**Pesquisa para Projeto Informático**

**CNN**

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and the reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

VER O ARTIGO COMPLETO PARA PERCEBER MAIS SOBRE CNN

Fonte: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

Redes neuronais convencionais (CNN)

A diferença entre métodos de *classificação de objetos* (como as redes da seção anterior) e métodos de *deteção de objetos*, é que na deteção de objetos você deseja:

* identificar ***quantos*** objetos de uma determinada categoria se encontram na imagem;
* identificar ***onde*** na imagem cada objeto se encontra.

Its different than image classifier as it identifies and localizes multiple objects within an image, whereas an image classifier assigns a single label to the entire image based on its predominant content.

**You Only Look Once: YOLO**

Em YOLO uma única rede convolucional prediz tanto os bounding boxes quanto as probabilidades de pertinencia a classe de cada objeto detectado. Para isso, YOLO funciona da seguinte forma:

1. toma-se uma imagem e divide-se-a em um *grid* *SxS* de células;
2. usando o grid como referência, gera-se *m* bounding boxes;
3. bounding boxes com probabilidade acima de um limiar são selecionados e usados para localizar o objeto dentro da imagem.

YOLO muito mais rápido (45 fps no set dos autores -> até duas ordens de grandeza) mais rápido do que algoritmos contemporâneos. Sua maior falha é inacurácia com objetos pequenos na imagem.

Cada célula do grid é usada para predizer *B* bounding boxes (*bbox*) e *C* probabilidades de classe. Uma predição de *bbox* possui 5 componentes: *(x, y, w, h, confiança)*. As coordenadas *(x, y)* representam o centro do bbox relativo à localização da célula (se o centro de um bbox não cair em uma célula ela não será responsável por ele e não vai representá-lo -> células só possuem uma referência a objetos cujo centro cair dentro delas). Essa coordenadas são normalizadas para [0, 1]. As dimensões do bbox *(w, h)*também são normalizadas para [0, 1], relativamente ao tamanho da imagem.

Uma imagem com bicicleta

Descrição gerada automaticamente

Fonte: <https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deteccao-de-objetos-em-imagens/>

**What Is an Image?**

Before we jump into image processing, we need to first understand what exactly constitutes an image. An image is represented by its dimensions (height and width) based on the number of pixels. For example, if the dimensions of an image are 500 x 400 (width x height), the total number of pixels in the image is 200000.

This pixel is a point on the image that takes on a specific shade, opacity or color. It is usually represented in one of the following:

* Grayscale - A pixel is an integer with a value between 0 to 255 (0 is completely black and 255 is completely white).
* RGB - A pixel is made up of 3 integers between 0 to 255 (the integers represent the intensity of red, green, and blue).
* RGBA - It is an extension of RGB with an added alpha field, which represents the opacity of the image.

Image processing requires fixed sequences of operations that are performed at each pixel of an image. The image processor performs the first sequence of operations on the image, pixel by pixel. Once this is fully done, it will begin to perform the second operation, and so on. The output value of these operations can be computed at any pixel of the image.

## **What Is Image Processing?**

Image processing is the process of transforming an image into a digital form and performing certain operations to get some useful information from it. The image processing system usually treats all images as 2D signals when applying certain predetermined signal processing methods.

## **Types of Image Processing**

There are five main types of image processing:

* Visualization - Find objects that are not visible in the image
* Recognition - Distinguish or detect objects in the image
* Sharpening and restoration - Create an enhanced image from the original image
* Pattern recognition - Measure the various patterns around the objects in the image
* Retrieval - Browse and search images from a large database of digital images that are similar to the original image

Fonte: <https://www.simplilearn.com/image-processing-article>

Object detectors typically work by analyzing the visual information in an image or video frame and producing bounding boxes that outline the regions where objects are detected. In addition to localizing the objects, object detectors often classify the detected objects into predefined categories or classes, indicating what type of object has been found.

Uma imagem com texto, captura de ecrã, diagrama, Paralelo

Descrição gerada automaticamente

There are various approaches to object detection, including traditional methods and deep learning-based methods. Traditional methods often involve handcrafted feature extraction and machine learning algorithms, such Viola-Jones Detector, Histogram of Oriented Gradients (HOG) Detector, and Deformable Part-based Model (DPM).

**TRADITIONAL ALGORITHMS**

There are various approaches to object detection, including traditional methods and deep learning-based methods. Traditional methods often involve handcrafted feature extraction and machine learning algorithms, such Viola-Jones Detector, Histogram of Oriented Gradients (HOG) Detector, and Deformable Part-based Model (DPM).

· **Viola-Jones Detector**: Haar-like features, which are simple rectangular features capturing intensity differences, are manually defined. These features are selected based on their ability to differentiate between object and non-object regions.

· **HOG Detector**: Histograms of oriented gradients are computed within local cells, capturing edge orientations in the image. These histograms are manually designed to capture the appearance of edges and contours in different orientations.

· **DPM**: While DPM models consider hierarchical part-based structures, the appearance and geometric models for each part are manually defined. The model's ability to account for deformations and spatial relationships is also designed based on prior knowledge.

These traditional approaches have contributed significantly to the development of object detection methods, offering insights into handling different object characteristics and challenges.

Deep learning-based approaches, on the other hand, have gained significant attention and success in recent years. Convolutional Neural Networks (CNNs) are a key technology in deep learning for object detection. They can learn to automatically extract relevant features from images and learn complex patterns that are representative of different object categories. There are two types of object detection architectures based on CNN, one-stage and two-stage detectors as below:

**One-Stage Detectors**:

One-stage detectors are designed to directly predict bounding box coordinates and class probabilities for multiple objects in a single pass through the network. These detectors are known for their simplicity and efficiency, as they eliminate the need for a separate proposal generation step. The key idea is to densely sample potential object locations and then predict the presence of an object and its associated bounding box in a single shot. Popular one-stage detectors are YOLO, SSD, RetinaNet, CenterNet, and YOLOX.

One-stage detectors are known for their speed and real-time capabilities. They perform detection in a single pass, which makes them faster for real-time applications while they end to have slightly lower accuracy compared to two-stage detectors. They may struggle with detecting small objects and handling object instances with significant size variations.

**Architecture (YOLO):**

· **Input Layer**: Accepts input images.

· **Convolutional Layers**: A series of convolutional layers for feature extraction.

· **Downsampling Layers**: Downsample the feature maps using strides or max-pooling.

· **1x1 Convolutional Layers**: Decrease the depth of feature maps and extract more compact features.

· **Upsampling Layers**: Increase the resolution of feature maps using techniques like nearest-neighbor upsampling.

· **Concatenation Layers**: Combine feature maps from different scales for multi-scale detection.

· **Detection Head**: Final layers for object classification and bounding box prediction.

**Two-Stage Detectors**:

Two-stage detectors, on the other hand, follow a two-step process. In the first stage, these detectors generate a set of region proposals or candidate object locations. These proposals are then refined and classified in the second stage. Two-stage detectors tend to have higher accuracy but may be slower due to the additional proposal generation step. Popular two-stage detectors are R-CNN, SPPNet, Fast R-CNN, Faster R-CNN, FPN, and S2ANet.

Two-stage detectors achieve higher accuracy than one-stage detectors. The two-stage process of region proposal and refinement allow for more accurate localization and classification while they are usually slower due to the additional region proposal step.

**Architecture (R-CNN):**

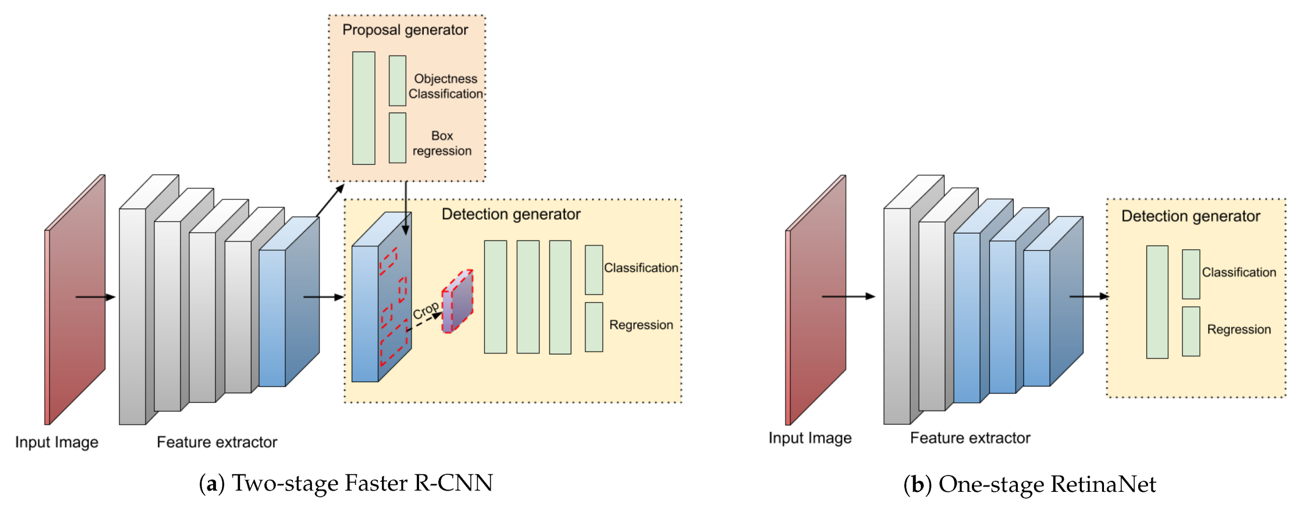
· **Input Layer**: Accepts input images.

· **Backbone Convolutional Layers**: A deep convolutional network (e.g., VGG, ResNet) for feature extraction.

· **Region Proposal Network (RPN)**: Generates potential object regions (proposals) based on the backbone features.

· **RoI Pooling/Align Layer**: Extracts fixed-size feature maps from each proposal for further processing.

· **Fully Connected Layers**: Shared layers for classifying and regressing bounding boxes for each proposal.

· **Object Detection Head**: Additional fully connected layers for refining classifications and bounding box predictions.

Fonte: https://www.linkedin.com/pulse/object-detection-from-traditional-techniques-modern-samer/

**Novo Artigo (https://neptune.ai/blog/object-detection-algorithms-and-libraries)**  
  
Object detection finds and identifies things in images, and it’s one of the biggest accomplishments of deep learning and image processing. One of the common approaches to creating localizations for objects is with the help of bounding boxes. You can train an object detection model to identify and detect more than one specific object, so it’s versatile.

[Object detection](https://machinelearningmastery.com/object-recognition-with-deep-learning/) models are usually trained to detect the presence of specific objects. The constructed models can be used in images, videos, or real-time operations. Even before the deep learning methodologies and modern-day image processing technologies, object detection had a high scope of interest. Certain methods (like SIFT and HOG with their feature and edge extraction techniques) had success with object detection, and there were relatively few other competitors in this field.

With the introduction of convolutional [neural networks](https://neptune.ai/blog/neural-network-guide) (CNNs) and the adaption of computer vision technologies, object detection became much more common in the current generation. The new wave of object detection with deep learning approaches opens up seemingly endless possibilities.

One-shot learning

What is one-shot learning? One-shot learning is a specialized approach in machine learning where the model can learn from only a single example of each class, making it a particularly compelling method to address problems with limited data. This approach performs exceedingly well even when training data is scarce, in contrast to traditional machine learning models that demand extensive datasets to achieve significant accuracy.

**ALGORITHMS**

**-HOG**

**-CNN , FAST CNN**

**-FASTER CNN**

The time for prediction is faster compared to other CNN methods. While R-CNN usually takes around 40-50 seconds for the prediction of objects in an image, the Fast R-CNN takes around 2 seconds, but the Faster R-CNN returns the optimal result in just about 0.2 seconds.

**-Region-based Fully Convolutional Network (R-FCN)**

Region-based Fully Convolutional Networks or R-FCN is a region-based detector for object detection. Unlike other region-based detectors that apply a costly per-region subnetwork such as Fast R-CNN or Faster R-CNN, this region-based detector is fully convolutional with almost all computation shared on the entire image.

R-FCN consists of shared, fully convolutional architectures as is the case of FCN that is known to yield a better result than the Faster R-CNN. In this algorithm, all learnable weight layers are convolutional and are designed to classify the ROIs into object categories and backgrounds.

**-SSD**

The [single-shot detector](https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11) for multi-box predictions is one of the fastest ways to achieve the real-time computation of object detection tasks. While the Faster R-CNN methodologies can achieve high accuracies of prediction, the overall process is quite time-consuming and it requires the real-time task to run at about 7 frames per second, which is far from desirable.

The single-shot detector (SSD) solves this issue by improving the frames per second to almost five times more than the Faster R-CNN model. It removes the use of the region proposal network and instead makes use of multi-scale features and default boxes.

LIMITATIONS:

-The SSD, while boosting the performance significantly, suffers from decreasing the resolution of the images to a lower quality.

-The SSD architecture will typically perform worse than the Faster R-CNN for small-scale objects.

When To Use SSD? – The single-shot detector is often the preferred method. The main reason for using the single-shot detector is because we mainly prefer faster predictions on an image for detecting larger objects, where accuracy is not an extremely important concern. However, for more accurate predictions for smaller and precise objects, other methods must be considered.

**-SPP-NET**

Spatial Pyramid Pooling (SPP-net) is a network structure that can generate a fixed-length representation regardless of image size/scale. Pyramid pooling is said to be robust to object deformations, and SPP-net improves all CNN-based image classification methods. Using SPP-net, researchers can compute the feature maps from the entire image only once, and then pool features in arbitrary regions (sub-images) to generate fixed-length representations for training the detectors. This method avoids repeatedly computing the convolutional features.

**-YOLO**

You only look once ([YOLO](https://pjreddie.com/darknet/yolo/)) is one of the most popular model architectures and algorithms for object detection. Usually, the first concept found on a Google search for algorithms on object detection is the YOLO architecture. There are several versions of YOLO, which we will discuss in the upcoming sections. The YOLO model uses one of the best neural network archetypes to produce high accuracy and overall speed of processing. This speed and accuracy is the main reason for its popularity.

The YOLO architecture utilizes three primary terminologies to achieve its goal of object detection. Understanding these three techniques is quite significant to know why exactly this model performs so quickly and accurately in comparison to other object detection algorithms. The first concept in the YOLO model is residual blocks. In the first architectural design, they have used 7×7 residual blocks to create grids in the particular image.

Each of these grids acts as central points and a particular prediction for each of these grids is made accordingly. In the second technique, each of the central points for a particular prediction is considered for the creation of the bounding boxes. While the classification tasks work well for each grid, it’s more complex to segregate the bounding boxes for each of the predictions that are made. The third and final technique is the use of the intersection of union (IOU) to calculate the best bounding boxes for the particular object detection task.

**→ Advantages of YOLO**

The computation and processing speed of YOLO is quite high, especially in real-time compared to most of the other training methods and object detection algorithms.

Apart from the fast computing speed, the YOLO algorithm also manages to provide an overall high accuracy with the reduction of background errors seen in other methods.

The architecture of YOLO allows the model to learn and develop an understanding of numerous objects more efficiently.

**→ Limitations of YOLO**

Failure to detect smaller objects in an image or video because of the lower recall rate.

Cant’t detect two objects that are extremely close to each other due to the limitations of bounding boxes.

**→ Versions of YOLO**

The YOLO architecture is one of the most influential and successful object detection algorithms. With the introduction of the YOLO architecture in 2016, their consecutive versions YOLO v2 and YOLO v3 arrived in 2017 and 2018. While there was no new release in 2019, 2020 saw three quick releases: YOLO v4, YOLO v5, and PP-YOLO. Each of the newer versions of YOLO slightly improved on their previous ones. The tiny YOLO was also released to ensure that object detection could be supported on embedded devices.

When To Use YOLO? – While all the previously discussed methods perform quite well on images and sometimes video analysis for object detection, the YOLO architecture is one of the most preferred methods for performing object detection in real-time. It achieves high accuracy on most real-time processing tasks with a decent speed and frames per second depending on the device that you’re running the program on.

**-RETINANET**

The [RetinaNet](https://towardsdatascience.com/review-retinanet-focal-loss-object-detection-38fba6afabe4" \t "_blank) model introduced in 2017 became one of the best models with single-shot object detection capabilities that could surpass other popular object detection algorithms during this time. When the RetinaNet Architecture was released, the object detection capabilities exceeded that of the Yolo v2 and the SSD models. While maintaining the same speed as these models, it was also able to compete with the R-CNN family in terms of accuracy. Due to these reasons, the RetinaNet model finds a high usage in detecting objects through satellite imagery.

The RetinaNet architecture is built in such a way that the previous issues of single-shot detectors are somewhat balanced out to produce more effective and efficient results. In this model architecture, the cross-entropy loss in the previous models is replaced with the focal loss. The focal loss handles the [class imbalance problems](https://neptune.ai/blog/how-to-deal-with-imbalanced-classification-and-regression-data) that exist in architectures like YOLO and SSD. The RetinaNet model is a combination of three main entities.

RetinaNet is built using three factors, namely the ResNet model (specifically ResNet-101), the feature pyramid network (FPN), and the focal loss. The feature pyramid network is one of the best methods for overcoming a majority of the shortcomings of the previous architecture. It helps in combining the semantic rich features of lower resolution images with that of the semantically weak features of the higher resolution images.

In the final output, we can create both the classification and regression models similar to the other object detection methods discussed previously. The classification network is used for appropriate multi-class predictions, while the regression network is built to predict the appropriate bounding boxes for the classified entities.

When to use RetinaNet? – RetinaNet is currently one of the best methods for object detection in a number of different tasks. It can be used as a replacement for a single-shot detector for a multitude of tasks to achieve quick and accurate results for images.

YOLO3 VS RetinaNet : <https://ai.stackexchange.com/questions/26635/what-are-the-main-differences-between-yolov3-and-retinanet-object-detection-algo>

**YOLOvsRETINANET**

**Background:**

The correct identification of pills is very important to ensure the safe administration of drugs to patients. Here, we use three current mainstream object detection models, namely RetinaNet, Single Shot Multi-Box Detector (SSD), and You Only Look Once v3(YOLO v3), to identify pills and compare the associated performance.

**Results**

The mean average precision (MAP) of RetinaNet reached 82.89%, but the frames per second (FPS) is only one third of YOLO v3, which makes it difficult to achieve real-time performance. SSD does not perform as well on the indicators of MAP and FPS. Although the MAP of YOLO v3 is slightly lower than the others (80.69%), it has a significant advantage in terms of detection speed. YOLO v3 also performed better when tasked with hard sample detection, and therefore the model is more suitable for deployment in hospital equipment.

**Conclusion**:

Our study reveals that object detection can be applied for real-time pill identification in a hospital pharmacy, and YOLO v3 exhibits an advantage in detection speed while maintaining a satisfactory MAP.

Study: <https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01691-8>

**PYTHON GUI FRAMEWORKS**

(<https://www.bairesdev.com/blog/best-python-gui-libraries/>)

Tao a dizer PYQT5

https://www.linkedin.com/pulse/guis-com-python-e-pyqt5-introdu%C3%A7%C3%A3o-guilherme-trevisan-linhares/?originalSubdomain=pt

### Methods of Image Labeling (<https://datagen.tech/guides/image-annotation/image-labeling/>)

#### **Manual Image Annotations**

A common way to label images is manual annotation. This is the process of manually defining labels for an entire image, or drawing regions in an image and adding textual descriptions of each region.

Image annotation sets a standard, which a computer vision algorithm tries to learn from. This means that any errors in labeling will be adopted by the algorithm, reducing its accuracy. This means that accurate image labeling is a critical task in training neural networks.

Manual annotation is typically assisted by tools that allow operators to rotate through a large number of images, draw regions on an image and assign labels, and save this data to a standardized format that can be used for data training.

Manual image annotation presents several challenges:

1. Labels can be inconsistent if there are multiple annotators, and to resolve this, images need to be labeled several times with majority voting.
2. Manual labeling is time consuming. Annotators must be meticulously trained and the process requires many iterations. This can delay time to market for computer vision projects.
3. Manual labeling is costly and is difficult to scale to achieve large datasets.

#### **Semi-Automated Image Annotations**

Manual image annotation is a time-consuming task, and for some computer vision algorithms, can be difficult for humans to achieve. For example, some algorithms require creating pixel maps indicating the exact boundary of multiple objects in an image.

Automated annotation tools can assist manual annotators, by attempting to detect object boundaries in an image, and providing a starting point for the annotator. Automated annotation algorithms are not completely accurate, but they can save time for human annotators by providing at least a partial map of objects in the image.

#### **Synthetic Image Labeling**

Synthetic image labeling is an accurate and cost-effective technique which can replace manual annotations. It involves automatically generating images that are similar to real data, in accordance with criteria set by the operator. For example, it is possible to create a synthetic database of real-life objects or human faces, which are similar but not identical to real objects.

The main advantage of synthetic images is that labels are known in advance—for example, the operator automatically generates images containing tables and chairs. In this case, the algorithm generating the images can automatically provide the bounding boxes of the tables and chairs in each image.

There are three common approaches to generating synthetic images:

* **Variational Autoencoders (VAE)**—these are algorithms that start from existing data, create a new data distribution, and map it back to the original space using an encoder-decoder method.
* **Generative Adversarial Networks (GAN)**—these are models that pit two neural networks against each other. One neural network attempts to create fake images, while the other tries to distinguish real and fake images. Over time, the system becomes able to generate photorealistic images that are difficult to distinguish from real ones.
* **Neural Radiance Fields (NeRF)**—this model takes a series of images describing a 3D scene and automatically renders novel, additional viewpoints from the same scene. It works by computing a five-dimensional ray function to generate each voxel of the target image.

Thread vs Threading  
  
In Python 3, thread has been renamed to \_thread. It is infrastructure code that is used to implement threading, and normal Python code shouldn't be going anywhere near it.

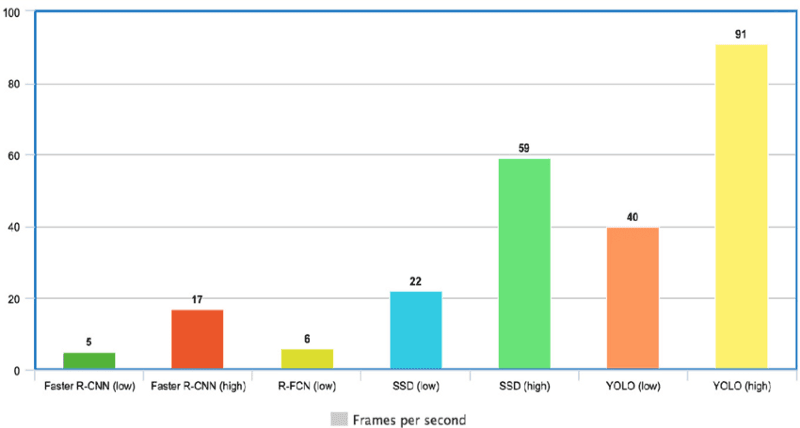
\_thread exposes a fairly raw view of the underlying OS level processes. This is almost never what you want, hence the rename in Py3k to indicate that it is really just an implementation detail.

threading adds some additional automatic accounting, as well as several convenience utilities, all of which makes it the preferred option for standard Python code.

**\_THREAD**: <https://docs.python.org/3/library/_thread.html#module-_thread>  
**THREADING**: <https://docs.python.org/3/library/threading.html#module-threading>

<https://stackoverflow.com/questions/3044580/multiprocessing-vs-threading-python>

https://www.datacamp.com/blog/yolo-object-detection-explained



<https://arxiv.org/abs/1506.02640>

<https://arxiv.org/pdf/1612.08242.pdf>

<https://builtin.com/machine-learning/non-maximum-suppression>

<https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31>

#### **Uso de threads ou processos ou dar predict de uma imagem com o yolo:**

Se usarmos threads temos de guardar os diferentes frames da imagem para que sejam apresentados ambos no ecrã quando se usa a função yolo.track(), já quando usamos processos podemos passar diretamente o frame para a função que iram ser mostradas as x janelas (x = numero de camaras)

**Metodologia de trabalho**

SCRUM.

**Arquitetura do programa**Porque escolher grab/retrieve em vez dads outras 2 (thread para captura e previsão; open/close)

**Comunicação entre os 2 programas:**quele vs sockets

1. **Sockets:**
   * Sockets provide a bidirectional communication channel between two processes over a network.
   * They are typically used for communication between processes running on different machines or the same machine but different addresses.
   * Sockets are suitable for scenarios where you need to communicate between processes that are not necessarily running within the same Python interpreter or on the same machine.
2. **Queues:**
   * Queues, such as the **queue.Queue** class in Python, provide a way for multiple threads or processes within the same Python interpreter to communicate by passing messages.
   * They are often used for communication between threads or processes running within the same Python program or on the same machine.
   * Queues are suitable for scenarios where you need to pass data between threads or processes within the same Python program.

In this case, where you have a GUI running in one thread and a script generating images in another thread within the same Python program, using a queue is a suitable approach. Queues are designed for communication between threads or processes within the same Python program and provide a convenient and thread-safe way to pass data.

Using a queue (**queue.Queue**) for communication between the image-generating script and the GUI thread allows you to safely pass generated images from the script to the GUI without worrying about concurrency issues such as race conditions or deadlocks.

**Thread para Atualizar DIspositivos**

Tivemos de substituir Thread(target=self.listar\_Thread).start() por

listar\_thread = ListarThread(self)  
listar\_thread.finished.connect(self.listar\_dispositivos)  
listar\_thread.start()

porque como agora temos no ConfigurarDipositivo window 2 widgets (páginas) através do stackedWidget, a thread começou a dar erro e não funcionava por não estar na Thread principal e (explicação chatgpt): “O erro que você está enfrentando ao clicar em "Procurar Dispositivos" após dividir em duas páginas provavelmente está relacionado ao fato de que você está tentando acessar elementos da interface do usuário (UI) de dentro de uma thread. No PyQt, todas as interações com a interface do usuário devem ser feitas na thread principal. Uma maneira de resolver isso é usar sinais e slots para atualizar a interface do usuário a partir da thread. Aqui está como você pode fazer isso:”

Basicamnete no PyQt a manipulação da interface (UI) tem de ser feita na thread principal

**Metodologia de trbalho**  
<https://asana.com/pt/resources/agile-methodology>

<https://www.lucidchart.com/blog/agile-vs-waterfall-vs-kanban-vs-scrum>

<https://www.altamira.ai/blog/scrum-vs-other-methodologies/>

https://www.linkedin.com/pulse/agile-vs-traditional-which-method-right-you-carlos-goetz/