

# Processamento Digital de Imagem

Mestrado em Engenharia Eletrotécnica e de Computadores

Mestrado em Engenharia Informática

## Protocolo do projeto Leitura Automática de matrículas





Uma empresa de tecnologia pretende desenvolver um algoritmo que faça a deteção de matrículas portuguesas e extraia os seus caracteres para fazer o controlo das entradas e saídas das instalações da empresa.

O algoritmo deve ser constituído por quatro módulos:

- **Deteção da matrícula:** módulo responsável pela deteção das matrículas. Neste módulo serão recolhidas imagens de veículos com matrícula portuguesa, com as quais será criado um dataset utilizado para treinar um modelo de *deep learning*. Depois do treino, os pesos (um ficheiro do tipo .pt, por exemplo, “best.pt”), gerados durante o treino, serão utilizados para efetuar inferências sobre imagens nunca antes vistas. Por fim, as imagens são guardadas com as anotações feitas durante as inferências e os respetivos ficheiros (do tipo .txt) que contêm as coordenadas das *bounding boxes* das deteções, de forma a ser utilizado no próximo módulo.
- **Recorte da matrícula:** módulo responsável pelo recorte da região onde foi detetada a matrícula. Com os valores presentes em cada ficheiro são calculadas as coordenadas do canto superior esquerdo e o canto inferior direito da *bounding box*. Por fim, é utilizada uma biblioteca que permita processar digitalmente imagens ([OpenCV](#)) para efetuar o recorte da área da *bounding box* e são guardadas as novas imagens que apenas contêm a matrícula.
- **Pipeline de processamento digital de imagem:** módulo responsável por extrair os caracteres presentes nas imagens recortadas, obtidas no módulo anterior. De forma a atingir este objetivo existem várias abordagens que podem ser utilizadas:
  - [OCR](#) (*Optical Character Recognition*) - processo que converte imagens com texto em texto capaz de ser compreendido por computadores.
  - *Pipeline* de processamento digital de imagem - segmentação e extração dos caracteres utilizando técnicas de processamento de imagem.
- **Análise de texto e correção de erros:** módulo responsável pela análise do texto extraído, de todas as imagens, e correção de potenciais erros cometidos pela abordagem utilizado no módulo anterior.

## Matrículas portuguesas

As matrículas portuguesas apresentam três grupos de dois caracteres, números ou letras. A sua disposição varia com o ano em que o veículo foi registado. Desde 1937, quando ocorreu a implementação do sistema geral de matrículas, foram efetuadas diferentes alterações às matrículas existindo, neste momento, quatro formatos distintos:

Matrícula	Ano	Formato	Pares de caracteres
	Até 1992	AA-11-11	Par 1 – 2 letras Par 2 – 2 números Par 3 – 2 números
	1992 a 2005	11-11-AA	Par 1 – 2 números Par 2 – 2 números Par 3 – 2 letras
	2005 a 2020	11-AA-11	Par 1 – 2 números Par 2 – 2 letras Par 3 – 2 números
	Desde 2020	AA-11-AA	Par 1 – 2 letras Par 2 – 2 números Par 3 – 2 letras

## Tarefas a realizar por módulo

### Software/plataformas a utilizar:

Para a realização do projeto devem ser utilizados as seguintes ferramentas:

- [Google Colab](#): serviço remoto de notebooks Jupyter que fornece recursos computacionais (CPU e TPU). Será utilizado para desenvolver o algoritmo, treinar o modelo de *deep learning*, fazer inferências, utilizar o classificador de imagens, utilizar [Grounding Dino](#) e [Segment Anything Model](#);
- [Google Drive](#): armazenamento na cloud. Será utilizado para guardar os datasets e os notebooks do Google Colab;

- [Roboflow](#): plataforma de anotação de dados e treino de modelos de deep learning. Será utilizada para anotar, normalizar e dividir os dados para treino, validação e teste.

## Módulo 1 - Detecção da matrícula

### Construção do dataset (Roboflow)

- Construção do dataset: recolher imagens diurnas (tiradas manualmente, obtidas da Internet, vídeo, etc..) de veículos (carros, carrinhas, camiões, motos...) com matrícula portuguesa frontais e traseiras, sem muita angulação (vertical ou horizontal), em que os caracteres da matrícula sejam bem visíveis. A condição de iluminação das imagens também deve ser variada. Quantas mais imagens melhor, embora o tempo de anotação seja maior (sugestão: 200 a 500 imagens).

Imagens adequadas para o treino			
			
Imagens não adequadas para treino			
			
Matrícula não se encontra totalmente visível	Matrícula não se encontra totalmente visível	Iluminação impede a visualização de todos os caracteres	

- Criar conta no site [Roboflow](#);
- Correr o [tutorial](#) de funcionamento de *Roboflow*;
- Criar um projeto novo e seleccionar como tipo de projeto “*Object detection (bounding box)*”;
- No projeto criado, fazer upload das imagens;
- Anotar as imagens, gerar uma *bounding box* onde se encontra a matrícula;



- Selecionar a funcionalidade “Generate”:
  - Definir a distribuição de imagens a utilizar no treino, validação e teste (por exemplo: 70% para treino, 15% para validação e 15% para teste);
  - Pré-processamento das imagens: manter a opção “Auto-Orient” e a opção “Resize” selecionada. O ajuste de tamanho das imagens pré-definido é 640x640 (valor recomendado, 128 a 640, ajustar conforme os resultados do treino). O tamanho definido deve ser menor que o tamanho, em média, das imagens do dataset. Quanto maior for o tamanho definido, maior será o tempo de treino. Por outro lado, quanto menor for o tamanho, menor será o detalhe detetável pelo modelo de *deep learning*;
  - Aumento de dados: consiste em aplicar transformações às imagens de forma a gerar novas imagens. Útil quando se tem poucos dados, mas não é obrigatório utilizar. Transformações a aplicar: “Flip” horizontal, “Crop” 5% a 15%, “Rotation” -15% a 15 %, “Brightness”, com “Brighten” e “Darken” selecionados, -15% a 15%, “Blur” 1.25px, “Noise” 5%. Os valores devem ser ajustados com base nos dados recolhidos e os resultados do treino;
  - Por fim, pressionar o botão “Generate”.

## Google Drive:

- Criar uma nova conta *Google (gmail)*;
- Entrar no *Google Drive*;

- Criar uma pasta para os datasets, uma pasta para os ficheiros notebooks Colab (ipynb) e uma pasta para imagens que serão utilizadas para efetuar as inferências.

## Treino do modelo de *deep learning* (Google Colab)

- Definir que modelo YOLO utilizar, por defeito pode escolher o YOLOv5, usado nas aulas;
- Efetuar o download do repositório e guardá-lo no *Google Drive*;
- No projeto do *Roboflow*, selecionar a funcionalidade “Versions” e depois selecionar “Export Dataset” escolher o formato em que o dataset deve ser exportado, com base no modelo de *deep learning* a utilizar;
- Efetuar o download do dataset para a máquina do utilizador. A pasta contém três pastas e um ficheiro do tipo .yaml. As três pastas que contêm os dados para o treino (“train”), a validação (“valid”) e o teste (“test”). Cada pasta contém mais duas, uma com as imagens (“images”) e uma com os dados da anotação (“labels”). O ficheiro contém os caminhos para cada uma das pastas.
- Substituir os caminhos presentes no ficheiro “data.yaml” pelos caminhos onde as pastas estarão presentes, no *Google Colab*;
- Guardar o dataset comprimido (*Winrar*, *7zip*, ...) no *Google Drive*;
- Criar a ligação entre o *notebook* e o *Google Drive*. Isto faz com que tudo o que se encontra no *Google Drive* seja acessível a partir do *notebook*;
- Efetuar o *unzip* do dataset;
- Definir os parâmetros de treino a utilizar pelo modelo:
  - “epochs”: número de iterações efetuadas no treino;
  - “img”: tamanho da imagem utilizada como input;
  - “batch”: número de imagens utilizadas por cada iteração do treino;
  - “cfg”: modelo que vai ser utilizado para efetuar o treino (yolov\_s, yolov\_m, yolov\_l, yolov\_x, \_ deve ser substituído pela versão do YOLO que está a ser utilizada);
  - “weights”: utilizar pesos pré-treinados (opcional, yolov\_s.pt, yolov\_m.pt, yolov\_l.pt, yolov\_x.pt, \_ deve ser substituído pela versão do YOLO que está a ser utilizada);
  - “name”: nome da diretoria ou caminho da diretoria onde serão guardados os resultados do treino;
  - “bb\_thickness”: a espessura da linha da bounding box (pré-definido = 3);
- Efetuar treino;
- Recolher resultados do treino;
- Validar resultados do treino;

- Se os resultados da validação e do treino tiverem diferenças significativas, efetuar novo treino com diferentes parâmetros ou fazer ajustes ao dataset, caso contrário avançar para a próxima tarefa.

## Inferir sobre novas imagens

- Definir o caminho das imagens fornecidas para avaliação do modelo;
- Definir parâmetros da inferência:
  - “img”: tamanho de input da imagem
  - “conf”: valor mínimo da confiança que cada deteção tem de ter para que a bounding box seja colocada na imagem;
  - “save\_txt”: guarda as imagens e as *labels* (ficheiros txt) das *bounding boxes* detetadas;
- Efetuar inferências;



- Avançar para o próximo módulo.

## Módulo 2 - Recorte da imagem com base nas coordenadas da *bounding box*

- Obter os caminhos dos ficheiros com as coordenadas das *bounding boxes* (utilizar biblioteca “glob”);
- Obter os caminhos de todas as imagens (utilizar biblioteca “glob”);
- Obter os valores presentes nos ficheiros (ler o ficheiro). Cada ficheiro é constituído por, pelo menos, uma linha de texto com cinco valores, tal como se explica no exemplo:






- Class (por exemplo, 1) – classe do objeto detetado, não relevante para a resolução deste problema;
- X (0.48) - o valor do ponto x, do centro da *bounding box*;
- Y (0.63) - o valor do ponto y, do centro da *bounding box*;
- Width (0.69) - o valor da largura da *bounding box*;
- Height (0.71) - o valor da altura da *bounding box*.

Exemplo de uma *label*:

```
0 0.49 0.616667 0.2225 0.0633333
```

- O nome de uma imagem e da sua *label* são exatamente iguais, apenas é diferente a extensão do ficheiro (“jpg” e “txt”, respetivamente). Com base nos nomes dos ficheiros efetuar para todas as imagens:
  - Ler a imagem;
  - Obter as dimensões da imagem (altura e largura, em pixéis), correspondente ao ficheiro;
  - Reverter a normalização das coordenadas geradas pelo modelo;
  - Calcular o ponto superior esquerdo e o canto inferior direito da *bounding box*;
  - Utilizar OpenCV para recortar a região da matrícula;

	
Imagem recortada com a espessura da <i>bounding box</i>	Imagem recortada sem a espessura da <i>bounding box</i>

- Redimensionar as imagens para que tenham, por exemplo, 300x75 pixéis.
- Guardar as imagens numa pasta, no *Google Drive*;
- Avançar para o próximo módulo.

### Módulo 3 - *Pipeline* de processamento digital de imagem

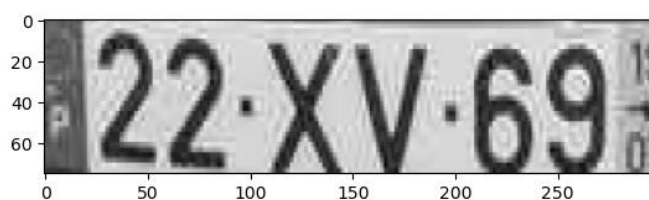
Com vista a extrair os caracteres da matrícula podem ser utilizadas várias abordagens:

#### ▪ **Abordagem 1 - Utilização de uma biblioteca OCR (Optical Character Recognition):**

- Instalar a biblioteca [OCR](#) ([PaddleOCR](#));
- Carregar o modelo responsável pelo reconhecimento do texto;
- Obter os caminhos das imagens recortadas (utilizar biblioteca “[glob](#)”);
- Aplicar o OCR sobre as imagens;
- Guardar os resultados do OCR num ficheiro “txt”.

#### ▪ **Abordagem 2 - Aplicação do método de Otsu:**

- Obter os caminhos das imagens recortadas (utilizar biblioteca “[glob](#)”);
- Ler as imagens;
- Converter a imagem de cores para preto e branco;



- Aplicar [filtros](#) antes ou depois do método de Otsu;
- Aplicar o [método de Otsu](#) (dá origem a uma nova imagem em que os pixéis ou são pretos ou brancos);





- A nova imagem (binarizada) tem de ter um maior número de pixéis pretos que brancos. Por isso, calcular o número de pixéis pretos e brancos, e no caso do número de pixéis brancos ser maior que o número de pixéis pretos inverter as cores utilizando a biblioteca "OpenCV";
- Aplicar operações morfológicas (erosão, dilatação, abertura, fecho);



- Fazer análise de componentes ("cv2.connectedComponents"), para filtrar as áreas que poderão ser caracteres do resto da matrícula, com base num limite mínimo e máximo de pixéis;
- Construir uma máscara onde só estarão presentes os caracteres;



- Utilizar o método ["findContours"](#) da biblioteca "OpenCV" com os seguintes argumentos;
- Ordenar os "contours" obtidos, da esquerda para a direita;
- Com base nas dimensões dos "contours" (altura, largura, área) extrair os caracteres. Alguns dos "contours" presentes na máscara podem não ser caracteres;
- Com base nos valores definidos extrair os caracteres. Isto gera uma nova imagem apenas do caracter, em que a letra ou número se encontra a branco e o fundo é preto. Será aplicado a todos os "contours" dentro dos mínimos e máximos definidos, que neste caso serão seis, um por cada caracter;



- Utilizar a biblioteca "OpenCV" para redimensionar as novas imagens (20x20);
- Utilizar o classificador de caracteres para classificar cada imagem;
- Guardar os resultados (string com 6 caracteres) num ficheiro txt.

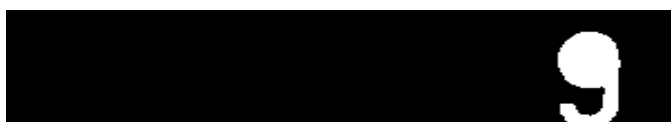
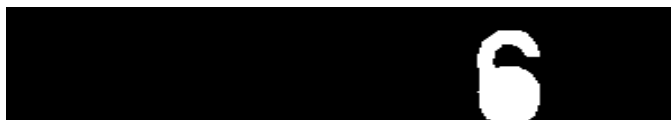
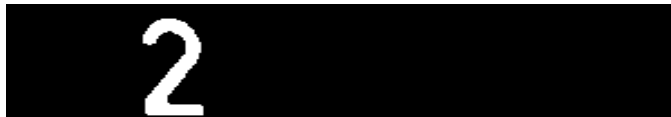
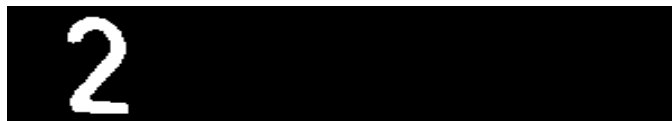
### ▪ **Abordagem 3 - Utilização da biblioteca Grounding Dino (deteção de caracteres) e Segment Anything Model (segmentação de caracteres):**

- Instalar bibliotecas [Grounding Dino](#) (GD) e [Segment Anything Model](#) (SAM);

- Obter os caminhos das imagens recortadas (utilizar biblioteca “[glob](#)”);
- Ler as imagens;
- Definir os parâmetros a utilizar no modelo GD:
  - “prompt”: palavra ou frase utilizada para detetar objetos nas imagens;
  - “box\_threshold”: define o valor mínimo de similaridade entre as *bounding boxes*;
  - “text\_threshold”: define o valor mínimo de similaridade que as palavras têm de ter quando são efetuadas as previsões das *labels* (com base no “prompt”).
- Aplicar a imagem sobre o modelo GD;



- Aplicar a nova imagem sobre o modelo SAM;
- Normalizar as coordenadas das *bounding boxes*;
- Obter as máscaras geradas para cada *bounding box* (imagens em que o caracter se encontra a branco e o fundo preto – a imagem terá as dimensões da imagem original e não apenas o caracter);



- Criar uma máscara que irá conter todas as máscaras obtidas;



- Utilizar o método [“findContours”](#) da biblioteca “OpenCV” com os seguintes argumentos;
- Com base nos valores definidos extrair os caracteres. Isto gera uma nova imagem apenas do caracter, em que a letra ou número se encontra a branco e o fundo é preto. Será aplicado a todos os “contours” dentro dos mínimos e máximos definidos, que neste caso serão seis, um por cada caracter;



- Utilizar a biblioteca “OpenCV” para redimensionar as novas imagens (20x20);
- Utilizar o classificador de caracteres para classificar cada imagem;
- Guardar os resultados (*string* com 6 caracteres) num ficheiro txt.

#### **Módulo 4 - Análise de texto e correção de erros**

- Verificar que erros existem nos resultados, cometidos pelo OCR ou classificação, nomeadamente caracteres que não sejam letras ou números com “.”, “-”, “ ”, entre outros e casos em que existe um par que contém uma letra e ou o número “0” ou o número “1” em que é necessário substituir os números pelas letras ou “O” ou “I”, respetivamente;
- Desenvolver um algoritmo que verifique o formato dos resultados e corrija os erros existentes;
- Guardar os resultados corrigidos;
- Comparar os resultados obtidos com as matrículas do dataset de teste.

**Apresentação e submissão do trabalho:** Os trabalhos podem ser realizados por grupos de 3 alunos. O código fonte da implementação do projeto e a apresentação PowerPoint usada para reportar os

principais resultados e conclusões do trabalho devem submetidos no SIDE. A apresentação oral do trabalho e a submissão no SIDE pode ser feita na última aula do semestre ou na data das avaliações por exame.