INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Visão Computacional Geoespacial aplicada à deteção de Pedreiras

TIAGO HÉLDER AZEVEDO ANDRADE
Julho de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Visão Computacional Geoespacial aplicada à deteção de Pedreiras

Tiago Hélder Azevedo Andrade

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento

Orientador: Fátima Rodrigues

Júri:
Presidente:

Vogais:

Porto, 3 de julho de 2022

Dedicatória

Esta dissertação é dedicada aos meus pais, ao meu irmão e a todos os que me apoiaram para concluir esta etapa.

Resumo

A exploração de massas minerais e pedreiras pode, por vezes, causar instabilidade no terreno próximo, aumentando o risco para a segurança pública. Para mitigar esses riscos, a Infraestruturas de Portugal faz inspeções a estes locais. Pretende-se perceber, com base num conjunto de imagens aéreas, previamente fornecido, quais as pedreiras que estão mais próximas de vias de acesso. Neste sentido foram analisadas as abordagens mais comuns para problemas de Visão Computacional, explorando algoritmos baseados em Redes Neuronais Convolucionais. Foram analisadas as diferenças entre a deteção de objetos *One-Stage* e *Two-Stage*. Implementou-se a solução proposta de acordo com as características do conjunto de dados. Delinearam-se ainda as estratégias para avaliar a solução. Dos modelos testados, aquele que melhor se adequa ao problema é o modleo U-net com *backbone* Resnet34 apresentado 94% de *Intersection over Union* e aproximadamente 95% de *accuracy*.

Palavras-chave: Visão Computacional, Inteligência Artificial, Imagens Aeroespaciais, Pedreiras

Abstract

The exploration of mineral masses and quarries can sometimes cause instability in closeby terrain, increasing the risk to public safety. To mitigate these risks, Infraestruturas de Portugal inspects these locations. The goal is to understand, based on a set of aerial images, previously provided, which quarries are closest to access roads. In this sense, the most common approaches to Computer Vision problems were analyzed, exploring algorithms based on Convolutional Neural Networks. Differences between *One-Stage* and *Two-Stage* object detetcions were analyzed. The proposed solution was implemented according to the characteristics of the dataset. Strategies to evaluate the solution were also outlined. Of the tested models, the one that best fits the problem is the U-net model with *backbone* Resnet34 presenting 94% of *Intersection over Union* and approximately 95% of *accuracy*.

Agradecimentos

Gostaria de agradecer a todos os que estiveram direta ou indiretamente envolvidos na conclusão desta fase académica e profissional.

À professora Fátima Rodrigues por aceitar ser a orientadora deste projeto, pelos conselhos dados e disponibilidade.

À empresa Devscope, e de modo particular ao David Mota, que me apoiaram e desafiaram a realizar a dissertação no contexto de um ambiente profissional.

E ainda a familiares e amigos que suportaram e apoiaram todo o percurso académico.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

Lista de Abreviações

CRISP-DM	**Cr**oss **I**ndustry **S**tandard **P**rocess (for) **D**ata **M**ining
RGB	**R**ed **G**reen **B**lue
CNN	**C**onvolutional **N**eural **N**etwork
GPU	**G**raphical **P**rocessing **U**nit
VGG	**V**isual **G**eometry **G**roup
ResNet	**Res**idual (Neural) **Net**work
FPN	**F**eature **P**yramid **N**etwork
SVM	**S**upport **V**ector **M**achine
HSV	**H**ue **S**aturation **V**alue
FCN	**F**ully **C**onvolutional **N**etwork
YOLO	**Y**ou **O**nly **L**ook **O**nce
UAV	**U**nmanned **A**erial **V**ehicle
PDMA	**P**akistan **D**isaster **M**anagement **A**uthority
ReLU	**Re**ctified **L**inear **U**nit
NCD	**N**ew **C**oncept **D**evelopment
FEI	**F**ront-**E**nd **I**nnovation
FAST	**F**unction **A**nalysis **S**ystem **T**echnique
TOPSIS	**T**echnique (for) **O**rder (of) **P**reference (by) **S**imilarity (to) **I**deal **S**olution
IoU	**I**ntersection **o**ver **U**nion
ANOVA	**An**alysis **o**f **Va**riation

Capítulo 1

Introdução

1.1 Contextualização

Para melhorar a qualidade de vida humana, a extração e exploração dos recursos naturais foi desempenhando um papel cada vez mais fundamental, fornecendo os meios para construir as ferramentas necessárias. Uma destas áreas é a pesquisa e exploração de massas minerais em pedreiras, as quais no ano de 2015 envolveram 5330 trabalhadores [1].

Para evitar situações de derrocadas e, por sua vez, situações de perigo, são feitas regularmente por funcionários da Infraestruturas de Portugal, inspeções às pedreiras mais próximas das suas vias.

1.2 Problema

Com o passar do tempo, pedreiras em exploração ocupam uma área mais extensa que a inicial, podendo as suas extremidades estar perto de vias de acesso. Atualmente a tarefa da inspeção das pedreiras é feita de tal modo que possam ser feitas inspeções a outras pedreiras que não as que estão mais próximas de vias de acesso. Uma menor atenção nestas pedreiras pode implicar que situações de risco passem despercebidas aumentando o risco para funcionários e eventualmente civis.

1.3 Objetivos

Este projeto surgiu como uma adaptação de um projeto realizado pela entidade acolhedora. Como tal foram delineados objetivos para alcançar uma solução que resolva ou tenha um impacto positivo no problema já descrito. Os objetivos delineados são os seguintes:

- Deteção de pedreiras e os respetivos contornos com base em imagens aéreas previamente fornecidas;
- Comparação de diferentes implementações da deteção de pedreiras quanto ao seu tempo de execução e exatidão.
- Deteção de vias de acesso e os respetivos contornos com base nos dados geográficos das imagens fornecidas e na consulta de um mapa de estradas de livre acesso;
- Criação de uma imagem que permita visualmente perceber a proximidade entre as extremidades da pedreira e as da via de acesso.

1.4 Contribuições

O projeto proposto visa contribuir para simplificar a escolha das pedreiras para a inspeção por parte das entidades responsáveis. Como tal, uma outra contribuição será a minimização de riscos para a segurança pública, sendo as pedreiras selecionadas as que estão mais próximas de vias de acesso. Por fim, o estudo de abordagens para a resolução do problema contribuirá para um aumento do conhecimento nesta área e com resultados que demonstrarão quão adequadas são as técnicas para a resolução do problema.

1.5 Metodologia

Tendo em conta a complexidade e natureza do problema descrito em 1.2, adotou-se o modelo de processo CRISP-DM [2]. Este modelo engloba 6 etapas que poderão ocorrer várias vezes ao longo do projeto e de modo não linear.

De acordo com [3], estas etapas são:

- **Análise do Negócio:** Nesta fase inicial, o objetivo é entender qual o funcionamento e quais as regras de negócio que estão associadas ao projeto. Devem ainda ser definidos os objetivos e criado um plano para o projeto. Após a Análise de negócio segue-se a Análise de Imagens.
- **Análise de Dados:** O foco nesta segunda fase passa por obter informações como o formato, quantidade, dimensões e análises estatísticas, percebendo ainda se as imagens têm a qualidade necessária. Ao acabar esta fase, poderá ser necessário regressar à fase de análise de negócio. Caso contrário, procede-se à Preparação de Imagens.
- **Preparação de Dados:** Nesta fase que se segue, para que as imagens possam ser usadas num modelo, podem ser alterados valores de saturação, luminosidade e outras características, de modo que a informação da imagem se torne evidente. Com estes aspetos assegurados, é possível avançar para a fase da Modelação.
- **Modelação:** Na fase da modelação, selecionam-se uma ou vários algoritmos de Mineração de Dados e, para cada um deles, deve ser criado um modelo que seja indicado para o tipo de dados considerados, tendo em conta as restrições dos algoritmos. Com os modelos já criados, é possível passar à sua avaliação.
- **Avaliação:** Esta fase da avaliação consiste em analisar os resultados que o modelo em questão obtém e, em caso de este atingir as metas, procede-se à Implantação; caso contrário pode ser necessário voltar às fases pévias.
- **Implantação:** Por fim, esta fase consiste em disponibilizar o modelo implementado e os recursos necessários.

1.6 Estrutura do Documento

Esta dissertação contém a seguinte estrutura de capítulos:

- **Capítulo 1 Introdução** É o capítulo atual, no qual é explicado o âmbito, a metodologia e os objetivos do projeto.
- **Capítulo 2 Estado da Arte** Neste capítulo são revistas tecnologias e abordagens atualmente usadas para resolver problemas semelhantes.

- **Capítulo 3 Análise de Valor** Neste capítulo é feita uma análise ao valor que a solução implementada poderá trazer, de acordo com os métodos propostos.
- **Capítulo 4 Desenho da Solução** O propósito deste capítulo é formular soluções que apliquem as boas práticas da engenharia de software. Pretende-se também procurar alternativas, explicando qual delas se deve adotar com base nas suas vantagens e desvantagens.
- **Capítulo 5 Implementação** Neste capítulo são apresentadas todas as decisões tomadas e também apresentadas as complexidades e características mais relevantes da implementação da solução escolhida.
- **Capítulo 6 Experimentação e Análise de Resultados** Com este capítulo visa-se comparar os resultados obtidos com a solução implementada e com outras opções, avaliando se a solução atinge os objetivos definidos.
- **Capítulo 7 Conclusão** Por fim, este capítulo reflete a dissertação na sua totalidade, mencionando de que modo o conhecimento adquirido pode ser usado no futuro.

Capítulo 2

Estado da Arte

Neste capítulo, pretende-se analisar as abordagens mais comuns para problemas de análise de imagens aéreas, percebendo os conceitos mais relevantes dessa área de investigação. Procura-se ainda analisar outros casos de estudo, no sentido de conhecer aspetos das suas soluções, que possam ser usados na implementação do projeto em questão.

2.1 Visão Computacional

Tal como o nome indica a visão computacional é a área que estuda o uso de computadores para obter informação a partir de conteúdos visuais como imagens e vídeo. Este campo da inteligência artificial tem o propósito de auxiliar humanos em tarefas que envolvam a análise de imagem tomando ou recomendando ações.

Para além da visão computacional ter elevada aplicabilidade para as mais diversas áreas permite ainda analisar os conteúdos com um menor investimento de tempo. Segundo [4] "duas tecnologias essenciais são usadas nesta área: um tipo de aprendizagem automática chamada aprendizagem em profundidade e também redes neuronais convolucionais." Como tal vão ser explorados estes conceitos, neste capítulo.

Alguns dos conceitos que fazem parte desta área são a Classificação de Imagens, Segmentação de Imagens e Deteção de Objetos [5].

2.1.1 Classificação de Imagens

A classificação de imagens é a técnica usada para atribuir uma dada classe a uma secção de uma imagem. Pode ser aplicado a contextos com mais do que uma classe por imagem não havendo a necessidade de localizar o objeto a classificar [5].

2.1.2 Deteção de Objetos

Na deteção de objetos, em adição à classe a que pertence, pretende-se ainda prever a localização do objeto de estudo. A localização do objeto de estudo pode ser representada com o uso de uma figura (geralmente um quadrilátero) à volta do objeto de estudo [5].

2.1.3 Segmentação de Imagens

A segmentação de imagens é a técnica que permite identificar a região de um dado objeto numa imagem. Nesta técnica é criada uma máscara de bits para cada região, podendo ser aplicada em imagens que contenham mais que uma classe. Tal como na Deteção de

Objetos, também a segmentação é usada com o intuito de saber a localização do objeto na imagem mas para além disso serve para identificar qual o contorno que o objeto de estudo tem [6].

A figura 2.1, exemplifica a distinção entre Deteção de Objetos e Segmentação de Imagens.

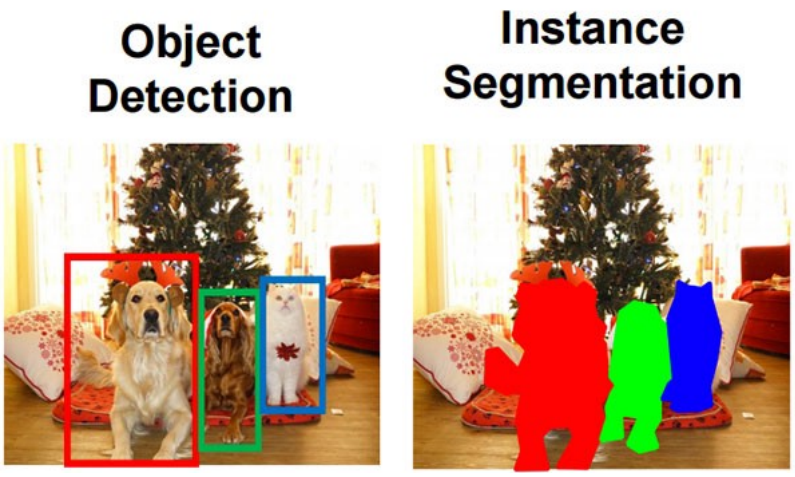

Figura 2.1: Deteção e Segmentação de Imagens [5]

2.2 Deep Learning

Machine Learning ou ML é uma subárea da Inteligência Artificial que usa algoritmos e dados para construir modelos preditivos. Por sua vez o *Deep Learning* ou DL é uma subárea de ML que lida com algoritmos inspirados na estrutura e função do cérebro humano. Os algoritmos de DL podem trabalhar com uma enorme quantidade de dados estruturados e não estruturados e envolvem várias camadas de redes neuronais artificiais [7].

No treino e avaliação de modelos de DL podem ser considerados até subconjuntos de dados, treino, validação e teste. O subconjunto de treino tem os dados que são fornecidos ao modelo, incluindo o valor atributo-objetivo. Este subconjunto é usado na fase do treino do modelo para identificar relações ou padrões entre os atributos. A fase do treino do modelo é também a que requer maior poder computacional sendo comum o uso de GPU's. Apesar de tanto GPU's como CPU's poderem ser compostos por mais do que um *core*, a quantidade de *cores* num GPU é superior à de um CPU. Isto permite que os GPU's consigam realizar mais tarefas em simultâneo [8].

Ainda na fase de treino do modelo, é comum o uso do sub conjunto de validação. Estes dados são usados para avaliar a qualidade das previsões que o modelo consegue fazer ao longo do treino alterando o modelo de acordo com isso. Um exemplo disto é a alteração dos pesos de uma rede neuronal [9].

A função do subconjunto de teste é avaliação do modelo treinado com base na comparação das previsões com os valores do atributo-objetivo das entradas deste subconjunto. Um dos obstáculos mais comuns é o *Overffiting*, que ocorre quando o modelo identifica o conjunto de treino de tal maneira que também as caraterísticas específicas de cada um dos elementos são também memorizadas. Isto diminui performance das previsões para entradas que não façam parte do subconjunto de teste [10].

Algumas das abordagens para evitar este problema são *Data Augmentation* e o uso de camadas de *DropOut*. *Data Augmentation* é o método que visa aumentar o subconjunto de treino fazendo altercações aleatórias às entradas do subconjunto de treino. *DropOut* é método que visa desativar de modo probabilístico neurónios de uma Rede Neuronal explicada em 2.3 [10].

2.3 Rede Neuronal

Uma Rede Neuronal é um algoritmo de classificação com um funcionamento inspirado no cérebro humano. Inicialmente consideram-se 3 tipos de camadas ilustradas na figura 2.2. A camada de *input*, a camada de *output*, e entre elas, uma ou várias camadas intermédias com um *bias* associado a cada um dos seus nós. O treino da rede é formado por várias épocas. Em cada época os valores da camada inicial são multiplicados por pesos e essas multiplicações são somadas entre si. Para cada nó da camada seguinte o somatório das multiplicações já feitas, é adicionado a um *bias* e esse valor é passado a uma função de ativação. Se o resultado da função de ativação for superior a 0 esse nó fica ativo, caso contrário fica inativo. Os nós ativos contêm os valores de entrada (valor passado à função de ativação) usados para repetir este processo para as próximas camadas até que a camada de *output* esteja preenchida finalizando então a época [11].

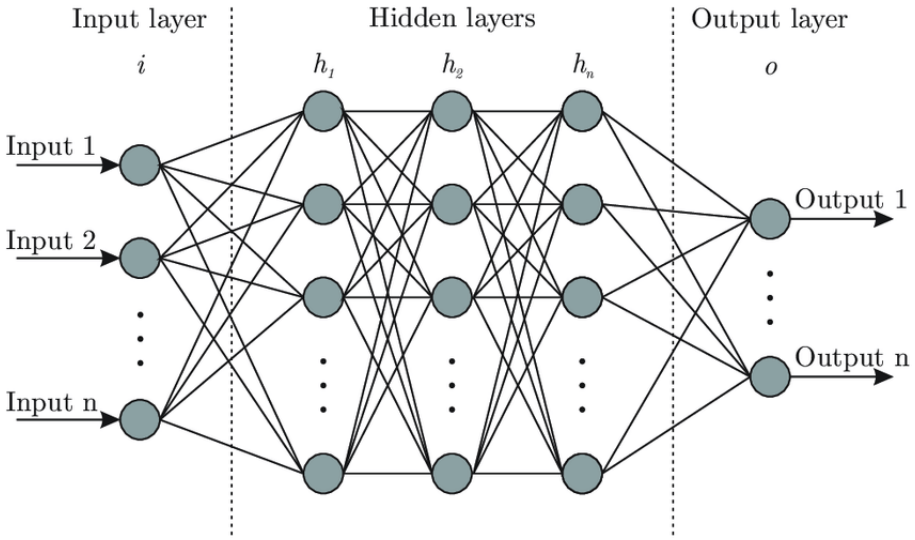

Figura 2.2: Representação de Rede Neuronal [12]

Na primeira iteração os valores dos pesos e os *bias* são gerados aleatoriamente e nas restantes épocas são alterados de modo a obter melhores resultados.

2.4 Rede Neuronal Convolucional

Tal como o nome indica uma Rede Neuronal Convolucional é um especificação de uma Rede Neuronal. Nesta especificação são normalmente usadas as operações de Convolução, *Pooling* e *Flatten*. Nestas operações é importante ter presente que uma imagem pode ser representada por uma matriz de números conhecida como bitmap ou mapa de bits.

2.4.1 Convolução

Para a operação de convolução, podem ser criados vários canais a partir do mapa de bits. Um canal é uma matriz com os valores de cada pixel para um dado formato de cores. Por exemplo, se for considerado o formato RGB, são consideradas 3 canais (cores), vermelho, verde e azul. Para cada canal é construída uma ou várias matrizes pré-definidas conhecidas como *kernel*. Posteriormente selecionam-se vários excertos do mapa de bits com dimensões iguais às da *kernel* e começando no canto superior esquerdo. Os valores da matriz selecionada e da *kernel* são usados para uma multiplicação de matrizes e o resultado é adicionado a uma matriz de resultados conhecida como *feature map*. O próximo excerto a ser selecionado do mapa de bits, deverá estar à direita do anterior ou caso isso não seja possível, a baixo e o mais à esquerda possível [13].

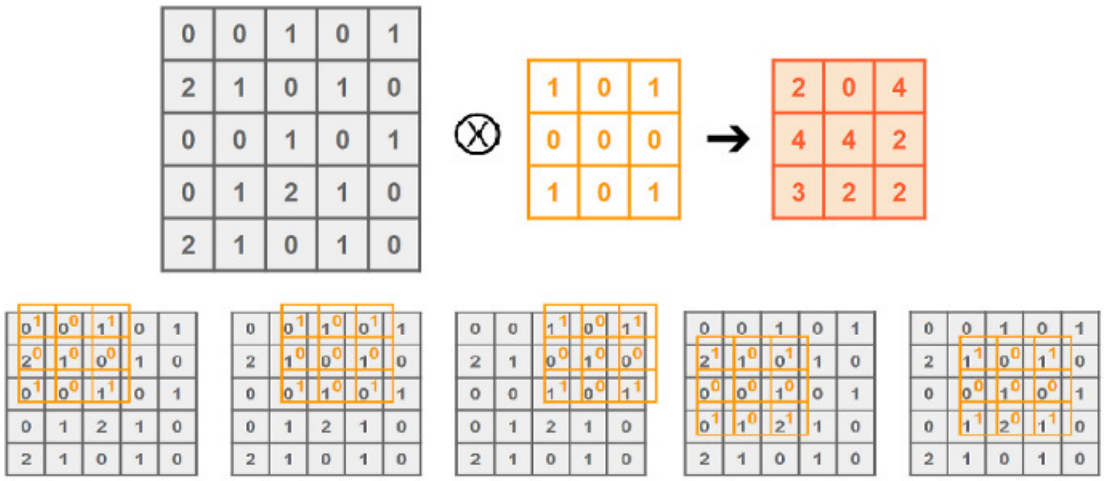

Figura 2.3: Exemplo de Convolução de Matriz [13]

A figura 2.3 demonstra este mesmo processo o qual se repete até que o mapa de características esteja totalmente preenchido, sendo que os resultados são adicionados pela mesma ordem que são selecionados os excertos do mapa de bits.

Podem ainda ser usados o *padding* e o *stride*. O *padding* consiste na adição de bits com o valor 0 à volta de cada canal. Se o *padding* for igual a 1 será adicionada uma destas beiras, se for igual a 2, duas beiras, e assim sucessivamente. O valor do *stride* é o número de unidades entre a posição de excertos da matriz. Por omissão, considera-se que o valor do salto é igual a 1, um excerto deverá estar à uma unidade direita do anterior (ou uma a baixo e o mais à esquerda possível) [13]. O objetivo da Convolução é extrair os diferentes contornos presentes na imagem [14].

2.4.2 Pooling

De modo semelhante à camada Convolucional, a camada de *Pooling* também seleciona áreas específicas do mapa de bits e está representado na figura 2.4.

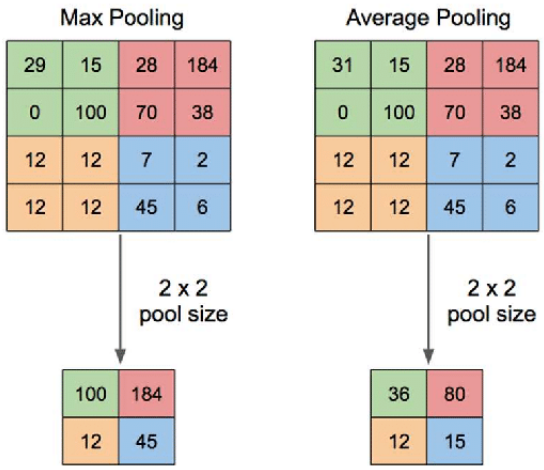

Figura 2.4: Exemplo da operação de *Pooling* [15]

A área selecionada é usada para uma de duas operações a qual insere o resultado numa matriz. Essas operações são o *Max Pooling* e o *Averege Pooling*. O *Max Pooling* consiste em calcular o maior valor da área selecionada e o *Averege Pooling* consiste em calcular o valor médio. O uso desta técnica permite reduzir as dimensões da matriz introduzida diminuindo o poder computacional necessário. Para alem disso o Pooling possibilita extrair apenas as caraterísticas mais dominantes [14].

2.4.3 Flatten

Depois dos processos de convolução e *Pooling*, que podem ocorrer várias vezes, a matriz resultante é achatada. Ou seja, as linhas dessa matriz são juntas formando assim uma grande lista de números. Esta lista de números é usada como a camada de input para as camadas totalmente conectadas. Camadas totalmente conectadas são semelhantes às camadas intermédias de uma rede neuronal [13].

2.5 Arquiteturas de Redes Neuronais

2.5.1 Alex-Net

A Alex-Net é uma dos primeiras redes baseada na rede neuronal convolucional com relativamente algum sucesso em reconhecimento de imagem. Na arquitetura proposta uma rede teria 5 camadas convolucionais e 3 totalmente conectadas usando ainda a função de ativação *softmax* [16] .

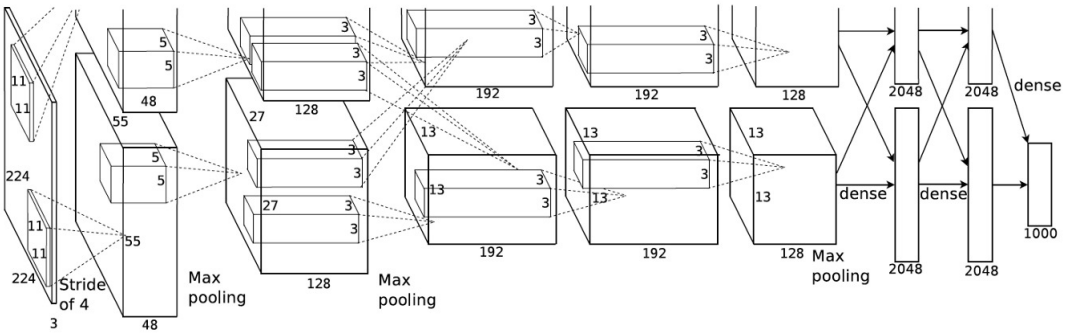

Figura 2.5: Arquitetura proposta da rede Alex-Net executada com 2 GPUs [16]

De acordo com a figura 2.5, existem camadas de *Max Pooling* após a primeira, segunda e quinta camadas convolucionais. A primeira camada convolucional tem 96 kernels de 11x11x3 e a segunda 256 kernels de 5x5x48. A terceira e quarta camadas tem ambas 384 kernels de tamanhos 3x3x256 e 3x3x192 respetivamente. A quinta camada tem 256 núcleos de dimensões iguais aos da quarta. As camadas totalmente conectadas tem 4096 neurónios cada [16].

2.5.2 VGG

O VGG ou *Visual Geometry Group* é uma rede neuronal usada para reconhecimento de imagem, baseada na arquitetura de uma rede neuronal convolucional. Os autores propuseram uma na qual seriam usadas 2 redes convolucionais com 16 e 19 camadas respetivamente com todas as camadas convolucionais com *kernels* de 3x3 [17].

Tabela 2.1: Estrutura das redes que constituem VGG

Rede de 19 Camadas	conv3-64	conv3-64	maxpool	conv3-128	conv3-128	maxpool
	conv3-256	conv3-256	conv3-256	conv3-256	maxpool	conv3-512
	conv3-512	conv3-512	conv3-512	maxpool	conv3-512	conv3-512
	conv3-512	conv3-512	maxpool	FC	FC	FC
Rede de 16 Camadas	conv3-64	conv3-64	maxpool	conv3-128	conv3-128	maxpool
	conv3-256	conv3-256	conv3-256		maxpool	conv3-512
	conv3-512	conv3-512		maxpool	conv3-512	conv3-512
	conv3-512		maxpool	FC	FC	FC

Segundo a arquitetura proposta, a qual está representada na tabela 2.1 seriam usadas 2 camadas convolucionais com 64 *kernels* seguidas de uma camada de *max pooling*, mais 2 camadas convolucionais com 128 *kernels* e mais uma de *max pooling*. De seguida a rede mais pequena, possui 3 camadas convolucionais de 256 *kernels*, uma camada de *max pooling*, seguida de 3 convolucionais com 512 *kernels* e mais uma camada de *max pooling* e ainda mais 3 camadas convolucionais de 512 *kernels*. De modo semelhante à rede de 16 camadas, a de 19 tem grupos de 4 camadas convolucionais em vez de grupos de 3. Por fim em ambas as redes são usadas mais uma camada de *max pooling*, 3 camadas totalmente conectadas e ainda a função de ativação *softmax* [17].

2.5.3 ResNet

A *ResNet*, apresentada por membros da Microsoft, é uma rede com aspetos semelhantes ao VGG. Esta rede, tem uma arquitetura constituída por: uma camada convolucional com uma *kernel* de 7x7. Todas as camadas convolucionais seguintes tem dimensões 3x3. Depois disso existem 6 camadas convolucionais com 64 *kernels* seguida uma camada de *max pooling*. Após esta camada de **pooling**, existem 6 camadas convolucionais com 64 *kernels* mais 8 camadas convolucionais com 128 *kernels*, seguidas de 12 camadas convolucionais com 256 *kernels* e ainda 6 camadas convolucionais com 512 *kernels*. Por fim existe uma camada de *average pooling* e uma camada totalmente conectada [18].

Um outro aspeto relevante do funcionamento desta rede é o uso de atalhos para os quais quando as dimensões da matriz de saída de uma camada, e de entrada para a camada seguinte são iguais, os valores são fornecidos para essa camada evitando o processamento de algumas destas. Nas camadas em que há alteração de dimensões (de *kernel*) podem ou não ser alteradas as dimensões dos valores introduzidos [18].

2.5.4 U-Net

A U-net é uma rede neuronal cuja arquitetura contém um caminho de compressão (lado esquerdo da figura 2.6) e o caminho de expansão (lado direito da figura 2.6). O caminho de compressão é constituído por duas convoluções de 3x3 sem margem, seguidas de uma unidade ReLU e ainda uma operação de 2x2 de *pooling* máximo com *stride* 2. O caminho de expansão consiste numa expansão do mapa de *features* seguido de uma convolução 2x2 que diminui assim o mapa de *features*. Depois é feita uma concatenação com o respetivo mapa de *features* do caminho de compressão e por fim, duas convoluções 3x3 mais uma unidade ReLU.

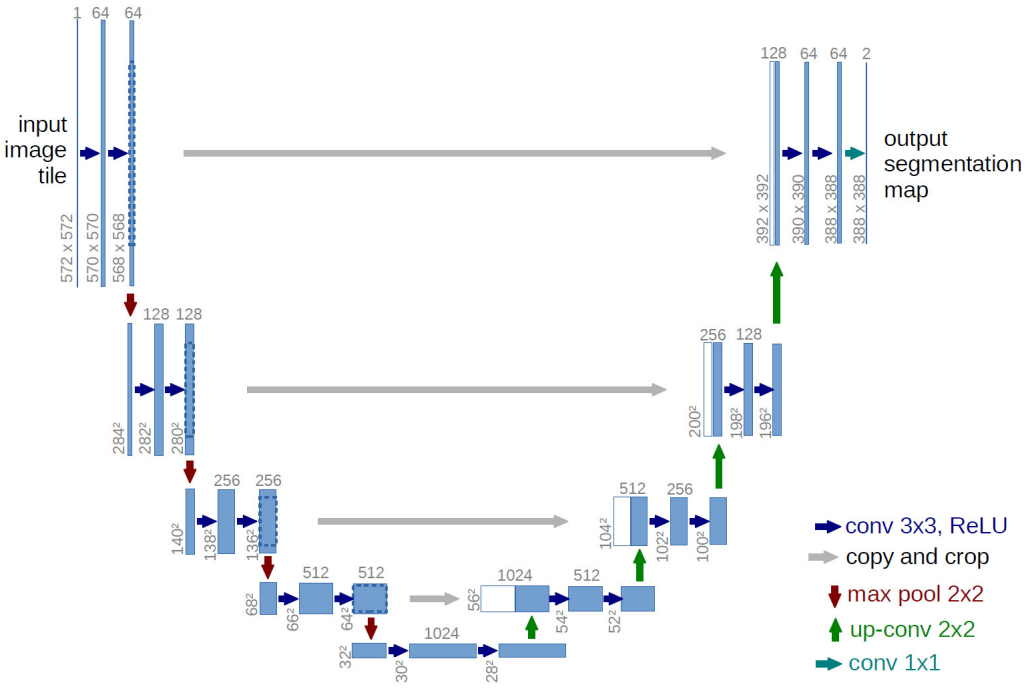

Figura 2.6: Representação da U-Net REF

2.5.5 Principais caraterísticas das arquiteturas mencionadas

Alguns dos aspetos aplicados na arquitetura AlexNet estão relacionados com a escolha da função de ativação e o tipo de *Pooling*. Apesar da função de ativação mais popular na altura em que esta arquitetura foi proposta ser função *tanh*, foi usada a função ReLU que provou ser mais rápida. Os autores perceberam também que uso de *Pooling* sobreposto minimizava o risco de *Overfit* [19].

Já a arquitetura VGG é baseada na arquitetura AlexNet propondo o uso de kernels mais pequenas e menor valor de *stride*. Para além disso esta arquitetura propõe ainda o uso de 3 unidades de ReLU. O uso de 3 unidades ReLU faz com que a função de ativação seja mais discriminatória e o uso de kernels menores leva um maior número de camadas que pode por sua vez levar a melhor performance [20].

Por sua vez a arquitetura ResNet tem uma grande vantagem em relação às anteriores. As arquiteturas anteriores contêm um grande número de camadas, melhorando assim a performance, mas apenas a custo de maiores tempos de treino. A ResNet propõe o uso de atalhos para os quais os valores de uma dada camada posterior evitando o processamento em camada intermédias. Deste modo a Resnet pode ter tempos de treino significativamente inferiores às 2 arquiteturas anteriores [18].

2.6 Avaliação de Modelos

Nesta secção serão apresentadas métricas de avaliação de modelos que podem ser usadas para avaliar modelos de visão computacional.

2.6.1 Matriz de Confusão

Num problema de classificação o objetivo é fazer previsões, as quais podem estar corretas ou incorretas. Quando uma previsão é feita, é atribuído um de vários valores possíveis [21]. Para este problema pode ser considerada uma matriz de confusão com as classes pedreira e ambiente. Deste modo, a tabela 2.2 seria uma representação de uma matriz de confusão.

Tabela 2.2: Matriz de Confusão para o problema

	Previsão	
	Pedreiras	Ambiente
Pedreiras	Verdadeiros Positivos	Falsos Positivos
Ambiente	Falsos Negativos	Verdadeiros Negativos

Com a matriz de confusão é possível calcular várias métricas para avaliar modelos de classificação. Algumas das mais comuns são:

- **accuracy** que é quociente entre o número de previsões corretas pelo total de previsões;
- **precision** que consiste no quociente entre o número de previsões positivas corretas pelo total de previsões positivas;
- **recall** que representa o quociente entre o número de previsões positivas corretas pelo total registos realmente positivos;
- **f1** a qual é a média harmónica da revogação e precisão;

2.6.2 Intersetion over Union

Outra métrica que pode ser usada para avaliar modelos que resolvem problemas de segmentação, é a *Intersetion over Union*. Para calcular esta métrica e para cada imagem que faz parte do conjunto de teste uma considera a área retangular à volta do objeto detetado pelo modelo e outra a área retangular à volta do objeto de treino. É também necessário saber a posição que as áreas ocupam em relação às extremidades da imagem. O valor da *Intersetion over Union* para essa imagem será igual ao quociente da interseção das duas áreas pela sua união [22], tal como demonstra a figura 2.7.

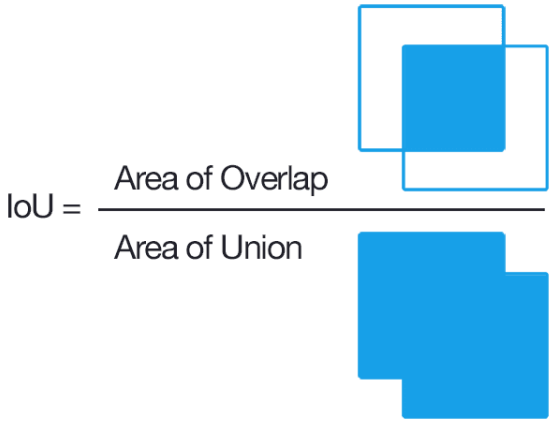

Figura 2.7: Fórmula da Interseção sobre União [22]

A *Intersetion over Union* usada para avaliar o modelos pode ser calculada como a média de *Intersetion over Union* de todas as imagens do conjunto de dados de teste.

2.6.3 Loss

Em problemas de otimização é comum o uso de uma função que avalie a qualidade de cada alternativa. Dependendo do contexto, quão maior o valor melhor a alternativa e vice-versa. Por norma, as funções de avaliação para redes neuronais calculam quão errada está a rede, como tal o resultado deste tipo de funções dá-se o nome *Loss*. Algumas das funções de *Loss* são a *Maximum Likelihood* e *Cross-Entropy* [23].

2.7 Casos de Estudo de Análise de Imagens Aéreas

Com esta subsecção pretende-se apresentar casos de estudo que resolveram problemas na área da análise de imagens aéreas. Estes casos de estudo poderão ter aspetos relevantes para o desenvolvimento do projeto em questão.

2.7.1 Software para reconhecimento de defeitos de telhados em imagens aereas [24]

O projeto descrito neste artigo pretende detetar defeitos em telhados de edifícios. Ao longo de elevados períodos de tempo e devido a fenómenos climatéricos os edifícios podem ficar danificados e apesar de ser possível fazer a sua manutenção essa tarefa é bem mais difícil em telhados tem um acesso difícil para humanos. A inspeção de telhados depende da época

do ano, tem custos de tempo e dinheiro e deve ser feita tendo em conta o propósito do edifício. Neste sentido foram recolhidas imagens de telhados com o uso de drones equipados com câmaras.

Os defeitos que se pretende que sejam detetados são as manchas causadas pela água da chuva e com tons mais escuros que o resto do telhado. A maneira mais simples de os detetar seria através da análise binária da imagem com um canal recebida transformada com HSV. A segmentação das imagens através da transformação HSV, proposta pelos autores, é composta pelos seguintes passos :

- 1. Suavizar a cor da imagem com o filtro Gaussiano para eliminar o ruído.
- 2. Fazer a transformação HSV com as componentes inseridas de matização, saturação e valor.
- 3. Escolher dos limites para a matização, saturação e valor da imagem e de acordo com esses valores criar uma cópia da imagem com os píxeis a 255 (branco) ou 0 (preto) dependendo do valor do pixel original estar ou não dentro do intervalo definido.
- 4. Na imagem criada no passo anterior procuram-se áreas de 8 ligações com um algoritmo de duas passagens. As áreas de 8 ligações são áreas para as quais um dado píxel os píxeis adjacentes e na diagonal têm valor igual [25]
- 5. Das áreas encontradas as mais pequenas são descartadas de acordo com o valor de limite mínimo, sendo as restantes consideradas defeitos dos telhados.
- 6. De acordo com a escala associada a cada imagem são considerados os perímetros e áreas das manchas detetadas.
- 7. Os resultados da segmentação são guardados em ficheiros .jpg e as áreas associadas a cada imagem são guardadas em ficheiros .xml .

Os autores detetaram que os valores usados para o limite máximo de V pode dramaticamente alterar os resultados reconhecendo sombras como defeitos do telhado.

Neste artigo são descritas 2 arquiteturas de redes neuronais convolucionais. A FCN13 com 13 camadadas convolucionais e a FCN17 com 17 camadas. O processo de treino dos modelos foi feito com 33 imagens e a validação com 2 e todas elas com 464 pixeis de comprimento e 825 de largura. Nestes processos foram usadas a *loss* e o *dice* como métricas para avaliar os modelos criados. O treino dos modelos foi feito com o otimizador Adam com velocidade de treino 0.00001. Os pesos recebidos na época 73 da FCN17 foram usados para teste. O valor da função de perda foi de 0.224 para os dados de treino e 0.021 para os de validação.

As abordagens sugeridas pelos autores para 80 épocas revelam que apesar das limitações do tamanho do conjunto de dados, os valores de *dice* chegam a 0.78 com o modelo FCN17 e 0.76 com o modelo FCN13.

Os autores concluíram com base no seu estudo que os modelos implementados conseguem identificar defeitos com valores aceitáveis nas medidas de *dice* e *loss function*. Foi ainda medido o tempo necessário para treinar cada um dos modelos tendo o FCN17 demorado aproximadamente 1270 segundos e o modelo FCN13 1072 segundos.

2.7.2 Deteção e reconhecimento de objetos em fotografias aereas usando redes neuronais convolucionais [26]

O projeto realizado pelos autores tem o principal objetivo de identificar diferentes objetos com imagens recolhidas por um satélite ou um veículo voador. Estas fotos podem ser alvo de falhas de luminosidade, compressão, ruído ou conter pequenos objetos. Neste contexto em especial os autores pretendem classificar os objetos detetados numa das seguintes classes : avião, edifício, carro, helicóptero, navio e veículo de grande porte.

A abordagem escolhida pelos autores é adoção da arquitetura YOLO versões 2 e 3. Estas arquiteturas lidam com a análise de imagens, não como um problema de classificação, mas como um problema de regressão. Como tal, uma única rede neuronal faz as previsões quanto à posição das secções de cada imagem e da classe a que cada secção pertence [27].

Na arquitetura YOLO 2, proposta por [28], usa-se a rede neuronal "Darknet-19"que é constituída por 19 camadas convolucionais, com *kernels* convolucionais de 3x3; 5 camadas de *max pooling*, uma camada global de *max pooling* e a função "softmax"como função de ativação. A Darknet-19 foi treinada com 160 épocas com a alteração de que a ultima camada convolucional foi substituída por outras 3 com 1024 *kernels* de dimensões 3x3, seguidos de uma camada convolucional 1x1 e número de saídas igual ao das classes. Nesta abordagem as entradas da 21ª camada convolucional são uma combinação das saídas da 13ª e 20ª camadas.

Já na arquitetura YOLO 3 [29], é usada a Darknet-53 com 53 camadas convolucionais, com kernels de 3x3 e 1x1. Em vez de serem usadas camadas de *pooling* as convoluções são feitas com uma margem igual a 2. No que toca a previsão, a YOLO 2 considera que apenas uma classe está presente para cada uma das regiões detetadas, no entanto, YOLO 3 usa regressão logística fazendo a previsão para várias classes. Na YOLO 3 a previsão é ainda feita para 3 escalas dividindo as dimensões da imagem introduzida em 32, 16 e 8.

Os autores compararam ainda as arquiteturas mencionadas entre si na deteção de pequenos objetos em fotografias. O conjunto de dados para treino contém 7869 imagens, o conjunto de imagens para validação tem 2082 e o de imagens de teste tem 1504. Todas as imagens de treino de modelo têm dimensões 608x608. Para a comparação as métricas usadas foram a *Intersection over Union* que mede a qualidade da deteção e classificação do modelo; e a *Precision* que é o quociente entre o número de deteções corretamente previstas e todas deteções previstas (corretas e erradas).

Tabela 2.3: Resultados Caso de Estudo 2

Parametro	YOLO 2	YOLO 3
	valor %	
Precision média para avião	88.09	88.66
Precision média para edifício	70.81	72.33
Precision média para carro	65.62	86.49
Precision média para helicóptero	82.81	87.62
Precision média para navio	84.97	86.37
Precision média para veículo de grande porte	65.29	76.50
Precision média	76.14	82.99
Índice de *Intersection over Union*	50.71	61.37

A tabela 2.3 contém os resultados obtidos pelos autores sendo possível detetar que a *Precision* média excede os 70% em ambas as arquiteturas sendo 6% superior com a YOLO 3. A maior diferença de desempenho nos resultados obtidos é na deteção de carros na qual a YOLO 3 é 20% superior. Notavelmente o índice de *Intersection over Union* da YOLO 3 é 10% superior.

Foi então concluído que ambas as arquiteturas usadas obtiveram resultados altos tendo YOLO 3 obtido os melhores resultados de *Precision* média. Apesar disso conclui-se também que a deteção baseada nestas arquiteturas pode ser usada para a identificação de pequenos objetos em imagens aéreas.

2.7.3 Aplicação de Aprendizagem em Profundidade em Imagens Aéreas UAV para a deteção de inundações. [30]

Neste artigo os autores propõem um sistema que seja capaz de identificar regiões inundadas e dar início a operações de socorro e reabilitação o mais cedo possível. Atualmente os sistemas de deteção de imagens usam satélites os quais demonstram atraso, e falta de exatidão. Como tal, foram usados veículos voadores sem nome em Swat no Paquistão.

Foram recolhidas, no total, 300 imagens capturadas por UAV em RGB e com alta resolução. As restantes imagens do conjunto de dados, foram recolhidas das bases de dados do PDMA (Pakistan Disaster Management Authority) contendo imagens anteriores e posteriores. As imagens mostram os danos causados por cheias e de que modo esses danos dificultam as atividades de socorro e reabilitação. Imagens que contivessem os rios "Swat"e "Daral", foram removidas do conjunto de dados, no sentido de impedir que o modelo as considerasse semelhantes a casos de inundação. Para corrigir questões topográficas, da inclinação da terra e distorção, foram usada a técnica de processamento "ortoretificação". Para corrigir ruído das imagens foi usado um filtro de mediana. O tamanho e luminosidade das imagens foi ajustado, para remover fundos indesejados. Foi a ainda feita aumentação de dados com base em recortes e alterações aleatórias.

Os objetos de interesse foram detetados com o uso de Aprendizagem Supervisionada, em específico *Haar Cascade Classifier*. Como o objetivo é o planeamento de ações de socorro de alta velocidade foi adotado este classificador devido à sua elevada velocidade computacional. O método foi utilizado para detetar ruas, pontes e edifícios através da análise da intensidade dos píxeis em áreas retangulares.

Devido existirem diferentes frequências das classes consideradas pelos autores foi usada uma função de balanceamento com base na mediana das frequências dessa classe. Essa função atribui a cada classe um peso. As classes consideradas pelos autores e as respetivas frequências estão presentes na tabela 2.4.

Tabela 2.4: Frequência de classes do Caso de Estudo 3 [30]

Classe	Frequência (%)
Edifícios	30.1
Estradas	42.8
Solo	11.9
Relva	10.0
Água	5.2
Pontes	1.1

Os autores treinaram um modelo de Rede Neuronal Convolucional com um camada convolucional, uma camada de *Max Pooling* e duas camadas totalmente conectadas. Na camada convolucional e na primeira conectada foi usada a função de ativação ReLU (*Rectified Linear Unit*).

Para testar o modelo foi o usado o *10-Fold-Cross-Validation* em que o conjunto de dados foi dividido em 10 partes. Cada uma das partes é usada como conjunto de teste sendo as restantes usadas como conjunto de treino. No processo de treino foi usada uma *learning rate* igual a 0.0001 e um número máximo de épocas igual a 5.

Com o conjunto de dados original os autores obtiveram 84.4% de *accuracy* num conjunto de dados de teste de 400 imagens. No entanto o valor de *accuracy* subiu para 91% quando a solução proposta foi aplicada ao conjunto de dados melhorado. Os resultados obtidos em ambas as situações estão descritos na tabela 2.5.

Tabela 2.5: Resultados Experimentais do Caso de Estudo 3 [30]

Métricas	Conjunto de Dados Original	Conjunto de Dados Melhorado
Accuracy	0.84	0.91
Precision	0.84	0.92
Recall	0.90	0.95
F1	0.87	0.93

Ao comparar os resultados obtidos com outras metodologias para deteção de cheias, os autores perceberam com base nos resultados da tabela 2.6 que apenas a *deep learning neural network* obteve um resultado melhor com uma diferença de 1%.

Tabela 2.6: Comparação de Resultados do Caso de Estudo 3 [30]

Método	*Accuracy*	Quantidade de Imagens	Local
Deep Learning Neural Network [31]	92%	1464	Lao Cai, Vietname
Semantic metadata and visual data with Convolutional Neural Network [32]	83.96%	6600	*Variados*
Random Forest Classifier [33]	87.5%	5000	Yuyao, China
Analytical Hierarchical Process [34]	84.4%	519	Cidade de Najran, Reino da Arábia Saudita
Support Vector Machines [35]	84.97%	1000	Terengganu, Malásia
Solução proposta (pelos autores) [30]	91%	3000	Swat, Paquistão

Assim, os autores concluíram que esta abordagem demonstra uma melhoria de *accuracy* em 5% com as alterações feitas ao conjunto de dados. Verificou-se que um conjunto de dados maior melhorou os resultados do modelo, apesar de que este seja dependente do conteúdo das

imagens. A solução proposta foi comparada com outras soluções, demonstrando resultados apenas superados pela *deep learning neural-network* por uma margem de 1%. Pelo que a solução é bastante promissora para ser usada no contexto descrito.

Capítulo 3

Análise de Valor

Este capitulo refere de modo mais detalhado a análise de valor deste projeto. A análise de valor é a abordagem sistemática que tem por objetivo melhorar o valor de um produto, projeto, ou processo. Nesta abordagem eliminam-se custos desnecessários sem comprometer a qualidade, confiabilidade e performance do objeto de estudo. [36]

3.1 New Concept Development Model

O desenvolvimento de um novo produto ou serviço, requer inovação no sentido de aumentar a qualidade do mesmo. O processo de inovação de contem as fase de Front-End Innovation, New Concept Development Model e comercialização. A FEI é a fase em que se define que produtos vão ser desenvolvidos e a comercialização é a fase em que se disponibiliza o produto [37].

O New Concept Development Model é o modelo que representa de modo simples e holístico a perspetiva definida na FEI. O modelo está dividido em 3 partes: a central, a interior e a exterior. A parte central do modelo refere a visão e estratégias por de trás da FEI. A parte exterior consiste em fatores do ambiente externo. Já a parte interior define 5 atividades que constituem a FEI [38], descritas nos seguintes sub-capítulos.

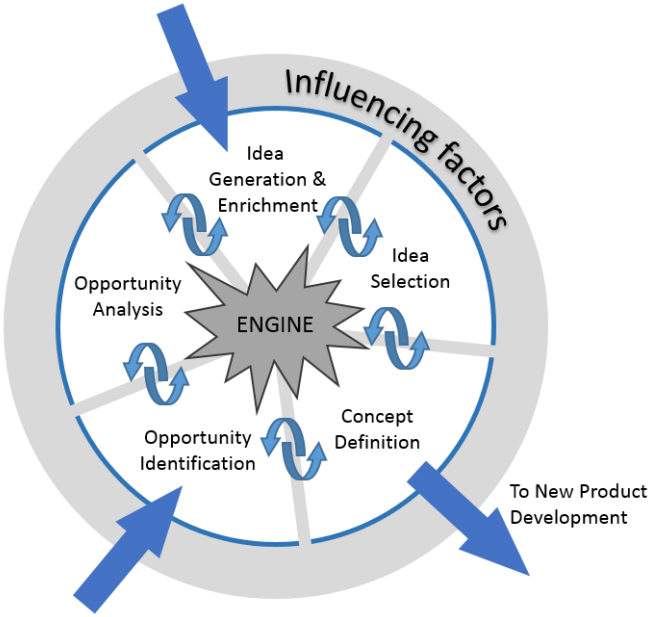

Figura 3.1: Representação do modelo NCD. (Koen et al., 2001)

A figura 3.1 é uma representação do modelo NCD a qual tem formato circular e setas entre cada uma das 5 fases da parte interior representando assim que o processo pode ser iterativo. Existem ainda 2 setas de entrada na "Identificação da oportunidade"e na "Geração e enriquecimento de ideias"uma vez que, estes podem ser os passos de iniciais do processo. Quando a definição do conceito estiver concluída pode-se avançar para o desenvolvimento do produto [37].

3.1.1 Identificação da oportunidade

Como já foi mencionado, o contexto deste projeto incide em facilitar o processo inspeção de áreas de exploração e extração de minérios. É importante ter em conta que com o passar do tempo a terreno sofre alterações devido à exploração propriamente dita e também a fenómenos climatéricos. Estas alterações sem a devida atenção podem ser a causa de instabilidade na área aumentando o risco para os funcionários e vias de acesso mais próximas.

3.1.2 Análise da oportunidade

A oportunidade para a resolução de um problema com este projeto deve-se ao apoio que um sistema de identificação da proximidade entre pedreiras e vias de acesso pode trazer aos técnicos da Infraestruturas de Portugal.

De acordo com a secção anterior ao longo do tempo, as áreas de exploração de minérios podem ficar mais inseguras. Como tal, existe a necessidade de fazer com que as inspeções feitas às várias pedreiras do país sejam feitas priorizando aquelas que acarretam um maior risco para a civilização e funcionários.

3.1.3 Geração e enriquecimento de ideias

Para a geração de ideias foram analisados trabalhos sobre o processamento de imagem, mais em concreto no ramo da visão aérea. Percebeu-se então que abordagens são usadas para a resolução de problemas desta natureza e também que tecnologias são mais comuns para tal.

3.1.4 Seleção de ideias

Tal como mencionado, neste projeto vai ser usada uma metodologia CRISP-DM, e como tal é necessário delinear como proceder em cada uma das suas etapas. Particularmente na fase da Modelação para a qual existem tecnologias para a ajudar a concretizar. Nesse sentido foram selecionadas algumas das frameworks mais comuns para aprendizagem em profundidade, sendo estas: TensorFlow, OpenCV, PyTorch e FastAI.

Para a comparação foram definidos os seguintes critérios:

- Popularidade - Representa o nível de popularidade da framework para Aprendizagem em Profundidade;
- Performance - Avalia o nível do tempo de execução das funcionalidades da framework;
- Funcionalidade - Representa a quantidade funcionalidades que tem disponível;
- Utilização - Mede a facilidade de utilização;

- Formato Imagens - Clarifica se existe ou não suporte para todos os formatos de imagem (incluindo .tif);

Technique for Order of Preference by Similarity to Ideal Solution

O TOPSIS é uma técnica para a análise de decisões multi-critério. A comparação de opções é feita com base num conjunto de critérios, podendo esta técnica ser usada em várias industrias como ferramenta para a análise de valor de um produto ou serviço. Para que a comparação possa ser feita é necessário que sejam definidos critérios relevantes para comparar as diversas opções. A opção selecionada é a que tem a menor distância geométrica até á solução ótima e ao mesmo tempo a maior distância geométrica até à pior solução [39].

Aplicando o TOPSIS para selecionar a framework de aprendizagem automática atribuiriam-se pesos aos critérios definidos. Os critérios "Popularidade", "Performance"e "Funcionalidade"tem um peso de 0.2 cada (20%); a "Utilização"0.1 (10%) e o "Formato Imagens"tem um peso de 0.3 (30%). Para cada critério e para cada framework foram atribuídos valores representados na tabela 3.1.

Tabela 3.1: Preenchimento inicial do TOPSIS

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens
Peso	0,20	0,20	0,20	0,10	0,3
Tensorflow	7	7	8	7	0
PyTorch	6	7	7	7	0
OpenCV	5	7	6	8	0
FastAI	6	7	7	7	1

Fazendo os cálculos propostos pela técnica e descritos no Anexo A obteve-se uma tabela de resultados 3.2, na qual a opção cujo valor for mais próximo que 1 é a que melhor se adequa de acordo com os critérios definidos.

Tabela 3.2: Tabela de Resultados TOPSIS

Framework	**Ci***
Tensorflow	0,58379854
PyTorch	0,359949403
OpenCV	0,113216268
FastAI	0,737939668

Como o valor de C_i^* da framework FastAI é o mais próximo de 1, esta será a framework explorada na implementação deste projeto.

3.1.5 Definição do conceito

Pretende-se assim que o sistema a desenvolver consiga com base nas imagens recebidas, identificar pedreiras, vias de acesso e a distancia entre elas. Isto permite entender a disposição que as pedreiras e vias de acesso têm entre si e sugerir aos funcionários responsáveis que a próxima inspeção as pedreiras cuja proximidade seja maior.

3.2 Valor

3.2.1 Definição de Valor

Para entender qual o valor que a solução pode ter é também necessário ter presente como pode ser definido o valor. Este pode ser definido como o quociente dos benefícios de um dado produto e dos custos ou uso desse produto [40]. Esta relação pode ser representada pela equação :

$$\text{Valor} = \frac{\text{Função}}{\text{Custo}} = \frac{\text{Benefícios Tangíveis e Intangíveis}}{\text{Custos de Tempo} + \text{Custos Financeiros} + \text{Custos Intangíveis}}$$

Desta forma, o Valor está diretamente dependente da perceção que o utilizador tem da utilidade e vantagens do produto sendo esta perceção positiva de acordo com o balanço de benefícios e custos.

3.2.2 Perceção de Valor

Apesar do valor poder ser representado com a equação da subsecção 3.2.1, a sua perceção é subjetiva. Pelo que diversos autores propoem diferentes modelos sendo um deles o modelo de [41]. Nesse modelo, são consideradas 3 perspetivas. Estas são o produto a qual mede a qualidade, caraterísticas e alternativas ao produto; o serviço que mede se o produto desempenha as funcionalidades e a relação a qual mede a relação entre o cliente e o negociante. Em cada uma dessas perspetivas podem ser consideradas os benefícios e sacrifícios. O modelo propõem que os aspetos considerados são os mencionados na tabela 3.3.

Tabela 3.3: Fatores da perceção de valor [41]

	Produto	**Serviço**	**Relação**
Benefícios	Alternativas, Qualidade, Personalização	Flexibilidade, Viabilidade, Fiabilidade, Competência Técnica	Imagem, Confiança, Solidariedade
Sacrifícios	Preço		Tempo, Esforço, Energia, Conflito

3.2.3 Proposta de Valor

A proposta de valar é a ferramenta que permite definir o valor que uma entidade tenciona fornecer a um cliente. Esta proposta deve fundamentar porque deve ser comprado um produto ou usado um serviço. Para além disso a proposta deve ainda ser feita de modo a que o cliente perceba o valor que poderá ganhar ou facilitar a resolução de um problema [42].

CANVAS da Proposta de Valor

Uma das ferramentas da proposta de valor é o canvas, que é uma framework que permite que produto ou serviço a desenvolver vai de encontro às necessidade do consumidor do final. Esta ferramenta serve ainda de ligação entre os segmentos de cliente e a proposta de valor [43]. O CANVAS da Proposta de Valor é constituído por duas partes "Proposta de Valor"e "Perfil do cliente", que estão representadas na tabela 3.4.

Tabela 3.4: Constituição do CANVAS da proposta de valor

Perfil do Cliente	Descrição
Ganhos	Os benefícios que o cliente espera ou precisa.
Dores	Riscos e complicações em fazer as tarefas.
Tarefas do Cliente	Tarefas ou problemas com que o cliente tem lidar.
Proposta de Valor	Descrição
Criadores de Ganho	Como o produto ou serviço oferece valor ao consumidor.
Atenuantes de Dor	Como o produto mitiga ou anula as complicações do cliente.
Produtos e Serviços	Aquilo que cria ganho, atenua a dor e beneficia o cliente.

Neste sentido, foi elaborado CANVAS da proposta de valor para o serviço que se pretende desenvolver.

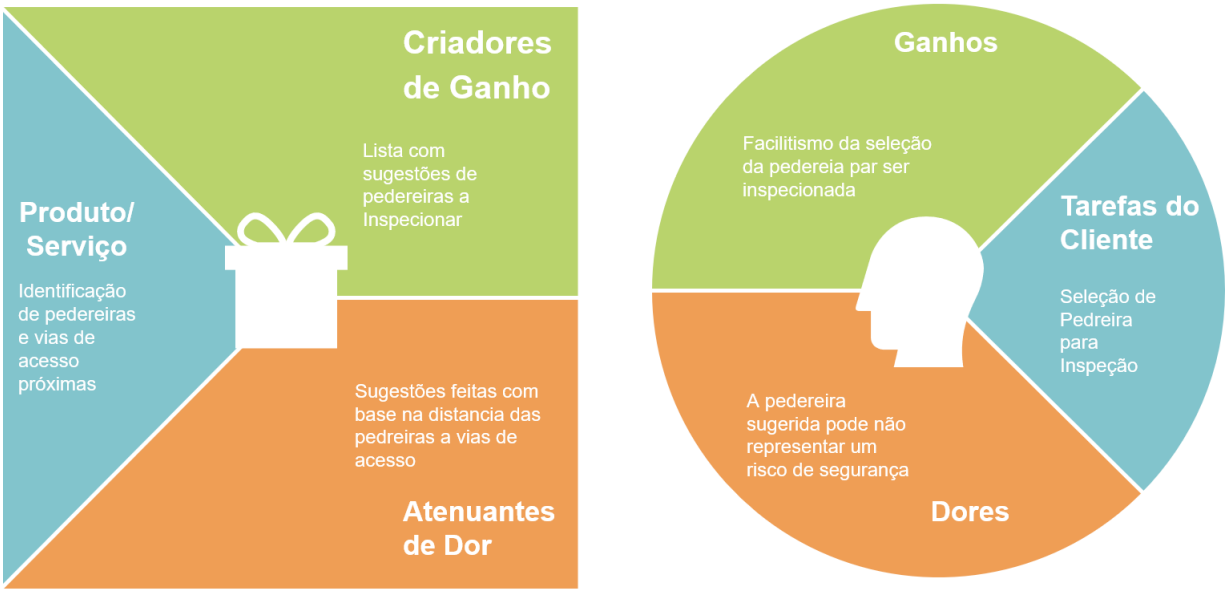

Figura 3.2: CANVAS da Proposta de Valor

Na figura 3.2, é possível ver que o cliente tem a tarefa de selecionar uma pedreira para que esta seja inspecionada. Nesta tarefa o cliente tem a "dor" de que ao escolher fazer a inspeção de uma pedreira, pode causar o atraso da inspeção uma pedreira com maior risco. Neste contexto o cliente, teria "ganho" em ter o processo de seleção da pedreira simplificado. Propõe-se então um "produto" que identifique em imagens aéreas pedreiras e vias de acesso. Este produto seria capaz de sugerir que pedreiras inspecionar sendo esta característica o "criador de ganho", Além isso este produto seria capaz de fazer as sugestões com base na distância às vias de acesso sendo esta característica o "atenuador de dor". Isto porque, uma pedreira próxima das vias de acesso representa o um maior risco.

3.2.4 Function Analysis System Technique

A técnica FAST serve para fazer uma representação visual das relações lógicas entre as funções do um produto ou serviço. Responde à perguntas, "Como ?" , "Porquê ?" e "Quando ?" tendo ainda o propósito de ilustrar e verificar se a solução vai de encontro às necessidades do cliente identificando funções em falta e/ou desnecessárias [44].

A figura 3.3 demonstra a análise feita às funções do sistema.

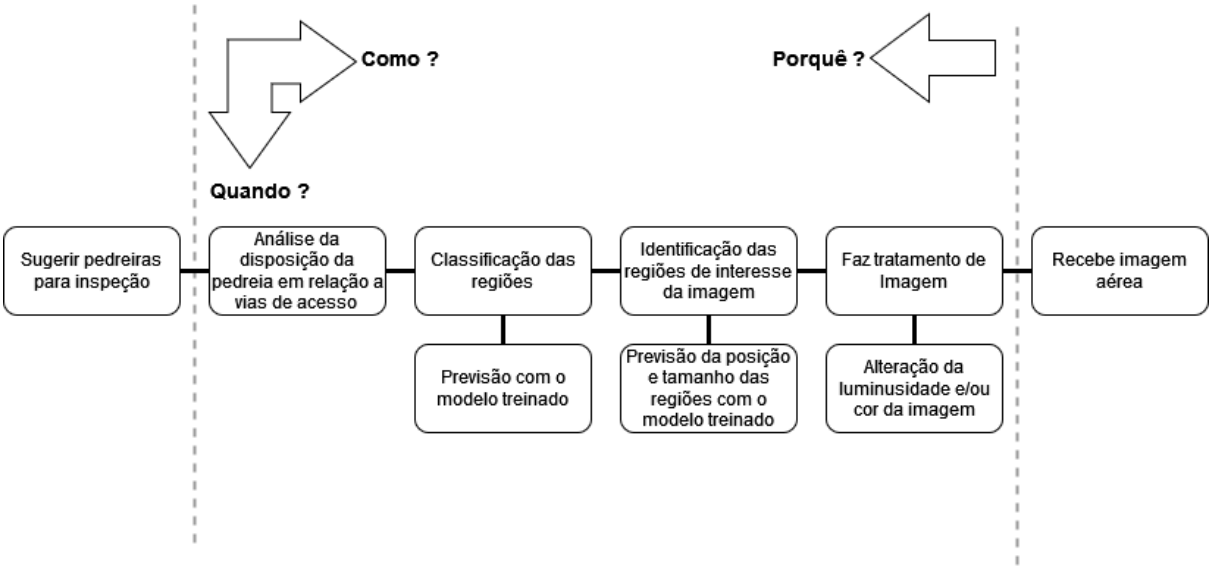

Figura 3.3: Diagrama FAST

Nesta análise FAST, existe no lado direito uma função de baixo nível "Recebe imagem aérea" e no lado esquerdo um a está mencionada uma função de alto nível "Sugerir de pedreiras para inspeção". Na parte central deste diagrama as funções que descrevem o processo de realizar a função de alto nível a partir da de baixo nível. Este diagrama está também organizado com a ordem lógica "Como ? - Porquê ?" sendo da parte central, mais à esquerda a função que define o principal objetivo do projeto ("Análise da disposição da pedreira em relação a vias de acesso"). As restantes funções descrevem como é atingido este objetivo sendo que as funções "Previsão com o modelo treinado", "Previsão da posição e tamanho com o modelo treinado" e "Alteração da luminosidade e/ou cor da imagem" estão mais a baixo porque são despoletadas quando outras acontecem.

Capítulo 4

Desenho da Solução

4.1 Identificação de Requisitos

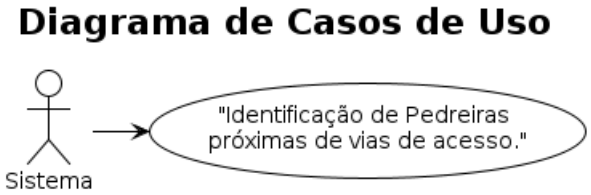

Figura 4.1: Diagrama de Casos de Uso

Tendo em conta o problema descrito em 1.2, foi identificado um requisito funcional para este sistema:

- **RF1:** Identificação de Pedreiras próximas de vias de acesso.

4.2 Análise de alternativas

Para a resolução de problemas de deteção de objetos existem dois tipos soluções. *Single-Stage Object Detection* e *Two-Stage Object Detection* representadas na figura 4.2. Destas duas, a *Single-Stage Detection* é mais antiga e usa um único modelo para localizar os objetos e para os classificar. Esta solução, de um modo geral, é a mais rápida das duas (em tempo de processamento) mas, ao mesmo tempo, é também a que faz mais previsões erradas. No caso da solução *Two-Stage Detection* é utilizado um modelo para cada uma das tarefas de localização e classificação dos objetos, fazendo menos previsões erradas com a contrapartida de um maior tempo de processamento [45].

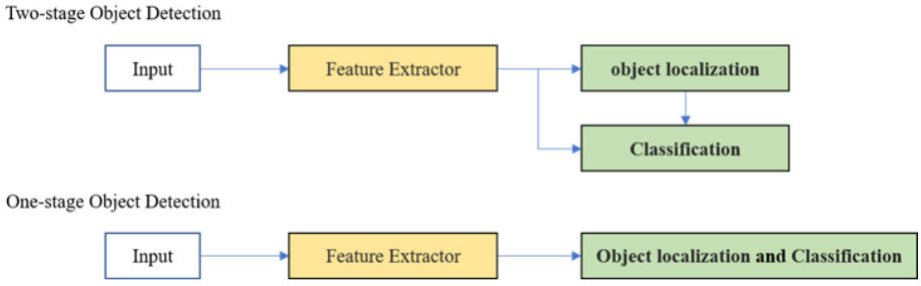

Figura 4.2: Arquiteturas da deteção *One-stage* e *Two-stage* [46]

Deste modo, é importante perceber qual destas arquiteturas de modelação se adequa melhor para o problema descrito. Ambas têm vantagens e desvantagens, e, neste caso em específico, ao optar pela opção que faz mais previsões corretas corre-se o risco de comprometer o tempo de execução e vice versa. Como tal, é preciso ter em conta o contexto do problema e, considerando que o objetivo é a sugestão de pedreiras próximas das vias de acesso, é mais importante obter previsões mais exatas, mesmo que estas demorem mais a serem processadas. Caso contrário, uma pedreira para a qual foram feitas previsões erradas relativamente à proximidade das vias de acesso pode acabar por ser inspecionada com menos prioridade ou frequência do que deveria.

Por isso neste projeto vai ser explorada a arquitetura *Two-Stage* para garantir a maior exatidão possível.

4.3 Solução Proposta

Através das alternativas mencionadas, propõe-se uma solução em que, com um conjunto de dados previamente fornecido, seja treinado um modelo que seja capaz de identificar, numa dada foto, pedreiras. O sistema proposto deverá ainda ser capaz de consultar um mapa de estradas livre para as zonas próximas à extremidade da pedreira. Com essa informação, o sistema deve ainda permitir a consulta visual da proximidade da pedreira e das vias de acesso.

A solução a implementar será constituída pela técnica de deteção de objetos que apresente melhor resultados, quando aplicada ao conjunto de dados. As versões mais recentes das redes descritas em 2.5 serão aplicadas ao problema e posteriormente testadas. É importante fazer a nota de que as redes *Alex-Net*, *VGG* e *ResNet* tem uma arquitetura *One-Stage* e a rede *U-Net* têm uma arquitetura *Two-Stage* pelo que, as redes *Alex-Net*, *VGG* e *ResNet* serão usadas como *backbone* da rede *U-Net*.

4.4 Metodologia de testes

Para avaliar se a solução implementada cumpre com os requisitos e que a qualidade do serviço fornecido não é inconsistente, a solução vai ser testada com métricas de avaliação de modelos e de acordo com a abordagem descrita no capítulo 6.

Capítulo 5

Implementação

Este capítulo da implementação visa expor as principais decisões tomadas no desenvolvimento do projeto e ainda explicar as partes mais relevantes da solução implementada. A explicação da solução é acompanhada de alguns excertos de código.

5.1 Ambiente de Desenvolvimento

Para desenvolver o sistema proposto e com o mínimo de acesso a uma GPU foi usado o Google Colab para o desenvolvimento de um Python Notebook que permita carregar os dados, treinar os modelos e ainda fazer inferência. De modo a que o Notebook em questão tivesse acesso aos dados, estes foram (com a devida autorização) persistidos na Google Drive sendo apenas necessária uma confirmação para que o Notebook tivesse o acesso.

Por fim, no sentido de acelerar o processo de treino e evitar as limitações do uso gratuito do Google Colab, foi ainda usado um GPU da entidade acolhedora. Deste modo foi possível fazer o treino dos modelos propostos para 50 épocas sem os imprevistos causados pelas limitações do Google Colab.

5.2 Segmentação de Pedreiras

Com o objetivo de identificar as margens das áreas exploradas pela exploração mineral foi fornecido um conjunto de imagens aéreas que contêm algumas das pedreiras de Portugal. O conjunto de dados fornecido continha ainda a informação de que áreas foram ou não exploradas, podendo ser usado numa solução que identifique as margens de qualquer pedreira com base no que foi fornecido.

5.2.1 Estrutura dos Dados

O conjunto de dados utilizado para o projeto, fornecido pela entidade acolhedora, foi subdividido em 3 pastas. A pasta "images" contem imagens aéreas das *batches* das pedreiras e áreas adjacentes. Estas *batches* têm formato .tif e 256 píxeis de altura e largura. O modelo de cores usado para estas imagens foi "rgbi" o qual contém respetivamente os canais "vermelho", "verde", "azul" e "intensidade". Neste formato cada pixel é constituído por 4 bytes (32 bits), sendo cada byte (8 bits) dedicados ao valor numérico associado a cada canal. Nos primeiros 3 bytes estão presentes 3 valores inteiros que representam a quantidade de cada uma das cores (os primeiros 3 canais) e intensidade pode ser visto como um número decimal entre 0 e 1 que mede a transparência.

Agrupando as *batches* do conjunto de teste é possível obter a imagem aérea da pedreira como um todo. A figura 5.1 a imagem aérea da pedreira do Bom Jesus.

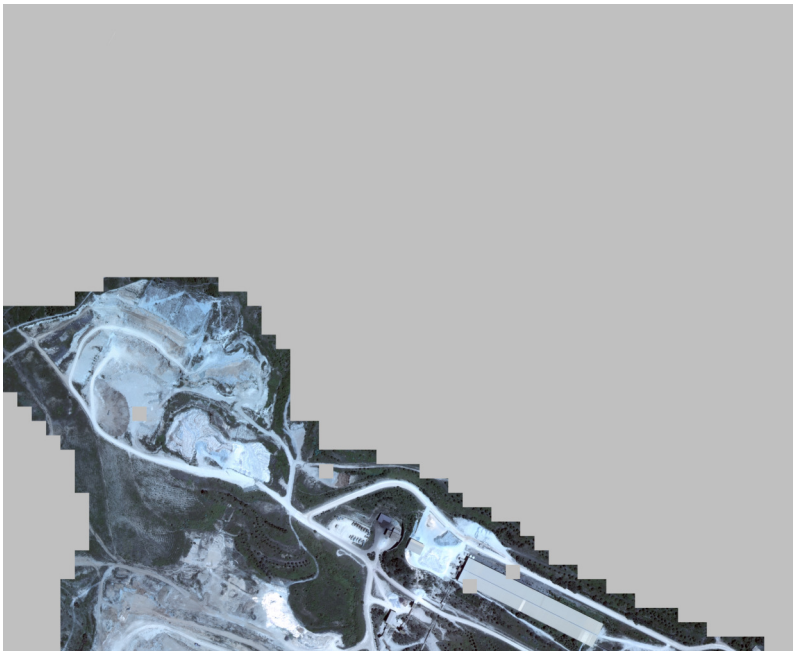

Figura 5.1: Imagem aérea da pedreira do Bom Jesus

Para cada ficheiro de imagem existe um ficheiro de texto (com o mesmo nome) e extensão "tfw". Por natureza estes ficheiros contém as informações associadas à respetiva imagem. Cada linha de texto desse ficheiro contêm um número pela seguinte ordem e significado REF:

- Distancia **horizontal** em metros que cada pixel representa. *(No conjunto de dados foi usado sempre o valor de 0.25 , ou seja, 25 cm.)*
- Rotação relativa ao eixo do **y**. *(No conjunto de dados foi usado sempre o valor 0.)*
- Rotação relativa ao eixo do **x**. *(No conjunto de dados foi usado sempre o valor 0.)*
- Distancia **vertical** em metros que cada pixel representa. *(No conjunto de dados foi usado sempre o valor de 0.25 , ou seja, 25 cm.)*
- Valor da coordenada do eixo do **x** do canto superior esquerdo da respetiva imagem.
- Valor da coordenada do eixo do **y** do canto superior esquerdo da respetiva imagem.
- **(Opcional)** Sistema de coordenadas utilizado. *(Não utilizado, mas para todos os ficheiros do conjunto de dados, as coordenadas das 2 linhas anteriores são apresentadas com o sistema* **epsg:3763***.)*
- **(Opcional)** Datum. *(Não utilizado.)*

Para além da pasta anterior existe ainda a pasta "labels" a que é constituída por imagens com as mesmas dimensões, nome e extensão das da pasta "images". No entanto, nestas imagens estão marcados em diferentes tons de preto as áreas para as quais existe ou não pedreira (quando "sobreposta" à respetiva imagem aérea). O modelo de cores destas imagens é Bitonal, uma vez que, existe apenas uma classe ("Pedreira") e são usados 2 tons de preto

para delinear as zonas que têm ou não pedreira. A técnica de encoding usada nestas imagens chama-se One-Hot e como tal as zonas de cada imagem que contêm pedreira têm o valor de cada pixel igual a 1 e as que não têm pedreira têm os píxeis com o valor 0. A figura 5.2 é agregação das máscaras associadas às *batches* da pedreiria demonstrada anteriormente. Esta figura tem marcado em preto a zona de pedreira e a cinzento o ambiente envolvente.

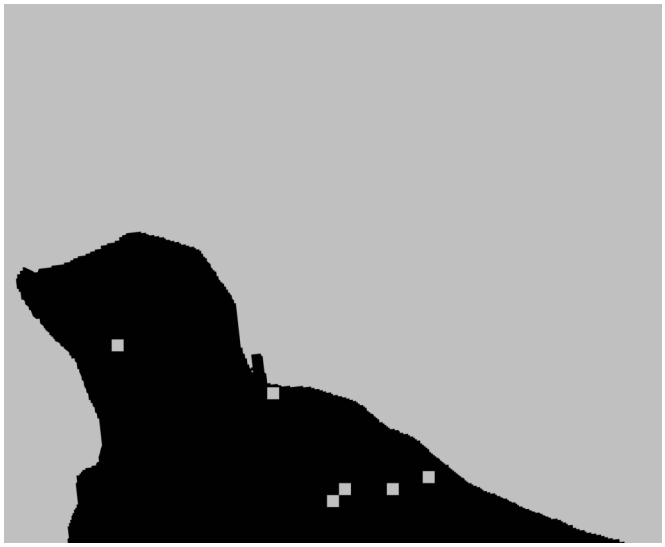

Figura 5.2: Máscara da pedreira do Bom Jesus

Por último, existe ainda a pasta "test", a qual tem outras 2 pastas "images" e "labels" com as mesmas características que as anteriormente explicadas.

A figura 5.3 demonstra um exemplo de como os ficheiros estão organizados no conjunto de dados usado.

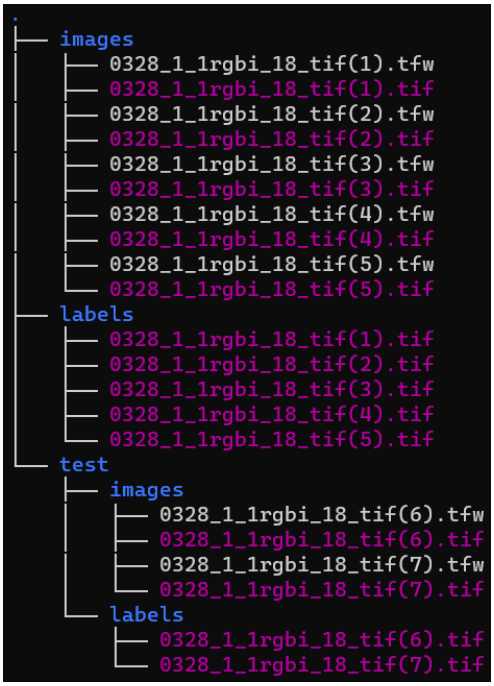

Figura 5.3: Estrutura da pasta que contem o Conjunto de Dados

5.2.2 Preparação dos Dados

Algumas ferramentas como o Tensorflow e PyTorch, que na subsecção 3.1.4 foram apresentadas como hipóteses para a implementação não têm suporte para imagens em formato .tif . Uma alternativa é a conversão do formato de imagem mas esta pode causar perda de informação/qualidade. A outra alternativa é o uso de uma ferramenta que tenha suporte para imagens .tif, tal como o FastAI.

No entanto o FastAI não permite o uso das implementações de modelos quando aplicados a imagens com 4 canais de cor. Para além disso, o FastAI não consegue destingir os 2 tons de preto nas imagens da pasta "labels" lendo-as a todas com um único tom de preto. A solução para estes problemas foi a conversão do modelo de cores.

Para ambos os casos foi implementada uma função de conversão usando a biblioteca "OpenCV" como ferramenta pois esta, permite a conversão entre vários modelos de cor.

```
1 import cv2
2
3 def convert_image(input_path, output_path) :
4     if not os.path.exists(output_path) :
5         image = cv2.imread(input_path, cv2.IMREAD_UNCHANGED)
6         step1 = cv2.cvtColor(image, cv2.COLOR_BGRA2RGB)
7         cv2.imwrite(output_path, step1)
```

Lista de Código 5.1: Conversão do modelo de cores das imagens aéreas

A função "**convert_image()**" demonstrada no excerto 5.1, recebe como argumentos o caminho para o ficheiro que se pretende ler e o caminho onde se pretende persistir o ficheiro convertido. O ficheiro é lido com função "**imread()**" e com o uso da constante "**IMREAD_UNCHANGED**" para não ignorar o canal alfa. Depois é feita a conversão propriamente dita com a função "**cvtColor()**" e com o uso da constante "**COLOR_BGRA2RGB**" usada para a especificar qual a conversão a fazer. Por último a imagem resultante é persistida.

```
1 import cv2
2
3 def convert_label(input_path, output_path) :
4     if not os.path.exists(output_path) :
5         label = cv2.imread(input_path, cv2.IMREAD_UNCHANGED)
6         step1 = cv2.cvtColor(label, cv2.COLOR_GRAY2RGB)
7         step2 = cv2.cvtColor(step1, cv2.COLOR_RGB2GRAY)
8         cv2.imwrite(output_path, step2)
```

Lista de Código 5.2: Conversão do modelo de cores das máscaras

Já função "**convert_label()**" demonstrada no excerto 5.2, é um tudo semelhante à anterior, diferindo apenas por ter 2 chamadas da função "**cvtColor()**" com as constantes "**COLOR_GRAY2RGB**" e "**COLOR_RGB2GRAY**" respetivamente.

5.2.3 Treino e Inferência

O conjunto de dados usado contem 15 pedreiras das quais uma foi usada para teste. Isto significa que foram usadas 2997 para teste e as restantes 7872 para os subconjuntos de

treino e validação. Destas 7872 imagens foram usadas 20% para validação, por isso aproximadamente 1574 imagens foram usadas para validação e as restantes 6298 foram usadas para treino.

Devido ao volume dados ser um pouco reduzido, e para evitar *Overfitting*, aplicou-se a técnica *Data Augmentation* para expandir o volume de dados. Nesse sentido foi usada a função "**aug_transforms()**" do FastAI. Esta função permite que se escolha que tipo de alterações podem ser feitas, e de que modo. Neste caso foram aplicadas as operações de inversão lateral (semelhante a uma espelhagem), rotação, zoom, alteração de luminosidade e contraste, e *warp*. As rotações, tem um limite máximo de 10º, o zoom tem um limite máximo de 10%, a escala máxima para a alteração de luminosidade é de 20% e a probabilidade de esta operação ser aplicada é de 75%. Por fim a operação de *warp*, que é o equivalente a posicionar mais atrás um dos lados da imagem, foi feita com um máximo de 20%.

Para a identificação de limites de pedreiras foram usadas redes neuronais convolucionais, mais concretamente a rede U-Net's. A U-net, seguindo uma arquitetura do tipo "Two-Stage", necessita que seja usada um outro modelo para fazer a classificação do objeto de estudo. Para este projeto foram usadas as redes AlexNet, VGG19 e ResNet34 como "backbone" da U-net. Estas foram as redes usadas, uma vez que, podem servir de classificadores e aplicam alguns dos conceitos fundamentais na área da análise de imagens.

Uma das decisões a tomar para configurar as redes é a escolha do valor de "*Learning Rate*"(LR). Este é um valor decimal entre 0 e 1 que gere o quanto os valores dos pesos das redes se alteram. Quão maior for o LR maior a diferença para cada um dos pesos entre 2 épocas seguidas. Para fazer esta escolha foi usada a função "**find_lr()**" do FastAI que faz um estudo da variação da função Loss em função da LR. Esta função retorna como sugestão o valor para o qual o decréscimo da Loss é maior.

A figura 5.4 resultante da função "**find_lr()**" mostra como varia a Loss em função de diferentes valores de LR para o modelo com a AlexNet como backbone.

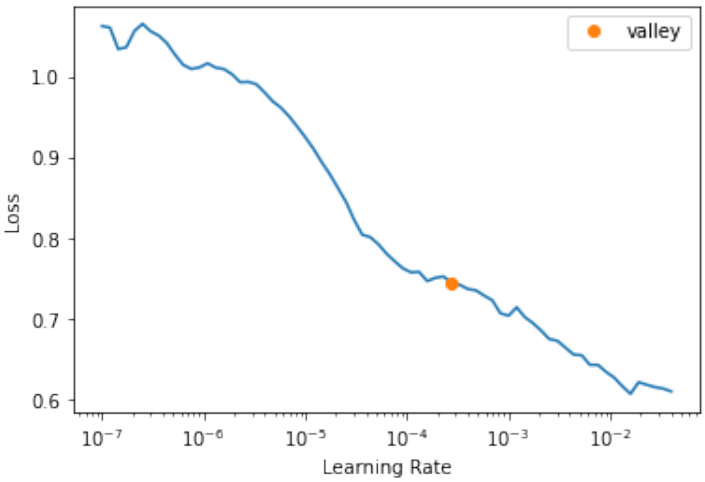

Figura 5.4: Alexnet LR

É possível observar que o valor sugerido é próximo de 1×10^{-4} e que a Loss diminui constantemente conforme diminui o valor de LR tomando a Loss valores entre 1 e 0.6 e o LR valores entre 1×10^{-7} e 1×10^{-1}.

A relação entre a Loss e a LR do modelo com o VGG19 como backbone está representada na figura 5.5 e foi obtida do mesmo modo que a anterior.

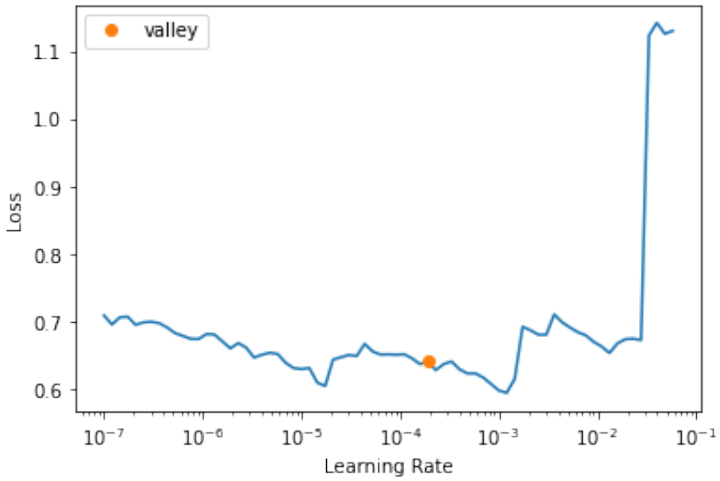

Figura 5.5: VGG19 LR

Neste caso foi também sugerido um valor próximo de 1×10^{-4} e observaram-se valores de Loss relativamente próximos de 0.7 enquanto a LR varia entre entre 1×10^{-7} e 1×10^{-1} e tendo valores próximos de 1 quando a LR está próxima de 1×10^{-1}. Esta subida abrupta de Loss pode ser um indício de que, com uma LR muito baixa, causando diferenças muito baixas nos seus pesos e *bias* entre épocas.

Do mesmo modo, está representada na figura 5.6 a relação entre Loss e LR para o caso da ResNet34.

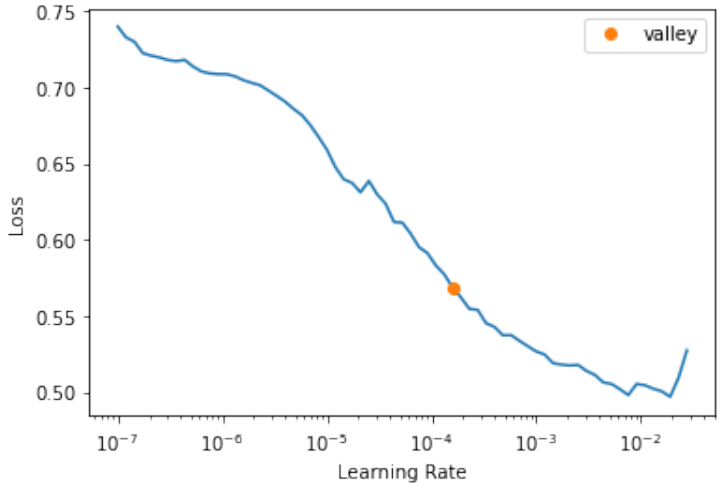

Figura 5.6: ResNet34 LR

Para os mesmos valores de LR este backbone foi o que apresentou os menores valores de Loss decrescendo de 0.75 a 0.5 conforme diminui a LR. O valor sugerido foi também próximo de 1×10^{-4}.

A figura 5.7 é resultante de previsões feitas de acordo com as *batches* da pedreira usada como exemplo. O modelo usado para obter estas previsões foi aquele que apresentou melhor desempenho, das arquiteturas avaliadas em 6.5.

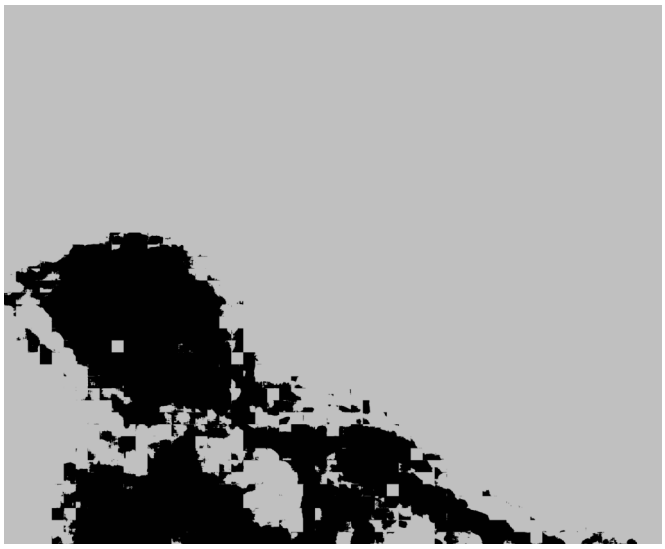

Figura 5.7: Previsão das margens da pedreira do Bom Jesus

5.3 Deteção de Estradas

No sentido de perceber qual a proximidade de estradas dos limites da pedreira identificados pelo sistema, existe a responsabilidade de detetar qual a rede de estradas presente de acordo com as informações geográficas presentes nos ficheiros .tfw .

```
from pyproj import Transformer
import requests
import os

def _get_API_key() :
    return # ..... .

def get_quarry_road_network(quarry_name, min_x, min_y, max_x, max_y) :
    roads_aerial_view = f"{quarry_name}_Roads.jpg"
    if not os.path.exists(roads_aerial_view) :
        upper_left_lat, bottom_right_lon = Transformer.from_crs("epsg
    :3763", "epsg:4326").transform(min_x, min_y)
        bottom_right_lat, upper_left_lon = Transformer.from_crs("epsg
    :3763", "epsg:4326").transform(max_x, max_y)

        # heigth, width = ... , ....
        url = f"https://www.mapquestapi.com/staticmap/v5/map?key={
    _get_API_key()}&boundingBox={upper_left_lat},{upper_left_lon},{
    bottom_right_lat},{bottom_right_lon}&size={width},{heigth}&type=dark"

        response = requests.get(url)
        print("Using MapQuestAPI.")

        with open(roads_aerial_view, "wb") as roads_file :
            roads_file.write(response.content)

    return roads_aerial_view
```

Lista de Código 5.3: Deteção de estradas

Para tal considera-se o nome das *batches* para a qual se pretende detetar a rede de estradas existente. A função "get_quarry_road_network()"presente no excerto 5.3, recebe como argumento esse nome, bem como os valores máximos e mínimos de latitude e longitude da pedreira em questão.

De seguida valida-se a existência de uma imagem com a informação pretendida na pasta do projeto (pasta que contem "images", "labels" e "test"). Caso o ficheiro exista persistido já há acesso à informação pretendida.

Caso contrário é feito um pedido do tipo GET com o protocolo HTTP. O pedido é feito numa API chamada MapQuestAPI a qual permite acesso a um mapa estático e retorna informação que permite criar uma imagem com a mesma localização que a respetiva imagem aérea. Esta solução permite minimizar a quantidade de pedidos necessária, uma vez que, estes só são feitos quando não existe persistida a informação que se quer.

As informações necessárias para construir o cabeçalho do pedido são os valores máximos e mínimos da latitude e longitude da área que se pretende e as dimensões pretendidas para a imagem resultante. Os valores da latitude e longitude podem ser obtidos com uma transformação dos argumentos da função. A figura 5.8 é resultante da consulta do mapa de estradas para a pedreira do Bom Jesus.

Figura 5.8: Mapa de estradas da zona da pedreira do Bom Jesus

5.4 Visualização da proximidade da pedreira e vias de acesso

De modo a perceber qual a proximidade das áreas de exploração mineira às estradas, é possível sobrepor as imagens do mapa de estradas e das previsões do modelo. A figura 5.9 demonstra visualmente este mesmo processo para a pedreira que tem sido considerada. Nesta figura a área com um tom mais escuro está prevista como pedreira. Neste caso é possível ver que existe uma via que passa na área de exploração.

Figura 5.9: Junção das previsão com o mapa de estradas da pedreira do Bom Jesus

Capítulo 6

Experimentação e Análise de Resultados

Este capítulo vai ter foco nas abordagens usadas para avaliar se a solução proposta resolve o problema em questão. Vão ser definidas as variáveis para as quais a solução vai ser avaliada e vão ser definidas as hipóteses nulas para os testes que vão ser feitos aos resultados obtidos. Conclusões serão retiradas através da análise dos resultados obtidos.

6.1 Objetivos de Teste

Vai ser testado se é possível detetar pedreiras e vias de acesso com base em imagens previamente fornecidas. Posteriormente pretende-se saber a distância que as pedreiras têm das vias de acesso para usar esse valor como indicador do risco para a segurança pública.

6.2 Variáveis de Teste

No contexto do problema e tendo em conta os requisitos definidos vai ser avaliada a exatidão da solução concebida em identificar pedreiras e vias de acesso tendo também em conta o tempo de execução que cada modelo poderá necessitar. Nesse sentido vão ser usadas as métricas de *accuracy* e IoU explicadas em 2.6 para medir a exatidão dos modelos implementados. Será considerado o número de segundos para medir o tempo necessário para gerar esses mesmos modelos.

6.3 Abordagem de Teste

Para obter uma amostra significativa de dados para os teste estatísticos, será repetida um determinado número de vezes para cada modelo implementando a técnica de avaliação de modelos *Cross-Validation*. Cada uma dessas amostras será usada no teste de ANOVA. Este foi o teste escolhido, uma vez que, serão considerados vários grupos para uma variável independente (modelo testado) e o teste será repetido para cada variável dependente (variável de teste).

6.3.1 Definição da Hipótese Nula

Tal como foi mencionado, as variáveis de teste a considerar são a *accuraccy*, IoU e o tempo de geração dos modelos (em segundos). Como tal para cada variável serão consideradas as hipóteses nula e alternativa. Em todas as hipóteses nulas (uma por variável) se assume que

não há diferença (significativa) da variância para as diferentes amostras. Por conseguinte, a hipótese alternativa será a de que as variâncias tenham todas valores diferentes.

6.4 Análise de Resultados

Após ter sido feita a implementação dos modelos referidos, foi feito o treino desses mesmos modelos. Deste modo foram obtidos os resultados de diferentes métricas para treinos feitos com 50 épocas. Como tal, estão demonstradas nas tabelas 6.1, 6.2, 6.3, 6.4 e 6.5, algumas estatísticas sobre os resultados, referentes ao treino do modelo U-net, com os *backbones* *AlexNet*, *VGG19* e *ResNet34*, para os valores da *Loss* de treino, *Loss* de validação, IoU, *accuracy* e tempo de treino, respetivamente. É possível fazer a consulta dos resultados obtidos para todas as 50 épocas do treino destes modelos no Anexo B.

Tabela 6.1: Estatísticas da *Loss* de treino dos modelos treinados

	AlexNet	VGG19	ResNet34
Média	0,155406	0,212926	0,163404
Variância	0,005165913	0,005460612	0,006160958
Desvio-Padrão	0,071874286	0,073895956	0,078491768

Tabela 6.2: Estatísticas da *Loss* de validação dos modelos treinados

	AlexNet	VGG19	ResNet34
Média	0,192932	0,257986	0,177316
Variância	0,002423769	0,001655329	0,004646779
Desvio-Padrão	0,049231788	0,040685732	0,068167286

Tabela 6.3: Estatísticas da IoU dos modelos treinados

	AlexNet	VGG19	ResNet34
Média	0,902106	0,870972	0,907142
Variância	0,00087018	0,000685496	0,001407898
Desvio-Padrão	0,02949881	0,026181979	0,037521967

Tabela 6.4: Estatísticas da *accuracy* dos modelos treinados

	AlexNet	VGG19	ResNet34
Média	0,920942	0,894194	0,925146
Variância	0,000657397	0,000593695	0,001034287
Desvio-Padrão	0,025639759	0,024365865	0,032160331

Tabela 6.5: Estatísticas do tempo de treino (em segundos) dos modelos

	AlexNet	VGG19	ResNet34
Média	475,52	1291,5	168,02
Variância	2,8896	53,37	0,0996
Desvio-Padrão	1,699882349	7,305477397	0,315594677

6.5 Testes Estatísticos

De acordo com a abordagem de testes proposta, os resultados relativos a cada uma das variáveis de teste propostas foram usados no teste de ANOVA executados com o auxílio do *Microsoft Excel*. Para todos os testes estatísticos, foram consideradas as hipóteses Nula e Alternativa da seguinte maneira:

- H_0 (Hipótese Nula) : as médias das amostras (de cada uma das varáveis de teste) **não têm** diferença significativa;
- H_1 (Hipótese Alternativa) : as médias das amostras (de cada uma das varáveis de teste) **têm** diferença significativa;

Para todos os T-test's vai ser considerada diferença significativa $\alpha = 0.05$.

6.5.1 Análise dos resultados de treino da IoU

Os resultados do teste de ANOVA para a IoU ou Índice de Jaccard do modelo U-net com os *backbones* referidos, estão apresentados na figura 6.1.

Anova: Single Factor - Jaccard Index

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
vgg19	50	43.5486	0.870972	0.000699
alexnet	50	45.1053	0.902106	0.000888
resnet34	50	45.3571	0.907142	0.001437

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.038383	2	0.019191	19.03864	4.44E-08	3.057621
Within Groups	0.148179	147	0.001008			
Total	0.186561	149				

Figura 6.1: Resultados do teste de ANOVA quando aplicados às amostras de IoU dos 3 *backbones*

Foram então obtidos os valores de $F = 19.03$ e $F_{crit} = 3.05$. Como $F > F_{crit}$, é possível rejeitar a hipótese nula. Neste contexto, isso significa que as médias dos valores de IoU para cada *backbone* são diferentes. De acordo com os valores do sumário os valores médios de IoU para os *backbones AlexNet* e *ResNet34* são ligeiramente maiores que o valor médio do VGG19, pelo que será feita uma comparação à média entre estas 2 amostras através do T-test. Para este T-test vão ser consideras as seguintes hipoteses:

- H_0 (Hipótese Nula) : a média da amostra da IoU do *backbone AlexNet* é **maior ou igual** à do *backbone ResNet34*;
- H_1 (Hipótese Alternativa) : a média da amostra da IoU do *backbone AlexNet* é **inferior** à do *backbone ResNet34*;

t-Test: Paired Two Sample for Means - Jaccard Index		
	alexnet	*resnet34*
Mean	0,902106	0,907142
Variance	0,000888	0,001437
Observations	50	50
Pearson Correlation	0,970622	
Hypothesized Mean Difference	0	
df	49	
t Stat	-3,09889	
P(T<=t) one-tail	0,001607	
t Critical one-tail	1,676551	
P(T<=t) two-tail	0,003214	
t Critical two-tail	2,009575	

Figura 6.2: Resultados do *T-test* quando aplicados às amostras de IoU dos *backbones AlexNet* e *ResNet34*

Tal como é possível verificar com a figura 6.2, foi obtido o valor de *p-value (one-tail)* = 0.0016. Como *p-value* $< \alpha$ é possível rejeitar a hipotese nula de que a média da IoU do conjunto da *AlexNet* é maior ou igual ao da *ResNet34*. Pode-se então concluir que *ResNet34* apresenta os maiores valores de IoU.

6.5.2 Análise dos resultados de treino da Accuracy

Os resultados do teste de ANOVA para a Accuracy do modelo U-net com os *backbones* referidos, estão apresentados na figura 6.3.

Anova: Single Factor - Accuracy						
SUMMARY						
Groups	*Count*	*Sum*	*Average*	*Variance*		
vgg19	50	44.7097	0.894194	0.000606		
alexnet	50	46.0471	0.920942	0.000671		
resnet34	50	46.2573	0.925146	0.001055		
ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.028186	2	0.014093	18.12973	9.17E-08	3.057621
Within Groups	0.114269	147	0.000777			
Total	0.142455	149				

Figura 6.3: Resultados do teste de ANOVA quando aplicados às amostras da *accuracy* dos 3 *backbones*

Foram então obtidos os valores de $F = 18.12$ e $F_{crit} = 3.05$. Como $F > F_{crit}$, é possível rejeitar a hipótese nula. Neste contexto, isso significa que as médias dos valores da accuracy para cada *backbone* são diferentes. De acordo com os valores do sumário os valores médios da Accuracy para os *backbones AlexNet* e *ResNet34* são maiores que o valor médio do VGG19, pelo que será feita uma comparação à média entre estas 2 amostras através do T-test. Para este T-test vão ser consideras as seguintes hipóteses:

- H_0 (Hipótese Nula) : a média da amostra da accuracy do *backbone AlexNet* é **maior ou igual** à do *backbone ResNet34*;
- H_1 (Hipótese Alternativa) : a média da amostra da accuracy do *backbone AlexNet* é **inferior** à do *backbone ResNet34*;

t-Test: Paired Two Sample for Means - Accuracy		
	alexnet	*resnet34*
Mean	0,920942	0,925146
Variance	0,000671	0,001055
Observations	50	50
Pearson Correlation	0,981286	
Hypothesized Mean Difference	0	
df	49	
t Stat	-3,43534	
P(T<=t) one-tail	0,000607	
t Critical one-tail	1,676551	
P(T<=t) two-tail	0,001214	
t Critical two-tail	2,009575	

Figura 6.4: Resultados do *T-test* quando aplicados às amostras da *accuracy* dos *backbones AlexNet* e *ResNet34*

Tal como é possível verificar com a figura 6.4, foi obtido o valor de *p-value (one-tail)* = 0.0006. Como *p-value* $< \alpha$ é possível rejeitar a hipotese nula de que a média da accuracy do conjunto da *AlexNet* é maior ou igual ao da *ResNet34*. Pode-se então concluir que *ResNet34* apresenta os maiores valores de accuracy.

6.5.3 Análise dos resultados do tempo de treino em segundos

Os resultados do teste de ANOVA para o tempo de treino do modelo U-net com os *backbones* referidos, estão apresentados na figura 6.5.

Anova: Single Factor - Time (Seconds)						
SUMMARY						
Groups	*Count*	*Sum*	*Average*	*Variance*		
vgg19	50	64575	1291.5	54.45918		
alexnet	50	23776	475.52	2.948571		
resnet34	50	8401	168.02	0.101633		
ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	33709782	2	16854891	879242.1	1.9E-300	3.057621
Within Groups	2817.96	147	19.1698			
Total	33712600	149				

Figura 6.5: Resultados do teste de ANOVA quando aplicados às amostras do tempo de treio (em segundos) dos 3 *backbones*

Foram então obtidos os valores de $F = 879242.1$ e $F_{crit} = 3.05$. Como $F > F_{crit}$, é possível rejeitar a hipótese nula. Neste contexto, isso significa que as médias dos valores do tempo de treino para cada *backbone* são diferentes. De acordo com os valores do sumário os valores médios do tempo de treino para os *backbones AlexNet* e *ResNet34* são muito menores que o valor médio do VGG19, pelo que será feita uma comparação à média entre estas 2 amostras através do T-test. Para este T-test vão ser consideras as seguintes hipoteses:

- H_0 (Hipótese Nula) : a média da amostra do tempo de treino (em segundos) do *backbone ResNet34* é **maior ou igual** à do *backbone AlexNet*;
- H_1 (Hipótese Alternativa) : a média da amostra do tempo de treino (em segundos) do *backbone ResNet34* é **inferior** à do *backbone AlexNet*;

t-Test: Paired Two Sample for Means - Time (seconds)		
	alexnet	*resnet34*
Mean	475,52	168,02
Variance	2,948571	0,101633
Observations	50	50
Pearson Correlation	0,167017	
Hypothesized Mean Difference	0	
df	49	
t Stat	1284,074	
P(T<=t) one-tail	7E-113	
t Critical one-tail	1,676551	
P(T<=t) two-tail	1,4E-112	
t Critical two-tail	2,009575	

Figura 6.6: Resultados do *T-test* quando aplicados às amostras do tempo de treino (em segundos) dos *backbones ResNet34* e *AlexNet*

Tal como é possível verificar com a figura 6.6, foi obtido o valor de *p-value (one-tail)* = 7×10^{-113}. Como *p-value* $< \alpha$ é possível rejeitar a hipotese nula de que a média do tempo de treino do conjunto da *ResNet34* é maior ou igual ao da *AlexNet*. Pode-se então concluir que *ResNet34* apresenta os menores valores valores de tempo de treino em segundos, por época.

Capítulo 7

Conclusão

O objetivo desta dissertação é a criação de um sistema de visão computacional que seja capaz de identificar pedreiras e consultar o mapa de estradas para essa zona. Este projeto foi inicialmente lançado pela Infraestruturas de Portugal e posteriormente adaptado para se enquadrar no contexto associado à elaboração do presente documento.

No sentido de entender melhor os conceitos necessários para a implementação da solução, foi feita uma revisão bibliográfica aos conteúdos de investigação científica relativos à visão computacional. Foram por isso analisados conceitos subjacentes como modelos de aprendizagem em profundidade e avaliação de modelos. Foram ainda analisados três casos de estudo no sentido de detetar os aspetos mais relevantes das suas soluções.

Foi feita uma análise de valor ao projeto, no sentido de perceber que problemas poderá o sistema resolver ou atenuar. Esta análise permite também melhorar o que se pretende desenvolver melhorando o valor que a solução terá.

A solução para o problema foi desenhada, identificando os principais requisitos do projeto e comparando alternativas. Neste caso, percebeu-se que existem duas alternativas para a arquitetura da solução sendo estas o reconhecimento de imagem *One-Stage* ou *Two-Stage*. Optar por uma destas alternativas implicava comprometer ou o tempo de treino do modelo ou a exatidão dos resultados sendo que neste caso procura-se não comprometer os resultados dado que, estes podem ter um impacto na segurança pública.

Foram ainda expostos os aspetos mais relevantes da implementação do projeto, começando pelo do ambiente de desenvolvimento. Foram explicadas caraterísticas do conjunto de dados e as decisões relativas ao treino da solução foram descritas explicando-se como foi escolhida a *Learning Rate*. Foi ainda demonstrado o método usado para fazer a consulta do mapa de estradas de acordo com as informações geográficas associadas às imagens aéreas.

Também foram referidas as abordagens para avaliar a qualidade da solução. Foi explicado que as métricas de avaliação de modelos *accuracy* e *Intersection over Union*, foram usadas para avaliar as previsões dos modelos a treinar, e também foi considerado o tempo em segundos para os avaliar quanto ao tempo de treino. Tendo em conta que se pretende que vários modelos sejam treinados, foi usado o teste estatístico ANOVA para cada uma das variáveis de estudo, tentando perceber a variância que o uso de diferentes algoritmos poderá causar. Para perceber qual das alternativas se destacava das restantes, foi ainda realizado para cada variável o T-test.

Com os testes feitos apuraram-se resultados do modelo U-net com diferentes *backbones*. De acordo com os resultados dos testes de ANOVA feitos para as variáveis de teste *IoU*, *accuracy* e tempo de treino (em segundos) foi possível concluir as variâncias de todos os

backbones são diferentes. Isto permitiu colocar hipóteses relativas às médias dos backbones que melhor se destacavam. Neste caso em particular viu-se que os *backbones* alexnet e resnet34 tinham melhores resultados em todas as variáveis de teste, concluindo-se ainda após os *t-test*'s que a resnet34 é, dos *backbones* testados, o que melhor se adequa ao problema. De um modo geral, é possível concluir que a segmentação de imagens para detetar as margens áreas de exploração mineral é minimamente viável alcançando com a solução proposta 94% de *Intersection over Union* e aproximadamente 95% de *accuracy*. É também possível analisar de modo visual a proximidade às vias de acesso, tal como foi demonstrado em 5.4.

7.1 Trabalho Futuro

No futuro, seria uma mais valia o desenvolvimento de uma heurística que meça a distância entre as margens das áreas de exploração mineral e outros pontos de interesse como vias de acesso e edifícios de modo a usar essa distância como índice de risco. Uma outra situação a explorar seria a adição de imagens de outras pedreiras ao conjunto de dados, em particular daquelas que tiverem vias de acesso e pontos de interesse nas suas proximidades.

Bibliografia

[1] . «Guião de Pedreiras». pt. Em: (), p. 104. url: https://www.dgeg.gov.pt/media/wzrozybv/gui%C3%A3o-das-pedreiras.pdf.

[2] *CRISP DM Help Overview*. url: https://www.ibm.com/docs/en/spss-modeler/SaaS?topic=dm-crisp-help-overview.

[3] *CRISP DM methodology*. Jun. de 2020. url: https://www.sv-europe.com/crisp-dm-methodology/.

[4] *What is Computer Vision? | IBM*. url: https://www.ibm.com/topics/computer-vision.

[5] Pulkit Sharma. *Image Classification vs Object Detection vs Image Segmentation*. en. Ago. de 2019. url: https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81 (acedido em 26/02/2022).

[6] Pulkit Sharma. *Image Segmentation | Types Of Image Segmentation*. en. Abr. de 2019. url: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/ (acedido em 20/02/2022).

[7] *What Is Deep Learning? | How It Works, Techniques & Applications*. en. url: https://www.mathworks.com/discovery/deep-learning.html (acedido em 20/02/2022).

[8] *VIDEO – What is a GPU?* en-us. Ago. de 2020. url: https://www.ibm.com/cloud/blog/video-what-is-a-gpu.

[9] Jason Brownlee. *What is the Difference Between Test and Validation Datasets?* en-US. Jul. de 2017. url: https://machinelearningmastery.com/difference-test-validation-datasets/.

[10] *What is Overfitting in Machine Learning? [And How to Avoid It]*. url: https://www.v7labs.com/blog/overfitting,%20https://www.v7labs.com/blog/overfitting.

[11] IBM Cloud Education. *What are neural networks?* url: https://www.ibm.com/cloud/learn/neural-networks.

[12] *Artificial neural network architecture*. en. url: https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051 (acedido em 20/02/2022).

[13] Jiwon Jeong. *The Most Intuitive and Easiest Guide for CNN*. en. Jul. de 2019. url: https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480 (acedido em 20/02/2022).

[14] Sumit Saha. *A comprehensive guide to Convolutional Neural Networks - the eli5 way*. Dez. de 2018. url: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[15] Muhamad Yani, S Irawan e Casi Setianingsih. «Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail». Em: *Journal of Physics: Conference Series* 1201 (mai. de 2019), p. 012052. doi: 10.1088/1742-6596/1201/1/012052.

[16] Alex Krizhevsky, Ilya Sutskever e Geoffrey E. Hinton. «ImageNet classification with deep convolutional neural networks». en. Em: *Communications of the ACM* 60.6 (mai. de 2017), pp. 84–90. issn: 0001-0782, 1557-7317. doi: 10.1145/3065386. url: https://dl.acm.org/doi/10.1145/3065386 (acedido em 20/02/2022).

[17] Karen Simonyan e Andrew Zisserman. «VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION». en. Em: (2015), p. 14. url: https://arxiv.org/pdf/1409.1556.pdf.

[18] Kaiming He et al. «Deep Residual Learning for Image Recognition». en. Em: *arXiv:1512.03385 [cs]* (dez. de 2015). arXiv: 1512.03385. url: http://arxiv.org/abs/1512.03385 (acedido em 20/02/2022).

[19] Jerry Wei. *AlexNet: The Architecture that Challenged CNNs*. en. Set. de 2020. url: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951.

[20] Jerry Wei. *VGG Neural Networks: The Next Step After AlexNet*. en. Jul. de 2019. url: https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c.

[21] Jason Brownlee. *What is a Confusion Matrix in Machine Learning*. en-US. Nov. de 2016. url: https://machinelearningmastery.com/confusion-matrix-machine-learning/ (acedido em 20/02/2022).

[22] Adrian Rosebrock. *Intersection over Union (IoU) for object detection*. en-US. Nov. de 2016. url: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (acedido em 20/02/2022).

[23] Jason Brownlee. *Loss and Loss Functions for Training Deep Learning Neural Networks*. en-US. Jan. de 2019. url: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.

[24] D Yudin et al. «Software for roof defects recognition on aerial photographs». Em: *Journal of Physics: Conference Series* 1015 (mai. de 2018), p. 032152. doi: 10.1088/1742-6596/1015/3/032152. url: https://doi.org/10.1088/1742-6596/1015/3/032152.

[25] *Pixel Connectivity - MATLAB & Simulink*. url: https://www.mathworks.com/help/images/pixel-connectivity.html (acedido em 20/02/2022).

[26] V A Pavlov e M A Galeeva. «Detection and recognition of objects on aerial photographs using convolutional neural networks». Em: *Journal of Physics: Conference Series* 1326.1 (out. de 2019), p. 012038. doi: 10.1088/1742-6596/1326/1/012038. url: https://doi.org/10.1088/1742-6596/1326/1/012038.

[27] Joseph Redmon et al. «You Only Look Once: Unified, Real-Time Object Detection». en. Em: *arXiv:1506.02640 [cs]* (mai. de 2016). arXiv: 1506.02640. url: http://arxiv.org/abs/1506.02640.

[28] Joseph Redmon e Ali Farhadi. «YOLO9000: Better, Faster, Stronger». en. Em: *arXiv:1612.08242 [cs]* (dez. de 2016). arXiv: 1612.08242. url: http://arxiv.org/abs/1612.08242.

[29] Joseph Redmon e Ali Farhadi. «YOLOv3: An Incremental Improvement». en. Em: *arXiv:1804.02767 [cs]* (abr. de 2018). arXiv: 1804.02767. url: http://arxiv.org/abs/1804.02767.

[30] Hafiz Suliman Munawar et al. «Application of Deep Learning on UAV-Based Aerial Images for Flood Detection». Em: *Smart Cities* 4.3 (2021), pp. 1220–1242. issn: 2624-6511. doi: 10.3390/smartcities4030065. url: https://www.mdpi.com/2624-6511/4/3/65.

[31] Dieu Tien Bui et al. «A novel deep learning neural network approach for predicting flash flood susceptibility: A case study at a high frequency tropical storm area». Em: *Science of The Total Environment* 701 (2020), p. 134413.

[32] Laura Lopez-Fuentes et al. «Multi-modal Deep Learning Approach for Flood Detection.» Em: *MediaEval* 17 (2017), pp. 13–15.

[33] Quanlong Feng, Jiantao Liu e Jianhua Gong. «Urban flood mapping based on unmanned aerial vehicle remote sensing and random forest classifier—A case of Yuyao, China». Em: *Water* 7.4 (2015), pp. 1437–1455.

[34] Ismail Elkhrachy. «Flash flood hazard mapping using satellite images and GIS tools: a case study of Najran City, Kingdom of Saudi Arabia (KSA)». Em: *The Egyptian Journal of Remote Sensing and Space Science* 18.2 (2015), pp. 261–278.

[35] Mahyat Shafapour Tehrany et al. «Flood susceptibility assessment using GIS-based support vector machine model with different kernel types». Em: *Catena* 125 (2015), pp. 91–101.

[36] G.David Hughes e Don C. Chafin. «Turning new product development into a continuous learning process». Em: *Journal of Product Innovation Management* 13.2 (1996), pp. 89–104. issn: 0737-6782. doi: https://doi.org/10.1016/0737-6782(95)00112-3. url: https://www.sciencedirect.com/science/article/pii/0737678295001123.

[37] Kristel Dewulf. «Sustainable Product Innovation: The Importance of the Front- End Stage in the Innovation Process». Em: *Advances in Industrial Design Engineering* (mar. de 2013), pp. 139–166. doi: 10.5772/52461. url: https://www.researchgate.net/publication/281199416_Sustainable_Product_Innovation_The_Importance_of_the_Front-_End_Stage_in_the_Innovation_Process.

[38] Peter Koen. *Front End Innovation - What is the New Concept Development (NCD) model?* url: http://frontendinnovation.com/fei/what-is-the-new-concept-development-ncd-model.

[39] Robert Soczewica. *What is TOPSIS?* en. Set. de 2021. url: https://robertsoczewica.medium.com/what-is-topsis-b05c50b3cd05.

[40] Lisa Wood. «Brands and brand equity: definition and management». Em: *Management Decision* 38.9 (jan. de 2000). Publisher: MCB UP Ltd, pp. 662–669. issn: 0025-1747. doi: 10.1108/00251740010379100. url: https://doi.org/10.1108/00251740010379100 (acedido em 27/02/2022).

[41] Jozée Lapierre. «Customer-perceived value in industrial contexts». Em: *Journal of Business & Industrial Marketing* 15.2/3 (jan. de 2000). Publisher: MCB UP Ltd, pp. 122–145. issn: 0885-8624. doi: 10.1108/08858620010316831. url: https://doi.org/10.1108/08858620010316831 (acedido em 27/02/2022).

[42] Alexandra Twin. *Value Proposition: Why Consumers Should Buy a Product or Use a Service*. en. Jul. de 2020. url: https://www.investopedia.com/terms/v/valueproposition.asp (acedido em 20/02/2022).

[43] B2B International. *What is the Value Proposition Canvas?* en. url: https://www.b2binternational.com/research/methods/faq/what-is-the-value-proposition-canvas/.

[44] . *Function Analysis and System Technique - FAST diagram*. en-us. Fev. de 2020. url: https://extrudesign.com/function-analysis-and-system-technique-fast-diagram/ (acedido em 22/02/2022).

[45] Chan-Yun Yang et al. «Deep Learning Based Real-Time Facial Mask Detection and Crowd Monitoring». Em: *Computing and Informatics* (nov. de 2021). url: https://

www.researchgate.net/publication/356127377_Deep_Learning_Based_Real-Time_Facial_Mask_Detection_and_Crowd_Monitoring.

[46] Chunxu Li. url: https://www.researchgate.net/figure/Comparison-of-one-stage-and-two-stage-object-detection_fig1_356127377.

Apêndice A

Cálculos auxiliares da técnica de comparação TOPSIS

De acordo com os pesos, critérios e opções definidas em 3.1.4, foi preenchida a tabela A.1.

Tabela A.1: TOPSIS: Matriz Inicial

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens
Peso	0,20	0,20	0,20	0,10	0,3
Tensorflow	7	7	8	7	0
PyTorch	6	7	7	7	0
OpenCV	5	7	6	8	0
FastAI	6	7	7	7	1

De seguida foram calculados na tabela A.2 o quadrado do valores preenchidos. Para cada critério (coluna) os quadrados foram todos somados tendo sido ainda calculada a raíz quadrada de cada uma das somas.

Tabela A.2: TOPSIS: Matriz com os valores ao quadrado

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens
Tensorflow	49	49	64	49	0
PyTorch	36	49	49	49	0
OpenCV	25	49	36	64	0
FastAI	36	49	49	49	1
Soma	146	196	198	211	1
Raiz Quadrada	12,0830	14,0000	14,0712	14,5258	1,0000

Como demonstrado na tabela A.3 o passo seguinte os valores inicialmente preenchidos são divididos pela já calculada raíz (da respetiva coluna).

Tabela A.3: TOPSIS: Matriz Normalizada Pesada

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens
Tensorflow	0,5793	0,5000	0,5685	0,4819	0,0000
PyTorch	0,4966	0,5000	0,4975	0,4819	0,0000
OpenCV	0,4138	0,5000	0,4264	0,5507	0,0000
FastAI	0,4966	0,5793	0,5793	0,5793	0,0828

Posteriormente cada valor da matriz normalizada (passo anterior) foi multiplicado pelo peso (da respetiva coluna) representado na tabela A.4.

Tabela A.4: TOPSIS: Matriz com valores multiplicados pelo peso de cada coluna

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens
Tensorflow	0,1159	0,1000	0,1137	0,0482	0,0000
PyTorch	0,0993	0,1000	0,0995	0,0482	0,0000
OpenCV	0,0828	0,1000	0,0853	0,0551	0,0000
FastAI	0,0993	0,1159	0,1159	0,0579	0,0248

A seguir, foram calculados os maiores valores para cada coluna ($A*$), os quais estão mencionados na tabela A.5.

Tabela A.5: TOPSIS: Determinar a solução ideal positiva $A*$

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens
Tensorflow	0,1159	0,1000	0,1137	0,0482	0,0000
PyTorch	0,0993	0,1000	0,0995	0,0482	0,0000
OpenCV	0,0828	0,1000	0,0853	0,0551	0,0000
FastAI	0,0993	0,1159	0,1159	0,0579	0,0248
Maior	0,1159	0,1159	0,1159	0,0579	0,0248

Foram também calculados os menores valores para cada coluna (A'), os quais estão mencionados na tabela A.6.

Tabela A.6: TOPSIS: Determinar a solução ideal negativa A'

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens
Tensorflow	0,1159	0,1000	0,1137	0,0482	0,0000
PyTorch	0,0993	0,1000	0,0995	0,0482	0,0000
OpenCV	0,0828	0,1000	0,0853	0,0551	0,0000
FastAI	0,0993	0,1159	0,1159	0,0579	0,0248
Menor	0,0828	0,1000	0,0853	0,0482	0,0000

Depois disso, foi feita a separação da solução ideal positiva, que consiste em calcular o quadrado da subtração dos valores pelo A* de cada coluna. Neste passo, tal como mostra a tabela A.7, são ainda calculadas somas de cada linha e a raíz quadrada de cada uma dessas somas ($S*$).

Tabela A.7: TOPSIS: Separação da solução ideal positiva

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens	Soma	Raiz Quadrada
Tensorflow	0,0000	0,0003	0,0000	0,0001	0,0006	0,0010	0,0311
PyTorch	0,0003	0,0003	0,0003	0,0001	0,0006	0,0015	0,0388
OpenCV	0,0011	0,0003	0,0009	0,0000	0,0006	0,0029	0,0539
FastAI	0,0003	0,0000	0,0000	0,0000	0,0000	0,0003	0,0166

A semelhança, foi feita a separação da solução ideal negativa, que consiste em calcular o quadrado da subtração dos valores pelo A' de cada coluna. Neste passo, tal como mostra a tabela A.8, são ainda calculadas somas de cada linha e a raíz quadrada de cada uma dessas somas (S').

Tabela A.8: TOPSIS: Separação da solução ideal negativa

Critério	Popularidade	Performance	Funcionalidade	Utilização	Formato Imagens	Soma	Raiz Quadrada
Tensorflow	0,0011	0,0000	0,0008	0,0000	0,0000	0,0019	0,0436
PyTorch	0,0003	0,0000	0,0002	0,0000	0,0000	0,0005	0,0218
OpenCV	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0069
FastAI	0,0003	0,0003	0,0009	0,0001	0,0006	0,0022	0,0466

Por fim, na tabela A.9, para cada opção (linha da tabela) foi calculado o valor de C_i^* que pode ser dado pela fórmula :

$$C_i^* = \frac{S'}{S' + S*}$$

Tabela A.9: TOPSIS: Proximidade relativa para a solução ideal

Framework	**Ci***
Tensorflow	0,58379854
PyTorch	0,359949403
OpenCV	0,113216268
FastAI	0,737939668

Apêndice B

Tabelas Treino

epoch	train_loss	valid_loss	IoU	acc_camvid	time
0	0.3374	0.3234	0.8322	0.8560	08:00
1	0.3101	0.2953	0.8402	0.8656	08:01
2	0.2914	0.2905	0.8428	0.8701	08:01
3	0.2932	0.2900	0.8327	0.8652	08:00
4	0.2777	0.2914	0.8515	0.8733	07:58
5	0.2672	0.2640	0.8574	0.8823	07:57
6	0.2488	0.2571	0.8610	0.8850	07:57
7	0.2425	0.2538	0.8660	0.8894	07:56
8	0.2335	0.2493	0.8678	0.8914	07:57
9	0.2346	0.2561	0.8697	0.8897	07:56
10	0.2306	0.2407	0.8777	0.8986	07:55
11	0.2170	0.2298	0.8788	0.9029	07:55
12	0.2086	0.2184	0.8857	0.9077	07:55
13	0.1925	0.2141	0.8887	0.9108	07:55
14	0.1932	0.2245	0.8782	0.9041	07:55
15	0.1831	0.1956	0.8970	0.9172	07:55
16	0.1787	0.2038	0.8989	0.9178	07:54
17	0.1673	0.1918	0.9010	0.9203	07:55
18	0.1573	0.1849	0.9063	0.9249	07:55
19	0.1597	0.1877	0.9019	0.9223	07:55
20	0.1556	0.1902	0.9008	0.9215	07:54
21	0.1398	0.1964	0.9072	0.9252	07:54
22	0.1514	0.1794	0.9048	0.9249	07:54
23	0.1299	0.1777	0.9095	0.9285	07:54
24	0.1394	0.1707	0.9138	0.9314	07:54
25	0.1282	0.1793	0.9140	0.9311	07:54
26	0.1183	0.1643	0.9189	0.9354	07:54
27	0.1232	0.1674	0.9151	0.9329	07:55
28	0.1188	0.1663	0.9142	0.9328	07:55
29	0.1147	0.1671	0.9182	0.9346	07:55
30	0.1024	0.1591	0.9227	0.9387	07:54
31	0.1121	0.1599	0.9216	0.9378	07:54
32	0.1045	0.1573	0.9229	0.9392	07:55
33	0.0999	0.1565	0.9249	0.9403	07:55
34	0.1001	0.1597	0.9236	0.9391	07:55
35	0.0994	0.1495	0.9274	0.9426	07:55

36	0.0905	0.1517	0.9271	0.9423	07:55
37	0.0926	0.1482	0.9267	0.9423	07:55
38	0.0933	0.1523	0.9278	0.9429	07:55
39	0.0960	0.1513	0.9276	0.9425	07:55
40	0.0881	0.1477	0.9291	0.9441	07:55
41	0.0839	0.1486	0.9292	0.9439	07:56
42	0.0862	0.1485	0.9304	0.9448	07:55
43	0.0833	0.1492	0.9297	0.9443	07:55
44	0.0864	0.1472	0.9302	0.9447	07:56
45	0.0828	0.1488	0.9305	0.9450	07:55
46	0.0806	0.1474	0.9303	0.9448	07:55
47	0.0799	0.1474	0.9305	0.9450	07:55
48	0.0845	0.1471	0.9306	0.9450	07:55
49	0.0801	0.1482	0.9305	0.9449	07:56

Tabela B.1: Tabela Treino AlexNet

epoch	train_loss	valid_loss	IoU	acc_camvid	time
0	0.3865	0.3879	0.7979	0.8226	21:50
1	0.3584	0.3648	0.8139	0.8321	21:52
2	0.3462	0.3422	0.8118	0.8433	21:51
3	0.3392	0.3276	0.8241	0.8531	21:50
4	0.3383	0.3128	0.8346	0.8591	21:44
5	0.3112	0.3114	0.8319	0.8594	21:41
6	0.3383	0.2997	0.8401	0.8644	21:36
7	0.3125	0.3129	0.8410	0.8628	21:32
8	0.2868	0.3051	0.8300	0.8619	21:28
9	0.3101	0.2984	0.8422	0.8684	21:28
10	0.2864	0.2871	0.8508	0.8762	21:25
11	0.2702	0.2824	0.8510	0.8783	21:25
12	0.2415	0.2770	0.8590	0.8826	21:23
13	0.2685	0.2679	0.8628	0.8869	21:25
14	0.2532	0.2744	0.8622	0.8843	21:23
15	0.2563	0.2648	0.8632	0.8879	21:24
16	0.2427	0.2696	0.8595	0.8864	21:24
17	0.2287	0.2667	0.8636	0.8902	21:25
18	0.2315	0.2668	0.8708	0.8933	21:25
19	0.2303	0.2533	0.8694	0.8935	21:25
20	0.2191	0.2532	0.8704	0.8954	21:25
21	0.2210	0.2427	0.8792	0.9013	21:25
22	0.1885	0.2534	0.8769	0.9005	21:26
23	0.2014	0.2372	0.8822	0.9047	21:25
24	0.1917	0.2426	0.8762	0.9008	21:27
25	0.1842	0.2471	0.8777	0.9020	21:27
26	0.1731	0.2384	0.8778	0.9026	21:27
27	0.1813	0.2496	0.8869	0.9084	21:28
28	0.1789	0.2407	0.8830	0.9060	21:28
29	0.1659	0.2302	0.8879	0.9085	21:28

30	0.1863	0.2297	0.8858	0.9085	21:30
31	0.1578	0.2291	0.8893	0.9109	21:30
32	0.1737	0.2232	0.8894	0.9113	21:31
33	0.1651	0.2236	0.8902	0.9123	21:28
34	0.1530	0.2231	0.8920	0.9130	21:31
35	0.1543	0.2248	0.8900	0.9117	21:31
36	0.1392	0.2302	0.8933	0.9145	21:31
37	0.1423	0.2315	0.8892	0.9121	21:31
38	0.1416	0.2253	0.8940	0.9149	21:32
39	0.1345	0.2162	0.8958	0.9163	21:32
40	0.1433	0.2260	0.8946	0.9153	21:33
41	0.1383	0.2202	0.8954	0.9161	21:33
42	0.1400	0.2364	0.8959	0.9164	21:34
43	0.1335	0.2273	0.8966	0.9171	21:34
44	0.1344	0.2191	0.8960	0.9170	21:34
45	0.1399	0.2153	0.8967	0.9174	21:36
46	0.1377	0.2155	0.8976	0.9178	21:36
47	0.1271	0.2206	0.8967	0.9172	21:35
48	0.1311	0.2307	0.8961	0.9165	21:36
49	0.1313	0.2236	0.8960	0.9165	21:35

Tabela B.2: Tabela Treino VGG19

epoch	train_loss	valid_loss	IoU	acc_camvid	time
0	0.3595	0.3488	0.8066	0.8397	02:47
1	0.3255	0.3084	0.8339	0.8611	02:48
2	0.2936	0.3123	0.8403	0.8606	02:48
3	0.2785	0.2890	0.8532	0.8751	02:49
4	0.2885	0.2783	0.8540	0.8758	02:49
5	0.2839	0.3150	0.8242	0.8600	02:49
6	0.2648	0.2672	0.8639	0.8863	02:48
7	0.2708	0.2548	0.8594	0.8854	02:48
8	0.2701	0.2833	0.8581	0.8844	02:48
9	0.2479	0.2357	0.8733	0.8964	02:48
10	0.2388	0.2494	0.8786	0.8986	02:48
11	0.2346	0.2497	0.8588	0.8883	02:48
12	0.2283	0.2202	0.8836	0.9038	02:48
13	0.2157	0.2091	0.8880	0.9103	02:48
14	0.2090	0.2064	0.8924	0.9119	02:48
15	0.2013	0.2011	0.8911	0.9123	02:48
16	0.1914	0.1856	0.9014	0.9210	02:48
17	0.1955	0.1911	0.9000	0.9206	02:48
18	0.1966	0.1886	0.8939	0.9171	02:48
19	0.1711	0.1731	0.9049	0.9247	02:48
20	0.1721	0.1719	0.9094	0.9269	02:48
21	0.1531	0.1675	0.9109	0.9297	02:48
22	0.1502	0.1621	0.9161	0.9332	02:48
23	0.1475	0.1628	0.9136	0.9319	02:48

24	0.1403	0.1667	0.9133	0.9295	02:48
25	0.1395	0.1564	0.9205	0.9361	02:48
26	0.1314	0.1658	0.9110	0.9307	02:48
27	0.1222	0.1414	0.9260	0.9417	02:48
28	0.1266	0.1420	0.9254	0.9419	02:48
29	0.1190	0.1344	0.9301	0.9451	02:47
30	0.1221	0.1350	0.9312	0.9452	02:48
31	0.1102	0.1465	0.9234	0.9406	02:48
32	0.1100	0.1237	0.9351	0.9493	02:48
33	0.1032	0.1280	0.9360	0.9493	02:48
34	0.1026	0.1270	0.9350	0.9494	02:48
35	0.0953	0.1177	0.9395	0.9526	02:48
36	0.0896	0.1149	0.9407	0.9534	02:48
37	0.0948	0.1176	0.9401	0.9530	02:48
38	0.0880	0.1179	0.9402	0.9532	02:48
39	0.0840	0.1143	0.9429	0.9553	02:48
40	0.0812	0.1093	0.9451	0.9571	02:48
41	0.0832	0.1093	0.9452	0.9571	02:48
42	0.0855	0.1066	0.9463	0.9581	02:48
43	0.0829	0.1088	0.9454	0.9574	02:48
44	0.0776	0.1094	0.9454	0.9574	02:48
45	0.0828	0.1092	0.9455	0.9574	02:48
46	0.0751	0.1080	0.9456	0.9576	02:48
47	0.0752	0.1073	0.9465	0.9582	02:48
48	0.0823	0.1084	0.9461	0.9578	02:48
49	0.0773	0.1088	0.9460	0.9578	02:48

Tabela B.3: Tabela Treino ResNet34