitar que o logotipo enviesasse o processo de aprendizado do modelo. Isso foi feito zerando a região da imagem onde o logotipo do COR está localizado, já que sua posição é estática. Como o logotipo está presente em todos os quadros, não foi possível recriar a região apagada com quadros intermediários.

Segundo, como algumas imagens podem apresentar claridade excessiva, os canais de imagem RGB originais foram convertidos para o espaço de cor YC_rC_b para permitir o ajuste no canal Y , reduzindo a claridade da imagem. Essa transformação no canal Y foi feita através de uma transformação linear na intensidade do canal. Os valores de pixel no canal Y são multiplicados por um fator que reduziria sua influência na imagem. Tal fator foi denominado fator de claridade (FC), um número real positivo entre 0 e 1, avaliado

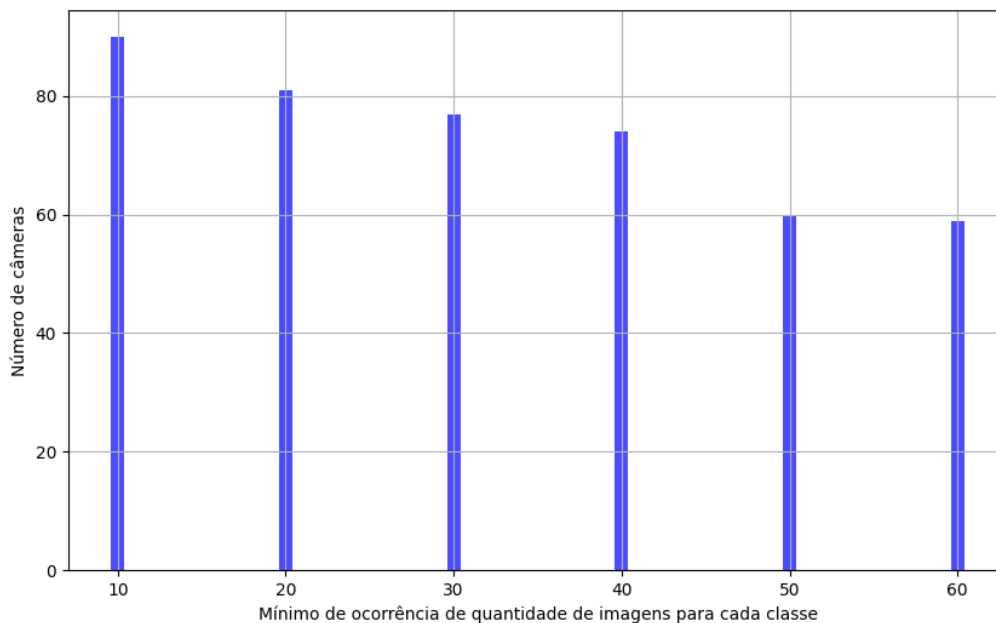


Figura 13: Quantidade de câmeras com um número mínimo de imagens de cada classe.

com um valor inicial de 0,5.

Então, cada informação de pixel YC_rC_b é convertida de volta para RGB , para ser mostrada nas telas do computador colorida como originalmente. Entretanto, outros valores de fator de claridade (FC) foram avaliados posteriormente na seção 5.2. As Figuras 14 e 15 mostram a mesma cena de um local onde o FC foi aplicado. É possível notar a diferença na intensidade do claridade da cena após a alteração no canal Y .

Para introduzir mais variedade às imagens no treinamento, em seguida foi aplicada uma inversão horizontal aleatória foi aplicada com uma probabilidade de 50% nos dados de entrada. Em outras palavras, cada imagem da divisão de treinamento tem 50% de chance de ser invertida horizontalmente ou não.

Além disso, transformações padrões de redimensionamento, corte e normalização de intensidades são usadas para preparar as imagens para cada modelo *CNN*. É aplicado aos dados inicialmente um processo de redimensionamento para transformar cada quadro ajustado ao tamanho esperado de cada modelo, onde as imagens com dimensão 854 x 480 pixels são redimensionadas para 256 x 256 pixels. Então eles são submetidos a um corte de dimensão 224 x 224 pixels de mesmo centro da imagem. Com exceção do *Inception*, que redimensiona para 342 x 342 pixels e realiza um corte central de 299 x 299 pixels. Os valores do conteúdo de cada pixel são redimensionados para estarem no intervalo [0,1].



Figura 14: Exemplo de região sem alteração no canal Y .

Finalmente, todo o conjunto de dados foi normalizado pela média e desvio padrão específicos de cada modelo: Todos os modelos avaliados têm a mesma matriz média de $(0,485 ; 0,456 ; 0,406)$ e matriz de desvio padrão de $(0,229 ; 0,224 ; 0,225)$ para cada canal RGB , usados para normalizar a imagem.

4.2 Modelos de classificação

Com base na nos trabalhos da literatura sobre alagamento urbano anteriormente apresentados, os modelos MobileNetV2 [67], VGG19 [74], InceptionV3 [76] e DenseNet [36] foram avaliados. Além disso, a arquitetura de *Visual Transformer* (ViT)[45] foi incluída. Essa adição foi feita porque vários modelos de classificação de imagens de alto desempenho são baseados em arquiteturas *Visual Transformers*, como $ViT - Huge$ usado no conjunto de dados CIFAR-10 [24] e $ViT - Large$ no conjunto de dados STL-10 [28, 43].

Todos os modelos foram treinados em 50 épocas, onde o *Early Stopping* foi empregado com uma restrição de 10 épocas e decaimento da taxa de aprendizagem de 0,1. Três



Figura 15: Exemplo de mesma região com 50% do valor original de Y .

diferentes taxas de aprendizagem foram aplicadas em todos os modelos, visto que as diferentes complexidades dos modelos podem resultar em diferentes taxas de aprendizagem ótimas. O *Stochastic Gradient Descent* (*SGD*) foi usado como otimizador inicialmente, entretanto, outros otimizadores foram testados após a avaliações dos modelos, como pode ser visto na seção 5.3.

A transferência de aprendizado foi aplicada a todos os modelos, inicializando-os com os pesos padrões do IMAGENET. Isso foi feito para utilizar os pesos aprendidos anteriormente para auxiliar no aprendizado dos pesos das novas classes [46]. Posteriormente, a última camada totalmente conectada foi modificada para ter apenas duas saídas, se adequando ao problema de classificação binária.

4.3 Avaliação dos modelos

Para avaliar o desempenho dos cinco modelos com diferentes taxas de aprendizagem, inicialmente foi utilizado a acurácia para distinguir o melhor modelo. Após a identificação

do melhor modelo, a precisão e revocação também foi utilizada para melhor entender seu desempenho.

Como o conjunto de dados está completamente balanceado, conforme mencionado na Seção 4.1, com ambas as classes igualmente representativas, esses avaliadores fornecem um cenário adequado para compreensão dos resultados. Adicionalmente, todas as câmeras possuem o mesmo número de imagens, garantindo assim uma representação equilibrada entre as fontes de dados e suas distribuições pela cidade.

Para calcular a acurácia, precisão e revocação, inicialmente foram obtidos os seguintes valores: Verdadeiros Positivos (TP), que correspondem ao número de rótulos positivos classificados corretamente pelo modelo; Verdadeiros Negativos (TN), que representam os rótulos negativos classificados corretamente; Falsos Positivos (FP), que são os rótulos positivos classificados incorretamente; e Falsos Negativos (FN), que indicam o número de rótulos negativos classificados de forma incorreta.

O primeiro índice de avaliação selecionada para análise foi a acurácia, pois ela fornece uma visão geral da capacidade do modelo em classificar corretamente os exemplos em um conjunto de dados balanceado.

A acurácia pode ser calculada pela seguinte fórmula:

$$Acuracia = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1)$$

O índice de avaliação de precisão indica a porcentagem de imagens classificadas como positivas que realmente pertencem à classe positiva [27]. É especialmente relevante para aumentar a confiabilidade do modelo na identificação correta da situação de inundação.

A Equação 4.2 formaliza o cálculo da precisão:

$$Precisao = \frac{TP}{TP + FP} \quad (4.2)$$

Além disso, a revocação pode ser empregada como um avaliador final para calcular a taxa de imagens positivas classificadas corretamente [54]. Ela é útil para garantir que o menor número possível de situações de inundação seja classificado incorretamente. A revocação mede a capacidade do modelo de identificar corretamente todas as instâncias relevantes. A equação a seguir define o cálculo da revocação:

$$Revocacao = \frac{TP}{TP + FN} \quad (4.3)$$

O próximo capítulo apresenta os resultados obtidos.

5 Experimentos e Resultados

Esta seção apresenta os resultados das cinco arquiteturas analisadas. Em seguida, a arquitetura com melhor desempenho é analisada mais detalhadamente em diferentes aspectos. Todos a pesquisa foi realizada em uma máquina com placa gráfica NVIDIA GeForce RTX 3060 Ti, processador Intel Core i7-11700K de 3,60GHz e 16GB de memória RAM.

5.1 Resultados dos Modelos

A Tabela 2 exibe a acurácia de cada modelo com diferentes taxas de aprendizagem usando *SGD* como otimizador. A taxa de aprendizagem de 1×10^{-5} apresentou melhores acurácias no geral, mas não gerou o melhor modelo. Nenhum modelo obteve boa acurácia com taxa de aprendizagem de 1×10^{-5} , indicando que uma taxa muito baixa limitou a capacidade dos modelos de convergir a uma boa resposta.

O *MobileNetV2* obteve os piores resultados com as outras duas taxas de aprendizagem. Já o *ViT* apresentou o melhor desempenho, sua acurácia tanto com taxa de aprendizagem de 1×10^{-3} quanto de 1×10^{-4} foram as melhores.

O desempenho do *MobileNetV2* pode ser explicado por sua arquitetura, projetada

Tabela 2: Acurácia dos cinco modelos com diferentes taxas de aprendizagem.

Modelos	Taxa de aprendizagem		
	1x10-03	1x10-04	1x10-05
VGG19	0,808	0,732	0,524
InceptionV3	0,795	0,731	0,529
DenseNet	0,802	0,658	0,592
MobileNetV2	0,781	0,608	0,534
ViT	0,816	0,826	0,511

para uso em dispositivos com menor poder computacional [67], possuindo, portanto, um número significativamente menor de parâmetros treináveis em comparação com as outras arquiteturas.

Um comportamento que todos os modelos tiveram, exceto o *ViT*, foi de ter menor acurácia de acordo com a diminuição da taxa de aprendizagem. Isso pode significar que taxa mais baixas não são adequadas para ajustar a grande quantidade de parâmetros em redes profundas.

Apesar de tanto o *VGG* quanto o *DenseNet* chegarem a acurácias de 80%, o *ViT* mostrou resultados ligeiramente melhores. Outro ponto a ser considerado é que enquanto o *InceptionV3* levou 47 minutos para completar seu treinamento, o *ViT* levou apenas 22 minutos no mesmo conjunto de dados. Portanto, o *ViT* foi escolhido como o modelo a ser utilizado na continuidade deste trabalho.

5.2 Avaliação do melhor modelo

Após a escolha da arquitetura de melhor desempenho, outras análises devem ser feitas para saber se é possível melhorar seus resultados e o quão adequado foi seu treinamento.

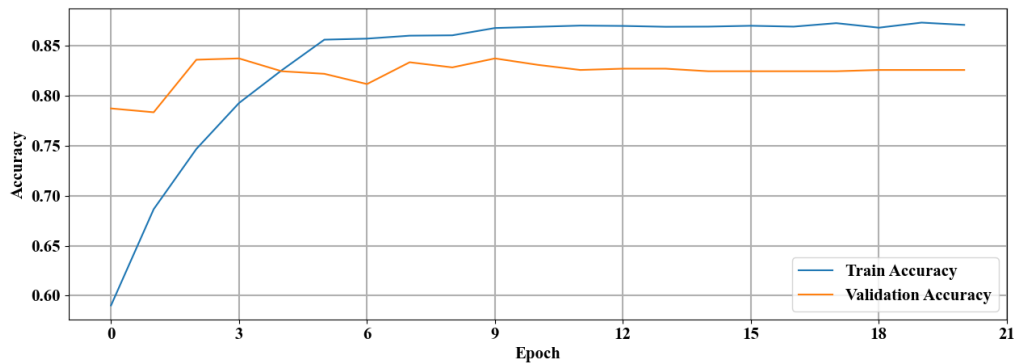
5.2.1 Diminuição da intensidade luminosa

Após a escolha da arquitetura de melhor desempenho, diferentes valores para nesta dissertação denominado de fator de claridade (*FC*), conforme mencionado em 4.1.3, foram avaliados usando o *ViT*, a fim de analisar se o modelo poderia ser aprimorado, levando em consideração o claridade excessivo em algumas imagens.

A Tabela 3 mostra o desempenho do *ViT* com quatro valores diferentes para *FC*, onde o valor 1 para *FC* significa que a intensidade luminosa da imagem não foi alterada, ou seja a quantidade de regiões com pixels na cor branca permanece a original. Reduzir o *FC* melhorou gradualmente o desempenho do modelo, porém um *FC* muito baixo impactou negativamente o modelo, como mostram os resultados com *FC* igual a 0,25. Reduzir o *FC* para 0,75 melhorou os resultados marginalmente, no entanto, a escolha inicial de 0,5 mostrou-se a melhor entre os valores testados.

Tabela 3: Redução de áreas de brancos e desempenho do *ViT*

Fatores de clareza	Acurácia	Precisão	Revocação
0,25	0,703	0,758	0,604
0,5	0,826	0,863	0,778
0,75	0,769	0,802	0,738
1	0,756	0,782	0,708

Figura 16: Acurácia do *ViT* durante treino e validação.

5.2.2 Curvas de acurácia e perda do melhor modelo

A Fig. 16 e a Fig. 17 mostram o *ViT* atuando em seu treinamento (linha azul) e validação (linha vermelha).

No treinamento, a acurácia (Fig. 16) converge para 0,87, enquanto na validação permanece em aproximadamente 0,83. No entanto, ao analisar apenas a acurácia no conjunto de validação, podemos observar que ela começou com valores relativamente altos antes de convergir. A pequena discrepância entre a acurácia de treinamento e validação indica que não houve *overfitting* dos dados, e o modelo conseguiu criar uma boa generalização.

Analisando a perda (loss) (Fig. 17) durante o treinamento, podemos ver que, apesar da alta acurácia no conjunto de treino, a perda ainda era relativamente alta antes de se estabilizar em torno de 0,37 (linha azul), enquanto o conjunto de validação se estabilizou em 0,43 (linha vermelha).

A pequena diferença na acurácia entre os conjuntos de treinamento e validação (Fig. 16) pode ser justificada pela introdução de características no conjunto de validação que não estão presentes no conjunto de treinamento.

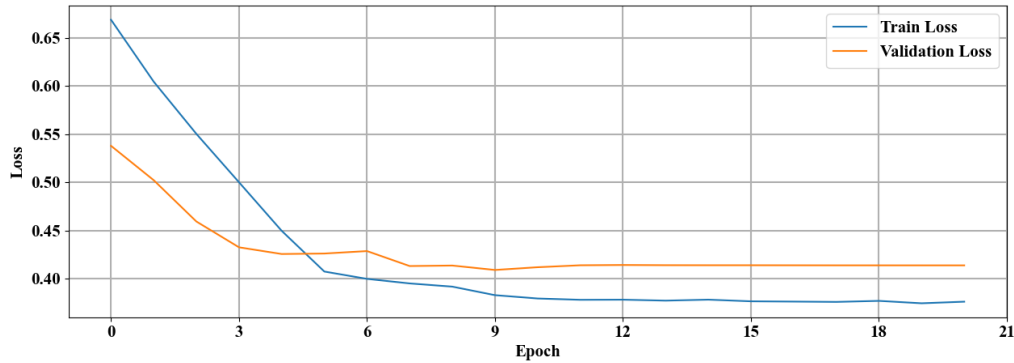


Figura 17: Perda do *ViT* durante treino e validação.

5.2.3 Melhor modelo com diferentes quantidades de câmeras

Com o objetivo de verificar se aumentar a quantidade de câmeras no conjunto de treino levaria a melhorias significativas de sua performance, a arquitetura *ViT* foi treinada com conjuntos de treinos menores. Esses conjuntos de treino possuem menores quantidades de câmeras, mas mantêm a quantidade de imagens por classe para cada câmera.

Para montar estes conjuntos a serem comparados ao utilizado, entre as 60 câmeras que compõem o original foram escolhidas aleatoriamente 30 câmeras para montar o segundo conjunto. Dessas 30 câmeras foram selecionadas aleatoriamente 15 câmeras para formar o terceiro conjunto. Portanto todos os três conjuntos de treinos possuem as mesmas 15 câmeras e o conjunto de 60 câmeras possui todo o conjunto de 30 câmeras mais 30 extras.

Assim o efeito de adicionar novas câmeras ao conjunto pode ser mais facilmente compreendido. A Tabela 4 mostra a relação entre o tamanho do conjunto de treino, sua acurácia e tempo de treino. O modelo foi testado no mesmo conjunto de teste que faz parte do conjunto de dados original.

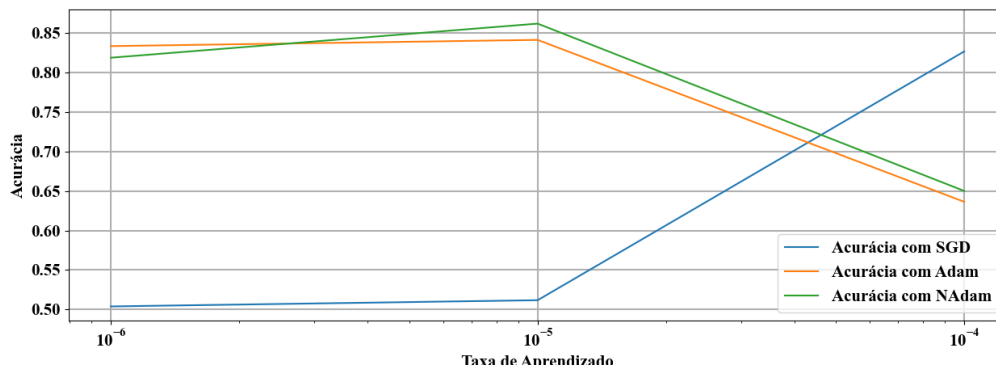
Tabela 4: Acurácia \times tempo de treino \times número de imagens usadas

Nº de Câmeras	Acurácia	Tempo (minutos)
15	0,728	17
30	0,804	18
60	0,826	21

Como esperado, aumentar a diversidade de câmeras no conjunto de treino e, consequentemente, aumentar a quantidade de imagens de treino aumentou a acurácia do modelo. Entretanto, o número utilizado está completamente adequado pois adicionar

Tabela 5: Acurácia do *ViT* com diferentes otimizadores.

Taxa de Aprendizado	Acurácia		
	SGD	Adam	NAdam
1x10-06	0,504	0,833	0,819
1x10-05	0,512	0,841	0,862
1x10-04	0,826	0,636	0,650

Figura 18: Acurácia do *ViT* com diferentes otimizadores.

mais câmeras não traria melhorias significativas na performance, visto que dobrando a quantidade de imagens de 1800 para 3600, ou seja, dobrando de 30 câmeras para 60 câmeras no conjunto de treino, somente representou uma melhoria de 0,022 na acurácia.

5.3 Comparação entre diferentes otimizadores

Apesar de inicialmente todos os modelos serem treinados utilizando *SGD* como otimizador, a escolha de diferentes otimizadores com diferentes taxas de aprendizado podem afetar a performance da *CNN*.

Baseado nos resultados de Dogo *et al.* [23], onde diferentes otimizadores foram comparados com redes neurais e conjuntos de dados de diferentes tamanhos e complexidades, essa dissertação comparou os resultados no *ViT* com *SGD*, *Adaptive Moment Estimation*[44] e o *Nesterov-accelerated Adaptive Moment Estimation*[25]. Todos os otimizadores utilizaram as mesmas configurações descritas na seção 4.2, com diferentes taxa de aprendizado iniciais.

Os resultados mostram que ambos *ADAM* e *NADAM* possuíam acurácias melhores que o *SGD*. O *NADAM* obteve a melhor acurácia: 0,86 com taxa de aprendizado de

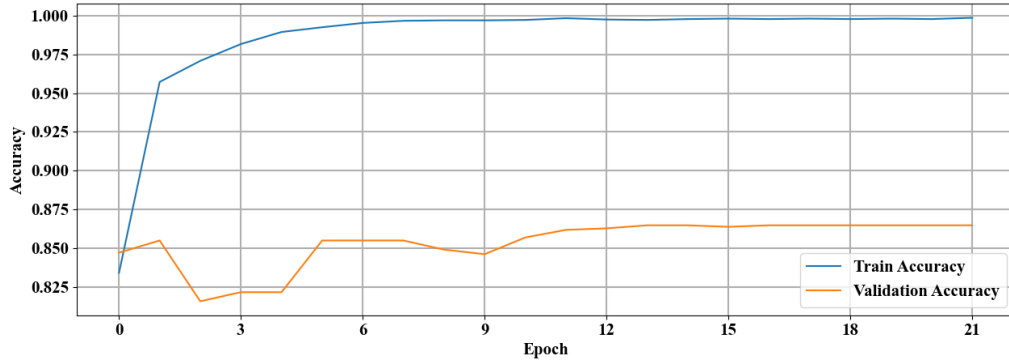


Figura 19: Acurácia do *ViT* com NAdam.

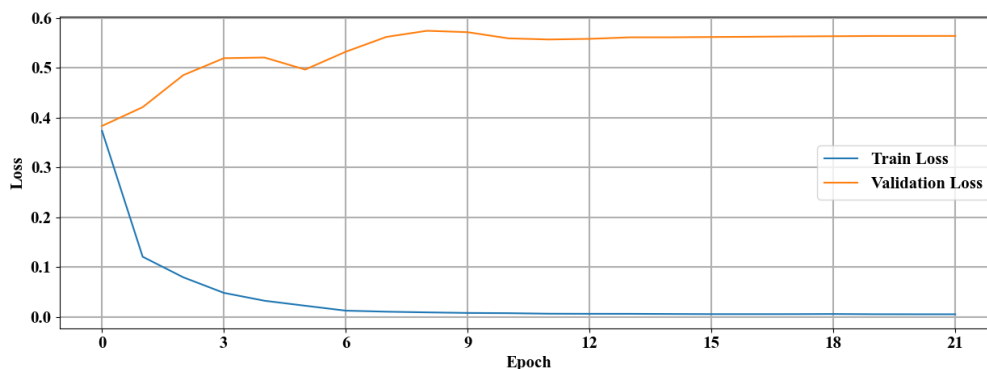
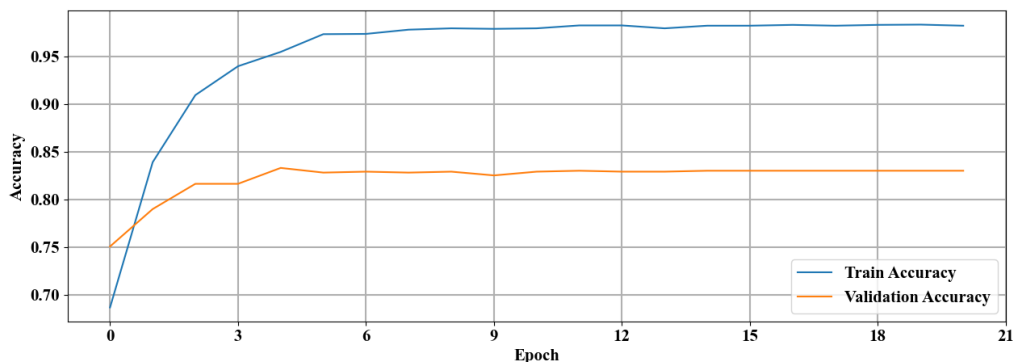
1×10^{-5} . Avaliando o *SGD*, taxas de aprendizado menores obtiveram piores resultados, provavelmente indicando que o passo foi muito pequeno para sair de mínimos locais, especialmente em uma arquitetura mais complexa como o *ViT*.

Nenhum dos otimizadores conseguiu boa performance com a taxa de aprendizado mais baixa, que pode ser explicado pelo modelo não ter feito atualizações significativas em seus pesos durante o treino. As duas versões do *ADAM* obtiveram melhor performance com uma taxa de aprendizado mais moderada, com o *NADAM* se sobressaindo. A utilização da aceleração de Nesterov para atualizações mais rápidas pelo *NADAM* pode ser o responsável pelo melhor desempenho.

As Figuras 19 e 20 mostram a acurácia e perda do *ViT* durante treino usando o *NADAM* com taxa de aprendizagem de 1×10^{-5} , onde a maior acurácia de validação foi obtida. Levando em consideração que o modelo estabilizou sua acurácia de treino em 100% enquanto a acurácia de validação ficou em volta de 86%, indica que o modelo não conseguiu generalizar bem os dados, indicando *overfitting*. Isso é corroborado pela curva de perda, onde a perda na validação aumentou, em contraste com a diminuição para quase zero da perda de treino.

Fazendo a mesma análise para a taxa de aprendizagem de 1×10^{-6} , onde a acurácia de validação foi um pouco menor, pode-se notar algumas semelhanças e diferenças. Ainda houve uma diferença significativa entre as acurácias de treino e validação, onde a acurácia de treino estabilizou por volta de 98%, enquanto a acurácia de validação ficou em 82%.

Entretanto, a curva de perda mostrou melhorias. Tanto a curva de treino quanto a curva de validação tenderam a diminuir, apesar de que a diminuição da perda na validação não ser tão significativa. Este modelo não mostrou sinais claros de *overfitting* como o modelo com maior taxa de aprendizagem, mas ainda demonstra sinais de melhorias.

Figura 20: Perda do *ViT* com NAdam.Figura 21: Acurácia do *ViT* com NAdam e menor taxa de aprendizagem.

Visto que o modelo *ViT* usando *NADAM* e taxa de aprendizagem de 1×10^{-6} obteve acurácia de validação semelhante a do modelo com o *SGD* de taxa de aprendizagem de 1×10^{-4} , entretanto, teve maior diferença entre as acurácias de treino e validação, assim como redução na função de perda, o modelo com *SGD* vai continuar a ser analisado como o melhor modelo, por possuir maior estabilidade.

5.4 Comparações entre cena diurna e noturna

Para melhor entender a iluminação da cena nos resultados da classificação do modelo [60], o melhor resultado do *ViT* com o *SGD* descrito na seção 5.3 foi avaliado em relação a suas métricas com a luz do dia e com luzes noturnas no local, com os resultados mostrados na tabela 6.

Em contraste com os resultados apresentados e descritos no capítulo 3, onde o modelo apresentou resultados substancialmente melhores em imagens diurnas em comparação com imagens noturnas [60], o modelo treinado no conjunto de dados do COR obteve

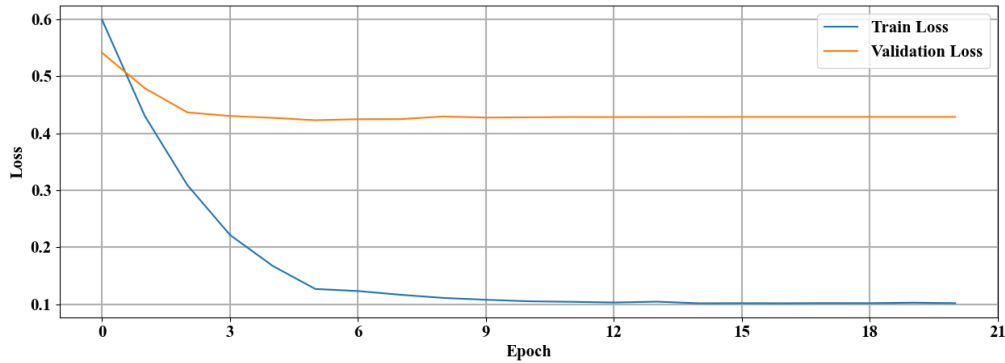


Figura 22: Perda do ViT com NAdam e menor taxa de aprendizagem.

Tabela 6: Métricas do ViT de dia e à noite.

Período do dia	Acurácia	Precisão	Revocação
Diurno	0,802	0,702	0,694
Noturno	0,758	0,800	0,758

menores diferenças na acurácia entre imagens diurnas e noturnas. Essa maior estabilidade da acurácia pode ser explicada pela redução da claridade da imagem através do pré-processamento realizado.

5.5 Outros conjuntos de dados

O modelo ViT treinado no conjunto do COR também foi testado em dois outros conjuntos de dados, o European Flood Dataset (EFD) [6] e o conjunto de dados de Sazara *et al.* [68]. O EFD é composto de 327 imagens de normalidade e 252 imagens de alagamento para um total de 579. O conjunto de dados de Sazara *et al.* possui 491 imagens, onde 238 são de normalidade e 253 são imagens de alagamento.

Podemos ver na Tabela 7 que o desempenho do modelo diminuiu em diferentes conjuntos de dados. Apesar do aumento da precisão, houve uma queda muito grande na acurácia, mostrando que o modelo não conseguiu generalizar tão bem em imagens com diferentes ângulos, qualidade e iluminação.

Tabela 7: Performance do *ViT* em outros conjuntos de dados

Conjunto de Dados	Acurácia	Precisão	Revocação
COR	0,826	0,863	0,778
EFD	0,731	0,892	0,714
Sazara	0,744	0,903	0,674

6 Conclusão

Com o objetivo de criar um modelo de classificação para classificar o estado de alagamento das ruas da cidade do Rio de Janeiro, esta pesquisa organizou um conjunto de dados original ao longo de meses de captação e classificação de imagens do sistema de câmeras do COR, disponibilizado [neste repositório de GitHub](#). Com este conjunto de dados, foi realizada uma comparação entre as arquiteturas *VGG*, *Inception*, *DenseNet*, *MobileNet* e *ViT* no problema de detecção de alagamentos para achar o melhor modelo entre arquiteturas selecionadas para alcançar o objetivo do trabalho.

Os resultados mostraram que o *ViT* superou as outras arquiteturas com diferentes taxas de aprendizagem. A alteração da intensidade luminosa da imagem teve impacto claro no desempenho do modelo e ajudou a diminuir a diferença da acurácia entre as imagens noturnas e diurnas, em comparação aos resultados descritos em Piedad *et al.* [60].

O modelo *ViT* se destacou usando *SGD* como otimizador e com taxa de aprendizagem de $1e^{-4}$. Foi o melhor não só pelas suas métricas mas pela sua maior estabilidade na curva de perda e pela menor discrepância entre acurácia de treino e validação.

Todos os modelos foram treinados utilizando imagens capturadas pelo sistema de câmeras COR, onde o *ViT* obteve acurácia de 83%. Este trabalho também analisou o modelo obtido em outras fontes de dados, o modelo conseguiu acurácia de 73% no conjunto de dados *EFD* [6], e de 74% no conjunto de dados de Sazara *et al.* [68].

Ao abordar o complexo problema de classificar imagens de alagamento de uma cidade grande e diversa como o Rio de Janeiro, este trabalho conseguiu alcançar seu objetivo de concluir uma abordagem inicial para a classificação automática da situação de alagamento de ruas, através de um conjunto de dados organizado e disponibilizado por esta dissertação baseado na própria cidade do Rio.

6.1 Trabalhos futuros

Nesta dissertação foi utilizado em todo o conjunto de treino o fator de claridade para diminuir o efeito de claridade excessiva em algumas imagens. Buscar uma abordagem que afete somente as imagens que possuam esse problema, e até mesmo somente na região com tal claridade.

A integração de diferentes conjuntos de dados ao treinamento do modelo pode impactar positivamente seu desempenho. Embora diferentes quantidades de tráfego tenham sido utilizadas no treinamento, outros eventos que podem ocorrer na rua ou ao seu redor, como manutenção das vias ou construções, não foram incluídos no conjunto de dados de treinamento. Portanto, essas situações poderiam ser classificadas incorretamente pelo modelo, e imagens desses eventos deveriam ser incluídas no conjunto de dados de treinamento.

Apesar do otimizador *SGD* inicialmente escolhido apresentar melhor desempenho, os resultados com o *NADAM* mostram que mais estudos sobre a escolha de diferentes valores para os parâmetros do otimizador podem aprimorar o modelo atual.

Além disso, alterações na arquitetura, como a criação de novas camadas demonstrado em Agung *et al.* [1], pode melhorar o desempenho do modelo uma vez que eles alcançaram uma precisão de 0,95 para o *MobileNetV2*, enquanto este trabalho alcançou uma precisão mais baixa de 0,78 com a mesma arquitetura.

O uso de outros instrumentos meteorológicos de medição, como pluviômetros instalados por toda a cidade, pode ser usado para corroborar as classificações resultantes do modelo gerado. Modelos baseados em imagens de satélites seriam importantes para uma maior eficácia de previsões. Outros aspectos importantes a serem incluídos são dados ligados ao histórico de regiões de alagamentos e a própria geografia da cidade, onde lugares próximos de encostas e mais baixos são as regiões que sabidamente devem ser mais monitoradas. No entanto, como o escopo da pesquisa foi o desenvolvimento de um modelo de classificação e algumas ruas não possuem sensores de nível de água, esse tipo de análise foi deixada de lado para uma abordagem inicial.

REFERÊNCIAS

- 1 AGUNG, Andi Sadri et al. Urban Flood Disaster Mitigation through Image Classification Using Transfer Learning Method with MobileNet Fine-tuning. In: 2023 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA). [S. l.: s. n.], 2023. Pp. 364–369. DOI: [10.1109/ICAMIMIA60881.2023.10427560](https://doi.org/10.1109/ICAMIMIA60881.2023.10427560).
- 2 AHMED, Imran et al. An Internet of Things and AI-Powered Framework for Long-Term Flood Risk Evaluation. **IEEE Internet of Things Journal**, v. 11, n. 3, pp. 3812–3819, 2024. DOI: [10.1109/JIOT.2023.3308564](https://doi.org/10.1109/JIOT.2023.3308564).
- 3 AKSHYA, J.; PRIYADARSINI, P.L.K. A Hybrid Machine Learning Approach for Classifying Aerial Images of Flood-Hit Areas. In: 2019 International Conference on Computational Intelligence in Data Science (ICCIDS). [S. l.: s. n.], 2019. Pp. 1–5. DOI: [10.1109/ICCIDS.2019.8862138](https://doi.org/10.1109/ICCIDS.2019.8862138).
- 4 ALHADY, M. Athallah Dzikri et al. Comparative Performance of Water Index for Water Segmentation Model Using U-Net Architecture. In: 2024 International Conference on Computer, Control, Informatics and its Applications (IC3INA). [S. l.: s. n.], 2024. Pp. 84–88. DOI: [10.1109/IC3INA64086.2024.10732848](https://doi.org/10.1109/IC3INA64086.2024.10732848).
- 5 ARORA, Gautam; R M, Bhavadharini. Disaster Scene Classification with Deep Learning: A Keras-Based Approach Utilizing Robotic Systems. In: 2024 IEEE Students Conference on Engineering and Systems (SCES). [S. l.: s. n.], 2024. Pp. 1–6. DOI: [10.1109/SCES61914.2024.10652476](https://doi.org/10.1109/SCES61914.2024.10652476).
- 6 BARZ, Björn et al. Enhancing Flood Impact Analysis using Interactive Retrieval of Social Media Images. Versão inglesa. **Archives of Data Science, Series A (Online First)**, v. 5, n. 1, a06, 21 s. online, 2018. ISSN 2363-9881. DOI: [10.5445/KSP/1000087327/06](https://doi.org/10.5445/KSP/1000087327/06).
- 7 BISHOP, Christopher. **Pattern Recognition and Machine Learning**. [S. l.]: Springer, 2007. ISBN 9780387310732.

- 8 BOUCHELKIA, Lina; TAHRAOUI, Ahmed; KHEDDAM, Radja. Post-Flood Disaster Mapping Using Deep Neural Network. In: 2024 IEEE Mediterranean and Middle-East Geoscience and Remote Sensing Symposium (M2GARSS). [S. l.: s. n.], 2024. Pp. 182–186. DOI: [10.1109/M2GARSS57310.2024.10537322](https://doi.org/10.1109/M2GARSS57310.2024.10537322).
- 9 BREIMAN, Leo. Random Forests. **Machine Learning**, v. 45, n. 1, pp. 5–32, 2001. ISSN 0885-6125. DOI: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324). Disponível em: <http://dx.doi.org/10.1023/A%3A1010933404324>.
- 10 CANNY, John. A Computational Approach to Edge Detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PAMI-8, n. 6, pp. 679–698, 1986. DOI: [10.1109/TPAMI.1986.4767851](https://doi.org/10.1109/TPAMI.1986.4767851).
- 11 CAO, Jingmiao et al. A Novel Wetland Classification Method Combined CNN and SVM Using Multi-Source Remote Sensing Images. In: IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium. [S. l.: s. n.], 2024. Pp. 4793–4796. DOI: [10.1109/IGARSS53475.2024.10640705](https://doi.org/10.1109/IGARSS53475.2024.10640705).
- 12 CHANDRAN, Divya V et al. Deep Learning-Based Flood Detection System Using Semantic Segmentation. In: 2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT). [S. l.: s. n.], 2024. v. 1, pp. 1584–1592. DOI: [10.1109/ICCPCT61902.2024.10673148](https://doi.org/10.1109/ICCPCT61902.2024.10673148).
- 13 CHEN, Liang-Chieh et al. **DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs**. [S. l.: s. n.], 2017. arXiv: [1606.00915 \[cs.CV\]](https://arxiv.org/abs/1606.00915). Disponível em: <https://arxiv.org/abs/1606.00915>.
- 14 CHEN, Tianqi; GUESTRIN, Carlos. XGBoost: A Scalable Tree Boosting System. In: PROCEEDINGS of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, California, USA: Association for Computing Machinery, 2016. (KDD '16), pp. 785–794. ISBN 9781450342322. DOI: [10.1145/2939672.2939785](https://doi.org/10.1145/2939672.2939785). Disponível em: <https://doi.org/10.1145/2939672.2939785>.
- 15 CHOLLET, Francois. **Deep Learning with Python**. [S. l.]: Manning Publications Co, 2019. ISBN 9781617294433.
- 16 CLARK, Alex. **Pillow (PIL Fork) Documentation**. [S. l.]: readthedocs, 2015. Disponível em: <https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>.

- 17 COPERNICUS. **Introducing Sentinel-1**. [S. l.: s. n.], 2025. Acessado em: 18 de Janeiro, 2025. Disponível em: <https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-1/Introducing_Sentinel-1>.
- 18 _____. **S2 Mission**. [S. l.: s. n.], 2025. Acessado em: 18 de Janeiro, 2025. Disponível em: <<https://sentiwiki.copernicus.eu/web/s2-mission>>.
- 19 COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, pp. 21–27, 1967. DOI: [10.1109/TIT.1967.1053964](https://doi.org/10.1109/TIT.1967.1053964).
- 20 CRAMER, J.S. **The Origins of Logistic Regression**. [S. l.], dez. 2002. Disponível em: <<https://ideas.repec.org/p/tin/wpaper/20020119.html>>.
- 21 DENG, Jia et al. Imagenet: A large-scale hierarchical image database. In: IEEE. 2009 IEEE conference on computer vision and pattern recognition. [S. l.: s. n.], 2009. Pp. 248–255.
- 22 DOERRY, Armin. Introduction to Synthetic Aperture Radar. In: 2019 IEEE Radar Conference (RadarConf). [S. l.: s. n.], 2019. Pp. 1–90. DOI: [10.1109/RADAR.2019.8835560](https://doi.org/10.1109/RADAR.2019.8835560).
- 23 DOGO, Eustace; AFOLABI, Oluwatobi; TWALA, Bhakisipho. On the Relative Impact of Optimizers on Convolutional Neural Networks with Varying Depth and Width for Image Classification. **Applied Sciences**, v. 12, p. 11976, nov. 2022. DOI: [10.3390/app122311976](https://doi.org/10.3390/app122311976).
- 24 DOSOVITSKIY, Alexey et al. **An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale**. [S. l.: s. n.], 2021. arXiv: [2010.11929](https://arxiv.org/abs/2010.11929) [cs.CV].
- 25 DOZAT, Timothy. Incorporating Nesterov Momentum into Adam. In: PROCEEDINGS of the 4th International Conference on Learning Representations. [S. l.: s. n.], 2016. Pp. 1–4.
- 26 EVGENIOU, Theodoros; PONTIL, Massimiliano. Support Vector Machines: Theory and Applications. In: v. 2049, pp. 249–257. ISBN 978-3-540-42490-1. DOI: [10.1007/3-540-44673-7_12](https://doi.org/10.1007/3-540-44673-7_12).

- 27 FAWCETT, Tom. An introduction to ROC analysis. **Pattern Recognition Letters**, v. 27, n. 8, pp. 861–874, 2006. ROC Analysis in Pattern Recognition. ISSN 0167-8655. DOI: <https://doi.org/10.1016/j.patrec.2005.10.010>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S016786550500303X>>.
- 28 GESMUNDO, Andrea. **A Continual Development Methodology for Large-scale Multitask Dynamic ML Systems**. [S. l.: s. n.], 2022. arXiv: [2209.07326 \[cs.LG\]](https://arxiv.org/abs/2209.07326).
- 29 GONZALEZ, Rafael C.; WOODS, Richard E. **Digital Image Processing**. 4. ed. [S. l.]: Pearson, 2018.
- 30 GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [S. l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.
- 31 HE, Kaiming et al. Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV). [S. l.: s. n.], 2017. Pp. 2980–2988. DOI: [10.1109/ICCV.2017.322](https://doi.org/10.1109/ICCV.2017.322).
- 32 HIDAYAT, Muh. Taufik et al. Enhanced Flood Detection on Highways: A Comparative Study of MobileNet and VGG16 CNN Models Based on CCTV Images. In: 2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA). [S. l.: s. n.], 2024. Pp. 125–130. DOI: [10.1109/ICSINTESA62455.2024.10747897](https://doi.org/10.1109/ICSINTESA62455.2024.10747897).
- 33 HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, pp. 1735–1780, nov. 1997. ISSN 0899-7667. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735). Disponível em: <<https://doi.org/10.1162/neco.1997.9.8.1735>>.
- 34 HOWARD, Andrew et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, abr. 2017. DOI: [10.48550/arXiv.1704.04861](https://doi.org/10.48550/arXiv.1704.04861).
- 35 HUANG, Binbin et al. WaterDetectionNet: A New Deep Learning Method for Flood Mapping With SAR Image Convolutional Neural Network. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, v. 17, pp. 14471–14485, 2024. DOI: [10.1109/JSTARS.2024.3440995](https://doi.org/10.1109/JSTARS.2024.3440995).
- 36 HUANG, Gao et al. Densely connected convolutional networks. In: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition. [S. l.: s. n.], 2017.

- 37 HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **The Journal of Physiology**, v. 160, n. 1, pp. 106–154, 1962. DOI: <https://doi.org/10.1113/jphysiol.1962.sp006837>. eprint: <https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1962.sp006837>. Disponível em: <<https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1962.sp006837>>.
- 38 IKOTUN, Abiodun M. et al. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. **Information Sciences**, v. 622, pp. 178–210, 2023. ISSN 0020-0255. DOI: <https://doi.org/10.1016/j.ins.2022.11.139>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0020025522014633>>.
- 39 ISLAM, Md Azharul et al. An integrated convolutional neural network and sorting algorithm for image classification for efficient flood disaster management. **Decision Analytics Journal**, v. 7, p. 100225, 2023. ISSN 2772-6622. DOI: <https://doi.org/10.1016/j.dajour.2023.100225>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2772662223000656>>.
- 40 ITSEEZ. **Open Source Computer Vision Library**. [S. l.: s. n.], 2015. <https://github.com/itseez/opencv>.
- 41 JAMUNADEVI, .C et al. Application of Deep Learning Algorithm for Prediction of Flood Severity. In: pp. 1637–1642. DOI: [10.1109/ICSCSS60660.2024.10625025](https://doi.org/10.1109/ICSCSS60660.2024.10625025).
- 42 JONY, Rabiul Islam; WOODLEY, Alan; PERRIN, Dimitri. Flood Detection in Social Media Images using Visual Features and Metadata. In: 2019 Digital Image Computing: Techniques and Applications (DICTA). [S. l.: s. n.], 2019. Pp. 1–8. DOI: [10.1109/DICTA47822.2019.8946007](https://doi.org/10.1109/DICTA47822.2019.8946007).
- 43 KABIR, H M Dipu. **Reduction of Class Activation Uncertainty with Background Information**. [S. l.: s. n.], 2023. arXiv: [2305.03238](https://arxiv.org/abs/2305.03238) [cs.CV].
- 44 KINGMA, Diederik; BA, Jimmy. Adam: A Method for Stochastic Optimization. **International Conference on Learning Representations**, dez. 2014.
- 45 KOLESNIKOV, Alexander et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In:
- 46 KOLESNIKOV, Alexander et al. **Big Transfer (BiT): General Visual Representation Learning**. [S. l.: s. n.], 2020. arXiv: [1912.11370](https://arxiv.org/abs/1912.11370) [cs.CV].

- 47 KOOL, Juliette et al. Seasonal inundation dynamics and water balance of the Mara Wetland, Tanzania based on multi-temporal Sentinel-2 image classification. **International Journal of Applied Earth Observation and Geoinformation**, v. 109, p. 102766, 2022. ISSN 1569-8432. DOI: <https://doi.org/10.1016/j.jag.2022.102766>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0303243422000927>>.
- 48 KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In_____. **Advances in Neural Information Processing Systems**. [S. l.]: Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.
- 49 LIM, Jae S. **Two-Dimensional Signal and Image Processing**. [S. l.]: Prentice-Hall, 1990.
- 50 LIU, Haiyang et al. A Comparison of Different Water Indices and Band Downscaling Methods for Water Bodies Mapping from Sentinel-2 Imagery at 10-M Resolution. **Water**, v. 14, n. 17, 2022. ISSN 2073-4441. DOI: [10.3390/w14172696](https://doi.org/10.3390/w14172696). Disponível em: <<https://www.mdpi.com/2073-4441/14/17/2696>>.
- 51 MAO, Anqi; MOHRI, Mehryar; ZHONG, Yutao. Cross-entropy loss functions: theoretical analysis and applications. In: PROCEEDINGS of the 40th International Conference on Machine Learning. Honolulu, Hawaii, USA: JMLR.org, 2023. (ICML'23).
- 52 MICHELIN. **What is Aquaplaning and how to avoid it?** [S. l.: s. n.], 2024. Acessado em: 02 de Novembro, 2024. Disponível em: <<https://www.michelin.co.uk/auto/advice/driving-tips/aquaplaning>>.
- 53 MULTIMEDIA EVALUATION, MediaEval Benchmarking Initiative for. **The 2017 Multimedia Satellite Task: Emergency Response for Flooding Events**. [S. l.: s. n.], 2017. Acessado em: 02 de Novembro, 2024. Disponível em: <<http://www.multimediaeval.org/mediaeval2017/multimediasatellite/>>.
- 54 MURPHY, Kevin P. **Machine Learning: A Probabilistic Perspective**. 1. ed. [S. l.]: The MIT Press, 2012. (Adaptive Computation and Machine Learning).
- 55 MURTAGH, Fionn. Multilayer perceptrons for classification and regression. **Neurocomputing**, v. 2, n. 5, pp. 183–197, 1991. ISSN 0925-2312. DOI: [https://doi.org/10.1016/0925-2312\(91\)90023-5](https://doi.org/10.1016/0925-2312(91)90023-5). Disponível em: <<https://www.sciencedirect.com/science/article/pii/0925231291900235>>.

- 56 MYIN, Moe Moe; THEIN, Thin Lai Lai. Flood Mapping System of Flooding Prone Area in Myanmar. In: 2024 IEEE Conference on Computer Applications (ICCA). [S. l.: s. n.], 2024. Pp. 1–5. DOI: [10.1109/ICCA62361.2024.10532869](https://doi.org/10.1109/ICCA62361.2024.10532869).
- 57 OHTA, Noboru; ROBERTSON, Alan. **Colorimetry**: Fundamentals and Applications. 1. ed. [S. l.]: Joh Wiley & Sons, 2005.
- 58 PALLY, R.J.; SAMADI, S. Application of image processing and convolutional neural networks for flood image classification and semantic segmentation. **Environmental Modelling and Software**, v. 148, p. 105285, 2022. ISSN 1364-8152. DOI: <https://doi.org/10.1016/j.envsoft.2021.105285>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1364815221003273>.
- 59 PECH-MAY, Fernando et al. Segmentation and Visualization of Flooded Areas Through Sentinel-1 Images and U-Net. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, v. 17, pp. 8996–9008, 2024. DOI: [10.1109/JSTARS.2024.3387452](https://doi.org/10.1109/JSTARS.2024.3387452).
- 60 PIEDAD, Eduardo Jr et al. Intelligent Flood Detection using Traffic Surveillance Images based on Convolutional Neural Network and Image Parsing. In: 2022 IEEE International Conference on Computing (ICOCO). [S. l.: s. n.], 2022. Pp. 220–225. DOI: [10.1109/ICOC056118.2022.10031718](https://doi.org/10.1109/ICOC056118.2022.10031718).
- 61 RAO, Chengping; LIU, Yang. Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization. **Computational Materials Science**, v. 184, p. 109850, 2020. ISSN 0927-0256. DOI: <https://doi.org/10.1016/j.commatsci.2020.109850>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0927025620303414>.
- 62 REDDY, K.Venkata Vinay Kumar; VIMAL, V.R. A Novel Approach on Improved Segmentation and Classification of Remote Sensing Images using AlexNet Compared over Linear Discriminant Analysis with Improved Accuracy. In: 2024 Second International Conference on Advances in Information Technology (ICAIT). [S. l.: s. n.], 2024. v. 1, pp. 1–6. DOI: [10.1109/ICAIT61638.2024.10690378](https://doi.org/10.1109/ICAIT61638.2024.10690378).
- 63 REZATOFIGHI, Hamid et al. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA, USA: IEEE Computer Society, jun. 2019. Pp. 658–666. DOI: [10.1109/CVPR.2019.00075](https://doi.org/10.1109/CVPR.2019.00075). Disponível em: <https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00075>.

- 64 RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: NAVAB, Nassir et al. (Ed.). **Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015**. Cham: Springer International Publishing, 2015. Pp. 234–241. ISBN 978-3-319-24574-4.
- 65 RUDER, Sebastian. An overview of gradient descent optimization algorithms, set. 2016. DOI: [10.48550/arXiv.1609.04747](https://arxiv.org/abs/1609.04747).
- 66 RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. **Nature**, v. 323, pp. 533–536, 1986. Disponível em: <<https://api.semanticscholar.org/CorpusID:205001834>>.
- 67 SANDLER, Mark et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S. l.: s. n.], 2018. Pp. 4510–4520. DOI: [10.1109/CVPR.2018.00474](https://doi.org/10.1109/CVPR.2018.00474).
- 68 SAZARA, Cem; CETIN, Mecit; IFTEKHARUDDIN, Khan M. Detecting floodwater on roadways from image data with handcrafted features and deep transfer learning. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). [S. l.: s. n.], 2019. Pp. 804–809. DOI: [10.1109/ITSC.2019.8917368](https://doi.org/10.1109/ITSC.2019.8917368).
- 69 SCHMIDHUBER, Jürgen. Deep learning in neural networks: An overview. **Neural Networks**, v. 61, pp. 85–117, 2015. ISSN 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2014.09.003>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0893608014002135>>.
- 70 SGHAIER, Souhir et al. Natural disasters detection and classification based on deep learning. In: 2023 3rd International Conference on Computing and Information Technology (ICCIT). [S. l.: s. n.], 2023. Pp. 334–339. DOI: [10.1109/ICCIT58132.2023.10273976](https://doi.org/10.1109/ICCIT58132.2023.10273976).
- 71 SHAH, Aditya et al. Machine Learning for Flood Susceptibility Mapping of the Chennai Floods of 2023. In: pp. 1–6. DOI: [10.1109/SPICES62143.2024.10779868](https://doi.org/10.1109/SPICES62143.2024.10779868).
- 72 SHAPIRO, Linda; STOCKMAN, George. **Computer Vision**. 1. ed. [S. l.]: Pearson, 2001.
- 73 SIDDIQUE, Mohammed; AHMED, Tasneem; HUSAIN, Mohd Shahid. An Integrated Image Classification Approach to Detect the Flood Prone Areas using Sentinel-1 Images. In: 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom). [S. l.: s. n.], 2023. Pp. 655–660.

- 74 SIMONYAN, K; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: pp. 1–14.
- 75 SUDIANA, Dodi et al. Performance Evaluation of 3-D Convolutional Neural Network for Multitemporal Flood Classification Framework With Synthetic Aperture Radar Image Data. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, v. 18, pp. 3198–3207, 2025. DOI: [10.1109/JSTARS.2024.3519523](https://doi.org/10.1109/JSTARS.2024.3519523).
- 76 SZEGEDY, Christian et al. Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA, USA: IEEE Computer Society, jun. 2015. Pp. 1–9. DOI: [10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594). Disponível em: [10.1109/CVPR.2015.7298594](https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594).
- 77 SZEGEDY, Christian et al. Rethinking the Inception Architecture for Computer Vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA, USA: IEEE Computer Society, jun. 2016. Pp. 2818–2826. DOI: [10.1109/CVPR.2016.308](https://doi.org/10.1109/CVPR.2016.308). Disponível em: [10.1109/CVPR.2016.308](https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.308).
- 78 THARWAT, Alaa et al. Linear discriminant analysis: A detailed tutorial. **Ai Communications**, v. 30, pp. 169–190, mai. 2017. DOI: [10.3233/AIC-170729](https://doi.org/10.3233/AIC-170729).
- 79 TOUFIQUE, S.M et al. Implementing Machine Learning Techniques to Forecast Floods in Bangladesh. In: pp. 1–6. DOI: [10.1109/ICECET61485.2024.10698703](https://doi.org/10.1109/ICECET61485.2024.10698703).
- 80 VAN ROSSUM, Guido; DRAKE JR, Fred L. **Python tutorial**. [S. l.]: Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.
- 81 VASWANI, Ashish et al. Attention is All you Need. In_____. **Advances in Neural Information Processing Systems**. [S. l.]: Curran Associates, Inc., 2017. v. 30. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- 82 VIMALA, .M et al. Flood Prediction using Supervised Machine Learning Algorithms. In: pp. 1093–1096. DOI: [10.1109/IC0SEC61587.2024.10722348](https://doi.org/10.1109/IC0SEC61587.2024.10722348).
- 83 VINEETH, V; NEEBA, E A. Flood Detection using Deep Learning. In: 2021 International Conference on Advances in Computing and Communications (ICACC). [S. l.: s. n.], 2021. Pp. 1–5. DOI: [10.1109/ICACC-202152719.2021.9708240](https://doi.org/10.1109/ICACC-202152719.2021.9708240).

- 84 XU, Hanqiu. Modification of normalised difference water index (NDWI) to enhance open water features in remotely sensed imagery. **International Journal of Remote Sensing**, Taylor & Francis, v. 27, n. 14, pp. 3025–3033, 2006. DOI: [10.1080/01431160600589179](https://doi.org/10.1080/01431160600589179). eprint: <https://doi.org/10.1080/01431160600589179>. Disponível em: [<https://doi.org/10.1080/01431160600589179>](https://doi.org/10.1080/01431160600589179).