

**CAROLINA MARI MIYASHIRO  
PEDRO GABRIEL MASCARENHAS MARONEZI**

**SEASTATEFY: FERRAMENTA PARA  
ESTIMAÇÃO DO ESTADO DO MAR A PARTIR  
DO PROCESSAMENTO DE IMAGENS**

São Paulo  
2022

**CAROLINA MARI MIYASHIRO  
PEDRO GABRIEL MASCARENHAS MARONEZI**

**SEASTATEFY: FERRAMENTA PARA  
ESTIMAÇÃO DO ESTADO DO MAR A PARTIR  
DO PROCESSAMENTO DE IMAGENS**

Trabalho apresentado à Escola Politécnica  
da Universidade de São Paulo para  
obtenção do Título de Engenheiro Enge-  
nharia Elétrica com ênfase em Computação.

São Paulo  
2022

**CAROLINA MARI MIYASHIRO  
PEDRO GABRIEL MASCARENHAS MARONEZI**

**SEASTATEFY: FERRAMENTA PARA  
ESTIMAÇÃO DO ESTADO DO MAR A PARTIR  
DO PROCESSAMENTO DE IMAGENS**

Trabalho apresentado à Escola Politécnica  
da Universidade de São Paulo para  
obtenção do Título de Engenheiro Enge-  
nharia Elétrica com ênfase em Computação.

Orientador:

Jaime Simão Sichman

Co-orientador:

Eduardo Aoun Tannuri

São Paulo  
2022

Aos nossos pais.

# AGRADECIMENTOS

Agradecemos aos nossos pais que nos acompanharam nesta longa jornada e nos apoiaram nos momentos difíceis, dando suporte, sempre presentes com palavras de encorajamento e força, nos mostrando o caminho a seguir.

Agradecemos aos nossos familiares e amigos que com seu incentivo nos fizeram chegar à conclusão do curso e começo de uma nova carreira.

Agradecemos ao nosso orientador Professor Doutor Jaime Simão Sichman sem cujo incentivo e apoio constantes esse trabalho não teria sido realizado.

Agradecemos ao nosso co-orientador Professor Doutor Eduardo Aoun Tannuri que nos orientou, nos introduziu a um novo campo de pesquisa, facilitou nosso planejamento e apoiou durante nossa jornada de criação do projeto com ideias e conselhos.

Por fim, agradecemos à Escola Politécnica pelo ambiente de aprendizado constante, onde pudemos nos desenvolver como pessoas e como profissionais, por onde fizemos novas amizades e parcerias que perdurarão para além de nosso tempo na universidade.

*“Nunca ande por trilhas, pois assim só  
irá até onde outros já foram”*

-- Alexander Graham Bell

# RESUMO

O estado do mar (*sea state*) é uma medida da agitação da superfície do mar (ondas) usada em diferentes contextos: segurança das operações de plataformas de petróleo e navios de perfuração e exploração oceânica; segurança costeira, prevenção de catástrofes; dentre outros. Atualmente, os métodos para classificar os estados do mar são baseados na representação estatística de parâmetros de onda ou por modelagem numérica. Esses métodos são caros, propensos a mau funcionamento do equipamento e exigem alto poder de computação e tempo. Desta forma, o objetivo estabelecido para o projeto foi a criação de uma ferramenta de classificação do estado do mar utilizando *Deep Learning*. Denominada ***SeaStatefy***, a plataforma é capaz de associar uma imagem da superfície do mar ao intervalo da Escala Beaufort mais semelhante. A partir dos conjuntos de imagens obtidos, experimentou-se diferentes arquiteturas de redes neurais convolucionais (CNNs), aplicando-se técnicas para tratar de *datasets* desbalanceados. O modelo final escolhido foi aquele que apresentou o melhor desempenho nos conjuntos de testes. A partir dele, desenvolveu-se uma plataforma que disponibilizasse a ferramenta para o usuário, permitindo o *upload* de uma imagem da superfície do mar e a identificação de seu respectivo estado do mar. De modo geral, obteve-se um modelo classificatório do estado do mar com uma acurácia superior a 90%, apesar de possuir algumas restrições quanto à generalização para imagens muito diferentes. A plataforma ***SeaStatefy*** resultou em uma interface simples, fácil e rápida de ser utilizada, podendo classificar imagens em poucos segundos e contendo informações sobre o processo de desenvolvimento do classificador a fim de atribuir maior confiabilidade aos resultados por ela produzidos.

**Palavras-Chave** – Estado do mar, *Deep Learning*, Escala Beaufort, Redes Neurais Convolucionais, Plataforma, Classificador de imagens.

# ABSTRACT

The sea state describes the degree of turbulence at the sea surface (waves) and is used in different contexts: safety of operations of oil platforms and vessels for drilling and ocean exploration; coastal security, disaster prevention; and more. Currently, sea state classification methods are based on statistical representation of wave parameters or numerical modeling. These methods are expensive, prone to equipment failure, and demand high computing power and time. Hence, this project proposed the creation of a sea state classification tool using Deep Learning. Named *SeaStatefy*, the platform is capable of associating an image of the sea surface with the most probable Beaufort scale interval. Using the sets of images obtained, different architectures of convolutional neural networks (CNNs) were experimented, and techniques to deal with unbalanced data sets were applied. The final model chosen was the one that presented the best performance in the test sets. Finally, to make the tool available to the user, a platform which allows the upload of an image of the sea surface and the identification of its respective sea state was developed. Overall, the sea state classification model obtained an accuracy greater than 90%, despite having some restrictions regarding generalization to very different images. The *SeaStatefy* platform implemented was a simple, easy and quick interface to use, being able to classify images in a few seconds and containing information about the classifier development process in order to provide greater reliability to the results produced by it.

**Keywords** – Sea State, Deep Learning, Beaufort scale, Convolutional Neural Networks, Plataform, Image classifier.



# LISTA DE FIGURAS

1	Exemplo de uma boia projetada para uso como plataforma de instrumentação meteorológica e oceanográfica. . . . .	17
2	Representação de uma imagem RGB em 3 camadas/canais (vermelho, verde e azul). Cada número representa a intensidade do pixel para cada uma das cores . . . . .	26
3	Representação da convolução de um kernel 3 x 3 com uma imagem 5 x 5 e o resultado desta operação. Para o campo receptivo da imagem (representado em laranja) o resultado é 4 . . . . .	27
4	Representação do deslocamento de um <i>kernel</i> 3 x 3 em uma imagem se movendo da esquerda para a direita, de cima para baixo . . . . .	27
5	Representação da adição de uma camada de <i>padding</i> de zeros à matriz . . . . .	28
6	Aplicação de um <i>max pooling</i> e de um <i>average pooling</i> em uma matriz 4 x 4 com um campo receptivo 2 x 2 e um deslocamento de 2 . . . . .	29
7	Representação de uma arquitetura geral de uma CNN . . . . .	29
8	Arquitetura da VGG-16 . . . . .	31
9	Bloco residual . . . . .	32
10	Arquitetura da ResNet-34 . . . . .	33
11	Bloco Residual <i>Bottleneck</i> . . . . .	34
12	Matriz de confusão para as classes positivo e negativo da perspectiva da Classe A . . . . .	35
13	Arquitetura da ResNet34-VGG . . . . .	53
14	Influência das dimensões da entrada no reconhecimento de padrões . . . . .	54
15	Gradiente descendente com uma taxa de aprendizado pequena (esquerda) e grande (direita). . . . .	57
16	Representação do <i>transfer learning</i> com parâmetros de extração de <i>feature</i> congelados . . . . .	59

17	Imagens representantes dos graus da escala Beaufort . . . . .	63
18	Arquitetura em camadas do projeto com front-end e back-end. . . . .	66
19	Operação de <i>center crop</i> em uma imagem de dimensões 1280x720 pixels, obtendo uma imagem final de dimensões 720x720 pixels. . . . .	67
20	Exemplo de um JSON retornado pela API para uma imagem da classe 3. . . . .	68
21	Fluxograma do funcionamento da API ao receber uma chamada. . . . .	69
22	Prototipagem da tela de Início usando o Figma . . . . .	71
23	Prototipagem da tela de Ferramenta usando o Figma . . . . .	72
24	Prototipagem da tela de resultados usando o Figma . . . . .	72
25	Prototipagem da tela de Saiba Mais usando o Figma . . . . .	73
26	<i>Home</i> da plataforma SeaStatefy . . . . .	75
27	Página da ferramenta da plataforma SeaStatefy . . . . .	76
28	Página da ferramenta da plataforma SeaStatefy após carregamento da imagem . . . . .	76
29	Exemplo de requisição HTTP utilizada para envio da imagem codificada ao back-end . . . . .	77
30	Página da ferramenta da plataforma SeaStatefy com os resultados da classificação . . . . .	77
31	Página de “saiba mais” da plataforma SeaStatefy . . . . .	78
32	Imagens representantes da gravação YJ (esquerda) e da gravação BS 4 (direita) . . . . .	84
33	Matriz de confusão do modelo <code>resnet34_pretrained</code> . As linhas representam as classes verdadeiras ( <i>True label</i> ) e as colunas representam as classes previstas pelo modelo ( <i>Predicted label</i> ) . . . . .	84
34	Matriz de confusão do modelo <code>vgg16_pretrained</code> nas imagens representantes da escala Beaufort. As linhas representam as classes verdadeiras ( <i>True label</i> ) e as colunas representam as classes previstas pelo modelo ( <i>Predicted label</i> ) . . . . .	87

35	Matriz de confusão do modelo <code>resnet34_oversample</code> nas imagens representantes da escala Beaufort. As linhas representam as classes verdadeiras ( <i>True label</i> ) e as colunas representam as classes previstas pelo modelo ( <i>Predicted label</i> ) . . . . .	88
36	Matriz de confusão do modelo <code>resnet152_pretrained</code> nas imagens representantes da escala Beaufort. As linhas representam as classes verdadeiras ( <i>True label</i> ) e as colunas representam as classes previstas pelo modelo ( <i>Predicted label</i> ) . . . . .	88
37	Matriz de confusão do modelo <code>resnet34_weighted_loss</code> nas imagens representantes da escala Beaufort. As linhas representam as classes verdadeiras ( <i>True label</i> ) e as colunas representam as classes previstas pelo modelo ( <i>Predicted label</i> ) . . . . .	89
38	Matriz de confusão do modelo <code>resnet34_oversample</code> treinado utilizando imagens do <i>dataset Stereo</i> junto do <i>dataset</i> MU-SSiD nas imagens do <i>dataset Stereo</i> . As linhas representam as classes verdadeiras ( <i>True label</i> ) e as colunas representam as classes previstas pelo modelo ( <i>Predicted label</i> ) . . .	92
39	Matriz de confusão do modelo <code>resnet34_oversample</code> treinado utilizando imagens do <i>dataset Stereo</i> junto do <i>dataset</i> MU-SSiD nas imagens representantes da escala Beaufort. As linhas representam as classes verdadeiras ( <i>True label</i> ) e as colunas representam as classes previstas pelo modelo ( <i>Predicted label</i> ) . . . . .	92
40	Matriz de confusão do modelo <code>resnet34_oversample</code> treinado utilizando imagens do <i>dataset Stereo</i> junto do <i>dataset</i> MU-SSiD nas imagens do <i>dataset</i> MU-SSiD. As linhas representam as classes verdadeiras ( <i>True label</i> ) e as colunas representam as classes previstas pelo modelo ( <i>Predicted label</i> )	93
41	Variação da acurácia de treinamento e validação do modelo <code>resnet34_oversample</code> treinado utilizando imagens do <i>dataset Stereo</i> junto do <i>dataset</i> MU-SSiD durante as épocas. . . . .	94
42	Variação do erro ( <i>loss</i> ) de treinamento e validação do modelo <code>resnet34_oversample</code> treinado utilizando imagens do <i>dataset Stereo</i> junto do <i>dataset</i> MU-SSiD durante as épocas. . . . .	95

# LISTA DE TABELAS

1	Estudos para classificação do estado do mar . . . . .	18
2	Estudos utilizados para a obtenção de <i>datasets</i> do estado do mar . . . . .	19
3	Especificações da escala Beaufort atual . . . . .	37
4	Especificações dos dados do <i>dataset Stereo</i> . . . . .	50
5	Especificações dos dados MU-SSiD . . . . .	51
6	Técnicas de <i>data augmentation</i> utilizadas em cada catálogo . . . . .	55
7	Frequência por classe . . . . .	56
8	Hiper parâmetros selecionados . . . . .	57
9	Especificações técnicas da máquina do TPN . . . . .	60
10	Modelos resultantes para cada cenário e arquitetura . . . . .	61
11	Modelos resultantes para cada cenário e arquitetura utilizando as novas imagens de MU-SSiD no treinamento . . . . .	62
12	Melhor modelo de cada arquitetura . . . . .	64
13	Intervalo de tempo decorrido para codificação, processamento e classi- ficação de uma imagem . . . . .	79
14	Acurácia por modelo da arquitetura VGG-16 para cada gravação do <i>dataset Stereo</i> . . . . .	82
15	Acurácia por modelo da arquitetura ResNet-34 para cada gravação do <i>da- taset Stereo</i> . . . . .	82
16	Acurácia por modelo da arquitetura ResNet-152 para cada gravação do <i>dataset Stereo</i> . . . . .	83
17	Acurácia por modelo da arquitetura ResNet34-VGG para cada gravação do <i>dataset Stereo</i> . . . . .	83
18	Acurácia geral por modelo para o <i>dataset Stereo</i> . . . . .	85

19	Acurácia geral por modelo para as imagens representantes da escala Beaufort e para o <i>dataset Stereo</i> . . . . .	86
20	Acurácia geral por modelo para as imagens representantes da escala Beaufort e para o <i>dataset Stereo</i> para os 4 melhores modelos . . . . .	87
21	Acurácia geral por modelo para as imagens do <i>dataset MU-SSiD</i> . . . . .	90
22	Acurácia geral por modelo para as imagens do <i>dataset MU-SSiD</i> , para as imagens representantes da escala Beaufort e para o <i>dataset Stereo</i> para os 4 melhores modelos treinados com imagens do <i>dataset Stereo</i> . . . . .	90
23	Acurácia geral por modelo para as imagens do <i>dataset MU-SSiD</i> , para as imagens representantes da escala Beaufort e para o <i>dataset Stereo</i> para os 4 melhores modelos treinados com imagens do <i>dataset Stereo</i> junto do <i>dataset MU-SSiD</i> . . . . .	91
24	Métricas para o modelo <code>resnet34_oversample</code> treinado utilizando imagens do <i>dataset Stereo</i> junto do <i>dataset MU-SSiD</i> nas imagens do <i>dataset Stereo</i>	93
25	Métricas para o modelo <code>resnet34_oversample</code> treinado utilizando imagens do <i>dataset Stereo</i> junto do <i>dataset MU-SSiD</i> nas imagens representantes da escala Beaufort . . . . .	93
26	Métricas para o modelo <code>resnet34_oversample</code> treinado utilizando imagens do <i>dataset Stereo</i> junto do <i>dataset MU-SSiD</i> nas imagens do <i>dataset MU-SSiD</i>	93

# SUMÁRIO

<b>Parte I: INTRODUÇÃO</b>	<b>15</b>
<b>1 Introdução</b>	<b>16</b>
1.1 Motivação . . . . .	18
1.2 Objetivo . . . . .	19
1.3 Justificativa . . . . .	21
1.4 Organização do trabalho . . . . .	22
<b>Parte II: DESENVOLVIMENTO</b>	<b>23</b>
<b>2 Aspectos Conceituais</b>	<b>24</b>
2.1 Inteligência Artificial . . . . .	24
2.2 Redes Neurais Convolucionais . . . . .	25
2.2.1 VGG-16 . . . . .	30
2.2.2 ResNet . . . . .	31
2.3 Métricas de desempenho . . . . .	34
2.4 Escala Beaufort . . . . .	36
<b>3 Metodologia de Trabalho</b>	<b>41</b>
<b>4 Especificação de Requisitos</b>	<b>43</b>
4.1 Requisitos do classificador . . . . .	43
4.2 Requisitos da plataforma . . . . .	44
4.2.1 Requisitos funcionais . . . . .	44
4.2.2 Requisitos não-funcionais . . . . .	44

<b>5</b>	<b>Tecnologias Utilizadas</b>	<b>46</b>
5.1	Docker . . . . .	46
5.2	Python . . . . .	46
5.3	PyTorch . . . . .	47
5.4	Heroku . . . . .	47
5.5	SciKit Learn . . . . .	47
5.6	NumPy . . . . .	48
5.7	Google Colab . . . . .	48
5.8	React . . . . .	48
5.9	FastAPI . . . . .	48
<b>6</b>	<b>Projeto e Implementação</b>	<b>49</b>
6.1	Aquisição do dataset . . . . .	49
6.2	Estudo de modelos . . . . .	51
6.3	Seleção de candidatos a modelo . . . . .	51
6.4	Pré-processamento das imagens . . . . .	53
6.5	Treinamento . . . . .	55
6.5.1	Técnicas para tratar <i>datasets</i> desbalanceados . . . . .	56
6.5.2	Escolha de hiper parâmetros de treinamento . . . . .	56
6.5.3	Desenvolvimento dos modelos . . . . .	58
6.6	Testes . . . . .	62
6.6.1	Modelos treinados no <i>dataset Stereo</i> . . . . .	62
6.6.2	Modelos treinados nos <i>datasets Stereo</i> e <i>MU-SSiD</i> . . . . .	64
6.7	Implementação da plataforma . . . . .	65
6.7.1	Back-end . . . . .	66
6.7.2	Front-end . . . . .	69
6.7.3	Integração e plataforma final . . . . .	74

<b>Parte III: EXPERIMENTOS E CONCLUSÃO</b>	<b>80</b>
<b>7 Resultados</b>	<b>81</b>
7.1 Modelos treinados no <i>dataset Stereo</i> . . . . .	81
7.1.1 Imagens do <i>dataset Stereo</i> . . . . .	81
7.1.2 Imagens representantes da escala Beaufort . . . . .	85
7.1.3 Imagens do <i>dataset MU-SSiD</i> . . . . .	89
7.2 Modelos treinados nos <i>datasets Stereo</i> e <i>MU-SSiD</i> . . . . .	90
<b>8 Conclusões</b>	<b>96</b>
8.1 Considerações finais . . . . .	96
8.2 Trabalhos Futuros . . . . .	97
<b>Referências</b>	<b>99</b>



# PARTE I

## INTRODUÇÃO

# 1 INTRODUÇÃO

O mar agrega diversos tipos de atividade de diferentes esferas socioeconômicas, tais como a produção de gás e petróleo, o transporte, o turismo, dentre outros, trazendo grande impacto na economia global. Por exemplo, o setor *offshore* de petróleo e gás produz 37% e 28% da participação global de petróleo e gás, respectivamente (LEGORBURU; JOHNSON; KERR, 2018), a indústria de transporte marítimo facilita mais de 80% do comércio global em volume (ASARIOTIS et al., 2018) e espera-se que o turismo marinho e costeiro contribua com 26% para a economia oceânica global em 2030 (LEPOSA, 2020).

Neste contexto, para todas estas atividades, as condições da superfície do mar são influenciadas pelas condições climáticas. O **estado do mar** (*sea state*) é uma medida da agitação da superfície do mar (ondas) e é usado para definir a segurança de uma navio em atravessar determinado trecho do oceano em navegação; para definir a segurança das operações de plataformas de petróleo e navios de perfuração e exploração oceânica, e na costa; para quantificar a possibilidade de agitação nas praias, ressacas, alagamentos; dentre outros. O **estado do mar** é definido pela força do vento, pela distância de busca de águas abertas disponível para geração de ondas e a duração de tempo durante a qual as ondas podem acumular energia do vento (THOMSON et al., 2016). Desta forma, é resultado da altura da onda, do período da onda e velocidade do vento.

As condições do estado do mar, em especial a altura das ondas, têm impactado, por exemplo, as estruturas *offshore* de petróleo e gás (VANNAK; LIEW; YEW, 2013) e até mesmo, a depender da magnitude, podem ter impacto sobre a costa, sendo seu conhecimento de interesse da guarda costeira para a prevenção de catástrofes.

Atualmente, a medição e a classificação dos estados do mar são realizadas através da apresentação estatística de parâmetros das ondas, tais quais a altura significativa de onda ( $H_s$ ) e o período médio de travessia de zero ( $T_z$ ) (LAINING et al., 1998). A aquisição destes parâmetros é realizada de várias fontes tradicionais, como boias marítimas, radares de ondas e observações de satélite (UMAIR; HASHMANI; HASAN, 2019). Na Figura 1, é apresentado um exemplo de uma boia utilizada para estas medições.

Figura 1: Exemplo de uma boia projetada para uso como plataforma de instrumentação meteorológica e oceanográfica.



Fonte: (MOORING, 2022)

No entanto, devido aos altos gastos de capital e despesas operacionais incorridos durante a instalação, operação e manutenção de fontes tradicionais de aquisição de dados, juntamente das anomalias de dados originadas por mau funcionamento do sensor ou falha de comunicação e dos altos recursos computacionais e requisitos de tempo de modelagem numérica, essas soluções são caras, especialmente para plataformas *offshore* de petróleo e gás.

Apesar destes meios supracitados, o estado do mar pode ser estimado empiricamente através da escala Beaufort, que classifica o estado do mar em 13 categorias com velocidades de vento e alturas de onda correspondentes. A escala fornece uma descrição concisa das características visuais de cada estado do mar e é adotada pela Organização Meteorológica Mundial (SINGLETON, 2008). Ao identificar essas características visuais, um estado do mar pode ser classificado segundo a escala.

Desta forma, somado aos recentes avanços tecnológicos e à evolução das técnicas de aprendizado de máquina, torna-se possível a medição e a classificação dos estados do mar utilizando um modelo de classificação de imagem de aprendizado profundo.

## 1.1 Motivação

Recentemente, a inteligência artificial e técnicas de aprendizado de máquina têm sido amplamente aplicado em diversos campos. Um exemplo em uma área próxima é a navegação autônoma de navios, que utiliza uma melhoria do ResNet para classificar marcadores de navegação marítima com alta precisão (PAN et al., 2020). Da mesma forma, modelos de aprendizado de máquina têm sido empregados como uma solução alternativa para prever e classificar as condições das ondas. Uma pesquisa recente sobre o tema relatou desempenho superior dessas novas abordagens em relação à modelagem de onda numérica contemporânea em termos de eficiência, portabilidade e menor complexidade computacional (UMAIR; HASHMANI; HASAN, 2019).

No entanto, estudos relacionados à aplicação de técnicas de aprendizado profundo em problemas de classificação do estado do mar são um assunto de interesse recente. Nesse contexto, estudos envolvendo vários tipos de conjuntos de dados do estado do mar e modelos de aprendizado profundo são raramente vistos, de modo que, para treinar e testar esses modelos, ainda é necessária uma grande quantidade de dados de parâmetros de onda adquiridos de fontes de dados tradicionais, havendo poucas opções de imagens rotuladas com suas respectivas medições.

Desta forma, os seguintes estudos foram tomados como base para o desenvolvimento do modelo classificatório, descritos na Tabela 1.

Tabela 1: Estudos para classificação do estado do mar

<b>Título</b>	<b>Ano</b>	<b>Modelo</b>	<b>Fonte dos dados</b>
<i>Application of deep learning in sea states images classification</i> (ZHANG; YU; QU, 2021)	2021	ResNet-152	Sensores óticos
<i>SpectralSeaNet: Spectrogram and convolutional network-based sea state estimation</i> (CHENG et al., 2020)	2020	2D CNN	Sensores de movimento de navio simulado
<i>Wave height inversion and sea state classification based on deep learning of radar sea clutter data</i> (LIU et al., 2021)	2021	CNN	Radar de banda X

Além destes, buscou-se conjuntos de dados do estado do mar, no entanto, os principais conjuntos de dados foram apresentados em formatos indesejados e usaram aquisição de dados de fontes tradicionais, de modo que os conjuntos de dados mais proeminentes e que melhor se encaixam no escopo deste projeto são apresentados na Tabela 2. Ambos os conjuntos utilizaram imagens obtidas em pesquisas seja em terra em superfícies estáveis, seja a bordo de navios realizando expedições. Outros conjuntos de dados, como dados de movimento de navios e conjuntos de dados de imagens de radar, também foram identificados. Contudo, devido aos seus métodos de aquisição e tipos de dados, eles também foram categorizados como inadequados para este estudo.

Tabela 2: Estudos utilizados para a obtenção de *datasets* do estado do mar

Título	Objetivo do estudo	Fonte dos dados
<i>A data set of sea surface stereo images to resolve space-time wave fields</i> (GUIMARÃES et al., 2020)	Investigar a geometria e a dinâmica das ondas oceânicas	Imagens
<i>A Novel Deep Learning Model for Sea State Classification Using Visual-Range Sea Images</i> (UMAIR et al., 2022)	Classificar o estado do mar a partir de uma imagem do mar de alcance visual	Imagens

## 1.2 Objetivo

Este projeto objetiva a criação de uma ferramenta de classificação do estado do mar de aprendizado profundo que classifica o estado do mar predominante usando apenas a imagem da superfície do mar. Para tal, há duas etapas principais a serem cumpridas: *(i)* a obtenção de um conjunto de imagens da superfície do mar que contenham informações acerca do estado do mar correspondente de cada imagem ou do conjunto como um todo e *(ii)* o processo de treinamento e classificação das imagens partindo-se dos fluxos de trabalhos propostos pelos estudos da Tabela 1.

Mais especificamente, devido à grande semelhança do escopo do problema e do tipo da fonte dos dados apresentados no estudo "Application of deep learning in sea states images classification" (ZHANG; YU; QU, 2021), optou-se por desenvolver um modelo que partisse

da arquitetura resultante deste artigo e propor outras arquiteturas a fim de averiguar a melhor combinação de técnicas para desenvolver o modelo final.

Dado que as arquiteturas de redes neurais convolucionais são compostas por duas partes fundamentais (a extração de *features* e as camadas de classificação chamadas de *fully connected layers*), propõe-se verificar o impacto da aplicação de diferentes estruturas para cada uma das partes para obter uma melhor combinação.

Desta forma, a partir dos dados selecionados e de uma ideia inicial de modelo, propõe-se um modelo de classificação do estado do mar baseado em aprendizado profundo que pode classificar o estado do mar a partir de uma imagem do mar de alcance visual adquirida de um único sensor óptico, que pode, por exemplo, estar montado em uma plataforma *offshore* de petróleo e gás.

A partir deste modelo, será criada uma plataforma denominada ***SeaStatefy*** que permite ao usuário realizar o *upload* de imagens da superfície do mar a fim de classificar seu estado do mar correspondente. Tal plataforma oferecerá uma interface simples e intuitiva para o usuário - o qual inicialmente é um pesquisador da área, mas podendo ser expandido para outros profissionais que possuam interesse em reconhecer o estado do mar. A ***SeaStatefy*** visa atender o uso em *desktops* e expor as técnicas e métodos utilizados na confecção do modelo classificatório, a fim de promover o desenvolvimento deste ramo de pesquisa.

Em suma, o objetivo consiste em:

1. Obtenção de um conjunto de dados da superfície do mar para a classificação dos estados do mar;
2. Estudo e comparação do estado da arte de modelos de classificação de aprendizado profundo nos conjuntos de dados de imagens do estado do mar obtidos.
3. Desenvolvimento de um novo modelo de classificação de imagens do estado do mar baseado em aprendizado profundo para monitoramento do estado do mar, seja em locais costeiros seja em *offshore*;
4. Avaliação do desempenho do modelo desenvolvido para a classificação do estado do mar;
5. Desenvolvimento da plataforma ***SeaStatefy*** que utilize o modelo desenvolvido para a classificação do estado do mar.

### 1.3 Justificativa

O desenvolvimento de uma plataforma e um modelo de classificação do estado do mar baseado em aprendizado profundo que pode classificar o estado do mar a partir de uma imagem do mar de alcance visual é importante para empresas de navegação, gestores de portos, empresas de exploração de petróleo, Marinha e Autoridades, guarda civil, dentre outros devido aos fatores supracitados como sucesso das operações das estruturas *offshore* de petróleo e gás e na segurança costeira promovida pela prevenção de catástrofes. O sistema, como descrito, caso colocado em produção, produziria classificações do estado do mar com boas confiabilidades.

Como visto anteriormente, sistemas de aprendizado de máquina para a classificação do estado do mar têm gerado resultados promissores com muito espaço para melhorias. Além disso, a solução de *Machine Learning (ML)* proposta não só minimiza os gastos de capital e despesas operacionais associados aos métodos tradicionais de classificação do estado do mar, mas também ajuda a reduzir a necessidade de uma força humana especializada para fins operacionais e de manutenção dos métodos tradicionais.

Em (ZHANG; YU; QU, 2021), um modelo ResNet-152 foi usado para classificar os estados do mar a partir de imagens ópticas da superfície do mar. O estudo dividiu os estados do mar em 10 categorias com base no movimento do navio e nas formas das ondas. Os dados de vídeo foram coletados de uma câmera de vídeo de alcance visual montada em um navio, tendo sido obtida uma precisão da validação do modelo de 89,3%. No entanto, o estudo não relata nenhuma medida para quantificar a precisão do teste do método proposto, nem menciona a precisão da classificação geral e do estado do mar. Além disso, o método, que adotou a divisão dos estados do mar em 10 categorias, não segue nenhum padrão reconhecido como a escala de Beaufort, por exemplo, o que reduz o significado prático real do estudo. Isto demonstra o espaço para crescimento em estudos nesta área que ainda está em seus estágios iniciais, apesar de já produzir resultados promissores.

Ademais, a separação natural da arquitetura de uma rede neural em extração de *features* e camadas de classificação permite utilizar a técnica de "dividir e conquistar", facilitando a otimização do sistema de forma modular em que cada problema pode ser melhor investigado, escolhendo-se o melhor modelo em cada etapa, desenvolvendo, ajustando e testando o modelo e a técnica selecionados. O mesmo vale para a arquitetura da plataforma dividida em *frontend* (interface com o usuário) e *backend* (processamento dos dados que contém o modelo classificatório), que também permite a manutenção e o desenvolvimento de forma modular, além de desacoplar a interface do modelo classificatório,

permitindo que esse seja refinado ao longo do tempo sem afetar a interface em si.

## 1.4 Organização do trabalho

A seguir no capítulo 2 são apresentados os conceitos teóricos acerca da Inteligência Artificial e do *Machine Learning* e seu desenvolvimento, juntamente de uma explicação das partes que constituem uma rede neural convolucional e as especificações de algumas das arquiteturas que serão utilizadas nos modelos deste documento. Em seguida, definem-se métricas de desempenho que aferem a qualidade de modelos de ML. Por fim, explica-se a definição da Escala Beaufort que é usada como guia para a classificação dos estados do mar neste projeto.

Na sequência, o capítulo 3 especifica as etapas propostas para o desenvolvimento do trabalho.

No capítulo 4, são apresentados os requisitos de desempenho do modelo classificador e os requisitos funcionais e não funcionais da plataforma *SeaStatefy* desenvolvida.

Em prosseguimento, no capítulo 5 são elencadas e descritas as tecnologias utilizadas no desenvolvimento do projeto.

Após este, apresenta-se o capítulo 6 que detalha cada um dos passos realizados em todo o desenvolvimento do projeto desde a aquisição dos conjuntos de dados até a implementação da plataforma junto de sua forma final que contém a interface e a API.

No capítulo 7 são mostrados os resultados provenientes dos testes propostos que demonstram a eficiência dos modelos desenvolvidos.

Por fim, no capítulo 8 apresentam-se as conclusões finais do projeto, ponderando-se os acertos e erros nas decisões, bem como os motivos para cada um deles. Juntamente são incluídas sugestões e propostas de linhas de pesquisa e de desenvolvimentos futuros.



# **PARTE II**

## **DESENVOLVIMENTO**

## 2 ASPECTOS CONCEITUAIS

A seguir encontram-se alguns conceitos que serão utilizados no decorrer deste trabalho. Inicialmente, define-se o que é a inteligência artificial e aborda-se as arquiteturas de redes neurais convolucionais que serão utilizadas na extração de *features* e na classificação das imagens do *dataset*, detalhando-se algumas arquiteturas específicas que serão utilizadas no desenvolvimento deste projeto. Em seguida são definidas as métricas empregadas na avaliação do desempenho dos modelos desenvolvidos. Por fim, define-se o que é o “estado do mar” e como este pode ser representado utilizando-se a Escala de Beaufort.

### 2.1 Inteligência Artificial

A Inteligência Artificial (IA) pode ser definida como o estudo do design dos chamados **agentes inteligentes**. Um **agente inteligente** consiste em um sistema que age de forma inteligente, isto é, age de forma apropriada de acordo com as circunstâncias em que se encontra a fim de atingir o seu objetivo específico, sendo flexível a ponto de se adaptar às variações externas em seu ambiente, aprendendo com experiências passadas e realizando as melhores escolhas dadas às limitações de percepção e de computação no intuito de maximizar as probabilidades de atingir o objetivo estabelecido (POOLE; GOEBEL; MACKWORTH, 1998).

Presente na computação há muitos anos, o início da inteligência artificial remonta a década de 1950 com pesquisas realizadas por Allen Newell , Herbert A. Simon e Marvin Minsky, dentre outros. A partir de então, estudiosos de diversas áreas como matemáticos, engenheiros e cientistas ao redor do mundo contribuíram para o desenvolvimento deste campo do conhecimento, de modo que hoje, a IA se faz presente no mercado e no cotidiano das pessoas.

A evolução da IA foi possibilitada principalmente pelo aprimoramento tecnológico das placas gráficas (GPUs) e pelo grande crescimento da quantidade de dados que são produzidos diariamente, bem como a maneira que esses dados são tratados e armazenados,

possibilitando o seu uso (fenômeno conhecido como *Big Data*). Com a melhoria das placas gráficas e das CPUs, tornou-se possível o processamento de uma grande quantidade de dados e operações e, conseqüentemente, o surgimento de algoritmos mais complexos e com treinamentos melhores de modo mais rápido devido ao paralelismo de operações oferecido pela arquitetura das GPUs. Ademais, com mais dados utilizáveis, o treinamento dos algoritmos torna-se mais eficiente, uma vez que há mais fontes de aprendizado para o aprimoramento do mesmo.

Das sub-áreas da Inteligência Artificial, o Aprendizado de Máquina - *Machine Learning (ML)* - tem crescido aceleradamente devido, em grande parte, à transformação digital pela qual a sociedade tem passado, o que trouxe para as empresas de tecnologia oportunidades de aplicações dessa sub-área para expandir seus produtos e mercados oferecendo serviços diferenciados a seus clientes. Dentre as aplicações do Aprendizado de Máquina encontram-se: carros autônomos, reconhecimento de fala, recomendação de conteúdo, detecção de fraudes, chatbots, análise de imagens médicas e diversas outras.

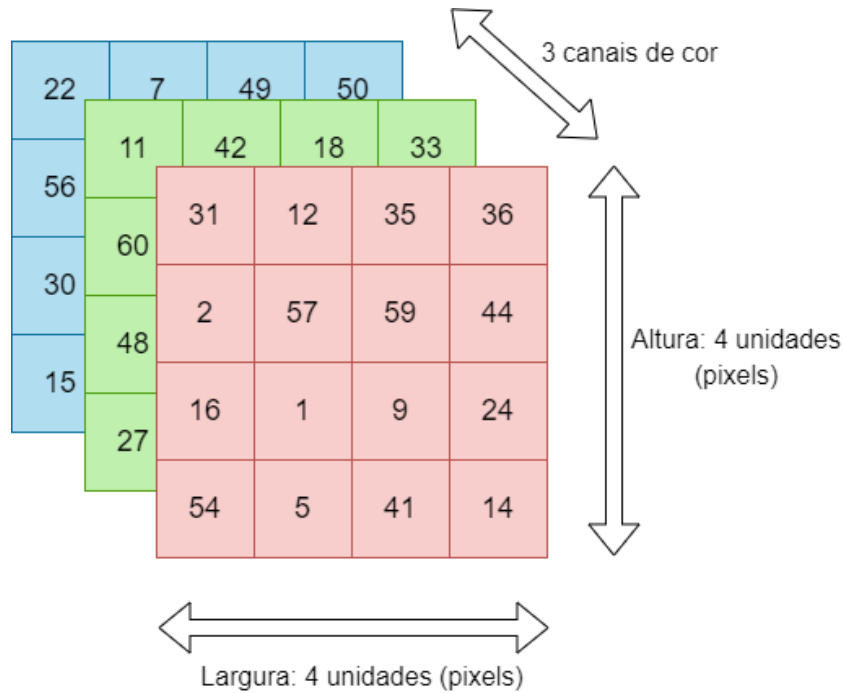
## 2.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais - *Convolutional Neural Network (CNN)* - são um dos algoritmos mais utilizados na classificação de imagens, sendo capazes de identificar e atribuir graus de importância a diversos aspectos das imagens de entrada, diferenciando-os entre si (ALBAWI; MOHAMMED; AL-ZAWI, 2017). Sua arquitetura é altamente inspirada no padrão de conectividade dos neurônios do cérebro humano.

As CNNs diferem de outras formas de redes neurais artificiais em que, em vez de focar na totalidade do domínio do problema, o conhecimento sobre o tipo específico de entrada é explorado. Isso, por sua vez, permite codificar recursos específicos da imagem na arquitetura, tornando a rede mais adequada para tarefas focadas em imagens - enquanto reduz ainda mais os parâmetros necessários para configurar o modelo (O'SHEA; NASH, 2015).

Nesse contexto, define-se um **tensor** como uma matriz multidimensional cujo elemento corresponde a cada pixel de cada canal de uma figura. A imagem de entrada de uma CNN consiste, basicamente, de um **tensor** 3D de dimensões (altura da imagem) x (largura da imagem) x (canais de entrada), de modo que, existem três canais (um para cada um dos três elementos do espectro RGB) para imagens coloridas conforme apresentado na Figura 2 abaixo. Caso seja utilizada a escala preto e branco, o tensor possuirá somente um canal.

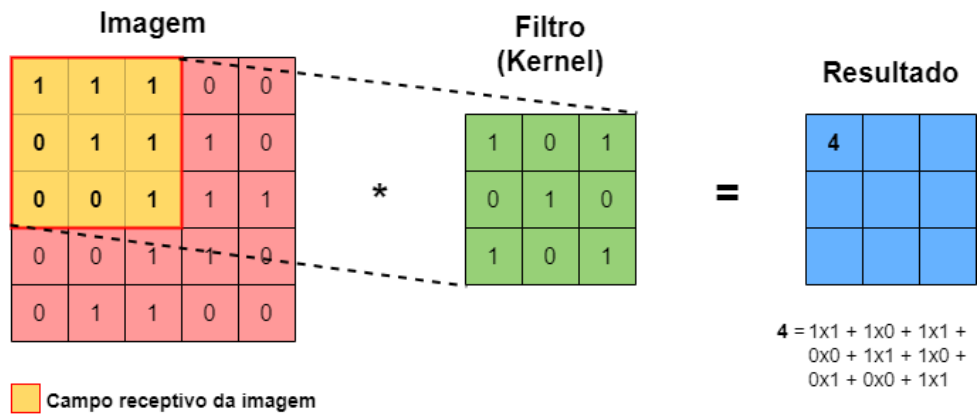
Figura 2: Representação de uma imagem RGB em 3 camadas/canais (vermelho, verde e azul). Cada número representa a intensidade do pixel para cada uma das cores



Fonte: os autores

Ao entrar na CNN, a primeira camada pela qual a imagem de entrada passa é a **camada convolucional** (*convolutional layer*), a qual é responsável por extrair as características de baixo nível (*low-level features*), tais como bordas, curvas, etc. Este processo se dá através da **convolução** da imagem com um **filtro** (*kernel*) que corresponde a uma matriz de números que são chamados de parâmetros ou pesos do filtro. Este kernel possui a mesma quantidade de canais que a imagem original, isto é, para o caso de uma imagem RGB, o filtro seria composto por 3 matrizes, cada uma aplicada em uma dimensão da imagem original. A convolução é obtida analisando-se um conjunto de pixels adjacentes, denominado **campo receptivo** (*receptive field*), sendo que cada pixel deste conjunto é multiplicado pelo valor do peso correspondente do filtro e por fim são somados, obtendo um único valor conforme representado na Figura 3.

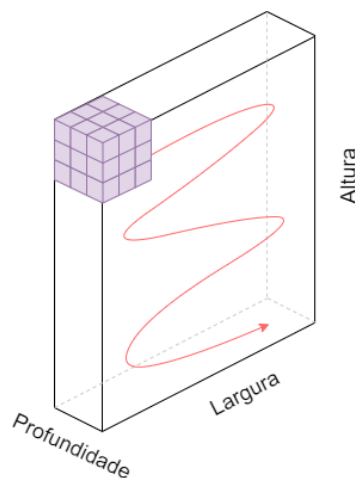
Figura 3: Representação da convolução de um kernel 3 x 3 com uma imagem 5 x 5 e o resultado desta operação. Para o campo receptivo da imagem (representado em laranja) o resultado é 4



Fonte: os autores

Após a convolução, o campo receptivo desloca-se pela matriz de entrada de acordo com um tamanho de passo estabelecido, o *stride*, conforme apresentado na Figura 4, de modo que, a cada deslocamento, calcula-se o valor da convolução entre o filtro e o novo campo receptivo, o qual é inserido em uma nova matriz denominada **mapa de ativação** ou **mapa de atributos** (*activation map* ou *feature map*). Este deslocamento pode ser realizado uma ou diversas colunas por vez, gerando um mapa de atributos maior ou menor, respectivamente.

Figura 4: Representação do deslocamento de um *kernel* 3 x 3 em uma imagem se movendo da esquerda para a direita, de cima para baixo



Fonte: os autores

Neste processo de convolução, a depender do tamanho do filtro utilizado e do ta-

manho do mapa de ativação resultante desejado, pode ser necessária a utilização de um *padding*, que consiste em um processo de adicionar camadas de zeros à matriz de entrada, como demonstrado na Figura 5, contribuindo para evitar problemas como o fato de pixels no meio serem usados com mais frequência do que pixels em cantos e bordas e, conseqüentemente, as informações nas bordas das imagens não seriam preservadas, assim como as informações no meio. Além disso, o *padding*, permite que a matriz resultante da convolução seja de tamanho maior ou igual à matriz de entrada.

Figura 5: Representação da adição de uma camada de *padding* de zeros à matriz

0	0	0	0	0	0	0
0	32	4	7	2	45	0
0	9	8	7	5	3	0
0	67	33	1	23	67	0
0	73	55	62	4	24	0
0	9	5	13	1	8	0
0	0	0	0	0	0	0

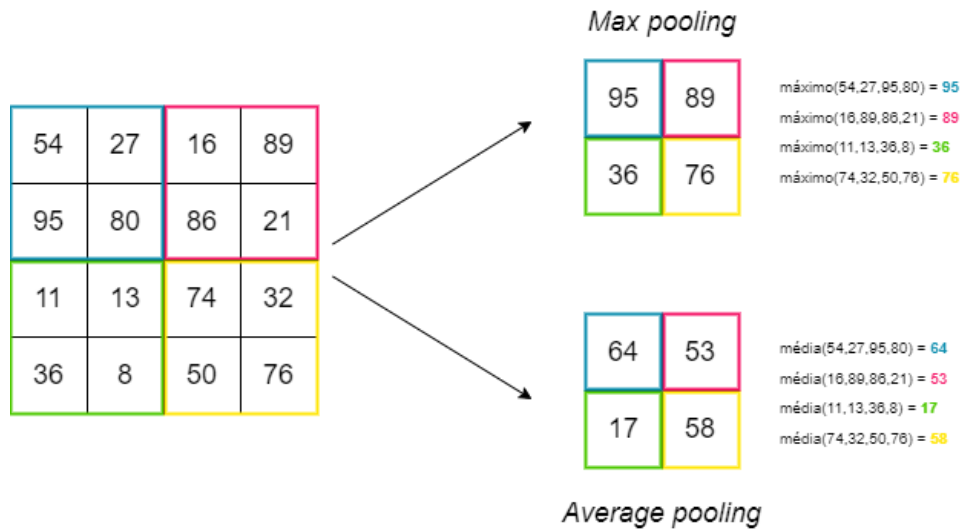
Fonte: os autores

Ao utilizar camadas convolucionais adicionais, sendo o mapa de atributos de uma camada a entrada da seguinte, a arquitetura também torna-se capaz de extrair **recursos de alto nível** (*high-level features*), conseguindo identificar os elementos/objetos contidos na imagem.

Entre as camadas convolucionais, há as **camadas de agregação** (*pooling layers*). Estas são responsáveis por reduzir as dimensões das matrizes de entrada, de modo a diminuir o poder computacional necessário para processar os dados, aumentando a sua eficiência. Ademais, estas camadas tornam a matriz de saída aproximadamente invariante a pequenas translações ou rotações em sua entrada, isto é, para pequenas translações ou rotações na matriz de entrada, grande parte dos valores de saída resultantes permanecem constantes. Desta forma, torna-se útil para extrair as características dominantes da imagem.

Existem dois tipos de pooling comumente utilizados: o *max pooling* e o *average pooling*. O *max pooling* seleciona o valor máximo do campo receptivo da matriz de entrada, enquanto o *average pooling* retorna a média dos valores analisados. A Figura 6 apresenta estes dois tipos de pooling em uma matriz de entrada 4 x 4. O conjunto camada convolucional seguida da camada de agregação forma a i-ésima camada de uma CNN.

Figura 6: Aplicação de um *max pooling* e de um *average pooling* em uma matriz 4 x 4 com um campo receptivo 2 x 2 e um deslocamento de 2

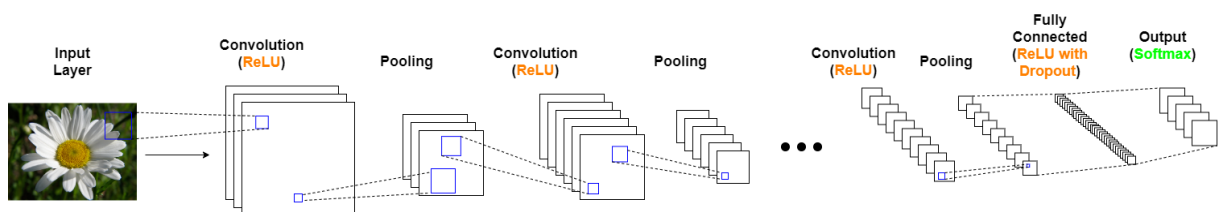


Fonte: os autores

O resultado final, após passar por diversas camadas convolucionais e de agregação, é uma matriz que contém um mapa de atributos com as *features* dominantes que melhor representam a imagem de entrada, restando, portanto, a necessidade de, a partir destes atributos, classificar a imagem segundo as classes pré-estabelecidas.

Este último estágio, a classificação, é realizado pela **camada totalmente conectada** (*fully-connected layer*). Tal camada corresponde a um Multi-Layer Perceptron (MLP) que recebe em sua entrada o tensor de *features* anteriormente processado e produz um vetor de  $N$  dimensões, sendo  $N$  a quantidade de classes do problema e cada elemento deste vetor correspondendo à probabilidade da entrada pertencer a cada uma das  $N$  classes. Neste processo, o modelo é capaz de distinguir entre características dominantes e de baixo nível a fim de compreender qual o impacto de cada atributo para diferenciar o pertencimento a cada uma das classes. Uma arquitetura completa geral de uma CNN é apresentada na Figura 7.

Figura 7: Representação de uma arquitetura geral de uma CNN



Fonte: os autores

A *fully-connected layer* é composta, essencialmente por nós que estão conectados entre si. Cada nó possui uma **função de ativação** que define a saída desse nó dada uma entrada ou conjunto de entradas e é usada para determinar a saída da rede neural como uma classe ou outra, mapeando os valores resultantes entre 0 e 1 ou -1 e 1, por exemplo. Alguns exemplos de *função de ativação* são a ReLU ( $R(z) = \max(0, z)$ ), a Sigmoid ( $\sigma(z) = \frac{1}{1+e^{-z}}$ ), dentre outras.

No processo de treinamento das arquiteturas, uma operação essencial é a *backpropagation* que consiste na prática de ajustar os pesos de uma rede neural com base na taxa de erro obtida na época anterior (i.e., iteração anterior). O ajuste adequado dos pesos garante menores taxas de erro, tornando o modelo confiável ao aumentar sua generalização.

Ao longo dos anos, inúmeras arquiteturas foram desenvolvidas e testadas com o propósito de solucionar problemas do mundo real. Dentre elas, destacam-se a VGG-16 e a ResNet.

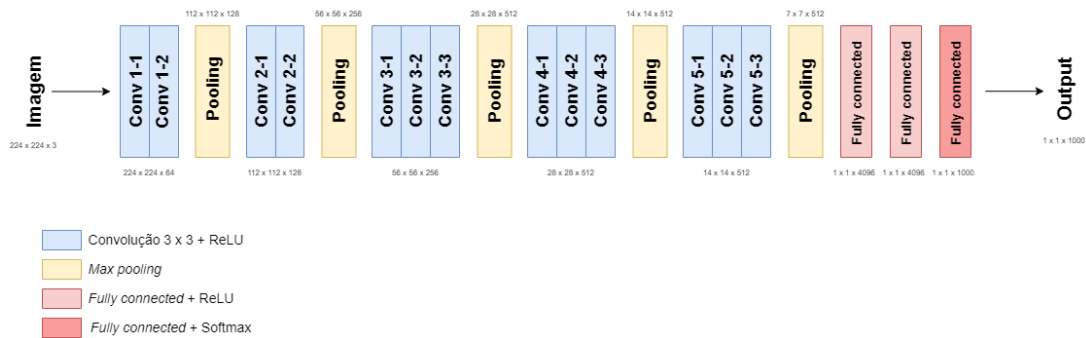
### 2.2.1 VGG-16

VGG-16 é uma CNN treinada com imagens do *ImageNet*, um banco de dados constituído por mais de 14 milhões de imagens divididas em, aproximadamente, 20000 categorias e rotuladas manualmente (LAB, 2022). Foi proposta em 2014 no paper “*Very Deep Convolutional Networks for Large-Scale Image Recognition*” pelos cientistas da Universidade de Oxford Karen Simonyan e Andrew Zisserman (SIMONYAN; ZISSERMAN, 2014).

O modelo recebe um tensor de 224 x 224 com três canais (R, G e B) como entrada. O *input* passa, primeiramente, por duas camadas convolucionais, cada uma composta por um *kernel* de dimensão 3 x 3, *stride* 1 e *padding* 1 e por uma função de ativação ReLU. Os mapas de ativação resultantes entram em uma camada de *max pooling* de campo receptivo 2 x 2 e 2 px de *stride*, reduzindo o tamanho dos mapas pela metade. Estes passam, então, por mais três sequências de camadas similares às mencionadas anteriormente. Por fim, utilizam-se três camadas totalmente conectadas, sendo a última seguida por uma função de ativação Softmax, produzindo um vetor de 1000 dimensões que contém a probabilidade de pertencimento a cada uma das 1000 classes (SIMONYAN; ZISSERMAN, 2014). A Figura 8 ilustra a arquitetura descrita.



Figura 8: Arquitetura da VGG-16



Fonte: os autores

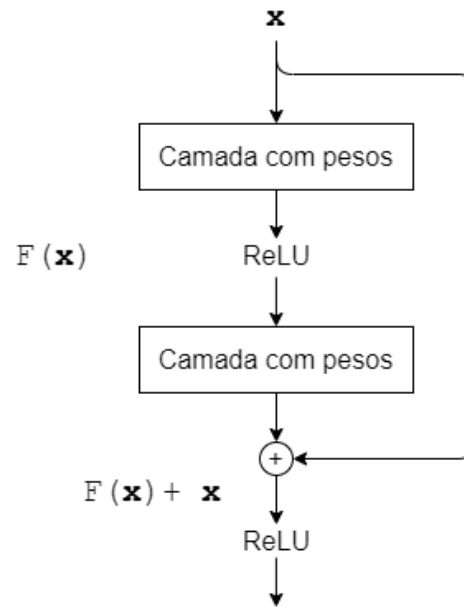
## 2.2.2 ResNet

A fim de modelar funções cada vez mais complexas, tende-se a aumentar a profundidade das redes neurais, uma vez que o aprendizado de *features* mais complexas ocorre de forma progressiva. No contexto das CNNs, isso significa, por exemplo, que as primeiras camadas são responsáveis pela detecção de cantos e bordas, seguidas pelas que identificam texturas que, por sua vez, são sucedidas pelas que são capazes de discernir objetos.

Uma rede profunda poderia ser construída mediante cópia das camadas de um modelo raso treinado e adição de novas camadas que somente mapeiam a entrada na saída (função identidade). Assim, espera-se que um modelo profundo desempenhe melhor ou minimamente tão bem quanto seu equivalente raso, situação que nem sempre é observada devido a não convergência do modelo para esta solução (HE et al., 2016).

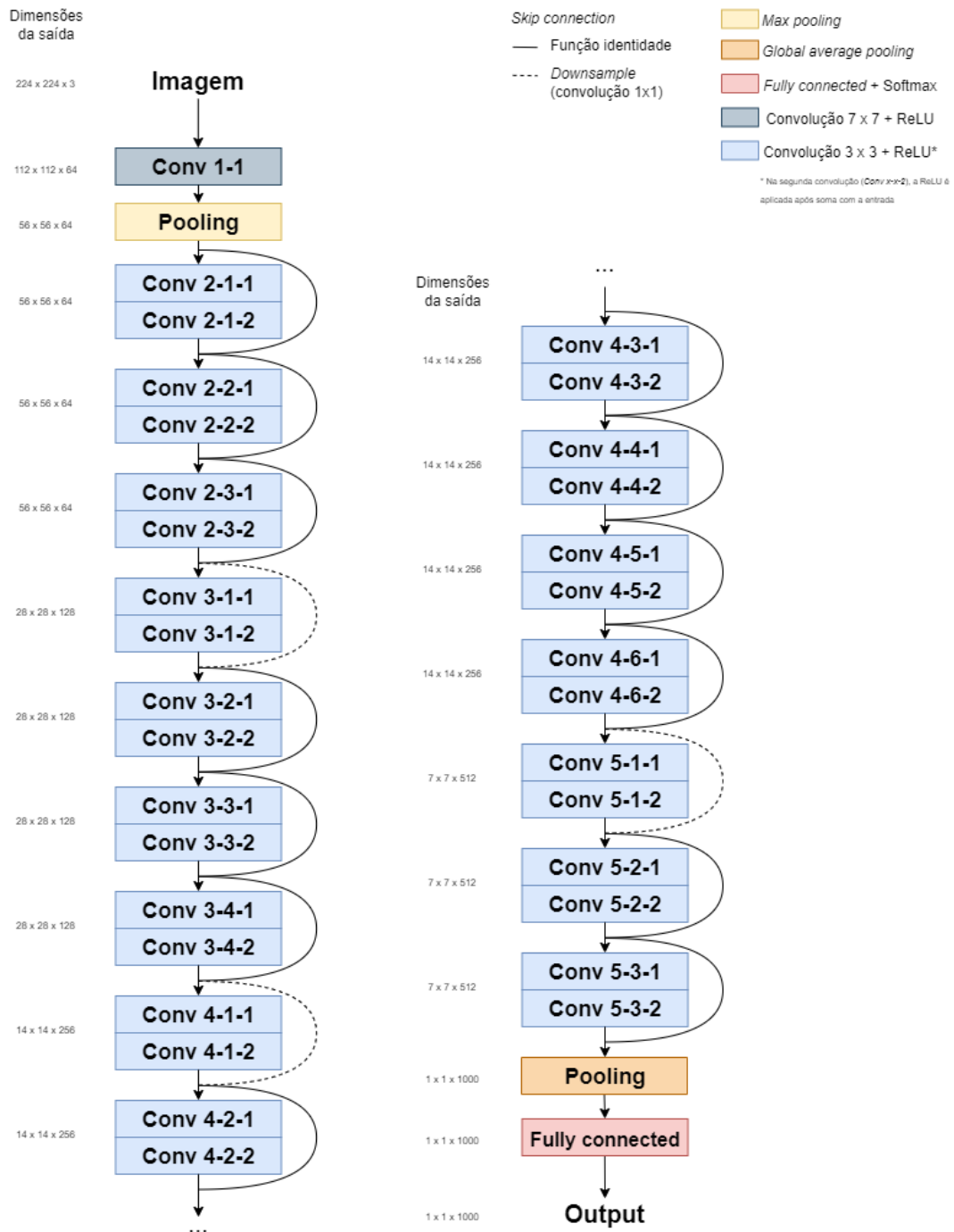
A ResNet surgiu para abordar este problema. Também denominada rede residual, é em uma arquitetura de CNN introduzida em 2015 pelos pesquisadores da Microsoft Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun no paper “*Deep Residual Learning for Image Recognition*” (HE et al., 2016). Ela propõe o uso de “blocos residuais”, isto é, uma sequência de duas ou mais camadas de convolução com uma “conexão salto” (*skip connection*), no qual o mapa de ativação final é somado, elemento a elemento e canal a canal, à entrada do bloco, conforme representado na Figura 9. A soma matricial, entretanto, requer que as dimensões da saída e da entrada sejam iguais e, para tal, duas estratégias são adotadas: recorre-se a uma concatenação de tensores ou elementos nulos (0) para aumentar as dimensões da entrada ou igualam-se as dimensões por meio da convolução da entrada do bloco com um *kernel* de 1 x 1 (HE et al., 2016). A Figura 10 detalha a arquitetura da ResNet-34, a qual é baseada em blocos residuais.

Figura 9: Bloco residual



Fonte: os autores

Figura 10: Arquitetura da ResNet-34

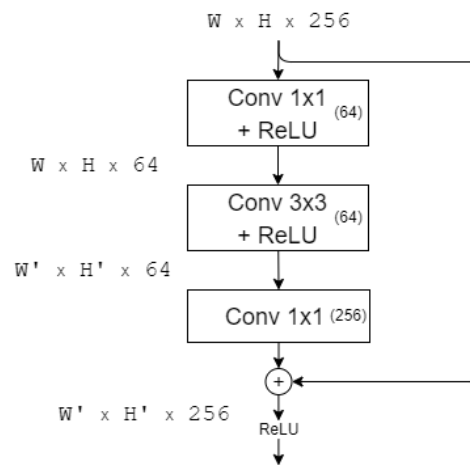


Fonte: os autores

As ResNets de 50 camadas ou mais (ResNet-50, ResNet-101, ResNet-152) empregam uma variante do bloco residual original, o chamado bloco residual *bottleneck*, a fim de reduzir a complexidade computacional. Ele é constituído por três camadas convolucionais com *kernels* de tamanho 1 x 1, 3 x 3 e 1 x 1, respectivamente, sendo as convoluções 1 x 1 usadas na redução ou aumento do número de canais de um mapa de ativação. Para

fins ilustrativos, um exemplo de abordagem *bottleneck* é apresentado a Figura 11. Uma entrada de dimensão  $W \times H \times 256$ , tem seu número de canais diminuído após a primeira convolução  $1 \times 1$ , resultando em um mapa de ativação de  $W \times H \times 64$ . Assim, na próxima convolução, é possível utilizar 64 *kernels* de dimensões  $3 \times 3 \times 64$ , ao invés de 64 de  $3 \times 3 \times 256$ , diminuindo a quantidade de cálculos efetuados e, conseqüentemente, o tempo de treinamento sem prejudicar significativamente a performance (HE et al., 2016). Por fim, utiliza-se uma convolução  $1 \times 1$  para restaurar o número inicial de canais.

Figura 11: Bloco Residual *Bottleneck*



Fonte: os autores

Nota-se que ambos os blocos residuais propiciam o aprendizado de funções identidade, dado que a definição de todos os pesos das camadas convolucionais como 0 ocasionaria em uma soma de um conjunto de convoluções com *kernels* nulos - o que resulta em um tensor nulo - com a entrada do bloco, mapeando-a, assim, na saída. Além disso, as *skip connections* permitem a propagação direta do gradiente para as camadas iniciais durante o passo de *backpropagation*, garantindo a atualização dos parâmetros e a não estagnação precoce da rede (HE et al., 2016).

## 2.3 Métricas de desempenho

A fim de atestar a qualidade de diferentes modelos classificatórios, existem algumas métricas de desempenho padrões que indicam não só a capacidade de acertar, mas também a de generalizar e o quão bem o classificador atua para cada uma das diferentes classes, prevenindo, assim, classificadores especialistas capazes de prever apenas uma classe bem e ter um mal desempenho ao rotular as demais. Dentre elas serão detalhadas a **acurácia**

(*accuracy*), a **precisão** (*precision*), a **revocação** (*recall*) e o **F1-score**.

Para computar estas métricas, as seguintes variáveis são necessárias:

- $V_P$ : Verdadeiro Positivo;
- $F_P$ : Falso Positivo;
- $V_N$ : Verdadeiro Negativo;
- $F_N$ : Falso Negativo;

A relação entre elas pode ser descrita pela **matriz de confusão** a seguir na Figura 12. A matriz de confusão é usada para avaliar a precisão de uma classificação, ela é tal que as linhas representam as previsões e as colunas representam os rótulos, portanto, o melhor cenário seria uma matriz diagonal, pois todos os rótulos seriam previstos como eles mesmos, o que significa 100% de precisão.

Figura 12: Matriz de confusão para as classes positivo e negativo da perspectiva da Classe A

		VERDADEIRO	
		Classe A	Classe B
PREDITO	Classe A	VP	FP
	Classe B	FN	VN

Fonte: os autores

A acurácia é a razão de previsões corretas, o que significa que o conjunto de rótulos previsto para uma amostra deve corresponder exatamente ao conjunto de rótulos correspondente na verdade. Ela pode ser definida matematicamente como:

$$Acurácia = \frac{V_P + V_N}{V_P + V_N + F_P + F_N} \quad (2.1)$$

Precisão é a capacidade do classificador não rotular como positiva uma amostra que é negativa, o que significa não classificar objetos de outras classes como objetos de uma classe específica  $C$ . Aqui, o melhor valor é 1 e o pior valor é 0. Ela pode ser definida como:

$$Precisão = \frac{V_P}{V_P + F_P} \quad (2.2)$$

Revocação é a capacidade do classificador de encontrar todas as amostras positivas, o que significa que, mesmo que o modelo seja preciso, ele deve ser capaz de encontrar todas as amostras de uma classe e não apenas um pequeno grupo delas. Ela pode ser definida como:

$$Revocação = \frac{V_P}{V_P + F_N} \quad (2.3)$$

O *F1-score* pode ser interpretado como uma média ponderada da precisão e da revocação, sendo que ele atinge seu melhor valor em 1 e a pior pontuação em 0 (SCIKITLEARN, 2022). A contribuição relativa da precisão e da revocação para o *F1-score* é igual. Portanto, ele é basicamente uma medida da acurácia de um modelo em um conjunto de dados. Ele pode ser definido como:

$$F1\text{-score} = 2 \times \frac{\text{precisão} \times \text{revocação}}{\text{precisão} + \text{revocação}} \quad (2.4)$$

## 2.4 Escala Beaufort

O **estado do mar** (*sea state*) é uma medida da agitação da superfície do mar (ondas) e é usado para definir a segurança de uma navio em atravessar determinado trecho do oceano em navegação; para definir a segurança das operações de plataformas de petróleo e navios de perfuração e exploração oceânica, e na costa; para quantificar a possibilidade de agitação nas praias, ressacas, alagamentos; dentre outros. O **estado do mar** é definido pela força do vento, pela distância de busca de águas abertas disponível para geração de ondas e a duração de tempo durante a qual as ondas podem acumular energia do vento (THOMSON et al., 2016). Desta forma, é resultado da altura da onda, do período da onda e velocidade do vento.

A escala Beaufort propõe uma estimativa empírica da velocidade do vento e altura média das ondas ao relacioná-las com aspectos físicos da superfície marítima, como altura aparente das ondas, presença de rebentação e espuma (estado do mar). Foi idealizada

pelo hidrógrafo irlandês Francis Beaufort em 1805 e, originalmente, associava 13 estados arbitrários de intensidade dos ventos com a velocidade de uma fragata totalmente equipada e a quantidade de vela que ela poderia carregar (WALLBRINK; KOEK, 2009). Tal formulação inicial, todavia, passou por diversas revisões a fim de torná-la mais relevante e funcional no contexto da navegação moderna. Entre elas, destacam-se a troca da obsoleta embarcação de guerra como referência pela aparência do mar, modificação apresentada pelo meteorologista britânico George Simpson no início do século XX, e a introdução pelo Meteorological Office (Met Office) da seguinte fórmula para calcular velocidade do vento (m/s) conforme o grau da escala Beaufort (B) (OFFICE, 2012; WALLBRINK; KOEK, 2009).

$$v = 0,836 \cdot B^{\frac{3}{2}}$$

Em 1926, foi adotada pela Organização Meteorológica Internacional, sofrendo novas atualizações em 1939 e 1946 (NEELY, 2012). A Tabela 3 representa a escala Beaufort atual. A velocidade do vento se refere ao valor médio previsto registrado a 10 metros acima do nível do mar e a altura das ondas, sua dimensão provável se observada em mar aberto.

Tabela 3: Especificações da escala Beaufort atual

Grau	Descrição	Velocidade do vento (m/s)	Altura média das ondas (m)	Aspecto do mar
0	Calmo	0 - 0,2	0	Espelhado
1	Aragem	0,3 - 1,5	0,1	Pequenas movimentações na superfície sem a formação de cristas de espuma
2	Brisa leve	1,6 - 3,3	0,2	Pequenas ondulações com cristas translúcidas e sem rebentação

---

3	Brisa fraca	3,4 - 5,4	0,6	Ondulações com cristas que ocasionalmente possuem espuma; rebentações mais frequentes
4	Brisa moderada	5,5 - 7,9	1	Pequenas ondas mais longas com cristas de espuma
5	Brisa forte	8 - 10,7	2	Ondas moderadas e longas com cristas de muita espuma
6	Vento fresco	10,8 - 13,8	3	Grandes ondas começam a se formar e cristas de espuma são mais extensas e recorrentes; presença de borrifos
7	Vento forte	13,9 - 17,1	4	Mar revolto com espuma e borrifos soprados na direção dos ventos

---



---

8	Ventania	17,2 - 20,7	5.5	Mar revolto com ondas moderadamente altas, rebentações e borrifos constantes; formação de faixas de espumas bem marcadas ao longo da direção do vento
9	Ventania forte	20,8 - 24,4	7	Mar revolto com ondas altas e visibilidade afetada
10	Tempestade	24,5 - 28,4	9	Mar revolto com ondas muito altas com longas cristas; presença de muita espuma que é soprada na direção do vento, formando densas faixas brancas, e torna a superfície do mar branca; visibilidade precária
11	Tempestade violenta	28,5 - 32,6	11.5	Mar revolto com ondas extremamente altas e coberto por manchas brancas de espuma; visibilidade precária

---

---

12	Furacão	$\geq 32,7$	14	Mar e ar preenchidos por espuma e borrifos; mar branco; visibilidade quase nula
----	---------	-------------	----	---

---

### 3 METODOLOGIA DE TRABALHO

No intuito de desenvolver este projeto, foram utilizadas imagens provenientes de *datasets* que contêm imagens da superfície do mar rotuladas segundo o grau da escala Beaufort correspondente ou com as informações de altura média das ondas para aquele conjunto de imagens. A partir destes dados, explorou-se arquiteturas de redes neurais convolucionais a fim de obter um modelo capaz de classificar as imagens em uma das 3 classes: mar calmo/leve (graus 0-3 da escala Beaufort); mar moderado/agitado (graus 4-7 da escala Beaufort); mar forte/extremo (graus 8-12 da escala Beaufort).

A metodologia seguida no desenvolvimento deste projeto envolveu os seguintes passos:

1. Busca e aquisição de *datasets* rotulados de imagens da superfície do mar;
2. Estudo de arquiteturas e modelos de CNNs, partindo-se de modelos utilizados em artigos relacionados a este problema ou a problemas similares;
3. Seleção e comparação de modelos candidatos para realizar a extração de *features* e a classificação das imagens;
4. Pré-processamento das imagens selecionadas, manipulando suas dimensões e aplicando técnicas de *data augmentation* para obter maior diversidade de dados;
5. Treinamento dos modelos selecionados a partir da escolha de hiper parâmetros e de técnicas para tratar *datasets* desbalanceados;
6. Definição e realização dos testes de desempenho dos modelos a fim de selecionar aquele com a melhor performance entre os candidatos;
7. Implementação e integração da plataforma com o modelo classificador, desenvolvendo uma interface para a ferramenta de classificação a partir do mapeamento do usuário principal da ferramenta.

A partir da abordagem proposta neste trabalho, buscou-se obter um classificador de imagens capaz de estimar o estado do mar a partir do processamento de imagens

da superfície do mar e que sua performance seja suficientemente boa, sendo dependente da qualidade dos dados de treinamento encontrados, mas oferecendo a possibilidade para melhorar seu desempenho a partir da adição de novas imagens em passos futuros, de modo que a ferramenta aqui desenvolvida possa servir como base para novos desdobramentos. A explicação e o detalhamento de cada uma das etapas elencadas na metodologia estão contidos no Capítulo 6 - Projeto e Implementação.

## 4 ESPECIFICAÇÃO DE REQUISITOS

Neste capítulo serão descritos os requisitos do sistema quanto às suas funcionalidades e à capacidade de desempenho esperada discutindo possibilidades de implementação, a partir das definições e conceitos apresentados nos tópicos anteriores.

O sistema a ser desenvolvido é constituído por duas partes principais, cada qual com requisitos específicos bem definidos. No que se refere ao **classificador**, deve-se garantir a acurácia satisfatória do modelo e a associação da imagem ao grau da escala Beaufort correspondente em um curto intervalo de tempo (da ordem de segundos no máximo). Em relação à **ferramenta gráfica** que viabiliza a interação entre inteligência artificial e usuário, faz-se necessário cumprir uma série de requisitos funcionais e não-funcionais que impactam diretamente na experiência e na jornada do usuário ao utilizar o produto desenvolvido. Desta forma, listam-se a seguir os principais requisitos para o protótipo do estimador do estado do mar.

### 4.1 Requisitos do classificador

De modo geral, em problemas de classificação, deve-se garantir que os dados empregados durante o treinamento do modelo estejam balanceados, isto é, que haja uma relação de equilíbrio entre a quantidade de exemplos por classe. Uma vez que o algoritmo de inteligência artificial visa minimizar erros, alimentar o classificador com um *dataset* cuja proporção de entradas por rótulo é discrepante implica no enviesamento do modelo em favor das classes majoritárias. Assim, novas entradas serão raramente catalogadas como pertencentes às classes minoritárias, o que representa um comportamento não desejado.

Como discutido anteriormente, propôs-se a utilização de uma CNN como base do modelo de classificação. O uso de redes neurais, todavia, exige um grande conjunto de dados, visto que busca-se replicar o mecanismo de processamento de imagens desempenhado pela visão humana: a habilidade de discernimento entre elementos e de identificação de atributos e padrões. Desse modo, devido à limitação do número de observações disponíveis,

optou-se pela adoção do *transfer learning*, técnica de *machine learning* baseada na ideia de superar o paradigma da aprendizagem isolada, mediante transferência de conhecimento obtido na resolução de uma tarefa semelhante. Para tal, as CNNs pré-treinadas disponibilizadas em repositórios *online*, por exemplo, o *PyTorch Hub*, foram analisadas, devidamente selecionadas e refinadas.

Por fim, posto que há três classes possíveis - mar calmo/leve, mar moderado/agitado e mar forte/extremo - assegurou-se que o modelo final detivesse uma acurácia superior a 33%, probabilidade de acerto para um palpite aleatório. Ademais, no que tange a sua integração com a plataforma gráfica, garantiu-se que a classificação fosse concluída dentro de um intervalo pequeno de tempo, da ordem de poucos segundos, a fim de proporcionar uma experiência mais agradável ao usuário final.

## 4.2 Requisitos da plataforma

### 4.2.1 Requisitos funcionais

Os requisitos funcionais correspondem às funções e aos componentes que de fato são desempenhadas pelo sistema, isto é, são as tarefas e serviços que este provê ao usuário. Para este projeto, visando à maior satisfação do usuário, foram elencados os seguintes requisitos funcionais:

- Ser capaz de, a partir do *upload* de uma imagem da superfície do mar, estimar o estado do mar correspondente;
- Interface simples para realizar o *upload* da imagem a ser classificada sem a necessidade de realizar login;
- Apresentar a classe predita pelo classificador junto de uma explicação sobre aquela classe: quais as características do mar, quais as condições de vento, altura das ondas e quais os níveis da escala de Beaufort correspondem à classe predita;
- Apresentar informações sobre o classificador, tais como qual é a sua capacidade preditiva, sua acurácia e seu desempenho segundo às métricas estabelecidas.

### 4.2.2 Requisitos não-funcionais

Os requisitos não-funcionais são todos aqueles relacionados à forma como o software tornará realidade o que está sendo planejado, isto é, eles descrevem *como* será feito ao invés

de *o que* será feito. Dito isto, os seguintes requisitos não-funcionais foram selecionados:

- O sistema deve se comunicar com o servidor Web, por meio do protocolo HTTP;
- A interface deve ser simples, agradável e de fácil utilização;
- Ser capaz de receber imagens e enviar para o servidor onde ocorrerá o processamento;
- A ferramenta deve ser acessada via *desktop*, não havendo a necessidade de uma versão *mobile*;
- O processamento e classificação da imagem devem ser realizados de forma rápida, bem como o envio dos resultados do back para o front-end.

## 5 TECNOLOGIAS UTILIZADAS

Nesta seção são apresentadas as tecnologias selecionadas pelo grupo para o desenvolvimento e execução do projeto.

### 5.1 Docker

O *Docker* é uma plataforma *open source* utilizada para desenvolver, implantar e administrar aplicações. Ele propõe o empacotamento de aplicativos em contêineres, ambientes isolados que contêm todas as dependências necessárias para executá-los (DOCKER, 2022).

A fim de usufruir do poder computacional - destaca-se a memória RAM, disco e placas gráficas - das máquinas do Tanque de Provas Numérico (TPN-USP), treinou-se o modelo de classificação dentro de um contêiner. Posto que tais computadores eram empregados por outros graduandos, mestrandos e professores em suas pesquisas, garantiu-se, assim, que não houvessem conflitos ou problemas de compilação e que as versões das bibliotecas externas importadas para o treinamento fossem as esperadas.

### 5.2 Python

*Python* é uma linguagem interpretada de alto nível usada com sucesso em milhares de aplicativos de negócios do mundo real em todo o mundo em uma grande variedade de cenários. É conhecida por ser extremamente versátil, devido a sua simplicidade e tipagem forte e dinâmica (PYTHON, 2022).

A escolha por utilizar esta linguagem se deu devido às suas características que incluem simplicidade e consistência, flexibilidade, acesso a bibliotecas e estruturas poderosas de IA e aprendizado de máquina (ML), independência de plataforma e grandes comunidades.



## 5.3 PyTorch

*PyTorch* é um *framework* de aprendizado de máquina de código aberto baseada na biblioteca Torch, usada para aplicativos como visão computacional e processamento de linguagem natural, desenvolvido principalmente pela Meta AI (PYTORCH, 2022).

Optou-se pela utilização deste *framework* devido à sua facilidade de implementação de modelos de redes neurais com um bom desempenho, visto que a complexidade inerente ao aprendizado de máquina é tratada internamente pela biblioteca PyTorch e escondido atrás de APIs intuitivas livres de efeitos colaterais e inesperados penhascos de desempenho (PASZKE et al., 2019).

## 5.4 Heroku

*Heroku* é uma plataforma como serviço (*Platform as a Service* - PaaS) baseada em um sistema de contêiner gerenciado, com serviços de dados integrados e um poderoso ecossistema para implantar e executar aplicativos modernos. A experiência do desenvolvedor *Heroku* é uma abordagem centrada em aplicativos para entrega de software, integrada com as ferramentas e fluxos de trabalho de desenvolvedor mais populares da atualidade, tais como Python e React (HEROKU, 2022).

O *Heroku* executa os aplicativos dentro de *dynos* — contêineres inteligentes em um ambiente de tempo de execução totalmente gerenciado e confiável. As aplicações em Node, Ruby, Java, PHP, Python, Go, Scala ou Clojure são implantadas em um sistema de compilação que produz um aplicativo pronto para execução. O sistema e as pilhas de linguagens são monitorados, corrigidos e atualizados, para que esteja sempre pronto e atualizado. O tempo de execução mantém os aplicativos em execução sem nenhuma intervenção manual.

## 5.5 SciKit Learn

O *SciKit Learn* é uma das bibliotecas mais utilizadas quando se trabalha com Aprendizado de Máquina em *Python*. Ela oferece ferramentas simples e efetivas para análise de dados preditiva e diversos algoritmos para classificação, regressão, clusterização, redução de dimensionalidades, pré-processamento de dados e avaliação de modelos (SCIKITLEARN, 2022).

## 5.6 NumPy

*NumPy* é uma biblioteca de código aberto para *Python* que viabiliza o uso de estruturas de dados poderosas, principalmente vetores multidimensionais, e a realização de cálculos numéricos e álgebra linear com alta performance (NUMPY, 2022). Sua utilização se deu na manipulação das imagens e nas operações algébricas entre os tensores que as constituem.

## 5.7 Google Colab

O *Google Colab* é um produto do Google Research, área de pesquisas científicas do Google. O *Colab* permite que qualquer pessoa escreva e execute código *Python* arbitrário pelo navegador e é especialmente adequado para aprendizado de máquina, análise de dados e educação (COLAB, 2022).

Por utilizar a computação em nuvem, o processamento do código desenvolvido é realizado pelas máquinas da Google ao invés de localmente. Isto permite o treinamento das redes neurais mais complexas que demandam maior poder computacional utilizando GPUs mais potentes.

## 5.8 React

O *React* é uma biblioteca JavaScript de código aberto com foco em criar interfaces de usuário em páginas web baseado na criação de componentes encapsulados que gerenciam seu próprio estado, permitindo atualizar e renderizar de forma eficiente apenas os componentes necessários na medida em que os dados mudam (REACT, 2022).

## 5.9 FastAPI

*FastAPI* é um framework web moderno e rápido (de alto desempenho) para construir APIs baseadas em *Python*. Ela possui como principais características: velocidade em performance e em facilidade de programação, robustez, simplicidade e compacticidade do código, o que torna o desenvolvimento de APIs uma tarefa mais simples e com bom desempenho (FASTAPI, 2022).

## 6 PROJETO E IMPLEMENTAÇÃO

A partir dos requisitos estabelecidos no capítulo 4 para a plataforma e para o classificador, propôs-se uma solução que os atendesse e, a seguir, são apresentadas as etapas do desenvolvimento do produto final obtido.

### 6.1 Aquisição do dataset

No intuito de obter um dataset robusto e com informações precisas, optou-se por utilizar fotografias tiradas a uma altura próxima ao nível do mar - como em plataformas petrolíferas ou navios em alto mar. Todavia, a necessidade de que seja conhecida a velocidade do vento ou a altura das ondas em cada registro limita a disponibilidade de *datasets* convenientes.

Foi empregado o catálogo *Stereo-Image Dataset* apresentado em (GUIMARÃES et al., 2020), o qual é composto por doze gravações feitas em diferentes momentos do dia por duas câmeras sincronizadas e posicionadas lado a lado em institutos de pesquisa localizados nos mares Negro (*Black Sea - “BS”*), Adriático (*Adriatic Sea - “AA”*), Amarelo (*Yellow Sea - “YS”*) ou Iroise (*Iroise Sea - “LJ”*). Desse modo, dispôs-se de doze sequências de pares de imagens (imagens *stereo*) capturadas a uma taxa entre 10 e 15 Hz. Uma vez que imagens consecutivas são praticamente iguais e, conseqüentemente, não agregam novas informações relevantes ao classificador, criou-se um *script* que baixasse imagens em intervalos de 1 em 1 segundo.

Tais imagens foram armazenadas no Google Drive e rotuladas segundo a Escala Beaufort a partir dos dados relativos à altura média das ondas que descrevem cada conjunto de imagens. Como explicitado na Tabela 4, verificou-se uma baixa diversidade de categorias das imagens, havendo poucas ou nenhuma amostra de várias classes, o que tornaria o modelo incapaz de obter um bom desempenho. Assim, simplificou-se o problema de 13 classes em apenas três: calmo/leve (1), moderado/agitado (2) e forte/extremo (3), os quais correspondem, respectivamente, aos graus originais 0 - 3, 4 - 7 e 8 - 12.

Tabela 4: Especificações dos dados do *dataset Stereo*

Gravação	Imagens baixadas	Total	Classe original	Classe simplificada
AA 1	3597	86322	4	2
AA 2	1798	43146	4	2
AA 3	1796	43104	4	2
YS	600	12000	4	2
BS 1	722	21658	1	1
BS 2	1198	28750	2	1
BS 3	1798	43148	2	1
BS 4	1798	43150	2	1
BS 5	1798	35962	3	1
BS 6	1798	43156	2	1
BS 7	1798	43156	3	1
LJ	1800	36000	10	3

Adicionalmente, utilizou-se o *dataset Manzoor-Umair Sea State Image Dataset* (MU-SSiD), introduzido pelo paper *A Novel Deep Learning Model for Sea State Classification Using Visual-Range Sea Images* (UMAIR et al., 2022). Ele é constituído por imagens da superfície de dois corpos d’água localizados no oeste da Malásia: o Estreito de Malaca e o Mar da China Meridional. Tais dados foram coletados entre Fevereiro de 2020 e Abril de 2021 durante dez experimentos conduzidos em terra - em plataformas estáveis que se estendiam sobre o mar ou próximas da costa - ou a bordo de um navio de cruzeiro.

Em cada ocasião, foram gravados múltiplos vídeos de diferentes durações e foi medida a velocidade média do vento correspondente. A partir de uma análise manual, os autores pré-selecionaram os registros que detinham características visuais representativas do grau da Escala Beaufort previamente estimado. Das gravações resultantes, eles extraíram imagens a uma taxa de quadros adequada e excluíram as que continham um excesso de fundo (céu) ou objetos em primeiro plano (como embarcações). Por fim, eles disponibilizaram mais de 100 mil imagens das classes 1 a 4 da Escala Beaufort, o que representa as classes 1 e 2 do problema simplificado. Para não inflar demasiadamente essas classes, agravando o desbalanceamento dos dados, empregou-se somente as imagens armazenadas nas pastas *Validation* e *Testing*, conforme detalhado na Tabela 5.

Tabela 5: Especificações dos dados MU-SSiD

Imagens baixadas	Total	Classe original	Classe simplificada
7200	25200	1	1
7200	25200	2	1
7200	25200	3	1
7200	25200	4	2

Ademais, levantou-se alternativas para expandir a base de dados. Dentre as opções estão a aplicação de técnicas de *data augmentation* - como transformações geométricas e *kernels* as quais geram variações das imagens com deslocamentos e rotações, por exemplo.

## 6.2 Estudo de modelos

Com o *dataset* em mãos, o passo seguinte foi o estudo de modelos de Aprendizado de Máquina para a classificação das imagens, bem como o entendimento do “estado do mar” e da escala Beaufort. Para tal, buscou-se modelos de CNNs utilizadas em artigos relacionados a este problema ou em problemas similares como: identificações de manchas de óleo na superfície do mar, identificação do clima a partir da vista do horizonte, etc.

A abordagem de classificação a partir de imagens adotada por *K. Zhang, Z. Yu and L. Qu*, “*Application of Deep Learning in Sea States Images Classification*,”2021 (ZHANG; YU; QU, 2021) foi tomada como base, acrescentando-se a experimentação de outras arquiteturas de redes convolucionais pré-treinadas e experimentando-se possíveis combinações de arquiteturas. Ademais, averiguou-se o impacto da aplicação de técnicas para o tratamento de *datasets* desbalanceados.

## 6.3 Seleção de candidatos a modelo

A extração de *features* consiste na passagem de cada imagem pelo conjunto de camadas convolucionais e de *pooling* da CNN em que são geradas, como saída, um mapa de atributos com as características mais relevantes da matriz de entrada (correspondente à imagem).

Dada à complexidade estrutural da rede neural e o aumento do número de parâmetros, usualmente treinar uma CNN do início (camadas convolucionais, de *pooling* e *fully-connected layers*) depende uma quantidade de tempo muito elevada. No intuito de melhorar a taxa de utilização das redes neurais e reduzir o consumo de tempo, fez-se

necessária a técnica de transferência de aprendizado (*transfer learning*). O principal objetivo desta técnica é aproveitar um modelo generalista treinado anteriormente em uma base de dados robusta para servir como base para novos modelos que podem reutilizar alguns de seus parâmetros ou partir deles para o cálculo de novos parâmetros em um tempo menor. Ao usar o modelo pré-treinado para resolver problemas semelhantes, não é necessário otimizar muitos parâmetros, mas apenas realizar ajustes finos, que podem alcançar bons resultados e reduzir o tempo usado para treinar o modelo (ZHANG, YU, QU, 2021).

Portanto, foram selecionadas 3 diferentes arquiteturas pré-treinadas no *ImageNet* - um *dataset* que contém milhões de imagens rotuladas manualmente de diversas categorias diferentes, tais como, cachorros, aviões, balões, etc - que possuem boa capacidade de generalização para imagens fora do *dataset* em que foram treinadas, a fim de selecionar o melhor modelo para aplicação deste projeto. Dentre os candidatos estão a VGG-16, ResNet-34 e a ResNet-152.

Ademais, a partir dos resultados obtidos nos testes destas arquiteturas, explicitados no capítulo 7 deste documento, notou-se que a VGG-16 desempenhava bem em um dos cenários de teste (1. Imagens do *dataset Stereo*), enquanto a ResNet-34, em outro (2. Imagens representantes da escala Beaufort). Desta forma, decidiu-se por criar uma nova arquitetura a partir da combinação destas duas, a fim de tentar unir os pontos fortes de cada uma e obter uma junção capaz de performar bem em ambos os cenários.

Tal combinação, denominada pelo grupo como ResNet34-VGG, consistiu em utilizar a camada de extração de *features* da ResNet-34 e uma camada de classificação baseada na *fully connected layer* da VGG-16. Realizou-se a combinação desta forma, uma vez que, dada a simplicidade da *fully connected layer* da ResNet-34 e a observação do desempenho da ResNet-152, identificou-se que a camada de extração de *features* era a responsável pelo melhor desempenho da arquitetura em um dos cenários de teste (2). Paralelamente, dado o fato de que o desempenho da VGG-16 neste mesmo cenário foi o pior entre as 3 arquiteturas somado ao seu desempenho superior no outro cenário de teste (1), concluiu-se que a camada de classificação foi a responsável pela sua melhor performance.

A arquitetura ResNet34-VGG é detalhada na Figura 13.

Figura 13: Arquitetura da ResNet34-VGG



Fonte: os autores

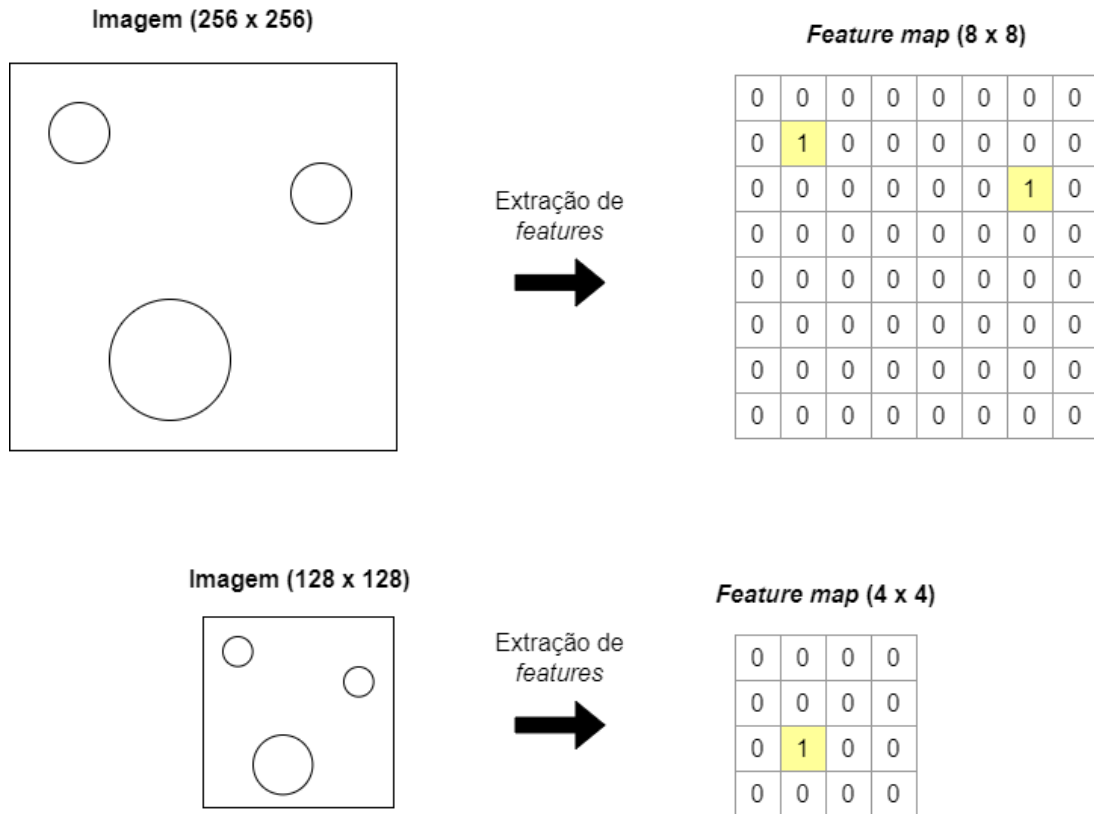
## 6.4 Pré-processamento das imagens

As imagens originais disponibilizadas em (GUIMARÃES et al., 2020) possuíam dimensões 2456 x 2058. A fim de torná-las quadradas - isto é, de largura e altura iguais - recortou-se tais figuras no centro (*center crop*), obtendo imagens de tamanho 2058 x 2058. Em seguida, optou-se por redimensioná-las para 224 x 224 (*resize*), devido às limitações computacionais e às redes pré-treinadas adotadas. No que tange ao *dataset* MU-SSiD, suas imagens já foram disponibilizadas pelos seus autores nas dimensões 224 x 224.

Imagens maiores, de 600 x 600, por exemplo, acarretaram o esgotamento da memória RAM no início da fase de treinamento dos modelos e se mostraram inviáveis. Ademais, como mencionado anteriormente, a VGG-16, ResNet-34 e a ResNet-152 foram treinadas previamente com imagens de 224 x 224 do *ImageNet*. Isso significa que, apesar de aceitarem entradas de outras dimensões, tais modelos aprenderam a identificar padrões de tamanhos específicos e que podem ser reconhecidos ou não a depender da largura e altura do *input*.

Para demonstrar a relevância das dimensões da entrada, considera-se uma CNN capaz de distinguir círculos com 40 px de diâmetro. Observa-se na Figura 14 que, para uma entrada de 256 x 256, o modelo é capaz de encontrar os dois círculos superiores, ao passo que com a redução para 128 x 128, somente o círculo inferior passa a ser localizado.

Figura 14: Influência das dimensões da entrada no reconhecimento de padrões



Fonte: os autores

Uma vez que as arquiteturas selecionadas assumiam uma entrada com três canais, foi necessário converter as imagens apropriadamente. As imagens *stereo*, tiveram o valor do seu único canal replicado para os demais, construindo, assim, um modelo RGB. As imagens MU-SSiD, por sua vez, deixaram de ser coloridas e foram transformadas em *grayscale* com três canais.

*Data Augmentation* é uma técnica para aumentar a quantidade e a qualidade de um conjunto de dados (SHORTEN; KHOSHGOFTAAR, 2019). Desta forma, a fim de melhorar a capacidade de generalização do modelo frente a novas imagens em padrões diferentes daqueles vistos no treinamento, o *data augmentation* foi aplicado tanto para o conjunto de treinamento quanto para o de validação, utilizando técnicas como transformação geométrica, filtros de kernel, mistura de imagens, apagamento aleatório e transformações como em (OLSEN et al., 2019). As técnicas aplicadas são elencadas na Tabela 6 abaixo. As técnicas são aplicadas aleatoriamente às imagens, ou seja, cada imagem pode possuir várias, algumas ou nenhuma das técnicas aplicadas.



Tabela 6: Técnicas de *data augmentation* utilizadas em cada catálogo

<b>Técnica</b>	<b><i>Stereo</i></b>	<b>MU-SSiD</b>
Espelhamento horizontal	X	X
Rotação	X	X
Brilho e contraste		X
Desfoque de movimento		X
Injeção de ruído		X
Deslocamento		X
Embaralhamento aleatório de partes da imagem		X
Correspondência de histograma		X
Equalização do histograma		X

Por fim, normalizou-se todas as imagens de modo que os valores de cada pixel pertencessem ao intervalo entre -1 e 1. A normalização ajuda a manter a consistência do gradiente calculado na fase de *backpropagation*, impedindo que ele se torne excessivamente grande. Isso ocorre visto que todas as *features* possuem valores de grandeza similar, acelerando a convergência do modelo.

## 6.5 Treinamento

O treinamento dos modelos das arquiteturas escolhidas foi realizado em diferentes situações a fim de averiguar a qualidade de técnicas implementadas e obter uma variedade de modelos que pudessem ser comparados entre si a fim de obter o melhor modelo possível.

A fim de explorar o máximo potencial oferecido pelos *datasets*, dadas as suas características de baixa variabilidade entre as imagens dentro de cada *dataset* e a grande diferença na quantidade de imagens representantes de cada classe, cada um dos *datasets* foi dividido em 50% para testes e 50% para o desenvolvimento do modelo. As imagens utilizadas para o desenvolvimento do modelo foram separadas, aleatoriamente, em 80% para o treinamento e 20% para validação. Com isso, os seguintes passos foram adotados para o processo de treinamento dos modelos: técnicas para *datasets* desbalanceados, escolha de hiper parâmetros de treinamento e desenvolvimento do modelo.

### 6.5.1 Técnicas para tratar *datasets* desbalanceados

Como observado na Tabela 7, o *dataset* é extremamente desbalanceado, sendo a quantidade de representantes da classe 1 cerca de 5,6 vezes a da classe 3. Neste cenário, dado que as redes neurais aprendem observando pontos de dados pertencentes a diferentes classes, a capacidade de classificação das classes com as menores frequências se viu prejudicada e isso ocasionou o *overfitting* das classes com mais amostras.

Tabela 7: Frequência por classe

Classe	Quantidade de imagens
Classe 1	10910
Classe 2	7791
Classe 3	1800

No intuito de tentar solucionar este problema, duas técnicas de tratamento de *datasets* desbalanceados foram testadas para o treinamento de cada uma das arquiteturas: o ***oversampling* das classes minoritárias** e o ***weighted loss***. O ***oversampling* das classes minoritárias** consiste em balancear a quantidade de imagens das classes a fim de que cada classe possua, aproximadamente, a mesma quantidade de imagens. Para tal aumenta-se aleatoriamente o número de observações que são cópias de amostras existentes das classes com menos observações (como a classe 3 neste problema).

Já o ***weighted loss*** implementa a ideia de punir com maior severidade o modelo quando comete um erro ao classificar uma amostra da classe minoritária. Para tanto, utilizou-se o inverso da frequência de cada classe como pesos para a função de perda de cada classe, ou seja, a classe 3 possui o maior peso enquanto a classe 1, o menor. Isto posto, valoriza-se mais as imagens da classe 3 de modo que os parâmetros do modelo (i.e. os pesos de cada nó) se configurem de modo a favorecer a classe 3 e classificar imagens que não se enquadram nas demais classes como pertencentes a esta classe minoritária.

### 6.5.2 Escolha de hiper parâmetros de treinamento

Para o treinamento dos modelos, é necessária a definição de alguns hiper parâmetros que impactam no desenvolvimento do modelo e previnem que ele aprenda apenas com os dados mostrados (*overfitting* e *underfitting*), tornando-o capaz de generalizar para outras situações possíveis. Os hiper parâmetros aqui definidos foram a **taxa de aprendizado** (*learning rate*) e o **número de épocas**, descritos na Tabela 8.

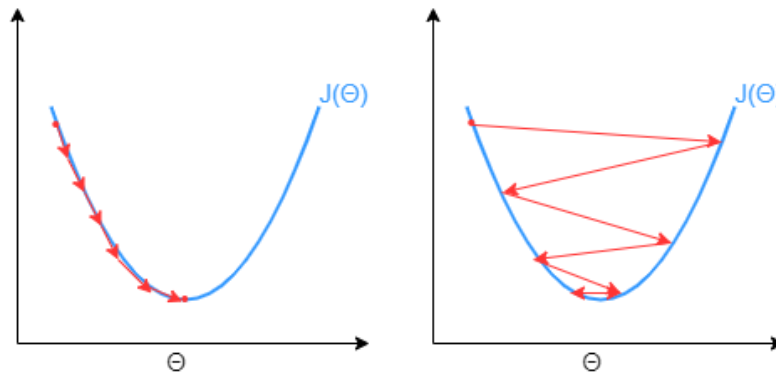
Tabela 8: Hiper parâmetros selecionados

Hiper parâmetro	Valor
Taxa de aprendizado	0.001
Número de épocas	10

### 6.5.2.1 Taxa de aprendizado

A **taxa de aprendizado** é um hiper parâmetro que define o quanto os pesos de uma rede são ajustados em relação ao gradiente de perda. Quanto menor o valor, mais lentamente a variação se desloca ao longo da inclinação descendente, o que, apesar de garantir que nenhum mínimo local será perdido (usando uma baixa taxa de aprendizado), também pode significar que levará muito tempo para convergir - especialmente se ficar preso em uma região de platô - ou que, até mesmo, não convirja (ZULKIFLI, 2018). A Figura 15 demonstra um exemplo de como o gradiente se comporta a depender da taxa de aprendizado.

Figura 15: Gradiente descendente com uma taxa de aprendizado pequena (esquerda) e grande (direita).



Fonte: os autores

Após experimentar alguns valores, optou-se pela taxa de aprendizado de  $0.001$ , visto que ele garantiu a convergência em poucas épocas, enquanto o valor de, por exemplo,  $0.01$ , não convergia e possuía um comportamento de uma taxa de aprendizado elevada como o da figura 15

### 6.5.2.2 Número de épocas

Uma época consiste em passar o *dataset* de treino inteiro uma única vez pela rede neural realizando o *forward*, em que cada entrada passa do início ao fim da rede, e o *back propagation*. A fim de prevenir o *overfitting*, adicionalmente utilizou-se uma etapa de validação após a passagem do *dataset* de treino a fim de computar a acurácia do modelo em um conjunto de dados não visto previamente de modo que, ao longo das épocas, o modelo com a melhor acurácia seja selecionado.

Uma vez que o conjunto de dados é limitado, a fim de otimizar o aprendizado, é utilizado o gradiente descendente, o qual é um processo iterativo, de modo que, atualizar os pesos da rede com uma única passagem ou uma época não é suficiente. Portanto, ao observar o treinamento das arquiteturas, notou-se que 10 épocas eram suficientes para que os modelos convergissem, dado que os mesmos atingiam sua acurácia máxima por volta da sexta ou da sétima época.

### 6.5.3 Desenvolvimento dos modelos

Definidos os hiper parâmetros e as técnicas para *datasets* desbalanceados, foram definidos 4 cenários de treinamento para cada arquitetura a fim de aferir o impacto das técnicas de balanceamento e o impacto do *transfer learning* utilizando redes pré-treinadas ao invés de promover o treinamento das camadas convolucionais do zero. São os 4 cenários:

1. **Arquitetura pré-treinada:** utiliza-se a rede pré-treinada no Image Net congelando-se os pesos das camadas de extração de *features* e, durante o treinamento, atualizando-se os pesos somente da *fully connected layer*;
2. **Arquitetura sem pré-treino:** durante o treinamento todos os pesos são atualizados - tanto os pesos das camadas de extração de *features* quanto os pesos da *fully connected layer*;
3. **Arquitetura pré-treinada utilizando *oversampling*:** utiliza-se a rede pré-treinada no Image Net congelando-se os pesos das camadas de extração de *features* e, durante o treinamento, atualizando-se os pesos somente da *fully connected layer*. Juntamente é utilizada a técnica de *oversampling* das classes minoritárias;
4. **Arquitetura pré-treinada utilizando *weighted loss*:** utiliza-se a rede pré-treinada no Image Net congelando-se os pesos das camadas de extração de *features*

e, durante o treinamento, atualizando-se os pesos somente da *fully connected layer*. Juntamente é utilizada a técnica de *weighted loss*;

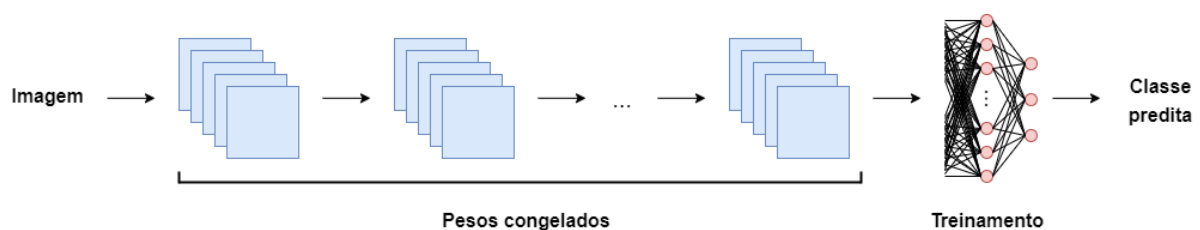
O algoritmo de treinamento foi implementado em Python, dada a grande diversidade de ferramentas e bibliotecas de Aprendizado de Máquina. Para desenvolver os modelos foi utilizado um *Jupyter Notebook*, ferramenta que permite um desenvolvimento incremental, sendo que, em cada etapa, é possível pausar a execução e gerar visualizações dos dados e das métricas para a análise.

Conforme mencionado anteriormente, cada um dos *datasets* presentes em (GUIMARÃES et al., 2020) foi dividido em 50% para testes e 50% para o desenvolvimento do modelo. As imagens utilizadas para o desenvolvimento do modelo foram separadas, aleatoriamente, em 80% para o treinamento e 20% para validação.

Para cada época, os dados de treinamento são utilizados como entradas no modelo, que gera uma predição e a partir dela mede-se sua divergência para o resultado esperado. Com esse erro calculado, alteram-se ligeiramente os pesos do modelo a fim de encontrar uma combinação de parâmetros que diminua o erro para todas as entradas inseridas (*backpropagation*). Após todos os dados de treinamento passarem uma única vez pela rede, caracterizando uma época, inicia-se a etapa de validação da época em que os dados de validação são utilizados como entrada para analisar o desempenho do modelo na classificação de imagens que não foram utilizadas na etapa de treinamento. Deste modo, avalia-se o desempenho do modelo quanto à sua capacidade de predição e generalização.

Ao realizar a etapa de treinamento, quando optou-se por aplicar o *transfer learning* aproveitando o modelo pré-treinado no Image Net (caso dos cenários 1, 3 e 4 descritos acima), congelou-se os parâmetros das camadas de extração de *features* a fim de que estes não fossem atualizados durante o *backpropagation*. Isto posto, apenas os pesos da *fully connected layer* do modelo foram atualizados, conforme representado na Figura 16.

Figura 16: Representação do *transfer learning* com parâmetros de extração de *feature* congelados



Fonte: os autores

Este processo com as etapas de treinamento e validação configura uma época. Para cada época o processo é executado novamente e, ao final da época, caso a acurácia do modelo obtido na época seja maior que a melhor acurácia até então observada, avaliadas no set de validação, o novo modelo é salvo e ele passa a ser o melhor modelo. Ao final das 10 épocas, o modelo com a melhor acurácia será o melhor modelo e é aquele que será definido como o modelo final a ser utilizado. Este método de desenvolvimento do modelo foi realizado para todas as arquiteturas em todos os 4 cenários supracitados.

O processo de treinar um modelo é dispendioso, demandando muito poder computacional dada a grande quantidade de imagens e a necessidade de GPUs para realizar as operações de álgebra linear inerentes ao processo de treinamento. Desta forma, dado que os notebooks do Google Colab não ofereciam poder computacional suficiente no plano gratuito para realizar o treinamento, optou-se por utilizar uma máquina presente no TPN USP (Tanque de Provas Numérico) cujas especificações principais se encontram na Tabela 9.

Tabela 9: Especificações técnicas da máquina do TPN

64 processadores AMD EPYC 7502P 32-Core Processor
RAM 64 Gb
3 GPUs

No intuito de acessar as máquinas remotamente, foi necessário obter acesso à VPN utilizada pelo TPN e então conectar-se à máquina via SSH. Uma vez conectado, dado que diferentes pessoas utilizam os computadores do TPN para diferentes projetos, é importante garantir que cada projeto utilize as dependências corretas (versões de bibliotecas, pacotes externos) sem comprometer os demais projetos. Para tal utilizou-se o Docker, uma ferramenta para empacotar aplicações com todas as dependências necessárias para que usuários consigam executá-las de modo simples em diversos hardwares diferentes, criando um contêiner que é responsável por criar este empacotamento e por executar o script de treinamento da rede.

Ao final da etapa de treinamento, obteve-se os modelos para cada combinação entre cada a arquitetura e cada um dos cenários descritos anteriormente, os quais são expressos na Tabela 10 abaixo. Além das arquiteturas originais da VGG-16, da ResNet-34 e da ResNet-152, também há a nova arquitetura proposta pelo grupo em que se combina a camada de extração de *features* da ResNet-34 com uma *fully connected layer* inspirada na VGG-16, conforme explicado anteriormente. Esta arquitetura, assim como as demais,

também foi treinada nos cenários 1, 3 e 4, mas não no cenário 2, uma vez que o desempenho das arquiteturas originais neste cenário foi inferior aos demais.

Tabela 10: Modelos resultantes para cada cenário e arquitetura

Nome do modelo	Cenário	Arquitetura
vgg16_pretrained	Pré-treinada	VGG-16
vgg16_not_pretrained	Sem pré-treino	VGG-16
vgg16_oversample	Pré-treinada com <i>oversampling</i>	VGG-16
vgg16_weighted_loss	Pré-treinada com <i>weighted loss</i>	VGG-16
resnet152_pretrained	Pré-treinada	ResNet-152
resnet152_not_pretrained	Sem pré-treino	ResNet-152
resnet152_oversample	Pré-treinada com <i>oversampling</i>	ResNet-152
resnet152_weighted_loss	Pré-treinada com <i>weighted loss</i>	ResNet-152
resnet34_pretrained	Pré-treinada	ResNet-34
resnet34_not_pretrained	Sem pré-treino	ResNet-34
resnet34_oversample	Pré-treinada com <i>oversampling</i>	ResNet-34
resnet34_weighted_loss	Pré-treinada com <i>weighted loss</i>	ResNet-34
resnet34_vgg_pretrained	Pré-treinada	ResNet34-VGG
resnet34_vgg_oversample	Pré-treinada com <i>oversampling</i>	ResNet34-VGG
resnet34_vgg_weighted_loss	Pré-treinada com <i>weighted loss</i>	ResNet34-VGG

Após obter os resultados dos testes para estes modelos (que são descritos na seção seguinte), selecionou-se os 4 melhores dentre eles e treinou-os acrescentando aos conjuntos de treinamento e validação as imagens do *dataset* MU-SSiD armazenadas na pasta *Validation*, as quais já contavam com observações geradas a partir das técnicas de *image augmentation* explicitadas anteriormente. O novo treinamento se deu nos mesmos moldes do anterior - 80% das imagens para treino e 20% para validação, atualizando-se apenas os pesos da *fully connected layer*, mantendo congelados os pesos da camada de extração de *features*. Os modelos finais obtidos se encontram na Tabela 11 abaixo.

Tabela 11: Modelos resultantes para cada cenário e arquitetura utilizando as novas imagens de MU-SSiD no treinamento

Nome do modelo	Cenário	Arquitetura
vgg16_pretrained_kaggle	Pré-treinada	VGG-16
resnet152_pretrained_kaggle	Pré-treinada	ResNet-152
resnet34_oversample_kaggle	Pré-treinada com <i>oversampling</i>	ResNet-34
resnet34_vgg_weighted_loss_kaggle	Pré-treinada com <i>weighted loss</i>	ResNet34-VGG

## 6.6 Testes

A última etapa consiste nos testes aos quais os modelos classificatórios são submetidos para avaliar o seu desempenho utilizando-se a validação cruzada e comparando a performance destes modelos entre si segundo a acurácia, a precisão, a revocação e o *F1-score*.

Foram elaborados dois macro cenários de testes que são pautados pelo conjunto de imagens utilizado no treinamento dos modelos: no primeiro utilizou-se apenas imagens do *dataset Stereo* no treinamento e no segundo utilizou-se tanto as imagens do *dataset Stereo* quanto as imagens do *dataset* MU-SSiD, conforme explicado na seção anterior.

### 6.6.1 Modelos treinados no *dataset Stereo*

Cada uma das gravações do *dataset Stereo* foi dividida em 50% para desenvolvimento do modelo e 50% para os testes. Neste macro cenário de teste, três micro cenários foram elaborados para aferir a capacidade de classificação e generalização dos modelos desenvolvidos para diferentes conjuntos de imagens.

#### 6.6.1.1 Imagens do *dataset Stereo*

Neste micro cenário, utilizou-se os 50% das imagens do *dataset Stereo* reservados para testes. Para cada gravação deste *dataset* (vide Tabela 4), calculou-se a acurácia de cada modelo a partir do número de acertos de classificação na gravação. Adicionalmente, utilizou-se a ferramenta de *classification report* do *SciKit Learn* para medir a acurácia, a precisão, a revocação e o *F1-score*, bem como para computar a matriz de confusão de cada um dos modelos listados na Tabela 10.



### 6.6.1.2 Imagens representantes da escala Beaufort

Neste micro cenário utilizou-se as imagens representantes da escala Beaufort (RAYNAL; DOERRY, 2010) presentes na Figura 17 para o teste. Apesar de ser um conjunto muito pequeno de imagens (apenas 13), dada a escassez de imagens rotuladas segundo os graus da escala Beaufort, optou-se por utilizá-lo para fins de comparação em capacidade de generalização de imagens mais diversas com relação às imagens de treino. Utilizou-se a ferramenta de *classification report* do *SciKit Learn* para medir a acurácia, a precisão, a revocação e o *F1-score*, bem como para computar a matriz de confusão de cada um dos modelos listados na Tabela 10.

Figura 17: Imagens representantes dos graus da escala Beaufort



Fonte: (RAYNAL; DOERRY, 2010)

Ao final deste teste, comparou-se a acurácia geral de cada modelo entre este micro cenário e o cenário *Imagens do dataset Stereo* e selecionou-se o melhor modelo de cada arquitetura (incluindo a arquitetura híbrida ResNet34-VGG), totalizado 4 modelos, os quais estão descritos na Tabela 12 a seguir. O critério para esta seleção foi o equilíbrio

da performance entre os dois cenários, isto é, escolher o modelo que maximiza ambas as acurácias.

Tabela 12: Melhor modelo de cada arquitetura

Nome do modelo	Cenário	Arquitetura
vgg16_pretrained	Pré-treinada	VGG-16
resnet152_pretrained	Pré-treinada	ResNet-152
resnet34_oversample	Pré-treinada com <i>oversampling</i>	ResNet-34
resnet34_vgg_weighted_loss	Pré-treinada com <i>weighted loss</i>	ResNet34-VGG

Vale ressaltar que, ao comparar as acurácias entre os dois cenários, inicialmente haviam apenas representantes das arquiteturas VGG-16, ResNet-34 e ResNet-152 e, após a análise dos resultados desta comparação, observou-se que a VGG-16 desempenhava melhor no cenário das imagens do *dataset Stereo*, enquanto a ResNet-34, no cenário das imagens representantes da escala Beaufort, conforme explicado anteriormente. Desta análise surgiu a ideia de criar a arquitetura híbrida ResNet34-VGG, a qual foi testada, posteriormente, nestes dois cenários.

### 6.6.1.3 Imagens do *dataset* MU-SSiD

O último micro cenário utiliza as imagens da pasta *Testing* do *dataset* MU-SSiD, às quais não foram aplicadas quaisquer técnicas de *image augmentation*. A partir dos 4 melhores modelos obtidos após os testes anteriores presentes na Tabela 12, utilizou-se a ferramenta de *classification report* do *SciKit Learn* para medir a acurácia, a precisão, a revocação e o *F1-score*, bem como para computar a matriz de confusão.

Optou-se por realizar este teste apenas nos 4 melhores modelos devido ao lançamento tardio do paper que contém este *dataset*, o qual ocorreu muito após a etapa de aquisição de *datasets* pelo grupo, de modo que treinar novamente todos os modelos utilizando as novas imagens demandaria muito tempo e impactaria no planejamento do projeto. Ademais, o desempenho dos modelos que não foram testados, por serem inferiores aos modelos escolhidos, dificilmente seria representativamente superior.

## 6.6.2 Modelos treinados nos *datasets* Stereo e MU-SSiD

Cada uma das gravações do *dataset Stereo* foi dividida em 50% para desenvolvimento do modelo e 50% para os testes. No desenvolvimento do modelo também foram utiliza-

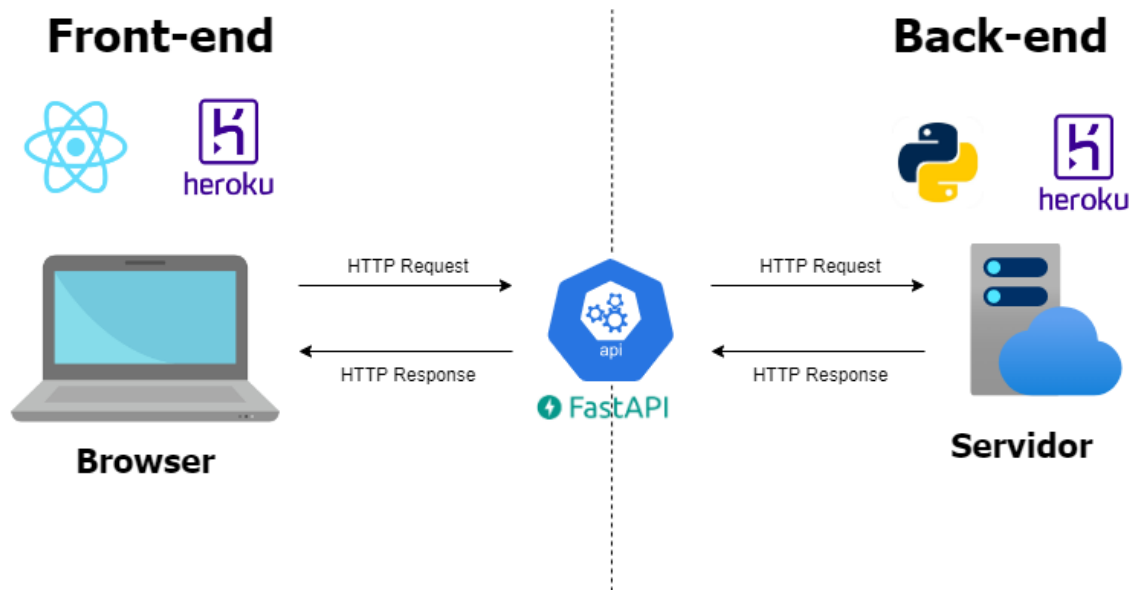
das as imagens da pasta *Validation* do *dataset* MU-SSiD. Neste macro cenário de teste, semelhantemente ao anterior, aferiu-se a acurácia, a precisão, a revocação e o *F1-score*, bem como para computou-se a matriz de confusão utilizando a ferramenta de *classification report* do *SciKit Learn* para os 4 melhores modelos (vide Tabela 11 na seção de *Treinamento*) para os 3 micro cenários de teste, isto é, teste com as imagens destinadas para teste do *dataset Stereo*, teste com as imagens representativas da escala Beaufort e teste com as imagens da pasta *Testing* do *dataset* MU-SSiD.

O intuito deste macro cenário é verificar o impacto da adição das novas imagens na capacidade preditiva dos modelos, ou seja, se as novas imagens melhoram, pioram ou não impactam na performance classificatória de cada modelo para cada um dos conjuntos de imagens de teste. A partir destas novas informações, comparou-se o desempenho das 4 arquiteturas na classificação dos 3 conjuntos de imagens entre antes (macro cenário com treinamento apenas em imagens do *dataset Stereo*) e depois (macro cenário com treinamento em imagens do *dataset Stereo* e do *dataset* MU-SSiD) da adição das novas imagens. A partir dos resultados, foi selecionado o modelo que possui o melhor equilíbrio na acurácia dos 3 conjuntos de imagens e que consegue classificar suficientemente bem imagens da classe 3, dado que a adição das novas imagens no treinamento aumentou ainda mais o desbalanceamento dos *datasets*, fazendo com que a proporção de imagens da classe 3 se tornasse ainda menor do que as demais.

## 6.7 Implementação da plataforma

Com o modelo do classificador selecionado e devidamente testado, o passo seguinte foi torná-lo acessível e intuitivo para o usuário final. Deste modo, optou-se por utilizar uma arquitetura cliente/servidor dividida entre front-end e back-end conforme demonstrado na Figura 18 abaixo.

Figura 18: Arquitetura em camadas do projeto com front-end e back-end.



Fonte: os autores

Nesta abordagem, a lógica do processamento e classificação da imagem de entrada é realizada no servidor (back-end) que contém uma instância da arquitetura do modelo selecionado e foi desenvolvido usando *Python*. No front-end encontra-se a interface com o usuário, desenvolvida utilizando *React*, em que é possível realizar o *upload* da imagem, bem como visualizar os resultados da classificação e informações relativas ao classificador que estão descritas no capítulo 4.

A comunicação entre front e back-end ocorre através de uma API RESTful desenvolvida utilizando a biblioteca Fast API. Uma API REST consiste em uma interface de programação de aplicações que possui um conjunto de definições e protocolos usado no desenvolvimento e na integração de aplicações.

Deste modo, o front-end envia uma solicitação HTTP para o back-end através da API contendo a imagem a ser classificada codificada em Base64 e, em seguida, o servidor é responsável por utilizar um *script* que processa e classifica a imagem utilizando o classificador desenvolvido. Por fim, o back-end envia uma resposta HTTP que contém os resultados da classificação para o front-end em formato JSON.

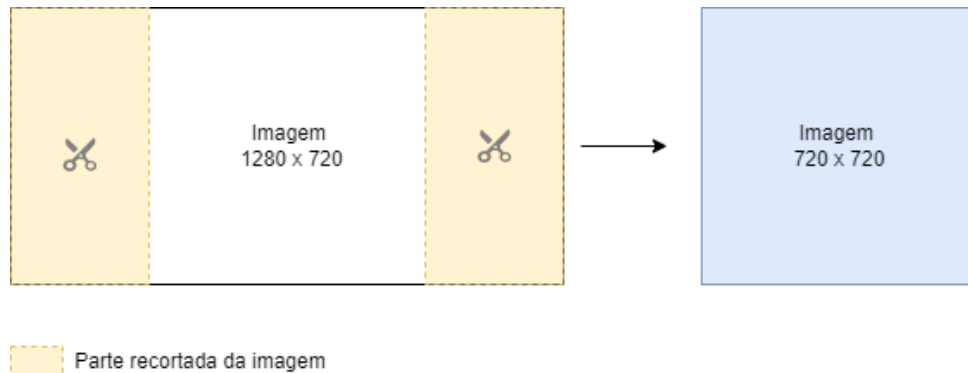
### 6.7.1 Back-end

Partindo do modelo classificatório selecionado `resnet34_oversample`, cujo desempenho e critérios para sua seleção são descritos no capítulo 7, desenvolveu-se, com a

linguagem de programação *Python*, um *script* de classificação das imagens da superfície do mar que utilizasse o modelo. Este *script* realiza a seguinte sequência de ações para classificar uma imagem:

1. Recebe uma imagem nos formatos `.PNG`, `.JPEG` ou `.TIFF` e converte-a em *grayscale* (i.e. imagem em preto e branco) garantindo que ela possua três canais (RGB);
2. Identifica a menor dimensão da imagem P&B e realiza um *center crop* segundo a menor dimensão, isto é, recorta de forma centralizada a imagem obtendo uma nova imagem quadrada com dimensões `menor dimensão x menor dimensão`, como ilustrado na Figura 19;

Figura 19: Operação de *center crop* em uma imagem de dimensões 1280x720 pixels, obtendo uma imagem final de dimensões 720x720 pixels.



Fonte: os autores

3. Redimensiona a imagem *croppada* para o tamanho 224x224 em que o modelo classificatório trabalha e converte-a em um tensor;
4. Realiza a classificação do tensor representante da imagem utilizando o modelo classificatório de estado do mar desenvolvido;
5. Ao final deste processo, o *script* retorna a classe à qual a imagem pertence (1, 2 ou 3).

Com o *script* de classificação em mãos, o passo seguinte foi o desenvolvimento da API responsável pela comunicação entre a interface e o servidor no qual é executado o *script* de classificação. Esta API RESTful foi desenvolvida utilizando o *Fast API*. Ela possui apenas um *endpoint* para a ferramenta, isto é, apenas um local em que essas solicitações (conhecidas como chamadas de API) são atendidas, que é o *endpoint* `/classificar`. Além dele há um *endpoint* que contém a documentação da API, o `/docs`.

A API recebe, através de uma requisição HTTP, uma imagem codificada em Base64, que consiste em um método para codificação de dados para transferência na Internet. Ao chegar na API, a imagem é então decodificada e convertida em seu formato original (.PNG, .JPEG ou .TIFF). Em seguida, executa-se o *script* de classificação utilizando esta imagem como entrada.

A partir da classe retornada pelo *script*, define-se um JSON (*JavaScript Object Notation*) de resposta contendo a classe predita para a imagem, bem como as informações que descrevem esta classe e quais foram as probabilidades encontradas pelo modelo classificatório para o pertencimento da imagem a cada uma das três classes. Este JSON é então retornado via resposta HTTP à interface. Um exemplo de JSON retornado pela API se encontra na Figura 20 a seguir.

Figura 20: Exemplo de um JSON retornado pela API para uma imagem da classe 3.

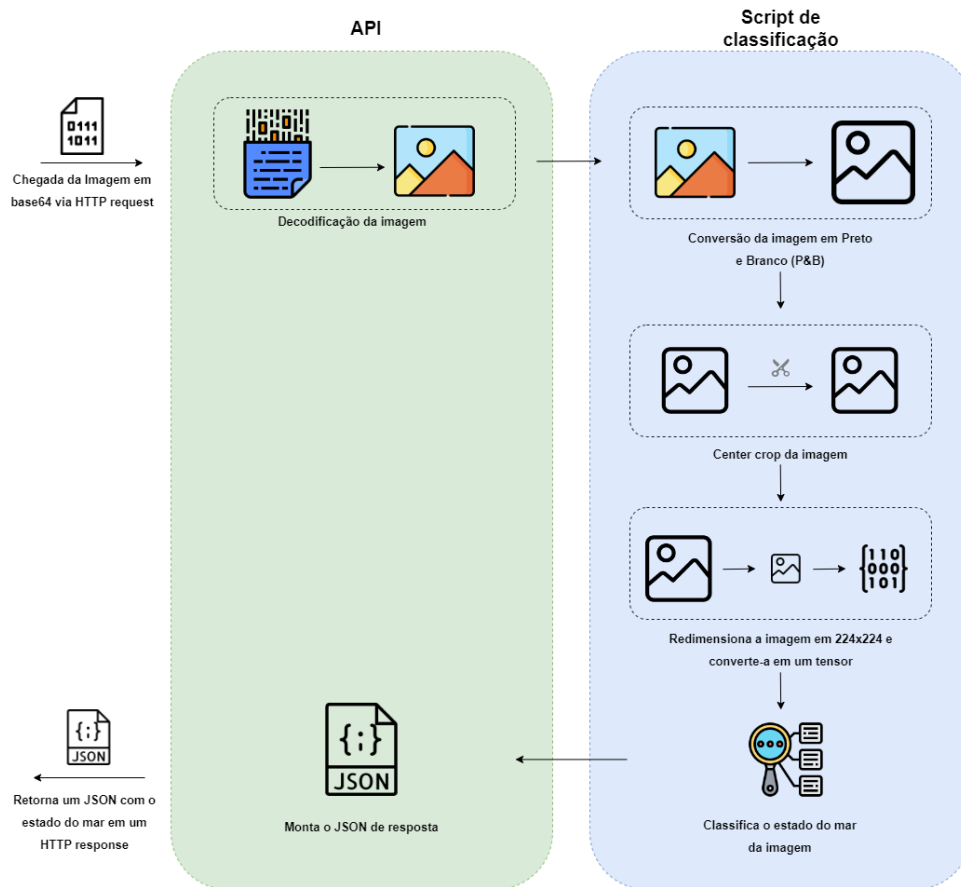


```
{
  "classe_id": 3,
  "classe": "8 - 12",
  "descricao": "Ventania a furacão",
  "velocidade_do_vento": "17,2 - 32,7 m/s",
  "altura_media_ondas": "5,5 - 14 m",
  "aspecto_mar": {
    "from": "Mar revolto com ondas moderadamente altas, rebentações e borrifos constantes; formação de faixas de espumas bem marcadas ao longo da direção do vento",
    "to": "Mar e ar preenchidos por espuma e borrifos; mar branco; visibilidade quase nula"
  },
  "probabilidades": [
    {
      "range": "Classes 0 - 3",
      "p": 0.1322353631258011
    },
    {
      "range": "Classes 4 - 7",
      "p": 0.6354637742042542
    },
    {
      "range": "Classes 8 - 12",
      "p": 99.23230743408203
    }
  ]
}
```

Fonte: os autores

O fluxo geral do back-end é ilustrado na Figura 21.

Figura 21: Fluxograma do funcionamento da API ao receber uma chamada.



Fonte: os autores

Finalmente publicou-se a API na plataforma Heroku no endereço <https://sea-state-classifier-api.herokuapp.com>.

## 6.7.2 Front-end

Primeiramente, a fim de projetar uma ferramenta de fácil utilização e definir os elementos a serem incluídos na interface, optou-se por conduzir um estudo de usuário por meio da entrevista, uma vez que esta captura qualitativamente a forma como um usuário representativo do grupo primário de usuários pensa. Assim, entrevistou-se o professor-doutor Eduardo Aoun Tannuri, docente e pesquisador na Poli-USP com pesquisas voltadas para previsões de condições oceânicas e outras áreas de interesse voltadas à exploração petrolífera.

De modo geral, compreendeu-se que o processo de definição do estado do mar para um vídeo ou imagem da superfície do mar é comparar manualmente imagens que retratam diferentes condições marítimas e alturas das ondas, analisando aspectos como graus de

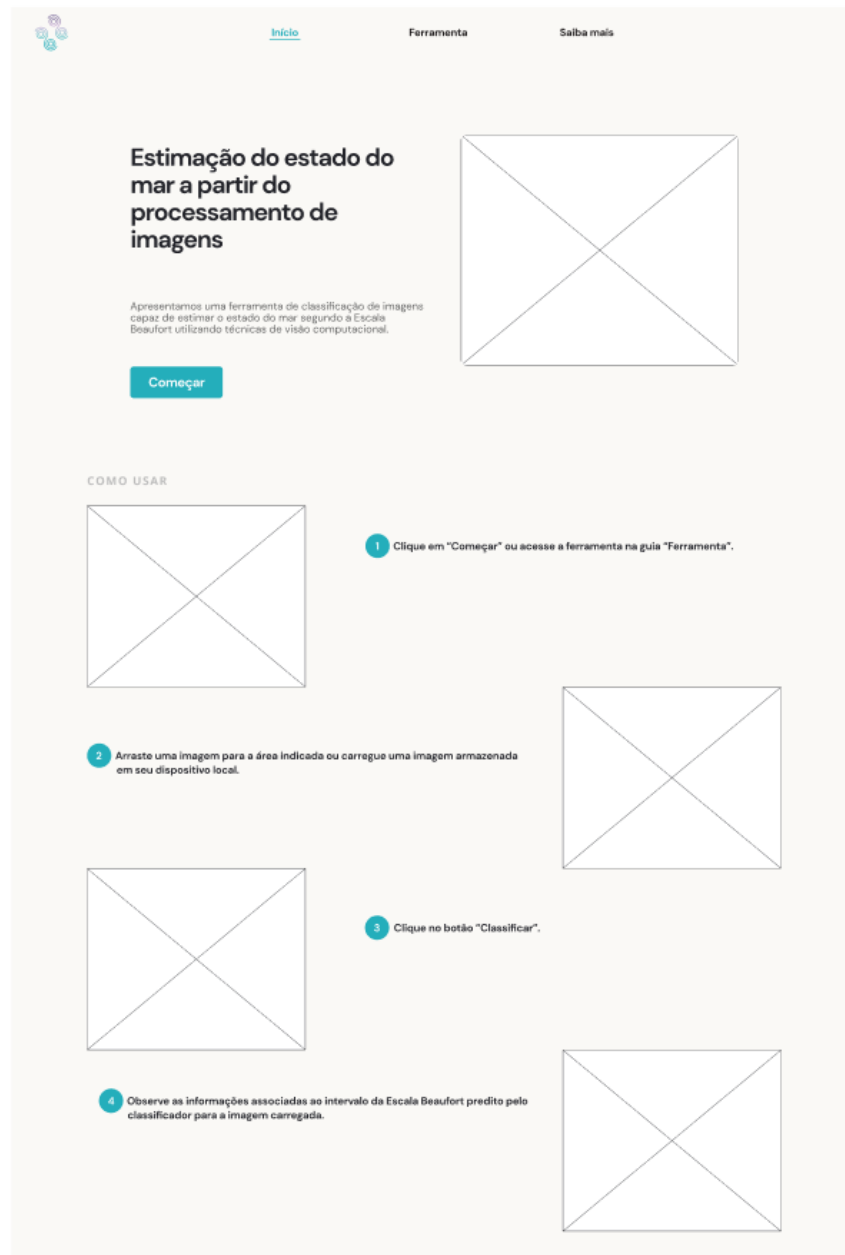
espuma, spray, rugosidade do mar, dentre outros. Alternativamente, pode-se empregar ondógrafos, sensores para medir a elevação da superfície. Ao propor uma aplicação que estime automaticamente o estado do mar correspondente a uma imagem, notou-se a relevância em exibir de forma clara o grau da Escala Beaufort associado, bem como a altura média aproximada das ondas. Ademais, para transmitir confiabilidade, é importante incluir uma base teórica que explique a metodologia adotada no desenvolvimento e inclua as especificações do modelo selecionado.

De posse de tais informações, criou-se um protótipo inicial focado no *design* da plataforma utilizando o editor gráfico Figma. O site é dividido em três abas principais:

- **Início:** apresenta brevemente a ferramenta e elenca instruções de como usá-la (Figura 22);
- **Ferramenta:** detém a área de *upload*, onde o usuário carrega a imagem a ser classificada (Figura 23). Ao receber do back-end o intervalo das classes da Escala Beaufort correspondente e suas especificações, exibe os resultados (Figura 24).
- **Saiba mais:** concentra de forma resumida o processo de desenvolvimento do modelo de visão computacional. Contem tópicos relacionados aos objetivos do projeto, suas motivações, metodologia e resultados do classificador em outras bases de teste (Figura 25).

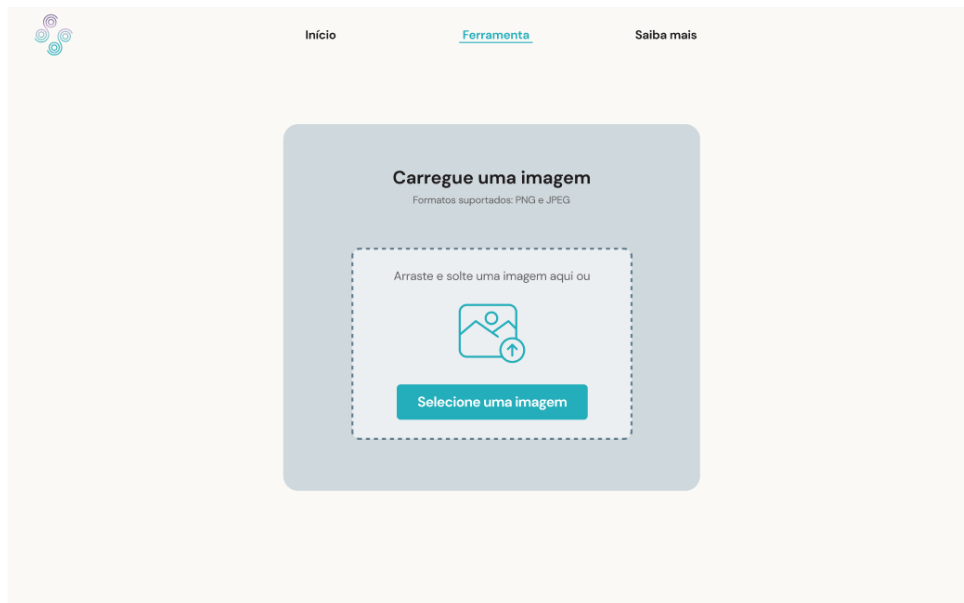


Figura 22: Prototipagem da tela de Início usando o Figma



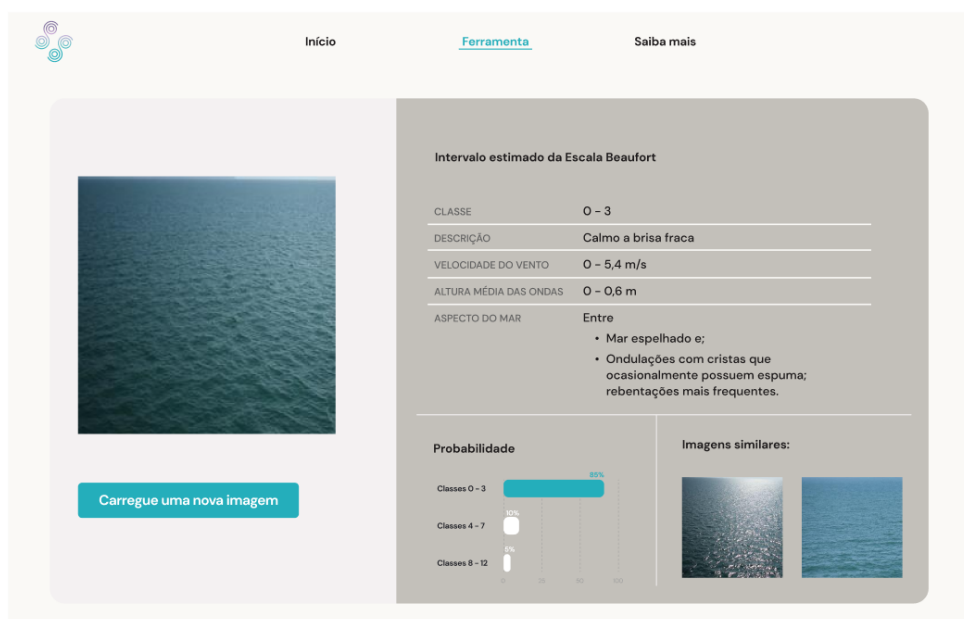
Fonte: os autores

Figura 23: Prototipagem da tela de Ferramenta usando o Figma



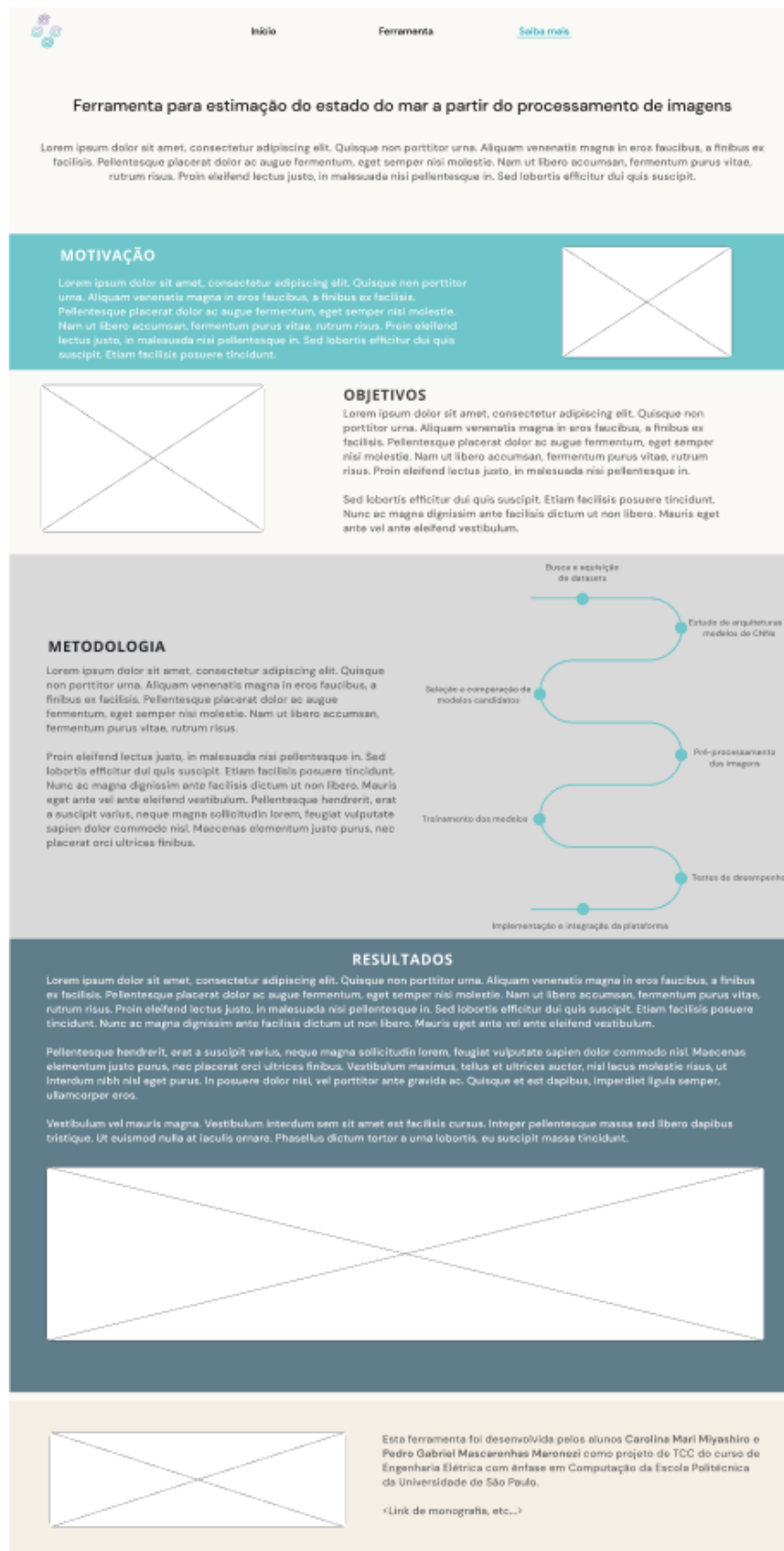
Fonte: os autores

Figura 24: Prototipagem da tela de resultados usando o Figma



Fonte: os autores

Figura 25: Prototipagem da tela de Saiba Mais usando o Figma



Fonte: os autores

Em seguida, implementou-se as páginas supracitadas utilizando a biblioteca React. Testou-se, então, a interface localmente, simulando parte da jornada do usuário:

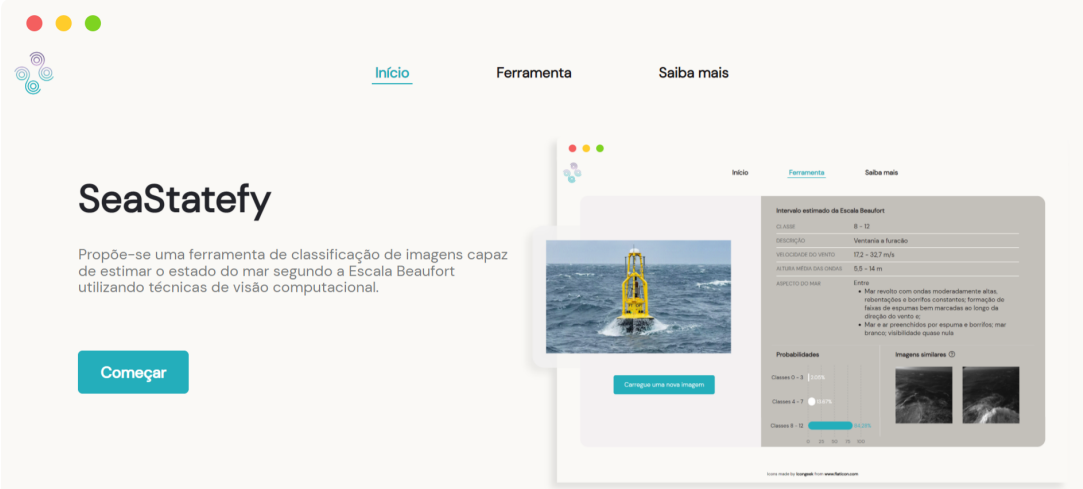
1. Seleção de uma imagem armazenada em um dispositivo local ou *drag-and-drop* da imagem na região delimitada;
2. Visualização de uma prévia da imagem escolhida e clique no botão "Classificar";
3. Codificação da imagem para Base64 e seu envio para o *endpoint* /classificar;
4. Exibição de um *loader* para comunicar ao usuário que a imagem está sendo processada;
5. Recebimento da resposta do servidor, contendo o intervalo estimado da Escala Beaufort, o que inclui informações referentes às classes, uma breve descrição, velocidade do vento, altura média das ondas e detalhes do aspecto do mar;
6. Exibição dos resultados na seção dedicada.

Por fim, publicou-se a aplicação na plataforma Heroku no seguinte endereço: <https://sea-state-classifier.herokuapp.com>.

### 6.7.3 Integração e plataforma final

A plataforma finalizada pode ser acessada em <https://sea-state-classifier.herokuapp.com>. Na aba de início (Figura 26), o usuário encontra uma breve introdução da ferramenta e um tutorial de como usá-la. Ao se dirigir para a página da ferramenta (Figura 27), é possível carregar uma imagem e visualizar a sua miniatura antes de enviá-la para ser classificada (Figura 28).

Figura 26: Home da plataforma SeaStatefy



**SeaStatefy**

Propõe-se uma ferramenta de classificação de imagens capaz de estimar o estado do mar segundo a Escala Beaufort utilizando técnicas de visão computacional.

**Começar**

**COMO USAR**

- 1 Clique em "Começar" ou acesse a ferramenta na aba "Ferramenta".
- 2 Arraste uma imagem para a área indicada ou carregue uma imagem armazenada em seu dispositivo local.
- 3 Clique no botão "Classificar".
- 4 Observe o intervalo da Escala Beaufort predito pelo classificador e suas informações correspondentes associadas à imagem carregada.

Icons made by Freepik, lcongeek from www.flaticon.com

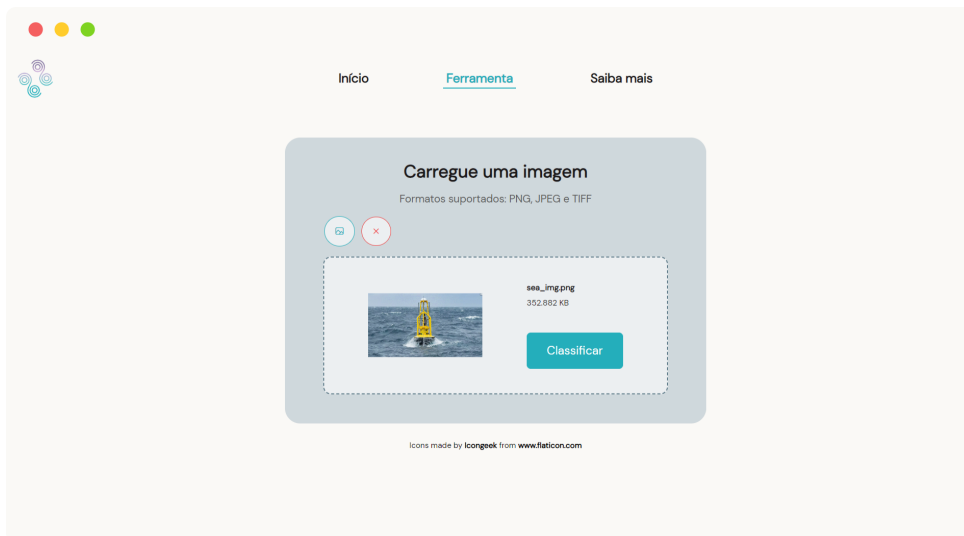
Fonte: os autores

Figura 27: Página da ferramenta da plataforma SeaStatefy



Fonte: os autores

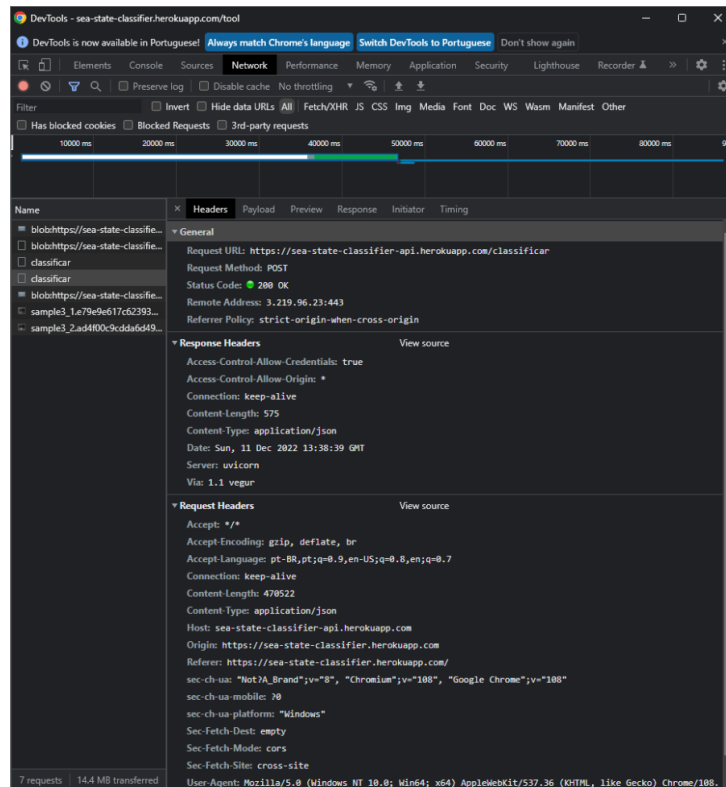
Figura 28: Página da ferramenta da plataforma SeaStatefy após carregamento da imagem



Fonte: os autores

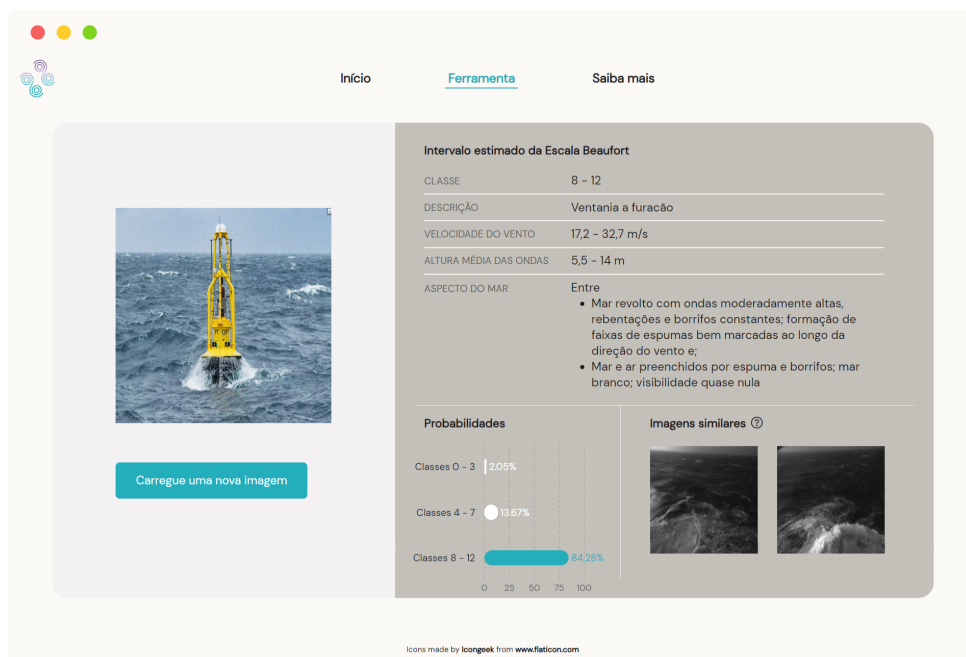
Codifica-se a imagem selecionada em Base64 e emprega-se uma requisição HTTP para mandá-la para o servidor (Figura 29). Exibe-se um *loader* para sinalizar ao usuário que a imagem está sendo processada e classificada e, por fim, apresenta-se a classificação estimada na área de resultados (Figura 30). Se for de interesse do usuário, informações referentes aos motivadores, objetivos e processo de desenvolvimento do projeto são encontrados na página de “saiba mais” (Figura 31).

Figura 29: Exemplo de requisição HTTP utilizada para envio da imagem codificada ao back-end




Fonte: os autores

Figura 30: Página da ferramenta da plataforma SeaStatefy com os resultados da classificação



Fonte: os autores

Figura 31: Página de “saiba mais” da plataforma SeaStatefy



[Início](#)   [Ferramenta](#)   [Saiba mais](#)

## SeaStatefy


Uma ferramenta para estimação do estado do mar a partir do processamento de imagens e utilizando visão computacional. Apresentam-se aqui os motivadores, os objetivos e o processo de desenvolvimento que culminaram na criação da plataforma SeaStatefy.

---

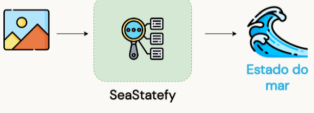
### MOTIVAÇÃO

O estado do mar (sea state) é uma medida da agitação da superfície do mar. Suas variáveis, em especial a altura das ondas, têm impacto, por exemplo, as estruturas offshore de petróleo e gás [1] e até mesmo a costa, sendo seu conhecimento de interesse da guarda costeira para a prevenção de catástrofes. Atualmente, os métodos para classificar os estados do mar são baseados na representação estatística de parâmetros de onda ou por modelagem numérica. Esses métodos são caros, propensos a mau funcionamento do equipamento e exigem alto poder de computação e tempo.

[1] WANNAK D, LIEW M.S, YEW, G.Z. Time Domain and Frequency Domain Analysis of Measured Motoccan Data for Malaysian Waters, Int. J. Geol. Environ. Eng. 2013, 7, 549-554.



---



### OBJETIVOS

Uma vez que modelos de aprendizado de máquina têm sido empregados como uma solução alternativa para prever e classificar as condições das ondas, o objetivo estabelecido para o projeto foi a criação de uma ferramenta de classificação do estado do mar utilizando Deep Learning. Denominada SeaStatefy, a plataforma deveria ser capaz de associar uma imagem da superfície do mar ao intervalo da Escala Beaufort mais semelhante:

- Graus 0 a 3 - Mar calmo/leve;
- Graus 4 a 7 - Mar moderado/agitado;
- Graus 8 a 12 - Mar forte/extremo.

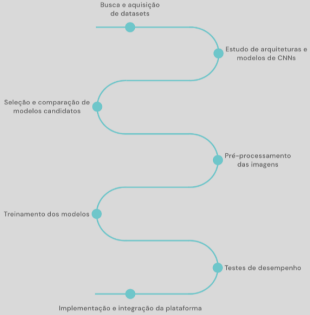
---

### METODOLOGIA

O processo de desenvolvimento consistiu em sete etapas principais. Primeiramente, buscou-se datasets de imagens da superfície do mar rotuladas segundo o grau da escala Beaufort correspondente ou anotadas com informações referentes a altura média das ondas para aquele conjunto de dados. Em seguida, estudou-se algumas arquiteturas de CNNs, partindo-se de modelos utilizados em artigos relacionados a este problema ou a problemas similares, e selecionou-se, então, os candidatos para realizar a extração de features e a classificação das imagens.

De posse dos modelos, partiu-se para o pré-processamento das imagens selecionadas, manipulando suas dimensões e aplicando técnicas de data augmentation para obter maior diversidade de dados. Treinou-se os modelos selecionados utilizando técnicas para datasets desbalanceados e comparou-se a performance de cada um com base em testes de desempenho e métricas de avaliação, como a acurácia.

Por fim, implementou-se e integrou-se a plataforma com o modelo classificador, desenvolvendo uma interface gráfica para a ferramenta de classificação a partir do mapeamento do seu usuário principal.



---

### RESULTADOS

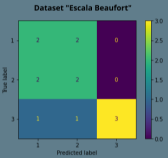
O modelo com o melhor desempenho foi a ResNet-34 pré-treinada, na qual adotou-se o oversampling como técnica de dataset desbalanceado, isto é, igualou-se a quantidade de dados em cada classe mediante duplicação das imagens das classes minoritárias de forma aleatória. Ele obteve o melhor equilíbrio de acertos nos cenários de teste propostos, uma média superior a 90%, conseguindo classificar relativamente bem imagens de um mar forte/extremo (graus 8 - 12 da Escala Beaufort) apesar da baixa quantidade de representantes. Sua acurácia em cada dataset é registrada abaixo.

Observou-se, entretanto, limitações na capacidade de generalização do modelo para novas imagens que sejam muito diferentes daquelas utilizadas no treinamento. Tal circunstância decorre da baixa diversidade dos conjuntos de imagens utilizados para o treinamento do modelo, tanto para imagens pertencentes a uma mesma classe quanto em representantes de classes diferentes, havendo majoritariamente amostras dos graus 0 a 3 da Escala Beaufort.

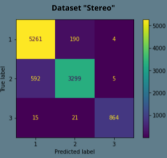
Em relação à plataforma SeaStatefy, foi possível desenvolver uma interface simples, fácil e rápida de ser utilizada, capaz de associar uma imagem ao intervalo da Escala Beaufort mais provável em poucos segundos.

Escala Beaufort	Dataset Stereo	Dataset MU-SSID
53,846%	91,832%	92,812%

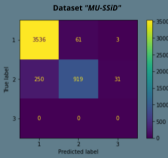
Dataset "Escala Beaufort"




Dataset "Stereo"



Dataset "MU-SSID"



---



Esta ferramenta foi desenvolvida pelos alunos Carolina Mari Miyashiro e Pedro Gabriel Mascarenhas Maronezi como projeto de TCC do curso de Engenharia Elétrica com ênfase em Computação da Escola Politécnica da Universidade de São Paulo.

Links relevantes:

- Monografia
- Demo

Icons made by Freepik, Icongeek, Eucalyp from www.flaticon.com



Mensurou-se o intervalo de tempo decorrido durante a classificação de uma imagem, isto é, desde a sua codificação para Base64 e o seu envio para o back-end até o recebimento da classificação estimada pelo front-end. Em média, tal processo demora 5,92 segundos, conforme exposto na Tabela 13.

Tabela 13: Intervalo de tempo decorrido para codificação, processamento e classificação de uma imagem

<b>Tamanho da imagem (kB)</b>	<b>Tempo (s)</b>
352,88	4.44
7,62	1.63
334,90	2.24
10000,11	15.37

# **PARTE III**

## **EXPERIMENTOS E CONCLUSÃO**

## 7 RESULTADOS

Nesta seção são apresentados os resultados para os testes descritos anteriormente. Tais resultados foram utilizados para aferir a qualidade e o impacto das arquiteturas e técnicas para *datasets* desbalanceados aplicadas, bem como verificar a diferença promovida pela adição dos dados do *dataset* MU-SSiD na performance dos melhores modelos. A partir destas informações, foi possível selecionar o modelo que apresenta o melhor equilíbrio em desempenho nos diferentes conjuntos de imagens, demonstrando sua capacidade de generalização.

### 7.1 Modelos treinados no *dataset Stereo*

Neste macro cenário, foram realizados três tipos de testes cujos resultados são apresentados abaixo. Ressalta-se que, para o desenvolvimento dos modelos deste conjunto de testes foram utilizadas apenas imagens do *dataset Stereo* no processo de treinamento e validação. A descrição deste *dataset* se encontra na Tabela 4 da Seção 6.1.

#### 7.1.1 Imagens do *dataset Stereo*

A acurácia dos modelos das arquiteturas VGG-16, ResNet-34, ResNet-152 e ResNet34-VGG para cada uma das gravações do conjunto de testes do *dataset Stereo* encontra-se nas Tabelas 14 a 17.

Tabela 14: Acurácia por modelo da arquitetura VGG-16 para cada gravação do *dataset Stereo*

Gravação	Pré-treinada	Sem pré-treino	<i>Oversampling</i>	<i>Weighted loss</i>
AA 1	0,99944	0,98833	0,97888	0,98944
AA 2	1	0,97877	0,97108	0,95773
AA 3	1	0,99666	0,91759	0,92650
YS	1	0,22333	0,96667	0,96333
BS 1	1	0,98892	1	1
BS 2	1	0,99499	1	1
BS 3	1	0,99778	1	1
BS 4	1	0,95773	1	1
BS 5	1	1	1	1
BS 6	1	1	1	1
BS 7	1	1	0,99889	1
LJ	1	1	1	1

Tabela 15: Acurácia por modelo da arquitetura ResNet-34 para cada gravação do *dataset Stereo*

Gravação	Pré-treinada	Sem pré-treino	<i>Oversampling</i>	<i>Weighted loss</i>
AA 1	0,97943	0,70650	0,98944	0,99333
AA 2	0,99778	0,92547	0,99444	1
AA 3	0,85523	0,90869	0,84855	0,86748
YS	0,38667	0,29667	0,40333	0,74
BS 1	0,98615	0,96399	0,99446	0,96676
BS 2	0,98998	0,08848	0,98331	0,96494
BS 3	0,87987	0,88209	0,87208	0,78643
BS 4	0,98443	0,23804	0,97330	0,95996
BS 5	1	1	1	1
BS 6	0,96329	1	0,96218	0,93993
BS 7	0,99666	1	0,99221	0,98554
LJ	0,9967	0,53667	0,99889	0,99889

Tabela 16: Acurácia por modelo da arquitetura ResNet-152 para cada gravação do *dataset Stereo*

Gravação	Pré-treinada	Sem pré-treino	<i>Oversampling</i>	<i>Weighted loss</i>
AA 1	0,99555	0,97999	0,99333	0,99778
AA 2	0,99666	0,96107	0,99778	1
AA 3	0,97550	0,99777	0,97105	0,98775
YS	0,50333	0,88333	0,55333	0,68
BS 1	1	0,04432	1	0,99723
BS 2	0,99666	0,00501	0,99666	0,97830
BS 3	0,93103	0,10679	0,94105	0,75306
BS 4	0,94438	0,00222	0,96218	0,77419
BS 5	0,96885	0,71079	0,98443	0,98331
BS 6	1	0,51047	1	1
BS 7	1	0,80311	0,99444	0,99555
LJ	0,99889	0,20111	0,99778	1

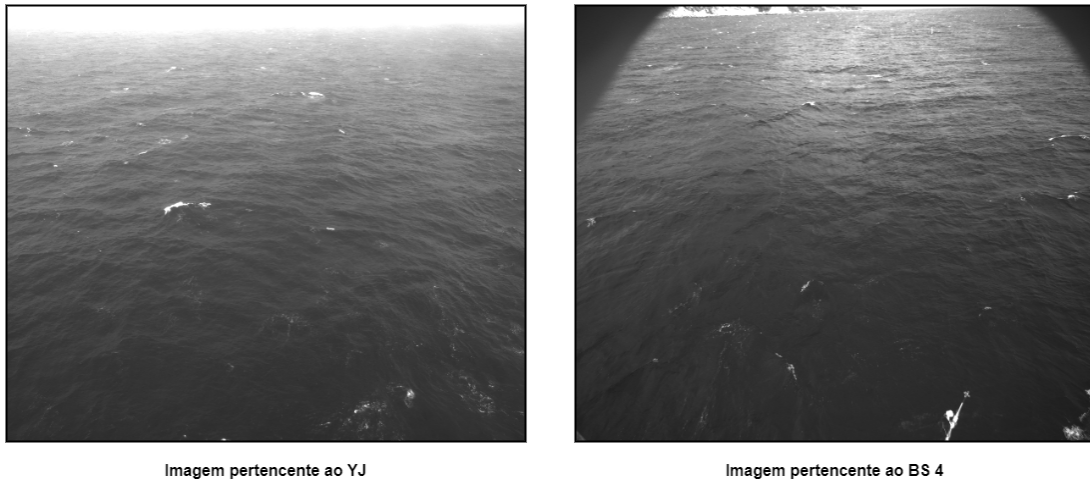
Tabela 17: Acurácia por modelo da arquitetura ResNet34-VGG para cada gravação do *dataset Stereo*

Gravação	Pré-treinada	<i>Oversampling</i>	<i>Weighted loss</i>
AA 1	0,99444	0,99833	0,98555
AA 2	0,98888	0,99889	0,99110
AA 3	0,86192	0,93875	0,89310
YS	0,50667	0,79333	0,53
BS 1	0,98892	0,99169	0,99446
BS 2	0,99666	0,97997	0,99446
BS 3	0,88432	0,84761	0,92325
BS 4	0,98220	0,95773	0,98776
BS 5	1	1	1
BS 6	0,97998	0,93882	0,97553
BS 7	0,99889	0,98443	0,99889
LJ	0,99778	0,99889	0,99778

Como pode ser observado, de modo geral os modelos se comportam bem apresentando elevadas acurácias para todas as gravações. No entanto, a gravação **YS** é a que a maioria dos modelos possui dificuldade para classificar (principalmente os modelos pertencentes

às arquiteturas ResNet). Tal dificuldade pode ser explicada pela proximidade entre as imagens desta gravação às imagens das gravações **BS** que pertencem à classe 1. Um exemplo de um representante de cada gravação é apresentado na Figura 32 abaixo.

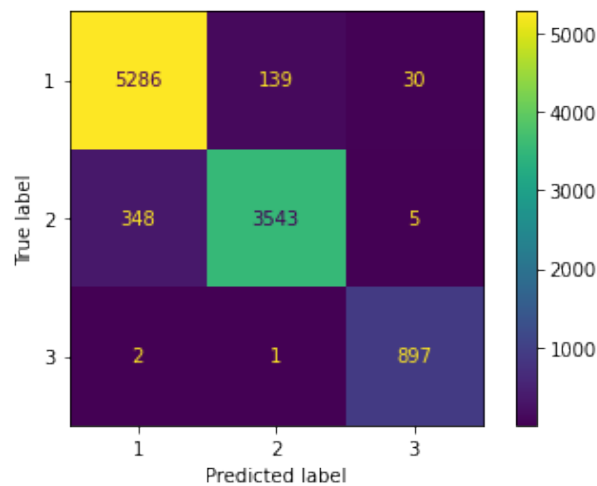
Figura 32: Imagens representantes da gravação YJ (esquerda) e da gravação BS 4 (direita)



Fonte: (GUIMARÃES et al., 2020)

Na matriz de confusão do modelo `resnet34_pretrained` na Figura 33, por exemplo, observa-se que os erros cometidos pelo modelo no que diz respeito à classe 2 geralmente são devido à classificá-los como pertencentes à classe 1, sendo que mais da metade destas classificações erradas são de imagens pertencentes à gravação **YS**. Este comportamento é semelhante para os demais modelos.

Figura 33: Matriz de confusão do modelo `resnet34_pretrained`. As linhas representam as classes verdadeiras (*True label*) e as colunas representam as classes previstas pelo modelo (*Predicted label*)



Fonte: os autores

A acurácia geral de cada modelo é apresentada na Tabela 18. Nota-se que, para este conjunto de imagens, apesar de todos os modelos, à exceção de suas versões sem pré-treinamento, possuem elevadas acurácias, a arquitetura VGG-16 é a que detém os resultados mais próximos a 1. Ademais, observa-se o impacto de utilizar o *transfer learning* a partir das redes pré-treinadas no ImageNet, uma vez que as versões sem o pré-treinamento obtiveram as piores performances, sendo os modelos baseados na arquitetura ResNet os mais afetados, o que pode ser explicado por sua camada de extração de *features* complexa ser a maior responsável por seu desempenho dada a simplicidade de sua *fully connected layer*.

Tabela 18: Acurácia geral por modelo para o *dataset Stereo*

Modelo	Acurácia
vgg16_pretrained	0,99990
vgg16_not_pretrained	0,96849
vgg16_oversample	0,98546
vgg16_weighted_loss	0,98693
resnet34_pretrained	0,94879
resnet34_not_pretrained	0,74100
resnet34_oversample	0,94810
resnet34_weighted_loss	0,94752
resnet152_pretrained	0,96830
resnet152_not_pretrained	0,57614
resnet152_oversample	0,97230
resnet152_weighted_loss	0,94449
resnet34_vgg_pretrained	0,95717
resnet34_vgg_oversample	0,96283
resnet34_vgg_weighted_loss	0,96303

### 7.1.2 Imagens representantes da escala Beaufort

As acurácias de cada modelo para as imagens representantes da escala Beaufort são apresentadas na Tabela 19. Nesta tabela é apresentada, comparativamente, as acurácias destes modelos para as imagens do *dataset Stereo*, para que possa comparar as acurácias entre os dois cenários e, deste modo, verificar o equilíbrio de performance entre eles.

Tabela 19: Acurácia geral por modelo para as imagens representantes da escala Beaufort e para o *dataset Stereo*

Modelo	Acurácia escala Beaufort	Acurácia <i>dataset Stereo</i>
vgg16_pretrained	0,30769	0,99990
vgg16_not_pretrained	0,30769	0,96849
vgg16_oversample	0,23077	0,98546
vgg16_weighted_loss	0,15385	0,98693
resnet34_pretrained	0,53846	0,94879
resnet34_not_pretrained	0,30769	0,74100
resnet34_oversample	0,61538	0,94810
resnet34_weighted_loss	0,53846	0,94752
resnet152_pretrained	0,46154	0,96830
resnet152_not_pretrained	0,38462	0,57614
resnet152_oversample	0,38462	0,97230
resnet152_weighted_loss	0,30769	0,94449
resnet34_vgg_pretrained	0,46154	0,95717
resnet34_vgg_oversample	0,53846	0,96283
resnet34_vgg_weighted_loss	0,61538	0,96303

Constata-se que, à parte da arquitetura híbrida ResNet34-VGG, a VGG-16 possui o melhor desempenho no cenário das imagens do *dataset Stereo*, enquanto os modelos da ResNet-34, performam melhor no cenário das imagens representantes da escala Beaufort. Esta situação levou, posteriormente, à ideia da criação da arquitetura híbrida ResNet34-VGG como mencionado anteriormente.

Ao testar esta nova arquitetura em ambos os cenários, percebe-se que a hipótese por trás de sua concepção (i.e., combinar as partes aparentemente responsáveis pelos bons desempenhos da VGG-16 e da ResNet-34 nos cenários de imagens do *dataset Stereo* e de imagens representantes da escala Beaufort, respectivamente) se confirmou, já que os modelos baseados na nova arquitetura mantiveram bons desempenhos em ambos os cenários de teste.

Ademais, foi possível observar que os modelos baseados na VGG-16 não desempenharam bem no cenário que continha imagens muito diferentes das utilizadas no treinamento. Apesar da escassez de imagens do cenário de teste com imagens representantes da escala Beaufort não permitir análises profundas, o mal desempenho da arquitetura neste cenário é um sinal de um possível *overfitting* em relação às imagens de treinamento, o que pode



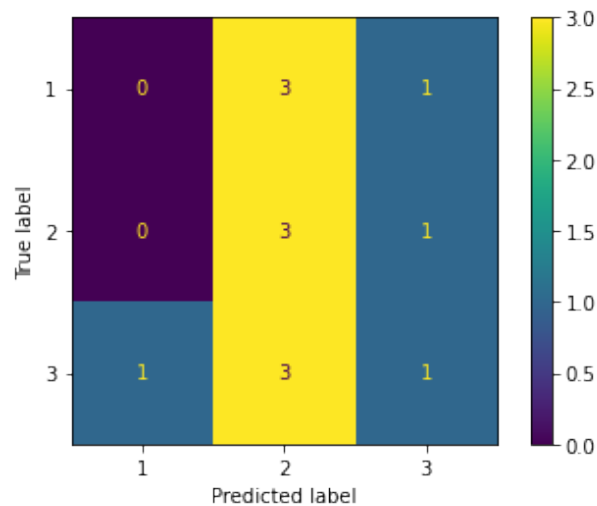
significar uma menor capacidade de generalização dos modelos.

A partir destes resultados, selecionou-se o melhor modelo de cada arquitetura. O critério de escolha foi o equilíbrio de performance entre os dois cenários, ou seja, a capacidade de maximizar ambas as acurácias. Além desta métrica, observou-se ainda as matrizes de confusão de cada modelo no que diz respeito às imagens representantes da escala Beaufort. Os 4 melhores modelos com suas respectivas acurácias são apresentados na Tabela 20. As matrizes de confusão de cada um destes modelos são apresentadas nas Figuras 34 a 37

Tabela 20: Acurácia geral por modelo para as imagens representantes da escala Beaufort e para o *dataset Stereo* para os 4 melhores modelos

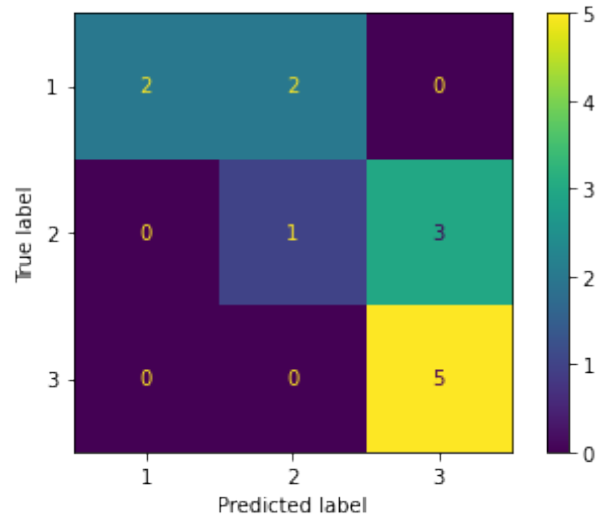
Modelo	Acurácia escala Beaufort	Acurácia <i>dataset Stereo</i>
vgg16_pretrained	0,30769	0,99990
resnet34_oversample	0,61538	0,94810
resnet152_pretrained	0,46154	0,96830
resnet34_vgg_weighted_loss	0,61538	0,96303

Figura 34: Matriz de confusão do modelo `vgg16_pretrained` nas imagens representantes da escala Beaufort. As linhas representam as classes verdadeiras (*True label*) e as colunas representam as classes previstas pelo modelo (*Predicted label*)



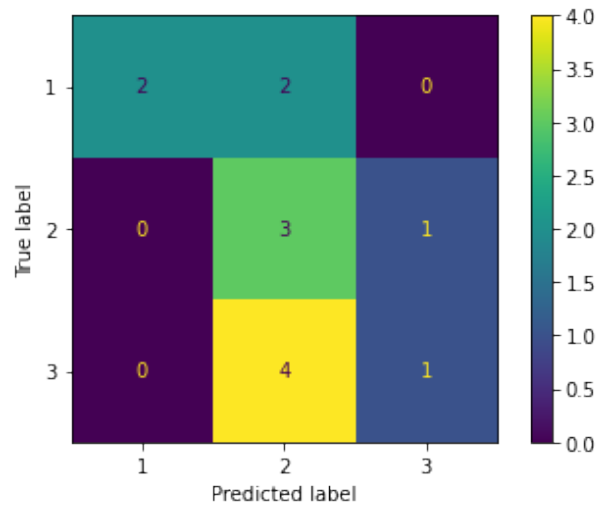
Fonte: os autores

Figura 35: Matriz de confusão do modelo `resnet34_oversample` nas imagens representativas da escala Beaufort. As linhas representam as classes verdadeiras (*True label*) e as colunas representam as classes previstas pelo modelo (*Predicted label*)



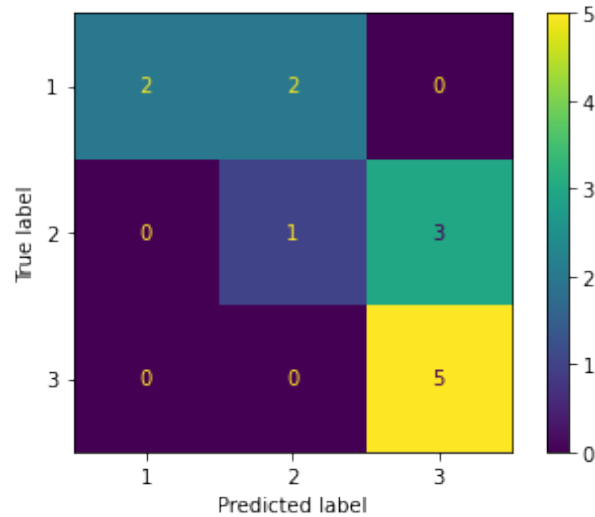
Fonte: os autores

Figura 36: Matriz de confusão do modelo `resnet152_pretrained` nas imagens representativas da escala Beaufort. As linhas representam as classes verdadeiras (*True label*) e as colunas representam as classes previstas pelo modelo (*Predicted label*)



Fonte: os autores

Figura 37: Matriz de confusão do modelo `resnet34_weighted_loss` nas imagens representantes da escala Beaufort. As linhas representam as classes verdadeiras (*True label*) e as colunas representam as classes previstas pelo modelo (*Predicted label*)



Fonte: os autores

A partir destas matrizes de confusão, observa-se que os modelos que possuem as camadas de extração de *features* baseadas na ResNet-34 possuem as melhores métricas de acurácia e de precisão para imagens pertencentes às classes 1 e 3, enquanto os demais modelos apresentam grande dificuldade em classificar imagens da classe 3 principalmente.

Adicionalmente, verifica-se que ambas as arquiteturas ResNet-34 e ResNet34-VGG possuem como seus melhores representantes os modelos que aplicam as técnicas para *datasets* desbalanceados. Este fato, corrobora com a explicação do motivo para ambos os modelos apresentarem melhores desempenhos em classificar imagens pertencentes à classe 3, dado que esta é a classe com menos imagens de treinamento, mas que, ao utilizar técnicas que compensem o desbalanceamento, é possível obter melhores resultados para esta classe.

### 7.1.3 Imagens do *dataset* MU-SSiD

As acurácias de cada modelo para as imagens do *dataset* MU-SSiD são apresentadas na Tabela 21. Dado que este *dataset* possui apenas imagens das classes 1 e 2, este teste visa medir a capacidade de generalização dos modelos para imagens pertencentes a estas duas classes, as quais são as que possuíam a maior quantidade de observações durante a etapa de treinamento.

Tabela 21: Acurácia geral por modelo para as imagens do *dataset* MU-SSiD

Modelo	Acurácia <i>dataset</i> MU-SSiD
vgg16_pretrained	0,02208
resnet34_oversample	0,13458
resnet152_pretrained	0,47771
resnet34_vgg_weighted_loss	0,19542

As acurácias apresentadas são baixas para os 4 modelos, mostrando a dificuldade dos modelos em generalização. Isso pode ser explicado pela pequena diversidade do *dataset* de treinamento, o qual, apesar das técnicas de *image augmentation*, ainda apresenta imagens relativamente parecidas entre si, com poucas variações entre ângulos das imagens, por exemplo.

A Tabela 22 abaixo resume as acurácias dos 4 modelos nos três micro cenários abordados nesta seção.

Tabela 22: Acurácia geral por modelo para as imagens do *dataset* MU-SSiD, para as imagens representantes da escala Beaufort e para o *dataset Stereo* para os 4 melhores modelos treinados com imagens do *dataset Stereo*

Modelo	Escala Beaufort	<i>Dataset Stereo</i>	<i>Dataset</i> MU-SSiD
vgg16_pretrained	0,30769	0,99990	0,02208
resnet34_oversample	0,61538	0,94810	0,13458
resnet152_pretrained	0,46154	0,96830	0,96303
resnet34_vgg_weighted_loss	0,61538	0,96303	0,19542

## 7.2 Modelos treinados nos *datasets Stereo* e MU-SSiD

As acurácias dos 4 modelos treinados utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD para os três cenários de teste são apresentados na Tabela 23.

Tabela 23: Acurácia geral por modelo para as imagens do *dataset* MU-SSiD, para as imagens representantes da escala Beaufort e para o *dataset Stereo* para os 4 melhores modelos treinados com imagens do *dataset Stereo* junto do *dataset* MU-SSiD

Modelo	Escala Beaufort	<i>Dataset Stereo</i>	<i>Dataset</i> MU-SSiD
vgg16_pretrained	0,38462	0,96156	0,96271
resnet34_oversample	0,53846	0,91932	0,92812
resnet152_pretrained	0,46154	0,91211	0,99
resnet34_vgg_weighted_loss	0,46154	0,96547	0,86833

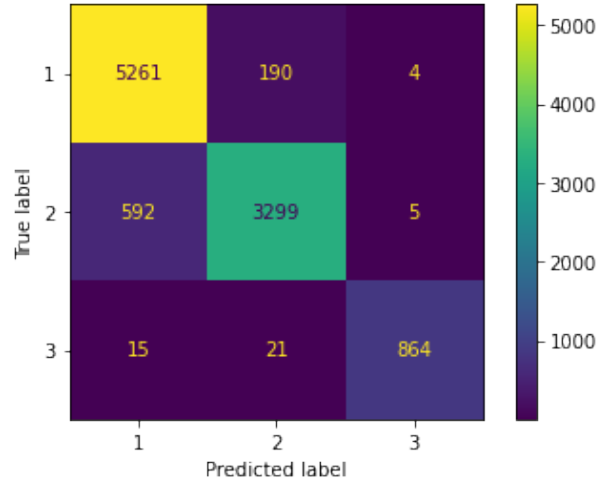
Comparando-se as acurácias dos modelos para imagens do *dataset* MU-SSiD, percebe-se que a adição das novas imagens no treinamento dos modelos contribuiu para um aumento expressivo na sua capacidade de classificar imagens do *dataset* MU-SSiD, o que mostra a possibilidade de acrescentar-se novas imagens aos conjuntos de treinamento para treinar os modelos a fim de aumentar a sua capacidade de generalização devido à maior diversidade do conjunto de treinamento.

No entanto, houve uma queda para a maior parte dos modelos nos outros dois cenários, o que pode significar um menor *overfitting* destes modelos a imagens do *dataset Stereo*, por exemplo. Apesar de haver um pequeno aumento de desempenho do `resnet34_vgg_weighted_loss` no *dataset Stereo* e das quedas nos demais modelos, não houve grandes variações na acurácia relativa a este *dataset*.

As grandes variações de acurácia quanto às imagens representantes da escala Beaufort se dão devido ao tamanho deste conjunto ser de apenas 13 imagens, de modo que acertar ou errar a classificação de uma única imagem causa uma variação de cerca de 7,7% na acurácia. Portanto, é interessante observar que as variações das acurácias neste cenário representam o acerto ou o erro de classificação de 1 ou 2 imagens, de forma que pode-se dizer que a variação do desempenho neste cenário também não foi tão grande.

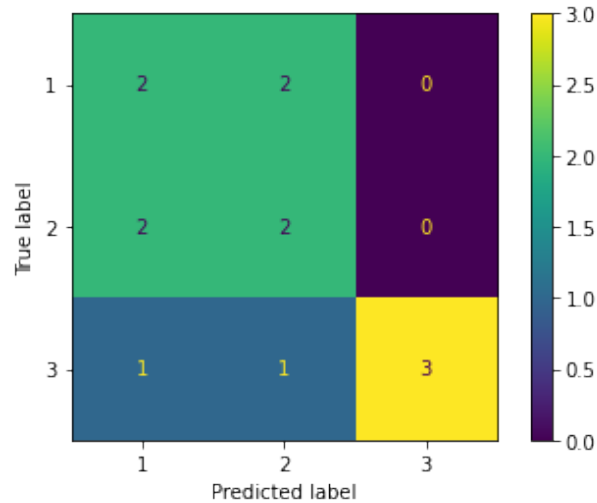
Com estes novos dados em mãos, o modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD foi selecionado como aquele que possui o melhor equilíbrio na acurácia dos 3 conjuntos de imagem, conseguindo classificar suficientemente bem imagens da classe 3 apesar da adição de novas imagens no treinamento ter aumentado ainda mais o desbalanceamento dos *datasets*. A matriz de confusão deste modelo para cada um dos três conjuntos de imagem é mostrada nas Figuras 38, 39 e 40. As métricas de precisão, revocação e *F1-score* para cada conjunto de imagens são mostradas nas Tabelas 24, 25 e 26. Ressalta-se que o conjunto de imagens do *dataset* MU-SSiD não possui imagens da classe 3.

Figura 38: Matriz de confusão do modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD nas imagens do *dataset Stereo*. As linhas representam as classes verdadeiras (*True label*) e as colunas representam as classes previstas pelo modelo (*Predicted label*)



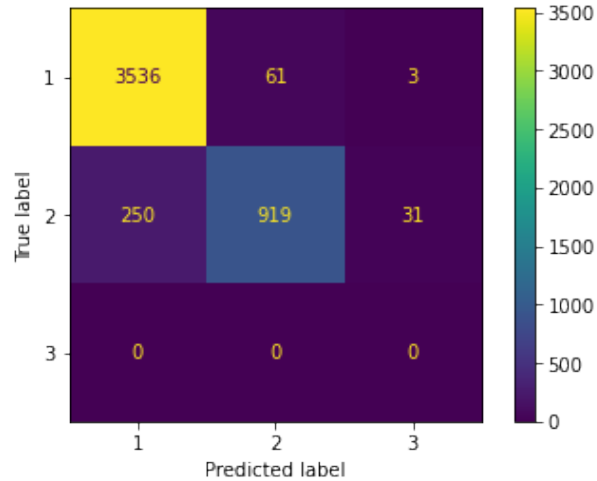
Fonte: os autores

Figura 39: Matriz de confusão do modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD nas imagens representantes da escala Beaufort. As linhas representam as classes verdadeiras (*True label*) e as colunas representam as classes previstas pelo modelo (*Predicted label*)



Fonte: os autores

Figura 40: Matriz de confusão do modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD nas imagens do *dataset* MU-SSiD. As linhas representam as classes verdadeiras (*True label*) e as colunas representam as classes previstas pelo modelo (*Predicted label*)



Fonte: os autores

Tabela 24: Métricas para o modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD nas imagens do *dataset Stereo*

Classe	Precisão	Revocação	<i>F1-score</i>	# imagens
1	0,89656	0,96444	0,92926	5455
2	0,93989	0,84677	0,89090	3896
3	0,98969	0,96	0,97462	900

Tabela 25: Métricas para o modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD nas imagens representantes da escala Beaufort

Classe	Precisão	Revocação	<i>F1-score</i>	# imagens
1	0,4	0,5	0,44444	4
2	0,4	0,5	0,44444	4
3	1	0,6	0,75	5

Tabela 26: Métricas para o modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD nas imagens do *dataset* MU-SSiD

Classe	Precisão	Revocação	<i>F1-score</i>	# imagens
1	0,93397	0,98222	0,95749	3600
2	0,93776	0,76583	0,84312	1200
3	0	0	0	0

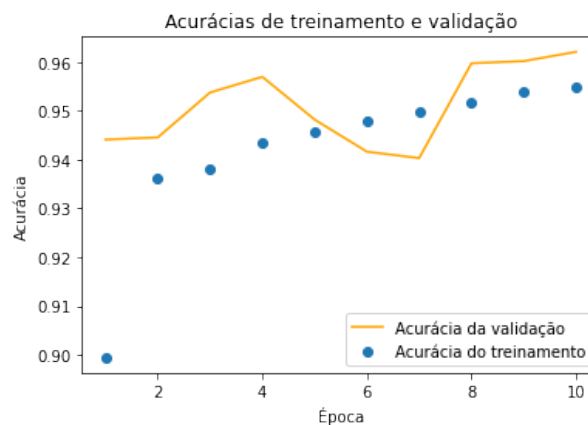
Verifica-se que o modelo selecionado consegue classificar bem imagens da classe 3, possuindo 98,9% de precisão nesta classe para as imagens do *dataset Stereo*. O desempenho do modelo ao classificar imagens dos *datasets Stereo* e MU-SSiD é elevado e sua qualidade é atestada pela diagonalidade de suas matrizes de confusão: apesar de não serem perfeitamente diagonais, a precisão e a revocação de cada uma das classes (e consequentemente o *F1-score*) são elevadas para ambos os cenários.

Quanto ao desempenho em relação às imagens representantes da escala Beaufort, apesar deste não ser um cenário com grande valor de análise devido à escassez de imagens, nota-se que o modelo faz certa confusão entre as classes 1 e 2, de modo que as imagens da classe 1, quando classificadas em outra classe é como classe 2 e vice-versa. Adicionalmente nota-se queda na precisão da classe 3 neste cenário se comparado com a versão do modelo treinada apenas no *dataset Stereo* que era de 100%. Esta queda é explicada pelo maior desbalanceamento provocado pela adição de novas imagens das classes 1 e 2 ao conjunto de treinamento.

No entanto, deve-se considerar que a qualidade das imagens representantes da escala Beaufort é baixa devido à presença de partes dos barcos ocupando espaços expressivos da imagem, o que dificulta a classificação por parte do modelo.

As Figuras 41 e 42 mostram um comparativo entre as etapas de treino e validação quanto à acurácia e ao erro (*loss*), respectivamente. Nelas nota-se certa proximidade no comportamento da acurácia e dos erros entre os conjuntos de treinamento e validação, com isso, dado que ambas as acurácias estão crescendo e ambos os erros diminuindo, pode-se afirmar que não há um cenário de *overfitting* muito proeminente.

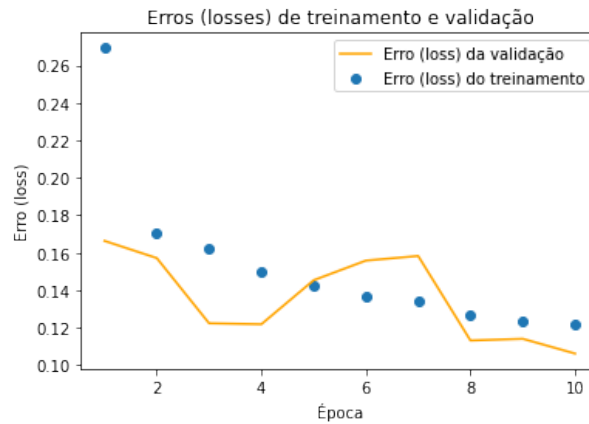
Figura 41: Variação da acurácia de treinamento e validação do modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset* MU-SSiD durante as épocas.



Fonte: os autores



Figura 42: Variação do erro (*loss*) de treinamento e validação do modelo `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset MU-SSiD* durante as épocas.



Fonte: os autores

Em linhas gerais, o modelo selecionado possui um bom desempenho nos três conjuntos de imagens, sendo capaz de exercer certo grau de generalização. Contudo, dada a baixa diversidade dos conjuntos utilizados no treinamento, a capacidade de generalização do modelo é prejudicada, uma vez que as imagens são parecidas entre si, apresentando poucas variações de ângulos, amostras de imagens de uma quantidade bem limitada de mares e grande desbalanceamento, contando com poucas amostras da classe 3. Apesar das limitações, percebe-se o potencial de evolução do modelo com a adição de novas imagens de treinamento, permitindo que os pesos atuais sirvam de base para o treinamento com novas imagens, trazendo maior diversidade ao conjunto de treino e, conseqüentemente, melhorando a capacidade de generalização do modelo.

## 8 CONCLUSÕES

O **estado do mar** (*sea state*) é uma medida da agitação da superfície do mar (ondas) e é usado para definir a segurança de um navio em atravessar determinado trecho do oceano em navegação; para definir a segurança das operações de plataformas de petróleo e navios de perfuração e exploração oceânica, e na costa; para quantificar a possibilidade de agitação nas praias, ressacas, alagamentos; dentre outros. Atualmente, os métodos para classificar os estados do mar são baseados na representação estatística de parâmetros de onda ou por modelagem numérica. Esses métodos são caros, propensos a mau funcionamento do equipamento e exigem alto poder de computação e tempo.

Desta forma, dado que modelos de aprendizado de máquina têm sido empregados como uma solução alternativa para prever e classificar as condições das ondas, o objetivo estabelecido para o projeto foi a criação de uma ferramenta de classificação do estado do mar de aprendizado profundo que classifica o estado do mar predominante usando apenas a imagem da superfície do mar, resultando na plataforma *SeaStatefy*.

### 8.1 Considerações finais

A metodologia de trabalho adotada para o desenvolvimento do modelo classificatório e arquitetura da plataforma se mostraram escolhas assertivas, uma vez que a metodologia permitiu o teste de diversas técnicas, situações e arquiteturas e suas combinações a fim de produzir o melhor modelo possível dentro das limitações do projeto; e a separação entre front e back end permitiu o desenvolvimento em paralelo da interface do usuário e do *script* executado no servidor, resultando em peças modulares de modo que é possível realizar alterações de interface sem impactar na capacidade de classificação e substituir o modelo classificatório no *script* por versões mais atualizadas.

Apesar das elevadas métricas do modelo classificatório resultante (o `resnet34_oversample` treinado utilizando imagens do *dataset Stereo* junto do *dataset MU-SSiD*), observou-se limitações em sua capacidade de generalização para novas imagens que sejam muito dife-

rentes daquelas utilizadas no treinamento, de forma que ainda há a uma boa capacidade de classificação, mas com uma queda de desempenho.

Tal circunstância decorre da baixa diversidade dos conjuntos de imagens utilizados para o treinamento do modelo, tanto para imagens pertencentes a uma mesma classe quanto em representantes de classes diferentes, havendo majoritariamente amostras dos graus 0 a 3 da escala Beaufort.

Ademais, notou-se que o modelo apresenta algumas dificuldades em distinguir imagens pertencentes aos graus presentes nas fronteiras das classes, isto é, imagens de grau 3 (classe 1) possuem características muito similares a imagens de grau 4 (classe 2), ocorrendo, por vezes, confusão por parte do modelo para estas imagens.

De modo geral, obteve-se um modelo classificatório do estado do mar com desempenho satisfatório e acima do esperado, dadas as limitações dos *datasets*, apesar de possuir algumas restrições quanto à generalização para imagens muito diferentes. A plataforma ***SeaStatefy*** resultou em uma interface simples, fácil e rápida de ser utilizada, podendo classificar imagens em poucos segundos e contendo informações sobre o processo de desenvolvimento do classificador a fim de atribuir maior confiabilidade aos resultados por ela produzidos. Os requisitos para o classificador e para a plataforma foram satisfeitos e o objetivo do trabalho foi bem sucedido, culminando na plataforma ***SeaStatefy***.

## 8.2 Trabalhos Futuros

Como trabalhos futuros, o primeiro passo seria a adição de novas imagens rotuladas segundo os graus da escala Beaufort para os conjuntos de treinamento, validação e teste, obtendo maior diversidade de imagens dentro de cada grau, bem como obtendo representantes de mais graus diferentes, o que possibilitaria o aumento da acurácia do modelo e também de sua granularidade, ou seja, o aumento do número de classes a fim de que, idealmente, cada classe contenha um único grau da escala Beaufort.

Ademais, como foi possível observar com o treinamento utilizando algumas imagens do *dataset* MU-SSiD, há a possibilidade de utilizar o *transfer learning* do modelo desenvolvido para o desenvolvimento de novos modelos que partam dos pesos já calculados e adicionem novas imagens ao treinamento, sem a necessidade de treinar o modelo do zero e permitindo a evolução contínua do classificador.

Outro ponto de melhoria seria a exploração de novas arquiteturas de CNNs além daquelas abordadas neste estudo e um refinamento maior dos hiper parâmetros utilizando

algum método mais sofisticado.

Por fim, para a plataforma ***SeaStatefy***, um ponto a ser desenvolvido é adicionar as funções de *upload* de múltiplas imagens e de vídeos a serem classificados ou até mesmo ter a opção de utilizar uma câmera (webcam ou celular) e realizar a estimativa em tempo real.

## REFERÊNCIAS

- ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. *2017 international conference on engineering and technology (ICET)*. [S.l.], 2017. p. 1–6.
- ASARIOTIS, R. et al. *Review of Maritime Transport 2018*. [S.l.], 2018.
- CHENG, X. et al. Spectralseanet: Spectrogram and convolutional network-based sea state estimation. In: IEEE. *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. [S.l.], 2020. p. 5069–5074.
- COLAB. *Colab*. 2022. Disponível em: <https://colab.research.google.com/>. Acesso em: 2022-07-09.
- DOCKER. *Docker Docs*. 2022. Disponível em: <https://docs.docker.com/>. Acesso em: 2022-10-28.
- FASTAPI. *FastAPI*. 2022. Disponível em: <https://fastapi.tiangolo.com/>. Acesso em: 2022-07-09.
- GUIMARÃES, P. V. et al. A data set of sea surface stereo images to resolve space-time wave fields. *Scientific data*, Nature Publishing Group, v. 7, n. 1, p. 1–12, 2020.
- HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.
- HEROKU. *Heroku*. 2022. Disponível em: <https://www.heroku.com/platform>. Acesso em: 2022-12-09.
- LAB, S. V. *ImageNet*. 2022. Disponível em: <https://www.image-net.org/>. Acesso em: 2022-10-12.
- LAINING, A. et al. Guide to wave analysis and forecasting. *Geneva, Switzerland: World Meteorological Organization*, 1998.
- LEGORBURU, I.; JOHNSON, K. R.; KERR, S. A. Offshore oil and gas. *Building Industries at Sea: “Blue Growth” and the New Maritime Economy*, p. 231–56, 2018.
- LEPOSA, N. Problematic blue growth: A thematic synthesis of social sustainability problems related to growth in the marine and coastal tourism. *Sustainability Science*, Springer, v. 15, n. 4, p. 1233–1244, 2020.
- LIU, N. et al. Wave height inversion and sea state classification based on deep learning of radar sea clutter data. In: IEEE. *2021 International Conference on Control, Automation and Information Sciences (ICCAIS)*. [S.l.], 2021. p. 34–39.
- MOORING. *SURFACE BUOY SYSTEMS*. 2022. Disponível em: <https://www.mooringsystems.com/surface.htm>. Acesso em: 2022-11-24.

- NEELY, W. *The Great Hurricane of 1780: The Story of the Greatest and Deadliest Hurricane of the Caribbean and the Americas*. [S.l.]: iUniverse, 2012.
- NUMPY. *NumPy*. 2022. Disponível em: <https://numpy.org/>. Acesso em: 2022-07-09.
- OFFICE, U. M. *National Meteorological Library and Archive Fact sheet 6—The Beaufort Scale*. [S.l.]: Met Office London, UK, 2012. Acesso em: 2022-06-10.
- OLSEN, A. et al. Deepweeds: A multiclass weed species image dataset for deep learning. *Scientific reports*, Nature Publishing Group, v. 9, n. 1, p. 1–12, 2019.
- O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.
- PAN, M. et al. Visual recognition based on deep learning for navigation mark classification. *IEEE Access*, IEEE, v. 8, p. 32767–32775, 2020.
- PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, v. 32, 2019.
- POOLE, D. I.; GOEBEL, R. G.; MACKWORTH, A. K. *Computational intelligence*. [S.l.]: Oxford University Press New York, 1998. v. 1.
- PYTHON. *Python*. 2022. Disponível em: <https://www.python.org/>. Acesso em: 2022-07-09.
- PYTORCH. *PyTorch*. 2022. Disponível em: <https://pytorch.org/>. Acesso em: 2022-07-09.
- RAYNAL, A. M.; DOERRY, A. W. *Doppler characteristics of sea clutter*. [S.l.], 2010.
- REACT. *React*. 2022. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 2022-07-09.
- SCIKITLEARN. *Scikitlearn: Machine Learning in Python*. 2022. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 2022-07-09.
- SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. *Journal of big data*, SpringerOpen, v. 6, n. 1, p. 1–48, 2019.
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- SINGLETON, F. The beaufort scale of winds—its relevance, and its use by sailors. *Weather*, Wiley Online Library, v. 63, n. 2, p. 37–41, 2008.
- THOMSON, J. et al. Emerging trends in the sea state of the beaufort and chukchi seas. *Ocean Modelling*, Elsevier, v. 105, p. 1–12, 2016.
- UMAIR, M.; HASHMANI, M. A.; HASAN, M. H. B. Survey of sea wave parameters classification and prediction using machine learning models. In: IEEE. *2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS)*. [S.l.], 2019. p. 1–6.

UMAIR, M. et al. A novel deep learning model for sea state classification using visual-range sea images. *Symmetry*, MDPI, v. 14, n. 7, p. 1487, 2022.

VANNAK, D.; LIEW, M. S.; YEW, G. Z. Time domain and frequency domain analyses of measured metocean data for malaysian waters. *International Journal of Geological and Environmental Engineering*, v. 7, n. 8, p. 549–554, 2013.

WALLBRINK, H.; KOEK, F. Historical wind speed equivalents of the beaufort scale, 1850-1950. *KNMI-Memorandum HISKLIM*, Citeseer, v. 13, 2009.

ZHANG, K.; YU, Z.; QU, L. Application of deep learning in sea states images classification. In: IEEE. *2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC)*. [S.l.], 2021. p. 976–979.

ZULKIFLI, H. Understanding learning rates and how it improves performance in deep learning. *Towards Data Science*, v. 21, n. 23, 2018.