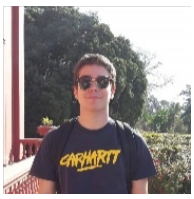
Aprendizagem Profunda

Deteção de Objetos em Imagens

Relatório de Desenvolvimento





Beatriz Monteiro PG53692

Daniel Furtado PG53754

Nuno Costa PG54121

Grupo 6

Aprendizagem Profunda 2023/2024

Universidade do Minho

Resumo

Este documento regista e documenta todo o trabalho realizado no projeto prático realizado no âmbito da Unidade Curricular Aprendizagem Profunda inserida no perfil de Sistemas Inteligentes. O domínio do problema é a deteção de objetos em imagens. O objetivo é pegar em modelos de deteção já existentes, como *YOLO*(*You Only Look Once*) ou *Faster R-CNN*, e comparar os seus resultados.

Contents

1 Introdução

Dando continuidade à exploração iniciada no trabalho de investigação da unidade curricular de Aprendizagem Profunda, foi-nos proposto aplicar na prática a comparação entre os modelos *YOLOv9* e *Faster R-CNN*. Neste contexto, abordamos a avaliação de desempenho desses modelos em tarefas de deteção de objetos, com foco em métricas como precisão, *recall* e *mAP* (*mean Average Precision*).

A aplicação prática desta comparação envolveu a implementação de ambos os modelos num conjunto de dados específico, com o objetivo de observar as diferenças em termos de eficiência e precisão na deteção de objetos. O *YOLOv9*, conhecido pela sua capacidade de realizar deteções em tempo real com alta velocidade, foi comparado ao *Faster R-CNN*, reconhecido pela sua robustez e alta precisão, embora com uma velocidade de inferência mais baixa.

Deste modo, neste relatório iremos abordar a metodologia utilizada neste trabalho prático, a descrição global de todo o *software/scripts* desenvolvidos, a exploração e tratamento dos dados, os modelos desenvolvidos e, por fim, uma síntese dos resultados e análise crítica.

2 Metodologia

A metodologia adotada abrange desde a escolha do *dataset* até à comparação final dos modelos, passando pelo desenvolvimento dos treinos de cada modelo. Cada etapa foi cuidadosamente executada para garantir a integridade e a eficácia dos processos, permitindo uma análise precisa e crítica do desempenho dos modelos em tarefas de deteção de objetos. Desta forma, as etapas da nossa metodologia são as seguintes:

- **Escolha do *dataset***: A primeira etapa da nossa metodologia consistiu na escolha do *dataset*, que foi o ***Rock Paper Scissors SXSW Computer Vision Project*** da plataforma *Roboflow*. Os dados foram organizados em pastas separadas para treino, validação e teste, garantindo que cada conjunto tivesse imagens com as respetivas *labels*, tanto em formato XML como TXT, contendo informações sobre os objetos presentes nas imagens.

- **Exploração e Tratamento dos dados**: Posteriormente, realizámos uma exploração dos dados, verificando quantas imagens cada conjunto continha, quais eram as classes dos objetos a serem identificados, se havia *labels* vazias e se todas as imagens tinham correspondência com a sua respetiva *label*. Após esta exploração, passámos para o tratamento dos dados de acordo com a análise realizada. Estes tópicos serão abordados de forma mais detalhada mais à frente no relatório.

- **Treino dos Modelos**: Após a exploração e tratamento dos dados, avançámos para o treino dos dois modelos de deteção de objetos. Neste treino, procurámos manter as condições o mais semelhantes possível, dentro dos hiperparâmetros utilizados, limitando o tempo de treino a uma hora. Isto permitiu-nos, posteriormente, comparar os resultados de forma equitativa.

- **Avaliação dos Modelos**: De seguida à fase de treino, passámos à fase de avaliação dos modelos, onde utilizámos as imagens do conjunto de teste para avaliar os modelos e obter métricas sobre o desempenho de ambos.

- **Usar o Modelo**: Nesta fase, utilizámos as imagens de teste e verificámos quais as detecções e classificações que cada modelo conseguiu realizar em algumas dessas imagens. Posteriormente, apresentaremos alguns exemplos.

- **Comparar resultados**: Esta última fase consistiu em comparar os resultados dos dois modelos e tirar conclusões relativamente aos resultados obtidos.

3 Descrição Global

A primeira tarefa deste projeto consistia na escolha de um *dataset* apropriado para o nosso projeto.

O grupo recorreu à plataforma *Roboflow* no processo de procura de um *dataset* apropriado para este trabalho, devido à sua facilidade de acesso, organização e compatibilidade com tarefas de deteção de objetos.

Os critérios utilizados para a procura do *dataset* foram:

1. Permita a deteção de objetos;
2. Números razoáveis de imagens;
3. Balanceamento do *dataset*.

Desta forma, o grupo escolheu o *dataset* ***Rock Paper Scissors SXSW Computer Vision Project***. Este *dataset* permite que os modelos sejam treinados para detetar se a posição das mãos das pessoas está no formato pedra, papel ou tesoura.

Para a construção do *dataset*, o autor solicitou a várias pessoas na *internet* que se gravassem a jogar "Pedra, Papel ou Tesoura". A partir desses vídeos, foram extraídos alguns *frames*, que resultaram em imagens que capturam pessoas a jogarem. É importante referir que, além de imagens do jogo, existem imagens de coisas não relacionadas.

O *dataset* já vinha dividido em conjuntos de treino, teste e validação. Estes conjuntos foram criados de forma aleatória, segundo o autor.

Para a realização do projeto, o grupo recorreu ao Google Colab, tirando partido dos GPUs gratuitos disponíveis. Optou-se por utilizar o Tesla T4, por ser a melhor opção disponível na versão gratuita.

Depois do trabalho de investigação feito previamente sobre este assunto, o grupo optou por testar e analisar o desempenho dos modelos *YOLOv9* e *Faster R-CNN*, como será possível constatar mais à frente neste relatório.

Quanto aos modelos, o *YOLOv9* utilizado foi obtido do *GitHub* em `https://github.com/SkalskiP/yolov9.git`, enquanto o *Faster R-CNN* utilizado foi obtido em `https://github.com/sovit-123/fastercnn-pytorch-training-pipeline.git`. Além desses modelos, o grupo também fez uso de pesos pré-treinados denominados *gelan.pt*, que foram posteriormente utilizados no treino do *YOLO*, e no *Faster R-CNN* foi utilizado o *backbone fasterrcnn_resnet50*.

4 Exploração e Tratamento dos dados

Como foi mencionado anteriormente, a exploração e o tratamento dos dados consistiram numa fase crucial da nossa metodologia. Nesta fase, realizámos uma análise detalhada para compreender a distribuição dos dados e identificar possíveis problemas, como *labels* vazias ou inconsistências entre as imagens e as respetivas *labels*. Após a exploração inicial, procedemos ao tratamento dos dados, que envolveu a limpeza e a organização dos conjuntos de dados para garantir que estavam adequadamente preparados para o treino dos modelos.

4.1 Exploração

Numa primeira fase, começámos por analisar o número de imagens em cada conjunto de dados e se todas as classes de objetos a serem identificados estavam presentes em todos os conjuntos. Assim, verificámos que o nosso conjunto de **dados total continha 7335 imagens**, sendo que o conjunto de **treino** tinha **6455 imagens**, o conjunto de **teste** continha **304 imagens** e o conjunto de **validação** possuía **576 imagens**. Importante destacar que todos os conjuntos apresentavam a presença de todas as classes de objetos.

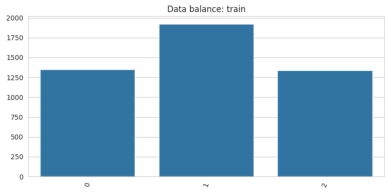
```
Quantidade de imagens para train: 6455
Classes de labels para train: {'0', '1', '2'}
Quantidade de imagens para test: 304
Classes de labels para test: {'0', '1', '2'}
Quantidade de imagens para valid: 576
Classes de labels para valid: {'2', '1', '0'}

Total de classes de labels no dataset: {'0', '1', '2'}
Total de imagens em todo o dataset: 7335
```

Após isso, verificamos se todas as imagens tinham uma *label* associada, garantindo que nenhuma imagem estivesse com *label* em falta. A exploração revelou que todas as imagens do *dataset* tinham *labels* associadas. Em seguida, investigamos quantas *labels* estavam vazias, ou seja, quantas não tinham nenhuma classe associada. Descobrimos que no conjunto de **treino havia 2516 *labels* vazias**, no conjunto de **teste havia 118 *labels* vazias** e no conjunto de **validação havia 238 *labels* vazias**.

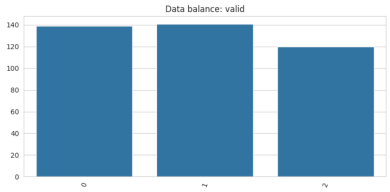
```
Número de labels vazias em train: 2516
Número de labels vazias em test: 118
Número de labels vazias em valid: 238
```

Posteriormente, procedemos à análise do balanceamento das diferentes classes em cada um dos conjuntos. No conjunto de treino, encontramos 1349 ocorrências da classe 0, 1924 da classe 1 e 1337 da classe 2. No conjunto de teste, registramos 72 ocorrências da classe 0, 65 da classe 1 e 57 da classe 2. Por fim, no conjunto de validação, observamos 139 ocorrências da classe 0, 141 da classe 1 e 120 da classe 2.



(a) Balanceamento do conjunto de treino

(b) Balanceamento do conjunto de teste



(c) Balanceamento do conjunto de validação

Por fim, verificamos a correspondência de cada número com a classe específica, onde a classe 0 representa o papel (*Paper*), a classe 1 representa a pedra (*Rock*) e a classe 2 representa a tesoura (*Scissors*).

```
{0: 'Paper', 1: 'Rock', 2: 'Scissors'}
```

4.2 Tratamento

Quanto ao tratamento dos dados, inicialmente o grupo optou por não realizar qualquer tratamento. No entanto, após o *feedback* dos docentes, decidiu-se remover as *labels* vazias juntamente com as imagens correspondentes. Como resultado, restaram **3939 imagens no conjunto de treino**, **186 no conjunto de teste** e **338 no conjunto de validação**.

```
Número de labels vazias apagadas em train: 2516
Número de labels vazias apagadas em test: 118
Número de labels vazias apagadas em valid: 238
```

5 Descrição dos modelos desenvolvidos

Tal como já foi mencionado anteriormente, os modelos que utilizamos *YOLOv9* e o Faster R-CNN.

Escolhemos estes modelos por se destacarem na deteção de objetos, cada um com as suas próprias vantagens. O *YOLOv9* é eficiente para deteções em tempo real devido à sua arquitetura otimizada que melhora a precisão e reduz a complexidade computacional, tornando-o ideal para aplicações que exigem alta velocidade e baixo consumo de recursos. Por outro lado, o *Faster R-CNN* é conhecido pela sua alta precisão ao identificar regiões de interesse e realizar deteções detalhadas, sendo amplamente utilizado em cenários onde a precisão é crucial.

Ao comparar o desempenho do *YOLOv9* e do *Faster R-CNN* com os mesmos hiperparâmetros, pretendemos avaliar como é que estes modelos se comportam sob condições controladas, destacando as suas forças e fraquezas em termos de precisão, velocidade e eficiência computacional. Esta análise ajudar-nos-á a determinar qual modelo é mais adequado para diferentes aplicações e cenários.

Como referido os hiperparâmetros utilizados nos dois modelos são praticamente os mesmos.

Learning Rate	0.001
Batch Size	16
Resize	640
Epochs	5/20

Table 1: Hiperparâmetros Utilizados

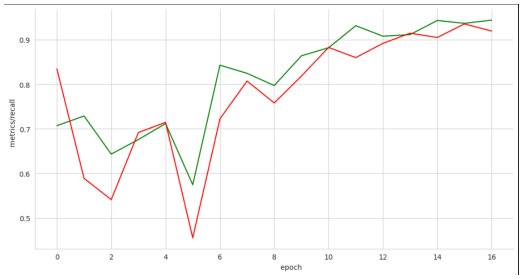
Descrição dos hiperparâmetros:

1. ***Learning Rate***: Controla o tamanho dos ajustes feitos nos pesos do modelo durante o treino, influenciando a rapidez e precisão da convergência do modelo.
2. ***Batch Size***: Determina o número de exemplos de treino usados numa iteração, afetando a eficiência computacional e a generalização do modelo.
3. ***Resize***: Define o tamanho das imagens de entrada, influenciando na consistência e eficiência do processamento durante o treino e inferência.
4. ***Epochs***: Indica quantas vezes o modelo passa pelos dados de treino, afetando a convergência do modelo e sua capacidade de generalização para novos dados.

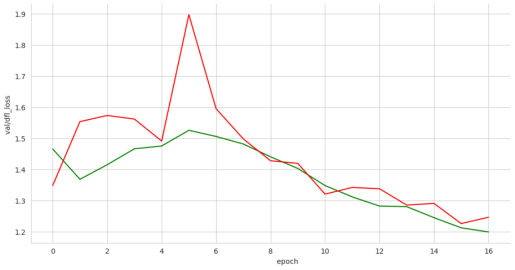
Para podermos comparar o desempenho dos modelos utilizamos como medida de referência o tempo, isto é, fizemos com que os 2 modelos fossem treinados durante 1 hora e em seguida comparamos os seus resultados.

Gráficos do treino do *YOLO*

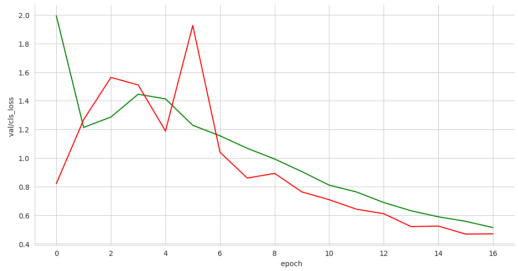
Após o treino do modelo *YOLOv9*, realizamos uma análise detalhada do desempenho. Os gráficos abaixo apresentam as métricas de precisão (*precision*) e *recall*, bem como as perdas durante o treino e validação.



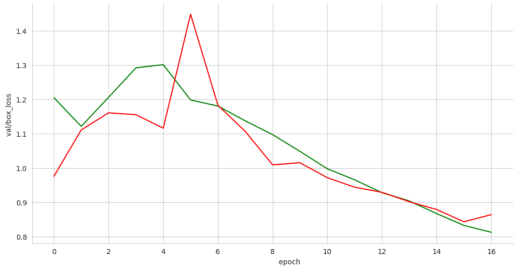
(a) *Precision*(Verde) e *Recall*(Vermelho)



(b) *dfl_loss*, Treino(Verde) e Validação(Vermelho)



(c) *cls_loss*, Treino(Verde) e Validação(Vermelho)



(d) *box_loss*, Treino(Verde) e Validação(Vermelho)

Através da análise destes gráficos, podemos observar, primeiramente, que ao longo das *epochs* tanto a precisão quanto o *recall* apresentam um aumento gradual, o que é positivo. Relativamente aos três

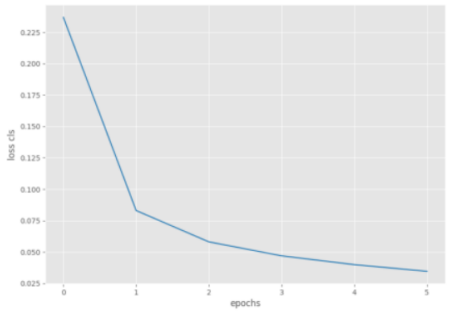
gráficos de perda, todos eles exibem uma diminuição consistente ao longo das *epochs* o que é um ótimo sinal.

No entanto, é importante notar um padrão peculiar durante o treino entre a 5ª e a 6ª *epoch*, onde ocorre um aumento momentâneo na *loss* e uma descida momentânea na precisão e no *recall*. Este comportamento anómalo pode ser atribuído a diversos fatores, como flutuações nos dados de treino, alterações na taxa de aprendizagem(*learning rate*), ou até mesmo questões relacionadas com a configuração do modelo.

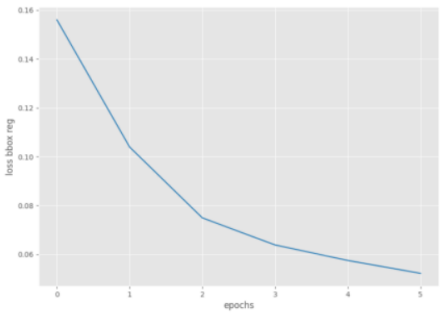
Gráficos do treino do *Faster R-CNN*

Assim como realizado para o *YOLOv9*, realizamos uma análise do desempenho do *Faster R-CNN*.

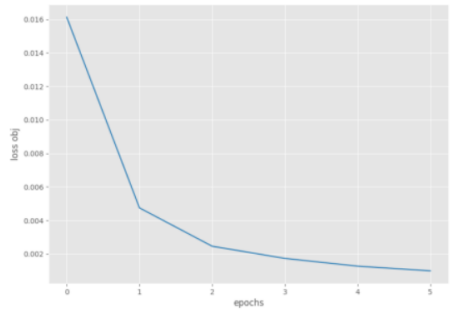
Os gráficos abaixo apresentam as métricas de precisão e *recall*, juntamente com as perdas durante o treino e validação, proporcionando uma compreensão abrangente do comportamento do modelo durante o processo de treino.



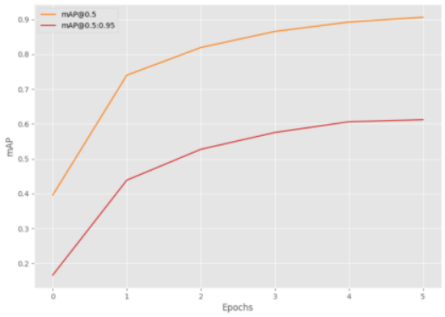
(a) *cls_loss*



(b) *loss_bbox*



(c) *loss_obj*



(d) *mAP*

Nestes gráficos, percebemos que ao longo das *epochs* o valor da função de perda (*loss*) diminui, é possível observar nos três primeiros gráficos, o que mostra que o erro vai sendo reduzido progressivamente à medida que o modelo é treinado. Esta diminuição na perda é indicativa de que o modelo está a aprender e a ajustar-se aos dados de treino.

Por outro lado, no último gráfico, vemos que a *mean Average Precision* (*mAP*) aumenta com o passar das *epochs*. Isso sugere que o desempenho do modelo está a melhorar na tarefa de deteção de objetos ao longo do tempo. Em termos práticos, um aumento no *mAP* significa que o modelo está a tornar-se mais preciso em identificar e localizar objetos.

6 Resultados e Análise

Após o treino dos dois modelos, avançámos para a avaliação de ambos, utilizando os dados de teste. Como indicadores de comparação, utilizámos a ***precision***, que mede com que frequência as previsões do nosso modelo estão corretas, e o ***recall***, que indica qual a percentagem de *labels* relevantes foram identificadas com sucesso.

YOLOv9

Ao analisar os resultados na seguinte tabela, observamos que o modelo *YOLOv9* apresenta uma precisão média de 88.6% e um *recall* médio de 89.5%. Ao analisarmos as classes individualmente, notamos que a precisão varia de 83.1% a 92.8%, e o *recall* varia de 82.2% a 97.0%.

Estes resultados demonstram que, com um treino limitado a uma hora e que permitiu ao modelo atingir **16 epochs**, o modelo *YOLOv9* foi mais eficaz na deteção de objetos nas imagens de teste utilizadas neste trabalho.

Class	***Precision***	***Recall***
all	0.886	0.895
Paper	0.831	0.822
Rock	0.928	0.892
Scissors	0.898	0.97

Table 2: Resultados da Avaliação no *YOLOv9*

Faster R-CNN

Na tabela que se segue, podemos concluir que os valores tanto da precisão quanto do *recall* são inferiores aos valores obtidos através da avaliação do *YOLOv9*.

Analisando os resultados, observamos que o modelo *Faster R-CNN* apresenta uma precisão média de 60.1% e um *recall* médio de 71.7%. Quando analisamos as classes individualmente, notamos que a precisão varia de 50.3% a 67.4%, e o *recall* varia de 65.7% a 75.8%.

Estes resultados indicam que o modelo *Faster R-CNN* pode não ser tão eficaz quanto o *YOLOv9* na deteção de objetos nas imagens de teste utilizadas neste trabalho.

Class	***Precision***	***Recall***
all	0.601	0.717
Paper	0.503	0.657
Rock	0.626	0.737
Scissors	0.674	0.758

Table 3: Resultados da Avaliação no *Faster R-CNN*

Análise sobre os resultados

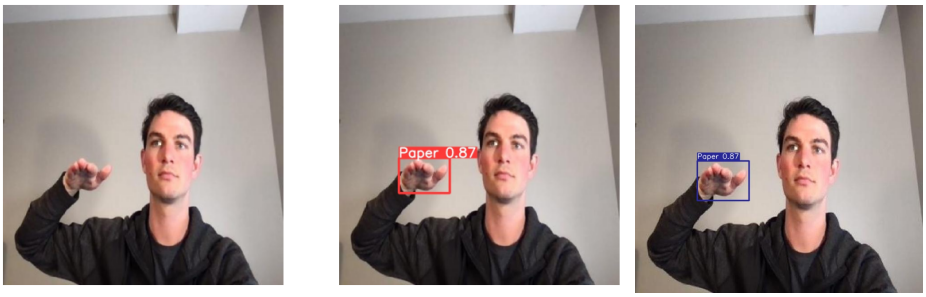
Numa análise global, podemos observar que ambos os modelos, *YOLOv9* e *Faster R-CNN*, foram capazes de realizar a deteção de objetos com relativo sucesso. No entanto, há diferenças significativas nos seus desempenhos, como evidenciado pelos resultados de precisão e *recall*. O modelo *YOLOv9* apresenta valores de precisão e *recall* mais elevados em comparação com o *Faster R-CNN*. Isso sugere que o *YOLOv9* é mais eficaz na identificação correta de objetos nas imagens de teste, proporcionando uma maior confiança nas suas previsões. Por outro lado, o *Faster R-CNN* obteve valores inferiores tanto de precisão quanto de *recall*, indicando que pode haver áreas onde o modelo falha em identificar corretamente os objetos ou onde identifica incorretamente objetos que não estão presentes nas imagens. Estes resultados são justificados pelo treino pouco rigoroso que foi aplicado, devido às limitações que enfrentámos. O grupo acredita que, com a utilização de um GPU sem restrições, poderíamos ter obtido melhores resultados, especialmente para o modelo Faster R-CNN. Em suma, embora ambos os modelos tenham sido capazes de realizar a tarefa de deteção de objetos, o *YOLOv9* demonstrou um desempenho superior em termos de precisão e *recall* em comparação com o *Faster R-CNN*. Esses resultados destacam a importância da escolha adequada do modelo para uma determinada aplicação, bem como da avaliação cuidadosa do desempenho antes da implementação em cenários do mundo real.

7 Exemplos de Deteções Realizadas

Em seguida são apresentadas algumas deteções feitas pelos modelos nos dados de teste.

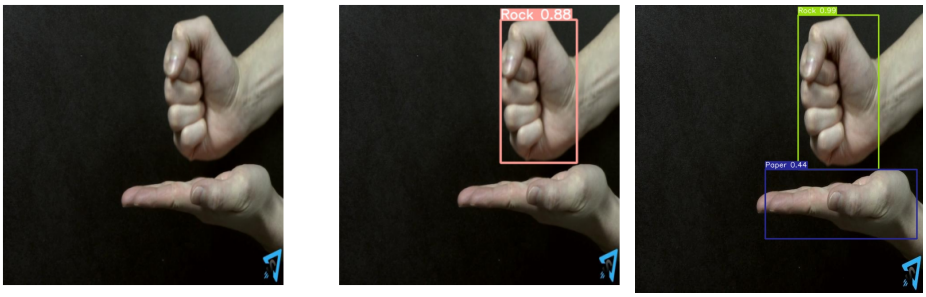
Nas imagens, é possível visualizar, primeiramente a imagem na qual ocorrerá a deteção e, em seguida 2 imagens, a primeira com a deteção feita pelo *YOLO* e a segunda pelo *Faster R-CNN*.

Nas imagens onde ocorreram as deteções existe uma caixa que delimita a deteção feita, com a *label* prevista associada e o grau de certeza.



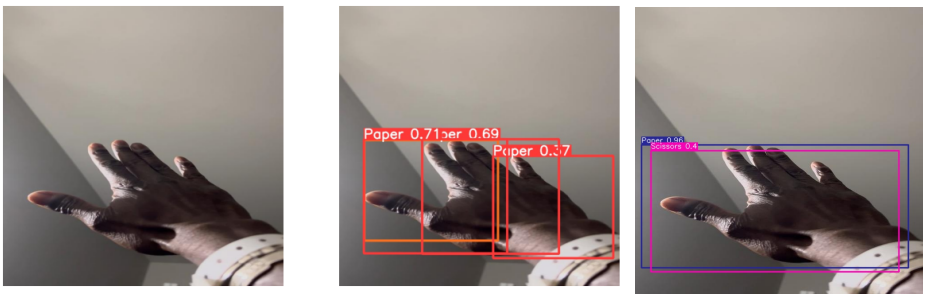
(a) *YOLO* (b) *Faster R-CNN*

Figure 4: Imagem de teste 1



(a) *YOLO* (b) *Faster R-CNN*

Figure 5: Imagem de teste 2



(a) *YOLO* (b) *Faster R-CNN*

Figure 6: Imagem de teste 3

8 Conclusão

Este trabalho prático permitiu-nos não só aplicar os conhecimentos adquiridos durante as aulas teóricas e práticas, mas também compreender e reter informações sobre diversos algoritmos de deteção de objetos. Apesar de algumas limitações, especialmente no que diz respeito à utilização do GPU, o grupo encontra-se satisfeito com os resultados alcançados dentro das possibilidades existentes.

A junção deste trabalho prático com a investigação realizada permitiu-nos concluir que ambos os modelos têm diferentes cenários de aplicação devido às suas características e complexidade. Por exemplo, os algoritmos *YOLO*, devido à sua capacidade de reconhecimento rápido e deteção em tempo real, são mais adequados para cenários que exigem processamento rápido e têm requisitos moderados de precisão, como sistemas de reconhecimento visual em tempo real para veículos. Por outro lado, o *Faster R-CNN*, apesar de ser mais lento na deteção, destaca-se pela precisão no reconhecimento, tornando-o mais adequado para cenários que exigem precisão mais elevada, mesmo que à custa de velocidades de processamento mais lentas. Esta distinção entre os modelos destaca a importância de considerar as necessidades específicas de cada aplicação ao selecionar o modelo de deteção de objetos mais apropriado.

Quanto ao futuro trabalho, o grupo estará atento aos novos lançamentos de algoritmos de deteção de objetos em visão computacional, buscando oportunidades para expandir as comparações com os modelos mencionados anteriormente. Além disso, pretendemos explorar melhores ferramentas para permitir treinos mais rigorosos e completos para os nossos modelos. Por fim, gostaríamos de aplicar técnicas de *data augmentation* aos nossos conjuntos de dados de treino, o que potencialmente melhoraria os nossos resultados.