Universidade da Beira Interior
Departamento de Informática

Nº 125 - 2021: Fashion Analysis from Surveillance Data taken from UAVs

Elaborado por:

Daniel Afonso de Resende

Orientador:

Professor Doutor Hugo Pedro Martins Carriço Proença

11 de julho de 2022

Agradecimentos

A conclusão e desenvolvimento deste trabalho não seria possível sem a ajuda essencial do meu orientador de projeto Prof. Doutor Hugo Proença, que sempre me ajudou quando precisava.

Agradeço também aos colegas que conheci no *Soft Computing and Image Analysis Laboratory* (SOCIALAB) por me terem acolhido de braços abertos e também por terem aproveitado cada oportunidade que eu lá estava para me ajudarem com tudo o que eu precisasse.

Gostaria também de agradecer aos meus colegas de curso e amigos mais próximos pela companhia que me fizeram durante esta caminhada de três anos.

Queria também por último, agradecer aos meus pais e à minha família, que sempre me apoiaram. Sem eles não seria a pessoa que sou hoje e nada disto seria possível.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

AP	*Average Precision*
AssA	*Association accuracy*
CNN	*Convolutional Neural Network*
COCO	*Common Objects in Context*
DL	*Deep Learning*
FN	*False negative*
FP	*False positive*
HOG	*Histogram of Oriented Gradients*
HOTA	*Higher Order Tracking Accuracy*
IA	*Inteligência Artificial*
IoU	*Intersection over Union*
LocA	*Location accuracy*
ML	*Machine Learning*
mAP	*Mean Average Precision*
MOT	*Multiple Object Tracking*
NMS	*Non-max Suppression*
OCR	*Object Character Recognition*
R-CNN	*Region-based Convolutional Neural Network*
RGB	*Red Green Blue*
RNA	*Rede Neuronal Artificial*
RPN	*Region Proposal Network*
ReLU	*Rectified Linear Unit*
RoI	*Region of Interest*
SOCIALAB	*Soft Computing and Image Analysis Laboratory*
SORT	*Simple Online and Real-time Tracking*
SOT	*Single Object Tracking*

SSD *Single-Shot Detector*

SVM *Support Vector Machine*

TP *True positive*

UBI Universidade da Beira Interior

YOLO *You Only Look Once*

Capítulo

1

Introdução

Neste primeiro capítulo são definidos os objetivos principais deste trabalho e documento. Assim, o capítulo encontra-se divido entre três secções, na qual a **secção 1.1** esclarece a ideia por trás da escolha de projeto e o contexto do documento, a **secção 1.2** elucida os objetivos do trabalho e do documento e finalmente, a **secção 1.3**, onde se encontra uma breve descrição da estrutura deste documento.

1.1 Enquadramento

O atual documento, integrante do projeto *Fashion Analysis from Surveillance Data taken from UAVs*, foi desenvolvido no domínio da unidade curricular de Projeto de Licenciatura e enquadra-se no terceiro ano da Licenciatura em Engenharia Informática na Universidade da Beira Interior (UBI).

1.2 Objetivos e Motivação

Os objetivos deste trabalho eram os seguintes:

- Implementação de um modelo de segmentação para caracterização de roupas.
- Implementação de uma base de dados de estilos de roupa observados.

1.3 Organização do Documento

De maneira a resumir e explicar o trabalho feito, o documento encontra-se organizado da seguinte maneira:

- **Capítulo 1** – onde é apresentado o projeto e os seus objetivos, a motivação por trás da escolha e do esforço, o enquadramento do trabalho e uma breve descrição do documento.
- **Capítulo 2** – descreve vários conceitos relacionados com *Inteligência Artificial* (IA) de um modo mais aprofundado, incluíndo a história por trás da mesma e algumas disciplinas que a constitui. São introduzidos também alguns modelos de exemplo. São mencionadas, no final, algumas técnicas de avaliação dos modelos.
- **Capítulo 3** – no qual são descritas as ferramentas e tecnologias utilizadas no trabalho além dos *datasets* utilizados para avaliar e treinar os modelos.
- **Capítulo 4** – são avaliados e interpretados os resultados da avaliação dos modelos obtidos.
- **Capítulo 5** – onde o trabalho feito é revisto e a implementação é analisada ao comparar os objetivos pretendidos com os realizados e, por último, uma breve discussão de trabalhos futuros e melhorias ao nível do sistema desenvolvido.

Capítulo

2

Técnicas de Aprendizagem Profunda

2.1 Introdução

A IA tem evoluído de modo exponencial recentemente, podendo ser aplicada com o objetivo facilitar e, até mesmo resolver vários problemas que possam existir no nosso dia-a-dia. Aplicada a identificação de objetos e reconhecimento facial, a IA tem-se tornado cada vez mais um tópico de interesse entre muitos e, como tal, tornou-se muito mais acessível. Tal é possível graças a algorítmos de *Deep Learning*, que são inspirados no modo de ganhar conhecimento do ser humano.
Ao longo deste capítulo serão explicados conceitos essenciais para entender a deteção de objetos e pessoas e também formas de deteção dos mesmos.

2.2 Visão Computacional

A Visão Computacional é um campo de IA que permite computadores e sistemas a obtenção de informação útil a partir de imagens, vídeos e outros tipos de *inputs* com o propósito do sistema tomar ações ou oferecer recomendações baseadas na informação obtida. A Visão Computacional é, portanto, o ramo de IA que faz com que os computadores possam "ver"e entender o que "veem".
As primeiras experiências começaram em 1959, quando alguns cientistas tentaram perceber como o cérebro de um gato reagia quando eram introduzidos certos *inputs*. No caso de imagens, o cérebro começava por analisar primeiro extremidades simples na imagem como, por exemplo, linhas retas. Dado isto,

os cientistas perceberam que o processamento de imagem deve começar sempre na análise de formas simples como as linhas retas. [2]
Pouco tempo depois, a tecnologia que permitia os computadores interpretar imagens foi desenvolvida. De seguida, em 1974, foi introduzida a tecnologia *Object Character Recognition* (OCR) cuja função era reconhecer texto numa imagem. Desde então, a tecnologia OCR foi introduzida no dia-a-dia na forma de processamento de documentos, reconhecimento de placas de identificação de automóveis e muitos outros.
Já em 1982, o cientista David Marr introduziu algoritmos que permitia a deteção de arestas, curvas, cantos e outras formas básicas. O reconhecimento de objetos com IA foi o tópico de interesse em 2000 e, um ano depois, as primeiras aplicações de algoritmos para reconhecimento facial tomaram lugar. [2]
Hoje em dia, a Visão Computacional é bastante aplicada de várias maneiras sendo a mais comum o reconhecimento facial e de objetos.

2.3 Inteligência Artificial

A IA é a ciência e a engenharia da criação de máquinas inteligentes, especialmente programas de computador inteligentes. [3]
A mesma pode ser definida também como a habilidade de uma máquina interpretar/aprender segundo *inputs* a ela fornecidos e usá-los de maneira a alcançar certas tarefas pré-definidas de um modo dinâmico/flexível. [3]
Para tal, a IA emprega técnicas tais como *Deep Learning* (DL) e *Machine Learning* (ML).

2.3.1 *Machine Learning*

ML é um ramo da IA que recorre a diversos algoritmos capazes de processar dados, aprender dos mesmos e aplicar o conhecimento ganho. Este método é inspirado na idea de que as máquinas, assim como o ser humano, podem aprender segundo o que lhes é dado, podendo identificar padrões e tomar decisões com pouca/nenhuma intervenção humana mesmo sem serem programadas para tal. [4] [5]
Para a aprendizagem, o ML depende de alguns algoritmos de aprendizagem diferenciados, cuja utilização deve ser ditada pela abordagem preferida, isto é, que tipos de dados vão ser introduzidos e posteriormente reproduzidos e que tipo de problema deve a máquina resolver.

2.3.1.1 Aprendizagem Supervisionada

A **Aprendizagem Supervisionada** consiste em "treinar"a máquina segundo um conjunto de dados pré-definido e identificado. Com o conhecimento ganho do conjunto de dados de treino, a máquina pode aplicar o que "aprendeu"em novos conjuntos de dados que nunca lhe foram introduzidos anteriormente.

2.3.1.2 Aprendizagem Não-Supervisionada

Ao contrário da **Aprendizagem Supervisionada** (2.3.1.1), a **Aprendizagem Não-Supervisionada** consiste em "treinar"a máquina também mas, desta vez, recorrendo a um conjunto de dados pré-definido mas não identificado. Como tal, não é necessária mão-de-obra muito intensiva para a criação de uma base de dados de treino, dado que os dados da mesma não necessitam de *labels*. Assim, a **Aprendizagem Não-Supervisionada** é um método mais versátil, ou seja, a máquina adapta-se e constroi uma rede de estruturas "escondidas"que se transformam durante a fase de treino.

2.3.1.3 Aprendizagem por reforço

A **Aprendizagem por reforço** é inspirada na maneira de como o ser humano aprende. Muitas das vezes, a aprendizagem é fruto de tentativa e erro. Para tal, o método da **Aprendizagem por reforço** coloca a máquina num ambiente onde as ações favoráveis são "recompensadas". Este método é bastante utilizado atualmente em casos como o filtro de *spam* no *e-mail*.

2.3.1.4 *Deep Learning*

O DL é tipo de ML que pode utilizar a Aprendizagem Supervisionada (2.3.1.1) e a Aprendizagem Não-Supervisionartrda (2.3.1.2) individualmente ou ambas simultaneamente. O DL tem sido utilizado cada vez mais recentemente para atacar certos tipos de problemas nas áreas de visão computacional e processamento de linguagem natural e, tal é possível devido ao recente aumento de capacidade computacional. Esta técnica utiliza uma grande quantidade de dados para formar estruturas complexas através de uma estrutura de algoritmos: a *Rede Neuronal Artificial* (RNA). Ao remontar uma rede de neurónios, o design da RNA é inspirado na estrutura biológica do cérebro humano e, portanto, dá origem a um sistema que aprende como tal. Para obter bons resultados, um modelo de DL deve ser treinado com uma enorme quantidade de dados. [6]

Um grande exemplo da aplicação de um modelo de DL é o AlphaGo da Google. Este programa possui uma RNA instruída apenas com dados sobre o jogo de tabuleiro Go e conseguiu ganhar contra o campeão mundial. [6]

2.4 Deteção de Objetos

A deteção de objetos é uma parte muito importante da visão computacional que procura encontrar objetos de certos tipos em imagens e vídeos digitais. Para tal, são desenvolvidos algoritmos que fornecem o que se quer encontrar nas imagens e nos vídeos (em forma de *frame*), devolvendo coordenadas que, juntas, integram uma caixa (*bounding box*) que encapsule os objetos de interesse na imagem/*frame*. [7]
A deteção de objetos pode ser feita de modo tradicional, utilizando técnicas de processamento de imagem tradicionais tais como o *Histogram of Oriented Gradients* (HOG), ou podem ser utilizadas as RNAs. Os métodos de deteção de objetos mais recentes costumam recorrer as RNAs, dado que estas são mais eficientes, robustas e versáteis. [7]
Os métodos de deteção de objetos mais recentes podem ser categorizados em dois tipos distintos: *one-stage detectors* e *two-stage detectors*. [7]
Os *object detectors* são algoritmos que extraem *features* a partir de uma imagem/*frame* digital. Os mesmos são responsáveis pela seleção da área onde serão detetados e classificados objetos numa imagem.

2.5 Segmentação de Objetos

Muitas das vezes não basta apenas saber, de um modo geral e relativamente pouco preciso, onde os objetos se encontram numa imagem. É também necessário saber a região que o objeto de interesse ocupa. A segmentação de objetos consiste na identificação de certas regiões/segmentos correspondentes a objetos de interesse numa imagem. Esta tecnologia é de grande importância para várias áreas tais como o processamento de imagem de satélite e a condução autónoma de carros [8].
A segmentação de objetos pode ser dividida em dois tipos:

- **Segmentação Semântica** - É atribuída uma classe a cada pixel da imagem. Como por exemplo na figura 2.1, é possível observar que cada pixel da estrada pertence à classe *Road* e é representado (na parte de baixo) pela cor roxa. É de notar também que as viaturas pertencem todas à mesma classe, mesmo possuíndo algumas características diferentes [8].

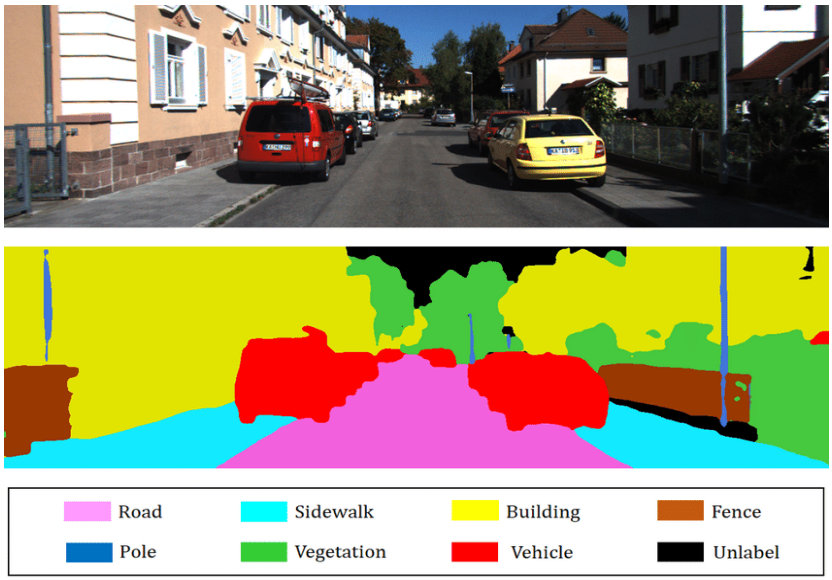

Figura 2.1: Exemplo de segmentação semântica.

- **Segmentação de Instância** - Multíplos objetos da mesma classe são representados em instâncias distintas. Na figura 2.2, é possível observar que os animais pertencem à mesma classe mas fazem parte de instâncias diferentes [8].

Figura 2.2: Exemplo de segmentação de instância.

2.6 *Tracking* de objetos

O *tracking* de objetos é uma parte importante de várias aplicações de modelos de DL atualmente. O mesmo consiste na atribuição de identificadores únicos para cada objeto de interesse numa *frame* e, em *frames* posteriores, manter a indentificação nos mesmos objetos, mesmo que os mesmos se tenham deslocado. [9]
O *tracking* de objetos pode ser definido de maneiras diferentes para certos casos de uso:

- ***Multiple Object Tracking* (MOT)** - Identificação de múltiplos objetos de classes diferentes e atribuição de um valor de identificação único a cada *bounding box* detetada.
- ***Single Object Tracking* (SOT)** - Em vez de identificar vários objetos, é focado apenas um certo objeto. O mesmo deve depois ser encontrado no resto das *frames*.

2.6.1 *SORT*

O *Simple Online and Real-time Tracking* (SORT) é um modelo de *tracking* de objetos do tipo MOT e foi um dos primeiros a alcançar tal feito. [9]
O SORT é composto por quatro componentes de modo sequencial:

1. ***Detection*** - São detetados os objetos de interesse (que queremos seguir), recorrendo a modelos de deteção de objetos tais como o *Faster Region-based Convolutional Neural Network* (R-CNN) (2.7.6) [9];
2. ***Estimation*** - A partir dos objetos detetados da frame anterior, novas posições dos mesmos são estimadas para a nova frame [9];
3. ***Data association*** - Com os dados obtidos é avaliada a previsão do passo anterior recorrendo a um *threshold* e, utilizando assim, a expressão de cálculo da *Intersection over Union* (IoU).
4. ***Creation and Deletion of Track Identities*** - Comparar o IoU calculado com um certo valor. Se inferior, um certo objeto é dado como desaparecido e, se o mesmo voltar a aparecer, é-lhe atribuído um novo identificador [10];

2.6.2 *DeepSORT*

Um grande problema do SORT (2.6.1) são as colisões de objetos (relativamente ao ponto de vista da câmera). No SORT, um alto número de colisões

leva à criação de muitas identidades novas, isto é, a perda de *tracking* de objetos afetados por colisões com outros. De modo a resolver o problema, o *DeepSORT* melhora a componente de associação de identificadores dos objetos ao recorrer a um modelo pré-treinado que incorpora a deteção de *features*. [10]

2.6.3 *StrongSORT*

Baseado no *DeepSORT*, o *StrongSORT* é um *upgrade* com algumas mudanças. Os métodos anteriores dependiam de componentes de alto esforço computacional. O *StrongSORT* torna o algoritmo mais eficiente e eficaz ao substituir a *backbone* (*Convolutional Neural Network* (CNN)) por uma *ResNeSt50* e a introdução de uma função de atualização de *features*. Com estas melhorias foi capaz de alcançar o recorde de avaliação segundo a métrica *Higher Order Tracking Accuracy* (HOTA) nos datasets *MOT17* e *MOT20*. [11]

2.7 Redes Neuronais Artificiais

Nesta secção são mencionados vários modelos de DL utilizados hoje e é explicado o modo de funcionamento de cada um. São também explicadas maneiras de avaliar os vários tipos de modelos mencionados.

2.7.1 *CNN*

A CNN é um tipo de RNA. Muito utilizada em aplicações de reconhecimento e deteção de objetos, a CNN foi especialmente desenvolvida para o processamento de imagens digitais e é um dos modelos de *deep learning* (2.3.1.4) mais populares atualmente.
A CNN é composta por uma camada de entrada, uma de saída e várias "escondidas"no meio. [12]
A camada *Convolutional* é a primeira camada da CNN e pode ser seguida de mais camadas *Convolutional* ou *Pooling*, sendo a *Fully-connected* a última camada. [12]

2.7.1.1 *Convolutional layer*

A camada *Convolutional*, tal como o nome indica, exerce um processo conhecido como a convolução e é essencial para uma CNN, necessitando de uma entrada de dados, um filtro e um mapa de *features*. Quando se quer inserir

imagens, os dados são de três dimensões: largura da imagem, altura da imagem e profundidade de cor. Para o caso de uma imagem *Red Green Blue* (RGB), a profundidade é três para cada píxel dado que, para o modelo RGB, cada um é composto por um valor de vermelho, um valor de verde e um de azul. Para encontrar *features* (*e.g.* arestas), é utilizado um *kernel* que procura pelas mesmas na imagem. O filtro/*kernel* é, tipicamente, uma matriz (3 x 3) e, para encontrar as *features*, é então aplicado na imagem de modo a aproveitar todos os píxeis da imagem. Por exemplo, numa imagem de tamanho (4 x 4), um filtro/*kernel* de dimensão (3 x 3) é aplicado quatro vezes à imagem, sendo efetuado o produto entre as duas matrizes. [12]

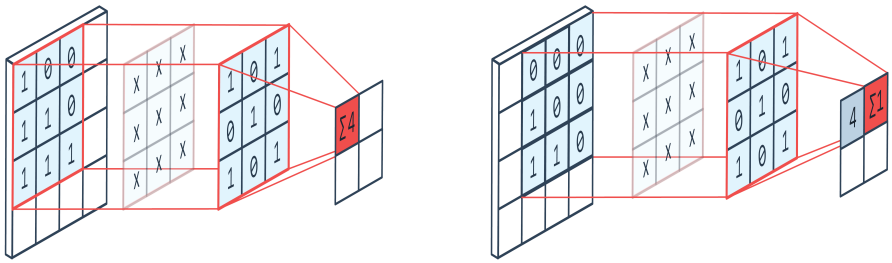

Figura 2.3: Exemplo de aplicação do filtro numa convolução

Como pode ser observado na figura 2.3, o filtro de tamanho (3 x 3) é aplicado numa matriz de tamanho (4 x 4). Na operação à esquerda encontra-se o produto entre o filtro e a primeira matriz da imagem seguida da segunda operação à direita, onde o filtro move-se segundo um *stride* (que neste caso é 1) e efetua o segundo produto. No final, o resultado é uma nova matriz com todas as operações feitas. [12]
Após cada operação de convolução, é aplicada uma transformação *Rectified Linear Unit* (ReLU) no resultado.

2.7.1.2 *Pooling layer*

O objetivo da camada de *Pooling* é a redução do tamanho do resultado da camada anterior. Para tal, a camada de *Pooling* aplica uma função de agregação de valores utilizando um filtro, cujo output será o resultado desta nova camada. Existem dois tipos de camadas de *Pooling*: *Average pooling* e *Max pooling*. [12]
Como é possível observar na figura 2.4, no caso do *average pooling*, o filtro move-se pela matriz de *input* da camada atual, devolvendo a média dos valores da matriz resultante do filtro. Já no caso do *max pooling*, o filtro move-se

também pela matriz de *input* da camada atual e, em vez da média, devolve o valor máximo da matriz resultante do filtro. [12]

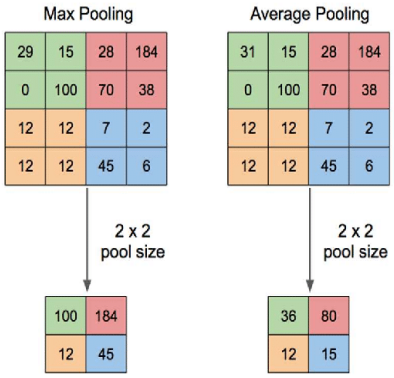

Figura 2.4: Exemplo das operações *max pooling* e *average pooling*

2.7.1.3 *Fully-connected layer*

Tal como o nome indica, a *Fully-connected layer* é uma camada composta de células interconectadas, isto é, células da camada anterior estão ligadas às células da camada de saída. Recorrendo às *features* resultantes de camadas anteriores, a função desta camada é a classificação da imagem, representada por uma probabilidade entre 0 e 1. [12]

2.7.2 *You Only Look Once*

Em 2015, o *Fast* R-CNN era um dos modelos mais rápidos na àrea de reconhecimento de objetos, demorando entre 2 a 3 segundos para conseguir efetuar a deteção numa imagem. Para aplicações de um modelo de reconhecimento de objetos em tempo real, um atraso destes impede a aplicação do modelo em tempo real. De modo a solucionar este problema, o *You Only Look Once* (YOLO) foi apresentado e procurava ser uma alternativa a modelos de reconhecimento de objetos da altura, tal como o *Fast* R-CNN (2.7.5) mencionado, sendo muito mais rápido e capaz de 45 *frames* por segundo. [13] [14] [15]
Outros modelos da altura, tais como o *Fast* R-CNN (2.7.5), "processam"os dados de *input* várias vezes, enquanto que o YOLO só o faz uma vez. [15]
O YOLO é *one-stage* ou seja, não funciona por regiões propostas. Em vez disso, o YOLO divide a imagem de *input* por células e insere tudo numa matriz que, no caso da figura 2.5, é (13 x 13). [14] [15] [16]

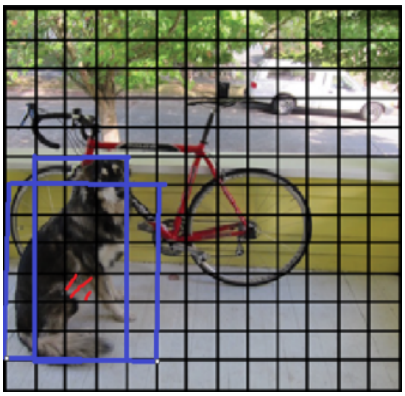

Figura 2.5: Divisão da imagem pelo YOLO

Depois da imagem ser dividida, são aplicados algoritmos de classificação e localização de objetos em cada célula da matriz. No final, é devolvido no máximo 5 *bounding boxes* e $5N$ valores probabilísticos para cada célula (nos casos em que a mesma contêm pelo menos um objeto), onde N representa a quantidade de classes a identificar. A maioria das *bounding boxes* resultantes não vão conter objetos e, para as remover, é aplicado o *Non-max Suppression* (NMS) que apenas remove as *bounding boxes* com as probabilidades mais baixas. [14] [15] [16]
A primeira versão do YOLO possuia alguns problemas tais como a dificuldade de detetar objetos mais pequenos. Tais problemas devem-se à maneira de como o YOLO é estruturado, no entanto, foram "aliviados"em *releases* futuras.

2.7.3 SSD

O *Single-Shot Detector* (SSD) foi proposto como uma alternativa mais rápida e precisa do que outros modelos tais como o YOLO e o *Faster* R-CNN. O SSD é baseado numa rede convolucional *feed-forward* que produz uma lista de objetos compostos por uma bounding box e uma pontuação. Este modelo é composto por duas componentes: o modelo *backbone* e o SSD *head*. O modelo *backbone* costuma ser um modelo de classificação de imagens pré-treinado como um extrator de *features* como por exemplo, o *ResNet*, ao qual foi retirada a última camada de classificação de um modo a obter uma rede neuronal capaz de extrair um significado de uma imagem. Já o SSD *head* é apenas uma ou mais camadas convolucionais adicionadas, posteriormente, ao *backbone* e os resultados obtidos são interpretados como classes de objetos e *bounding boxes*, que representam a localização e a classificação dos objetos reconhecidos (figura 2.6).

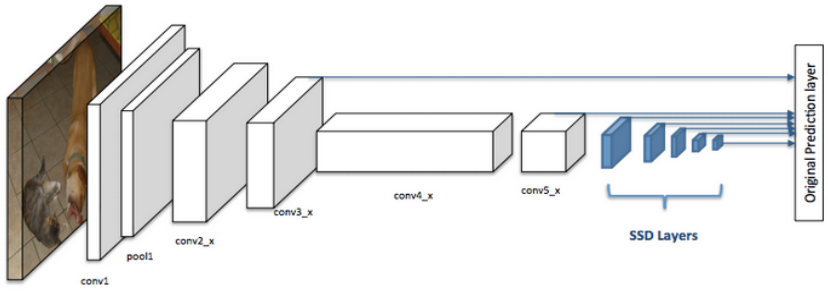

Figura 2.6: Arquitetura de uma rede neuronal convolucional típica com um detetor SSD

O SSD é relativamente mais rápido que outras CNN devido à sua técnica de deteção de objetos em regiões da imagem. Muitas das vezes é utilizada uma técnica simples tal como a *sliding window*, em que uma perquena janela passa por todas as regiões da imagem de um modo sequencial e, por cada região que passa, prevê os objetos nessa região. Já o SSD efetua uma divisão da imagem em grelha na qual cada célula é responsável por prever o(s) objeto(s) que contem.
Alguns objetos desta lista muitas das vezes apontam para diferentes deteções no mesmo objeto. Para isto, o modelo SSD aplica (num passo seguinte) o NMS para 'concentrar' estes pontos de alto número de deteções.

2.7.4 R-CNN

A rede neuronal convolucional profunda R-CNN foi desenvolvida em 2014 por um grupo de investigadores da *UC Berkeley*. Esta rede consegue detetar 80 tipos de objetos diferentes em imagens.

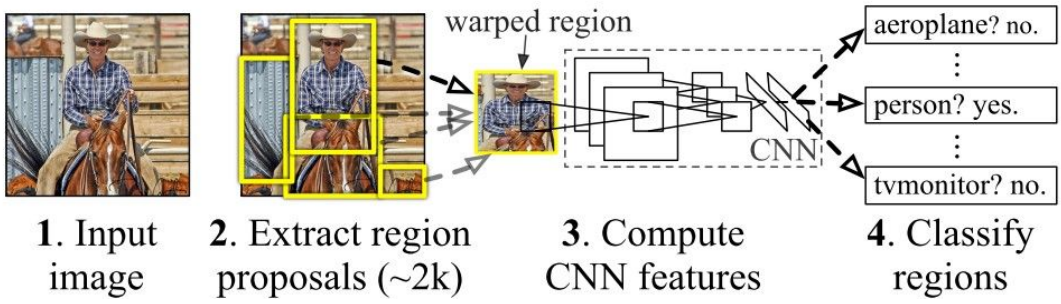

Figura 2.7: Funcionamento do modelo R-CNN

É possível dividir o modelo R-CNN em 3 partes funcionais:

- Escolha de 2000 regiões na imagem com o algoritmo *Selective Search*.
- Extração de um vetor de 4096 *features* para cada região escolhida.
- Classificar cada região escolhida (como *background* ou *object*) utilizando um algoritmo *Support Vector Machine* (SVM) pré-treinado.

No entanto, este modelo é pouco utilizado atualmente devido a limitações tais como a sua execução não ser em tempo real dado que cada região escolhida é analisada sequencialmente pela CNN para a extração de *features*. O modelo Fast R-CNN foi proposto como uma extensão ao R-CNN.

2.7.5 Fast R-CNN

Os mesmos autores do R-CNN conseguiram melhorar o modelo que desenvolveram e, assim, publicaram o *Fast* R-CNN. Este modelo é um bocado diferente do anterior:

- Foi desenvolvida a camada de *Pooling* de *Region of Interest* (RoI) que extrai vetores de *features* a partir de todas as propostas na mesma imagem.
- Com a camada de *Pooling* de RoI, o *Fast* R-CNN consegue ser mais rápido do que o R-CNN.
- O *Fast* R-CNN é mais preciso do que o R-CNN.

A camada RoI *Pooling* transforma cada região proposta da imagem numa grelha de valores. De seguida, é aplicada uma operação denominada de *max pooling* que irá calcular o maior valor em cada secção de valores da grelha. Se as dimensões da grelha forem (2×2), então o tamanho do vetor de *features* extraído será 4. O vetor extraído será processado por camadas *fully connected* e o *output* é processado por duas camadas finais: uma camada *softmax* e uma camada *fully connected*. A camada *softmax* final serve para prever as probabilidades de cada classe para cada objeto detetado e a camada *fully connected* final irá devolver uma previsão das *bounding boxes* dos objetos detetados na imagem, como pode ser observado na figura 2.8

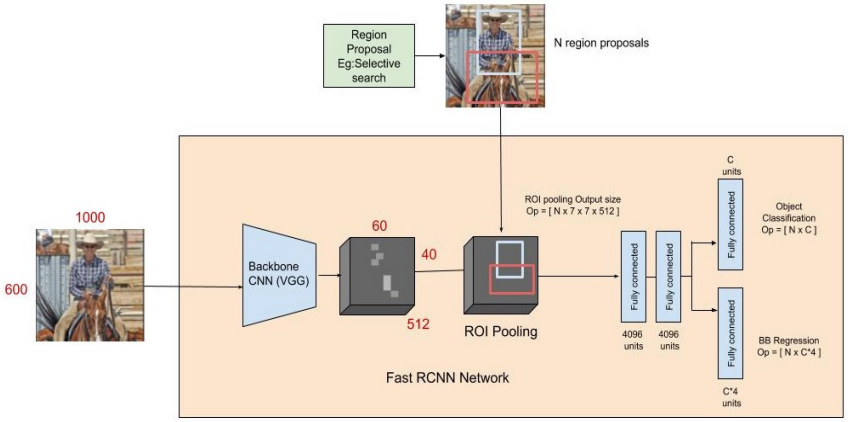

Figura 2.8: Arquitetura do modelo *Fast* R-CNN

No modelo R-CNN, cada região proposta na imagem é processada de um modo sequencial ou seja, apenas pode ser processada uma região de cada vez. O *Fast* R-CNN é mais rápido porque consegue processar várias regiões de uma vez. O modelo *Fast* R-CNN possui, no entanto, uma desvantagem: depende do algoritmo *Selective Search* para a escolha de regiões da imagem. Com isto e mais, o *Faster* R-CNN foi introduzido.

2.7.6 Faster R-CNN

O *Faster* R-CNN é uma extensão do *Fast* R-CNN que é mais rápido do que o anterior devido à implementação da *Region Proposal Network* (RPN). Além da RPN, foi também introduzido o conceito das *anchor boxes*. A arquitetura da *Faster* R-CNN é composta por duas partes: a RPN e o *Fast* R-CNN. (figura 2.9)

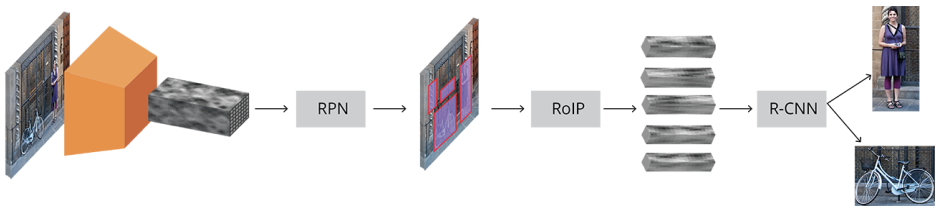

Figura 2.9: Arquitetura do modelo *Faster* R-CNN

A RPN é responsável pela escolha das propostas das regiões da imagem e o *Fast* R-CNN deteta os objetos nas regiões propostas. No início é inserida uma imagem na RPN, que devolve uma lista de regiões onde a probabilidade de existirem objetos de interesse seja alta. As regiões escolhidas são então introduzidas à camada de *pooling* de RoI que, por sua vez, irá devolver as *features* de cada objeto de interesse. De seguida, o *Fast* R-CNN classifica o conteúdo das *bounding boxes* como o objeto que acha ser ou *background* (no caso de querer ignorar) e, por último, ajusta as coordenadas das *bounding boxes*, de modo a destacar melhor o objeto detetado.

2.8 Avaliação de modelos

Depois de treinado num *dataset*, um modelo deve ser avaliado. A base da avaliação de modelos de classificação e deteção de objetos costuma ser sempre a comparação da previsão de um modelo e o seu *ground truth*, segundo um certo conjunto de dados (*dataset*).

2.8.1 *Intersection over Union*

O IoU, também conhecido como Índice de Jaccard, é uma métrica bastante utilizada na avaliação de modelos de classificação e deteção de objetos. Na base deste modelo encontra-se a comparação entre as *bounding boxes* dos objetos detetados e a *ground truth* dos mesmos para um determinado *dataset*. [17]
A expressão 2.1 é utilizada para calcular o IoU de uma deteção, onde a é uma *bounding box* prevista e b a *ground truth* do objeto:

$$IoU(a,b) = \frac{|a \cup b|}{|a \cap b|} \tag{2.1}$$

$$0 \leq IoU(a,b) \leq 1 \tag{2.2}$$

Como observado também na equação 2.2, o IoU encontra-se sempre entre 0 e 1, sendo 1 o melhor valor possível ou seja, a previsão é muito parecida com o que é suposto.
O resultado da equação comparado com um *threshold* irá indicar se a previsão é descartada ou não e permite agrupar os resultados da seguinte forma:

- ***True positive*** **(TP)** - Houve uma previsão de objeto e o valor do IoU da previsão encontra-se acima do *threshold* pré-definido.

- ***False positive* (FP)** - Houve uma previsão de objeto mas o valor do IoU da previsão encontra-se abaixo do *threshold* pré-definido porque o *ground truth* não define nenhum objeto.
- ***False negative* (FN)** - Não houve uma previsão de objeto mas o *ground truth* define um objeto.

2.8.2 Curva *Precision-Recall*

A curva *precision-recall* representa o equilíbrio entre precisão e sensibilidade (*recall*). A precisão e a sensibilidade são dadas pelas expressões 2.3 e 2.4, respetivamente:

$$P(TP,FP) = \frac{TP}{TP+FP} \tag{2.3}$$

$$R(TP,FP) = \frac{TP}{TP+FN} \tag{2.4}$$

Onde TP, FP e FN são os *true positives*, *false positives* e *false negatives*, respetivamente.
Um modelo onde a precisão é elevada e a sensibilidade é baixa é um modelo que devolve poucos resultados, no entanto, os mesmos são altamente precisos ou seja, são muito precisos quando comparados com as suas *ground truth*. Já um modelo onde a sensibilidade é elevada e a precisão baixa devolve muitos resultados, sendo que os mesmos são pouco precisos ou seja, as previsões são pouco precisas quando comparadas com as suas *ground truth*.

2.8.3 *Mean Average Precision* (mAP)

Muito utilizada hoje em dia, a métrica *Mean Average Precision* (mAP) utiliza a precisão e a sensibilidade do modelo e aplica às duas funções um *threshold*.
Para calcular a mAP, é primeiro necessário calcular a *Average Precision* (AP) com a expressão 2.5:

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{2.5}$$

E de seguida obtém-se o mAP pela expressão 2.6:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.6}$$

Por exemplo, um mAP_{50} representa um mAP cujo *threshold* é 0.5.

2.8.4 Avaliação de modelos de segmentação semântica de objetos

Como já foi referido, a segmentação semântica é efetuada ao atribuir uma classe de objeto a cada píxel de uma imagem (2.5). Como tal, é possível avaliar um modelo de segmentação semântica utilizando as mesmas técnicas de avaliação de modelos de deteção de objetos. A ideia é semelhante, no entanto, em vez de serem utilizadas *bounding boxes*, são utilizadas máscaras. É, portanto, possível calcular o IoU dado que as operações podem ser aplicadas tanto a retângulos/quadrados como também a outras formas, desde que sejam em duas dimensões. A média de IoU para todas as classes consta uma boa métrica para modelos de segmentação semântica. [17] [18]

2.8.5 Avaliação de modelos de *tracking* de objetos do tipo MOT

Atualmente, existem várias métricas capazes de avaliar modelos do tipo MOT. Uma das mais recentes é a HOTA. Esta métrica é calculada de acordo com três componentes:

- **Localização** - A localização é dada pelo cálculo de uma IoU. O cálculo é então efetuado recorrendo à expressão 2.1 para todos os pares (previsão, *ground truth*), originando vários *Loc-IoU*. A média de todos os *Loc-IoU* fornece a *Location accuracy* (LocA). [19]

- **Deteção** - A deteção mede o alinhamento entre todos os pares (previsão, *ground truth*) e é dada pelo cálculo da expressão $DetIoU = \frac{|TP|}{|TP|+|FN|+|FP|}$, onde *TP*, *FN* e *FP* representam os *true positives*, *false negatives* e *false positives* de todos os pares (previsão, *ground truth*), respetivamente. [19]

- **Associação** - A associação mede a capacidade de *tracking* do modelo, isto é, a capacidade de manter a mesma identificação para o mesmo objeto ao longo de frames distintas. É calculada recorrendo à expressão $Ass-IoU = \frac{|TP_a|}{|TP_a|+|FN_a|+|FP_a|}$, que mede o alinhamento entre duas sequências de *tracking*. Com este valor, podemos então calcular a *Association accuracy* (AssA) ao fazer a média dos valores *Ass-IoU* de cada classe. [19]

O HOTA segundo um *threshold* α é dado pela expressão 2.7:

$$HOTA_\alpha = \sqrt{DetIoU_\alpha \cdot AssA_\alpha}[19] \tag{2.7}$$

O resultado HOTA final é dado pela expressão 2.8 [19]:

$$HOTA = \int_{0 \leq \alpha \leq 1} HOTA_{\alpha} \tag{2.8}$$

2.9 Conclusões

O foco principal deste capítulo foi o estado da área de IA atual e também alguma história sobre a mesma. Neste capítulo foram mencionados e explicados vários conceitos e problemas de IA atuais, assim como também algumas soluções desenvolvidas para os mesmos. Foram também introduzidas as arquiteturas e modos de funcionamento de algumas RNA utilizadas atualmente. No fim, foi explicada a avaliação destes modelos.

Capítulo

3

Abordagem Proposta

3.1 Introdução

Este capítulo descreve a abordagem proposta para a resolução dos objetivos do trabalho. Descreve as ferramentas utilizadas na secção **3.2** para de seguida mencionar e explicar a arquitetura de sistema na secção **3.3**. Na secção **3.4** encontram-se mencionados os *datasets* utilizados para avaliar e treinar os modelos para, de seguida, descrever os modelos utilizados nas secções **3.5**, **3.6** e **3.7**. Finalmente, a secção **3.8** conclui este capítulo e introduz o próximo ligeiramente.

3.2 Ferramentas utilizadas

3.2.1 *Hardware*

Encontra-se mencionado na tabela 3.1 o computador utilizado para treinar e correr os modelos deste trabalho.

Tipo de Componente	**Nome**
CPU	Intel(R) Core(TM) i7-10700K @ 3.80GHz
RAM	16 GB
GPU	NVIDIA GeForce RTX 3080

Tabela 3.1: Especificações do computador utilizado para treino e inferência dos modelos utilizados neste projeto.

3.2.2 Linguagens de scripting

Neste trabalho foi utilizada a linguagem **Python** devido à imensidade de documentação disponível e a facilidade de uso. Foi também utilizada a linguagem **Bash** para manipulação de ficheiros, tratamento de pastas e organização estrutural dos datasets.

3.2.3 Ambiente de desenvolvimento

O trabalho foi realizado no sistema operativo *Windows 10 Pro N 64bit*, versão 21H2.

3.2.4 *Frameworks*

Foram utilizadas várias *frameworks* ao longo do trabalho com o intuito de facilitar a extração de dados e manipulação dos modelos.

3.2.4.1 *FastAI*

O *FastAI* é uma *framework* de DL implementada em PyTorch. Foi utilizada devido à sua extensiva e simples documentação, além da sua rápida implementação.

3.2.4.2 *Pytorch*

O *PyTorch* esteve sempre presente dado que que a utilização do *FastAI* implica a presença do *PyTorch*.

3.2.5 Bibliotecas

As bibliotecas listadas na tabela 3.2 foram também utilizadas na implementação deste projeto.

Biblioteca	**Descrição**	**Versão**
Pillow	Manipulação de imagens	9.0.1
NumPy	Algebra linear	1.22.3
OpenCV	Funções relacionadas com visão computacional	4.5.5
MatPlotLib	Visualização gráfica de dados	3.5.2
Pandas	Funções para manipulação/análise de dados	1.3.5
os	Operações relacionadas com o sistema operativo	-

Tabela 3.2: Nome e descrição das bibliotecas utilizadas neste trabalho.

3.2.6 *Label Studio*

O *Label Studio* é uma ferramenta *open-source* concebida para visualização de vários tipos de dados e também para a criação de *labels* para um *dataset*. Esta ferramenta foi utilizada para fazer a anotação do *dataset* criado com o drone. (3.4.4).

3.3 Arquitetura de sistema

O diagrama do sistema proposto para cumprir os objetivos do trabalho encontra-se representado na figura 3.1.

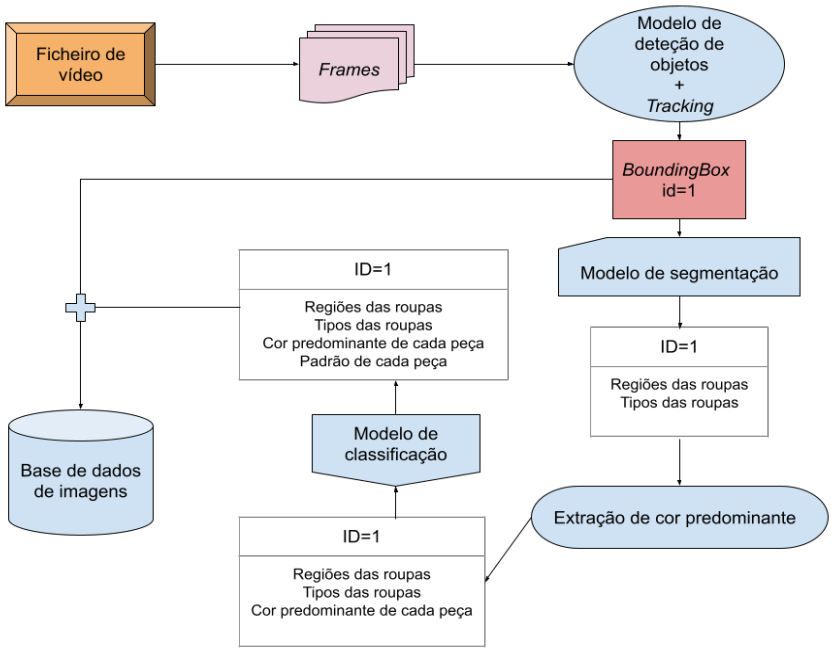

Figura 3.1: Diagrama da arquitetura do sistema proposto.

O ficheiro de vídeo é colocado numa lista de espera, na qual é compartido em *frames* que serão introduzidas à vez num modelo que deteta pessoas e faz *tracking* MOT. Para cada pessoa detetada é criada uma *bounding box* e um identificador único que irá pertencer à pessoa até que o fluxo de *tracking* da mesma seja interrompido. A *bounding box* e o identificador são enviados ao modelo de segmentação, cuja função é recortar a *bounding box* da *frame* original e efetuar a segmentação da pessoa. Este modelo irá devolver as regiões da roupa e o tipo de roupa previstos pelo modelo. Cada pedaço de roupa é

então classificado para que se possa determinar o padrão. De seguida, para cada região, é extraída a cor predominante da mesma e inserida num objeto final. Este objeto será então guardado na base de dados juntamente com a *bounding box* que recorta a pessoa detetada e o identificador da mesma.

3.4 *Datasets*

Para treinar e avaliar os modelos, foram importados e adaptados bastantes datasets, dado que muitos deles são diferentes uns dos outros.

3.4.1 *MS COCO*

O MS *Common Objects in Context* (COCO) é um *dataset* que contém 330000 imagens, das quais encontram-se no máximo 200000 anotadas. O dataset possui também 80 categorias de objetos. [20]
Este *dataset* foi utilizado para treinar e avaliar o modelo de deteção de pessoas.

3.4.2 *People Clothing Segmentation*

O *dataset People Clothing Segmentation* contém cerca de 2000 imagens e as respetivas máscaras. Cada imagem possui o tamanho de (825 × 550) píxeis. Este *dataset* descreve, em cada imagem, até 58 tipos de roupa distintos, ou seja, o número de classes é 59 (a contar com o background). As imagens consistem de fotos frontais onde a roupa se encontra bem visível e a luminusidade é alta [21].

3.4.3 *Basic Pattern*

O *Basic Pattern* contém 3800 imagens aleatórias de peças de roupa com padrões. O objetivo do uso deste *dataset* foi a criação e treino de uma CNN que conseguisse prever o padrão de cor na peça de roupa. O *dataset* está dividido em 11 tipos de padrões diferentes, possuindo cerca de 380 imagens para cada [22].

3.4.4 *Drone Dataset*

Este *dataset* foi criado para avaliar os modelos utilizados. Para tal, foram gravados 3 vídeos de 1 a 3 minutos a diferentes horas do dia de modo a encapsular a variação de luminosidade durante o dia. Os vídeos foram gravados

com o drone DJI Phantom 4 Pro V2.0 a 3.0 metros de altura. A resolução de cada frame é (1280 × 720) píxeis e o vídeo foi gravado a 29.90 *frames* por segundo. Foram escolhidas, de modo aleatório para cada vídeo, 15 *frames* para serem anotadas na ferramenta *Label Studio* (3.2.6). As *frames* são anotadas com *bounding boxes* à volta das pessoas.

3.5 Modelo de deteção e *tracking* de pessoas

Para deteção de pessoas foi utilizado o modelo YOLOv5. O mesmo encontra-se disponível em `https://github.com/mikel-brostrom/Yolov5_StrongSORT_OSNet` junto do modelo de *tracking*.
O YOLOv5 foi descarregado já pré-treinado (YOLOv5m) no *dataset MS COCO*. (3.4.1). [23]

3.6 Modelo de segmentação de roupa

Foram criados dois modelos de segmentação de roupa para serem testados. O primeiro modelo era uma *U-Net* cuja função de ativação era a sigmoide, o otimizador era *Adam* e a função de *loss* era *Binary Cross Entropy*. O segundo modelo foi também uma *U-Net*, onde a função de ativação foi *Mish* e o otimizador passou a ser *Ranger*, enquanto que a função de *loss* manteve-se igual. O learning rate para ambos os modelos foi $1 * 10^{-3}$ e foram treinados por 50 *epochs*. Os modelos foram treinados no *dataset People Clothing Segmentation* com uma *split* de 70-20-10 (treino, teste e validação, respetivamente).

3.7 Modelo de classificação do padrão de roupa

Foi criado um modelo CNN para classificar a roupa e fazer uma previsão do padrão da mesma. A função de otimização foi a *Adam* e o modelo foi treinado no *dataset Basic Pattern* durante 50 *epochs* com uma *split* de 70-20-10 (treino, teste e validação, respetivamente). A *learning rate* inicial foi $3 * 10^{-3}$.

3.8 Conclusões

Este capítulo oferece uma introdução às ferramentas e tecnologias utilizadas no trabalho e foram explicados os modelos utilizados e os detalhes dos mesmos. Foi também introduzida a arquitetura do sistema e como o mesmo iria

funcionar. O próximo capítulo visa apresentar os resultados dos modelos e uma breve avaliação dos mesmos.

Capítulo

4

Implementação e Testes

4.1 Introdução

Neste capítulo são apresentados os resultados obtidos para cada modelo utilizado neste trabalho. De seguida e de acordo com os resultados, cada modelo é avaliado de forma objetiva e subjetiva. No final, é apresentada uma avaliação final relativa ao funcionamento dos três modelos juntos, quando submetidos a um dataset customizado.

4.2 Avaliação dos modelos

Os modelos utilizados foram avaliados de duas formas diferentes: subjetiva e objetiva. A forma objetiva consiste em avaliar os modelos de forma automática, apresentando valores matemáticos e recorrendo a fórmulas para tal. No entanto, esta forma de avaliação pode induzir em erro e, como tal, é introduzida a forma subjetiva. A avaliação subjetiva é feita ao analisar diretamente os resultados manualmente e, com uma grande quantidade deles, tentar generalizar as razões de falha.

4.2.1 Modelos de deteção e *tracking* de pessoas

Os modelos utilizados para a deteção e *tracking* de pessoas já se encontravam treinados, como foi referido em 3.5.
Como já foi mencionado também em 3.5, foi utilizado o modelo pré-treinado *yolov5m*.
Em baixo encontra-se a avaliação subjetiva e objetiva do modelo.

4.2.1.1 Avaliação Objetiva

O modelo utilizado foi o *YOLOv5m* e foi treinado no *dataset MS COCO* (3.4.1). Na tabela 4.1, são apresentados os resultados do modelo pré-treinado e de outros também disponíveis e na tabela 4.2 encontram-se os hiperparâmetros utilizados.

Modelo	**Tamanho de imagem**	$mAP_{0.5:0.95}$	$mAP_{0.5}$
YOLOv5n	(640x640)	28.0	45.7
YOLOv5s	(640x640)	37.4	56.8
YOLOv5m	(640x640)	45.4	64.1
YOLOv5l	(640x640)	49.0	67.3
YOLOv5x	(640x640)	50.7	68.9

Tabela 4.1: Avaliação de modelos pré-treinados do YOLOv5 [1].

Hiperparâmetro	**Valor**
Epochs	300
Learning Rate Inicial	0.01
Learning Rate Final	0.01
Momentum	0.937
Weight decay (Otimizador)	$5 * 10^{-4}$
Epochs de aquecimento	3
Loss gain de bounding box	0.05
Loss gain de classe	0.5
IoU threshold	0.2

Tabela 4.2: Hiperparâmetros utilizados no treino do YOLOv5 no dataset MS COCO [1].

4.2.1.2 Avaliação Subjetiva

O modelo pré-treinado do YOLOv5 foi avaliado de forma subjetiva no dataset referido em 3.4.4, onde se encontram várias anotações de bounding boxes de pessoas em diversas frames de um vídeo, extraídas de modo aleatório.
Foram, portanto, observados dois pontos de falha principais.

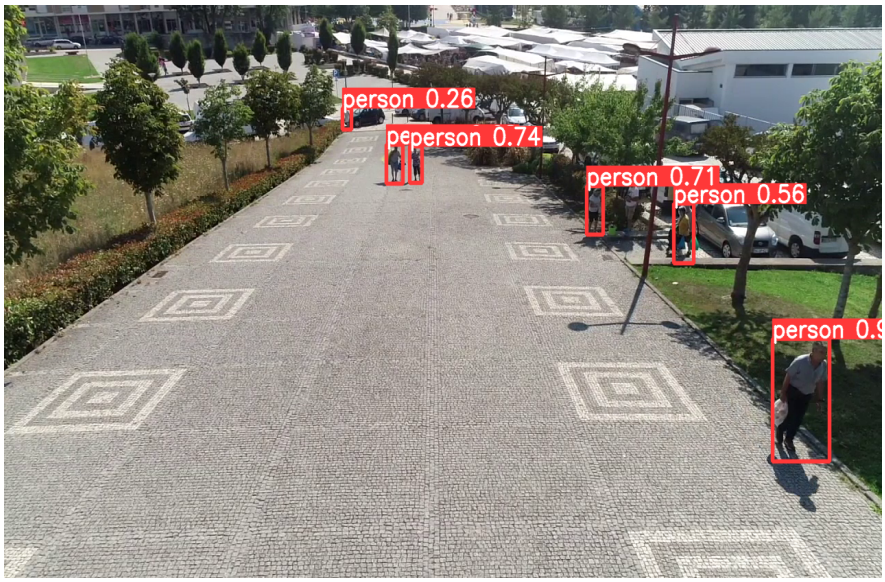

Figura 4.1: Inferência do modelo pré-treinado sob uma frame aleatória do dataset referido em 3.4.4.

É possível observar na figura 4.1 que:

- O modelo atribui uma menor probabilidade de classe às pessoas que se encontram mais afastadas da câmera.

- O modelo não consegue detetar pessoas que são obstruídas por outras pessoas (relativamente ao ponto de vista da câmera).

4.2.2 Modelos de segmentação da roupa

Para a segmentação de roupa, foram criados dois modelos já mencionados em 3.6. O treino e a validação foram feitos num só *dataset* (3.4.2). Uma comparação entre os parâmetros utilizados na construção de ambos os modelos encontra-se representada na figura 4.3.

Parâmetro	Primeiro	Segundo
Arquitetura Base	U-Net	U-Net
Encoder	ResNet34	ResNet34
Função de ativação	Mish	Sigmóide
Função de otimização	Ranger	Adam
Função de loss	Flat Cross Entropy	Binary Cross Entropy
Epoch	50	50
Learning Rate	0.001	0.001

Tabela 4.3: Comparação entre os modelos de segmentação criados.

Ambos os modelos adotam o *ResNet34*. O primeiro modelo utiliza a *Mish* para função de ativação, a *Ranger* para função de otimização e Flat Cross Entropy como a função de loss. Depois, o segundo modelo foi construído e foi utilizada a função de ativação sigmóide, a função de otimização *Adam* e a função de loss *Binary Cross Entropy*.

4.2.2.1 Avaliação Objetiva

Ambos os modelos foram treinados e avaliados segundo o *split* mencionado em 3.4.2.

	Primeiro	Segundo
Accuracy	0.9811	0.9139
Loss	0.5020	0.4993

Tabela 4.4: Resultados de validação do primeiro e segundo modelo.

Como se pode observar na tabela 4.4, o primeiro modelo obteve as pontuações $accuracy_{val} = 0.9811$ e $loss_{val} = 0.5020$ e o segundo modelo obteve as pontuações $accuracy_{val} = 0.9139$ e $loss_{val} = 0.4993$.

4.2.2.2 Avaliação Subjetiva

Ambos os resultados foram decentes a nível de *accuracy* e *loss*. Nas figuras 4.2 e 4.3 estão presentes resultados de frames aleatórias do *dataset* mencionado em 3.4.4.

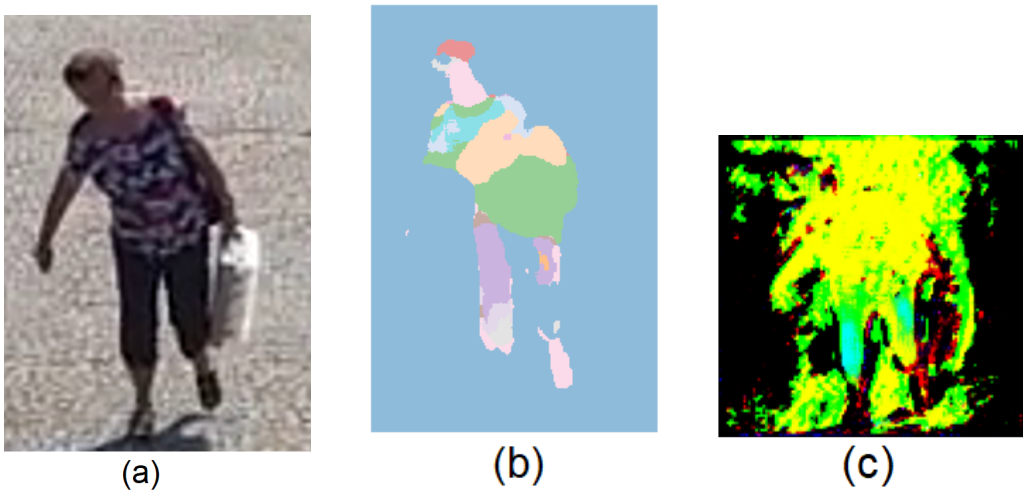

Figura 4.2: (a) Original (b) Modelo 1 (c) Modelo 2

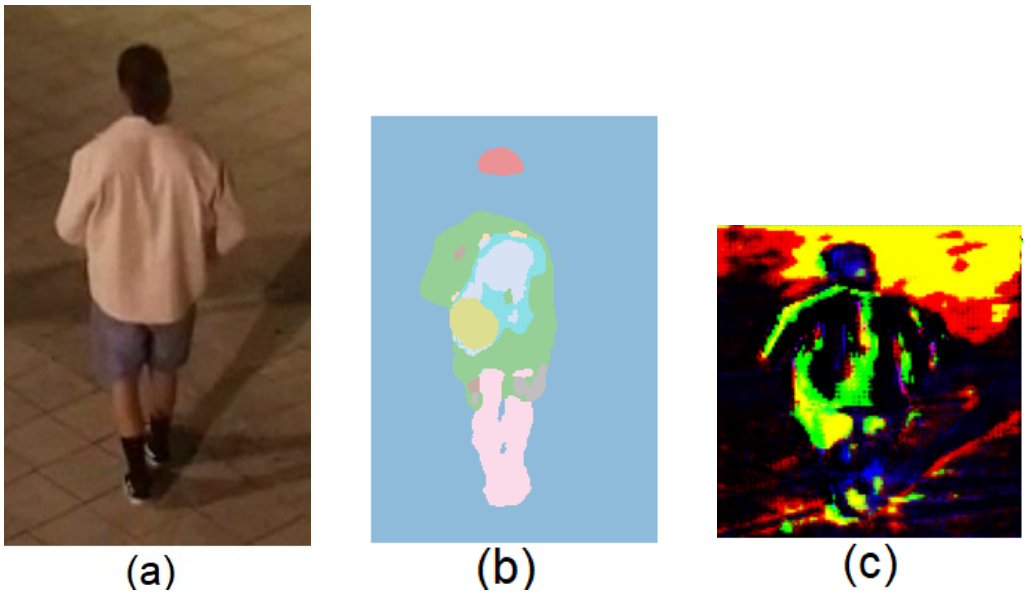

Figura 4.3: (a) Original (b) Modelo 1 (c) Modelo 2

Pela observação dos resultados, o segundo modelo produz muito mais ruído do que o primeiro modelo. Embora os mesmos não sejam os mais desejados, é possível utilizar o primeiro modelo para a extração do tipo de roupas e uma pequena região das mesmas.

4.2.3 Modelo de classificação de padrões

A base do modelo de classificação de padrões utilizado é a CNN e os parâmetros de treino do modelo encontram-se na tabela 4.5.

Parâmetro	Valor
Encoder	ResNet34
Epoch	50
Learning rate	0.0005
Função de otimização	Adam

Tabela 4.5: Parâmetros de treino do modelo de classificação de padrões.

4.2.3.1 Avaliação Objetiva

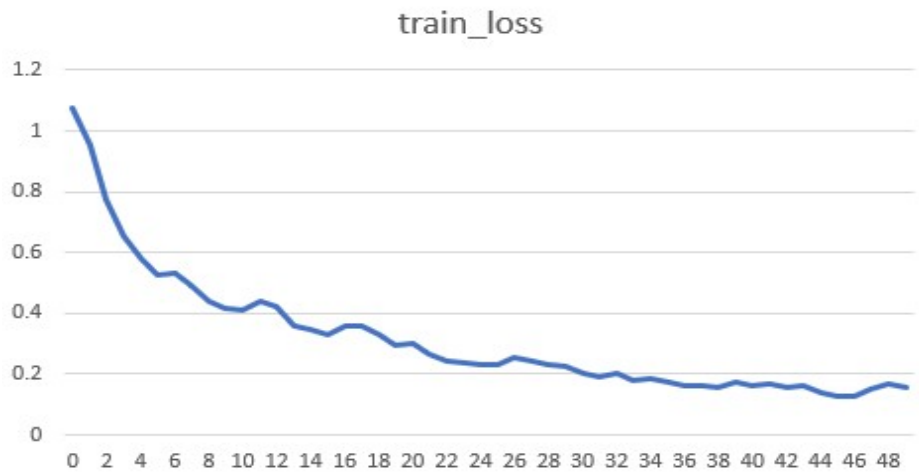

Figura 4.4: Evolução do *loss* ao longo do período de treino.

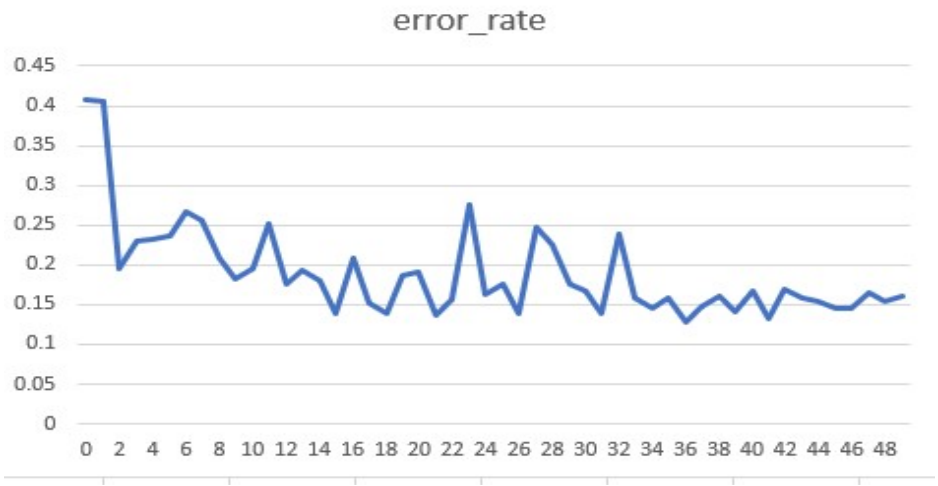

Figura 4.5: Evolução da taxa de erro ao longo do período de treino.

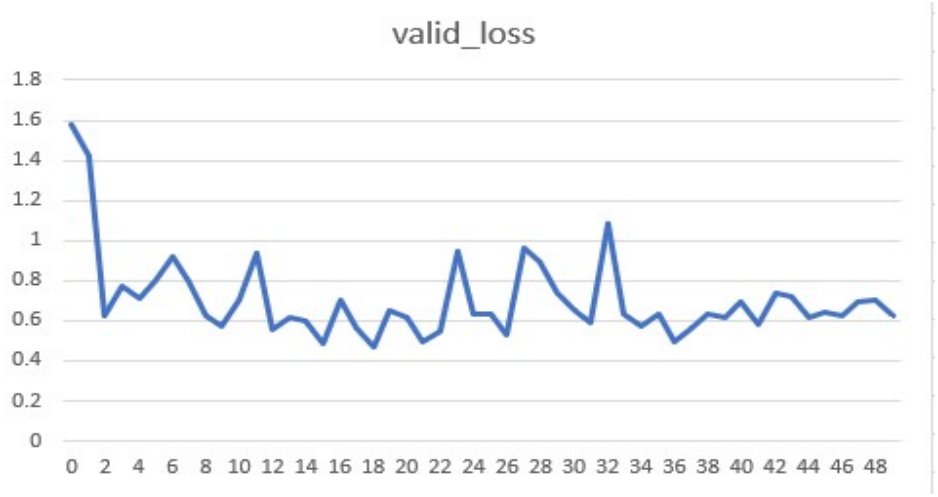

Figura 4.6: Evolução do *loss* de validação ao longo do período de treino.

Como é possível observar nas figuras 4.4 e 4.5, os valores *train loss* e *error rate* diminuiram ao longo do período de treino.

Observando a figura 4.6, é possível notar que o *valid loss* desceu até o epoch 34 onde depois permaneceu mais ou menos no intervalo de valores [0.45, 0.8[. Estes valores são muito altos, no entanto, pode-se concluir na secção 4.2.3.2 que as previsões do modelo não são injustificadas.

4.2.3.2 Avaliação Subjetiva

Figura 4.7: Previsões sobre algumas peças do dataset 3.4.3 e as suas *ground truths*

As previsões feitas e representadas na 4.7 são do tipo **PREVISÃO/GROUND TRUTH/LOSS/PROBABILIDADE**. Em ambas as peças de roupa representadas na figura, o modelo preveu de uma maneira muito interessante. Na peça à esquerda, o modelo preveu que é do tipo 'animal', ou seja, que a textura seria parecida à de um animal. No entanto, a realidade é que a textura é constituída de pequenos corações, mas, vista com uma pequena resolução, a textura parece a de um animal. O mesmo acontece no vestido à direita, onde a previsão é que o padrão da peça é às riscas e a *ground truth* afirma que o padrão assemelha o de um animal.
Dado estes resultados, é possível utilizar este modelo para prever o padrão das roupas segmentadas.

4.3 Sistema final

Como já foi explicado em 3.3, o sistema funciona seguindo uma sequência de eventos:

1. **Deteção e *tracking* de pessoas** - Um vídeo é decomposto em *frames* que são alimentadas aos modelos YOLOv5 e StrongSORT, onde o YOLOv5 deteta as pessoas e o StrongSORT é encarregue de fazer o *tracking*.

Cada *bounding box* com um ID novo é recortada e encaminhada para o próximo modelo;

2. **Segmentação de roupas e classificação** - Cada *bounding box* recebida é processada pelo modelo que, por sua vez, devolve uma lista de regiões de peças de roupa e o tipo de roupa presente em cada região, encaminhando estes para o próximo modelo;

3. **Classificação do padrão de roupas** - Uma vez recebidas as regiões, é feito um *crop* da *frame* inicial na forma da região da roupa e a mesma é classificada de acordo com o seu padrão. Se a probabilidade de um ou mais padrões forem superiores a um certo *threshold*, a maior probabilidade é aceite. Caso contrário, a roupa é declarada como se não tivesse padrão;

4. **Estimador de cor** - Para cada região é estimada a cor predominante.

O maior *setback* do sistema é o modelo de segmentação que não é capaz de destacar corretamente a roupa das pessoas, como se pode observar na figura 4.8.

Figura 4.8: Inferência do sistema num vídeo do dataset 3.4.4.

4.4 Conclusões

Neste capítulo dá-se por concluída a análise dos resultados dos modelos. É também feita uma avaliação objetiva e subjetiva para cada resultado de cada modelo. É também explicado o sistema final e são apresentados alguns resultados do mesmo, sendo estes também sujeitos a avaliação subjetiva.

Capítulo

5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

Este projeto concedeu uma passagem ao mundo da IA. Para o desenvolvimento do mesmo foram estudados vários modelos de DL de classificação, segmentação de objeto e tracking de objetos. Para a possibilidade de utilizar e re-utilizar estes modelos, foram também estudadas várias métricas que permitem a avaliação dos mesmos. Assim, foi apresentado um método que procurasse cumprir o objetivo do projeto: *fashion analysis from surveillance data taken from UAVs*. O método é composto por 4 partes que funcionam de modo sequencial, sendo 3 dessas modelos e a última um algoritmo de visão computacional. De maneira a medir a performance do sistema, o mesmo foi avaliado de forma subjetiva.
Ao longo do projeto foram procuradas alternativas para a solução atual. Espera-se que a experiência ganha ao desenvolver este trabalho sirva de alicerce para futuro conhecimento relativo a esta área.

5.2 Trabalho Futuro

O sistema encontra-se longe de ser perfeito. No entanto, o mesmo não necessita de ser perfeito para ser utilizável. Assim, devem-se destacar alguns pontos fracos do sistema e formas de os resolver:

- **Modelo de segmentação de roupas fraco** - Podem ser comparados mais modelos de segmentação de roupas, de modo a procurar o melhor para tal cargo e devem, também, ser utilizados *datasets* de maior dimensão. Os melhores modelos a serem utilizados seriam os de segmentação de

instância mas, devido a incompatibilidades de versões de bibliotecas, não foi possível a importação de bibliotecas que o permitisse.

- **Algoritmo de deteção de cores pouco robusto** - O algoritmo utilizado para determinar a cor depende da cor predominante na *frame* em que a mesma foi determinada. Ou seja, a cor que se encontra na *frame* pode não ser a cor exata ao vivo. Deverão ser estudadas outras formas para a determinação da cor da peça.
- **Utilização de uma câmera com melhor qualidade** - A relativamente baixa resolução da câmera utilizada para a criação do dataset 3.4.4 afetou ligeiramente a performance do modelo de deteção de pessoas, dado que as mesmas eram indistinguíveis quando distantes da câmera.

No fim, acredito que a primeira fase foi cumprida de forma parcial devido à má performance do modelo de segmentação que, por si, afetou a performance do sistema final. Não foi, no entanto, concluída a segunda fase do trabalho, que envolvia a utilização de uma base de dados que guardasse cada pessoa detetada pelo drone e a sua escolha de roupa naquele preciso momento. Tal sistema poderia ser implementando utilizando um tipo de anotação (categoria, atributo), onde a categoria é o tipo de roupa e o atributo seria a cor e padrão. Tal anotação tornaria a pesquisa de elementos que satisfizessem as condições desejadas muito mais simples.

Bibliografia

[1] ultralytics. YOLOv5. [Online] https://github.com/ultralytics/yolov5. Último acesso a 20 de junho de 2022.

[2] IBM. Computer vision – use machine learning and neural networks to teach computers to see. [Online] https://www.ibm.com/topics/computer-vision. Último acesso a 20 de junho de 2022.

[3] J. McCarthy. What is artificial intelligence? *Stanford*, 2004.

[4] K. Heinrich C. Janiesch, P. Zschech. Machine learning and deep learning. *Electronic Markets*, 31:685–695, 2021.

[5] SAS Institute. Machine Learning – o que é e qual a sua importância?, 2019. [Online] https://www.sas.com/pt_br/insights/analytics/machine-learning.html Último acesso a 20 de junho de 2022.

[6] Hans-Dieter Wehle. Machine learning, deep learning, and ai: What's the difference? 07 2017.

[7] VISO.AI. Object Detection in 2022: The Definitive Guide. [Online] https://viso.ai/deep-learning/object-detection/. Último acesso a 20 de junho de 2022.

[8] Kanan Vyas. Object Segmentation. [Online] https://medium.com/visionwizard/object-segmentation-4fc67077a678. Último acesso a 20 de junho de 2022.

[9] viso.ai. Object Tracking in Computer Vision (Complete Guide). [Online] https://viso.ai/deep-learning/object-tracking/. Último acesso a 8 de julho de 2022.

[10] LearnOpenCV. Understanding Multiple Object Tracking using DeepSORT. [Online] https://learnopencv.com/understanding-multiple-object-tracking-using-deepsort. Último acesso a 8 de julho de 2022.

[11] Yunhao Du, Yang Song, Bo Yang, and Yanyun Zhao. Strongsort: Make deepsort great again, 2022.

[12] IBM Cloud Education. Convolutional Neural Networks. [Online] `https://www.ibm.com/cloud/learn/convolutional-neural-networks`. Último acesso a 21 de junho de 2022.

[13] Grace Karimi. Introduction to YOLO Algorithm for Object Detection. [Online] `https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/`. Último acesso a 2 de julho de 2022.

[14] Geeks for Geeks. YOLO : You Only Look Once – Real Time Object Detection. [Online] `https://www.geeksforgeeks.org/yolo-you-only-look-once-real-time-object-detection/`. Último acesso a 2 de julho de 2022.

[15] Analytics Vidhya. A Practical Guide to Object Detection using the Popular YOLO Framework – Part III (with Python codes). [Online] `https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python`. Último acesso a 2 de julho de 2022.

[16] Appsilon. YOLO Algorithm and YOLO Object Detection. [Online] `https://appsilon.com/object-detection-yolo-algorithm/`. Último acesso a 3 de julho de 2022.

[17] pyimagesearch.com. Intersection over Union (IoU) for object detection. [Online] `https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/`. Último acesso a 8 de julho de 2022.

[18] Jeremy Jordan. Evaluating image segmentation models. [Online] `https://www.jeremyjordan.me/evaluating-image-segmentation-models/`. Último acesso a 20 de junho de 2022.

[19] J. Luiten, Osep A., and P. Dendorfer. A note on a simple transmission formula. *International Journal of Computer Vision*, 129:548–578, 2020.

[20] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[21] Wei Yang, Ping Luo, and Liang Lin. Clothing co-parsing by joint image segmentation and labeling. *CoRR*, abs/1502.00739, 2015.

[22] RIZWAN ALI SHAH. Basic Pattern Dataset (BPD) Women Clothing. [Online] `https://www.kaggle.com/datasets/rizzsum/basic-pattern-women-clothing-dataset`. Último acesso a 20 de junho de 2022.

[23] Mikel Broström. Real-time multi-camera multi-object tracker using yolov5 and strongsort with osnet. `https://github.com/mikel-brostrom/Yolov5_StrongSORT_OSNet`, 2022.