

Universidade do Minho

Escola de Engenharia

Departamento de Informática

João Carlos da Silva Costa

Deteção de Padrões em imagens SAR

December 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Carlos da Silva Costa

Detecção de Padrões em imagens SAR

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
Cesar Analide Freitas Silva Costa Rodrigues

December 2021

COPYRIGHT AND TERMS OF USE FOR THIRD PARTY WORK

This dissertation reports on academic work that can be used by third parties as long as the internationally accepted standards and good practices are respected concerning copyright and related rights.

This work can thereafter be used under the terms established in the license below.

Readers needing authorization conditions not provided for in the indicated licensing should contact the author through the RepositóriUM of the University of Minho.

LICENSE GRANTED TO USERS OF THIS WORK:



CC BY

<https://creativecommons.org/licenses/by/4.0/>

AGRADECIMENTOS

O desenvolvimento de um projeto desta dimensão foi sem dúvida o capítulo mais desafiante da minha vida até ao momento. Por esse motivo, esta dissertação não se trata de uma conquista apenas minha. Ao longo desta jornada recebi muito apoio e orientação.

Começo por agradecer à instituição Universidade do Minho por me ter acolhido e formado ao longo destes 5 anos.

Ao professor doutor Cesar Analide Rodrigues, meu orientador, deixo um obrigado por todo o aconselhamento, ajuda e paciência que permitiu a permanência no rumo certo e a manutenção da motivação num projeto muitas vezes desmotivador.

Um obrigado à empresa Tekever e seus responsáveis, por me terem proporcionado esta oportunidade de trabalhar num tema extremamente interessante e complexo que me desafiou todos os dias neste último ano.

Quero deixar um agradecimento particular ao Engenheiro Bruno Silva, meu supervisor na empresa Tekever, por ter sido incansável no *feedback* e na ajuda a qualquer problema encontrado. Foi uma peça crucial no desenvolvimento da dissertação.

Aos meus pais, António e Nazaré, a quem devo tudo, um agradecimento profundo pelos sacrifícios que fizeram para que eu pudesse ter a formação que sempre quis. Por estarem sempre disponíveis e prontos a ajudar em qualquer momento da minha vida.

Ao meu irmão, Gonçalo, agradeço a paciência e as brincadeiras nos momentos em que precisava e não fui a melhor companhia.

Um agradecimento especial à Angélica, alicerce ao longo de todo percurso, pelo apoio incondicional nos momentos mais difíceis, pelas palavras motivadoras e principalmente por acreditar mesmo quando eu não acreditava. Sem ela não teria conseguido.

Por fim, aos meus amigos e "companheiros de batalhas", agradeço a paciência, os momentos de descontração e a "simples" presença nos momentos em que era necessária.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

Since the last century, concerns about safety and welfare issues have increased at the same pace as technological advances. Synthetic Aperture Radar (SAR) is one way of creating images using active sensors, which has the advantage that it can be used in any climate and time of day. Due to its complexity, the use of these images is time consuming and requires an experienced user. This study aims to find a way to handle these images, in terms of geographic registration and image interpretation, in a more efficient way.

Based on the literature on geographic registration techniques, image comparison and segmentation in SAR images, those with the greatest potential to solve the problems identified are developed and tested. The analysis of the solutions demonstrates that the developed methods have the ability to evolve into complete and efficient tools which are capable of answering the identified problems. The results indicate that the SAR-Harris SAR image comparator proves to be a stable enough algorithm to include in a semi-automatic registration tool. Concerning the SAR image classification and segmentation, the U-Net neural network presented values of accuracy near 70%, exhibiting the ability of the network to integrate a SAR image classification system. In the case of the C-Means algorithm, despite not reaching the U-Net accuracy values, accuracy values in range 50-60% , it presents itself as an extremely versatile algorithm, also useful in the segmentation task. Still, there is a great margin of progress in the automation of the processes of geographic registration and segmentation of SAR images.

KEYWORDS SAR, Artificial Intelligence, Machine Learning, Deep Learning, Neural Networks, Segmentation, Computer Vision

RESUMO

Desde o século passado, as preocupações em assuntos de segurança e bem-estar têm aumentado ao mesmo ritmo que os avanços tecnológicos.

Synthetic Aperture Radar (SAR) é uma das formas de criar imagens utilizando sensores ativos, que possuem a vantagem de poder ser utilizado em qualquer clima e hora do dia. Devido à sua complexidade, a utilização destas imagens é muito custosa em termos de tempo e requer mão-de-obra qualificada. Este estudo tem o objetivo de encontrar uma forma de lidar com estas imagens, em termos de registo geográfico e interpretação da imagem, de forma mais eficiente.

Com base na literatura acerca de técnicas registo geográfico, comparação entre imagens e segmentação em imagens SAR, são desenvolvidas e testadas aquelas que apresentam maior potencial para resolver os problemas identificados. A análise das soluções demonstram que os métodos estudados e desenvolvidos tem a capacidade de evoluir para ferramentas mais completas e eficientes capazes de responder aos problemas identificados. Os resultados indicam que o comparador de imagens SAR SAR-Harris revela ser um algoritmo estável e robusto o suficiente para originar uma ferramenta de registo semi-automática. Já em termos de classificação e segmentação de imagens SAR, a rede neuronal U-Net apresenta valores de *accuracy* próximos dos 70%, exibindo a capacidade da rede de integrar um sistema de classificação de imagens SAR. No caso do algoritmo C-Means apesar de não atingir os valores de precisão da U-Net, atingindo valores de 50-60%, apresenta-se como um algoritmo extremamente versátil e robusto, características úteis na tarefa de segmentação. Ainda assim, há uma grande margem de progressão na automatização dos processos de registo geográfico e de segmentação de imagens SAR.

PALAVRAS-CHAVE SAR, Inteligência Artificial, *Machine Learning*, *Deep Learning*, Redes Neurais, Segmentação, Visão por Computador

CONTEÚDO

I O PROBLEMA E BASE TEÓRICA

1	INTRODUÇÃO	4
1.1	Contexto e Motivação	4
1.2	Problema e Desafios	5
1.3	Objectivos	7
1.4	Organização da dissertação	7
1.5	Planeamento e Calendarização	8
2	ESTADO DE ARTE	9
2.1	Synthetic Aperture Radar	9
2.1.1	Interpretação das Imagens	11
2.2	Inteligência Artificial	13
2.2.1	Machine Learning	14
2.2.2	Desenvolvimento de Soluções de IA	33
2.2.3	Computer Vision	38

II METODOLOGIA

3	REGISTO GEOGRÁFICO DE IMAGENS SAR	42
3.1	Registo baseado em ASIFT	42
3.1.1	Procedimento	42
3.1.2	Testes	46
3.1.3	Solução Porposta	48
3.2	Registo baseado em SAR-Harris	50
3.2.1	Procedimento	51
3.2.2	Testes	52
3.3	Sumário	56
4	SEGMENTAÇÃO DE IMAGENS SAR	58
4.1	Redes Neurais	59
4.1.1	U-Net	61
4.1.2	DSCNN	70

4.2	Fuzzy C-Means Clustering	77
4.2.1	Algoritmo	77
4.2.2	Testes	81
4.3	Sumário	95
 III INTERPRETAÇÃO E CONSIDERAÇÕES FINAIS		
5	DISCUSSÃO E ANÁLISE DE RESULTADOS	98
6	CONCLUSÕES E TRABALHO FUTURO	102

LISTA DE FIGURAS

Figura 1	Esquema que representa o cenário de obtenção das imagens. (Crédito: (Olmsted, 1993))	10
Figura 2	Diferentes comprimentos de onda utilizados por radares SAR e suas aplicações. (Crédito: NASA)	11
Figura 3	Esquema de distorções geométricas associadas ao SAR. (Crédito: ESA)	12
Figura 4	Exemplo de distorções geométricas associadas ao SAR. (Crédito: (Le Toan, 2007))	13
Figura 5	Processo de transformação dos dados Raw numa imagem SAR. (Crédito: (Moreira et al., 2013))	13
Figura 6	Ramos de Inteligência artificial. (Crédito: (Mills, 2016))	14
Figura 7	Diferentes tipos de aprendizagem	15
Figura 8	Na figura da esquerda vemos o perceptron e na figura da direita está representado um neurónio humano	17
Figura 9	Principais funções de ativação usadas em NN: sigmoid , TanH e ReLU	18
Figura 10	Exemplo de uma rede neuronal. Rede com 4 inputs, 2 hidden layers, e uma output layer	19
Figura 11	Exemplificação da procura do mínimo local numa função exemplo	21
Figura 12	Exemplificação do cálculo durante a operação de convolução. Goodfellow et al. (2016)	24
Figura 13	Operação que ocorre nas camadas convolucionais. (Crédito: towardsdatascience.com)	25
Figura 14	Operação de pooling. (Crédito: towardsdatascience.com)	25
Figura 15	Diferença entre uma RNN e uma rede neuronal convencional (feed-forward). (Crédito: Built-in.com)	27
Figura 16	Processamento de uma frase por uma recurrent neural network. (Crédito: Phi (2018))	28
Figura 17	Demonstração ilustrada da influência das diversas fases na última palavra. (Crédito: Phi (2018))	28
Figura 18	Arquitetura de uma rede GAN.	29
Figura 19	Metodologia SEMMA. créditos: researchgate.net	34
Figura 20	Metodologia CRISP-DM. créditos: researchgate.net	35
Figura 21	Comparação entre o algoritmo ASIFT e SIFT. Imagens com diferença de 90º de ângulo (Crédito: (Morel and Yu, 2009))	40
Figura 22	Diagrama do procedimento do registo com ASIFT.	43
Figura 23	Teste 0 - Comparação entre a imagem ótica e a imagem SAR do mesmo local.	46

Figura 24	Teste 1 - Comparação entre a imagem ótica e a imagem SAR.	47
Figura 25	Teste 2 - Comparação entre a imagem ótica e a imagem SAR com rotação 180°.	47
Figura 26	Teste 3 - Comparação entre a imagem ótica e a imagem SAR com resize.	48
Figura 27	Teste 4 - Comparação entre a imagem ótica com rotação 180° e a imagem SAR com alteração filtro.	49
Figura 28	Processo de registo de imagem através do algoritmo SAR-Harris.	52
Figura 29	Teste 1 - Comparação entre imagens SAR do dataset SEN12MS.	53
Figura 30	Teste 2 - Comparação entre imagens SAR do dataset SEN12MS, com ângulo.	54
Figura 31	Teste 3 - Comparação entre imagens SAR do dataset Ger Imagery (1).	54
Figura 32	Teste 4 - Comparação entre imagens SAR do dataset Ger Imagery (2).	55
Figura 33	Teste 5 - Comparação entre imagens SAR da Tekever.	55
Figura 34	Exemplos de imagens que compõem o dataset SEN12MS e a anotação respetiva.	59
Figura 35	Exemplos de imagens que compõem o dataset Ger Imagery e a anotação respetiva.	60
Figura 36	Diagrama do processo de segmentação com rede neuronal.	60
Figura 37	Representação gráfica da rede U-Net (Crédito: (Morel and Yu, 2009))	61
Figura 38	Classificação U-Net, treinada com SEN, de uma imagem do dataset SEN12MS , com diferentes opções.	64
Figura 39	Classificação U-Net, treinada com SEN, de uma imagem do dataset SEN12MS	65
Figura 40	Classificação U-Net, treinada com SEN, de uma imagem do dataset Ger Imagery	65
Figura 41	Classificação de uma imagem SAR, da Tekever, pela U-Net, treinada com SEN.	66
Figura 42	Classificação pela U-Net, treinada com SEN, de uma imagem SAR da Tekever, numa localização diferente.	66
Figura 43	Comparação entre uma uma imagem do dataset SEN12MS e do SAR da Tekever	67
Figura 44	Alterações da imagem através da utilização do método Gamma Correction	67
Figura 45	Classificação das imagens alteradas através do método Gamma Correction	68
Figura 46	Classificação U-Net, treinada com Ger Imagery, de uma imagem do dataset Ger Imagery	69
Figura 47	Classificação U-Net, treinada com Ger Imagery, de uma imagem do dataset Ger Imagery numa localização diferente.	69
Figura 48	Classificação U-Net, treinada com Ger Imagery, de uma imagem do dataset SEN12MS	69
Figura 49	Classificação da U-Net, treinada com Ger Imagery, de uma imagem da Tekever.	70
Figura 50	Comparação entre a classificação com inicialização aleatória (esquerda) e classificação ideal (direita) (Crédito:Geng et al. (2017))	71
Figura 51	Treino de toda a DSCNN.	74
Figura 52	Classificação da rede DSCNN - Exemplo 1.	76

Figura 53	Classificação da rede DSCNN - Exemplo 2.	76
Figura 54	Exemplo de decomposição por wavelet. (Crédito: Wasilewski (2021))	77
Figura 55	Comparação entre a classificação com inicialização aleatória (esquerda) e classificação ideal (direita). (Crédito: Wasilewski (2021))	78
Figura 56	Diagrama representativo da parte iterativa do algoritmo C-Means.	87
Figura 57	Comparação entre a classificação com inicialização aleatória (esquerda) e classificação ideal (direita) (Crédito: savyakhosla (2021))	88
Figura 58	Processo de seleção dos centróides em dados exemplo. (Crédito: savyakhosla (2021))	88
Figura 59	Resultado do C-Means numa imagem do dataset SEN12MS	89
Figura 60	C-Means aplicado a uma imagem pertencente ao dataset Ger Imagery - Exemplo 1.	90
Figura 61	Comparação entre a Imagem original , classificação C-Means e a Máscara real - Ger Imagery (1).	90
Figura 62	Resultado do C-Means numa imagem do dataset Ger Imagery - Exemplo 2.	91
Figura 63	Comparação entre a Imagem original , classificação C-Means e a Máscara real - Ger Imagery (2).	91
Figura 64	Resultado do C-Means numa imagem fornecida pela Tekever	92
Figura 65	Resultado do algoritmo C-Means com $c = 3$	93
Figura 66	Resultado do algoritmo C-Means com $c = 5$	94

LISTA DE TABELAS

Tabela 1	Tabela de planeamento com as tarefas e respetivas datas.	8
Tabela 2	Legenda de variáveis a usar no exemplo de backpropagation	21
Tabela 3	Opções API	43
Tabela 4	Resumo de testes	49
Tabela 5	Resumo dos testes dos comparadores para registo geográfico.	57
Tabela 6	Execução do algoritmo C-Means - SEN12MS.	82
Tabela 7	Execução do algoritmo C-Means - Ger Imagery (Exemplo 1).	82
Tabela 8	Execução do algoritmo C-Means - Ger Imagery (Exemplo 2).	83
Tabela 9	Execução do algoritmo C-Means - Tekever.	84
Tabela 10	Resumo dos testes dos algoritmos de segmentação	96

LISTA DE PROCEDIMENTOS

1	Treino SCAE	73
2	K-Means++	80

ABREVIATURAS E ACRÓNIMOS

3D	Três Dimensões	ILKFCM	<i>Intensity and Location Kernel Fuzzy C-Means</i>
AE	<i>Auto Encoder</i>	IoU	Intersection over Union
ANN	<i>Artificial Neural Network</i>	ISRO	Indian Space Research Organization
API	<i>Application Programming Interface</i>	LSTM	<i>Long Short-Term Memory</i>
ASIFT	<i>Affine Scale-Invariant Feature Transform</i>	MDPI	<i>Multidisciplinary Digital Publishing Institute</i>
BAIR	<i>Berkeley AI Research</i>	MIEI	Mestrado Integrado em Engenharia Informática
BPTT	<i>Backpropagation Through Time</i>	ML	Machine Learning
CAE	<i>Contractive Auto Encoder</i>	MLR	<i>Multiple Linear Regression</i>
CNN	<i>Convolution Neural Network</i>	NASA	<i>National Aeronautics and Space Administration</i>
CPU	<i>Central Processing Unit</i>	RADAR	<i>Radio Detection And Ranging</i>
CRISP-DM	<i>Cross-Industry Standard Process for Data Mining</i>	RANSAC	<i>RANdom SAmple Consensus</i>
CUDA	<i>Compute Unified Device Architecture</i>	ReLU	<i>Rectified Linear Unit</i>
DL	Deep Learning	RNN	<i>Recurrent Neural Network</i>
DoG	<i>Difference of Gaussian</i>	ROEWA	<i>Ratio of Exponentially Weighted Averages</i>
DR	Diário da República	SAR	<i>Synthetic Aperture Radar</i>
DSCNN	<i>Deep Supervised Contractive Neural Network</i>	SCAE	<i>Supervised Contractive Auto encoder</i>
FCM	<i>Fuzzy C-means Clustering</i>	SEMMA	<i>Sample, Explore, Modify, Model, and Asses</i>
GAN	<i>Generative Adversial Network</i>	SGD	<i>Stochastic Gradient Descent</i>
GPU	<i>Graphical Processing Unit</i>	UM	Universidade do Minho
GRU	<i>Gated Recurrent Units</i>		
HTTP	<i>Hypertext Transfer Protocol</i>		

Parte I

O PROBLEMA E BASE TEÓRICA

INTRODUÇÃO

1.1 CONTEXTO E MOTIVAÇÃO

O presente documento aborda o projeto desenvolvido ao longo do quinto ano do Mestrado Integrado em Engenharia Informática (MIEI), na instituição Universidade do Minho (UM). Sob orientação do Professor Doutor Cesar Analide Rodrigues, sob supervisão do Engenheiro Bruno Silva da empresa Tekever.

Desde o século passado, as preocupações em assuntos de segurança e bem estar têm aumentado ao mesmo ritmo que os avanços tecnológicos. Também a preocupação com a vigilância é comum a todas as sociedades e Estados-Governo. Os investimentos em monitorização e a proliferação de novas e melhores tecnologias de vigilância são uma realidade no mundo em que vivemos. (Estêvão, 2014)

Provenientes de todo Mundo surgem notícias que fazem referência à vigilância e à procura pela tecnologia de ponta que dá a quem a possuir uma vantagem face os seus inimigos, quer sejam estes humanos ou naturais. Em maio de 2020, foi publicado no Diário da República (DR) Portuguesa uma resolução por parte do Conselho de Ministros que autoriza a Força Aérea a comprar 12 *drones*, num investimento aproximado de 4.5 milhões de euros de maneira a reforçar a vigilância florestal. (Lusa, 2020)

Do outro lado do Atlântico a *National Aeronautics and Space Administration* (NASA) está num projeto conjunto com *Indian Space Research Organization*(ISRO), o *NASA-ISRO Synthetic Aperture Radar* (NI-SAR), para que em 2022, segundo a organização americana, um satélite equipado com SAR orbite o planeta Terra. Projeto que tem um custo na ordem dos 1.5 Biliões de dólares americanos, e que tem como objetivo o estudo do planeta para melhor entendimento do mesmo e dos fenómenos que ocorrem. (NASA, 2020)

Na Europa, uma das empresas com maior desenvolvimento reconhecido em SAR é a ICEYE, uma empresa espacial com sede em Espoo na Finlândia, e recentemente um dos seus projetos conseguiu um financiamento de cerca de 87 milhões de dólares, para enviar para o espaço 12 satélites equipados com SAR até ao fim de 2021, elevando assim o valor de mercado do SAR. (Werner, 2020) Em outubro 2019, o Departamento de Defesa da Rússia realizou um voo experimental de um *drone* espião equipado com SAR, projeto que tem vindo a decorrer desde há uns anos e que custou dezenas de milhões de euros. (SAR Journal, 2019)

No que toca a valor de mercado, o SAR tem impressionado e segundo a britânica TECHNAVIO, vai merecer cada vez mais relevância. A TECHNAVIO realizou estudo do mercado SAR e estima que para o período 2020-2024 o valor do mercado irá aumentar cerca de 870 milhões de dólares. (Business Wire, 2020)

Desde observação ambiental, mapeamento de recursos do planeta e até sistemas militares estão a tender para a utilização de sistemas aéreos para estes fins, quer sejam óticos ou radar, caso do SAR.

Por norma, as imagens óticas necessitam de ser obtidas durante o dia e com o clima favorável, só assim se tornam perfeitamente visíveis e compreensíveis. No entanto, existem imensas situações que exigem que as imagens sejam captadas durante a noite ou com mau tempo. Nessas circunstâncias o uso dos sistemas convencionais de obtenção de imagens não produz os resultados esperados.

A utilização de um *Synthetic Aperture Radar* (SAR) permite aliviar estas restrições. Uma vez que tira proveito das características de propagação de sinais radar, o SAR não está dependente da altura do dia nem das condições atmosféricas existentes. Com respostas únicas de terreno que só frequências radar oferecem, o SAR vem para complementar a imagem ótica. O processamento digital desta informação permite a obtenção de imagens de alta resolução, que facilita a deteção e identificação de certos pontos de interesse, principalmente em situações em que os sistemas óticos não conseguem atuar.

Atualmente, os temas monitorização e vigilância estão mais enraizados e emergentes que nunca na sociedade, por esse motivo faz todo sentido trabalhar na procura das melhores abordagens do estado de arte nesta área, quer seja em termos de performance ou eficiência. Em particular, SAR apesar de nos últimos anos estar a despontar o interesse de muitos investigadores, há uma margem de progressão e de acordo com os dados de mercado cada vez mais justifica o investimento.

O objetivo principal é desenvolver, através de uma perspetiva de *Machine Learning* (ML), a capacidade de mapeamento e interpretação de imagens capturadas por um radar de tipo SAR, de maneira a tornar este processo mais rápido e com menor intervenção humana.

A TEKEVER é empresa portuguesa fundada em 2001 com uma presença forte no mercado internacional de vigilância, oferecendo aos seus clientes *Intelligence as-a-service*. Com presença nas áreas de veículos aéreos não-tripulados (drones), Espaço e projetos de soluções tecnológicas. A empresa prepara-se para lançar um projeto envolvendo um SAR. Assim, a presente dissertação surge na procura pelo aperfeiçoamento das capacidades do radar, para que dessa forma o mesmo seja utilizado em pleno potencial, de uma maneira eficaz e eficiente. O domínio abrangido pelo documento irá centrar-se nesse estudo por tecnologia estado de arte no que toca a processamento de imagens recolhidas pelo SAR.

1.2 PROBLEMA E DESAFIOS

Esta dissertação procura colmatar algumas lacunas que foram identificadas em conjunto com a TEKEVER relativamente à utilização e exploração do SAR.

O estudo de soluções existentes na comunidade científica para os problemas identificados poderá oferecer uma maior utilidade e eficiência na utilização deste tipo de radar.

Assim, o primeiro problema identificado foi a forma como é realizado o mapeamento das imagens, capturadas pelo SAR, para coordenadas geográficas reais. O processo é feito manualmente, e apesar de ser feito por um ser Humano trazer associado um possível rigor ao processo de identificação, acarreta também custos que pesam mais que os seus benefícios, nomeadamente, o tempo de duração e a dificuldade para a pessoa encarregue

de realizar a tarefa. Torna-se assim um processo de extrema ineficiência e com necessidade efetiva de ser "automatizado", pelas suas vantagens quer em termos de tempo, quer de eficiência.

Da mesma forma, manualmente, é realizado o processo de identificação de pontos de interesse na imagem. Esta situação apresenta as mesmas vantagens e desvantagens do processo anterior, se por um lado conseguimos uma classificação rigorosa das imagens. Por outro, a quantidade de tempo necessária para um Humano realizar a identificação e anotação dos pontos de interesse na imagem, não é passível de ser ignorada. Por fim, a interoperabilidade do SAR que pode ser utilizado em meio aéreo (incorporado em *drones*) ou espacial (quando integrado em satélites), resulta em diferentes resoluções e complexidade das imagens, este tipo de mudanças exige mais trabalho por parte das pessoas encarregues destas tarefas.

Os problemas identificados acima dificultam a utilização do SAR em ambiente de produção, o estudo, desenvolvimento e comparação das soluções desenvolvidas permitirá optar pela forma de resolver o problema que mais se enquadra com os objetivos computacionais e de qualidade.

No entanto, no caminho para a solução dos problemas descritos acima encontram-se alguns obstáculos que se espera sejam ultrapassados com sucesso.

O primeiro desafio encontrado será o processamento eficiente das imagens SAR, uma vez que estas imagens apresentam maior complexidade que as imagens óticas comuns, processar estas imagens de uma forma otimizada e eficiente apresenta alguma dificuldade e bastante trabalho de pesquisa e desenvolvimento.

Uma outra possível adversidade encontrada durante a construção da solução será a necessidade de anotação das imagens para os modelos de classificação supervisionado, esta tarefa requer um trabalho manual inicial exaustivo, que se manifesta de forma clara no tempo gasto para a tarefa.

Ainda na componente anterior à construção da solução em si, a quantidade de dados por si só será um desafio. Isto porque algoritmos de *Deep Learning* exigem quantidade astronómicas de dados e uma vez que o SAR está ainda em voos teste, a quantidade de imagens poderá não ser a ideal. Juntando a isto a impossibilidade de realizar *data augmentation* (técnica de aumento artificial da quantidade de dados), devido as características únicas das imagens SAR, não permite uma atenuação deste obstáculo.

No que concerne à solução de *Machine Learning* e a sua construção, a construção de um algoritmo eficaz e eficiente de mapeamento geográfico e segmentação exigem uma implementação profundamente fundamentada teoricamente, para que seja possível justificar os resultados positivos, ou negativos, e evoluir a partir destes.

Por fim, e como visto anteriormente, a interoperabilidade do SAR trará alguns desafios à etapa de construção, uma vez que será necessário um algoritmo eficaz tanto no processamento imagens aéreas como espaciais (mais complexas), que obriga à personalização dos algoritmos de maneira a transpor esta adversidade.

Estes serão os problemas e desafios que serão encontrados durante a resolução dos problemas encontrados. Obstáculos estes que deverão ser ultrapassados para que os objetivos desta dissertação sejam concluídos com sucesso.

1.3 OBJECTIVOS

Os objetivos desta investigação surgem do processo de análise e levantamento de problemas relacionados com a utilização de imagens SAR.

Assim, este projeto procura:

Desenvolver a capacidade de lidar imagens SAR no contexto do mundo real.

Sendo este um conceito bastante abstrato e alto nível, é importante referir que este se fragmenta por dois caminhos distinto, que caracterizam duas componentes da solução. Uma componente diz respeito ao mapeamento de imagens SAR do ponto de vista geográfico. Enquanto a outra se refere processamento e interpretação da imagem.

Nesse sentido, os objetivos e metas para este projeto são as seguintes.

- Localização da imagem SAR no mundo real, por forma a diminuir o trabalho humano envolvido neste processo, através da identificação das coordenadas geográficas (Latitude e Longitude);
- Classificação e segmentação da imagem de acordo com as classes "Água", "Zona Agrícola", "Zona Florestal", "Zona Urbana". De maneira a avaliar esta meta será utilizada a métrica *Pixel Accuracy* e é previsto um valor superior a 0.7.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

A dissertação encontra-se dividida em 2 partes e 6 capítulos.

O presente capítulo, pode ser visto como uma componente introdutória à investigação onde é fornecido ao leitor o contexto de desenvolvimento, as suas motivações, obstáculos e os seus objetivos.

Depois, surge o estado de arte relacionado com o projeto a ser desenvolvido. No estado de arte é possível observar a análise da literatura no que concerne aos objetivos em questão, com o objetivo de encontrar metodologias que sejam capazes de colmatar as lacunas encontradas. É feita uma análise teórica fundamentada por artigos e pesquisas de alguns casos pertinentes para o desenvolvimento da dissertação. Serve também como forma de fornecer ao leitor, um *background* teórico dos termos e dos conceitos utilizados.

O capítulo nº3, de nome "Registo geográfico de imagens SAR", aborda como o nome indica, o desenvolvimento de uma solução para o 1º problema identificado (i.e., o registo manual de imagens SAR). O capítulo inicia-se com um pequeno resumo do componente. De seguida, é estudada a utilização do algoritmo ASIFT como comparador de imagens a utilizar na ferramenta de localização geográfica. É também estudado o uso de um outro comparador, (SAR-Harris), com características diferentes do ASIFT. Em cada uma dessas secções, o comparador é testado. No final do capítulo, é feito um resumo do capítulo.

Depois de abordar o componente 1, é momento de procurar uma resposta para o problema relacionado com a interpretação da imagem. O capítulo nº4 inicia-se da mesma forma que o anterior, com uma descrição do problema. A primeira solução em estudo é a utilização de redes neuronais como classificador de imagens.

Posteriormente, é realizada uma investigação acerca do uso do método não supervisionado C-Means. No final do capítulo é apresentado um sumário do capítulo.

Posteriormente é apresentada ao leitor uma avaliação dos resultados encontrados e a sua importância, no capítulo referente à discussão. Ao longo da dissertação são apresentados os testes às componentes e às soluções propostas, e na discussão é o momento de comparar alternativas e relacionar os resultados com o estado de arte e os objetivos que a investigação procura alcançar. Apresenta também os argumentos que irão suportar as conclusões finais do projeto.

No último capítulo da Dissertação é feito um sumário global da investigação. É feita a análise de sucesso dos resultados obtidos face às metas propostas, com base na avaliação feita na discussão. São, ainda, abordadas as dificuldades encontradas ao longo de todo o processo de desenvolvimento. Por fim, são apresentadas, como trabalho futuro, algumas propostas de melhoria de cada componente.

1.5 PLANEAMENTO E CALENDARIZAÇÃO

Esta secção tem o propósito de apresentar o planeamento feito para o projeto. Na Figura ?? é possível ver o planeamento através de um Diagrama de Gantt e na Tabela 1 encontra-se a informação do diagrama de um modo mais compacta.

Tabela 1: Tabela de planeamento com as tarefas e respetivas datas.

Tasks	Start Time	End Time
Estudo Estado de Arte	01/10/2020	15/01/2021
Preparar Defesa Estado de Arte	15/01/2021	20/01/2021
Desenvolvimento de Algoritmos - Registo geográfico	20/01/2021	15/04/2021
Análise de resultados das Soluções - Registo geográfico	10/04/2021	16/05/2021
Desenvolvimento de Algoritmos - Segmentação	17/05/2021	15/09/2021
Análise de resultados das Soluções - Segmentação	28/07/2021	13/11/2021
Escrita do documento da Dissertação	01/10/2020	29/12/2021
Preparar defesa da Dissertação	20/12/2021	-

ESTADO DE ARTE

Neste capítulo são apresentadas as fundações teóricas dos conceitos que serão desenvolvidos nos capítulos futuros desta dissertação. Inicialmente será dada uma breve contextualização acerca do SAR e do seu funcionamento.

De seguida, uma abordagem *Top-Down* em termos conceptuais, iniciando-se assim com Inteligência artificial, passando pelo *Machine Learning*, que inclui a aprendizagem supervisionada e não supervisionada. A teoria relativa às redes neuronais, em particular *Convolution Neural Networks*, *Recurrent Neural Network* e *Generative Adversial Networks*, do lado supervisionado e ao algoritmo *Fuzzy C-Means Clustering*, no lado não supervisionado, representa a base para os desenvolvimentos futuros. O capítulo termina com a área de visão por computador, através da introdução ao algoritmo *Affine Scale-Invariant Feature Transform* (ASIFT), que apresenta bastante utilidade na comparação de imagens.

2.1 SYNTHETIC APERTURE RADAR

Quando se fala de ‘Imagem de satélite’, grande maioria das pessoas associa a uma imagem ótima – uma fotografia – capturada por uma câmara extremamente potente. No entanto, as imagens óticas não são a única forma de ver a superfície da Terra a partir de um satélite ou de um avião. (Brown, 2020)

Estes sistemas óticos são chamados sensores passivos, uma vez que detetam o reflexo ou a emissão de radiação eletromagnética proveniente de uma fonte natural.

Pelo contrário, os sensores ativos utilizam fontes de energia gerada artificialmente para detetar as respostas dos objetos irradiados. Um exemplo destes sensores são os *RADio Detection And Ranging* (RADAR) que transmitem as ondas de rádio através de uma antena e dependendo das propriedades do objeto irradiado pode receber toda, parte ou nenhuma energia que foi emitida (componente deteção do radar). O sinal recebido viaja uma quantidade de tempo proporcional à distância entre a antena e o objeto (componente Ranging do radar). Estes possuem inúmeras vantagens face à utilização de sensores óticos, como:

- Disponibilidade de utilização em operações diurnas e noturnas (uma vez que não depende da iluminação do sol);
- As nuvens e a chuva têm pouca interferência na sua utilização, é tolerante a qualquer tipo de clima atmosférico;

- Altamente sensível às propriedades seus alvos (i.e., propriedades dielétricas, rugosidade das superfícies e até da própria estrutura do alvo);
- É capaz de penetrar as superfícies dos alvos.

No entanto, as utilizações de tecnologias deste calibre possuem alguns inconvenientes, maioritariamente devido à sua complexidade. Em primeiro lugar, o próprio funcionamento e processamento são processos muito complexos. Por ser extremamente sensível, as imagens contêm *speckle*, que é o nome dado ao ruído da imagem, e dificulta a interpretação.

O Synthetic Aperture Radar (SAR) introduzido por Wiley (1985), matemático na empresa Goodyear Aircraft Company é uma das formas de criar imagens utilizando sensores ativos. Consiste num radar, maioritariamente aéreo ou espacial, que tem a sua antena a emitir radiação para o lado ('*sidelooking*'). (Doerry and Dickey, 2004). De maneira similar ao radar convencional, o SAR envia sequencialmente pulsos eletromagnéticos e os ecos ('*backscatter*') são recolhidos pela antena do radar, no caso particular do SAR os tempos de transmissão e receção acontecem em posições diferentes pelo facto de a plataforma em que este está inserido estar em movimento. Uma combinação coerente entre os sinais recebidos permite a construção das imagens SAR como se de uma antena maior, do que a que de facto é, se tratasse.

O movimento da plataforma permite o *scan* do terreno na direção do movimento (azimuth). O raio é direcionado para o lado e para baixo em direção da superfície. O feixe é largo verticalmente e por isso interceta a superfície numa oval com o eixo estendido na direção da zona alvo (range) que representa a distancia de visão. (Olmsted, 1993)

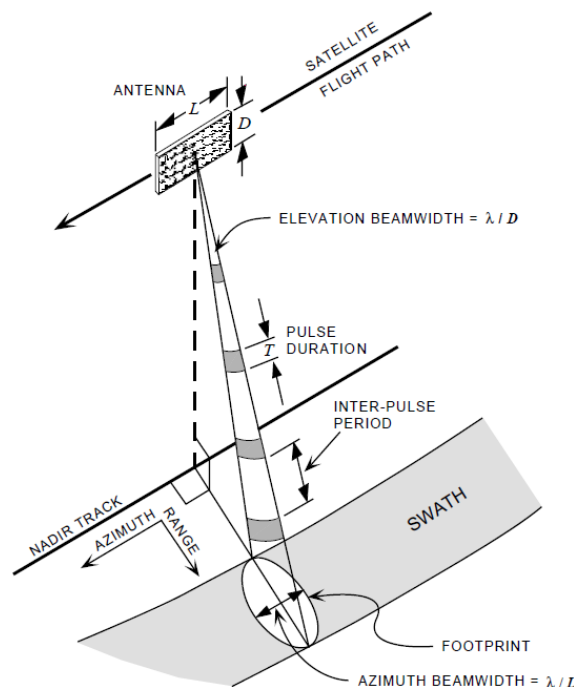


Figura 1: Esquema que representa o cenário de obtenção das imagens. (Crédito: (Olmsted, 1993))

A imagem resultado da captura do SAR, Figura 1, é normalmente uma representação 2D da reflexão de uma zona alvo que inclui dimensões de *range* e *azimuth*. (Moreira et al., 2013)

O comprimento de onda utilizado pelo radar é uma característica muito importante a considerar quando se trabalha com SAR, uma vez que esta determina de que maneira o sinal radar irá interagir com a superfície e quão fundo uma onda consegue penetrar por entre um meio. (Herndon et al., 2020)

Band	Frequency	Wavelength	Typical Application
Ka	27 – 40 GHz	1.1 – 0.8 cm	Rarely used for SAR (airport surveillance)
K	18 – 27 GHz	1.7 – 1.1 cm	rarely used (H ₂ O absorption)
Ku	12 – 18 GHz	2.4 – 1.7 cm	rarely used for SAR (satellite altimetry)
X	8 – 12 GHz	3.8 – 2.4 cm	High resolution SAR (urban monitoring; ice and snow, little penetration into vegetation cover; fast coherence decay in vegetated areas)
C	4 – 8 GHz	7.5 – 3.8 cm	SAR Workhorse (global mapping; change detection; monitoring of areas with low to moderate penetration; higher coherence); ice, ocean maritime navigation
S	2 – 4 GHz	15 – 7.5 cm	Little but increasing use for SAR-based Earth observation; agriculture monitoring (NISAR will carry an S-band channel; expands C-band applications to higher vegetation density)
L	1 – 2 GHz	30 – 15 cm	Medium resolution SAR (geophysical monitoring; biomass and vegetation mapping; high penetration, InSAR)
P	0.3 – 1 GHz	100 – 30 cm	Biomass. First p-band spaceborne SAR will be launched ~2020; vegetation mapping and assessment. Experimental SAR.

Figura 2: Diferentes comprimentos de onda utilizados por radares SAR e suas aplicações. (Crédito: NASA)

Na Figura 2 pode ser um resumo dos diferentes comprimentos de onda. Por exemplo a radiação do tipo X tem pouca capacidade de penetrar numa floresta e por isso interage maioritariamente com o topo das árvores. No entanto, a banda S já permite uma melhor interação com o tronco das árvores. O comprimento de onda não afeta só as florestas, influencia também a interação com o solo ou até gelo. (Herndon et al., 2020)

2.1.1 Interpretação das Imagens

É possível obter um mapa de terreno geradas a partir do SAR, no entanto a uma grande diferença entre imagens óticas e SAR. As imagens SAR são não são consideradas imagens no seu sentido literal. Assim de maneira a entender imagens SAR é necessário compreender os seguintes conceitos.

- **Foreshortening:** Como o SAR é um radar, os resultados do '*backscatter*' serão organizados na imagem com base na distância entre o alvo e a antena, o que causa algumas distorções geométricas na imagem. Como se pode ver na Figura 3 a inclinação A-B é comprimida no plano do '*slant*' porque o sinal chega ao ponto B pouco tempo depois de chegar ao ponto A. Isto faz com que objetos com inclinação como uma montanha, pareça mais inclinada e com um canto brilhante, Figura 4;
- **Layover:** Este é um caso extremo de '*foreshortening*' em que um objeto é tão alto que o sinal radar chega primeiro ao ponto B e depois ao A (Figura 3). No plano, o ponto B é colocada mais perto do sensor e o ponto A fica escurecido, como se o topo fosse desenhado no pé da montanha, Figura 4;

- **Shadowing:** Este fenômeno é causado pelos mesmos motivos que existem sombras numa fotografia (i.e., existe um bloqueio no caminho da radiação), como se pode ver na Figura 3. No entanto, em imagens óticas é possível ver as sombras através da radiação refletida pela atmosfera, nas imagens SAR não existe informações, uma vez que não há sinal refletido, Figura 4.

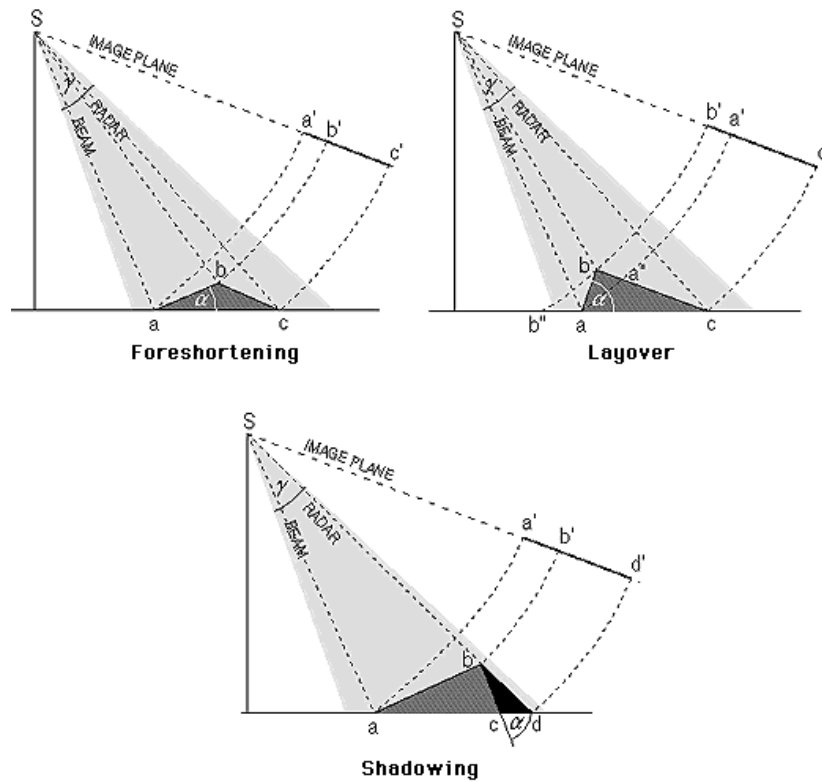


Figura 3: Esquema de distorções geométricas associadas ao SAR. (Crédito: ESA)

Embora uma imagem radar pareça uma versão monocromática de uma imagem ótica, essa impressão é errada. A intensidade dos pixels numa imagem radar não é dependente da cor de um objeto, mas sim da quantidade de energia transmitida, das propriedades materiais do objeto, da estrutura física do mesmo, e do ângulo pelo qual é visto. Assim uma imagem representa processos físicos e cada pixel representa medições. A imagem é interpretada com base no entendimento desses processos físicos. O processamento de uma imagem SAR, de forma breve, é resultado de uma operação de convolução sobre os dados *Raw* com uma função de referência de 'range'. Seguido de uma outra operação de convolução desta vez sobre uma função de referência de 'azimuth' que varia consoante o 'range'. O mesmo pode ser visto na Figura 5

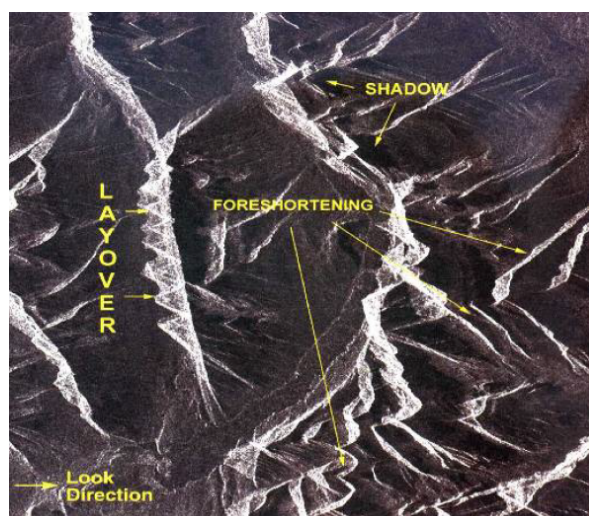


Figura 4: Exemplo de distorções geométricas associadas ao SAR. (Crédito: (Le Toan, 2007))

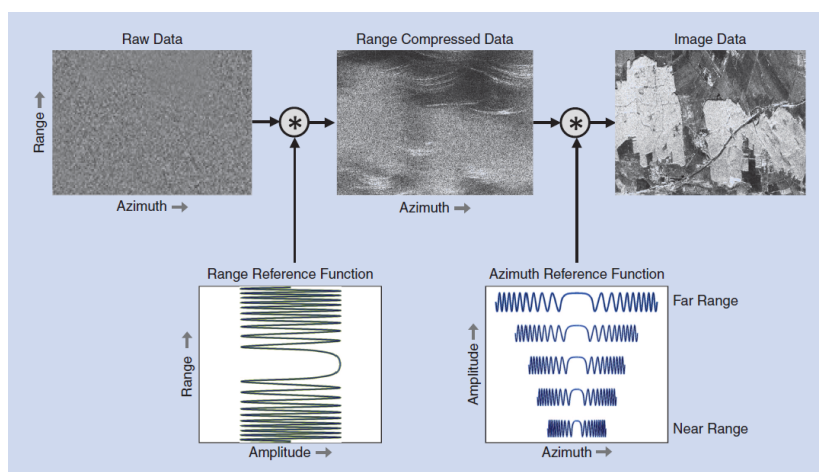


Figura 5: Processo de transformação dos dados *Raw* numa imagem SAR. (Crédito: (Moreira et al., 2013))

2.2 INTELIGÊNCIA ARTIFICIAL

Desde o seu aparecimento na década de 40, do Século XX, que está demonstrado que os computadores conseguem ser preparados para executar tarefas de extrema complexidade. Depois do desenvolvimento da máquina "enigma" que ajudou os aliados a vencer a guerra, Turing (1995) questionou-se acerca da possibilidade de uma máquina pensar.

Passados 80 anos, o termo Inteligência artificial é banalizado na sociedade. Segundo o dicionário de oxford, o termo 'Inteligência Artificial' "refere-se ao estudo e desenvolvimento de sistemas computacionais capazes de imitar o comportamento inteligente dos humanos". A inteligência humana aparece na definição de forma intencional, uma vez que nem mesmo um comportamento extremamente complexo por parte de um inseto é caracterizado como inteligente. A diferença está na capacidade de adaptação, como diz Coopeland (2020), Professor de filosofia da Universidade de Canterbury na Nova Zelândia.

Niko Tiberghen, vencedor de um prêmio Nobel, realizou um estudo com a espécie Esgana-gato (*Gasterosteus aculeatus*) em que usou um peixe de madeira de maneira a perceber qual o efeito no peixe que estava a guardar o território, e verificou o peixe territorial não apresentava a capacidade de distinção entre o peixe de madeira e um peixe invasor. A sua incapacidade de se adaptar demonstra que o animal não apresenta sinais de inteligência. (Madgwick, 2017)

Apesar de o conceito de inteligência ser muito controverso na história da psicologia e muitos investigadores na área apresentarem diferentes definições para o termo, as correntes atuais tendem a definir, de um modo generalizado, a inteligência como a habilidade de aprender através da experiência, de reconhecer problemas, e de resolver problemas. (Cherry, 2019)

Nos dias de hoje a inteligência artificial assume diversas formas e o conceito está, como dito anteriormente espalhado pelas diversas áreas.

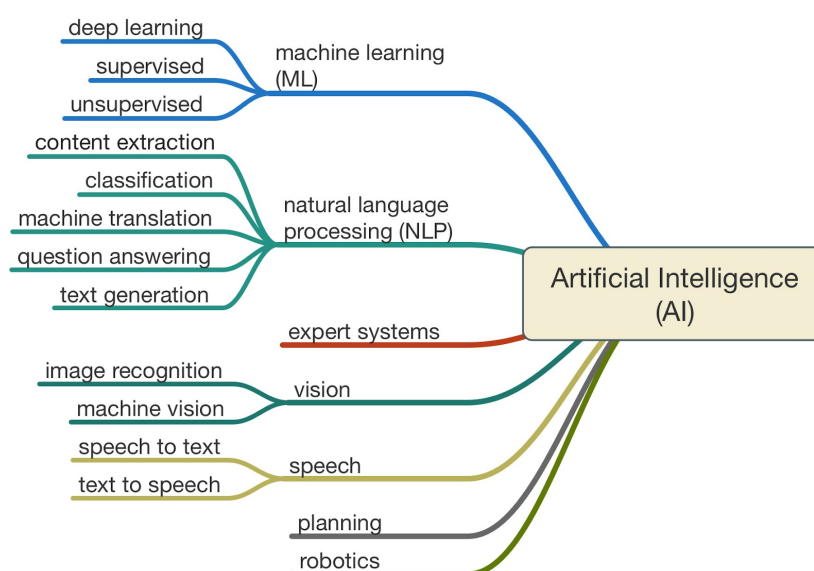


Figura 6: Ramos de Inteligência artificial. (Crédito: (Mills, 2016))

Na Figura 6 encontram-se ilustrados os ramos de inteligência artificial, segundo (Mills, 2016), que se estendem desde a robótica até ao processamento de linguagem natural. Os avanços científicos e tecnológicos vividos nos últimos anos levam-nos a incorporar inteligência em praticamente todas as atividades.

2.2.1 Machine Learning

Aprendizagem é, como referenciado anteriormente, uma das características de um ser inteligente, e diz respeito ao processo de adquirir novos comportamentos, habilidades ou conhecimento.

A primeira vez que o termo foi associado a máquinas foi em 1952, por Arthur Samuel da IBM, quando este construiu um programa para jogar damas. (Stanford Edu)

O programa era capaz de memorizar todas as posições que tinha visto e combinava esta informação com um

valores de uma função objetivo. Samuel chamou de *rote learning* e foi então o primeiro a referenciar o termo "machine Learning".

Para Samuel, *machine learning* (ML) é definido como "capacidade de um computador aprender sem ser explicitamente programado".

Machine Learning é o estudo de algoritmos computacionais que permitem a programas evoluírem automaticamente através da experiência. (Mitchell, 1997)

Ambas as definições, apesar de separadas quase 40 anos, referem-se ao termo como capacidade de resolver um problema com base em conhecimento adquirido anteriormente e não determinístico.

Existem várias formas de atingir o objetivo de fazer uma máquina evoluir de uma forma não explicitamente programada, dependendo do tipo de experiência que é lhe é dada. Assim, é possível dividir a aprendizagem em três categorias.

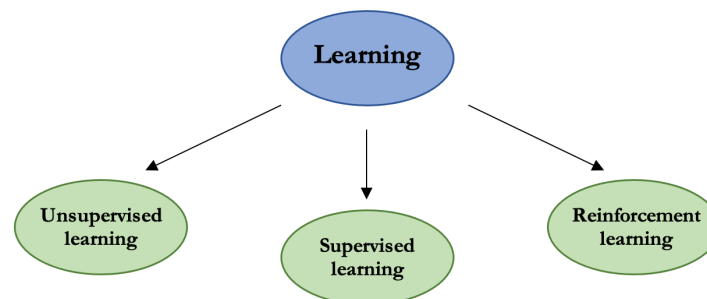


Figura 7: Diferentes tipos de aprendizagem

A Figura 7 divide então os tipo de aprendizagem em **Supervised** (supervisionada), **Unsupervised** (não supervisionada), e **Reinforcement** (por reforço).

O objetivo da primeira é através de dados de input X fazer com que a máquina aprenda o mapeamento para um outro conjunto de dados, também fornecido, Y (*labeled data*). O processo de aprendizagem é muito semelhante a um professor ensinar os seus alunos através da utilização de cartões com figuras. Este tipo de aprendizagem é normalmente utilizado em problemas de classificação.

O segundo tipo de aprendizagem existente tem o nome de não supervisionado. Em contraste com o anterior no processo de treino deste método não é fornecido ao sistema qualquer ponto de chegada. São fornecidos dados de input X e é no final do processo a máquina agrupa ou organiza os dados de uma maneira inteligente e com sentido. Tipo de aprendizagem muito utilizados por empresas como Netflix e Amazon em sistemas de recomendação, por exemplo através do agrupamento de pessoas de acordo com os gostos e partindo daí sugerir ao utilizador outros produtos que as pessoas do mesmo *cluster* gostaram, ou juntar produtos por sugerir ao utilizador outros produtos idênticos.

A terceira forma de aprendizagem demonstrada na figura diz respeito a *reinforcement learning* ou aprendizam por esforço, este tipo de aprendizagem é baseado em tentativa-erro. Isto é possível através da utilização de uma função ou algoritmo que provoca um sentimento de recompensa positiva quando a ação traz efeitos positivos e

uma penalização quando a ação não é positiva. Este tipo de algoritmos tem sido bastante usado em problemas industriais envolvendo *robots*.

Mais informações acerca dos métodos de treino pode ser consultada em [Haldorai and Ramu \(2019\)](#) ou em [Russell and Norvig \(2009\)](#).

A seleção do método indicado para o problema é uma tarefa muito importante na construção de algoritmos *machine learning*, a escolha do método de aprendizagem errado pode levar a resultados não satisfatórios ou até ao fracasso da solução. De seguida, vão ser abordados os modelos supervisionados e não supervisionados mais relacionados com o projeto, bem como ferramentas mais utilizadas no desenvolvimento de soluções *machine learning*.

Supervised Learning

Aprendizagem supervisionada é definida por [Russell and Norvig \(2009\)](#), como a aproximação de uma função h a uma função f que dado um conjunto de dados (x_1, y_1) em que cada y é gerado pela função $y = f(x)$. x e y não necessitam de ser obrigatoriamente números. De uma forma geral, este método consiste na aprendizagem por parte da máquina de uma função h que transforma dados X em Y . A função h é uma das possíveis dentro de espaço S que é definido como espaço de hipóteses. Por vezes, não é possível o encontro perfeito entre a função real f e a hipótese h e por isso é escolhido uma hipótese h^* , dentro de S que é caracterizada pela equação

$$h^* = \operatorname{argmax} f(x, y)$$

Que seleciona como ótima a melhor aproximação à função f .

Antes do início do treino propriamente dito o conjunto de Dados D é dividido em D_{train} , D_{val} , D_{test} que correspondem às três fases do processo de aprendizagem de um algoritmo supervisionado, como é referido em [Merkle \(2018\)](#). A 1ª é fase de treino do algoritmo, em 2ª Fase de validação do treino, e por fim, a fase de teste do resultado do treino do modelo.

- **Fase de Treino:** Durante esta fase é quando o modelo é alimentado com os dados que possuem *label*, de maneira a que este encontre neles os padrões necessários para reconhecer determinada *label*. Ao longo do processo de treino os parâmetros da função aproximada vão sendo aproximados de acordo com uma função de *Loss* que penaliza previsões incorretas com o objetivo de encontrar os parâmetros com menor erro de previsão. Este processo é feito repetitivamente, o que resulta numa melhoria de performance do modelo para os dados de treino que o modelo está constantemente a "ver".
Porém, não quer dizer que o mesmo se verifique para novos dados. Aí entra a fase de validação.
- **Fase de Validação:** Esta fase do treino serve para fazer uma avaliação da capacidade de generalização do modelo que estamos a treinar. Nesta fase são enviados para o modelo dados que não possuem *label* para que este procure os padrões que recolheu dos dados de treino e identifique assim a *label* dos dados novos. A avaliação do modelo é feita sobre a *loss function* nestes dados. Sendo este um processo iterativo (treino, validação) até que se encontre o melhor modelo.

- *Fase de Teste*: O objetivo desta fase é de averiguar a performance do modelo, já com o treino concluído, num novo conjunto de dados sem *label*. É avaliado por norma também no valor da *loss function* para o registo. Esta fase é de extrema importância uma vez que aqui é avaliada a capacidade de generalização do modelo treinado e consequentemente a aptidão do modelo realizar a tarefa para que foi desenhado e construído.

A qualidade de uma qualquer algoritmo de *machine learning* depende diretamente da qualidade dos seus dados, isto significa que na preparação de um modelo para uma determinada tarefa é importante fornecer ao mesmo dados de treino (X, Y) reais (i.e., o modelo deve treinar com dados aproximados àqueles que futuramente irá receber quando estiver em modo produção). (Kuhn and Johnson, 2019)

Os algoritmos supervisionados que vamos abordar são as redes neurais e seus derivados.

Artificial Neural Networks

A introdução do conceito de *Artificial Neural Network* (ANN ou NN), em português "Rede Neuronal Artificial", remonta início da década de 40 quando McCulloch and Pitts (1943) desenvolveram um sistema conceptual baseado no funcionamento do cérebro humano. No seguimento deste trabalho, Rosenblatt (1958) desenvolve assim este conceito e cria o primeiro *perceptron* no final de 1950s. Um *perceptron* é então uma unidade computacional, que imita o funcionamento do neurónio, e que dado um input produz um output. De uma forma simples, o output do *perceptron* é dado pela seguinte equação.

$$z = \sum_{i=0}^n x_i w_i \quad (1)$$

$$y = f(z) = \begin{cases} 0 & , z \leq 0 \\ 1 & , z > 0 \end{cases} \quad (2)$$

Assim, o *perceptron* retornava o valor 1 caso fosse ativado, ou então o valor 0, caso contrário. Muito semelhante ao funcionamento do neurónio humano que apenas possui os estados ativado (1) e não ativado (0).

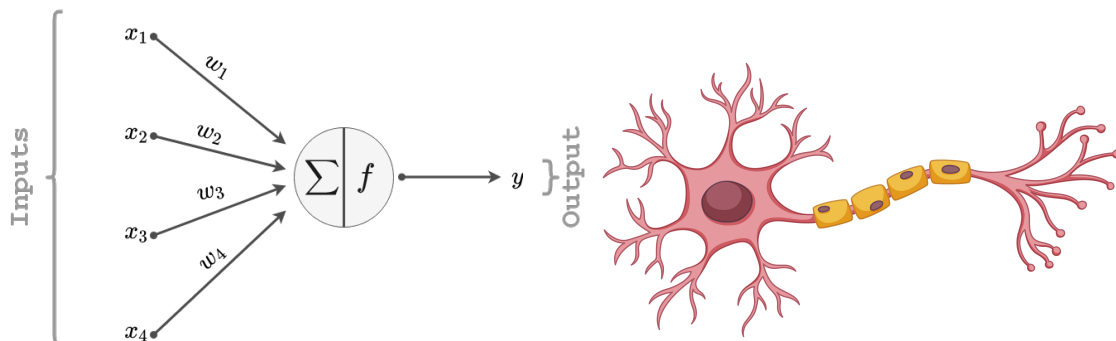


Figura 8: Na figura da esquerda vemos o *perceptron* e na figura da direita está representado um neurónio humano

A Figura 8 tem o propósito de mostrar a semelhança entre o *perceptron* e um neurónio humano. Tal como no neurónio humano, um neurónio das ANN recebe os inputs pela sua camada de input, realiza alguns cálculos e envia o output pela outra extremidade.

Como se pode verificar pela equação (2), o output de um neurónio corresponde ao somatório da multiplicação dos pesos e dos inputs e o resultado é canalizado por uma função de ativação f . existem inúmeras funções de ativação que podem ser usadas em redes neuronais, as mais conhecidas são então, **sigmoid**, **TanH** e a **ReLU**

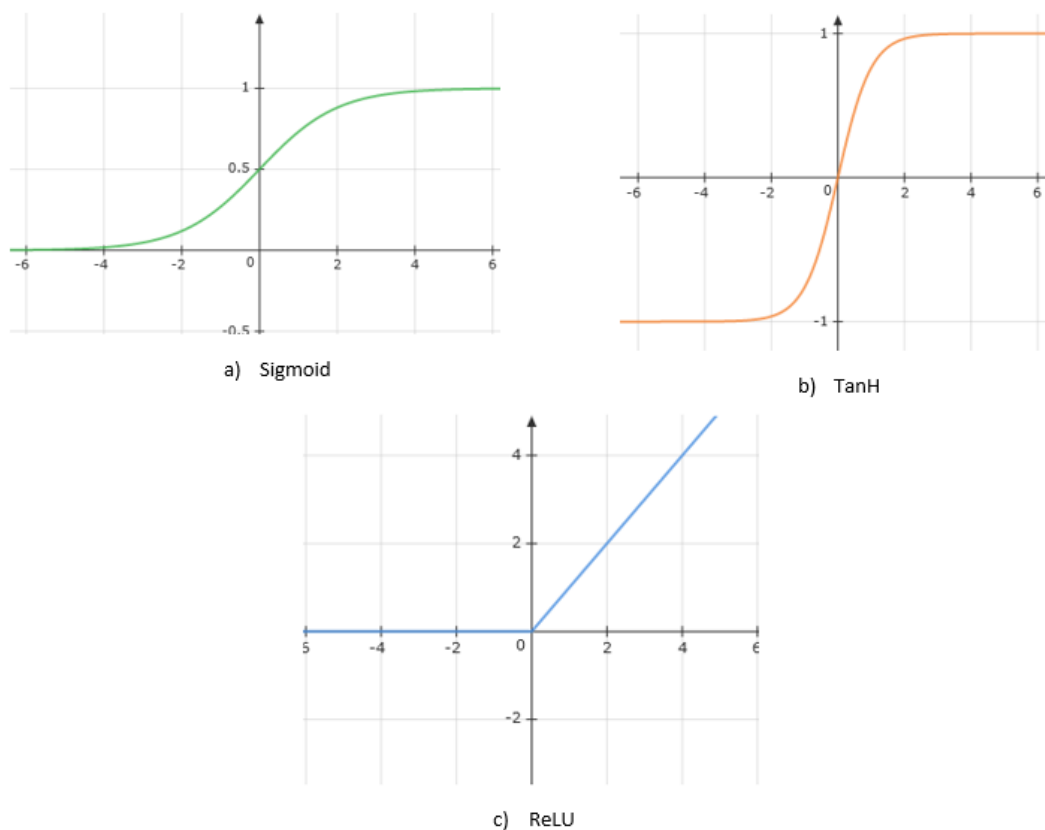


Figura 9: Principais funções de ativação usadas em NN: **sigmoid**, **TanH** e **ReLU**

Como é possível ver na Figura 9-a) a função *Sigmoid* apresenta um intervalo de valores entre 0 e 1, por este motivo são biologicamente plausíveis, uma vez que tem um comportamento semelhante ao do neurónio humano. Porém, a derivada da função *Sigmoid* é sempre < 1 , característica problemática uma vez que faz desvanecer o gradiente durante a propagação.

Na Figura 9-b) encontra-se a função *TanH*, que é similar à *sigmoid*, mas os seus valores variam entre -1 e 1 . E uma vez que a sua derivada é 1 em $x = 0$, torna esta função preferível em relação à *Sigmoid*.

Por fim, na Figura 9-c), está representada a função *Rectified Linear Units* (ReLU), uma função muito idêntica à função identidade, mas que toma o valor de 0 se $x < 0$. A ReLU é muito estável uma vez que a sua derivada apenas toma os valores 0 (quando $x < 0$) e 1 (quando $x > 0$). Uma vantagem da função face às restantes é a sua eficiência face às vistas acima e é uma das descobertas que contribuiu de forma significativa para a recente

popularidade de *Deep Learning*. A sua desvantagem é a tendência para a "morte" de neurónios que a usam, isto acontece devido ao facto nos locais onde a função é negativa, a derivada passa a 0 e assim os valores desse neurónio não são atualizados.

A utilização de um simples *perceptron* torna apenas possível um aprendizagem em que exista uma separação linear dos dados, o que limita imenso a sua utilidade. Por forma a colmatar esta barreira, é possível a ligação de diversos neurónios e distribuídos por diferentes camadas numa grafo acíclico, formando assim uma rede neuronal ou também chamada de *perceptron* multi camada. Existe também um outro tipo de neurónio, *bias b* cuja função é ajudar a rede a controlar quando a função de ativação é de facto ativada. Mais informações acerca da necessidade do *bias* pode ser encontrada em (Gebel, 2020). Numa rede neuronal formada por várias camadas, normalmente a primeira camada é denominada de *input layer*, a última (onde saem os resultados) é denominada de *output layer* e as camadas que existam entre as anteriores têm o nome de *hidden layers*.

A Figura 10 mostra um exemplo de uma rede neuronal com uma *input layer*, uma *hidden layer* e uma *output layer* e todas as ligações existentes entre elas. É importante notar que a arquitetura de uma rede neuronal pode variar de acordo com o número de camadas, unidades por camada, e funções de ativação.

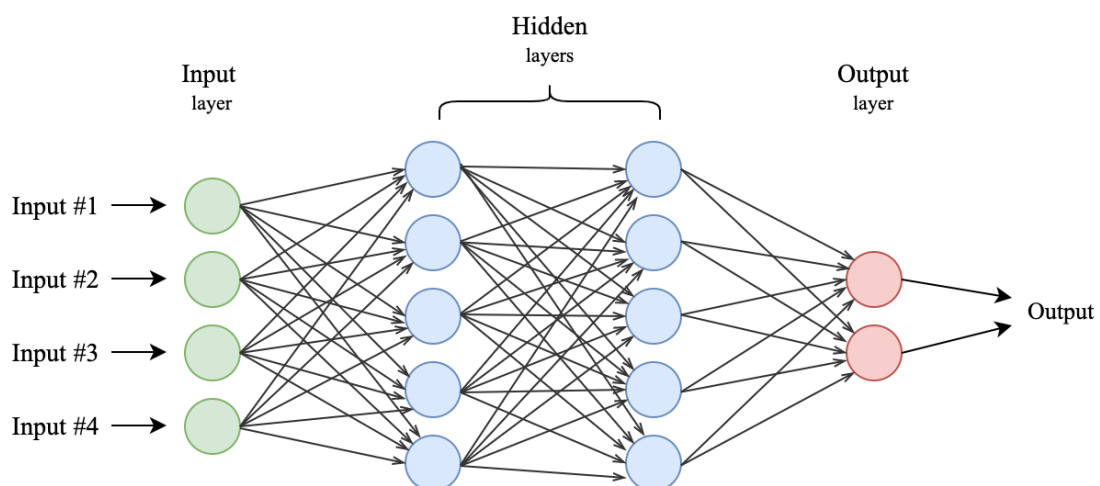


Figura 10: Exemplo de uma rede neuronal. Rede com 4 inputs, 2 *hidden layers*, e uma *output layer*

Como explicado anteriormente, a ligação entre nodos é ponderada, e de forma similar ao funcionamento do cérebro, quanto mais o peso da ligação mas influência esta tem sobre os seguintes. O objetivo é que na passagem dos dados pela rede, mais informação é extraída e como consequência mais esta aprende acerca dos dados. No final da rede, na camada de output, é onde a rede responde aos dados que lhe foram introduzidos.

Fase de Treino Uma vez que o treino das NN é um processo supervisionado, o objetivo do treino é fazer o modelo aprender a fazer as previsões corretas com base num conjunto de dados de input. Tal como dito anteriormente, a função de *loss* é a responsável por nos dizer o quão bom é um modelo na execução de uma determinada tarefa. O caso das NN não é exceção, para a construção de um bom modelo é necessária a construção de uma boa função de *Loss*, como exemplo será utilizado a função de erro médio quadrático que é

definida pela equação.

$$C = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3)$$

em que n representa o número de amostras, y é o resultado que se quer que a rede aprenda e $\hat{y}$ é a previsão da rede. Faz-se então a diferença entre estes valores e por fim são elevados ao quadrado de maneira a penalizar tanto diferenças positivas como negativas.

No início do processo de treino todos os pesos são inicializados com valores aleatórios, esta inicialização aleatória não é obrigatória, apesar de ser o modo *default* de inicialização. Logicamente que esta atribuição aleatória de pesos está longe dos valores ótimos. Inicia-se assim o processo de treino com um valor altíssimo de *Loss* e o objetivo é no final do processo de treino obter valores de *Loss* de menor dimensão possível. Trata-se assim de um problema de minimização da função *Loss*. A procura pela minimização da *Loss function* e não maximização de uma outra função de desempenho deve-se ao facto de ser mais fácil otimizar a função de *Loss*, e ao longo dos tempos foram desenvolvidos diversos algoritmos de otimização.

São exemplos o *Stochastic Gradient Descent* (SGD) explicado em Bottou (1998), ADAM introduzido por Kingma and Ba (2014), AdaGrad referido em Duchi et al. (2011). Mais algoritmos de otimização podem ser encontrados no artigo escrito por Smolyakov (2018), em que aborda diversos algoritmos e como funcionam.

O processo de treino é então dividido por duas etapas, o primeiro tem o nome de *feedforward* e o segundo *backpropagation*. (Schmidhuber, 2015) O primeiro diz respeito ao processo de evolução da informação ao longo da rede e é explicada através da equação (4), em que $a^{(l)}$ representa a matriz com os valores de ativação da camada l e que vão servir de entrada para a a camada $l + 1$, W representa a matriz com todos os pesos da rede e b diz respeito à matriz de *bias* da camada, f representa a função de ativação.

$$a^{(l)} = f(Wa^{(l-1)} + b) \quad (4)$$

O resultado é uma nova matriz com os valores de ativação a , da camada l .

Apesar de *feedforward* fazer parte do processo de treino, o *core* da rede neuronal é o processo de *backpropagation*. O processo de *backpropagation* em termos conceptuais é simples, consiste no ajuste dos pesos e dos *bias* ao longo da rede de maneira a ser obtido o output desejado.

Como dito anteriormente, durante o processo de treino é procurada a minimização da função de *Loss*, ou seja, é procurado um mínimo local de uma função. Como forma de exemplificar, considere-se a seguinte função $y(x) = 2 * \sin(x) + 2 * \cos(\frac{x}{2})$, e de maneira a encontrar um mínimo local é necessário calcular a derivada (uma vez que a função derivada dá-nos a tendência da função), e verificada a tendência a direção a seguir é indicada com base na tendência. Se a tendência for positiva, move-se para a esquerda, se for negativa move-se para a direita. O efeito é parecido com uma bola a descer uma colina.

A Figura 11 representa a procura pelo mínimo na função exemplo descrita acima, na etapa $T = 1$ a derivada é negativa e por isso a bola vai no sentido da direita como indica a seta, já em $T = 2$ a derivada é positiva e por isso a bola move-se para a esquerda.

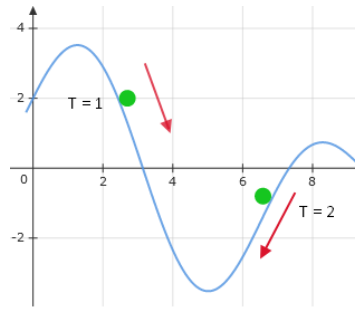


Figura 11: Exemplificação da procura do mínimo local numa função exemplo

Com a função *Loss* da rede é (praticamente) a mesma situação, mas uma vez que esta possui várias variáveis, ao contrário do exemplo que apenas possui a abcissa, é necessário realizar a derivada em relação a cada uma e calcular assim o vetor gradiente.

De maneira a explicar o funcionamento do algoritmo de *backpropagation* de uma forma mais prática, e usando o exemplo de Hansen (2019).

Seja,

Tabela 2: Legenda de variáveis a usar no exemplo de *backpropagation*

L = Última Camada	w = Pesos
$L - 1$ = Penúltima camada	a = Ativações
l = Camada genérica	b = Bias
y = Valor real a prever	f = Função de ativação
C = Função custo	$z = w \times a + b$

$$z^{(L)} = w^{(L)} \times a^{(L-1)} + b \quad (5)$$

$$a^{(L)} = f(z^{(L)}) \quad (6)$$

$$C = (a^{(L)} - y)^2 \quad (7)$$

O gradiente é calculado a partir de três equações, uma para os pesos (equação (8)), uma para os *bias* (equação (9)) e outra para as ativações (equação (10))

$$\frac{\partial C}{\partial w^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}} \quad (8)$$

$$\frac{\partial C}{\partial b^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial b^{(L)}} \quad (9)$$

$$\frac{\partial C}{\partial a^{(L-1)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \quad (10)$$

Estas equações apenas calculam o gradiente para a última camada (L). Para propagar o gradiente ao longo da rede (sentido contrário), as camadas anteriores (camadas iniciais) têm dependência das seguintes (finais). Assim para calcular o gradiente na camada ($L - 1$) são efetuados os seguintes cálculos:

$$\frac{\partial C}{\partial w^{(L-1)}} = \overbrace{\frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}}}^{\text{Cálculo Reutilizado}} \frac{\partial C}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \quad (11)$$

$$\frac{\partial C}{\partial b^{(L-1)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \quad (12)$$

$$\frac{\partial C}{\partial a^{(L-2)}} = \frac{\partial C}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial C}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial w^{(L-2)}} \quad (13)$$

E assim sucessivamente ao longo da rede, até atualizar todos os pesos e *bias*. Nas últimas camadas a serem atualizadas a maior parte dos cálculos serão reutilizados, o que não agrava a complexidade computacional do algoritmo.

Uma vez calculado o vetor gradiente e de forma análoga à função simples representada na Figura 11 é necessário caminhar na direção oposta e por isso temos que adicionar à expressão do gradiente o sinal de menos (-) que nos leva na direção oposta. Posto isto a atualização dos valores de pesos e *bias* é dado pelas expressões

$$w^{(l)} = w^{(l)} - \text{learning_rate} \times \frac{\partial C}{\partial w^{(l)}} \quad (14)$$

$$b^{(l)} = b^{(l)} - \text{learning_rate} \times \frac{\partial C}{\partial b^{(l)}} \quad (15)$$

As expressões acima mostram como atualizar o valor de um **peso** e de um **bias**, o processo é realizado para todos os pesos e todos os *bias* de maneira a fazer a rede evoluir. A variável **learning rate** serve para determinar o tamanho dos "saltos" dados em direção ao mínimo. Por norma o valor do *learning rate* vai descendo com avanço do treino de maneira a dar "passos" mais pequenos à medida que a função de custo C vai chegando a um mínimo.

Há medida que se vai movendo de camada em camada para trás, mais derivadas parciais de valores de pesos e *bias* são adicionadas, de qualquer modo é importante voltar a referir que é necessário mover o gradiente até ao início da rede para que todos os pesos e *bias* sejam atualizados e assim permitir que a rede aprenda.

De uma forma sumariada o algoritmo pode ser definido através dos seguintes passos:

1. Inicializar os pesos de forma aleatória, e os *bias* a 0;
2. Calcular para cada neurónio de cada camada a *forwardpropagation*
3. Calcular o gradiente e atualizá-lo ao realizar a *brackpropagation*
4. Atualizar os pesos com base nos valores do vetor gradiente.

Na última década, através do concurso ImageNet Large Scale Visual Recognition Challenge (ILSVRC), permitiu o aparecimento e o *Boom* de redes neurais com mais camadas que o normal, estas redes entram na categoria de *Deep Learning*. Apesar das arquiteturas conceituais de *deep learning* remotarem aos anos 80s só na última década é que foi possível evoluir o conceito e trazê-lo para os concursos e estado de arte.

Andrew Ng, fundador da Google Brain, define como core e motivação do *Deep Learning* são "Redes neurais gigantes" e os "conjuntos de dados a que conseguimos aceder".

Geoff, em 2016 numa talk para Royal Society acerca do tema *Deep Learning*, onde apresentou razões para o insucesso de *backpropagation* e estes são os mesmos apresentados por Andrew acerca da ascensão tardia de *Deep Learning* que são: 1) Os *datasets* que anotados eram milhares de vezes mais pequenos; 2) os computadores eram milhões de vezes mais lentos. *Deep learning* veio assim potenciar a nova era de *Machine Learning*. Estes algoritmos são especialmente favorecidos pelo uso de *Graphical Processing Units* (GPUs) otimizadas na multiplicação de matrizes, o que torna os processo de treino e otimização incomparavelmente mais rápidos.

Uma outra descoberta que revolucionou as redes neurais e *Deep Learning*, foi a introdução das camadas de *Dropout*, por [Srivastava et al. \(2014\)](#), na tentativa de resolver os problemas de generalização (*overfit*) que as redes podem sofrer durante a fase de treino. Os problemas de *overfit* acontecem quando depois do treino, os pesos da rede estão tão otimizados para os dados de treino que a rede não consegue atingir bons resultados com novos dados. A ideia do *dropout* é simples por natureza. A camada "deita fora" um conjunto aleatório de ativações nessa camada ao atribuir-lhes o valor 0. Isto é particularmente útil uma vez que deste modo elimina alguma redundância ou acomodação por parte da rede aos dados, obrigando-a assim a melhorar em termos de generalização. Uma nota importante sobre estas camadas é que apenas operam durante a fase de treino, e não nas fases de validação e teste.

Ainda em contexto de *deep learning* vamos explorar dois algoritmos que são fruto destes novos avanços na área, *Convolutional Neuronal Networks* e *Recurrent Neural Networks*.

CONVOLUTION NEURAL NETWORKS *Convolutional Neural Network* é tipo de rede neuronal artificial usada em sistemas de reconhecimento e processamento de imagem devido à sua capacidade de processamento de dados em formato de *grid*. Este tipo de rede teve um papel fundamental no desenvolvimento histórico de *Deep Learning*, uma vez que são um exemplo perfeito de uma aplicação, em ML, bem sucedida de conhecimentos obtidos através do estudo do cérebro humano.

Na sua forma mais básica, as redes neurais convolucionais podem ser vistas como uma rede que usa imensas cópias do mesmo neurónio. Isto permite que a rede tenha imensos neurónios e seja capaz de expressar computacionalmente, enquanto o número de parâmetros - os valores que descrevem como os neurónios se comportam - que precisam de ser aprendidos relativamente pequeno. ([Olah, 2014](#))

As CNNs surgiram pela primeira vez nos anos 90, quando [LeCun et al. \(1989\)](#) desenvolveu um sistema de reconhecimento de dígitos escritos à mão. LeCun, agora diretor de departamento de investigação AI da Facebook, foi um dos maiores impulsionadores desta tipologia. O atual interesse por *Deep Learning* e pelas CNNs começou quando [Krizhevsky et al. \(2012\)](#) ganhou o concurso imageNet com uma CNN.

A arquitetura das CNN tem por base três camadas: 1) *Convolution layer*, 2) *Pooling Layer*, 3) *Fully connected Layer*.

1) Convolution Layer: As *convolution layers* são o *core* das CNN, e é também aqui que ocorre o maior peso computacional. Esta camada é responsável por detetar e extrair *features* a partir do input. Esta operação é realizada através do produto entre 2 matrizes. Uma das matrizes é uma porção pequena dos dados de entrada e a outra é um conjunto de parâmetros de aprendizagem, denominado de *kernel* ou *filter*. A relação espacial entre *pixels* é mantida através da utilização de porções pequenas de dados.

Durante esta fase, o *kernel* desliza tanto horizontal como verticalmente. Este processo produz uma representação da imagem denominada de *feature map* ou *activation map*, quanto maior o número de *filters* mais *features* são retirados da imagem. O deslizamento tem o nome de *stride*.

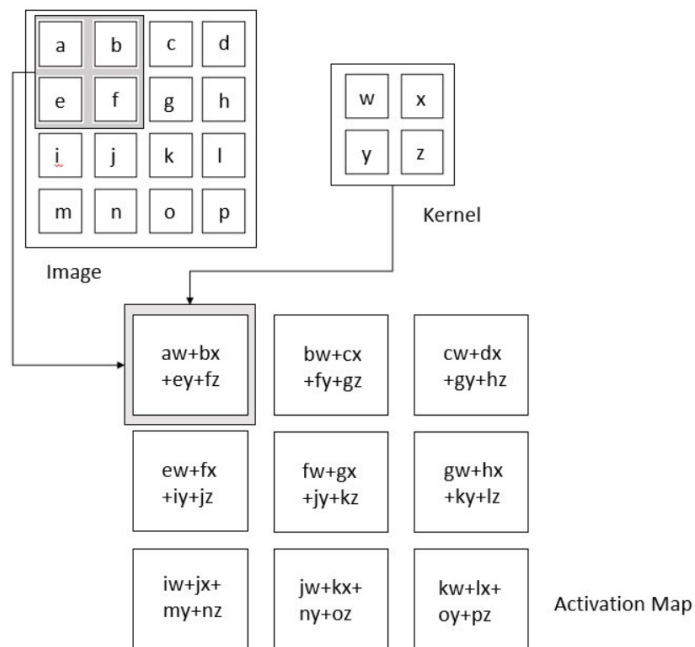


Figura 12: Exemplificação do cálculo durante a operação de convolução. [Goodfellow et al. \(2016\)](#)

O resultado da operação de convolução é dado pela seguinte expressão, para um volume de entrada de $M \times M \times D_{in}$, D_{out} é dado pela profundidade de K que representa o *kernel*, S representa o *stride* (que descreve a deslocação do *filter*), e P é referente ao *padding* (cuja função é preservar a informação através da adição de P filas na fronteira).

$$M_{out} = \frac{M - K + 2P}{S} + 1$$

Tendo como exemplo a Figura 13, temos então $M = 6$, $D = 1$ que resulta num volume de entrada de $6 \times 6 \times 1$. O *kernel* é questão tem o volume de $3 \times 3 \times 1$, ou seja $K = 3$. Em termos de *stride* a janela apenas se

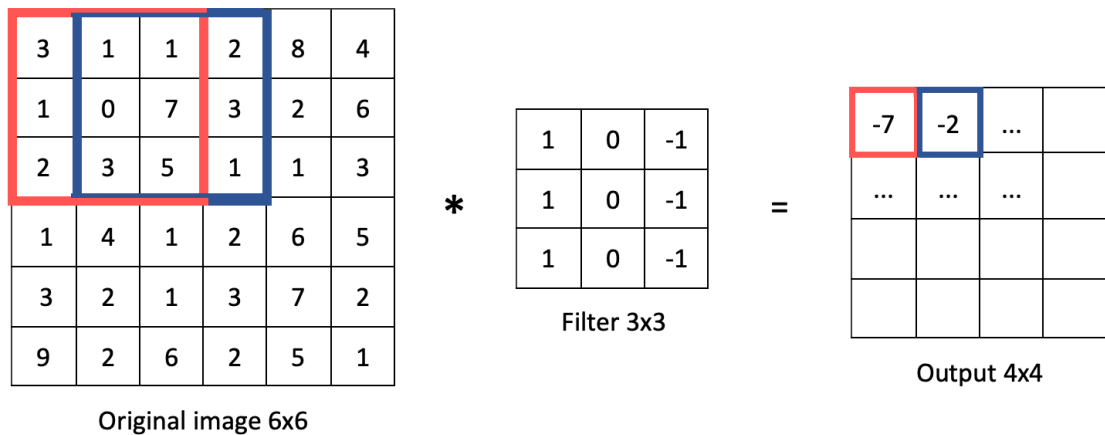


Figura 13: Operação que ocorre nas camadas convolucionais. (Crédito: towardsdatascience.com)

move uma unidade, e por isso $S = 1$. Como não existem filas de valores nas fronteiras da matriz, o *padding* P é de 0. Por aplicação da fórmula descrita acima, temos que $M_{out} = \frac{6 - 3 + 2(0)}{1} + 1 = 4$.

Uma vez que a operação de convolução é linear e as imagens não apresentam padrões lineares, são adicionadas, normalmente, a seguir as camadas convolucionais o que alguns autores chamam de camadas de não linearidade. Estas operam funções não lineares sobre os *activation maps*. Estas funcionam como funções de ativação abordadas anteriormente. São exemplo as funções Sigmoid, TanH, ReLU.

2) Pooling Layer: A seguir às camadas convolucionais, seguem-se as camadas de Pooling. O objetivo destas camadas é diminuir o tamanho espacial dos feature maps que favorece a rede em termos de computação. Esta operação é realizada substituindo o output das camadas convolucionais em determinados locais de acordo com as características dos vizinhos no *feature map*. Existem diversas funções de *pooling* como *average of neighborhood*, *L2 norm of neighborhood*, *weighted average based on distance to the central*. No entanto o processo mais popular de *pooling* tem o nome de *max pooling*, nesta função o output toma a forma do valor máximo entre os vizinhos.

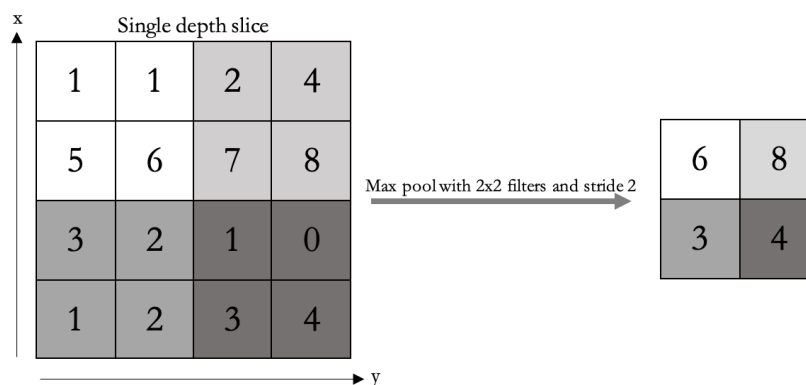


Figura 14: Operação de *pooling*. (Crédito: towardsdatascience.com)

O tamanho resultante da operação de *pooling*, para um volume de entrada de $M \times M \times D$ é dado pela seguinte expressão. Uma vez que a operação não varia com a profundidade, o valor D mantém-se.

$$M_{out} = \frac{M - K}{S} + 1$$

Aplicando a equação ao exemplo descrito na Figura 14, temos um volume de entrada de $4 \times 4 \times 1$ e um *filter* e *stride* do mesmo tamanho, (normalmente 2) e o resultado é um $M_{out} = \frac{4 - 2}{2} + 1 = 2$.

É importante notar que a operação de *pooling* é um invariante de translação que significa que um objeto é reconhecível independentemente do local onde aparece no *frame*.

3) Fully connected Layer: Esta componente final é responsável pelo mapeamento entre as *features* extraídas pelas camadas convolucionais e os outputs desejados. Esta componente é uma rede neuronal artificial normal supracitada.

Como dito anteriormente, as CNNs são extremamente importantes e utilizadas em tarefas de processamento de imagem. De forma concreta, são utilizadas em problemas relacionados com 1) deteção de objetos, reconhecimento facial em Sermanet et al. (2014) e Simonyan and Zisserman (2015), 2) segmentação semântica como mostra Shelhamer et al. (2017), 3) também há investigações no sentido de comparação de imagens através destas redes, como mostra Zagoruyko and Komodakis (2015)

O estado das CNN é também abordado pelo seu autor, em LeCun et al. (2010).

O processamento de imagens SAR também beneficiou muito da evolução das CNN, em Henry et al. (2018) foi utilizada uma arquitetura CNN com o objetivo de segmentar as imagens SAR com identificação das estradas, com um valor de 44% sobre a métrica *Intersection over Union* (IoU), o que demonstra o potencial destas arquiteturas para resolução de problemas envolvendo imagens SAR.

Em termos de deteção urbana, Shahzad et al. (2019), procurou desenvolver uma CNN capaz de identificar edifícios. Uma vez que seria útil a junção de características das imagens óticas com as imagens SAR de maneira a obter melhores e diferentes resultados. É exemplo dessa investigação a arquitetura *siamese* CNN descritas em Mou et al. (2017), com capacidades de operar sobre imagens SAR e óticas.

RECURRENT NEURAL NETWORK O ser humano interpreta e entende as situações com base no contexto. Consideremos o exemplo "Paris é a capital de ____". A resposta é facilmente França. A resposta só é fácil porque lemos a frase como uma sequência lógica e com dependência entre as palavras e não simplesmente como uma sequência de palavras a ser interpretada por si só. (Ducan, 2017)

As *Recurrent Neural Networks* (RNN) possuem uma arquitetura com capacidade de explorar as características sequenciais dos dados. Assim As *Recurrent Neural Network* são um tipo de rede neuronal que, apesar de também aprender com dados de treino, se distingue das restantes pela sua "memória", uma vez que utilizam informação de inputs anteriores de maneira a influenciar o input e output do passo atual. Ao contrario de redes neuronais convencionais em que se assume que os inputs e outputs são independentes entre eles, as RNN dependem do elemento anterior dentro da sequência.

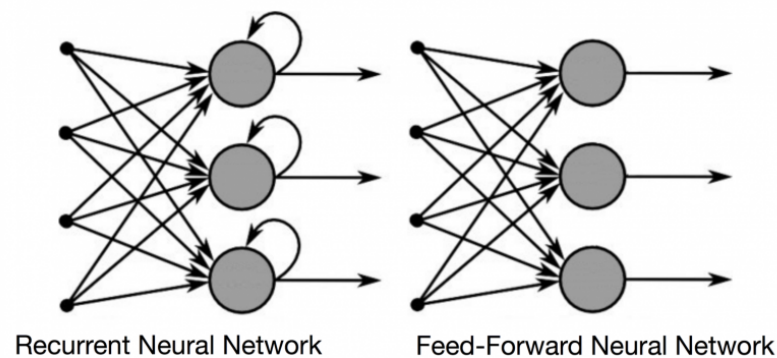


Figura 15: Diferença entre uma RNN e uma rede neuronal convencional (feed-forward). (Crédito: Built-in.com)

É importante referir que as RNN podem ser usadas numa variedade de cenários dependendo dos inputs que lhe são enviados e output desejado. Assim, estes cenários podem ser divididos em 3 classes principais:

- Input sequencial e output sequencial. Usado em tradução automática e *tagging* de discurso. Como [Singh et al. \(2017\)](#) constatou. Também [Vathsala and Holi \(2020\)](#) utilizou esta arquitetura para traduzir dados do *Twitter*.
- Input sequencial e output simples. Particularmente útil p.e numa tarefa de análise de sentimentos, através do envio de uma frase e o objetivo é receber o *feedback* da rede acerca do estado emocional do autor. Em [Majumder et al. \(2019\)](#) é possível ver que forma como esta arquitetura superou o estado de arte em dois *datasets*. Os [Jabreel and Moreno \(2019\)](#) também optaram por esta abordagem na classificação de *tweets*.
- Input simples e output sequencial. Nesta classe estão casos como descrição de uma imagem. Exemplos desta classe são vistos em [Tang and Han \(2016\)](#) ou [Gurjar et al. \(2017\)](#), em que os autores tiram partido das características das RNN para gerar uma descrição.

Em termos de treino, e como o seu antecedente, as RNN treinam através de *feedforward* e *backpropagation*, porém com algumas alterações. Uma vez que os parâmetros são partilhados por todos os passos da rede, o gradiente de cada output não depende apenas do calculo do passo que está a executar no momento mas também os passos anteriores. Este tipo de treino é chamado de *Backpropagation Through Time* (BPTT).

De seguida, será possível ver o processamento da frase "what time is it?" por parte de uma RNN, dividido em etapas desde a primeira palavra até à resposta da rede.

Como é possível ver no processo descrito em Figura 16 e o a ilustração presenta na Figura 17 há medida que avançamos na rede a primeira palavra perde influência na rede e isso pode não parecer um problema para uma frase como "Que horas são?", no entanto, com frases maiores em que o contexto é apresentado fica muito distante do fim da frase, ocorre o que é apelidado do "*short-term memory*". O problema acontece devido a fenómeno chamado "*vanishing gradient problem*", que também acontece noutras arquiteturas de NN e

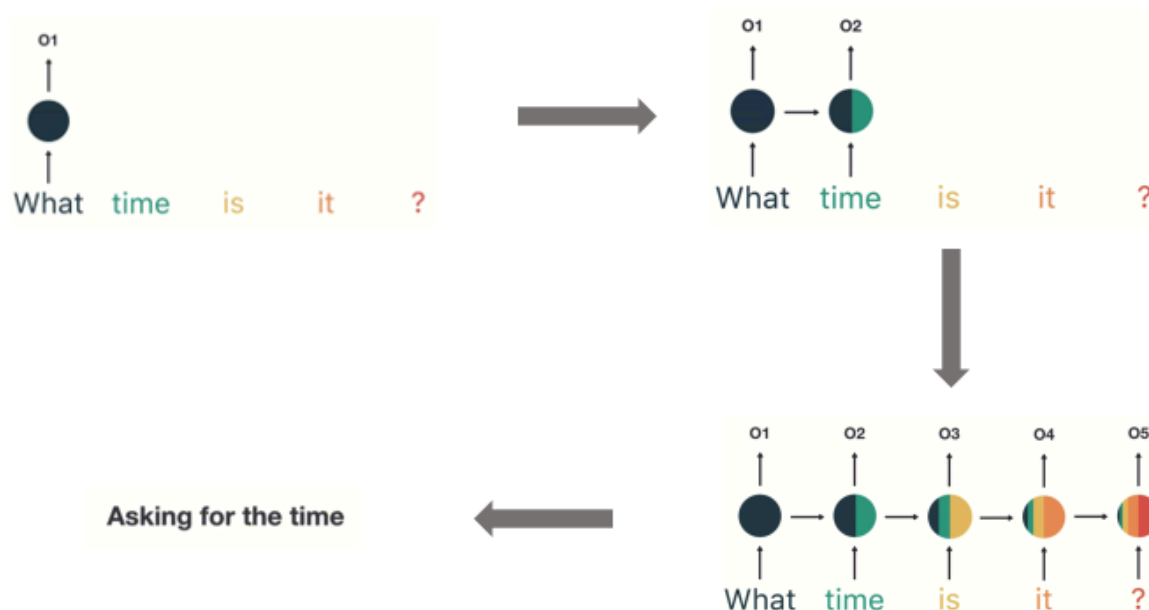


Figura 16: Processamento de uma frase por uma *recurrent neural network*. (Crédito: [Phi \(2018\)](#))



Figura 17: Demonstração ilustrada da influência das diversas fases na última palavra. (Crédito: [Phi \(2018\)](#))

é intrínseco à natureza do algoritmo de *backpropagation*. Uma vez que o gradiente é o responsável pelo ajuste dos pesos da rede e este é calculado tendo em conta as camadas anteriores, se os ajustes da camada anterior forem pequenos os ajustes da camada atual serão mais pequenos. Essa situação pode causar diminuição acentuada do gradiente ao longo da rede, a consequência dessa diminuição é a falha na aprendizagem das primeiras camadas uma vez que estes serão ligeiramente modificados. Na situação exemplo, caso a rede não consiga considerar as palavras "Que" e "horas", daria o sua previsão baseada apenas em "são?", o que é muito ambíguo, até para um humano. ([Phi, 2018](#))

De maneira a colmatar esta lacuna nas RNN, foram desenvolvidas duas redes especializadas. A primeira tem o nome de *Long Short-Term Memory* (LSTM), que foi originalmente publicada por [Hochreiter and Schmidhuber \(1997\)](#). E a segunda é *Gated Recurrent Units* (GRU), introduzida por [Cho et al. \(2014\)](#). As duas redes tem em comum o facto de usarem *gates* (portas) que são capazes de aprender informação para adicionarem ou removerem dos *hidden state*. Esta habilidade permite ultrapassar os problemas de *Short-Term memory*.

A arquitetura da rede CNN proposta por [Shahzad et al. \(2019\)](#), para identificar edifícios em imagens SAR, recorre a uma arquitetura RNN para fazer a segmentação da imagem. Uma outra aplicação de RNNs em imagens SAR é proposta por [Minh et al. \(2017\)](#), para mapear a vegetação através de comparações distantes temporalmente. Esta investigação resultou numa rede com *accuracy* acima dos 90%. Resultados estes que

demonstram a capacidade por parte das RNN em processar e diferenciar imagens temporalmente distantes de imagens SAR.

GENERATIVE ADVERSIAL NETWORK *Generative Adversarial Networks* utiliza uma abordagem baseada na teoria de jogos ao contrário das redes neurais convencionais. Introduzida por Goodfellow et al. (2014) esta rede aprende a gerar baseada numa distribuição de treino através de um jogo de 2 jogadores. As duas entidades são o *Generator* e o *Discriminator*. Estes dois adversários estão constantemente em guerra ao longo do processo de treino. Uma vez que é adotado um método competitivo de aprendizagem não é necessário preocupar-nos acerca de aproximar uma função de densidade.

Generator, é o responsável pela geração de imagens o mais semelhante possível às imagens do *dataset*. *Discriminator* tem a função de tentar adivinhar quais as imagens foram geradas pelo *generator* e quais imagens são as reais, atribuindo o valor 1 (se for real) e 0 se for *fake*.

Assim, o objetivo do *generator* é enganar o *discriminator* e o objetivo do *discriminator* é não se deixar enganar pelo *generator*. Para gerar as melhores imagens é necessário a melhor combinação de *generator* e *discriminator*, porque se o *generator* não for bom o suficiente para conseguir enganar o *discriminator*, o modelo nunca irá convergir. Se o *discriminator* não for bom o suficiente para distinguir as imagens reais das geradas, o modelo vai convergir com imagens que não fazem sentido.

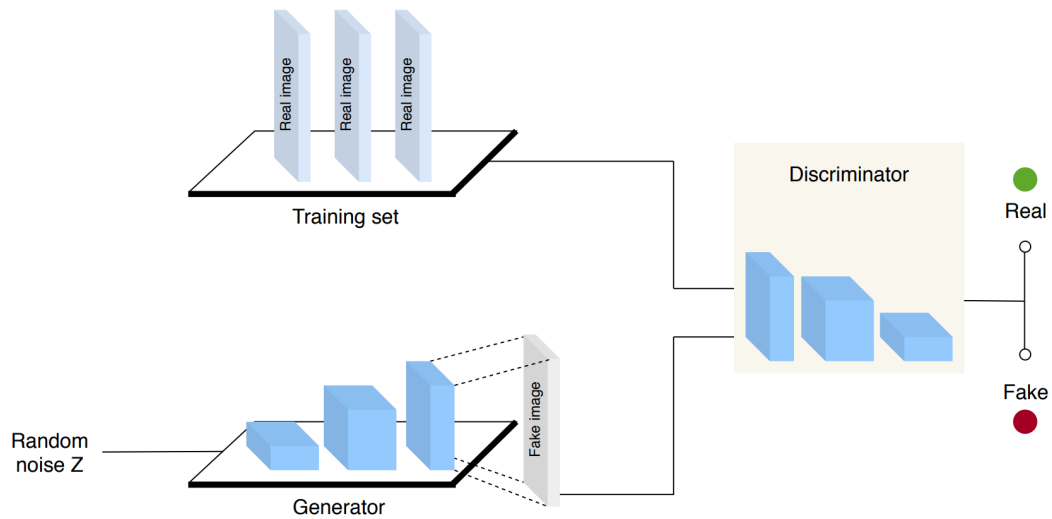


Figura 18: Arquitetura de uma rede GAN.

Segundo os autores, e uma vez que esta rede representa uma dualidade de ações e o treino é realizado em conjunto, a função objetivo do treino é representado por um *minmax*.

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{train}} \log \underbrace{D_{\theta_d}(x)}_{\text{Output D para dados reais}} + \mathbb{E}_{z \sim p(z)} \log \left(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Output D para dados gerados por G}} \right) \right]$$

O *discriminator* tenta maximizar a função objetivo, tal que $D(x)$ é perto de 1 (real) e $D(G(z))$ é perto de 0 (falso), é possível fazer o gradiente ascendente sobre a função objetivo. Por outro lado, *generator* procura minimizar a função objetivo de maneira a que $D(G(z))$ seja perto de 1 (o que significa que conseguiu enganar o *discriminator*) e por isso é possível fazer gradiente descendente sobre a função objetivo. Ao alternar entre gradiente ascendente e descendente a rede é treinada.

$$\max_{\theta_d} [\mathbb{E}_{x \sim p_{train}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z)))] \quad (16)$$

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \quad (17)$$

O gradiente ascendente representado pela equação (16), é utilizado para treinar o *discriminator*, e o gradiente descendente, equação (17), tem a função de treinar o *generator*.

No entanto, na aplicação do algoritmo foi perceptível pelos autores que a função objetivo não estava a funcionar como pretendido, isto acontecia porque quando era gerada uma amostra com grande probabilidade de ser classificada como falsa era suposto aprender com o gradiente, porém nesta fase o gradiente é praticamente lisa o que dificulta o processo de treino. Assim, ao invés de procurar minimizar a probabilidade de o *discriminator* estar correto, procurar maximizar a probabilidade de o *discriminator* estar errado. Nesse seguimento, a fórmula do gradiente do *generator* foi atualizada para a forma descrita na equação (18).

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log (D_{\theta_d}(G_{\theta_g}(z))) \quad (18)$$

Assim, quando as amostras forem más por parte do *generator* o gradiente obriga a uma maior alteração no *generator* e à medida que o *generator* for melhorando as alterações vão sendo mais pequenas. De forma sumariada o Algoritmo de treino pode ser definido da seguinte forma.

1. Treinar o *discriminator* a partir dos dados de treino, com o objetivo de o fazer aprender o conjunto de dados real;
2. Gerar novos dados com o *generator* e enviá-los ao *discriminator* para que este faça a sua previsão em relação à veracidade das imagens;
3. Atualizar (através do seu gradiente) o *Discriminator* com base nos resultados da sua previsão;
4. Atualizar (através do seu gradiente) o *generator* de acordo com os resultados obtidos pelas imagens geradas por si.

Este processo deve ser realizado iterativamente até obter o sucesso desejado.

Esta abordagem está a ser cada vez mais explorada, uma vez que o facto de ser 2 *player-game* obriga o *generator* a evoluir ao longo do processo originando assim melhores imagens geradas por parte do *generator*. Porém, o facto de se treinar 2 redes quase no mesmo instante pode resultar num processo instável.

A aplicação destas redes é extensa e os seus horizontes tendem a aumentar, [Karras et al. \(2019\)](#) criou uma arquitetura GAN que aumenta o detalhe das imagens humanas geradas por GAN através da aplicação de ruído em vários pontos da rede. Uma outra famosa aplicação das GAN foi proposta por [Zhu et al. \(2017\)](#), consiste no que o autor denominou de *CycleGAN* e é capaz de realizar *image-to-image translation* isto é passar de uma imagem e atribuir-lhe propriedades de outras imagens. São exemplo disso, a passagem de uma fotografia para uma pintura de Van Gogh ou Monet.

A introdução de GANs no universo SAR, veio trazer novas possibilidades e diferentes maneiras de resolver alguns problemas. Uma forma de segmentação foi proposta por [Geng et al. \(2017\)](#), através do recurso combinado GANs e *autoencoders*, obtendo para alguns objetos qualidade na segmentação superior a 90% e os objetos com pior cerca de 50%, resultados muito satisfatórios que incentivam a investigação e o investimento nestas arquiteturas.

Uma outra abordagem potencializada pelas GAN é a geração de imagens SAR de alta qualidade. O resultado é extremamente interessante, ultrapassando os resultados de algumas técnicas manuais de melhoria de imagens SAR.

Num artigo publicado no instituto *Multidisciplinary Digital Publishing Institute MDPI* na área de *remote sensing*, [Reyes et al. \(2019\)](#), propôs a utilização da arquitetura CycleGAN para q geração de imagens óticas a partir de imagens SAR. Foi medida o resultado da aplicação através da comparação do algoritmo de segmentação das estradas apresentado por [Henry et al. \(2018\)](#). Os resultados não superaram a aplicação do algoritmo à imagem SAR, daí, os autores concluíram que as imagens SAR possuem informação que não é preservada na passagem da imagem para ótica, e que por isso a aplicação da rede para gerar imagens óticas não traz benefícios à rede de segmentação.

Unsupervised Learning

Aprendizagem não-supervisionada, segundo Alex Graves e Kelly Clancy da DeepMind, é um paradigma desenhado de modo a criar agentes inteligentes autónomos que extraem informações dos dados sem um objetivo em particular.

Ao contrário da abordagem supervisionada, é alimentado para o modelo de ML dados que não foram anotados ou classificados e é suposto que este aprenda padrões com o mínimo de intervenção humana possível.

O processo de treino dos modelos com abordagem não supervisionada é conceptualmente simples. Durante o processo de treino, o algoritmo analisa os dados de entrada, que como dito anteriormente não possuem qualquer classificação, e tenta extrair conclusões a partir da sua estrutura. Assim, destes algoritmos é expectável que gere um output específico com base em dados não organizados de entrada.

Os maiores problemas face ao seu dual, supervised learning, são: 1) o facto de existir alguma dificuldade em definir a *accuracy* dos outputs, 2) necessidade de um análise exaustiva por parte do utilizador de maneira a perceber o porquê da distinção feita pelo modelo. Este tipo de algoritmos é bastante utilizado em processo de *anomaly detection*, *clustering* de dados.

FUZZY C-MEANS Fuzzy C-means Clustering (FCM) é um algoritmo não supervisionado, introduzido por [Dunn \(1973\)](#). O algoritmo trabalha atribuindo associação de cada dado a cada *cluster* com base na distancia entre o ponto e o *cluster*. Quanto mais perto está o ponto do *cluster* mais este pertence ao dito *cluster*. A cada iteração o valor de associação de um ponto (dado por μ) aos *clusters* c é atualizado.

A atualização do valor de associação é dada pela seguinte expressão.

$$\mu_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{d_{ij}}{d_{ik}} \right)^{\frac{2}{m-1}}} \quad (19)$$

A atualização dos centróides é dada pela expressão.

$$c_j = \frac{\sum_{i=1}^n \mu_{ij}^m x_i}{\sum_{i=1}^n \mu_{ij}^m}, \forall j = 1, 2, \dots, c \quad (20)$$

O Objetivo desta abordagem é minimizar a função objetivo $G(U, C)$, em U é a matriz com todas as associações existentes e C é o conjunto de *clusters*.

$$G(U, C) = \sum_{i=1}^n \sum_{j=1}^c \mu_{ij}^m \|x_i - c_j\|^2 \quad (21)$$

O algoritmo funciona da seguinte forma:

1. Os *clusters* (c) são iniciados de uma forma aleatória;
2. É calculada a associação de cada ponto dos dados a todos os *clusters*, através da equação (19);
3. São calculados os *fuzzy centers* com recurso à equação (20).

O algoritmo é executado iterativamente até que valor mínimo da função objetivo (equação (21)) seja atingido ou a condição de paragem $\|U^{k+1} - U^k\| < \beta$, em que k é o número da iteração e β representa um *threshold*, seja satisfeita.

Em relação ao seu antecessor, *k-means*, FCM obtém melhores resultados em dados que sobrepostos. Uma outra vantagem é o facto de ao invés de atribuir a associação a um único *cluster*, FCM relaciona os pontos com todos os *cluster* e assim pode pertencer a mais que um *cluster*.

O problema de especificar o número de *clusters* *a priori* mantém-se neste algoritmo o que representa uma desvantagem, bem como a definição do critério de paragem que dependendo do seu valor pode mais custoso em termos computacionais ou no caso contrário, produzir piores resultados.

Formas interessantes de segmentar uma imagem SAR, foram propostas por [Feng et al. \(2013\)](#) e, posteriormente, [Xiang et al. \(2014\)](#) que consistem na aplicação do algoritmo não supervisionado de *clustering*, FCM, de maneira a segmentar as imagens SAR de acordo com a distância com os seus vizinhos.

2.2.2 Desenvolvimento de Soluções de IA

O desenvolvimento de produtos de software e de soluções de inteligência artificial é, logicamente, muito diferente. Segundo [Lambert \(2020\)](#), o desenvolvimento de software baseia-se na procura ou desenvolvimento do algoritmo mais eficiente, a linguagem mais adequada e estrutura de dados certa para tratar do fluxo de dados, que é cerne da aplicação.

Por outro lado, no desenvolvimento de uma solução inteligente, o objetivo é desenvolver um sistema que exiba capacidade de adaptar o seu comportamento tendo em conta os dados que recebe e processa, em produção serão, provavelmente, nunca vistos.

Assim, o desenvolvimento é baseado no melhor tratamento dos dados, os melhores modelos e algoritmos para os entender. Mesmo assim o fator *deployment* e eficiência está sempre ciente, tal como ocorre no desenvolvimento de software. Por este motivo, iniciar um projeto de ML na componente de codificação de modelos ou visualização dos dados sem definir um processo resulta por norma num projeto mais demorado, sem estrutura e em último caso, falhado.

Uma vez que a definição de uma metodologia de desenvolvimento é de extrema importância, durante a presente secção vai ser feita uma análise de vários métodos de desenvolvimento. Iniciando pelos processos que não são, ou não devem ser utilizados na construções de soluções *Machine Learning*, *Waterfall* e *Agile*. Posteriormente, serão apresentadas as metodologias mais usadas em projetos ML, metodologia SEMMA e a metodologia CRISP-DM.

WATERFALL PROCESS O desenvolvimento em cascata, em português, é processo muito comum no desenvolvimento de software que relata um desenvolvimento sequencial. Fazendo comparação com a construção de uma ponte, inicialmente prepara-se o solo, posteriormente coloca-se as fundações e só depois se inicia a construção da ponte em si. A razão de este modelo não ser utilizado é bastante simples, deve-se ao facto de ser impossível realizar o desenvolvimento de um modelo de forma sequencial. Nunca o *dataset* está completamente concluído antes da fase de treino do modelo, é necessário ir atualizando os dados e por consequência atualizando os modelos. ([Shahandeh, 2018](#))

AGILE O método agile é a prática mais comum no desenvolvimento de software. Promove a iteração contínua de desenvolvimento e teste durante a o período de desenvolvimento. A diferença entre Agile e Waterfall é que os processo de desenvolvimento e teste ocorre em simultâneo. Agile foca-se na interação entre pessoas, processo e ferramentas, entrega de software funcional, e constante colaboração com o cliente de maneira a responder rapidamente a alterações ao plano. Não há dúvidas que o aparecimento desta metodologia revolucionou a maneira como as empresas desenvolvem e trabalham as aplicações. No entanto, as características únicas dos projetos de AI dificultam a implementação de Agile. Sendo um modelo de *machine learning* consiste num código algorítmico com dados, parâmetros, hiper-parâmetros. Assim é provável que um mesmo algoritmo com dados diferentes dê um modelo diferente, e que um outro algoritmo com os mesmos dados também dê um modelo diferente. E daí surgem questões como, que algoritmo entregar, que dados entregar, entregar ambos,

não entregar nenhum. Por este motivo é necessário novas abordagens de maneira a aumentar a usabilidade de Agile no desenvolvimento de ML.

SEMMA SEMMA é a sigla de "Sample, Explore, Modify, Model, and Asses"(Amostra, exploração, preparação, treino, Avaliação), e é uma metodologia muito utilizada que foi desenvolvida por *SAS Institute*. Num artigo, [Dadouche \(2018\)](#) descreve o processo da seguinte forma.

- **Sample:** O processo inicia-se com o o particionamento dos dados. O *dataset* deve ser grande o suficiente para conter informação suficiente mas não muito grande para não comprometer a eficiência.
- **Explore:** Esta é a fase de análise e compreensão de dados, normalmente executada com um conjunto de visualizações.
- **Modify:** Também é conhecido como preparação de dados.
- **Model:** Nesta fase, o foco está na aplicação de vários algoritmos, a fim de criar modelos que possivelmente forneçam o resultado desejado.
- **Assess:** A avaliação dos resultados dos modelos que mostram a fiabilidade e utilidade dos modelos criados.

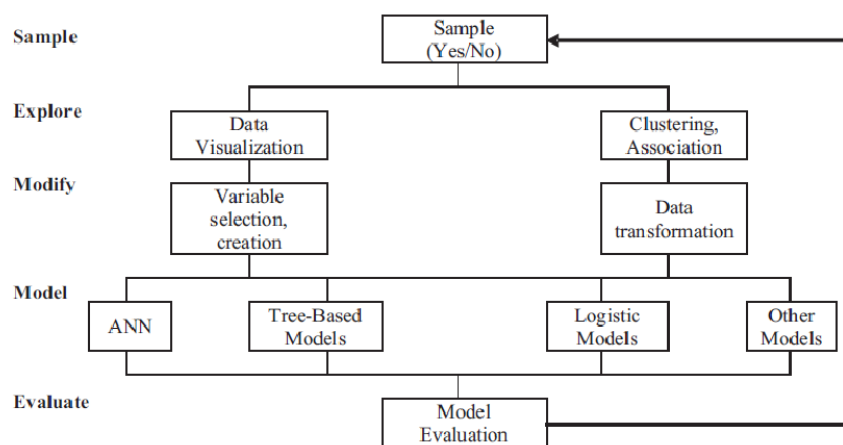


Figura 19: Metodologia SEMMA. créditos: researchgate.net

O facto de ser um método iterativo apresenta-se com uma forte vantagem face a outras formas de desenvolvimento. Porém, o facto de não abordar todo espectro de um projeto de negócio, como o problema, leva muitas empresas e equipas a não usar este processo.

CRISP-DM Esta é a técnica mais usada em projetos de ML. O acrónimo quer dizer "CRoss-Industry Standard Process for Data Mining", e foi introduzida em 1996. As fases do CRISP-DM são as seguintes, descritas no artigo escrito por [Saltz and Hotz \(2018\)](#).

- **Business understanding:** Esta fase inicial centra-se na compreensão dos objectivos e requisitos do projecto a partir de uma perspectiva empresarial;
- **Data understanding:** A fase de compreensão dos dados começa com uma recolha inicial de dados e prossegue com as actividades de modo a familiarizar-se com os dados, identificar problemas de qualidade dos dados, descobrir os primeiros conhecimentos sobre os dados;
- **Data preparation:** A fase de preparação dos dados abrange todas as actividades para construir o conjunto de dados final (dados a serem enviados para os modelos) a partir dos dados iniciais;
- **Modeling:** O foco, nesta fase, está na aplicação de vários algoritmos, a fim de criar modelos que forneçam o resultado ambicionado;
- **Evaluation:** Nesta fase já existe um candidato a modelo final. Antes de proceder a deployment do modelo, é importante avaliar de forma minuciosa se o modelo atinge correctamente os objectivos empresariais;
- **Deployment:** Dependendo dos requisitos e funcionalidade do sistema, pode ser apenas um relatório técnico do modelo criado ou uma solução em *cloud*. Normalmente, o *deploy* é responsabilidade do cliente.

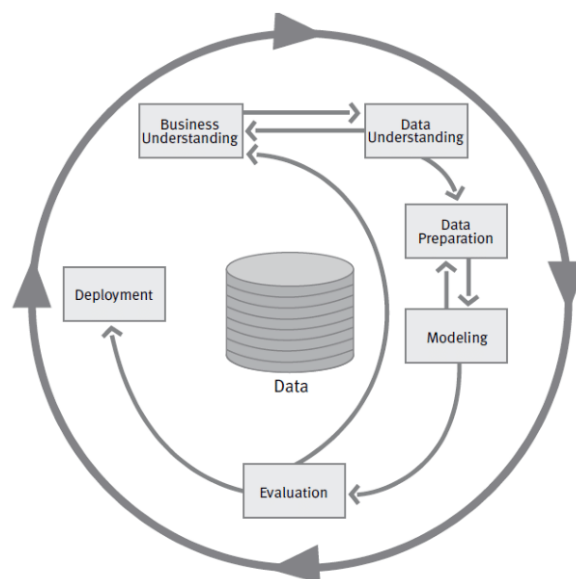


Figura 20: Metodologia CRISP-DM. créditos: researchgate.net

A utilização deste método apresenta vantagens com o facto de ser um processo iterativo, um processo completo que engloba todas as vertentes de um projeto desde estudo do problema até ao deploy da solução. Os contras da utilização desta metodologia são a forte componente de documentação, o que torna o processo burocrático, e a não consideração pelos avanços tecnológicos como o fenómeno *Big Data*.

Ferramentas

É extremamente difícil definir a melhor linguagem para ML, uma vez que a escolha de uma linguagem, como em qualquer problema, depende do objetivo da peça de software, do *background* do utilizador e até das normas da empresa. O artigo escrito por Voskoglou (2017) faz referência a um inquérito feito a mais de 2000 cientistas de dados com a intenção de perceber qual a linguagem que utilizam nos seus projetos, bem como a linguagem que usaram quando estavam a iniciar o processo de aprendizagem.

Em termos de popularidade a linguagem Python lidera o pacote, com 57% dos cientistas de dados e programadores de *Machine Learning*. C/C++ é o segundo em relação ao Python com (44%) a terceira mais utilizada é R com 7%.

Este estudo indica que Python já se tornou parte integrante de ML - evoluiu assim para a linguagem preferencial dos cientistas de dados. O mesmo não se pode dizer em relação ao R, que é na sua maioria considerado pelos analistas de dados e estatísticos, uma vez que foi inicialmente criada para eles.

A escolha de Python ao longo do tempo deve-se sobretudo à sua simplicidade (i.e., é uma linguagem *open-source*, de alto nível, que pode ser utilizada como *scripting*) que torna a linguagem rápida e simples. Um outro fator com muito peso na escolha do Python é a quantidade de bibliotecas disponíveis para o desenvolvimento na linguagem, por exemplo *NumPy* e *Pandas*. Por fim, a sua enorme comunidade permite aos seus utilizadores aceder a tutoriais e procurar ajuda em fóruns.

Em 2012 nasceu uma linguagem que tem potencial de se aproximar de Python na área de ML. O seu nome é Julia e foi desenvolvida para oferecer performance (que é uma forte desvantagem), fácil utilização e ao mesmo tempo ser uma linguagem dinamicamente tipada que permite grande interatividade. Porém, uma vez que é uma linguagem jovem, ainda não possui comunidade suficiente para ganhar autoridade face aos concorrentes. Por esse motivo, o Python será a linguagem utilizada neste projeto. De seguida, serão abordadas as ferramentas ML mais utilizadas no mercado.

TENSORFLOW Publicado pela Google (novembro, 2015) no seu blog, "Hoje orgulhamo-nos de anunciar o lançamento de código aberto de TensorFlow, ferramenta de desenvolvimento de ML. TensorFlow é genérico, flexível, portátil, fácil de usar, e *open source*." Esta ferramenta é um ecossistema completo para resolver problemas desafiadores do mundo real por meio de ML, como criação fácil de modelos, produção robusta de ML em qualquer lugar. A aplicação permite que equipas e utilizadores criem as suas próprias *smart apps* e obtenham resultados inteligentes baseados em dados que fazem sentido.

Tensorflow é uma ferramentas mais utilizadas pela comunidade de *machine learning*, e segundo artigos dos sites [data flair \(2018\)](#) e [comparecamp \(2019\)](#) tal deve-se ao facto de ser *open-source*, continuo suporte, ser de uma *big tech*, possível implementação mobile, e a possível exploração do paralelismo. Os mesmos autores indicam que ser mais lento que outras ferramentas, apenas suportar *GPUs* da NVIDIA são algumas das razões que afastam os utilizadores da utilização do tensorflow.

Não tem suporte nativo para Windows; Mais lento em termos de performance comparação torch; Atualmente, as únicas *GPUs* suportadas são as da NVIDIA e o único suporte linguístico completo é o do Python

PYTORCH Publicada a primeira versão em setembro de 2016, pela “Facebook’s AI Research lab” Muito semelhante ao TensorFlow, tem o objetivo principal de tornando os modelos mais fáceis de escrever. PyTorch é baseado em Torch, uma estrutura para fazer cálculos rápidos que é escrita em C. Torch tem um invólucro Lua para a construção de modelos.

As suas principais características são: 1) Transição rápida e eficaz entre os modos desenvolvimento e produção, 2) Um ecossistema rico em ferramentas e bibliotecas e 3) apoio ao desenvolvimento nas áreas visão por computador, processamento de linguagem natural.

PyTorch oferece o mesmo *backend* de Torch numa interface Python. Os programadores construíram-no desde o início para tornar os modelos fáceis de escrever para os programadores Python. O código subjacente, de baixo nível C e C++ é otimizado para executar código Python. Oferece também uma excelente otimização e utilização de memória, o comportamento do modelo é mais transparente e possui uma boa compatibilidade com a biblioteca *NumPy*.

Com PyTorch é possível correr uma rede neural enquanto a constrói, linha a linha, o que facilita o debug.

PyTorch necessita de *third-party software* para realizar a visualização, isto é uma clara desvantagem face ao seu concorrente direto (TensorFlow), segundo Kurama (2020).

APACHE SPARK Esta ferramenta foi desenvolvida no AMPLab da Universidade da Califórnia, em 2010. Apache Spark é a combinação de diversas bibliotecas que interagem de maneira a permitir ao utilizar uma utilização mais completa.

Como pode ser visto em Spark (2020), os diferentes componentes que compõem a ferramenta são os seguintes.

O Spark Core é o motor de execução geral subjacente à plataforma Spark, sobre o qual todas as outras funcionalidades são construídas. Fornece capacidades de computação *in-memory* para fornecer velocidade, um modelo de execução generalizada para suportar uma grande variedade de aplicações, e APIs Java, Scala, e Python para facilidade de desenvolvimento.

O componente Spark Streaming permite poderosas aplicações interativas e analíticas tanto em *streaming* como em dados históricos, ao mesmo tempo que herda a facilidade de utilização e as características de tolerância a falhas de Spark. Integra-se prontamente com uma grande variedade de fontes de dados populares, incluindo HDFS, Flume, Kafka e Twitter.

Para Machine Learning spark possui a biblioteca MLlib. Uma biblioteca de ML escalável que fornece tanto algoritmos de alta qualidade e com alta performance. A biblioteca esta disponível em Java, Scala, e Python.

GraphX é um motor de computação gráfica desta ferramenta que permite aos utilizadores construir, transformar e analisar interactivamente os dados estruturados em gráficos.

As principais vantagens na utilização de Apache Spark são a sua rapidez, facilidade de utilização, suporte de várias linguagens, *Open-source* e preparado para *Big Data*.

Porém, possui alguns inconvenientes para os desenvolvedores que são a sua quantidade reduzida de algoritmos, não tem processos de otimização automática e possui um fraco gestor de ficheiros.

CAFFE Desenvolvido por [Jia et al. \(2014\)](#), Berkeley AI Research (BAIR), que se caracteriza pela criação e otimização sem codificação rígida, a fácil transição entre *Central Processing Unit (CPU)* e *GPU*. Os seus criadores referem que a sua velocidade torna esta ferramenta perfeita para investigação. Caffe tem também forte presença na comunidade.

Segundo o autor, esta ferramenta é particularmente interessante em problema de processamento de imagem, e outra vantagem que possui é a sua interface Python e *Application Programming Interface (API)* que são bastantes úteis.

Um dos problemas na utilização de Caffe é a necessidade de escrever C++ / CUDA para novas camadas de *GPU*, não é boa para problemas onde se usam RNNs ou GANs, o uso de grandes redes (GoogLeNet, ResNet) é bastante complicado e não possui nenhum apoio comercial e por essa razão está "cair em desuso". ([Yin, 2017](#))

2.2.3 Computer Vision

De um modo geral, Visão por computador é um ramo da inteligência artificial que se foca em ajudar os computadores a ver. De um modo abstrato, o objetivo dos problemas de visão por computador é usar os dados da imagem observada para inferir alguma coisa acerca do mundo. ([Prince, 2012](#)).

A visão por computador abrange tarefas que incluam "ver" ou sentir um estímulo visual, compreender o que está a ser visto, e extrair informação complexa para uma forma que possa ser utilizada noutros processos. Este ramo simula e automatiza estes elementos dos sistemas de visão humana através da utilização sensores, computadores, e algoritmos de ML.

Por ser uma área tão vasta, a sua aplicabilidade estende-se desde veículos autónomos à agricultura, passando pela robótica, deteção de objetos e até retalho.

Com os avanços na área de Deep Learning, as fronteiras de aplicação de *Computer Vision* estão constantemente a ser desafiadas e ultrapassadas.

Na secção seguinte encontra-se desenvolvido um algoritmo de comparação de imagem que tem apresentados resultados impressionantes na comparação de imagens.

ASIFT

Affine Scale-Invariant leature Transform (ASIFT) é um método de comparação de imagem, proposto por [Morel and Yu \(2009\)](#). Trabalho inspirado no algoritmo SIFT proposto em [Lowe \(1999\)](#) em que o objetivo inicial do método é comparar duas imagens que podem ser deduzidas uma da outra através de rotação, translação, e escala.

O algoritmo ASIFT trata dos dois parâmetros que estão em falta no algoritmo base, que são os ângulos que definem a orientação da câmara.

Assim, ASIFT simula três parâmetros: 1) a escala, 2) longitude do ângulo da câmara e 3) latitude do ângulo da câmara (equivalente a *tilt*) e obtém os restantes (translação e rotação) através de normalização.

Apesar de inicialmente se pensar o contrário simular todas as vistas dependentes que dependem destes 2 padrões é possível. O método permite identificar com *accuracy* características que tenham sofrido distorções

muito grandes. Os métodos mais avançados dificilmente excedem as inclinações de transição de 2 (SIFT). ASIFT pode lidar com as inclinações de transição até 36 e superiores.

O algoritmo procede segundo os seguintes passos.

1. Cada imagem é transformada através da simulação de todas as possíveis distorções causadas pela alteração dos eixos de orientação da câmara. As imagens requerem uma correção inicial através da aplicação de um filtro *antialiasing* (Gaussian).
2. Estas rotações e inclinações são realizadas para um número finito de ângulos de latitude e longitude. As imagens simuladas mantêm-se próximas de qualquer outra vista possível gerada por outros valores de latitude e longitude.
3. Todas as imagens geradas são comparadas utilizando o algoritmo de *matching* SIFT.

A primeira operação do algoritmo SIFT é a extração dos *keypoints* de cada imagem, um processo incremental.

Numa fase inicial, a imagem é convolucionada com filtros de Gauss a diferentes escalas, e posteriormente é calculada a diferença entre Gaussians (DoG) entre as imagens alteradas. Depois de calculados das diferenças, os pixels são comparados com os vizinhos da mesma escala e com escalas vizinhas. Se o valor do pixel for um máximo ou um mínimo, então é selecionado como sendo um *keypoint* candidato.

A fase anterior produz um número grande de candidatos, e muitos deles instáveis, a próximo passo é realizar uma interpolação da DoG usando uma expansão quadrática de Taylor de maneira a obter uma posição precisa do candidato diminuindo assim a presença de pontos muito próximos.

Através do valor da expansão de Taylor de segunda ordem é possível perceber e eliminar os pontos com menor contraste de acordo com um determinado *threshold*. É necessário recorrer também à matriz Hessiana de segunda ordem de maneira a identificar os pontos com resposta às formas da imagem, mas que não são robustos na presença de ruído.

Estas operações permitem rejeitar alguns candidatos a *keypoints*.

Uma vez eliminados os candidatos mais 'fracos', é altura de atribuir uma ou mais orientações a cada *keypoint*. Este passo é realizado com base na orientação do gradiente da imagem.

$$magnitude = \sqrt{(G_x)^2 + (G_y)^2} \quad (22)$$

$$\phi = \arctan\left(\frac{G_x}{G_y}\right) \quad (23)$$

No final destas operações, os *keypoints* possuem uma localização numa escala e uma orientação, o que assegura a invariância destes face a rotação, escala e posição na imagem.

Por fim, é necessário calcular o vetor descritor de cada *keypoint*, o chamado *keypoint descriptor*. Para criar esse identificador de *Keypoint* é inicialmente considerada uma vizinhança de 16x16 em volta do ponto. Por sua vez, esse bloco de 16x16 é dividido em sub-blocos de 4x4. Para cada sub-bloco são calculadas as orientações e magnitudes gerando um histograma com 8 *bins* que representam 8 direções. Uma vez que existem 4x4 histogramas e cada um possui 8 bins, o vetor possui 128 elementos ($4 * 4 * 8 = 128$).

No final, existe um vetor de Kp composto por N elementos em que cada elemento é um *keypoint descriptor* dos pontos que foram computados como *Keypoints* pelo algoritmo ASIFT. Estes pontos são calculados individualmente para cada imagem e não existe qualquer relação entre as mesmas nesta fase.

Por fim, é feita a associação entre as imagens, através de um algoritmo de *Match*, que é responsável pela correspondência mais provável entre os *keypoints* extraídos anteriormente pelo algoritmo ASIFT.

Um dos *matchers* mais utilizados é o BFmatcher que significa Brute Force Matcher (*match* por Força Bruta, em português), em que cada descritor de *Keypoint* do conjunto 1 é comparado com todos os descritores do conjunto 2 de maneira a encontrar o mais aproximado.

Por outro lado, é também possível a utilização de um *matcher* baseado em *clustering*. O método Flann realiza um treino sobre o conjunto 1 otimizando assim procura pelo melhor *match*. Tornando-se mais rápido que o método **BF** em *datasets* maiores.

De maneira a comprovar a utilidade e eficácia do seu algoritmo os autores realizaram uma série de comparação com diferentes objetos e de perspectivas diferentes.

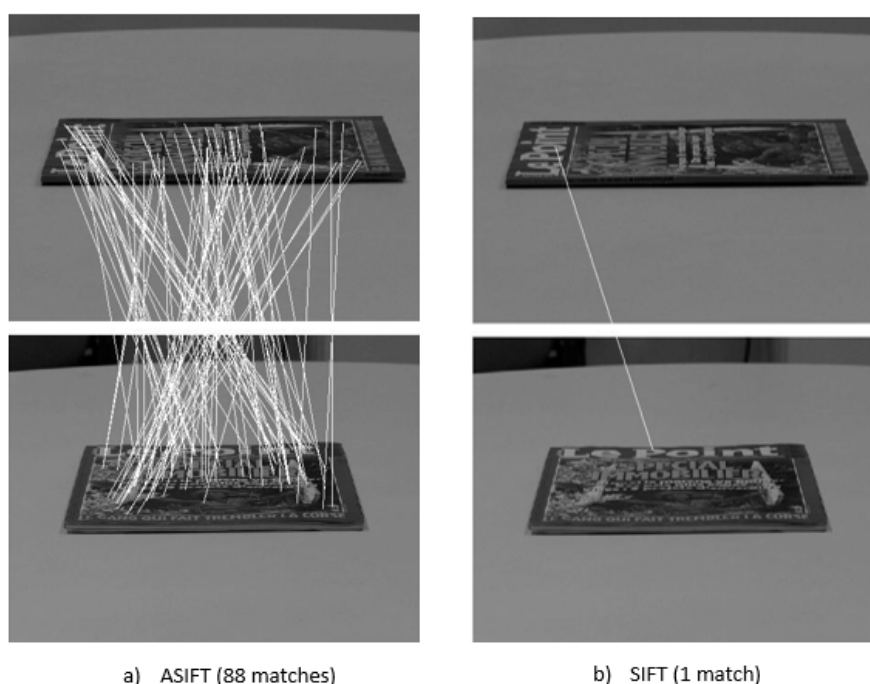


Figura 21: Comparação entre o algoritmo ASIFT e SIFT. Imagens com diferença de 90° de ângulo (Crédito: (Morel and Yu, 2009))

A Figura 21 mostra a comparação de algoritmos ASIFT e SIFT na comparação de uma revista, e é claramente visível a superioridade do algoritmo ASIFT. Estes resultados mostram o potencial que o algoritmo tem em problemas onde a comparação de imagens é fulcral.

Parte II

METODOLOGIA

REGISTO GEOGRÁFICO DE IMAGENS SAR

Um dos problemas identificados foi a necessidade de registo e anotação manual das imagens SAR capturadas, que resulta num processo custoso e demorado.

Assim, é extremamente útil encontrar um mecanismo que permita uma diminuição significativa da intervenção humana no registo. Com isto, será possível registar mais imagens em menos tempo que se traduz num processo mais eficiente.

Ao longo deste capítulo serão apresentadas algumas abordagens para a resolução deste problema. Iniciando pela especificação do funcionamento, uma secção referente aos testes da solução, e por fim, sugestões e propostas de soluções com vista ao melhoramento da solução.

3.1 REGISTO BASEADO EM ASIFT

A primeira abordagem idealizada trata-se de um comparador Ótico-SAR que, de forma sucinta, consiste numa aplicação que através de uma imagem satélite geograficamente registada, seria capaz de encontrar semelhanças com a imagem SAR com recurso ao algoritmo ASIFT, descrito em Secção 2.2.3, e através destas semelhanças geo-localizar a imagem SAR.

O objetivo desta abordagem é a remoção praticamente total do trabalho humano no processo de registo, já que todo o trabalho do registo passa a ser realizado pela aplicação, desde a obtenção da imagem ótica para a comparação, até à criação do ficheiro GeoTiff com as coordenadas geográficas da imagem.

3.1.1 Procedimento

Como referido, o objetivo é a redução do trabalho humano no registo, por isso o sistema apenas recebe como *input* a imagem a registar e as coordenadas onde a captura foi realizada e fornece como *output* a imagem SAR devidamente geo-localizada.

Todo o processo de registo é realizado pelo sistema e segue as seguintes etapas:

1. **Introdução dos Inputs:** A primeira fase é o envio da imagem SAR a ser registada e as coordenadas do radar aquando da captura;

2. **Obtenção da imagem ótica:** A imagem ótica, devidamente registada, que será usada para comparação é carregada para o sistema;
3. **Tratamento da imagem SAR:** Fase de pré-processamento da imagem SAR;
4. **Comparação de imagens:** Comparação entre a imagem ótica e SAR com recurso ao algoritmo ASIFT e com o objetivo de extrair a matriz de Homografia;
5. **Registo da "nova" imagem:** Com base na matriz de homografia é calculada a posição das imagens SAR;
6. **Criação dos GeoTiff:** Por fim, é criado e guardado um ficheiro GeoTiff com a imagem SAR e as respetivas coordenadas geográficas.

Uma forma mais fácil de perceber o fluxo da informação é através da visualização de um diagrama do processo, este pode ser visto na Figura 22.

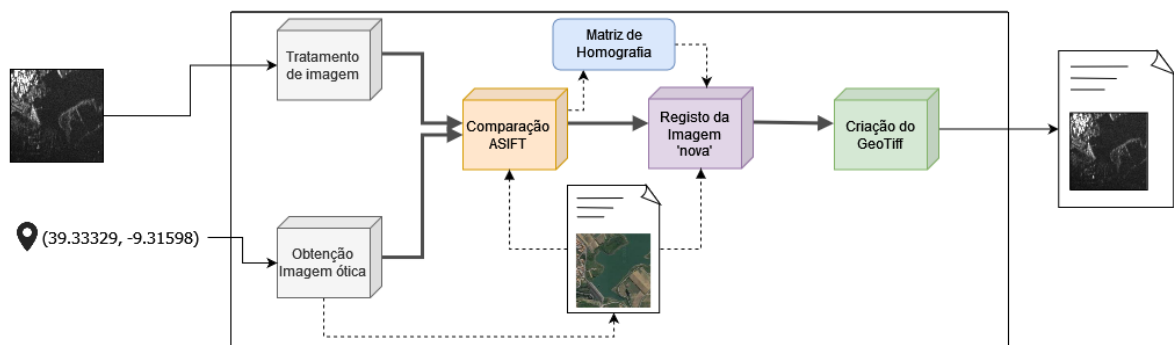


Figura 22: Diagrama do procedimento do registo com ASIFT.

De forma mais completa, o processo inicia-se com a receção na aplicação da imagem SAR e das coordenadas do radar aquando da obtenção da imagem. Os dados seguem caminhos diferentes.

As coordenadas geográficas alimentam a parte responsável pela obtenção da imagem ótica de satélite da zona em redor do radar, esta imagem é obtida através da API pública MapQuest, que nos permite através de pedidos HTTP obter imagem de um respetivo local. O pedido permite também a definição de opções como tipo de imagem (satélite, estradas, entre outras), o zoom, e a resolução da mesma.

Por uma questão de simplificação a tipologia, o Zoom e a resolução da imagem foram otimizados manualmente, e por isso serão predefinidos.

Tabela 3: Opções API

Opção	Valor
Tipo de Imagem	satélite
Zoom	17
Resolução	1920x1920

Como é possível ver na tabela Tabela 3, a imagem será do tipo **satélite**, uma vez que esta ser estudada a comparação entre uma imagem SAR e uma imagem ótica.

O Zoom foi fixado no valor **17** com base na melhor correspondência entre as distâncias de obtenção das imagens SAR e ótica.

A resolução das imagens utilizada é de **1920x1920**, dado que é o máximo permitido pela API.

Uma vez definidas estas opções, o único cálculo necessário é para encontrar o centro da imagem ótica. Para a sua obtenção é necessário ter em conta que o radar captura as imagens lateralmente, ou seja, o sensor do radar não está a enviar e receber sinais verticalmente, mas sim com um ângulo para a lateral (*Look Angle*), sendo conhecido esse ângulo e altitude da plataforma é possível calcular a área irradiada através dos seguintes passos.

- **Passo 1** - calcular o vetor deslocamento do SAR;

$$x = x_t - x_{t-1}, y = y_t - y_{t-1}$$

- **Passo 2** - calcular a distancia do SAR ao ponto que está a iluminar;

$$d = height * \tan(\alpha)$$

- **Passo 3** - calcular o ângulo que o vetor deslocamento faz com uma paralela ao equador;

$$\cos(\beta) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}$$

- **Passo 4** - Calcular os deslocamento em latitude e longitude;

$$\Delta y = \sin(\beta) * d, \Delta x = \cos(\beta) * d$$

- **Passo 5** - calcular as coordenadas do centro do *patch*;

$$x_{centro} = x_{sar} + \Delta x, y_{centro} = y_{sar} + \Delta y$$

Sabendo que o fator distância é relevante em imagens SAR (i.e., perde pormenor com a distância), não é recomendada a seleção do centro da imagem SAR como centro da imagem ótica.

Dessa forma, a imagem ótica é obtida com centro próximo da $\frac{1}{4}$ da imagem SAR de maneira a estar situada na parte útil da imagem SAR, a primeira metade.

Com estes critérios, é obtida a imagem ótica da zona em questão, e de seguida é armazenada com o objetivo de ser utilizada para a identificação de outra imagem SAR quando for detetado que o radar está próximo dessa zona.

Dessa forma, é poupado tempo de acesso à API e por isso apresenta vantagens em termos de tempo. Para que uma imagem seja considerada na zona próxima de outra é preciso que a distância entre os centros seja inferior a um determinado *threshold* ($\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} < d$).

Assim, sempre que o cálculo de um centro seja efetuado, é verificada a existência de imagens guardadas e que correspondam a zona próxima.

No caso de existir, é utilizada essa mesma imagem e poupado um acesso à API.

No caso negativo, é realizada a obtenção da imagem ótica com base no centro calculado.

É importante referir que as fórmulas de distância apenas podem ser utilizadas em ambiente de desenvolvimento, uma vez que para uma ferramenta em produção é necessário ter em conta a curvatura da terra no cálculo de distâncias. Para isso deverá ser utilizada a **fórmula de Haversine**.

Por outro lado, a imagem SAR segue para pré-processamento, uma etapa importante uma vez que a imagem SAR é, como explicado anteriormente, muito afetada por ruído. Durante esta fase a imagem passa por um filtro, resultando numa imagem mais significativamente mais ‘limpa’.

Passar uma imagem por um filtro significa a alteração dos valores dos pixels com base nos valores dos pixels que estão na sua vizinhança. Existem dois tipos de filtros, o primeiro grupo de filtros são baseados em pesos pré-definidos que fazem a ‘limpeza’ da imagem, são exemplos a média e mediana.

Por outro lado, existem filtros em que os pesos são adaptativos com base no grau de *speckle*. São exemplos de filtros dessa categoria os filtros: **Lee**, **Lee-Sigma** e **Frost**.

Neste caso foi selecionado o **filtro Lee** para preparar a imagem para a fase seguinte.

$$I_{output} = \bar{I} + W * (I - \bar{I}) \quad (24)$$

$$W = \frac{\sigma_k^2}{(\sigma_k^2 + \sigma^2)} \quad (25)$$

Uma vez preparadas as imagens, é iniciada a comparação das imagens com o **algoritmo ASIFT**, descrito anteriormente.

Tendo o algoritmo de *Match* encontrado semelhanças entre as imagens, muitas vezes é possível encontrar alguns falsos positivos. De maneira a encontrar a relação entre *keypoints* é utilizado um algoritmo de nome Random Sample Consensus (RANSAC). Este algoritmo introduzido por **Fischler and Bolles (1981)**, é um método iterativo para estimar parâmetros com capacidade de lidar com conjuntos de dados com grandes quantidades de *outliers*, gerando soluções candidatas através da utilização do número mínimo de dados possível.

A utilização deste algoritmo permite uma melhoria de resultados aquando da computação da matriz de homografia.

A matriz de Homografia é uma matriz de 3x3 que possui informação de translação, rotação e escala que permite a transformação de uma imagem na outra. É através desta matriz que a imagem será registada, sabendo as coordenadas da imagem ótica e a transformação necessária de uma imagem na outra é possível calcular as coordenadas correspondentes a num determinado conjunto de pontos, e dessa forma conseguir posicionar a imagem SAR com base na posição da imagem ótica.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (26)$$

A Eq 26 reflete o procedimento, com recurso à matriz de Homografia H , através do qual é possível obter um ponto (x', y') na imagem SAR, que é o análogo de (x, y) na imagem ótica registada.

3.1.2 Testes

Para os testes deste abordagem ao registo geográfico serão apenas utilizadas imagens da Tekever, já que são as únicas em que é possível enquadrar numa determinada área geográfica a zona de captura de maneira a obter o seu dual ótico para assim seguir o algoritmo como foi pensado.

A partir do teste inicial (Figura 23), com a comparação entre uma pista de aviação registada pelo SAR e a mesma pista em imagem ótica, os resultados apesar de não serem ótimos, indicavam possibilidade de esta abordagem trazer os resultados desejados.

No seguimento foram realizados testes menos concretos e mais centrados no que seria o objetivo do sistema.



Figura 23: Teste 0 - Comparação entre a imagem ótica e a imagem SAR do mesmo local.

No teste nº1 (Figura 24) quando se testou a imagem SAR na sua totalidade face a uma imagem ótica com o nível de Zoom mais reduzido, o sistema demonstrou incapacidade de geo-localizar a imagem SAR com aquele nível de Zoom na imagem ótica.

Iniciaram os testes com recurso a imagens com nível de Zoom superior, e com o *crop* das imagens SAR para uma área mais pequena. Mesmo assim, os resultados mantiveram-se negativos.

De seguida, e com o objetivo de tentar perceber onde é que o sistema estava a falhar, a imagem SAR foi rodada de maneira a ficar com a mesma orientação que a imagem ótica, facilitando a operação. A alteração não surtiu o efeito positivo.



Figura 24: Teste 1 - Comparação entre a imagem ótica e a imagem SAR.

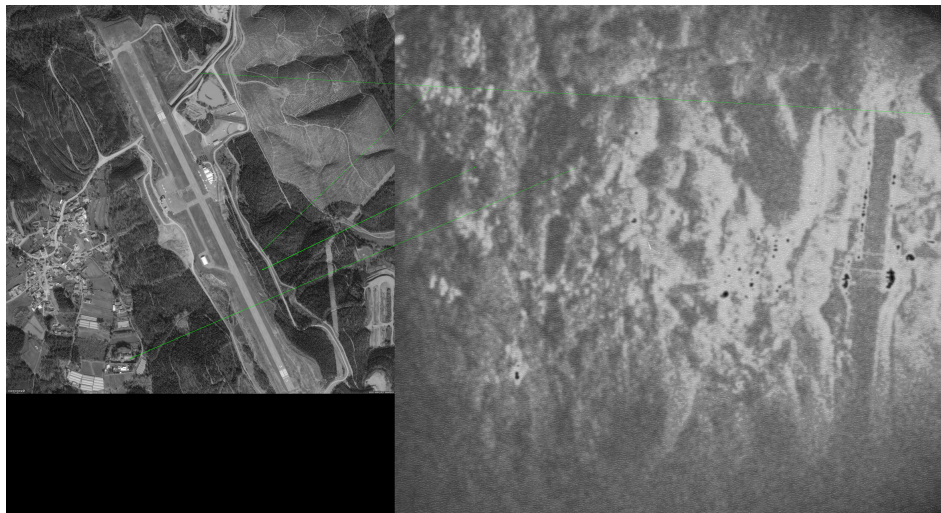


Figura 25: Teste 2 - Comparação entre a imagem ótica e a imagem SAR com rotação 180°.

Os resultados (Figura 25) mostraram que mesmo assim não foi possível enquadrar de forma correta as imagens, este resultado refuta a hipótese de que o sistema não estava a detetar pelo facto de a transformação ser muito complexa.

Uma das possíveis causas para os maus resultados pode ser atribuída à inconsistência dos descritores, originada pelo número elevado de *Keypoints*, que resulta num *Matcher* incapaz de realizar a correspondência correta entre pontos.

Como forma de tentar reduzir o número de *Keypoints* as imagens sofreram um *resize*.

Apesar de existir uma diminuição no número de *keypoints* detetados pelo ASIFT, os resultados não melhoraram (Figura 26) o que mostra que o *Matcher* não está a conseguir juntar os *keypoints* correspondentes.



Figura 26: Teste 3 - Comparação entre a imagem ótica e a imagem SAR com *resize*.

Posteriormente, foi testada a junção entre rotação da imagem ótica com a alteração do tamanho das imagens. O resultado desta junção melhorou a combinação de alguns pontos, apesar dessa ligeira melhoria os resultados continuam longe de estar positivos.

Uma vez que a junção entre o *resize* das imagens e a rotação da imagem teve efeitos positivos, os testes seguirem esse caminho e foi alterada uma outra variável, o Filtro, a imagem SAR foi filtrada com um filtro de tamanho 5.

Dessa alteração (Figura 27) surge uma ligeira melhoria no *matching* dos pontos.

No entanto, continuam a existir umas *Gray Areas* que confundem o *matcher* e por isso destabilizam todo o processo de localização.

Foram realizados alguns testes, com outras abordagens como passar filtros sobre a imagem satélite, *resize* das imagens para outros tamanhos. Porém, os seus resultados não melhoraram ou em alguns casos pioraram a prestação do sistema.

Os problemas levantados a esta abordagem, são provavelmente resultado da combinação entre **deteção de Keypoints** e o **match** dos mesmo entre as duas imagens.

De seguida é apresentada uma tabela resumo dos testes realizados.

3.1.3 Solução Porposta

De maneira a tentar resolver alguns dos problemas levantados, foram idealizada 2 caminhos diferentes:

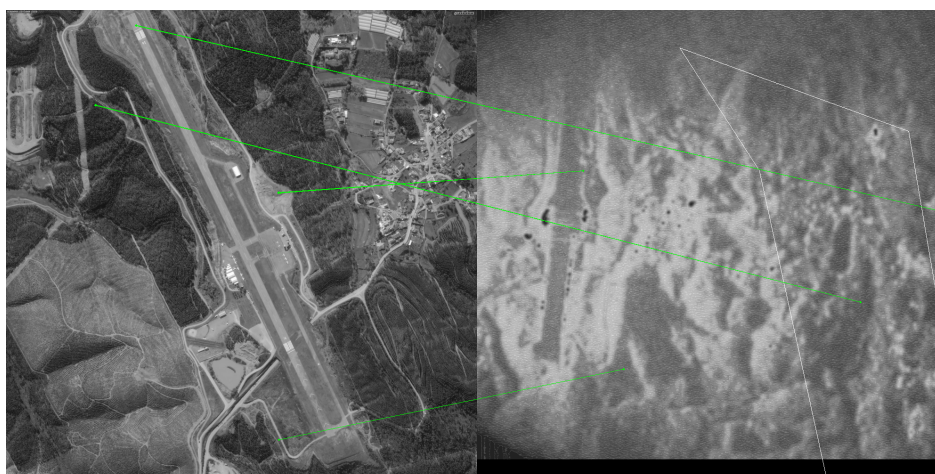


Figura 27: Teste 4 - Comparação entre a imagem ótica com rotação 180° e a imagem SAR com alteração filtro.

Tabela 4: Resumo de testes

Teste	SAR	Patch	Zoom	Filter (size)	Crop	Rotation	Resize	Resultado
0	Pista	Pista	17	Sim (3)	Sim	Não	Não	+
1	Completa	Pista	16	Sim (3)	Não	Não	Não	-
2	Completa	Pista	17	Sim (3)	Sim	Sim	Não	-
3	Completa	Pista	17	Sim (3)	Sim	Não	(700, 500)	-
4	Completa	Pista	17	Sim (5)	Sim	Sim	Não	-
5	Completa	Pista	17	Sim (5)	Sim	Sim	(700, 500)	-
6	Completa	Pista	17	Sim (5)	Sim	Não	(1920,1500)	-
7	Completa	Pista	17	Sim (3)	Sim	Não	(1200,900)	-
8	Completa	Pista	17	Sim (5)	Sim	Não	(1000,700)	-
9	Completa	Pista	17	Sim (3)	Sim	Não	(700,700)	-
10	Completa	Pista	17	Sim (7)	Sim	Sim	(700,500)	-

- Uma variante desta abordagem em que é utilizado o mesmo sistema sistema para realizar a mesma operação de uma forma, possivelmente, mais eficaz;
- Alteração da forma de comparação para duas imagens SAR.

O primeiro, diverge da solução implementada no momento de entrada no sistema de deteção dos *Keypoints*, em que ao invés de encontrar os *keypoints* da imagem SAR na sua totalidade, são utilizados pequenos blocos da imagem SAR. Estes *keypoints* são todos comparados com a imagem satélite e o bloco com maior confiança nos *matches* é utilizado como bloco principal para a localização da imagem. De forma conceptual, esta forma de abordar o problema leva à diminuição de *keypoints* por comparação e a uma possível redução de *matches* errados uma vez que só o melhor bloco é utilizado.

A segunda alternativa consiste na alteração de toda a forma de registo. Ao invés da comparação ser feita entre uma imagem SAR e uma imagem Ótica, esta passa a ser feita entre duas imagens SAR. Esta alteração na forma de comparação permite que o processo de identificação de *keypoints* seja realizado entre imagens com

característica idênticas, que tem como efeito uma maior semelhança entre os descritores e por consequência um melhor registo geográfico.

3.2 REGISTO BASEADO EM SAR-HARRIS

Uma vez que a utilização de uma imagem ótica como comparador não obteve resultados satisfatórios, a análise dos testes realizados mostra que o método de deteção de *features* do algoritmo ASIFT não é suficiente robusto quando a imagem possui ruído, como é caso da imagem SAR

A primeira tentativa de aumentar a performance do algoritmo foi a utilização do Logaritmo da imagem. Esta alteração permite que o ruído da imagem seja aditivo ao invés de multiplicativo.

Apesar de diminuir o problema do ruído multiplicativo, esta mudança não trouxe resultados com melhorias significativas.

Identificado o problema da enorme disparidade entre imagens óticas e imagens SAR, e da complexidade que envolve uma solução desse nível, o foco volta-se para uma solução que envolva imagens a comparação de imagens SAR, uma vez que possuem com características idênticas.

Então, [Dellinger et al. \(2012\)](#) propõem a utilização de uma nova forma de encontrar semelhanças entre imagens SAR, chamaram ao método **SAR-Harris**. Este algoritmo apesar de em termos conceptuais seguir o mesmo raciocínio que o método ASIFT, apresenta algumas diferenças importantes para lidar com imagens SAR.

A primeira diferença face ao método ASIFT é na forma como é realizada a extração dos *keypoints*. A utilização do **detetor Harris** definido em [Dufournaud et al. \(2000\)](#), face à utilização da DoG no ASIFT, é justificada pelo facto de a utilização de DoG levar a inúmeros falsos positivos. ([Dellinger et al., 2012](#))

No entanto, o detetor falha na consistência quando a resolução da imagem apresenta variações significativas, fazendo diminuir o número de falsos positivos e o número de corretos.

Um outro problema está relacionado com a seleção da escala, em que a função Harris tem dificuldades em atingir máximo em representações no espaço 3D, e se forem detetados poucos *keypoints* a representação da imagem não se torna tão robusta como é pretendido.

Na tentativa de resolver esse problema é utilizado, o autor adapta o a função Harris definida em [Mikolajczyk and Schmid \(2001\)](#). A cada nível do plano de escala é detetado o máximo local. E por forma a obter uma representação mais compacta, para cada ponto candidato encontrado em diferentes níveis, se este é o máximo da direção da escala.

Assim, para a extração, é criada uma representação do espaço-escala para a função Harris.

$$C_{SH}(x, y, \alpha) = K_{\sqrt{2} \cdot \sigma} \star \begin{bmatrix} (G_{x,\alpha})^2 & (G_{x,\alpha}) \cdot (G_{y,\alpha}) \\ (G_{x,\alpha}) \cdot (G_{y,\alpha}) & (G_{y,\alpha})^2 \end{bmatrix} \quad (27)$$

Em que $G_{x,\alpha}$, $G_{y,\alpha}$ representam os gradientes horizontais e verticais.

A definição dos gradientes é também diferente na forma como é calculada no algoritmo SIFT, proposto originalmente. Nesse, o gradiente é calculado em relação à magnitude e, como relatado, essa forma de calcular o gradiente leva a um número elevado de falsas orientações em áreas de elevada refletividade.

De maneira a combater esse problema, será utilizado um gradiente de rácio proposto pelos autores, e que ficou demonstrado em vários estudos como Touzi et al. (1988) e Oliver et al. (1996) ser o mais indicado no que concerne a problemas com ruído multiplicativo. O gradiente utilizado é o *Ratio of Exponentially Weighted Averages* (ROEWA), introduzido por Fjortoft et al. (1998). E representa uma melhoria face os anteriores, alcançada através do cálculo da média local com pesos exponenciais. O gradiente de rácio de magnitude e orientação é assim definido por:

$$G_{n,\alpha} = \sqrt{(G_{x,\alpha})^2 + (G_{y,\alpha})^2} \quad (28)$$

$$G_{t,\alpha} = \arctan \left(\frac{G_{y,\alpha}}{G_{x,\alpha}} \right) \quad (29)$$

Com $G_{x,\alpha}$ e $G_{y,\alpha}$ a serem calculados da seguinte forma.

$$G_{x,\alpha} = \log (R_{1,\alpha}) \quad (30)$$

$$G_{y,\alpha} = \log (R_{3,\alpha}) \quad (31)$$

$$R_{i,\alpha} = \frac{M_{1,\alpha}(i)}{M_{2,\alpha}(i)} \quad (32)$$

Em que para um ponto (a,b):

$$M_{1,\alpha}(i) = \int_{x=R} \int_{y=R^+} I(a+x, b+y) \times e^{-\frac{|x|+\alpha|y|}{\alpha}} \quad (33)$$

$$M_{2,\alpha}(i) = \int_{x=R} \int_{y=R^-} I(a+x, b+y) \times e^{-\frac{|x|+\alpha|y|}{\alpha}} \quad (34)$$

I representa a **Intensidade** num ponto

Depois de obtidos os descritores dos *keypoints*, o *matching* é entre imagens é passível de ser feito da mesma forma que na abordagem ASIFT, ou seja, por força bruta ou através de um algoritmo de *clustering*.

3.2.1 Procedimento

Todo o Algoritmo foi repensado de maneira a diminuir o nível de complexidade da solução.

A nova forma de localizar uma imagem SAR tem por base uma imagem SAR de referência, previamente registada pelo utilizador. O novo processo de registo, para cada conjunto de imagens que tenham partes sobrepostas, segue as seguintes etapas:

1. **Registo da Imagem Referência:** Em primeiro, é registada manualmente a imagem SAR que servirá de referência para um conjunto de imagens SAR;
2. **Load das Imagens SAR:** Depois de registada a imagem de referência, são carregadas para o programa as imagens SAR a serem localizadas, com base na previamente registada;

3. **Comparação entre imagens:** O algoritmo SAR-Harris é utilizado de forma a realizar a comparação entre as diferentes imagens, de onde é extraída a matriz de homografia;
4. **Registo das "novas" imagens:** Com base na matriz de homografia é calculada a posição das imagens SAR;
5. **Criação dos GeoTiff:** Por fim, é criado e guardado um ficheiro GeoTiff com a imagem SAR e as respetivas coordenadas geográficas.

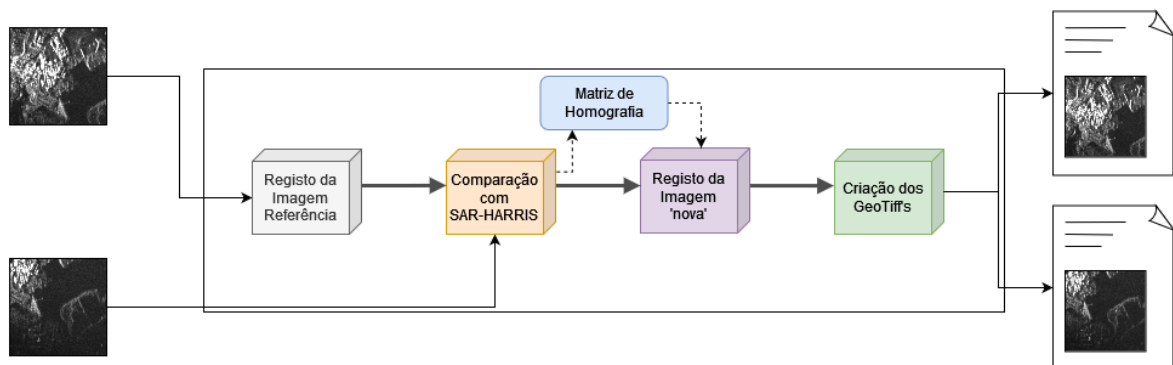


Figura 28: Processo de registo de imagem através do algoritmo SAR-Harris.

As diferenças no algoritmo podem ser vista com maior facilidade na Figura 28, que mostra o processo de registo com recurso ao SAR-Harris.

Apesar da diminuição da complexidade de o algoritmo representar mais trabalho para o utilizador da aplicação devido ao facto de necessitar de localizar manualmente uma imagem para servir de referência para registar as imagens da vizinhança.

A comparação entre imagens SAR antevê melhores resultados que a abordagem totalmente automática, e com recurso a imagens óticas, que foi apresentada anteriormente.

3.2.2 Testes

A alteração do método de registo e do algoritmo por detrás da comparação de imagens prevê um aumento significativo de performance da componente de geo-localização.

De maneira a perceber se esta melhoria conceptual no procedimento se traduz efetivamente num avanço do ponto de vista prático, o algoritmo é testado face a diversas imagens, com diferentes características.

Estes testes servem não só para confirmar a qualidade de comparação entre imagens mas também a robustez do algoritmo face a diferentes imagens SAR.

Assim sendo, o SAR-Harris será posto à prova com 3 imagens de categorias diferentes:

- Imagens do *dataset* SEN12MS, do satélite **Sentinel**;
- Imagens pertencentes ao *dataset* Ger Imagery;

- imagens fornecidas pela Tekever.

Um ponto importante é que tanto nos testes apresentados apenas será exibido o momento de comparação entre as imagens para o cálculo da **matriz de homografia**.

As restantes fases do processo são analíticas e por isso não apresentam uma representação gráfica.

O primeiro teste a ser exibido é o referente a uma imagem do SEN12MS.

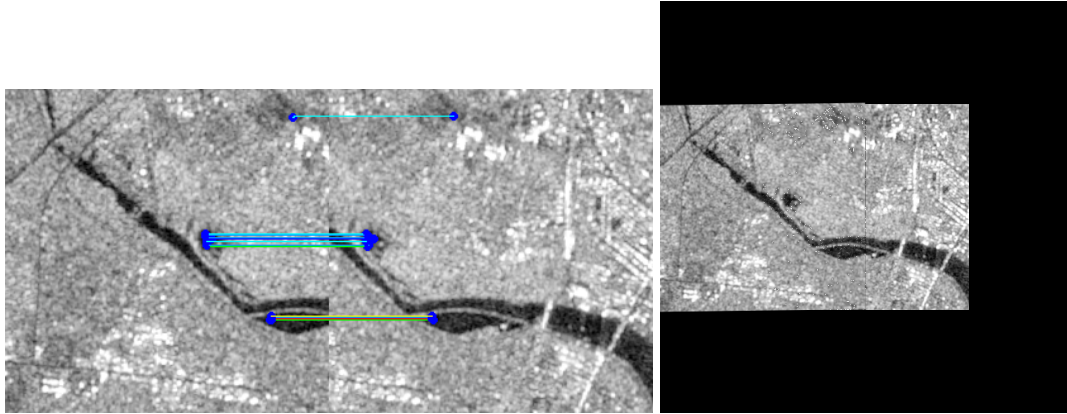


Figura 29: Teste 1 - Comparação entre imagens SAR do *dataset* SEN12MS.

Na imagem Figura 29 são exibidas 2 imagens que fazem parte do algoritmo, a da esquerda diz respeito ao resultado do algoritmo, ao seja aos *keypoints* que são assinalados como semelhança entre as imagens. Como é possível verificar, o *matching* dos *keypoints* apresenta uma qualidade elevadíssima apenas permitida pela alteração do formato para 2 imagens com a mesma tipologia e o algoritmo adaptado para imagens com as características das imagens SAR. Na imagem da direita, é feita a uma simulação da junção das imagens através da matriz de homografia, é também visível a capacidade do algoritmo de encontrar a transformação que melhor adequa a "nova" à imagem referência. Todo este processo teve a duração de **32 segundos**.

A matriz de Homografia neste exemplo tem a seguinte forma:

$$H = \begin{bmatrix} 1.001 & 0.003 & -129.182 \\ -0.012 & 1.002 & 1.652 \\ 0 & 0 & 1 \end{bmatrix}$$

De seguida, foi escolhida uma imagem do mesmo *dataset* em que a transformação é mais complexa (i.e., a imagem a referência apresenta um ângulo face à imagem referência).

Os resultados apresentados na Figura 30 mostram que o algoritmo realiza um descrição de *keypoints* robusta ao ponto de permitir a existência de um ângulo entre imagens comparadas. O tempo de execução do algoritmo é sensivelmente **inferior a 35 segundos**.

No teste seguinte, o algoritmo recebe 2 imagens do *dataset* Ger Imagery, que apresenta características distintas do SEN12MS.

As Figura 31 e Figura 32, demonstrar que o algoritmo é bastante robusto a alterações no formato da imagem, conseguindo mesmo assim encontrar as semelhanças mais relevantes para a tarefa de comparação e por con-

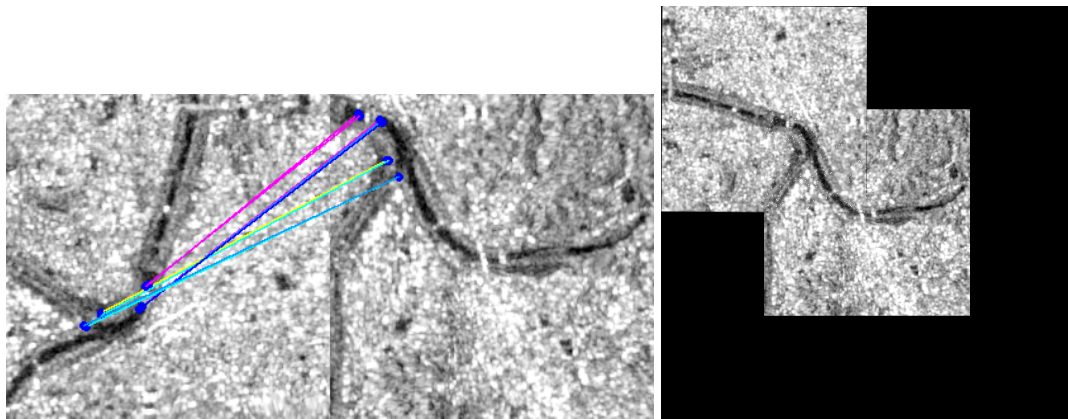


Figura 30: Teste 2 - Comparação entre imagens SAR do *dataset* SEN12MS, com ângulo.

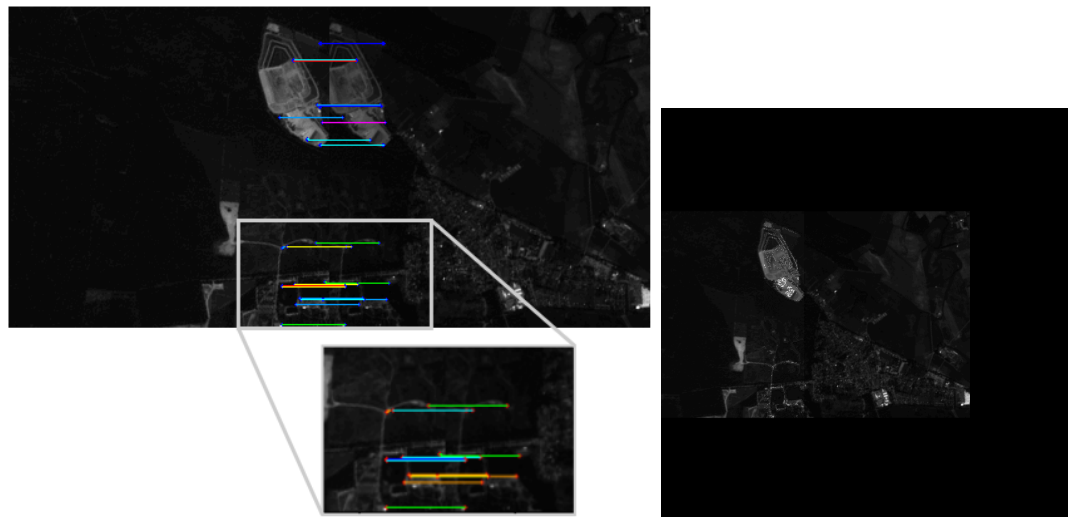


Figura 31: Teste 3 - Comparação entre imagens SAR do *dataset* Ger Imagery (1).

sequência para uma correta geo-localização. A tempo de execução do algoritmo duplicou face aos exemplos anteriores. Assim, neste caso os tempo foram de **62 e 64 segundos**, respetivamente.

Por fim, o teste a ser apresentado é o teste referente a imagens Tekever cujas características são diferentes das anteriores e apresentam o teste final ao algoritmo e ao procedimento.

A Figura 33 representa o resultado da aplicação do algoritmo de comparação a 2 imagens fornecidas pela Tekever. Os resultados, mais uma vez, comprovam a robustez do processo que perante imagens com características diferentes, encontra de forma correta semelhanças para a realização do registo geográfico. O tempo de execução do algoritmo nesta imagem é perto 160 segundos, aproximadamente **2 minutos e 40 segundos**.

De notar que a duração foi 3x superior à duração do exemplo anterior, e cerca de 5x superior à do primeiro exemplo.

Este aumento no tempo de execução do algoritmo surge do aumento da complexidade das imagens da Tekever face as anteriores.

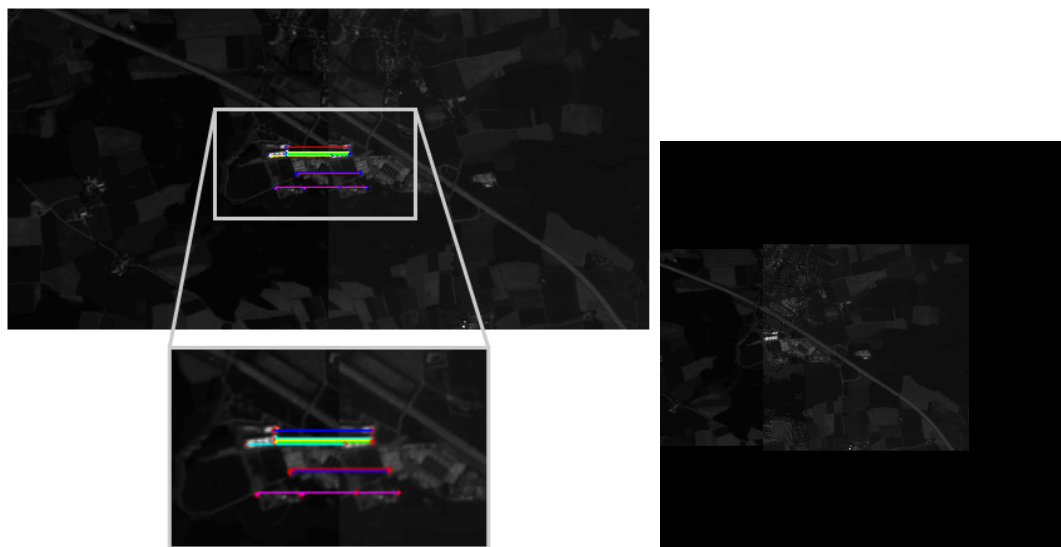


Figura 32: Teste 4 - Comparação entre imagens SAR do *dataset* Ger Imagery (2).

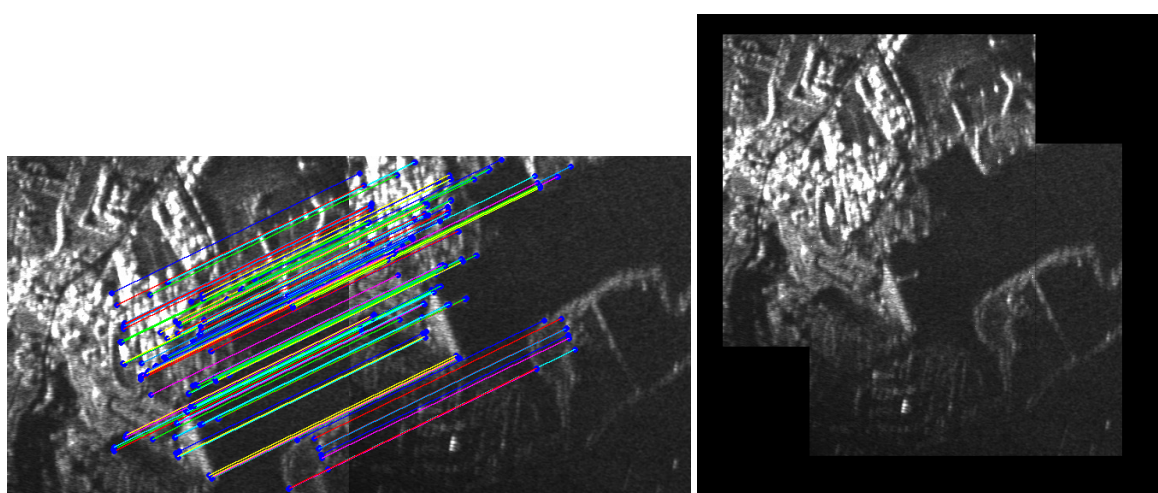


Figura 33: Teste 5 - Comparação entre imagens SAR da Tekever.

3.3 SUMÁRIO

Este capítulo tem como objetivo o estudo e desenvolvimento de uma forma de facilitar o trabalho humano envolvido no registo geográfico de imagens SAR.

A primeira abordagem a ser sugerida tratou-se de uma comparação entre diferentes imagens, uma SAR e outra ótica, Secção 3.1.1. Por representar uma forma rápida e fácil de realizar o registo geográfico de imagens SAR, eliminando praticamente por completo a necessidade de realizar a localização das imagens por parte de um utilizador.

No entanto, esta solução não obteve os resultados esperados e revelou alguns problemas, causados pelas diferentes características dos sensores de obtenção das imagens e, por consequência, das imagens em si, Secção 3.1.2. Como visto, a abordagem envolvendo o algoritmo ASIFT revelou-se incapaz de encontrar semelhanças necessárias entre imagens de maneira a possibilitar um correto registo.

As questões levantadas pela abordagem anterior apontaram para um método de registo diferente do inicialmente planeado.

Se por um lado for utilizada uma comparação SAR-SAR, ao invés da utilização uma comparação SAR-Ótica, em que uma das imagens SAR (anotada) serve de referência para as imagens das redondezas, a solução exige maior trabalho do utilizador durante o processo, Secção 3.2.1.

Por outro lado, a qualidade do registo tende a aumentar, uma vez que imagens com as mesmas características torna a comparação um processo mais equilibrado. Com algumas modificações ao algoritmo, surge um comparador (SAR-Harris) capaz de suprimir os problemas das imagens SAR e assim aumentar o número de semelhanças encontradas entre as imagens, Secção 3.2.1.

Tabela 5: Resumo dos testes dos comparadores para registo geográfico.

Algoritmo	Descrição Teste	Resultado	Figura
ASIFT	Inicial, com Zoom elevado	Positivo	Figura 23
	Nível Zoom mais reduzido	Negativo	Figura 24
	Imagens com a mesma orientação	Negativo	Figura 25
	Imagem com resize	Negativo	Figura 26
	Alteração de filtro	Melhor, mas Negativo	Figura 27
SAR-Harris	Imagem SEN12MS	Positivo	Figura 29
	Imagem SEN12MS, Ângulo diferente	Positivo	Figura 30
	Imagens Ger Imagery	Positivo	Figura 31 Figura 32
	Imagem Tekever	Positivo	Figura 33

SEGMENTAÇÃO DE IMAGENS SAR

Uma das inconveniências das imagens SAR é, como referido anteriormente, a elevada complexidade de interpretação da imagem. Desta dificuldade surge o segundo problema identificado, a identificação de locais e características na imagem é realizada por uma pessoa com experiência em SAR. E mesmo para alguém com experiência este é um processo minucioso e demorado.

Com o objetivo de facilitar esta tarefa, foi idealizado um componente capaz de realizar a segmentação da imagem.

Segmentação de imagem é o termo utilizado para a divisão/classificação dos píxeis de uma imagem de acordo com um critério. Um algoritmo de segmentação recebe uma imagem como *input* e cria uma máscara (colorida ou não) da imagem para uma mais fácil e objetiva interpretação do que está na imagem.

Como em qualquer problema/algoritmo de *machine learning* a qualidade do modelo é resultado da qualidade dos dados.

Assim, é de extrema importância para o sucesso de um algoritmo de segmentação que exista uma grande quantidade de dados, e que estes estejam bem anotados.

Uma vez que as imagens fornecidas são insuficientes para o treino de uma Rede Neuronal, foi necessária a utilização de *datasets* públicos. Os *datasets* utilizados nesta componente são o **SEN12MS** e o **Ger Imagery**.

O *dataset* público **SEN12MS**, é uma amostra com cerca de 45753 imagens com algum tratamento, e que fazem parte do *dataset* original SEN12 que contém imagens dos satélites *Sentinel-1* e *Sentinel-2*, anotadas. (Schmitt et al., 2019)

É possível ver alguns exemplos de imagens na Figura 34.

O segundo *dataset* utilizado tem o nome de **Ger Imagery**, e apenas contém imagens de duas cidades alemãs, com resolução de 8149x5957 e 5596x6031, respetivamente. (Bahmanyar et al., 2018)

As imagens são divididas em 1970 imagens de 256x256 píxeis. Apesar de possuir um número muito inferior de imagens em relação ao SEN12MS, estas encontram-se com uma anotação significativamente mais minuciosa, como é possível verificar na Figura 35.

Ao longo desta secção serão apresentadas algumas formas de resolver o problema descrito, na tentativa de encontrar a que entre o estado de arte tem a capacidade de melhor resolver o problema e pelo contrário, as abordagens que não possuem interesse para o problema em questão.

A primeira abordagem à resolução do problema surge através da utilização de redes neuronais.

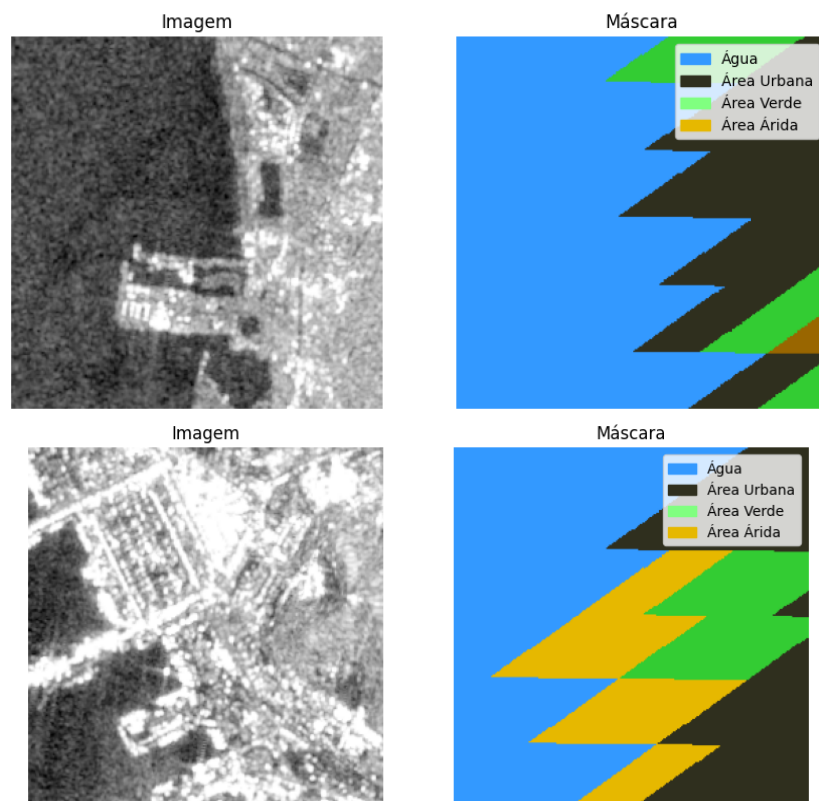


Figura 34: Exemplos de imagens que compõem o *dataset* **SEN12MS** e a anotação respetiva.

4.1 REDES NEURONAIS

No que diz respeito a problemas de segmentação, as abordagens mais utilizadas consistem na utilização de redes neurais.

Daí, a primeira tentativa de resolver este problema é através da utilização deste tipo de soluções.

Assim, quando a rede neuronal estiver preparada para ser utilizada o processo será realizado da seguinte forma.

1. Pré-Processamento da Imagem;
2. Inferência com recurso a rede neuronal;
3. Preparação da imagem para exibição.

A Figura 36, representa em forma de diagrama o processo de segmentação de uma imagem SAR, com recurso a uma rede neuronal.

Assim, uma vez recolhida uma imagem SAR, esta passa por um pré-processamento de maneira a diminuir a influência do *speckle* na imagem e assim facilitar sua compreensão.

Posteriormente, a imagem entra no componente mais importante de todo o algoritmo de segmentação, a rede neuronal.

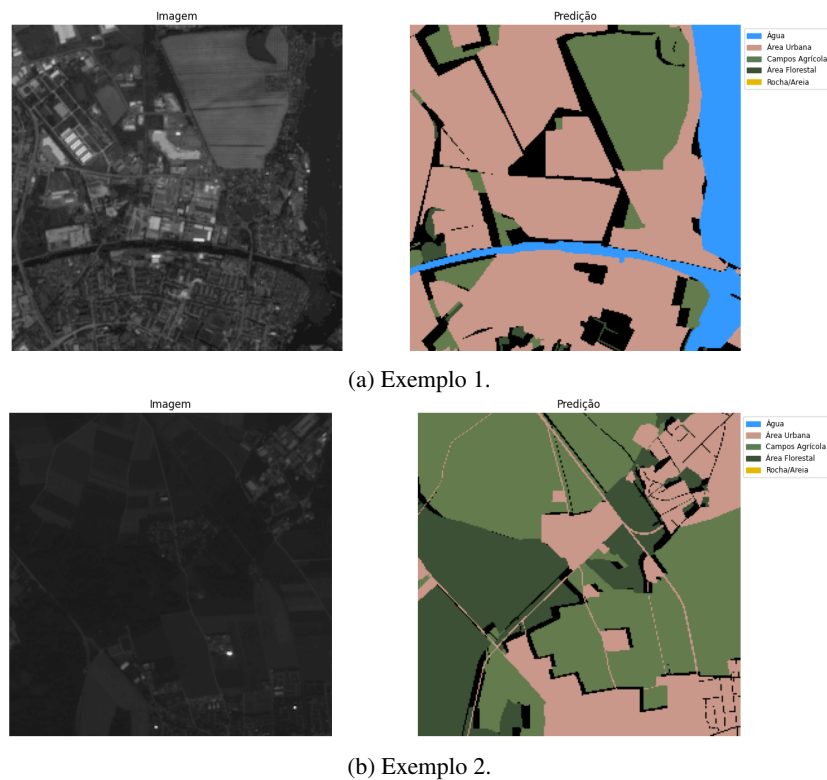


Figura 35: Exemplos de imagens que compõem o *dataset Ger Imagery* e a anotação respetiva.

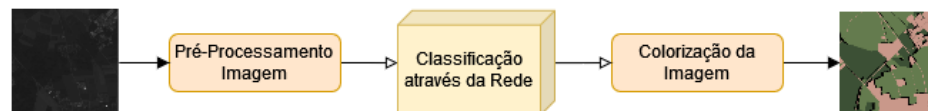


Figura 36: Diagrama do processo de segmentação com rede neuronal.

Para que os resultados da rede neuronal sejam os esperados é relevante controlar o seu treino. Na construção do *script* de treino, por questão de organização e simplificação foi utilizado um **Jupyter Notebook**.

Assim, este *script* está dividido em 3 fases.

1. **Load Dataset:** Na fase inicial do processo de treino é realizado o tratamento destes *datasets*, devido ao facto de estarem organizados num formato diferente daquele que é o necessário para a utilização. Para facilitar a forma como os dados são carregados, foi construída uma classe encarregue de carregar para memória os conjuntos de imagens. Uma vez que seria necessária uma capacidade gigantesca de memória para poder trazer as imagens todas para memória para serem processadas, a **classe Dataset** é responsável por carregar as imagens necessários no momento de treino, de acordo com o tamanho do *batch*. Uma vez que a classe funciona através de caminhos para as imagens, os dados em memória ocupam um espaço significativamente inferior.

2. **Treino da Rede:** A segunda etapa do processo de treino é responsável por carregar o modelo, preparar o mesmo para o treino e, por fim, iniciar o seu treino. O *Load* do modelo é realizado com recurso a um *script* externo que, a utilização deste *script* facilita a alteração de modelo de *DeepLearning*, uma vez que funciona como uma 'base de dados' de modelos em que é possível escolher qual o que queremos utilizar no treino. Depois, de carregado o modelo pretendido, é necessário compilar o mesmo atribuindo-lhe uma função de *Loss* e métricas de desempenho. O processo de atribuição da *Loss* é análogo ao processo de seleção do modelo, a função é selecionada de um documento externo. Por fim, são atribuídas as variáveis de treino, como as *epochs* (que são as vezes que o modelo vai inferir sobre as imagens) e *batch size* (conjunto de imagens a ser carregado). Terminada esta fase de definição de modelo, *Loss* e variáveis, é iniciado o treino da rede.
3. **Análise de resultados:** De maneira a guardar os resultados do treino, o histórico do treino é armazenado para posteriormente ser utilizado e analisado. De forma concreta, são guardados os pesos da rede, e as métricas ao longo das *epochs*. No final do treino da rede, são utilizadas as métricas previamente definidas para uma visão inicial da eficácia do processo de treino.

4.1.1 U-Net

A rede U-Net, proposta por [Ronneberger et al. \(2015\)](#), apresenta resultados bastante satisfatórios numa vasta gama de utilizações e problemas.

Pela sua abrangente utilização e resultados, esta arquitetura vai ser utilizada como teste *default* para os seguintes métodos de segmentação.

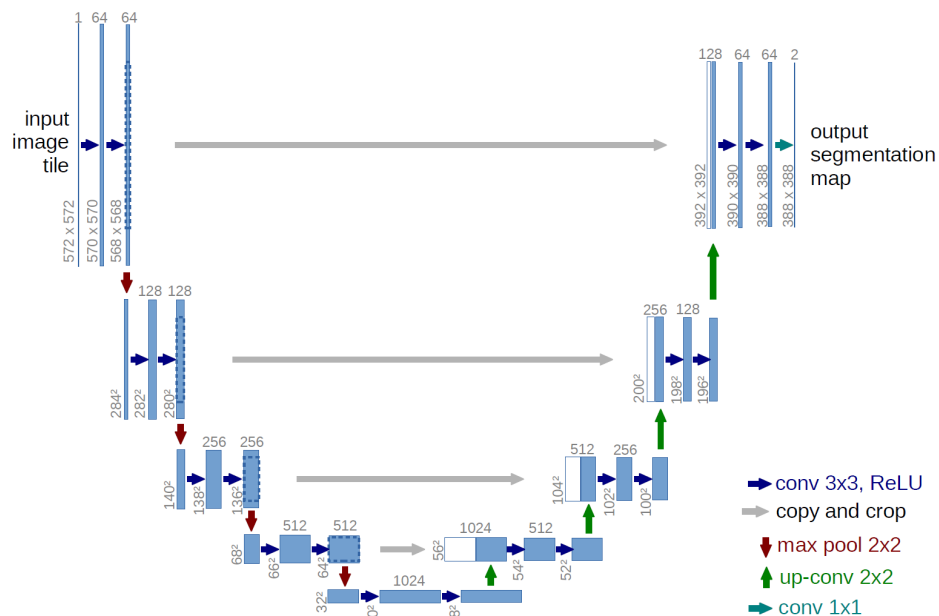


Figura 37: Representação gráfica da rede U-Net (Crédito: (Morel and Yu, 2009))

Como podemos ver na Figura 37, o nome U-Net surge da representação em formato de U que facilita a percepção do que acontece as imagens quando passam por esta rede neuronal.

A arquitetura U-Net faz parte de uma categoria de redes chamadas **Autoencoders (Encoder-Decoder)**.

Isto é, a parte inicial da rede, de nome *feature extractor*, é responsável por extrair informação da imagem através da sequência de filtros que vão realçando as características mais relevantes da imagem.

No fim, a base do U fazendo analogia à imagem, resulta um *array z* que é a versão **encoded** da imagem, ou seja, as **features mais relevantes da imagem**. Este *array* é o ponto de entrada da componente de **Decode** da rede.

No que concerne a componente de *decode*, são realizadas **operações opostas** às de *encode* e assim é realizado um aumento progressivo do *output* até este ter a resolução da imagem original. Ao introduzir um classificador na última camada de *decode* procuramos a previsão da classe, realizando assim a segmentação da imagem original.

Um componente de importante nestas redes são as **skip connections**, que são ligações intermédias entre o *encoder* e o *decoder* que permitem à rede inferir sobre *features* a diferentes níveis de escala.

Treino

SEN12MS O primeiro treino realizado foi com recurso ao *dataset* **SEN12MS**.

As imagens do *dataset* estão em formato **tiff**. Por esse motivo, as imagens foram separadas do ficheiro tiff em que estavam armazenadas, guardadas com identificadores de região, estação, tipo, *patch* e nº de exemplar (p.e, ROIs1868_summer_s1_4_p48). Com o mesmo nome, de maneira a identificar o seu dual, as máscaras foram também extraídas de um ficheiro tiff para uma outra diretoria.

As variáveis de treino utilizadas foram as seguintes.

- **Função Loss:** *Sparse categorical crossentropy*;
- **Otimizador:** Rmsprop;
- **Número de épocas:** 10;
- **Batch size:** 8.

O treino foi realizado com recurso a uma placa gráfica GTX 960M, e demorou, aproximadamente, 9 horas, dando uma média de 55 minutos por época. Os resultados do treino foram os seguintes.

- **Varição da Loss:** A *Loss* iniciou-se com valores muito próximos dos 13.50, e com estabilização na época número 8 com o valor **1.00**;
- **Accuracy:** A *accuracy* final ronda os **70%**, com início nos 50%.

Os resultados mostram que o modelo possui uma **boa capacidade de classificação**, e que apresenta grande potencial na resolução do problema de segmentação de imagens SAR. Porém, só aquando dos testes é que se pode verificar o que realmente representam estes valores.

GER IMAGERY De seguida, foi realizado um treino com os dados do *dataset Ger Imagery*. Todas os parâmetros de treino se mantiveram.

Após **40 minutos** de execução, os resultados foram os seguintes.

- **Variação da Loss:** O valor da *Loss* na primeira *epoch* era de **1.3452**. No final do treino, a métrica apresentou o valor **0.6122**.
- **Accuracy:** A *accuracy* inicial do modelo rondava os 55%. Na conclusão do processo de treino a *accuracy* registada foi de **79%**.

Como se pode ver pelos resultados acima, o processo de treino decorreu de forma satisfatória apontando para uma rede capaz de realizar uma ótima classificação.

O facto de existirem poucos exemplos de treino no *dataset* pode tornar o modelo pouco robusto e consistente nas suas classificações.

já o facto de possuir aproximadamente 20x menos imagens de o SEN12MS torna a execução do processo de treino significativamente mais rápido como se pode verificar com uma **duração 15x inferior** ao treino com SEN12MS.

Na secção seguinte serão apresentados os testes às duas redes treinadas.

Testes

O teste de problemas de segmentação é mais fácil do que testar as semelhanças entre imagens, como feito anteriormente. Por ser uma rede neuronal é treinada sobre um conjunto de dados anotados, e assim age como uma *black box*, em que entra um conjunto de imagens e saem máscaras, de acordo com as semelhanças que a rede encontrou entre as imagens 'novas' e as vistas anteriormente.

Dessa forma, para validar o desempenho do modelo ML, não é necessário realizar uma verificação de cada imagem em particular. É possível criar um conjunto de imagens para classificação e no final calcular uma métrica de desempenho sobre essas mesmas imagens, comparando com as suas anotações.

De maneira a criar uma noção mais visual do que significam estes valores, foi desenvolvido um programa, que recebe como *inputs*:

- **Modelo:** Que fará a inferência;
- **Imagem:** A imagem a segmentar;
- **Máscara:** A máscara anotada da imagem (se existir);
- **Overlap:** Opção que permite sobrepor a imagem com a máscara calculada.

SEN12MS Assim, foi realizado um teste à rede que treinou com o *dataset* SEN com cerca de 200 imagens que a rede não tinha visto. A validação obteve uma *accuracy* média de **72.1%**. Apesar de mostrar bons números, é bastante complexo ter noção do significado verdadeiro desse números.

Posto isto, e com ajuda do recém-criado programa, é possível ver as previsões da rede e verificar o comportamento da rede. A Figura 38 representa uma imagem classificada pela rede treinada, serve também de exemplo para mostrar como o programa exibe as imagens (i.e., com e sem *overlap*). Na figura é possível, ver que a segmentação apresenta bons resultados, podendo até ser referido que apresenta maior detalhe na segmentação que a própria anotação que, como referido anteriormente, não possui anotações muito minuciosas.

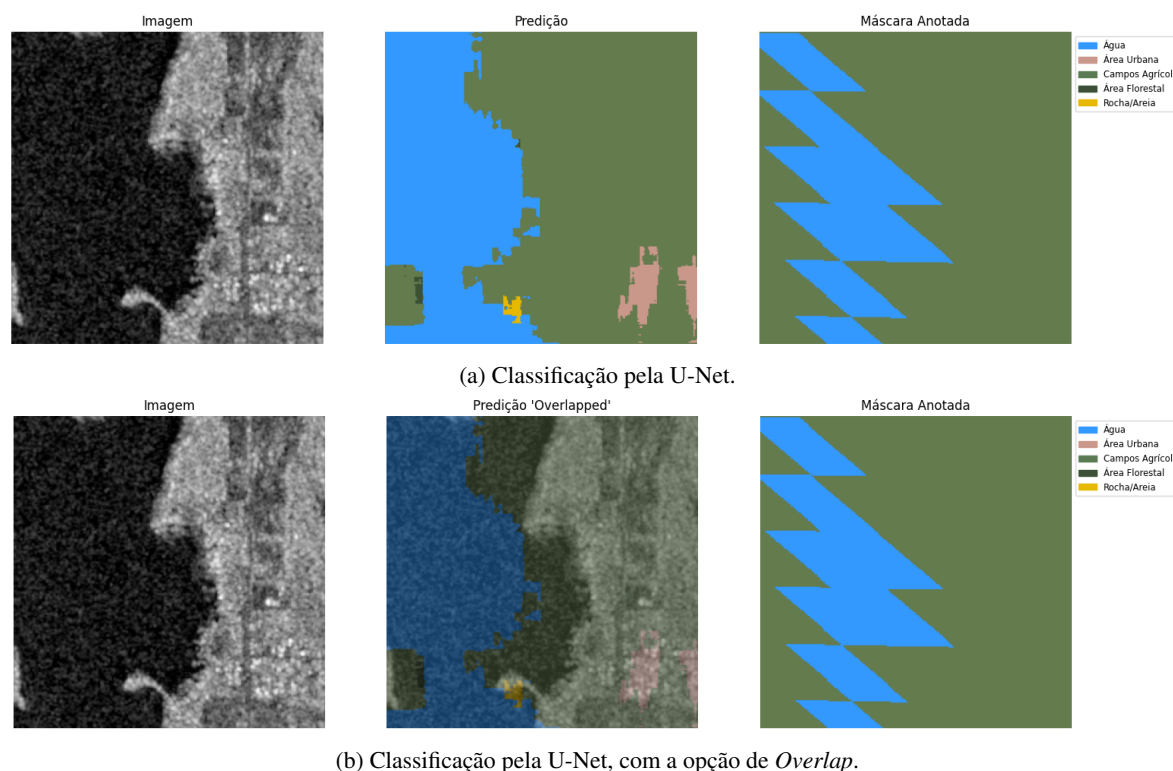


Figura 38: Classificação U-Net, treinada com SEN, de uma imagem do *dataset* SEN12MS, com diferentes opções.

No entanto, a previsão está longe de ser perfeita.

Na Figura 39 são visíveis algumas falhas na classificação, apesar de acertar em grande parte dos pixels (**accuracy** um pouco acima os **73%**), na parte inferior é facilmente distinguível a olho nu a diferença de contraste entre o elemento água, que normalmente se apresenta mais escuro, e 'terra' que possui um tom mais claro.

Uma das causas possíveis para este tipo de falhas é a falta de detalhe nas anotações que gera cálculos contraditórios durante o treino da rede.

A aplicabilidade da rede é testada com utilização de imagens fora do SEN12MS.

A Figura 40 exibe as classificações feitas pela rede sobre uma das imagens pertencente ao *dataset* **Ger Imagery**. Como é possível verificar, os resultados não são da mesma ordem dos anteriores (*dataset* SEN12MS). A qualidade da classificação teve uma **diminuição significativa**.

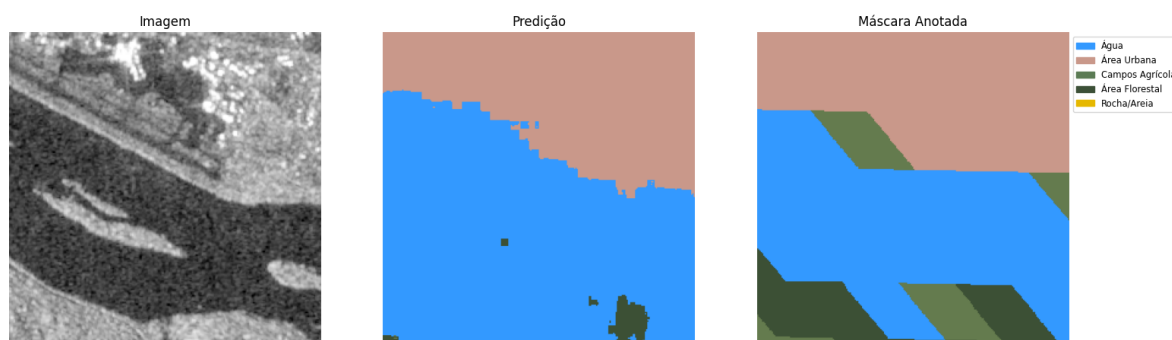


Figura 39: Classificação U-Net, treinada com SEN, de uma imagem do *dataset* SEN12MS.

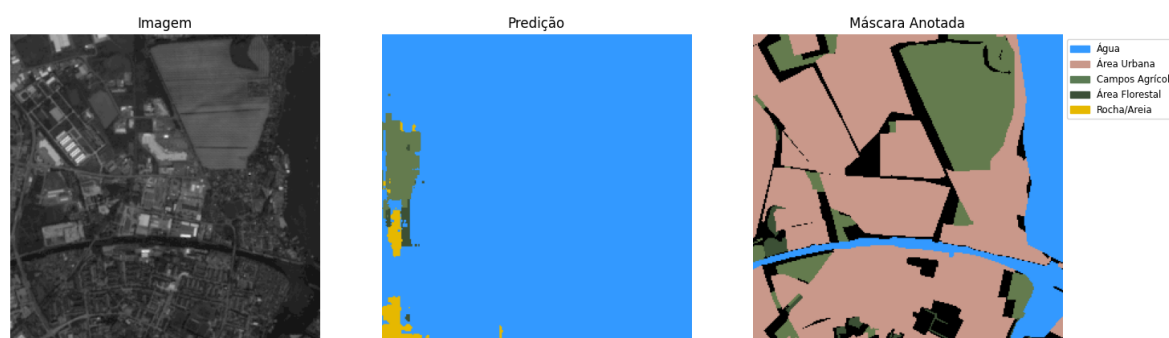


Figura 40: Classificação U-Net, treinada com SEN, de uma imagem do *dataset* Ger Imagery.

A existência da anotação permite, uma vez mais, quantificar o desempenho da rede, que neste caso se encontra nos **18.7%** de **accuracy**. Permite também a representação visual do **baixo** valor de **accuracy**. É possível observar na imagem central (que representa a classificação), uma enorme mancha de azul, indicativa de água, que cobre grande parte da imagem, enquanto a máscara nos mostra que grande parte dessa área é 'Zona Urbana'. Este tipo de oscilações na qualidade da classificação levanta questões acerca da robustez da rede perante imagens SAR diferentes.

As seguintes figuras (Figura 41 e Figura 42), são testes da rede neuronal a imagens da **Tekever**, com o objetivo de averiguar o comportamento da mesma face a imagens com características diferentes da que a rede treinou e ainda, diferentes das que compõem o *dataset* Ger Imagery.

Na Figura 41 é exibida uma imagem captada pela Tekever. Na parte direita da imagem está exibida a predição feita pela rede.

Quando se observa a imagem da direita da Figura 41, a classificação apresenta o mesmo problema da imagem do Ger Imagery, uma mancha enorme de azul o que significa que a rede interpretou grande parte da imagem SAR como sendo água. Apesar de não existir uma máscara que indique um valor numérico de desempenho e facilite a interpretação da imagem SAR, é perceptível que a classificação está errada.

De maneira a retirar melhores conclusões acerca da classificação tão enviesada para a classe água, foi realizada a inferência sobre uma outra imagem fornecida pela Tekever, desta vez de uma zona diferente.

É de notar que aconteceu o mesmo que nos testes anteriores. Na imagem da direita da Figura 42, que representa a classificação da rede, é visível uma mancha de azul (água) que cobre grande parte da imagem.

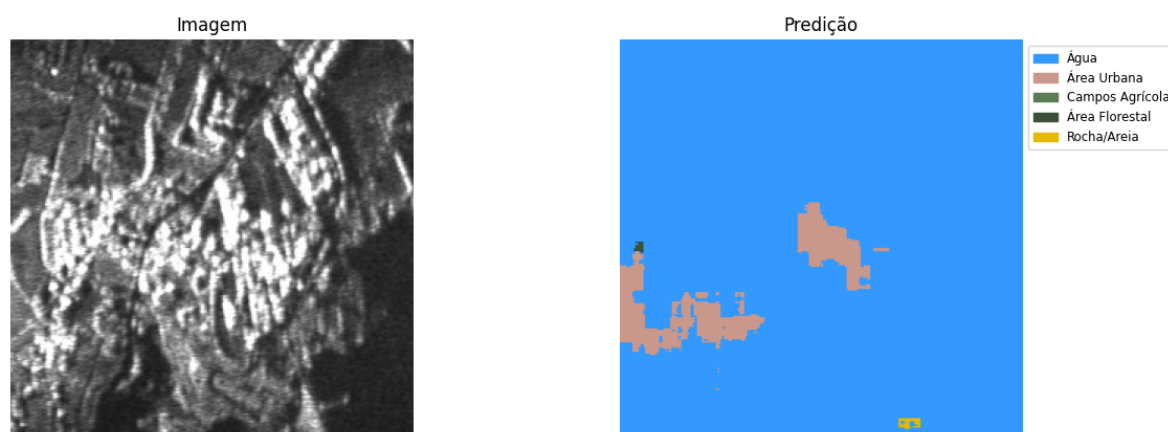


Figura 41: Classificação de uma imagem SAR, da Tekever, pela U-Net, treinada com SEN.

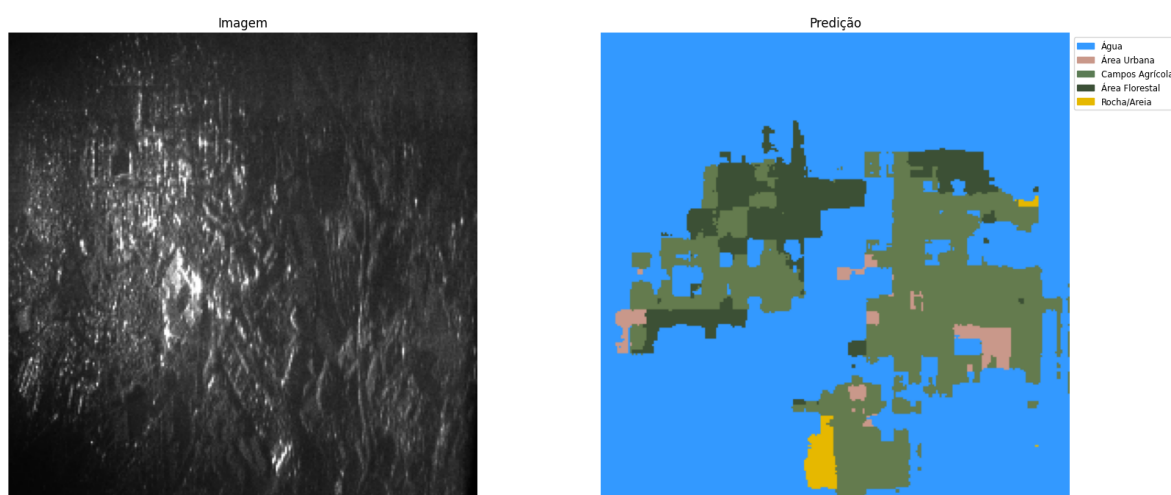


Figura 42: Classificação pela U-Net, treinada com SEN, de uma imagem SAR da Tekever, numa localização diferente.

Existe a possibilidade de serem as características da Figura 41 que estejam 'confundir' a rede.

Posto isto, é importante realizar um processo de análise sobre o problema.

A Figura 43 tem o objetivo de comparar visualmente as diferentes imagens, a da esquerda que representa as imagens do **dataset SEN12MS** e à direita a imagem SAR fornecida pela **Tekever**.

A diferença entre as imagens que mais salta à vista é o facto de as imagens captadas pelo SAR da **Tekever**, possui uma **tonalidade mais escura** do que as imagens captadas pelos satélites *Sentinel-1* & 2.

Uma forma de testar se esta diferença está a fazer muita influência sobre a classificação da rede é tentar aproximar as cores da imagem da Tekever às cores das imagens do **dataset**. O inverso não faz tanto sentido uma vez que a rede treinou sobre o **dataset SEN12MS**.

Para aproximar as tonalidades foi utilizado um método chamado **Gamma Correction** (correção de Gamma, em português) que controla o brilho geral de uma imagem.

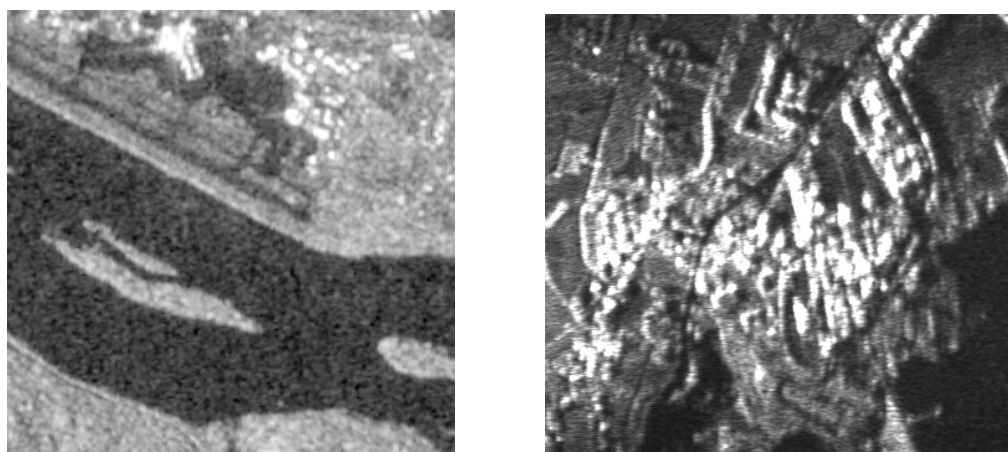


Figura 43: Comparação entre uma uma imagem do *dataset* **SEN12MS** e do SAR da **Tekever**.

Assim, o processo será aumentar progressivamente o brilho da imagem e perceber que efeito esse aumento faz na classificação.

As seguintes imagens representam o aumento progressivo de brilho da imagem SAR fornecida pela Tekever.

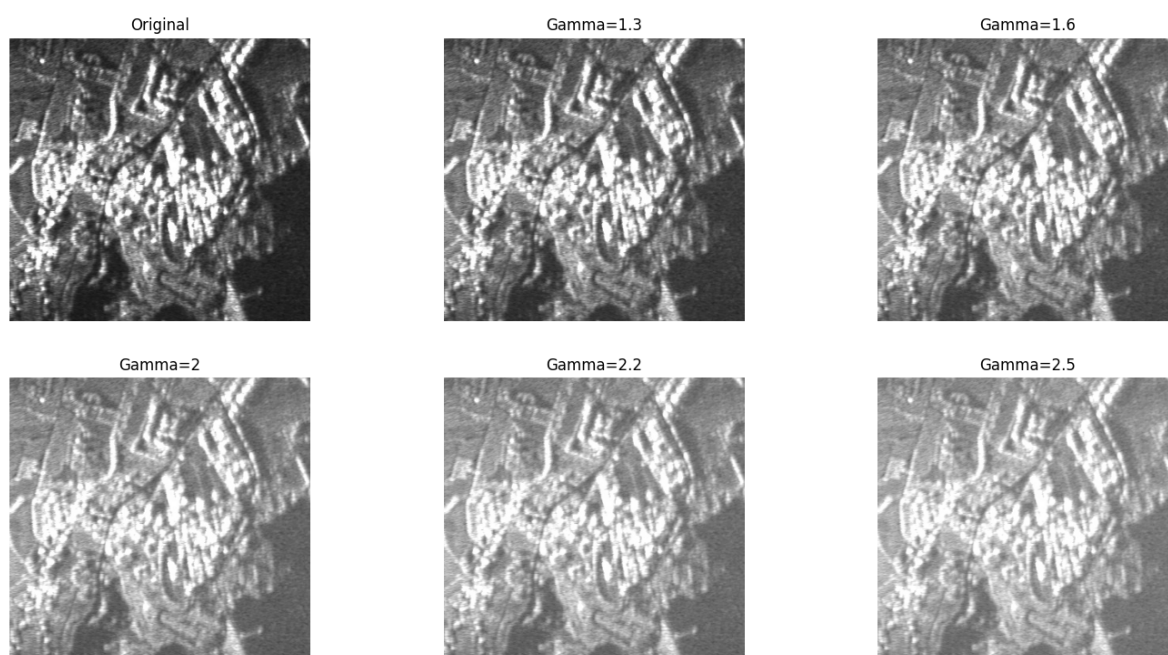


Figura 44: Alterações da imagem através da utilização do método *Gamma Correction*.

Foi então testada a classificação da U-Net nestas imagens, por forma a descobrir se a alteração do brilho da imagem tem influência na classificação e assim confirmar a possível explicação para a tendência de classificação como água das imagens fornecidas pela Tekever.

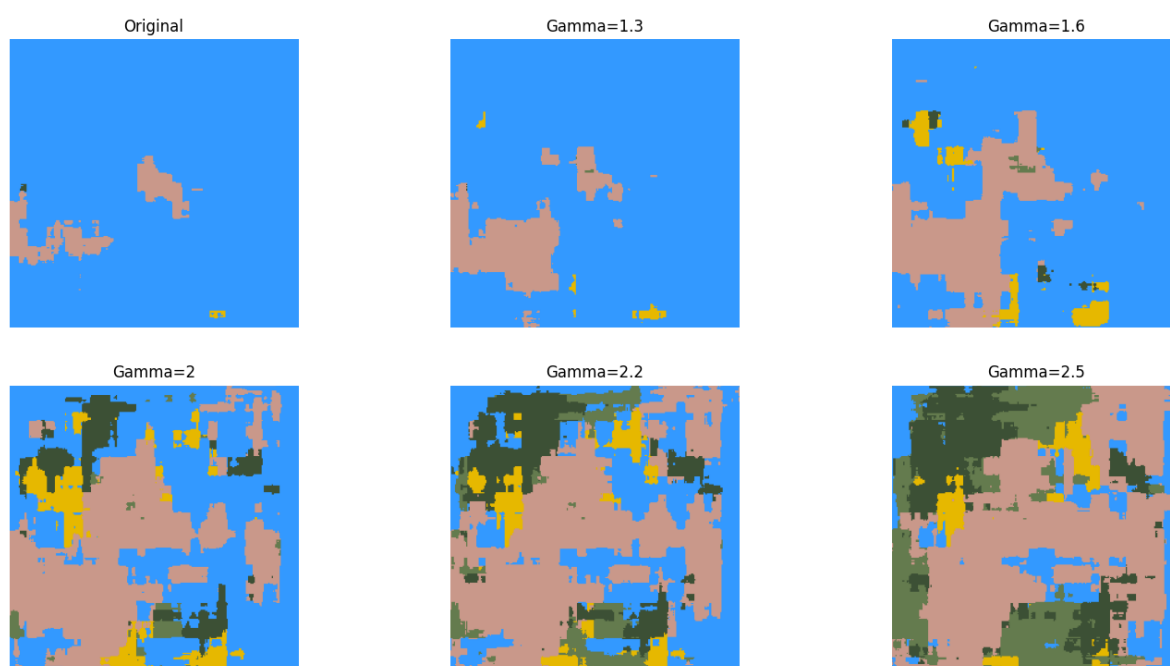


Figura 45: Classificação das imagens alteradas através do método *Gamma Correction*.

Como é possível verificar na Figura 45, à medida que a *gamma* aumenta (i.e., que o brilho da imagem aumento), a percentagem de imagem que a rede classifica como água, **diminui**. Esta diminuição de erros na classificação vem confirmar a hipótese levantada anteriormente, de que a rede é influenciada pela tonalidade da imagem que lhe é enviada para testar.

GER IMAGERY Da mesma forma que no teste anterior, a rede que treinou sobre os dados do **Ger Imagery** foi testada quer em quantidade quer em qualidade.

O resultado da validação com cerca de **100 imagens** que compõem o *dataset*, e que a rede não viu, obteve um resultado de **73%**.

Novamente, a rede revela que apresenta bons resultados com imagens de características semelhantes ao seu treino.

De seguida, é importante compreender o que significam os 70% em termos de segmentação da imagem e a robustez da rede quando testada com imagens diferentes.

As primeiras imagens a serem analisadas são aquelas que compõem o *dataset*.

A Figuras 46 e a Figura 47 exibem a enorme capacidade de segmentação por parte da rede.

É visível na Figura 46 que a quantidade de erros de classificação é muito reduzida, com algumas classificações de pixels como sendo água e na parte inferior da imagem, uma confusão entre Zona Urbana e Zona Agrícola.

O mesmo acontece na Figura 47, o que revela um problema na identificação de água.

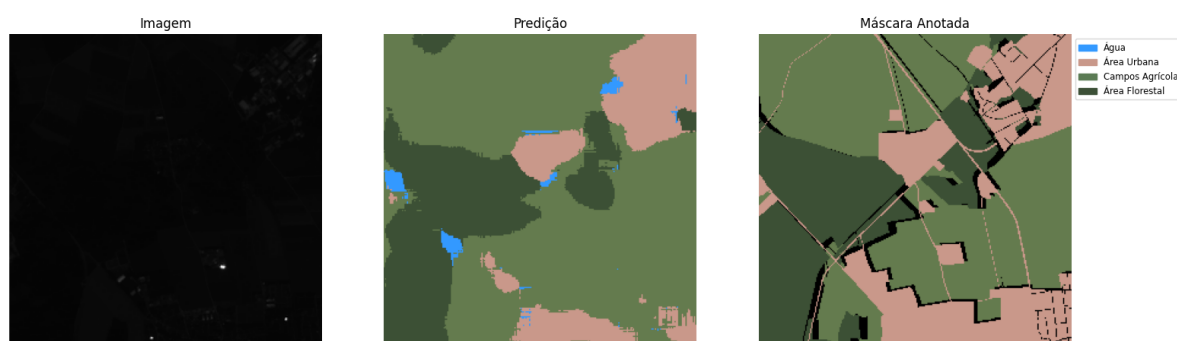


Figura 46: Classificação U-Net, treinada com Ger Imagery, de uma imagem do *dataset* **Ger Imagery**.

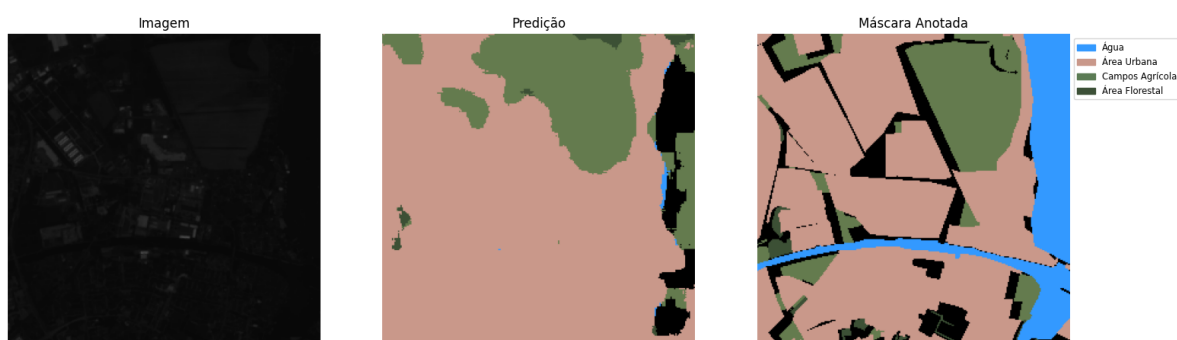


Figura 47: Classificação U-Net, treinada com Ger Imagery, de uma imagem do *dataset* **Ger Imagery** numa localização diferente.

Uma das possíveis causas para essa incapacidade é a semelhança de tons em zonas com água e as restantes. Com exceção dessas zonas, a segmentação da imagem apresenta resultados muito bons.

É agora momento de perceber se a qualidade de classificação se mantém quando são utilizadas diferentes imagens. A primeira imagem não relacionada com o *dataset* de treino é a imagem do **SEN12MS** que é utilizada anteriormente.

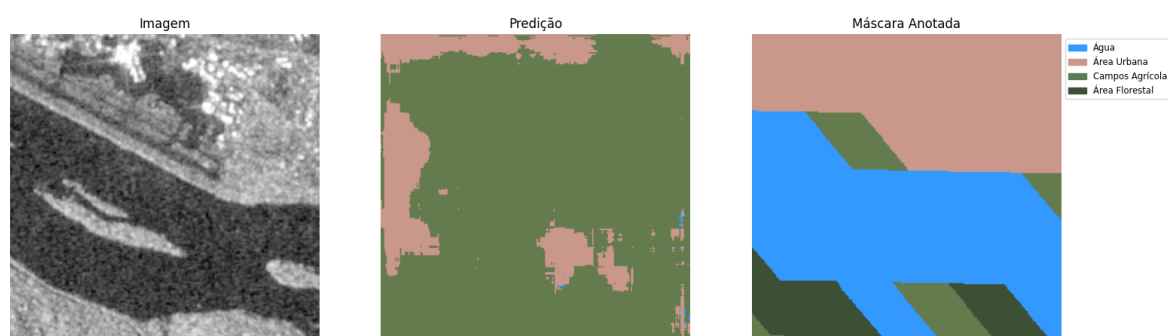


Figura 48: Classificação U-Net, treinada com Ger Imagery, de uma imagem do *dataset* **SEN12MS**.

A Figura 48 mostra uma classificação muito fraca por parte da rede. Grande parte da imagem é apresentada como Zona Agrícola, quando se percebe claramente a existência de água.

Este erro aponta o mesmo problema que no exemplo anterior, a incapacidade de diferenciar de forma clara, pixels que representem ambiente com presença de água.

No entanto, neste caso a causa mais provável para este erro é o desconhecimento, por parte da rede, das variações na intensidade das cores e o que representam em imagens deste *dataset*.

Ao contrário da rede que treinou sobre o SEN12MS, esta treinou sobre imagens de tons mais escuros. Essa diferença no treino revelou-se agora um fator-chave na classificação de uma imagem SAR.

Por fim, resta testar e analisar o *output* da rede quando utilizada uma imagem fornecida pela **Tekever**.

À priori, existe uma maior semelhança entre a imagem Tekever e o *dataset* de treino, do que entre os *datasets* SEN12MS e Ger Imagery.

Sendo assim, é esperado um melhor resultado face ao exemplo anterior.

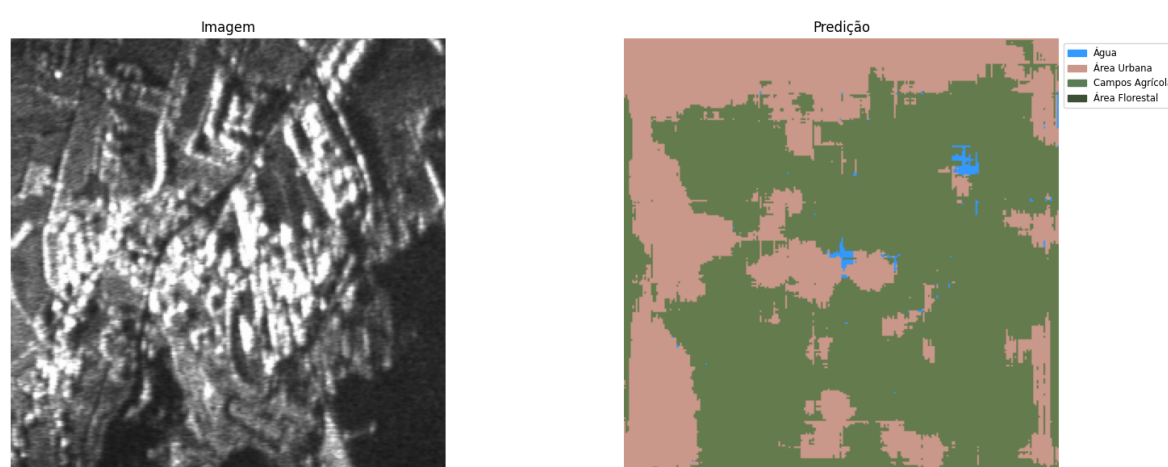


Figura 49: Classificação da U-Net, treinada com Ger Imagery, de uma imagem da Tekever.

Da Figura 49, é possível retirar que mesmo sendo a imagem mais semelhante do contexto de treino que o exemplo anterior a rede não é capaz de realizar uma segmentação correta.

A mesma causa dos exemplos anteriores é apontada como sendo a mais provável, a incapacidade de adaptação da rede a variações que não estavam presentes em ambiente treino.

Esta influência que a imagem faz na classificação revela-se um problema, uma vez que, devido às suas características, as imagens SAR podem diferir umas das outras.

Se o modelo de ML estiver muito preso às tonalidades das imagens as classificações saem muito imprevisíveis e muito pouco robustas a variações, algo que não é de todo desejável para uma ferramenta deste género.

4.1.2 DSCNN

Como visto anteriormente a U-Net apresentou resultados satisfatórios na segmentação de imagens SAR dos *datasets* SEN12MS e Ger Imagery, o que demonstra o esperado potencial dos AE em tarefas de segmentação.

Porém, existem também alguns problemas associados e que são comuns na utilização de redes neurais. Neste caso, é possível ver que a rede adaptou-se ao intervalo de cores das imagens existentes no *dataset* e

assim a classificação só é boa quando as imagens se aproximam do aspeto das imagens do *dataset*. Esta característica é limitadora para imagens SAR, uma vez que há a possibilidade de existirem diferenças de tonalidade entre imagens devido às características físicas do sensor.

Na tentativa de resolução desta limitação surge a procura por uma nova rede capaz de realizar uma classificação mais robusta.

Sendo a arquitetura de *autoencoder* a mais utilizada, é natural que a nova rede possua um formato semelhante, com alguns *updates* que tornem a classificação e segmentação mais consistente e com mais qualidade.

Geng et al. (2017) apresenta uma solução com uma visão diferente sobre *encode* e *decode* de um *autoencoder*. A abordagem promete ser capaz de lidar com as particularidades das imagens SAR e trazer melhores resultados à tarefa de segmentação. De uma forma simplificada, a imagem SAR passa por uma série de *encoders* com capacidade de extrair informações diferentes da imagem.

À medida que a imagem flui pela rede, diferentes características vão sendo evidenciadas e no final, um classificador é responsável pela atribuição da classificação a cada pixel, atribuindo a máscara à imagem original, concluindo a tarefa.

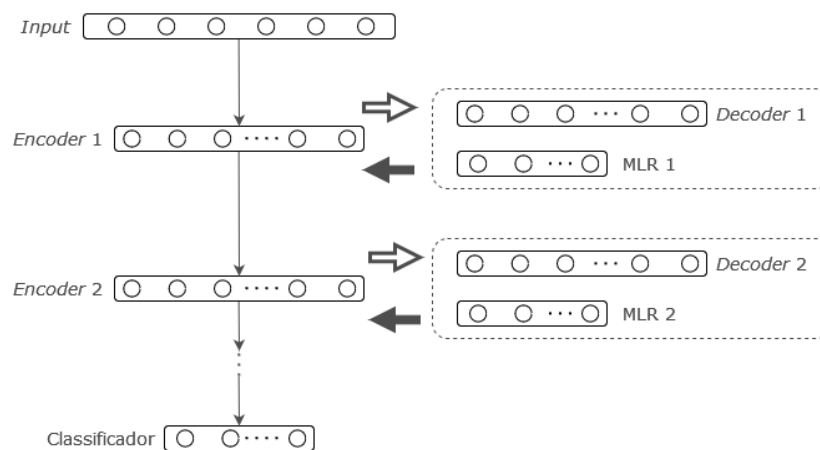


Figura 50: Comparação entre a classificação com inicialização aleatória (esquerda) e classificação ideal (direita) (Crédito:Geng et al. (2017))

Esta arquitetura completa a anterior no sentido em que cada camada da rede é em si um autoencoder, o que permite uma otimização das *features* da imagem que a rede está a recolher.

A imagem sofre uma diminuição de especificidade e depois o respetivo aumento da forma que é desejada (neste caso em forma de máscara), mas sim repetidas operações de diminuição e aumento resultando numa extração mais completa de características da imagem.

Como referido anteriormente, esta arquitetura caracteriza-se por ser constituída por diversos autoencoders. Este tipo de autoencoder apresenta 2 particularidades que o diferencia..

- São utilizados **Contractive AutoEncoders (CAE)**;
- São **supervisionados**.

Devido a estas características o autor deu-lhes o nome de **Supervised Contractive Autoencoders (SCAE)**.

A utilização de CAEs é justificada pela capacidade que estes possuem de tornar a rede mais robusta a pequenas variações no *dataset* de treino. Esta capacidade é obtida através da utilização de um termo de penalização na função de *Loss*, que é o objetivo a minimizar. Este termo utiliza a **Forbenius norm** da **matriz jacobiana da ativação** da componente de *encode*.

É supervisionado uma vez que no treino do *autoencoder* é sustentado e avaliado de acordo com o resultado esperado. A realização de treino supervisionado direciona os pesos da rede para o *output* desejado. Assim no final do CAE é adicionado um módulo **Multinomial Logistic Regression (MLR)** que relaciona as *labels* de treino e o *update* dos parâmetros.

Em relação ao treino de cada camada, que por sua vez é um SCAE, pode ser dividido em 2 partes. A primeira tem o nome de **pré-treino**, e diz respeito ao treino das componentes de *encode* e *decode* da rede. A segunda tem a função de fazer o **update** da componente *encode* da rede, de acordo com os resultados do MLR. O processo de *encode* para um *dataset* X , e para as *labels* correspondentes y , é dado da seguinte forma.

Primeiro é realizada a operação de *encode*.

$$h_i^m = f \left(W_1^m h_i^{m-1} + b_1^m \right) \quad (35)$$

Posteriormente, o *decode* é dado pela seguinte expressão.

$$\hat{x}_i^m = g \left(W_1^{m'} h_i^m + b_2^m \right) \quad (36)$$

Por fim a expressão que caracteriza a função objetivo do pré-treino é definida da seguinte forma.

$$J_{pretrain}^m(\theta^m) = \frac{1}{2N} \sum_{i=1}^N (\|\hat{x}_i^m - h_i^{m-1}\|_2^2 + \lambda \sum_{j=1}^{n_m} \left(h_{i,j}^m (1 - h_{i,j}^m) \right)^2 \|W_{1,j}^m\|_2^2) + \frac{\delta}{2} \|W_1^m\|_F^2 \quad (37)$$

Na função, o primeiro termo define o erro de reconstrução entre o input e o output. Por sua vez, o segundo termo é a representação do *contractive penalty*, referido anteriormente, que torna a representação do *encoder* mais robusta. Por fim, o último termo diz respeito ao *weight decay* que penaliza valores elevados dos parâmetros.

No final deste processo de pré-treino as *labels* e as representações *encoded* h são importadas para o módulo MLR. O módulo tem o objetivo de captar as informações mais relevantes entre a representação *encoded* e as *labels*. Este processo é definido por

$$\rho_i = \frac{1}{\sum_{k=1}^C e^{W_{2,k}^{MT} h_i^{M-1}}} \begin{bmatrix} e^{W_{2,1}^{MT} h_i^{M-1}} \\ e^{W_{2,2}^{MT} h_i^{M-1}} \\ \dots \\ e^{W_{2,C}^{MT} h_i^{M-1}} \end{bmatrix} \quad (38)$$

A função objetivo desta componente

$$J_{update}^m(\theta^m) = -\frac{1}{N} \sum_{i=1}^N y_i \log(p_i^m) + \frac{\tau}{2} \|W_2^m\|_F^2 \quad (39)$$

No final destas duas fases de treino o *update* dos parâmetros foram otimizados de acordo com a seguinte função objetivo.

$$J_{global}^m(\theta^m) = \arg \min \left\{ J_{pretrain}^m(\theta^m) + \gamma J_{update}^m(\theta^m) \right\} \quad (40)$$

Para o treino das SCAE é utilizado o algoritmo **mini-batch stochastic gradient descent**.

O Algoritmo de treino de uma SCAE é realizado da seguinte forma.

Procedimento 1 Treino SCAE

- 1: $T \leftarrow$ Número de Iterações
 - 2: $\theta \leftarrow \text{rand}()$ ▷ Inicialização aleatória de parâmetros
 - 3: **for** $t = 1$ **to** T **do**
 - 4: Realizar o *encode* Equação (35);
 - 5: Realizar o *decode* Equação (36);
 - 6: Calcular o valor da função $J_{pretrain}^m$ Equação (37);
 - 7: Realizar o *update* dos parâmetros θ ;
 - 8: **end for**
 - 9:
 - 10: **for** $t = 1$ **to** T **do**
 - 11: Calcular a probabilidade por MLR Equação (38);
 - 12: Calcular o valor da função J_{update}^m Equação (39);
 - 13: Realizar o *update* dos parâmetros θ ;
 - 14: **end for**
 - 15: **Output:** Parâmetros da SCAE.
-

Múltiplas SCAE são emparelhadas de maneira a formar um *Deep Network*. Onde o *output* de umas camadas serve, como verificado, para o *input* da seguinte.

No topo da rede, e de maneira a realizar a tarefa a que foi idealizada (i.e., realizar a segmentação de uma imagem), encontra-se uma camada responsável pela inferência sobre a classe a que cada um dos pixels da imagem pertence. Esta camada recorre à função **softmax** para realizar a classificação.

A *Loss function* desta última camada é dada pela seguinte expressão.

$$J' \left(W_1^m \Big|_{m=1}^M, b_1^m \Big|_{m=1}^M \right) = \arg \min \left\{ \sum_{i=1}^N \sum_{m=1}^M \|h_i^m - h_i^{m-1}\|_2^2 + \eta \sum_{m=1}^M \|W_1^m\|_F^2 \right\} \quad (41)$$

Todos os pesos e *bias* da rede são otimizados desde a camada final até a inicial.

À semelhança da anterior, esta função de *loss* é resolvida de acordo com **mini-batch stochastic gradient descent**.

De uma forma geral o treino de toda a rede é realizado da seguinte forma.

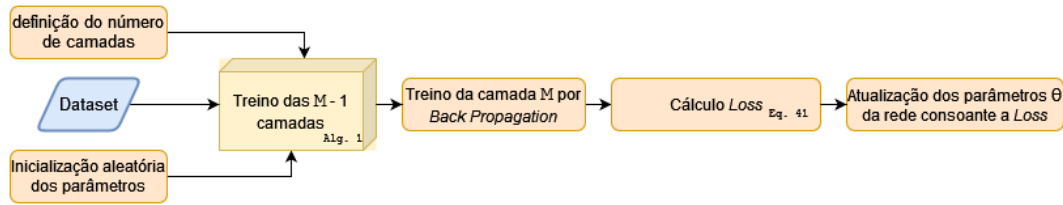


Figura 51: Treino de toda a DSCNN.

No final do treino de toda a rede, como ilustrado na Figura 51, é realizada a tarefa de classificação através do cálculo da probabilidade de um pixel X_i pertencer a uma classe c .

A Equação (42) é a expressão utilizada para a identificação da classe de cada pixel.

$$C(x_i) = \arg \max_{c=1,\dots,C} p(\hat{y}_i = c | h_i^M; W^M, b^M) \quad (42)$$

Treino

Da mesma forma como foi realizado com a rede neuronal U-Net, o treino da DSCNN recorreu à classe **Dataset** previamente definida e construída de maneira a responder às necessidades do treino de redes neurais.

Assim, a classe é responsável pelo carregamento e disponibilização das imagens como explicado anteriormente.

SEN12MS O primeiro *dataset* a ser enviado para o treino da rede é o SEN12MS. No treino, a rede construída apresentará como parâmetros os recomendados pelo seu autor. Desse modo, os parâmetros de treino da rede são os seguintes.

- **Número de camadas:** 3;
- **Nodos por camada:** 256, 128, 64;
- **Learning Rate:** 1×10^{-3} .

Em relação ao treino, este foi realizado segundo as seguintes diretrizes.

- **Função Loss:** *Sparse categorical crossentropy*;
- **Otimizador:** Adam;
- **Número de épocas:** 30;
- **Batch size:** 3.

À semelhança do que foi visto anteriormente, o treino seria realizado com recurso a uma GTX 960M. No entanto, **não foi possível a conclusão deste treino.**

A rede revelou-se demasiado exigente no que toca a recursos, terminando o treino momentos após ser iniciado. Foram feitos vários esforços na alteração variáveis de maneira a tentar diminuir a carga da rede com custo elevado para a duração do treino. Mesmo assim a exigência da rede em termos computacionais é muito superior ao que a plataforma de treino conseguia oferecer. Assim, o treino e teste desta rede com o *dataset* SEN12MS termina, com vista a uma otimização das operações da rede e um aumento de poder computacional para que a eficácia desta rede seja corretamente testado.

GER IMAGERY Segue-se então com o treino sobre o *dataset* Ger Imagery, uma vez que não foi possível terminar o teste anterior, foi também realizado este treino como forma de comparação.

Tanto as definições da rede, como os parâmetro de treino mantiveram-se nos mesmo valores do treino realizado com o SEN12MS.

Este treino tem o objetivo de perceber se a existência de mais trabalho por *epoch* na procura por *features* mais relevantes, tem influência no treino de tal ordem que possibilita a utilização de *datasets* mais pequenos em tarefas deste tipo. O treino teve a duração de aproximadamente **7 horas e 25 minutos**.

No final, os resultados apresentados pelas métricas *accuracy* e *loss* são os seguintes.

- **Varição da Loss:** A *loss* do classificador tinha o valor de 1.561, o valor registado no final foi de 1.328.
- **Accuracy:** A *accuracy* final ronda os **40%**, com início nos 38%.

Este resultado permite-nos concluir que não houve uma evolução significativa ao longo do treino. Os dados antecipam uns resultados de treino que não se categorizam como satisfatórios.

Apesar de executar o treino, os valores registados nas métricas e no tempo de execução mostram os problemas de otimização e recursos abordados anteriormente.

Foram testadas outras configurações de variáveis de treino, porém na grande maioria delas não foram apresentadas vantagens e nas restantes os recursos necessários para concluir o treino eram superiores aos disponíveis.

Daí segue que para a fase de testes apenas será utilizada esta arquitetura.

Testes

Uma vez treinada a rede e apesar de existirem valores de métricas durante o treino, não é fácil saber ao certo o significado desses valores na segmentação de uma imagem que a rede não viu até então.

Por isso, e à semelhança da U-Net, o teste desta rede foi realizado com um conjunto grande de imagens de maneira a obter valores de métricas, como ***accuracy***, mais robustos e fieis à realidade possível

SEN12MS O teste sobre a rede treinado com o conjunto de imagens do *dataset* SEN12MS é como referido suprimido, uma vez que o treino não foi finalizado e por isso não existe uma rede pronta a testar.

GER IMAGERY Seguimos assim para as imagens do *dataset* Ger Imagery, a *accuracy* de teste ronda os 39%.

À primeira vista, os valores muito constantes de *accuracy* registada ao longo do treino e agora na fase de testes são por norma indícios de problemas na inferência na rede. Provavelmente, a rede está a classificar toda a imagem da mesma forma.

Mesmo assim é importante ver se realmente estes valores representam um problema ou se é apenas erro de classificação.



Figura 52: Classificação da rede DSCNN - Exemplo 1.



Figura 53: Classificação da rede DSCNN - Exemplo 2.

Das Figura 52 e Figura 53 é possível perceber que o valor da *accuracy* é baixo uma vez que a rede está a atribuir a mesma classe a todos os pixels da imagem, como esperado. Este problema pode estar relacionado com a baixa quantidade de dados de treino, que não permitem à rede realizar a inferência correta sobre as imagens.

Uma outra causa pode ser a desadequação das variáveis de treino. Porém, como explicado não é possível variar estes valores da forma desejado devido ao custo computacional que a alteração implica.

Terminado o estudo sobre as redes neurais, surge o método não supervisionado C-Means.

4.2 FUZZY C-MEANS CLUSTERING

Como forma de estudar a capacidade que os métodos não supervisionados possuem na classificação SAR, foi selecionado o algoritmo de *Clustering*, **Fuzzy C-Means** (Bezdek et al., 1984).

Este, foi selecionado pelos bons resultados apresentados em Rahman and Islam (2021) e Beevi et al. (2010).

Por se tratar de imagens SAR e por isso, com características diferentes das imagens óticas foi porposto uma adaptação do Fuzzy C-Means, denominado de **Intensity and Location Kernel Fuzzy C-Means (ILKFCM)**. (Xiang et al., 2014)

Este é composto por 3 fases distintas.

A imagem SAR é decomposta em *wavelets* e é-lhe extraída a energia.

A fase seguinte diz respeito ao cálculo o *Fuzzy Factor* com base na intensidade e localização dos pixels da imagem SAR.

Por fim, o *Fuzzy Factor* e a energia da imagem entram no processo de *Clustering* propriamente dito.

O algoritmo em detalhe é descrito de seguida.

4.2.1 Algoritmo

O desenvolvimento de todo o algoritmo foi pensado e baseado na abordagem de Xiang et al. (2014), bem como as fórmulas utilizadas.

Como referido, a primeira etapa do algoritmo consiste na decomposição *wavelet* da imagem SAR. Este tipo de decomposição tem a capacidade de representar *features* ao longo dos eixos x e y .

Na decomposição por *wavelet*, a imagem é decomposta em 4 canais, o primeiro possui informação de baixa-frequência, os seguintes dizem respeito a detalhes da imagem na componente horizontal, vertical e oblíqua. O resultado de uma decomposição de uma imagem exemplo é demonstrado na Figura 54.



Figura 54: Exemplo de decomposição por *wavelet*. (Crédito: Wasilewski (2021))

A energia de cada canal indica a informação dominante da imagem na frequência espacial respetiva, pode ser entendido como textura. Uma janela de 8 x 8 na decomposição tem provado ser uma forma eficiente de representar a textura.(Mallat, 2009)

É também importante notar que é recomendado a utilização de 2 ou 3 níveis de decomposição. Níveis superiores a 1, no que toca a decomposição é realizado sobre a informação de baixa-frequência.

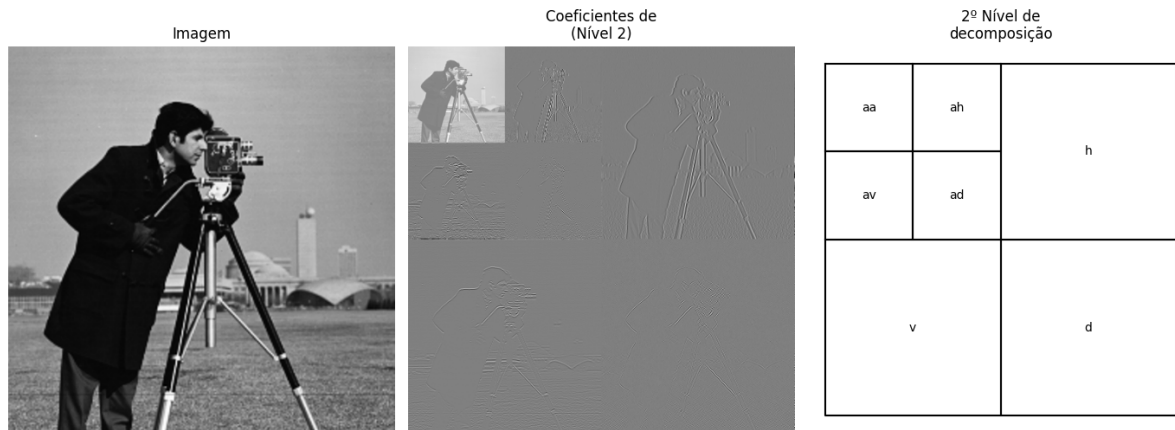


Figura 55: Comparação entre a classificação com inicialização aleatória (esquerda) e classificação ideal (direita). (Crédito: Wasilewski (2021))

A Figura 55 demonstra o resultado da decomposição de nível 2 da imagem do exemplo anterior. É possível ver também um esquema de organização dos canais, que ao invés de 4 (em decomposições de nível 1), são agora 7.

Depois dos 2 níveis de decomposição é extraída a energia de cada um dos 7 canais, segundo a seguinte fórmula.

$$e = \frac{1}{MN} \sum_{m=1}^M \sum_{n=1}^N |x(m, n)| \quad (43)$$

Originando um 7-D array.

A incorporação de um *fuzzy factor* tem o objetivo de superar alguns problemas dos algoritmos C-Means. Este fator tem três características especiais.

- Consideração a localização e intensidade dos pixels vizinhos o que melhora a robustez do algoritmo face a *speckle* e *outliers*.
- No cálculo é utilizada a imagem original evitando assim passos de pré-processamento, que pode causar perda de alguma informação na imagem.
- Existem poucos parâmetros para serem otimizados.

O factor consiste em 2 componentes, um voltado para a distância espacial e outro direcionado para intensidade.

Para cada pixel, o fator de distância espacial é calculado em relação ao pixel central da janela de vizinhança utilizando a distância *Euclidean*.

$$w_{sd} = \frac{1}{d_{ij}^S + 1} \quad (44)$$

Uma vez que a distância espacial não é suficiente para refletir as relações entre os pixels vizinhos. É considerado também a distância em termos de intensidade. Uma vez que a fórmula de distância *Euclidean* não é robusta com ruído multiplicativo, que é o caso das imagens SAR. Será utilizado *relativity ratio* como método de calcular a distância em termos de intensidade entre 2 pixels.

$$w_{id} = 1 - \log(d_{ij}^I) \quad (45)$$

$$d_{ij}^I = \frac{1}{M} \sum_{k=1}^M \frac{I_{N_i}(k)}{I_{N_j}(k)}, \quad I_{N_j}(k) \neq 0 \quad (46)$$

De maneira a calcular o fator é utilizado o logaritmo natural, dado que *ratio relativity* deve ser mapeado para um distancia racional.

O *Fuzzy factor* é calculado com a junção dos dois sub-fatores.

$$w_{ij} = w_{sd} \cdot w_{id} \quad (47)$$

Depois do cálculo das componentes que irão fazer parte do processo, é iniciada a iteração do C-Means. A iteração leva como *input* o *Fuzzy factor* e o *array* de *features* da imagem.

Como é visível na Figura 56, logo após a definição das variáveis como número de *clusters* e fator *Fuzzy*, ocorre a inicialização da matriz *V*. Por norma este passo é feito de forma aleatória, no entanto, devido à natureza do algoritmo C-Means a fraca inicialização dos centróides pode resultar numa fraca classificação de pontos devido à sensibilidade do algoritmo ao local da inicialização.

De maneira a tentar superar este problema, será utilizado uma forma mais inteligente de inicialização. Não só garante melhores resultados de *clustering* como melhora a eficiência do algoritmo que, devido à complexidade das imagens SAR, tende a necessitar de muito trabalho durante as iterações. O processo utilizado na inicialização tem o nome **K-Means++**, proposto por Arthur and Vassilvitskii (2007).

É realizado conforme explicado no Algoritmo 2.

O *Fuzzy factor* calculado anteriormente entra na iteração na forma de matriz *G*. No cálculo desta matriz é considerado o valor de *W* (*fuzzy factor*), a posição dos centróides através da matriz *V* e também a matriz de associação entre cada pixel e cada centróide *U*, através da fórmula.

$$G_{ki} = \sum_{j \in N_i, j \neq i} w_{ij} (1 - U_{kj})^m \|\Phi(p_i) - \Phi(v_k)\|^2 \quad (48)$$

Procedimento 2 K-Means++

```

C ← Número de Clusters
N ← Número de pontos do dataset
data ← Pontos que compõem o dataset
cent0 ← rand()                                ▷ Primeiro centróide é selecionado aleatoriamente.
for c = 1 to C do
  for p = 1 to N do                                ▷ Para cada ponto.
    Calcular a distância do ponto p até ao mais próximo dos centróides anteriores;
  end for
  Selecionar o próximo centróide com base na maior distância aos anteriores;
end for
Output: Array dos centróides

```

Na construção de G é calculada a distancia entre **non-linear maps** que compõem o **feature space** através da simplificação com recurso ao **kernel Gaussian radial basis function**. Assim a distância pode ser reescrita como.

$$\|\Phi(p_i) - \Phi(v_k)\|^2 = 2(1 - K(p_i, V_k)) \quad (49)$$

O *kernel* é definido como

$$K(p_i, V_k) = \exp\left(-\frac{\|p_i - V_k\|^2}{\sigma}\right) \quad (50)$$

Da mesma forma que qualquer algoritmo C-means, o centro do todo a algoritmo consiste na atualização de duas matrizes referentes aos centróides. A matriz U que diz respeito à associação de um determinado ponto, neste caso pixel, ao centróide. E V que nos indica o centróide de cada *cluster*. Estes valores são atualizados através das fórmulas.

$$U_{ki} = \frac{1}{\sum_{j=1}^c \left(\frac{2(1-K(p_i, V_k)) + G_{ki}}{2(1-K(p_i, V_j)) + G_{ji}} \right)^{\frac{1}{m-1}}} \quad (51)$$

$$V_k = \frac{\sum_{i=1}^N (U_{ki}^m K(p_i, V_k) p_i)}{\sum_{i=1}^N (U_{ki}^m K(p_i, V_k))} \quad (52)$$

A função objetivo, que é responsável pela avaliação da evolução do algoritmo, é definida de acordo com o sugerido pelo autor.

$$J_m = \sum_{i=1}^N \sum_{k=1}^c [U_{ki}^m \|\Phi(p_i) - \Phi(v_k)\|^2 + G_{ki}] \quad (53)$$

Tendo em conta a matriz U , o **Kernel de Gauss** e a matriz G , numa tentativa de obter uma função objetivo capaz de envolver e realçar as componentes mais importantes da iteração. Estas variáveis são explicadas em maior detalhe de seguida.

Por fim, este processo iterativo de atualizações é realizado até que seja satisfeita a condição de paragem (Equação (54)) ou o número de iterações atinja o valor máximo previamente definido.

$$\max \left(U^{(i)} - U^{(i+1)} \right) < \epsilon \quad (54)$$

Uma vez concluído o algoritmo, a classificação de cada pixel é extraída da matriz de associação U , através do Max.

$$C_i = \arg_k(\max(U_{ki})) \quad (55)$$

4.2.2 Testes

Devido às suas características, a forma de testar este algoritmo não é a mesma das redes neuronais, a análise dos resultados da classificação tem que ser feita manualmente. A impossibilidade de calcular o desempenho automaticamente deve-se à imprevisibilidade de saber que *cluster* vai corresponder a que classe. Por exemplo, numa determinada imagem as zonas urbanas podem ser representadas pelo *cluster* nº0, noutra imagem pode ser o nº1.

Ao longo desta secção serão apresentadas alguns cenários de teste e respetivos resultados com objetivo de perceber o comportamento do algoritmo, quais os seus pontos fortes e quais as suas fraquezas.

Por questões de simplificação serão apenas expostas algumas imagens dos teste e dos tipos de resultados que o algoritmo está a apresentar.

Para cada uma da categoria de imagens existentes (SEN12MS, Ger Imagery, Tekever), vai ser testado o algoritmo de acordo com 3 configurações que diferem na variável que diz respeito ao número de iterações. Os restantes parâmetros mantêm-se durante as 3 configurações de maneira a isolar o estudo do efeito das iterações no resultado. Assim, os parâmetros utilizados nos 3 testes são os seguintes.

- **Iterações:** 100, 500, 1000;
- **Fuzzy:** 2;
- **Clusters:** 4.

O número do fator de *fuzzy* é 2, por ser o valor recomendado pelo autor e por outros diversos autores em problemas deste tipo.

O número de *clusters* selecionado por defeito foi de 4, na tentativa de induzir o algoritmo a distinguir os 4 tipos de superfície que temos visto até então (**Água, Área Urbana, Campo Agrícola, Área Florestal**).

No entanto, esta divisão pode ser errónea, uma vez que a organização dos *clusters* é feita de acordo com as *features* extraídas, e por isso estas podem levar o modelo a focar-se noutro tipo de detalhes. Por outro lado, não é garantida a existência de todos estes elementos numa imagem.

Ainda assim é um número que faz sentido do ponto de vista das características físicas das zonas referidas.

Tabela 6: Execução do algoritmo C-Means - SEN12MS.

Imagem	Iterações	Duração (min)	J_m inicial	J_m Final
SEN12MS	100/100	16	1,134,162.67	960,234.34
	500/500	79	1,041,583.54	802,618.38
	1000/1000	161	1,124,465.19	768,687.22

De seguida são apresentados os resultados do algoritmo em alguns imagens dos 3 *datasets*, bem como uma breve descrição de como decorreu o processo de segmentação.

A segmentação do *dataset* **SEN12MS**, é representada pela figura Figura 59 e os dados resumo da execução são exibidos na Tabela 6.

Os números da Tabela 6 mostram melhorias, em termos de *Loss*, por configuração de **0.15%**, **0.22%** e **0.31%**, respetivamente.

À medida que as iterações aumentam, a área a verde aumenta também, o que pode ser visto como uma especialização do modelo na identificação deste tipo de características. Em todas as configurações o algoritmo apenas terminou quando foi atingido o número máximo de iterações definido. Estes resultados sugerem que o algoritmo tende de melhorar a sua classificação com aumento das iterações, e ainda não atingiu o seu valor de convergência.

Uma vez que as anotações deste *dataset* não são exatas, como foi possível verificar, não apresenta qualquer vantagem apresentar uma comparação direta entre a máscara e o resultado do algoritmo.

De seguida, vão ser ilustrados os resultados da execução do algoritmo nas imagens do **Ger Imagery**.

Como visto anteriormente, as máscaras que compõem este *dataset*, estão bastante precisas e completas, por esse motivo, faz sentido apresentar as anotações da imagem juntamente com a segmentação realizada pelo algoritmo.

Tabela 7: Execução do algoritmo C-Means - Ger Imagery (Exemplo 1).

Imagem	Iterações	Duração (min)	J_m inicial	J_m Final
Ger Imagery	100/100	17	1,130,483.54	599,936.83
	157/500	27	1,127,114.51	529,799.65
	172/1000	28	1,124,229.48	523,633.52

Na Tabela 7 e na Figura 60, são apresentados os resultado da execução do algoritmo quando fornecida uma das imagens que compõem o *dataset* Ger Imagery. A Tabela 7 mostra que na 2ª e 3ª configuração, o algoritmo convergiu antes de atingir o número de iterações definido, terminaram assim após 157/500 e 172/1000.

A **diferença** no tempo de execução quando o número de iterações aumenta, face ao caso anterior, é bastante **significativa**, e deve-se ao **rápida convergência** nesta imagem, que ocorre por volta da **160ª iteração**.

A função *Loss* apresenta nas diferentes configurações melhorias de **0.46%**, **0.52%** e **0.53%**, respetivamente.

Já que as anotações deste *dataset* apresentam elevada precisão, é interessante perceber se o resultado do algoritmo não supervisionado vai de encontro à real segmentação da imagem. Nesse sentido foi avaliada a *accuracy* da classificação e para as 3 configurações os resultados foram: 0.554 , 0.555 e 0.555, respetivamente.

Neste caso, o aumento das iterações implica uma especialização na classe referente a zona urbana. Daí o valor da *accuracy* acompanhar de certa forma o aumento das iterações.

Por ser a imagem que apresenta melhor resultado em termos de *accuracy*, a imagem representativa da configuração de 500 iterações será utilizada para mostrar o desempenho do algoritmo face a máscara real.

Pelo que mostra a Figura 61, o algoritmo realizou uma segmentação bastante boa da imagem. Com pequenos erros na classificação como água em alguns pixeis, e com terreno agrícola em outros.

Contudo, ao analisar a imagem original, é possível ver que em muitas das zonas que nos locais onde o algoritmo errou, são zonas confusas da imagem onde as 'pegadas' são bastante idênticas e daí a confusão na classificação.

Tabela 8: Execução do algoritmo C-Means - Ger Imagery (Exemplo 2).

Imagem	Iterações	Duração (min)	J_m inicial	J_m Final
Ger Imagery 2	100/100	11	1,143,288.42	851,741.53
	123/500	19	1,139,859.16	666,145.66
	146/1000	21	1,083,744.63	605,670.79

Já a Figura 62, exibe o resultado da classificação de uma imagem das cidades existentes no **Ger Imagery**.

A Tabela 8 exibe a mesma característica identificada na imagem anterior (i.e., a rápida convergência do algoritmo), foi também verificada nesta. No entanto, esta ocorreu ainda mais cedo, o algoritmo estabilizou por volta da iteração **140**, com a configuração de 500 iterações a terminar após a 120ª, e a de 1000 iterações após 146.

No que concerne a variações de *Loss*, para as configurações utilizadas, ocorreram melhorias de **25%**, **42%** e **44%**, respetivamente.

Devido ao facto de ambas as imagens apresentarem valores de convergência entre as 150 e as 170 iterações, e os valores de melhoria em termos de *loss* aparentarem uma estabilização, é previsível que **não haja benefício** no aumento de iterações para números maiores que 200.

A melhoria em termos de **função de loss**, foi inferior à anterior, com uma melhoria de cerca **0.28%**, transitando de um valor inicial de **1,172,756.28** para **841,351.47** no final das cerca de 30 iterações Nesta imagem as *accuracies* registadas foram de 0.355 , 0.553 e 0.545.

Ao analisar a Figura 62, é notória a diferença entre o resultado de 100 iterações e as restantes configurações. Os resultados da *accuracy* praticamente duplicou das 100 iterações para as 500, o que demonstra que o algoritmo conseguiu delinear uma separação entre as zonas verdes.

Tal como no exemplo anterior, a configuração de 500 iterações é a escolhida para a comparação com a máscara real já que apresenta melhor *accuracy*.

A Figura 63 mostra a capacidade do algoritmo de fazer uma boa distinção entre zonas verdes, o que revela robustez no algoritmo. No entanto, e principalmente na parte inferior da imagem o algoritmo teve dificuldade em

distinguir a zona urbana dos terrenos agrícolas. No entanto, e como mostram os números podemos considerar uma **boa classificação** por parte do algoritmo.

Tabela 9: Execução do algoritmo C-Means - Tekever.

Imagem	Iterações	Duração (min)	J_m inicial	J_m Final
Tekever	100/100	23	1,845,285.35	1,327,341.58
	266/500	49	1,906,898.59	1,114,940.03
	312/1000	59	2,267,910.11	1,179,141.57

Os dados e resultados da segmentação da imagem da Tekever é apresentada na Tabela 9 e na Figura 64. Uma vez que na 2ª configuração o algoritmo realiza 266 iterações e na 3ª termina após 312 iterações, segunda a Tabela 9, é possível associar a convergência do algoritmo a valores de iteração próximos dos **iteração nº300**.

De forma semelhante às imagens anteriores, a convergência do algoritmo aponta para uma estabilização nas melhorias. No entanto, em imagens Tekever, o algoritmo demonstra maior dificuldade em convergir, já que realiza, sensivelmente, **2x mais iterações que os exemplos do Ger Imagery**.

As melhorias de *Loss*, neste caso, apontam melhorias de **0.28%**, **0.41%** e **0.48%**, nas 3 configurações testadas.

Desta análise segue que o aumento das iterações para números superiores a 300 **não traz qualquer benefício à classificação**.

O aumento progressivo de iterações permitiu ao algoritmo aperfeiçoar as associações feitas, apesar de não existir diferenças significativas nas imagens, é possível verificar uma diminuição da área a azul.

Devido à inexistência de máscaras nas imagens fornecidas pela Tekever, a única forma de avaliar o desempenho da segmentação é através da análise visual das imagens.

Uma das características mais visíveis na Figura 59 e na Figura 64 é a presença de pequenas secções de azul numa parte da imagem, onde geograficamente não existe água. Uma das causas possíveis para este atribuição de classe deve-se a uma das características da imagem SAR, falada anteriormente, que é o *shadowing*, que ocorre quando a imagem sofre o efeito de sombra. Existe também a possibilidade de o azul se tratar de zonas em que as características do eco da radiação são equivalente às de água, como estradas ou até terrenos inundados.

3 CLUSTERS Uma vez que não existe uma relação direta entre as *labels* dos *datasets* e o número de *clusters*, sendo a coloração feita em uniformidade com as anotações apenas por uma questão de simplificação na visualização, faz todo sentido executar o algoritmo com diferentes números de *clusters*

O primeiro teste com diferente número de *clusters* foi executado com $c = 3$ (3 *clusters*). Este número surge na tentativa de perceber se existe a possibilidade de algum dos *clusters* dos testes anteriores pode ser suprimido, e que tipo de associações são feitas quando as categorias diminuem. O resultado ideal do ponto de vista empírico seria a fusão das classes **floresta** e **zona agrícola**, numa classe mais abstrata **zona verde**.

Por outro lado, é estudada a influência da diminuição do número de centróides no tempo de execução do algoritmo.

Assim, as imagens utilizadas nos testes anteriores são utilizadas neste teste, de maneira a manter o processo de análise mais consistente e fácil de efetuar. O número de iterações utilizado foi de 500, por ser um valor que se revelou suficiente para a convergência na maioria dos casos anteriores.

Iniciando o processo de análise deste teste pelos tempos de execução é de notar uma significativa melhoria.

A imagem do *dataset* **SEN12MS**, que no teste anterior teve uma duração de 80 minutos aproximadamente, viu agora o seu tempo reduzido em 22% passando a executar a segmentação em **61 minutos**.

No que diz respeito às imagens do *dataset* **Ger Imagery** é possível verificar uma melhoria bastante mais significativa, entre os 45-50%, com tempo de execução de **14 minutos** e **15 minutos**, respetivamente.

Já a segmentação da imagem da **Tekever** teve uma duração de 42 minutos, que representa uma melhoria em **7 minutos**, face ao teste com 4 centróides.

Na Figura 65 estão representados os resultados do C-Means quando o número de centróides definidos é 3. Com exceção da Figura 65(c), em que a distinção entre floresta e terreno agrícola é clara para o algoritmo, é possível ver a desmaterialização da classe floresta a convergir em Zona Verde com a classe Zona Agrícola. Existiram também alguns pixels em que a associação foi mais forte à classe Zona Urbana, no entanto fenómeno raro.

Como esperado, grande parte em grande parte das situações o algoritmo conseguiu 'agrupar' os pixels referentes às classes de zona verde.

No caso da imagem SAR, Figura 65(d), a situação difere um pouco das restantes, a remoção de um *cluster* não trouxe qualquer vantagens com exceção do tempo de execução, uma vez que esta alteração resultou num maior número de erros, principalmente na classe que foi classificada como água. De uma forma geral, é vantajoso o teste com 3 *clusters*, a redução no tempo de execução compensa as pequenas inconsistências na classificação.

5 CLUSTERS Da mesma forma que faz sentido o estudo da diminuição do número de *clusters*, é também interessante considerar a possibilidade de o algoritmo conseguir encontrar mais classes com significado.

Dessa forma, foi analisado o resultado do algoritmo quando o número de centróides é 5, mais 1 que o utilizado de forma *default*.

À priori, é esperada uma classe que define melhor separação entre as classes que representam zona verde (**zona agrícola e floresta**), dado que o eco, nas imagens SAR, varia com a canóia dos elementos (i.e., a imagem SAR varia consoante a altura e constituição das árvores e arbustos).

De forma análoga aos testes anteriores, para garantir consistência nos resultados, as imagens e as variáveis do algoritmo são mantidas (à exceção do número de *clusters*, obviamente).

O tempo de execução do algoritmo **aumentou significativamente** em todas as imagens face ao exemplo base de 4 centróides.

Com um **aumento de 20 minutos** no caso da imagem **SEN**, que teve a duração de **100 minutos**.

Já as imagens do *dataset* **Ger Imagery** tiveram uma duração de aproximadamente **mais 10 minutos**, contando agora com tempos de execução de **37 minutos** e **27 minutos**, respetivamente.

Por fim a imagem **Tekever** foi a que registou um maior aumento. A duração da classificação teve muito próxima de **2x a duração no teste *default***. A segmentação desta imagem teve a duração de cerca de **90 minutos**.

A nova classe é exibida em todas as imagens da Figura 66 pela cor cinzento que está legendado como desconhecido, isto porque da análise da imagem original e da máscara da imagem (quando relevante) não surge a identificação clara do que está o algoritmo a considerar como nova classe.

Os resultados inconclusivos aliados ao aumento do tempo de execução do algoritmo evidenciam a inutilidade de prosseguir a classificação do algoritmo com 5 ou mais *clusters*.

O algoritmo C-Means apresentou resultados bastantes bons, o que demonstra a elevada capacidade de segmentação por parte dos algoritmos não supervisionados, em particular o utilizado.

O facto de ser não-supervisionado apresenta algumas desvantagens, como a imprevisibilidade e inconsistência na classificação de imagens.

Assim, seria extremamente pertinente a procura por um algoritmo semi-supervisionado baseado no C-Means que permita relacionar a classificação do algoritmo com uma anotação da imagem e assim tornar a segmentação mais robusta.

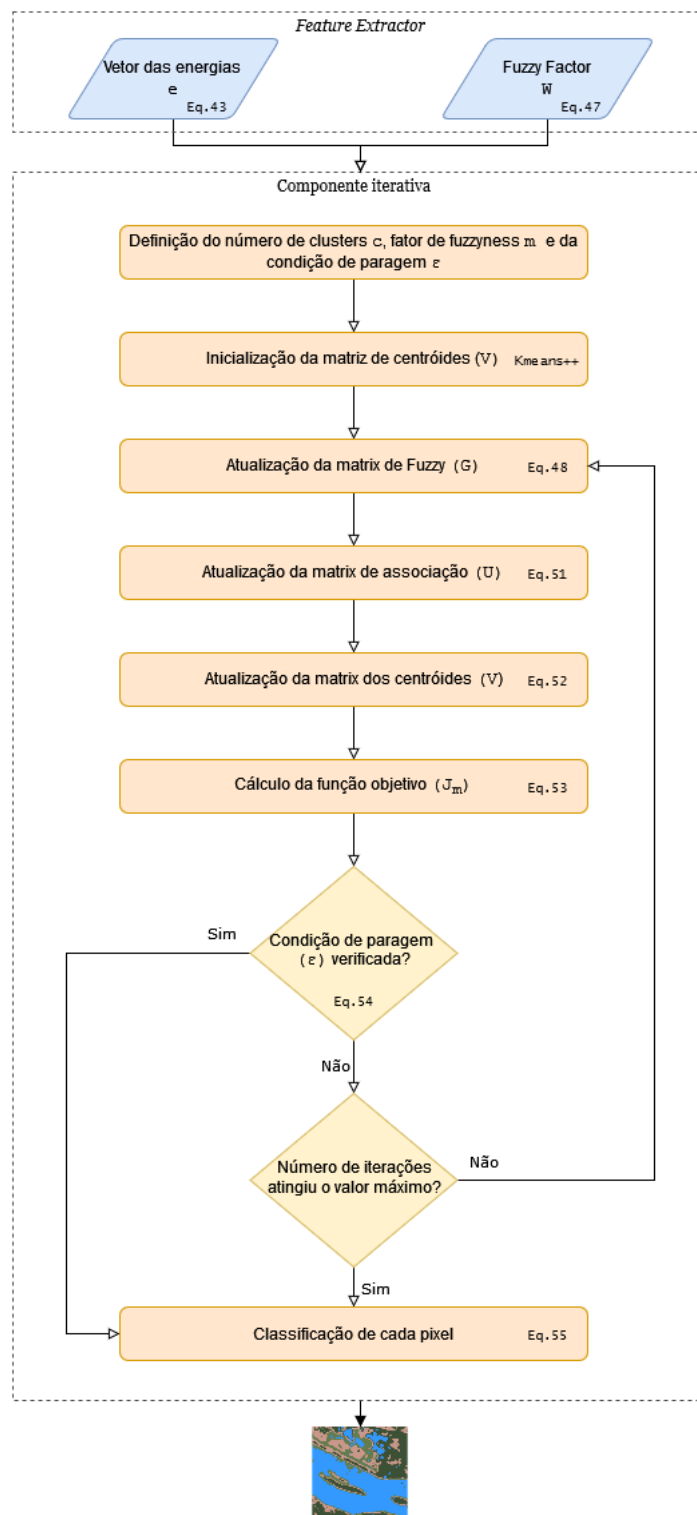


Figura 56: Diagrama representativo da parte iterativa do algoritmo C-Means.

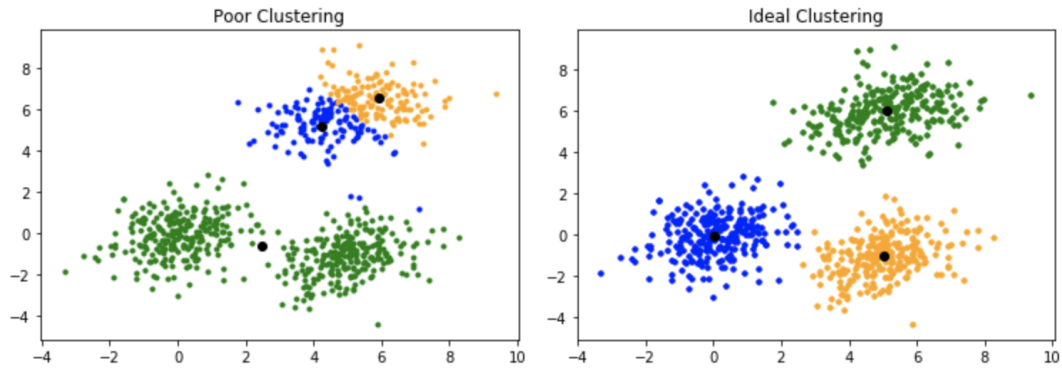
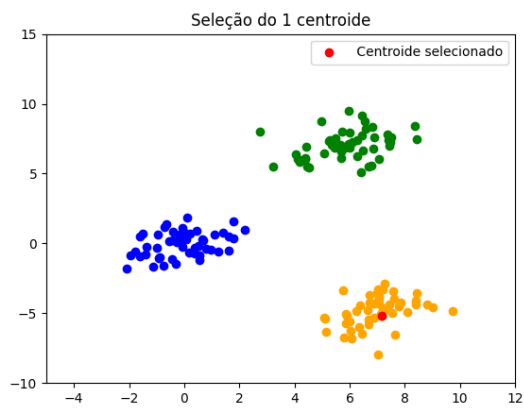
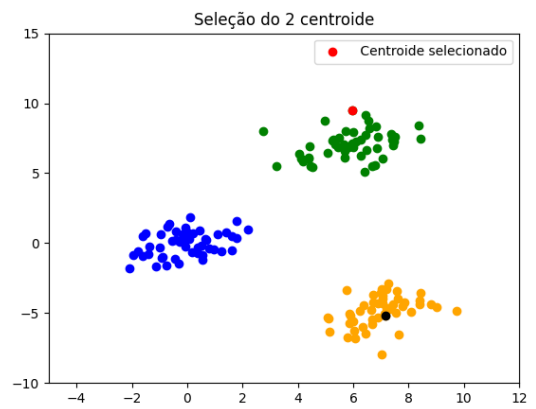


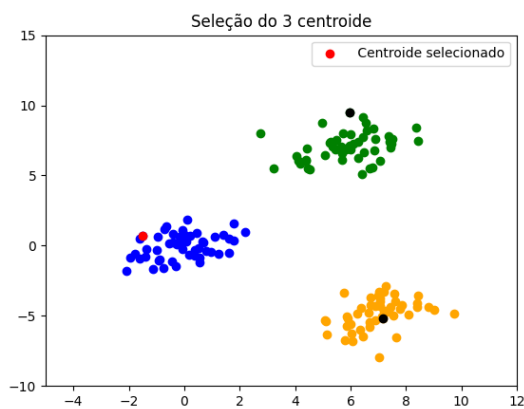
Figura 57: Comparação entre a classificação com inicialização aleatória (esquerda) e classificação ideal (direita) (Crédito: [savyakhosla \(2021\)](#))



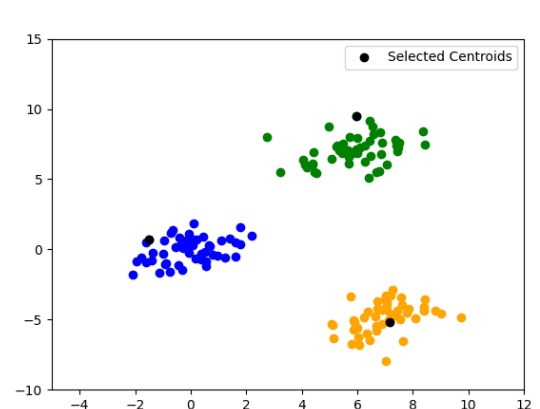
(a) Seleção do **primeiro** Centróide.



(b) Seleção do **segundo** Centróide.



(c) Seleção do **terceiro** Centróide.



(d) Resultado da seleção dos *clusters* de acordo com o algoritmo **Kmeans++**.

Figura 58: Processo de seleção dos centróides em dados exemplo. (Crédito: [savyakhosla \(2021\)](#))

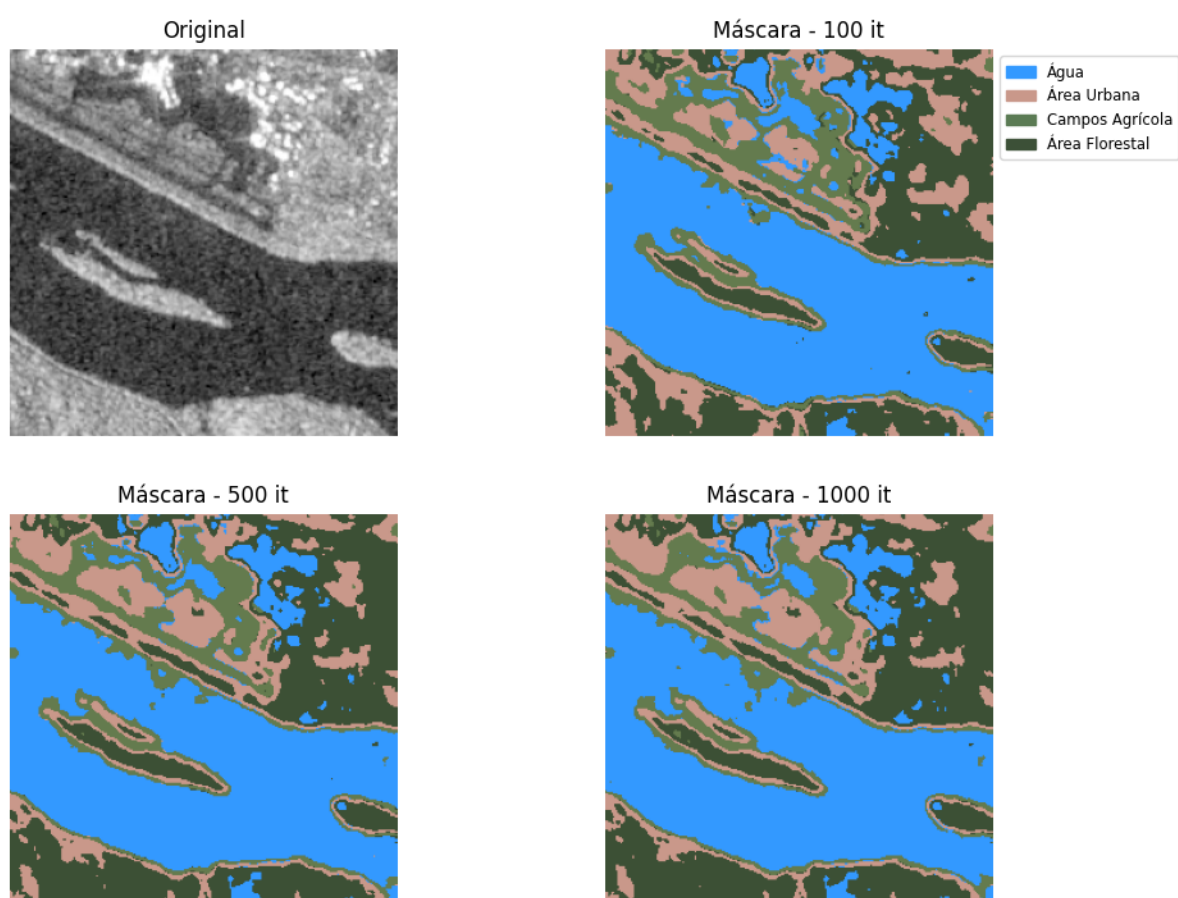


Figura 59: Resultado do **C-Means** numa imagem do *dataset* SEN12MS.

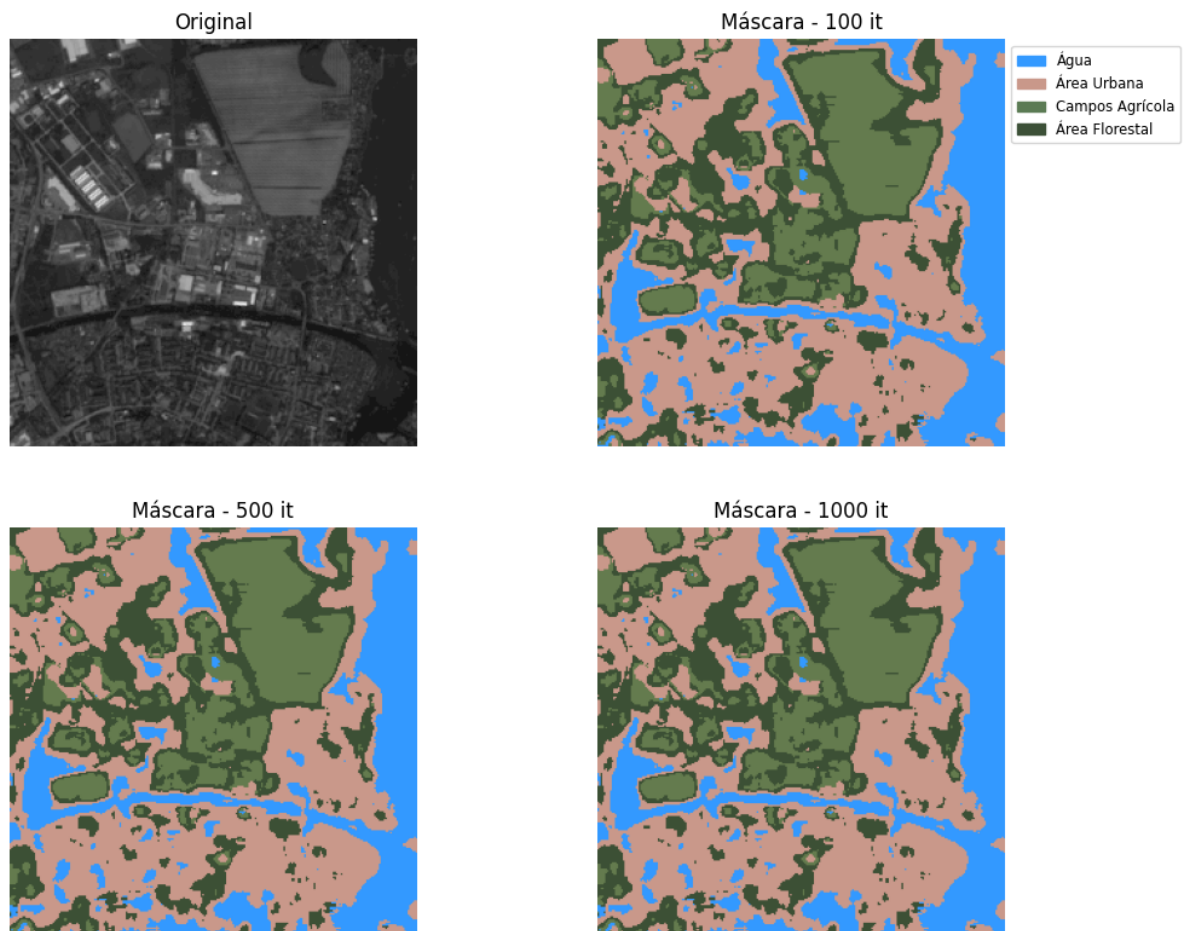


Figura 60: **C-Means** aplicado a uma imagem pertencente ao *dataset Ger Imagery* - Exemplo 1.

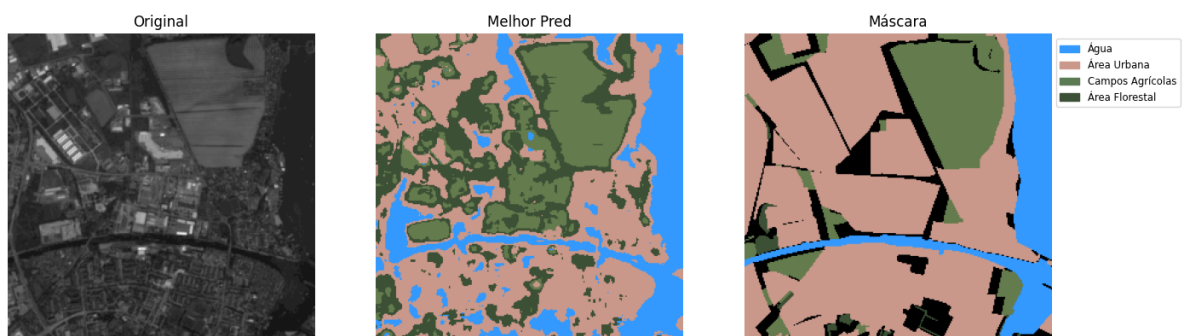


Figura 61: Comparação entre a **Imagem original**, **classificação C-Means** e a **Máscara real** - Ger Imagery (1).

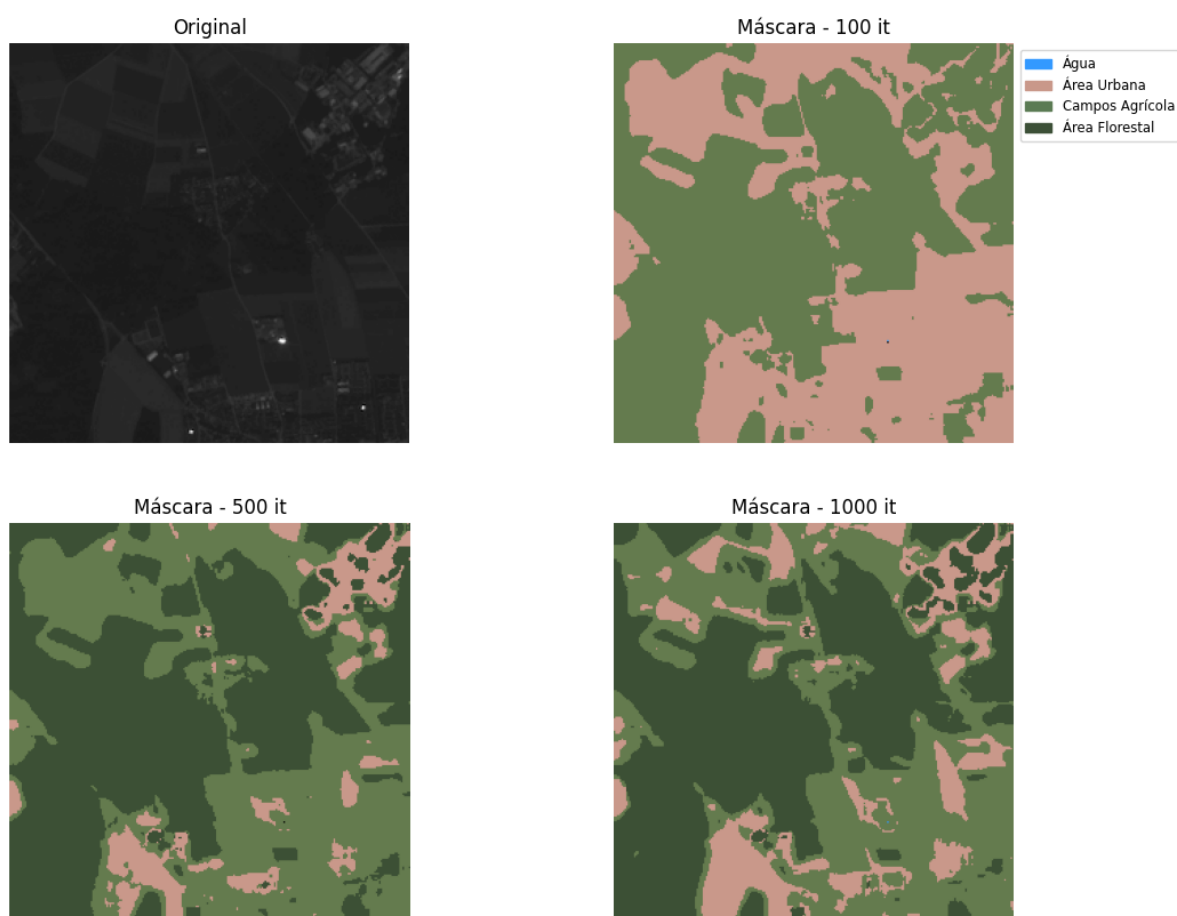


Figura 62: Resultado do **C-Means** numa imagem do *dataset Ger Imagery* - Exemplo 2.

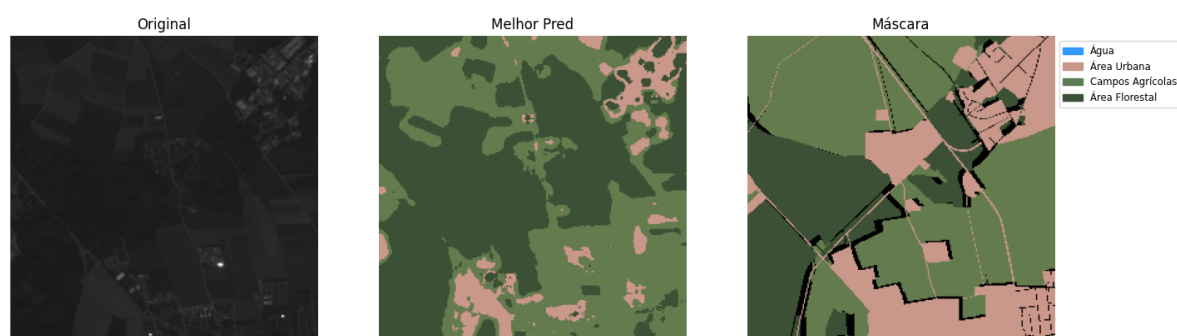


Figura 63: Comparação entre a **Imagem original**, **classificação C-Means** e a **Máscara real** - Ger Imagery (2).

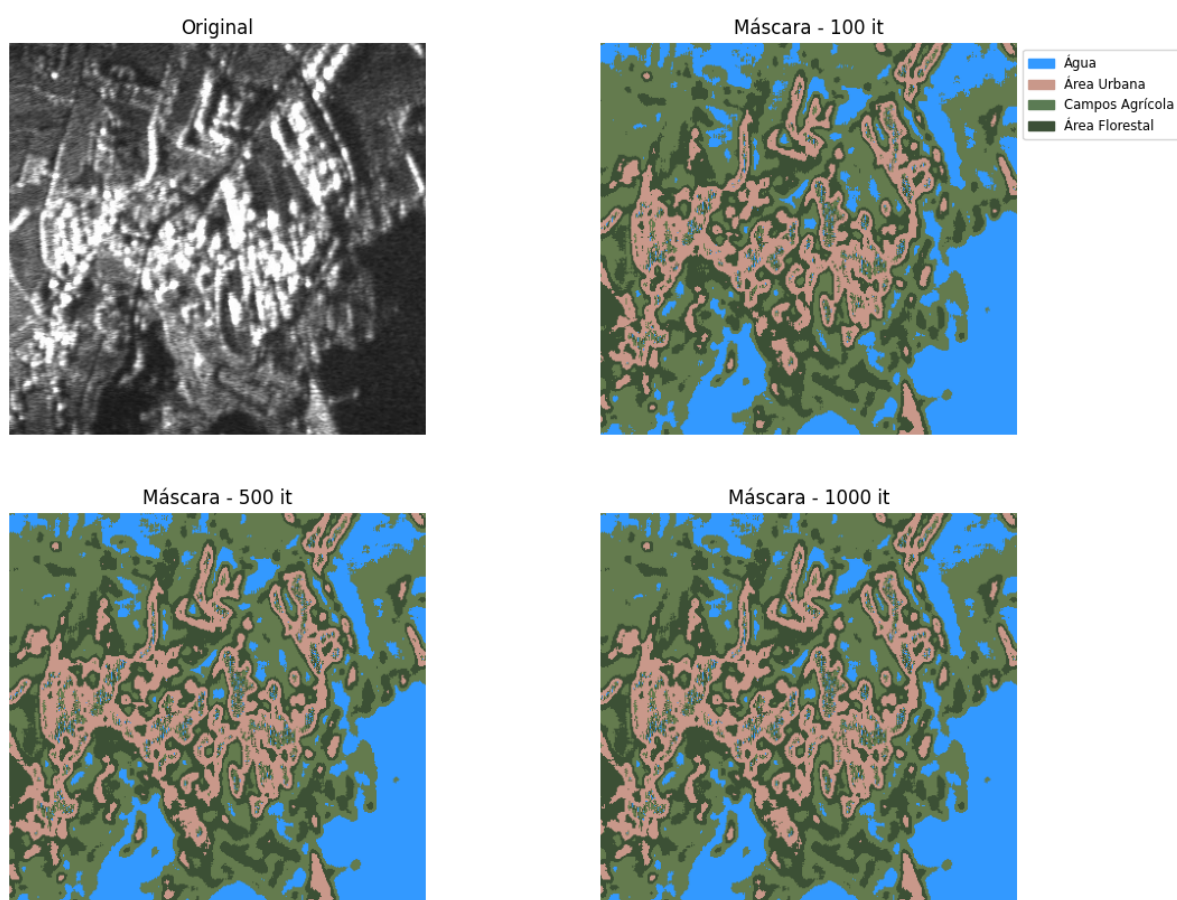
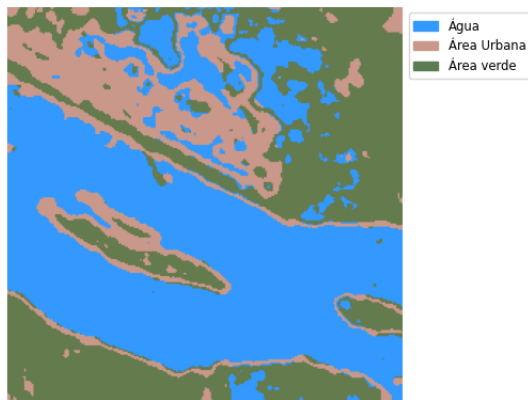
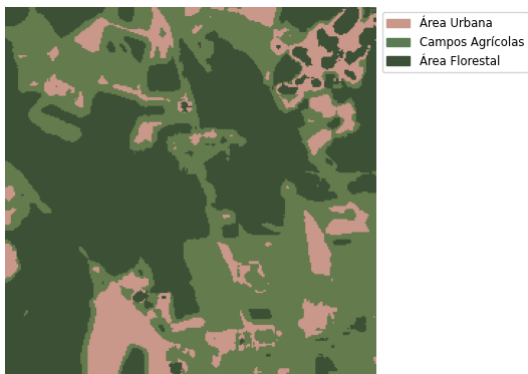
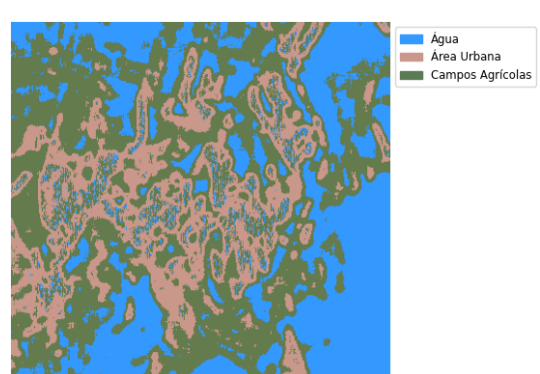
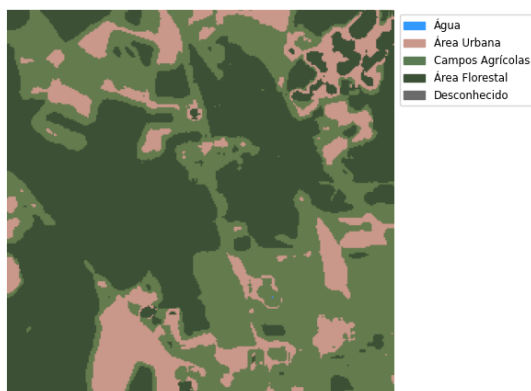
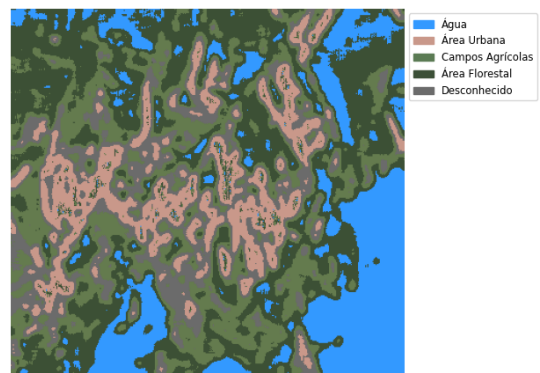


Figura 64: Resultado do **C-Means** numa imagem fornecida pela **Tekever**.

(a) Imagem do *dataset* SEN12MS.(b) Imagem do *dataset* Ger_Imagery.(c) Imagem do *dataset* Ger_Imagery.

(d) Imagem Tekever.

Figura 65: Resultado do algoritmo C-Means com $c = 3$

(a) Imagem do *dataset* SEN12MS.(b) Imagem do *dataset* Ger Imagery.(c) Imagem do *dataset* Ger Imagery.

(d) Imagem Tekever.

Figura 66: Resultado do algoritmo C-Means com $c = 5$

4.3 SUMÁRIO

Ao longo deste capítulo foram propostos alguns métodos para resolver o 2º problema identificado, a interpretação de uma imagem SAR.

A primeira solução, a U-Net, foi escolhida por ser uma das redes mais utilizadas em problemas de segmentação. No final do processo de treino é visível o potencial desta rede. Mesmo sendo imagens SAR, que são bastante diferentes das imagens normais, a rede obteve um bom desempenho, sendo apenas relatado um problema, que pode melhor definido como uma característica, que é a adaptação da rede ao *dataset* de treino. Esta característica piora o desempenho da rede quando utilizadas imagens diferentes das do *dataset* de treino. Esta contrariedade impulsiona a procura por uma forma de extrair *features* mais relevantes que potenciem uma classificação mais robusta de imagens.

A solução DSCNN aparece com este potencial, através da utilização de uma estrutura de AE's encadeados. No entanto, não foi possível perceber se todo o potencial da rede seria alcançado, uma vez que a rede construída é exigente do ponto de vista computacional, não permitindo finalizar o treino de algumas configurações. Nas configurações que permitiram a execução do treino, os resultados revelaram a incapacidade da rede distinguir entre classes nas imagens.

Por fim, como forma de estudar a capacidade dos métodos não-supervisionados de realizar a segmentação de imagens, neste caso em particular, imagens SAR. Foi assim desenvolvido o método C-Means com cálculos sobre a energia da imagem através da decomposição por *wavelets*. Os resultados foram bons, o que realça a enorme capacidade que o algoritmo possui de agrupar de forma correta segmentos da imagem com características idênticas. A vantagem desta abordagem, face às anteriores, reside no facto de não haver uma adaptação a um certo tipo de imagem.

Tabela 10: Resumo dos testes dos algoritmos de segmentação

Algoritmo	Dataset Treino	Teste	Resultado	Figura
U-Net	SEN12MS	SEN12MS	Positivo	Figura 38
		Ger Imagery	Negativo	Figura 40
		Tekever	Negativo	Figura 41
	Ger Imagery	Tekever - com correção	Positivo	Figura 45
		SEN12MS	Negativo	Figura 48
		Ger Imagery	Positivo	Figura 46
		Tekever	Negativo	Figura 49
DSCNN	SEN12MS Treino não Concluído	-	-	-
	Ger Imagery	Ger Imagery	Negativo	Figura 52
Fuzzy C-Means Clustering	-	SEN12MS	Positivo	Figura 59
	-	Ger Imagery	Positivo	Figura 61
	-	Tekever	Positivo	Figura 64

Parte III

INTERPRETAÇÃO E CONSIDERAÇÕES FINAIS

DISCUSSÃO E ANÁLISE DE RESULTADOS

Nesta secção serão abordados os resultados obtidos ao longo da dissertação, com foco nas respostas encontradas aos problemas identificados. Para cada problema, serão analisadas todas as alternativas encontradas para o resolver.

Esta dissertação teve como objetivos responder a duas necessidade encontradas na utilização de SAR.

1. Registo geográfico completamente manual;
2. Interpretação difícil da imagem.

O primeiro problema que a dissertação procura colmatar é a necessidade das imagens SAR serem registadas por um ser humano.

A análise feita aos resultados sugere que o comparador ASIFT quando utilizado em imagens SAR-ótica não apresenta capacidade de realizar o *matching* correto entre as imagens e por consequência o correto registo da imagem SAR.

Apesar de no primeiro teste (Figura 23) o resultado da utilização deste comparador ser animador, os restantes testes revelam muitas inconsistências nos descritores na etapa de *matching*. Mesmo com alterações que potenciam a correta identificação de descritores, como a alteração de tamanho da imagem (Figura 26) e a alteração de filtro (Figura 27) para a diminuição do número de *keypoints* e por consequência reduzir o número de falsos *matches*, os testes realizados chumba a possibilidade de utilização do ASIFT, nestas condições, tal como foi testado no processo de registo. Os resultados divergem dos relatados em Markiewicz et al. (2019), em que foi apresentado um método com recurso a um comparador ótico capaz de enquadrar de forma confiável as imagens. Esta divergência deve-se, provavelmente, às diferenças entre as imagens SAR utilizadas pelo autor em Markiewicz et al. (2019) que apresentam um grau elevado de resolução e pré-processamento, face a imagens com uma resolução significativamente inferior e com métodos de pré-processamento mais rudimentar que foram utilizadas nos testes realizados. Esta disparidade na resolução aliada à conhecida diferença entre imagens SAR e imagens óticas tornam o processo inapto a ser utilizado numa ferramenta de geo-localização.

Estes resultados permitem inferir que a comparação entre imagens óticas e imagens SAR é um processo delicado, de imensa complexidade e que no caso de não existirem condições ótimas para a comparação, esta deve ser evitada.

Por outro lado, os dados obtidos da alteração do comparador para SAR-SAR sugerem que este é capaz de realizar de forma eficaz o *matching* de imagens, o que se traduz num registo fidedigno.

Logo no primeiro teste realizado (Figura 29) torna-se clara a capacidade que o comparador possui de encontrar semelhanças entre as imagens. A robustez do algoritmo é reforçada pela Fig30, onde é possível ver que mesmo com imagens de diferentes orientações são encontrados os pontos de semelhança. As restantes figuras exibem o constante desempenho das comparações em imagens com diferentes características.

Estes resultados confirmam os resultados de Dellinger et al. (2012), que mostram que com alterações no algoritmo SIFT, nomeadamente no espaço de identificação de *keypoints*, algumas particularidades das imagens SAR são tidas em conta de forma a tornar este comparador uma opção robusta para a tarefa de comparação de imagens SAR.

A eficácia do comparador permite a criação de uma ferramenta capaz de reduzir o trabalho humano no processo de registo geográfico das imagens SAR.

Os dois processos desenvolvidos e testados apresentam diferenças significativas na sua forma de atuação. A transição de comparador SAR-Ótico para um comparador SAR-SAR foi justificada pela incapacidade que o primeiro apresentou em realizar de forma eficaz a comparação entre as imagens testadas e provenientes de diferentes sensores.

A mudança de paradigma de comparação não deve ser interpretada como um abandono à metodologia proposta inicialmente, mas sim como um ponto de partida para um estudo mais aprofundado acerca da temática. Já em relação ao método SAR-Harris que apresentou capacidade de executar a tarefa, o foco deve ser a otimização do algoritmo.

A segunda questão à qual a investigação procura encontrar resposta remete à interpretação das imagens SAR. Representa assim um problema de classificação e segmentação.

Os testes efetuados demonstram que os métodos tradicionais de classificação, as redes neurais, evidenciam a capacidade de realizar a segmentação em imagens SAR com eficácia semelhante à encontrada quando são utilizadas imagens óticas. Com *accuracies* acima de 70% nos respetivos *datasets* de treino, a rede U-Net proposta por Ronneberger et al. (2015), apresenta um enorme potencial nesta tarefa.

À semelhança dos testes realizados em Emek and Demir (2020), os resultados desta rede nesta investigação foram bastante positivos.

Possui apenas uma limitação, comum nas rede neurais, que é a adaptação aos dados de treino. Este problema é visível quando a imagem SAR é testada com imagens de diferentes conjuntos de dados como na Figura 41 e na Figura 49 em que os tons mais escuros da imagem forçam a classificação idêntica na grande maioria da imagem. Para sustentar esta conclusão foi realizada uma experiência envolvendo a aproximação da tonalidades. Na Figura 44, é possível verificar que a imagem em questão sofre um aumento progressivo do brilho, através da operação *gamma correction*. A classificação da rede para estas imagens é exibido na Figura 45, que mostra um aumento progressivo de áreas com classificação correta, que no limite do brilho volta a diminuir.

Como forma de tentar diminuir a influência do problema anterior no problema de classificação foi desenvolvida uma rede neuronal com características diferentes e com bons resultados, a rede DSCNN de Geng et al. (2017). Com limitações em termos de recursos para efetuar o treino desta rede, muitas das configurações de treino não foram finalizadas, não podendo haver uma total comparação com o resultado obtido pelos autores. No entanto

da análise possível da rede DSCNN, esta não acompanha, em certa medida, o potencial atribuído em [Geng et al. \(2017\)](#) e posteriormente em [Zhu et al. \(2020\)](#), uma vez que os resultados obtidos demonstraram a incapacidade da rede de realizar a tarefa com qualidade.

Para ser possível ter a percepção da utilidade desta rede, a utilização de uma plataforma de treino com bons recursos computacionais, bem como uma otimização da rede e dos seus processos são sugeridos.

Os resultados do estudo sobre os algoritmo não supervisionados, em particular o C-Means, compravam que este possui a capacidade de realizar a classificação e segmentação de uma imagem SAR. Como demonstrado em [Xiang et al. \(2014\)](#). Todos os exemplos em que foi testado o algoritmo corresponderam de forma positiva (Figura 59, Figura 61, Figura 63, Figura 64), mesmo com a alteração do número de *clusters* (Figura 65, Figura 66). Nos exemplos das imagens do *dataset* Ger Imagery (Figura 61, Figura 63), é possível realizar uma análise de *accuracy* das classificações, e com valores a ronda os 54%, o resultado é positivo. Diferente dos 70% da U-Net, no entanto, é considerado satisfatório por se tratar de um método não-supervisionado.

Por responder bem e de forma consistente nas diversas situações, o C-Means é uma alternativa viável para o objetivo.

Porém, o facto de ser um método não supervisionado apresenta duas limitações, a sua imprevisibilidade de resultados e o tempo de execução quando comparado com uma rede neuronal previamente treinada. Assim, o desenvolvimento de um método semi-supervisionado baseado no C-Means e que seja capaz de suprimir a imprevisibilidade do método aliado a uma otimização dos processos, transformam este algoritmo num *go to* nos problemas de classificação e segmentação

Em suma, o algoritmo SAR-Harris de comparação de imagens SAR-SAR é neste contexto o algoritmo mais capaz de responder ao problema de registo geográfico de imagens SAR.

Já no que concerne ao problema de segmentação, no final desta dissertação é possível encontrar duas formas muito diferentes de classificar uma imagem SAR.

Se por um lado a rede neuronal U-Net apresentou bons resultados quando testada sobre as imagens de características semelhantes àquelas encontradas durante o seu treino.

Por outro, o método não supervisionado revelou ser de uma versatilidade extraordinária, capaz de classificar de forma competente qualquer tipo de imagem SAR. No final tudo se resume à utilização desejada, se for necessária uma ferramenta capaz de classificar um tipo específico de imagem com uma eficácia excelente, é recomendado o uso da rede neuronal U-Net, desde que treinada com um *dataset* de qualidade.

No caso de ser necessária uma ferramenta capaz de classificar qualquer tipo de imagem, sem qualquer problema no prejuízo de precisão, o método C-Means é um sério algoritmo a considerar.

É importante destacar também, que como foi visto ao longo do projeto, as imagens SAR diferem de acordo com as características de aquisição do sensor, ou seja pequenas alterações no comprimento de onda pode resultar em imagens SAR completamente diferentes. Esta característica influencia os resultados da dissertação e também os resultados em testes futuros.

No entanto, não foi possível a aproximação das imagens de teste uma vez que o sensor da Tekever é proprietário.

Por motivos lógicos, a aproximação das imagens de treino e teste resultam em melhores algoritmos e melhores resultados.

CONCLUSÕES E TRABALHO FUTURO

Neste secção será realizada a análise de todo este caminho percorrido ao longo deste projeto até então. Será feito um sumário da dissertação de maneira a fornecer uma vista geral sobre o projeto e o seu desenvolvimento. De seguida, será realizada uma análise e reflexão acerca do trabalho desenvolvido face aos problemas levantados e objetivos propostos. Por fim, é importante abordar as limitações encontradas bem como novas ideias e caminhos interessantes para pesquisa e desenvolvimento futuros.

Foram encontrados dois problemas que influenciam a utilização das imagens capturadas por um radar do tipo SAR. O SAR é um sensor ativo, já que as suas imagens são resultado do eco da radiação que o próprio irradia.

Esta particularidade, aliada às diferentes configurações da radiação emitida pelo sensor tornam as imagens de um SAR bastante mais complexas que as imagens óticas convencionais.

O primeiro problema identificado está relacionado com o registo geográfico das imagens. A segunda questão levantada tem o foco voltado para a dificuldade de interpretação da imagem.

Para responder à questão relacionada com o registo geográfico de uma forma automática, foram procuradas alternativas no sentido de comparar imagens previamente registadas com as imagens a registar. Para esse efeito foram estudadas duas formas de comparação, uma baseada em comparação SAR-Ótica (ASIFT) e uma outra suportada em comparação SAR-SAR (SAR-Harris). De notar, que a utilização do SAR-Harris representa maior trabalho para o utilizador. Os resultados indicam que a utilização de um método que recorre à comparação de imagens com as mesmas características (SAR-SAR), tende a obter melhores resultados que a utilização de um comparador SAR-ótico, para o efeito.

O *trade-off* entre maior trabalho do utilizador e qualidade de registo, justifica a opção de comparação SAR-SAR com registo manual de uma imagem referência.

Assim, relativamente ao objetivo 1, é possível dizer que não foi atingido na sua máxima extensão, isto porque não foi possível até então encontrar uma solução que permita um registo totalmente independente de um utilizador. Ainda assim, é possível atribuir o objetivo como alcançado já que o trabalho humano fica reduzido significativamente com a utilização de uma ferramenta de registo que pode ser construída com base na comparação entre imagens SAR.

A segunda questão, interpretação da imagem, representa um problema de segmentação. Do estudo realizado surge a possibilidade de utilização de dois métodos de segmentação, já que os resultados mostram que tanto

a rede U-Net, como o método não supervisionado C-Means tem potencial para realizar de forma competente a tarefa desejada.

No que diz respeito ao objetivo 2, este foi atingido com sucesso, pelo facto de terem sido encontradas evidências que confirmam a literatura e a possibilidade de realizar a segmentação através dos algoritmos abordados, facilitando assim a interpretação das imagens SAR.

Ao longo do processo de desenvolvimento da dissertação surgiram algumas dificuldades. A primeira limitação encontrada no desenvolvimento do projeto foi a identificação dos problemas a resolver e definição dos requisitos. Contudo, após algumas reuniões e negociações esta adversidade foi ultrapassada.

Como foi antevisto, a quantidade e qualidade dos dados revelou-se um problema. Por serem imagens com características especiais não é fácil encontrar ou até construir um *dataset* capaz de alimentar na quantidade correta modelos de *Deep Learning*. E quando existem *datasets* disponíveis, as diferenças entre as imagens dos diferentes *datasets* (devido às características de aquisição) prejudicam os resultados.

No entanto, o maior obstáculo encontrado no desenvolvimento do projeto é a complexidade da temática em estudo. A constante leitura de artigos e publicações cada vez mais específicas e detalhadas de temas que até então nunca foram vistos ou referidos, prejudicou o decorrer normal do projeto.

Por outro lado, todos estes obstáculos tiveram um significativo impacto em termos de aprendizagem. O estudo e a necessidade de adaptação a problemas encontrados permitem o desenvolvimento de qualidades pessoais que não se desenvolvem de outra forma.

Como em qualquer área científica, a pesquisa e desenvolvimento não vê fim, este caso não é exceção.

As tecnologias e arquiteturas testadas são encaradas como um apontador para o caminho a seguir. Nesse sentido há alguns pontos que devem ser melhorados numa abordagem futura a este projeto.

Um dos primeiros pontos a sugerir na continuidade desta investigação é a utilização de dados reais, em quantidades expressivas e anotações perfeitamente detalhadas que permitam um desenvolvimento de ferramentas fieis à realidade e com um resultado claro.

O processamento de imagens SAR tem um peso expressivo em tudo o que é feito com as mesmas. Dessa forma e embora não fazendo parte do espectro deste projeto seria extremamente relevante uma investigação profunda acerca desta temática.

Uma vez que algumas destas ferramentas são para uso de utilizador, a preparação para *deploy* de algumas ferramentas para utilização em testes com interação humana é recomendada.

Em cada problema identificado nesta dissertação, ficaram por desenvolver algumas sugestões de solução. Em particular, o desenvolvimento da DSCNN de maneira a efetivamente testar a arquitetura é sugerido. O desenvolvimento e teste dessas propostas seriam um ótimo contributo para a comunidade científica o rumo da investigação no tema.

É importante também num projeto deste calibre uma adequação mais personalizada das ferramentas ao tipo de dados para o qual foram desenhadas. As ferramentas desenvolvidas beneficiam de uma otimização para que o seu desempenho quer em termos de velocidade de execução quer em resultados, alcançando todo o seu potencial.

Em suma, ambos os objetivos desta dissertação foram atingidos com sucesso. É importante destacar a enorme diferença entre os conteúdos aqui estudados e aqueles que foram abordados até então, e a necessidade de alguns melhoramentos para obter novos e mais completos resultados.

Por fim, dissertação oferece mais do que ferramentas para um determinado objetivo, apresenta vários *insights* acerca das possíveis abordagens a seguir nos problemas de registo e segmentação de imagens SAR.

BIBLIOGRAFIA

- David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. volume 8, pages 1027–1035, 01 2007. doi: 10.1145/1283383.1283494.
- Reza Bahmanyar, Daniela Espinoza-Molina, and Mihai Datcu. Multisensor earth observation image classification based on a multimodal latent dirichlet allocation model. *IEEE Geoscience and Remote Sensing Letters*, 15: 459–463, 2018.
- S. Zulaikha Beevi, M. Mohammed Sathik, and K. Senthamaraikannan. A robust fuzzy clustering technique with spatial neighborhood information for effective medical image segmentation: An efficient variants of fuzzy clustering technique with spatial information for effective noisy medical image segmentation. *2010 Second International conference on Computing, Communication and Networking Technologies*, pages 1–8, 2010.
- James C. Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers Geosciences*, 10(2):191–203, 1984. ISSN 0098-3004. doi: [https://doi.org/10.1016/0098-3004\(84\)90020-7](https://doi.org/10.1016/0098-3004(84)90020-7). URL <https://www.sciencedirect.com/science/article/pii/0098300484900207>.
- Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. URL <http://leon.bottou.org/papers/bottou-98x>. revised, oct 2012.
- Jason Brown. Sar 101: An introduction to synthetic aperture radar, Feb 2020. URL <https://www.capellaspace.com/sar-101-an-introduction-to-synthetic-aperture-radar/>.
- Business Wire. Synthetic aperture radar (sar) market - roadmap for recovery from covid-19: Rising preference for the integrated c4isr to boost market growth: Technavio, Sep 2020. URL <https://www.businesswire.com/news/home/20200928005278/en/Synthetic-Aperture-Radar-SAR-Market>.
- Kendra Cherry. How different psychologists have evaluated intelligence, Oct 2019. URL <https://www.verywellmind.com/theories-of-intelligence-2795035>.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

- comparecamp. Tensorflow review: Pricing, pros, cons & features, May 2019. URL <https://comparecamp.com/tensorflow-review-pricing-pros-cons-features/>.
- B.J. Coopeland. Artificial intelligence, Oct 2020. URL <https://www.britannica.com/technology/artificial-intelligence>.
- Abdel Dadouche. Machine learning in a box (part 2): Project methodologies - dzone ai, May 2018. URL <https://dzone.com/articles/machine-learning-in-a-box-week-2-project-methodolo-1>.
- data flair. TensorFlow Pros and Cons - The Bright and the Dark Sides, Feb 2018. URL <https://data-flair.training/blogs/tensorflow-pros-and-cons/>.
- Flora Dellinger, Julie Delon, Yann Gousseau, Julien Michel, and Florence Tupin. Sar-sift: A sift-like algorithm for applications on sar images. *2012 IEEE International Geoscience and Remote Sensing Symposium*, 2012. doi: 10.1109/igarss.2012.6350671.
- Armin W. Doerry and Fred M. Dickey. Synthetic Aperture Radar. *Optics and Photonics News*, 15(11):28, 2004. ISSN 1047-6938. doi: 10.1364/opn.15.11.000028.
- Felipe Ducau. Text generation with recurrent neural networks (rnns), Apr 2017. URL <https://blog.paperspace.com/recurrent-neural-networks-part-1-2/>.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 07 2011.
- Y. Dufournaud, C. Schmid, and R. Horaud. Matching images with different resolutions. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 1, pages 612–618 vol.1, 2000. doi: 10.1109/CVPR.2000.855876.
- J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, 3, 1973.
- R. Emek and N. Demir. Building detection from sar images using unet deep learning method. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIV-4/W3-2020: 215–218, 11 2020. doi: 10.5194/isprs-archives-XLIV-4-W3-2020-215-2020.
- Tiago Vaz Estêvão. O novo paradigma da vigilância na sociedade contemporânea - 'who Watches the Watchers'. *Observatorio*, 8(2):155–169, 2014. ISSN 16465954.
- Jie Feng, L.C. Jiao, Xiangrong Zhang, Maoguo Gong, and Tao Sun. Robust non-local fuzzy c-means algorithm with edge preservation for sar image segmentation. *Signal Processing*, 93(2):487 – 499, 2013. ISSN 0165-1684. doi: <https://doi.org/10.1016/j.sigpro.2012.08.024>. URL <http://www.sciencedirect.com/science/article/pii/S0165168412003179>.

- Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, 1981.
- R. Fjortoft, A. Lopes, P. Marthon, and E. Cubero-Castan. An optimal multiedge detector for sar image segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 36(3):793–802, 1998. doi: 10.1109/36.673672.
- Łukasz Gebel. Why we need bias in neural networks, Aug 2020. URL <https://towardsdatascience.com/why-we-need-bias-in-neural-networks-db8f7e07cb98>.
- Jie Geng, Hongyu Wang, Jianchao Fan, and Xiaorui Ma. Deep Supervised and Contractive Neural Network for SAR Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(4):2442–2459, 2017. ISSN 01962892. doi: 10.1109/TGRS.2016.2645226.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. Generative adversarial nets. *ArXiv*, 06 2014.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- Sonu Gurjar, Shivam Gupta, and Rajeev Srivastava. Image content description using lstm approach. pages 01–12, 07 2017. doi: 10.5121/csit.2017.70901.
- Anandakumar Haldorai and Arulmurugan Ramu. Supervised, unsupervised and reinforcement learning -a detailed perspective. *Journal of Advanced Research in Dynamical and Control Systems*, 11:429–433, 01 2019.
- Casper Hansen. Neural networks: Feedforward and backpropagation explained, Aug 2019. URL <https://mlfromscratch.com/neural-networks-explained/#/>.
- Corentin Henry, Seyed Majid Azimi, and Nina Merkle. Road segmentation in SAR satellite images with deep fully convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 15(12):1867–1871, 2018. ISSN 15580571. doi: 10.1109/LGRS.2018.2864342.
- Kelsey Herndon, Franz Meyer, Africa Flores, Emil Cherrington, and Leah Kucera. What is synthetic aperture radar?, Apr 2020. URL <https://earthdata.nasa.gov/learn/backgrounders/what-is-sar>.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.
- Mohammed Jabreel and Antonio Moreno. A deep learning-based approach for multi-label emotion classification in tweets. *Applied Sciences*, 9(6), 2019. ISSN 2076-3417. doi: 10.3390/app9061123. URL <https://www.mdpi.com/2076-3417/9/6/1123>.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

M. Kuhn and K. Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Chapman & Hall/CRC Data Science Series. CRC Press, 2019. ISBN 9781351609463.

Vihar Kurama. Pytorch vs. tensorflow: Which framework is best for your deep learning project?, Sep 2020. URL <https://builtin.com/data-science/pytorch-vs-tensorflow>.

Nathan Lambert. Machine learning engineer versus software engineer, Mar 2020. URL <https://towardsdatascience.com/machine-learning-engineer-versus-software-engineer-fb59f8cba9dd>.

Thuy Le Toan. Introduction to SAR Remote Sensing. *European Space Agency training website*, page 74, 2007. URL <https://earth.esa.int/landtraining07/D1LA1-LeToan.pdf>.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL <https://doi.org/10.1162/neco.1989.1.4.541>.

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, pages 253–256, 2010. doi: 10.1109/ISCAS.2010.5537907.

D. G. Lowe. Object recognition from local scale-invariant features. 2:1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

Lusa. Força aérea investe em drones para operações de vigilância contra incêndios - renascença, May 2020. URL <https://rr.sapo.pt/2020/05/19/pais/forca-aerea-investe-em-drones-para-operacoes-de-vigilancia-contraincendio-noticia/193383/>.

- Phil Madgwick. From instinct to intelligence: has ai taken a wrong turning?, Jun 2017. URL <https://becominghuman.ai/from-instinct-to-intelligence-has-ai-taken-a-wrong-turning-e84582dc4600>.
- Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. Dialoguernn: An attentive rnn for emotion detection in conversations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6818–6825, Jul. 2019. doi: 10.1609/aaai.v33i01.33016818. URL <https://ojs.aaai.org/index.php/AAAI/article/view/4657>.
- Stephane G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *Fundamental Papers in Wavelet Theory*, 2009. doi: 10.1515/9781400827268.494.
- Jakub Markiewicz, Karol Abratkiewicz, Artur Gromek, Wojciech Ostrowski, Piotr Samczyński, and Damian Gromek. Geometrical matching of SAR and optical images utilizing ASIFT features for SAR-based navigation aided systems. *Sensors (Switzerland)*, 19(24), 2019. ISSN 14248220. doi: 10.3390/s19245500.
- Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.
- Nina Marie Merkle. *Geo-localization Refinement of Optical Satellite Images by Embedding Synthetic Aperture Radar Data in Novel Deep Learning Frameworks*. PhD thesis, 2018.
- K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 525–531 vol.1, 2001. doi: 10.1109/ICCV.2001.937561.
- Michael Mills. Artificial Intelligence in Law : the State of Play 2016. *Thomston Reuters, Legal Executive Institute*, page 6, 2016.
- Dinh Ho Tong Minh, Dino Ienco, Raffaele Gaetano, Nathalie Lalande, Emile Ndikumana, Faycal Osman, and Pierre Maurel. Deep Recurrent Neural Networks for mapping winter vegetation quality coverage via multi-temporal SAR Sentinel-1. *arXiv*, 15(3):464–468, 2017. ISSN 23318422.
- Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997. ISBN 0070428077.
- Alberto Moreira, Pau Prats-Iraola, Marwan Younis, Gerhard Krieger, Irena Hajnsek, and Konstantinos P. Papathanassiou. A tutorial on synthetic aperture radar. *IEEE Geoscience and Remote Sensing Magazine*, 1(1):6–43, 2013. ISSN 21686831. doi: 10.1109/MGRS.2013.2248301.
- Jean Michel Morel and Guoshen Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009. ISSN 19364954. doi: 10.1137/080732730.
- Lichao Mou, Michael Schmitt, Yuanyuan Wang, and Xiao Xiang Zhu. A CNN for the identification of corresponding patches in SAR and optical imagery of urban scenes. *2017 Joint Urban Remote Sensing Event, JURSE 2017*, pages 17–20, 2017. doi: 10.1109/JURSE.2017.7924548.

NASA. Isro synthetic aperture radar, Apr 2020. URL <https://eosps.nasa.gov/missions/nasa-isro-synthetic-aperture-radar>.

Christopher Olah. Conv nets: A modular perspective, Jul 2014. URL <https://colah.github.io/posts/2014-07-Conv-Nets-Modular/>.

Christopher John Oliver, David Blacknell, and Richard White. Optimum edge detection in sar. 1996.

Coert Olmsted. Scientific SAR User ' s Guide by Coert Olmsted , Senior Programmer / Analyst July 1993
Scientific SAR User ' s Guide. (July), 1993.

Michael Phi. Illustrated guide to recurrent neural networks, Sep 2018. URL <https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5e>

S. Prince. *Computer Vision: Models, Learning, and Inference*. 2012.

Takowa Rahman and Md. Saiful Islam. Image segmentation based on fuzzy c means clustering algorithm and morphological reconstruction. In *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, pages 259–263, 2021. doi: 10.1109/ICICT4SD50815.2021.9396873.

Mario Fuentes Reyes, Stefan Auer, Nina Merkle, Corentin Henry, and Michael Schmitt. SAR-to-optical image translation based on conditional generative adversarial networks-optimization, opportunities and limits. *Remote Sensing*, 11(17):1–19, 2019. ISSN 20724292. doi: 10.3390/rs11172067.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009. ISBN 0136042597.

Jeffrey Saltz and Nicholas Hotz. Crisp-dm, 2018. URL <https://www.datascience-pm.com/crisp-dm-2/>.

SAR Journal. Russia's sar drone takes flight, Oct 2019. URL <http://syntheticapertureradar.com/russias-sar-drone-takes-flight/>.

savyakhosla. K-means++ algorithm, Jul 2021. URL <https://www.geeksforgeeks.org/ml-k-means-algorithm/>.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: <https://doi.org/10.1016/j.neunet.2014.09.003>. URL <https://www.sciencedirect.com/science/article/pii/S0893608014002135>.

- M. Schmitt, L. H. Hughes, C. Qiu, and X. X. Zhu. Sen12ms – a curated dataset of georeferenced multi-spectral sentinel-1/2 imagery for deep learning and data fusion. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W7, 2019. doi: 10.5194/isprs-annals-iv-2-w7-153-2019.
- Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.
- Sina Shahandeh. Methodology for management of artificial intelligence projects, Dec 2018. URL <https://www.linkedin.com/pulse/methodology-management-artificial-intelligence-sina-shahandeh>.
- Muhammad Shahzad, Michael Maurer, Friedrich Fraundorfer, Yuanyuan Wang, and Xiao Xiang Zhu. Buildings detection in VHR SAR images using fully convolution neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 57(2):1100–1116, 2019. ISSN 01962892. doi: 10.1109/TGRS.2018.2864716.
- Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2572683.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.
- Shashi Singh, Ajai Kumar, Hemant Darbari, Lenali Singh, Anshika Rastogi, and Shikha Jain. Machine translation using deep learning: An overview. pages 162–167, 07 2017. doi: 10.1109/COMPTELIX.2017.8003957.
- Vadim Smolyakov. Neural network optimization algorithms, Jan 2018. URL <https://towardsdatascience.com/neural-network-optimization-algorithms-1a44c282f61d>.
- Apache Spark, Oct 2020. URL <https://spark.apache.org/>.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- month=Jan Stanford Edu, year=2021. URL <http://infolab.stanford.edu/pub/voy/museum/samuel.html>.
- Shijian Tang and Song Han. Generate image descriptions based on deep rnn and memory cells for images features. 02 2016.
- R. Touzi, A. Lopes, and P. Bousquet. A statistical and geometrical edge detector for sar images. *IEEE Transactions on Geoscience and Remote Sensing*, 26(6):764–773, 1988. doi: 10.1109/36.7708.

- A. M. Turing. *Computing Machinery and Intelligence*. MIT Press, Cambridge, MA, USA, 1995. ISBN 0262560925.
- M. K. Vathsala and G. Holi. *RNN based machine translation and transliteration for Twitter data*. 2020.
- Christina Voskoglou. What is the best programming language for machine learning?, May 2017. URL <https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7>.
- Filip Wasilewski. Overview of multilevel wavelet decompositions¶, Jun 2021. URL <https://pywavelets.readthedocs.io/en/latest/ref/2d-decompositions-overview.html>.
- Debra Werner. Iceye raises \$87 million in series c investment round, Sep 2020. URL <https://spacenews.com/iceye-series-b-round/>.
- C. A. Wiley. Synthetic Aperture Radar. *IEEE Transactions on Aerospace and Electronic Systems*, (3):206–211, 1985.
- Deliang Xiang, Tao Tang, Canbin Hu, Yu Li, and Yi Su. A kernel clustering algorithm with fuzzy factor: Application to SAR image segmentation. *IEEE Geoscience and Remote Sensing Letters*, 11(7):1290–1294, 2014. ISSN 1545598X. doi: 10.1109/LGRS.2013.2292820.
- Li Yin. Comparison between deep learning tools, Mar 2017. URL <https://liyin2015.medium.com/comparison-between-deep-learning-tools-42560e61a5d5>.
- Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June(i):4353–4361, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7299064.
- Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.
- Xiao Xiang Zhu, Sina Montazeri, Mohsin Ali, Yuansheng Hua, Yuanyuan Wang, Lichao Mou, Yilei Shi, Feng Xu, and Richard Bamler. Deep Learning Meets SAR. *arXiv*, pages 1–23, 2020. ISSN 23318422.

NB: place here information about funding, FCT project, etc in which the work is framed. Leave empty otherwise.