Image processing for mapping purposes (014855)

Winter 2020

“Low sample size instance segmentation”

Presented by - Zaruhi (Zara) Papyan
ID 316648799

1 Introduction

A bird is a vertebrate[1] animal adapted for flight. Everything about the anatomy of a bird reflects its ability to fly and can be used to distinguish them from other spices. The wings, for example, are shaped to create lift. The leading edge is thicker than the back edge, and they are covered in feathers that narrow to a point. Furthermore, birds have a unique digestive system that allows them to eat when they can—usually on the fly—and digest later – how is this relevant to your project? I thought that you specified previous features cause you'll use them in order to identify birds later on? . – see previous remark. You can write about the shape of their beaks – instead of having a mouth "inside" their body it stands outside or whatever – but I think in this course you're supposed to identify properties which will help the machine learn and distinct them – no ?

Birds are migrating animals they have a remarkable homing instinct, allowing them to return to the same area year after year, even when their migration takes them halfway around the world. How this remarkable feat is accomplished has been the topic of many studies. The bird most known for its migration skills is the homing pigeon. This class of birds have been used extensively as test subjects in order to develop a better understanding of migration and homing abilities. *Floriano Papi* from the University of Pisa suggests that homing pigeons may use an olfactory map[2], i.e. the pigeon can smell its direction to and from a target. By moving closer to the target – the beginning point smell will get weaker while the target's smell will presumably grow stronger. In theory, the gradient map of odors that could be created, which will cover a distance of up to 310 miles, will be able to provide some directional information, even if the pigeon were suddenly dropped into a new location.

A couple of weeks ago my neighbor Gideon, who owns several birds' spices including parrots, pigeons and chickens, decided to experiment with the birds migrations abilities and train his parrots to fly back home from a faraway distance. Gideon search for a way to monitor easily which bird turned back and which turns missing. In order to help him with this mission, came the idea for the course project: develop a program that will automatically identify the parrots and even distinguish their types in the future from a single image. (The next step could be a video detection).

There were two main challenges:
1. Identify the parrots over every background the photos might have
2.Distinguish the parrots from other birds' spices

In this project I focused in the first challenge.

[1] **Vertebrate** are animals with backbones
[2] **Olfactory map** is a map based on the smell sense

2 Background

The course' final project purpose is to use a machine in order to successfully identify chosen objects in given pictures. Two popular methods to achieve that are:
1. classical image processing technique i.e. threshold with histogram-based methods
2. artificial intelligence (AI) and more precisely the Mask R-CNN for instance segmentation.
In order to implement the first method, which we studied during the course, the data, i.e. photos, must follow certain requirements. As it happens, my data couldn't fulfil those requirements, as the photos where taken by me in complex environment. Those, I used it as an opportunity to extend my knowledge of image processing, and study and implement the second method.

Mask R-CNN, Mask Rapid Convolutional Neural Networks -includes 4 tasks: Image Classification, Object Localization, Object Detection, Semantic Segmentation, Instance Segmentation etc. as shown in **Figure 2.1**.

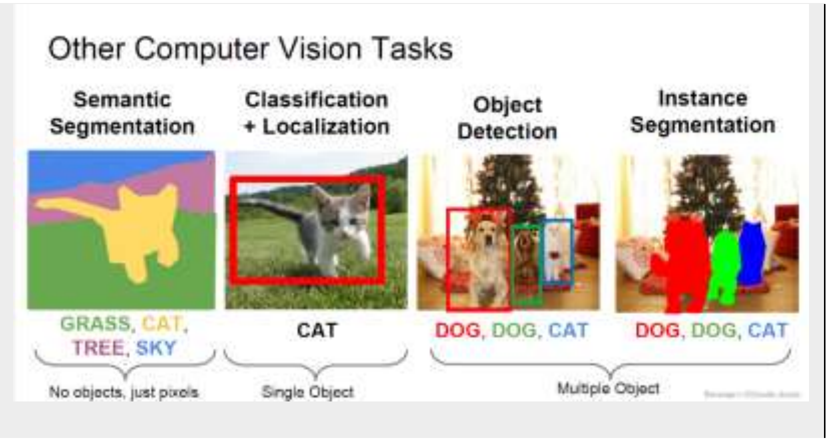


Figure 2.1: Computer Vision Tasks
taken from Stanford course CS-231 .2017. lecture 11

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through machine perception, by labeling or clustering the raw input. The patterns NN recognize are numerical, contained in vectors, into which all real-world data (be it images, sound, text, or time series) must be translated to. It defers from the ideas we were introduced in the course by deflecting the most difficult step of identifying and specifying object's attributes. Those, for better and worse, NN makes human deduction and intervention redundant, and might even recognize patterns and attributes which human senses are not honed to identify.

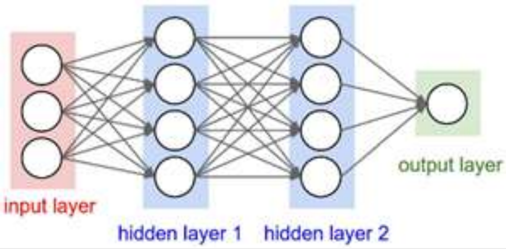
The use of NNs helps cluster and classify objects that may be understood as layers[3] on top of the stored data. It helps to group unlabeled- data[4] according to similarities among example inputs and to classify data when there is a labeled dataset to train on. Let us note that deep NN's involve algorithms for reinforcement learning, classification, and regression. Notice, there is a difference between a regular NN and a convolutional NN (*ConvNet*) as shown and explained in **Figure 2.2**. A ConvNet is made up of

[3] The core building block of NNs is the **layer**, a data-processing module that can be thought as a filter for data

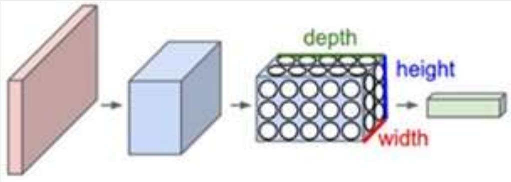
[4] **Labeled data** is data that has been annotated and formatted with one or multiple labels. When a **label** is the thing we predict; It is the output for an input feature.

[5] The **bounding box** is a rectangular box that can be determined by the x, y axis coordinates in the upper-left corner and the x, y axis coordinates in the lower-right corner.

Layers. Every Layer has a simple API: It transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters.



(NN)



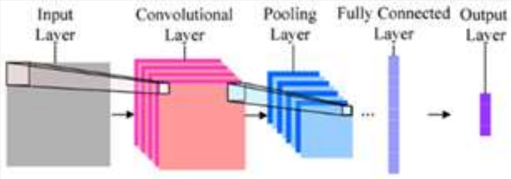
(CNN)

Figure 2.2: [Upper Left] A regular three-layer Neural Network. [Lower left]: A ConvNet arranges its neurons in 3-dim (width, height, depth), as visualized in one of the layers.

Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

taken from Stanford course CS-231 2017, lectures 4-5

Basically, the training of a CNN involves, finding of the right values on each of the filters so that an input image when passed through the multiple layers, activates certain neurons of the last layer so as to predict the correct class.



(Representation of ConvNet)

To better understand core Neural Network (NN), let us introduce a definition: classification, detection, and segmentation. Classification task depends upon labeled datasets, i.e. the human must transfer his knowledge to an appropriate dataset in order to unable the NN to learn the correlation between the two. It's called: supervised learning.

Object Detection is like classical object recognition but with only two classes of *object classification*: object bounding-boxes[5] and non-object bounding boxes.

Semantic and Instance Segmentation. The first annotates images s.t. each pixel in the image belongs to a single class, as opposed to *object detection* where the bounding boxes of objects can overlap over each other. At the same time, the latest dives a bit deeper, it identifies, for each pixel, the object instance it belongs to.

NN in this project will be used to **detect** parrots in an image. I will use sort of object detection (bounding boxes) and will include the idea of segmentation (labeling).

On January the 24th, 2018 an article with the headline “$\mathrm{Mask\ R - CNN}$“ was published by *Kaiming He, Georgia Gkioxari, Piotr Doll´ar and Ross Girshick*. It introduced a general framework for object instance segmentation.

From the article’s abstract:

> “… Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.”

Couple of technical details about the code I will use:
[1] The framework that was chosen for the project was **PyTorch** (primarily developed by Facebook's AI Research lab). Since we are already familiar from the course with NumPy arrays, the PyTorch Tensor class will be immediately familiar. PyTorch similar to NumPy, but have GPU acceleration and automatic computation of gradients, which makes it suitable for calculating backward pass data automatically starting from a forward expression. The difference is shown in **Figure 2.3**
[2] The data set includes images that contain parrots.
[3] Masks[6] for each parrot that were made with a help of an online tool called “$labelbox$[7]” by me.
[4] It was convenient to use the $google.colab.drive.mount$ function to mount the drive with the colab notebook and a couple of more install code lines were added to the algorithm for more convenient work.

An appendix is added to the work with a short explanation of how to run the project.

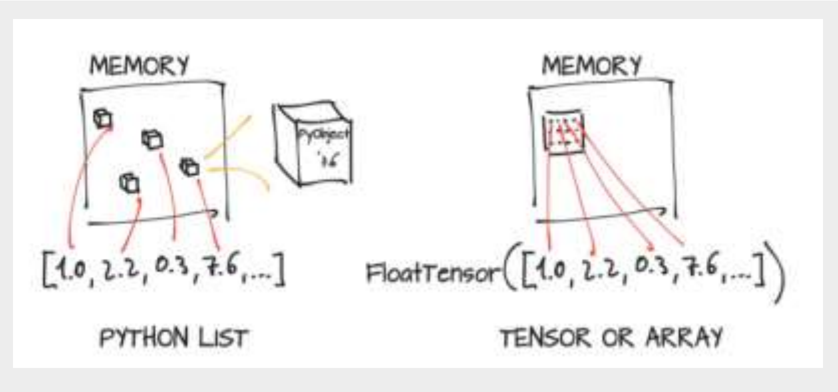


Figure 2.3: List vs Tensors

taken from a book “*Deep-Learning-with-PyTorch” written by Eli Stevens and Luca Antiga* Page 20

[6] **Masking** is an image processing method in which we define a small 'image piece' and use it to modify a larger image.

[7] **Labelbox** is a tool that allows us to create binary images, in which each pixel is zero or one, depending on whether the pixel contains a fly

3 A description of the Model

A detection of an object is considered as a classical problem in the field of image processing. Many tools have been developed for this area and stand to our disposal. In the latest years, with the entrance of the neural networks to our lives, assignments like detection got some additional tools and approaches. On the next section let us primarily present an approach from the NN's field, that was discovered at the year of 2018 called by "An instance segmentation with Mask R-CNN".

Let us first remind ourselves the project in general charts: there is a data of images and an object that needs to be detected in it. When the data is a bunch of individual images s.t. every image is composed of parrots and a different complicated background (bird houses, the cage itself, branches and ropes) as shown in **Figure 3.1**. The main idea that the algorithm used in the project came to solve was – to be able to detect all the parrots in each image from the data set. When the chosen answer to this question was using Instance segmentation with Mask R-CNN.

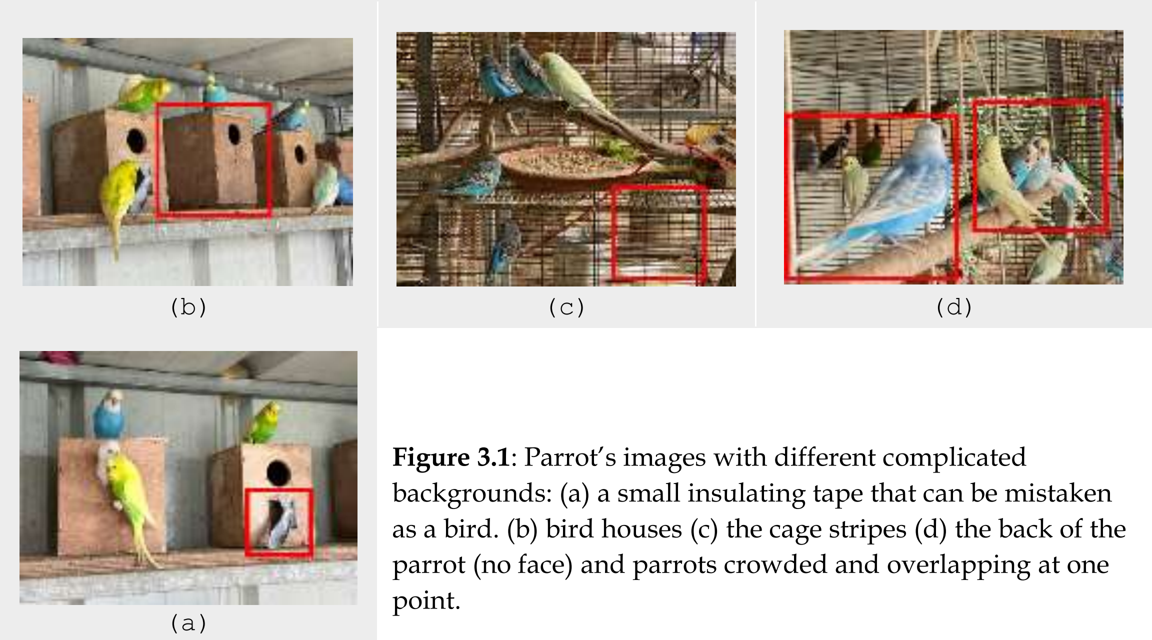
The main reason for choosing this approach was curiosity; to see the differences between this method and those studied in the course - during the working processes, and of course at the finale result. Another reason was to investigate whether the new approach can bypass or even overtake the downsides of regular image processing methods.

Since the main requirement of the project was to identify each pixel to which class it belongs – it was clear that a sort of segmentation is required. The preference was toward instance segmentation because that method could also identify the object instance it belongs to (as shown in **Figure 2.1**.). The above article served as the main building block for the algorithm.

The first step of the algorithm was to find a tool to create some **segmentation masks** for the parrots, i.e. determine for each pixel if it is zero or one, depending on whether the pixel contains a parrot or not. It was accomplished, even in complex images, using online pen and the free tool LabelBox. An example for an output image is shown in **Figure 3.2**. For the beginning of the training, labeling only two images was actually enough to get pretty good results. Also, it was important that each parrot would be drawn as an individual object at the masks tool. The exported final data was a JSON type file, i.e. the data was encrypted and looked like a 'dictionary typescript'[8]. The JSON file included some useful information for the future calculation of the bounding box coordinates for each mask.

The amount of images I had to generate the Data was relatively small comparing to regular network usual required amount for training. Therefore came the idea to use deep learning with the help of not only transfer learning, but also a data augmentation.

[8] **dictionary typescript** has keys and values



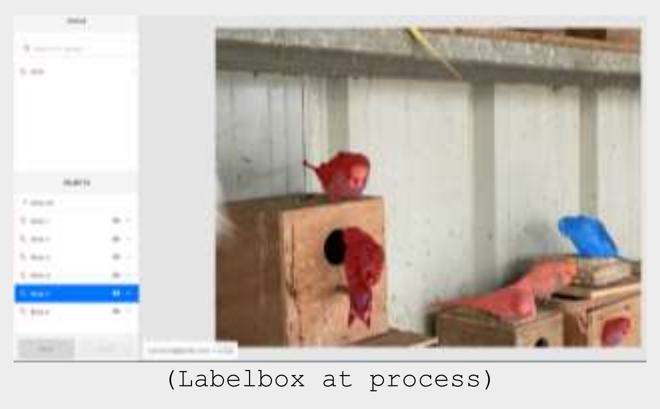
(b)

(c)

(d)

(a)

Figure 3.1: Parrot's images with different complicated backgrounds: (a) a small insulating tape that can be mistaken as a bird. (b) bird houses (c) the cage stripes (d) the back of the parrot (no face) and parrots crowded and overlapping at one point.



(Labelbox at process)

Figure 3.2: Labelbox tool output; Each object is highlighted in blue (there are 27 of those). All the masks together are in red.

The idea of Transfer learning shatters the common misconception that you need **a lot** of a data if you want to train/use CNNs. Conventional machine/deep learning algorithms, so far, have been traditionally designed to work in isolation. i.e. the algorithms were trained to solve specific tasks. But the transfer learning changed the game. This approach was able to overcome the isolated learning paradigm and utilize the knowledge that was acquired for one task - to solve related ones. In more detail, the transfer learning enabled to take the pre-trained weights of an already trained model (one that has been trained on millions of images belonging to 1000's of classes, on several high power GPU's for several days) and use these already learned features to predict new classes.

The basic network that a transfer learning is made of is called ResNet-50. It is a popular model for the ImageNet image classification (AlexNet, VGG, GoogLeNet). It is a 50-layer deep neural network architecture based on residual connections, which are connections that add modifications with each layer, rather than completely changing the signal.

Let us present a way how to create the network and to import a pre-trained ResNet-50 model. The first step is freezing all the ResNet-50's convolutional layers as shown in **Figure 3.3**, and training only the last two fully connected (dense)ones. Since the current classification task is only from two classes (compared to 1000 classes of ImageNet), the last layer needs to be adjusted by the next steps:

(a) loading pre-trained network, cutting off **its head** and freezing its weights
(b) adding custom dense layers
(c) setting the optimizer and loss function[9]

(The loading of the ResNet-50 in PyTorch takes no effort.) In the Algorithm we can see it on section [7].

As for data loading - normally, the images can not all be loaded at once, as doing so would be too much for the memory to handle. By using the GPU's performance boost, it allows that a few images at once can be processed. Therefore, the solution is to load images in *batches* (e.g. 32 images at once) using data generators[10]. Each pass through the whole dataset is called an epoch. Let us notice that the data generators are also used for preprocessing: resizing and normalizing the images to make them as ResNet-50 accepts them (224 x 224 px, with scaled color channels). And last but not least, data generators are used to randomly perturb images on the fly, or in other words such a change is called Data augmentation.

Recent advances in deep learning models have been largely attributed to the quantity and diversity of data gathered in recent years. Data augmentation is the strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.

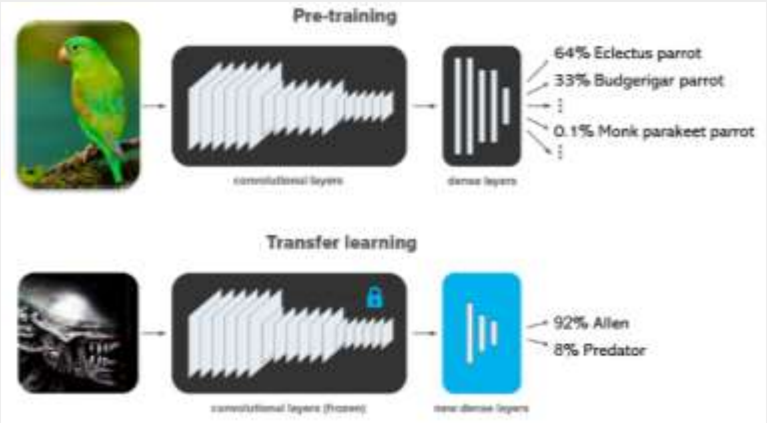
Now, let us begin to understand the main algorithm. Let us denote that there are 11 code sections in the algorithm, where: sections [1]-[3] are mainly responsible for installing the shell and the mounting the code with the drive. Section [4] displays an example of a classical parrot image from the data. Section [5] reads the JSON file and, as we have mentioned before, uses the keys and the values of the dictionary to get the data for the masks of the birds. Let us notice that for our convenience we reduced the image size (otherwise the code runs for too long and does not necessarily get better results). On section [6] we define a class for the parrot's dataset and calculate the bounding-box coordinates for each mask. The output shows all possible masks and prints the number of objects.

Now, let us go back to the article in order to understand the next code sections. Let us remark that the Instance segmentation is challenging because it requires the correct detection of all objects in an image while also precisely segmenting each instance. The article shows that with a method called Mask R-CNN, we can extend Faster R-CNN by adding a branch for predicting segmentation masks on each RoI[11], in parallel with the existing branch for classification and bounding box regression as shown in in **Figure 3.4**. (a)

[09] **Loss function** is used to optimize the parameter values in NN models. It maps a set of the values for the net onto a scalar value that indicates how well those parameters accomplish the task that the net was intended in doing.

[10] **Test generation** is the process of creating a set of test data/cases for testing the adequacy of new/revised software applications.

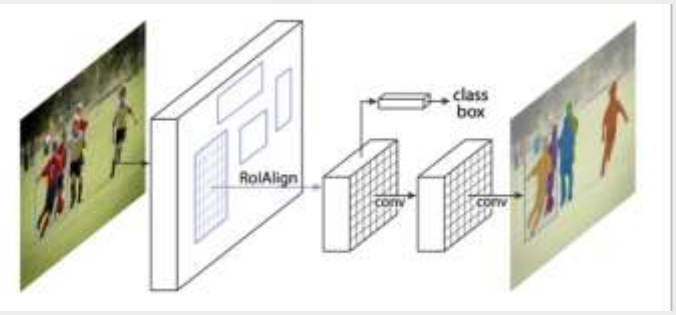
[11] **The Region of interest** (RoI) are samples within a data set identified for a particular purpose.



(Pre training VS Transfer learning)

Figure 3.3: The difference between the outputs of a Pre training and a transfer learning) where the Convolutional network is frozen)

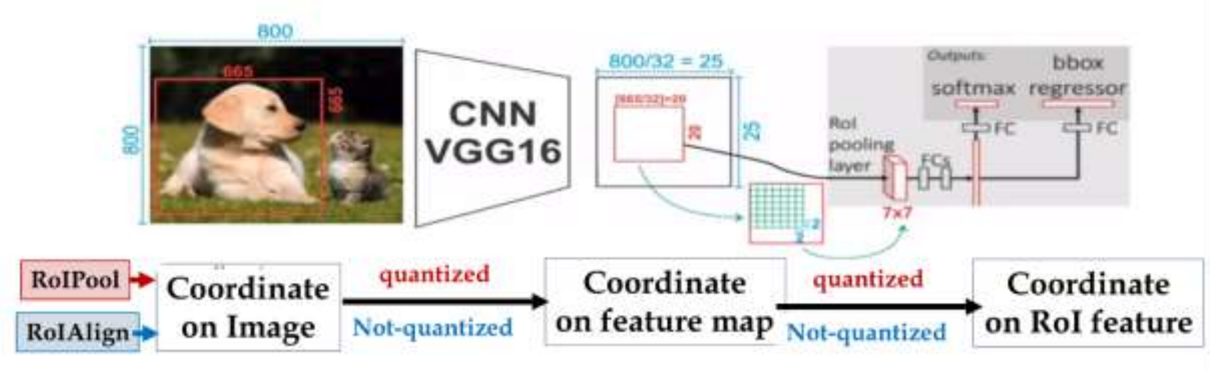
The original image is taken from an internet article named: *Keras vs PyTorch: how to distinguish Aliens vs Predators with transfer learning*



(a)

Figure 3.4: (a) The extension of Faster R-CNN by the addition of a branch for predicting segmentation masks on each RoI. (b) RoIPooling vs RoIAlign layer - when the benefit of RoIAlign layer is that it removes the harsh quantization of RoIPool, properly aligning the extracted features with the input. (meaning no more misalignments)

taken from the article "Mask R-CNN"



(b)

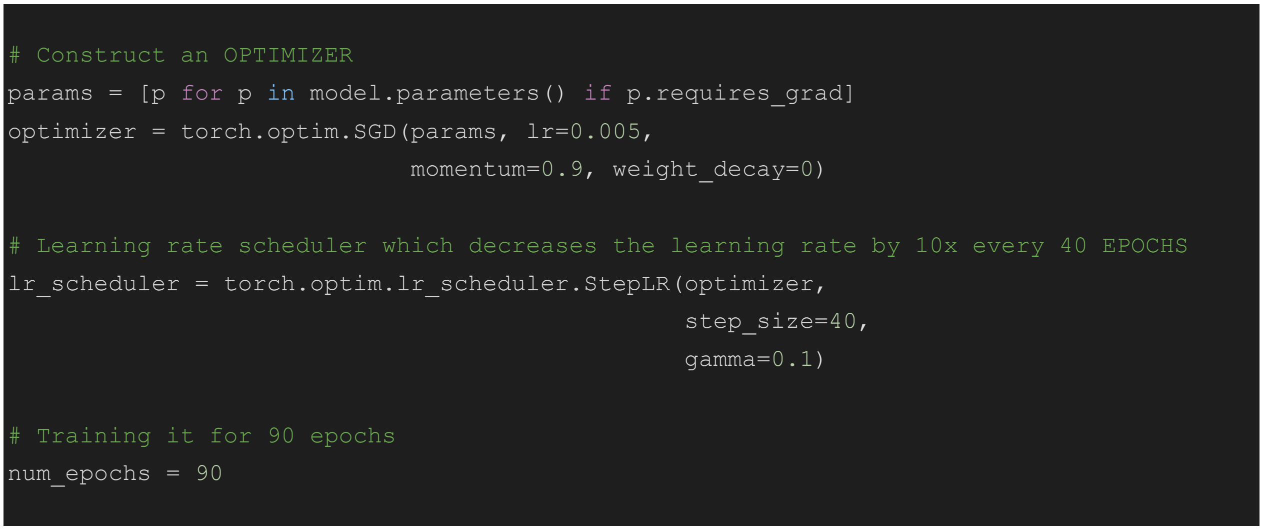
Let us summarize that - in principle, Mask R-CNN is an intuitive extension of Faster R-CNN yet constructing the mask branch properly is critical for the good results. It adds only a small computation overhead and by that it enables a fast system and rapid experimentation. In addition, another small change that was made with Mask R-CNN – let us say that they "fixed" the pixel-to-pixel alignment between network inputs and outputs by using the RoIAlign layer as shown in **Figure 3.4**. (b) .
This change that seems kind of minor actually made a large impact: it improved mask accuracy by relatively 10% to 50%, showing bigger gains under stricter localization metrics.

Furthermore, Mask R-CNN is conceptually simple, like the Faster R-CNN it has two outputs for each candidate object, a class label and a bounding-box offset; and in addition, is added a third branch that outputs the object mask. So, let us conclude that this idea might be intuitive.

Another thing that the article found essential was to decouple *mask* and *class prediction* i.e. to predict a binary mask for each class independently, without competition among classes, and rely on the network's RoI classification branch to predict the category.(In contrast, FCNs usually perform per-pixel multi-class categorization, which couples segmentation and classification, and based on our experiments works poorly for instance segmentation.)

And as mentioned before, on section [7] we indeed import a pre-trained ResNet-50 model. But, there as no use at of the weight freeze. (We my use it in the future for getting even better results). Then, on section [8] we use the dataset and define the data augmentation as explained before and then split the dataset into train set and test set (the last one). Later at the analysis the different choices that we can make at the data set will be more understandable.

Finally, Section [9] instantiates the model and the optimizer and trains our network (on GPU) with the learning rate scheduler which decreases the learning rate by 10x every 40 epochs when:



```
# Construct an OPTIMIZER
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005,
                            momentum=0.9, weight_decay=0)

# Learning rate scheduler which decreases the learning rate by 10x every 40 EPOCHS
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                               step_size=40,
                                               gamma=0.1)

# Training it for 90 epochs
num_epochs = 90
```

The most important part from the run of the training is the Loss calculation that we will mention in the analysis part.

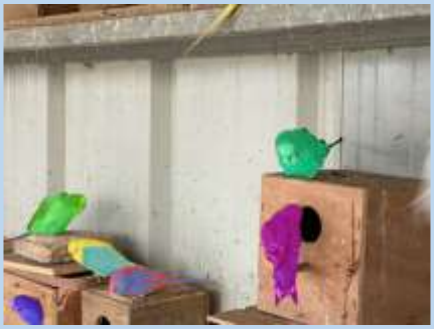
The last sections [10]-[11] show as the interpretation of our results, together with trained images and a test.

4 Results and Analysis

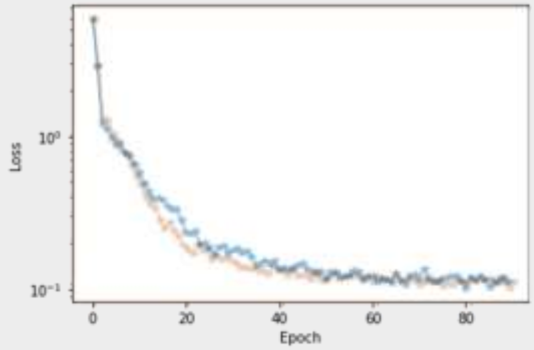
On the previous section we have explained the form of the algorithm; So now we can discuss the finale results.

First, lets go back to our training network and analyze our results of the loss depending on Epochs:
Parameters: **lr** = 0.005, **momentum** = 0.9, **weight_decay** = 0, **step_size** = 40, **gamma** = 0.1

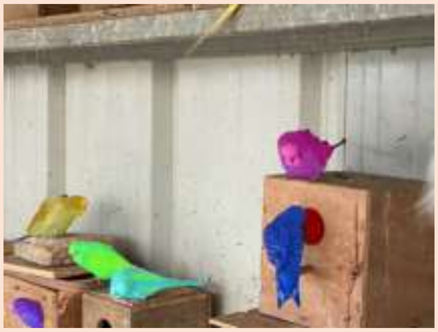
Data containing of 3 Images all together (file2.JSON). We got the next results:



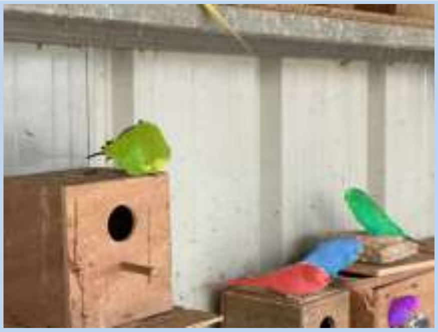
train_Img[1]



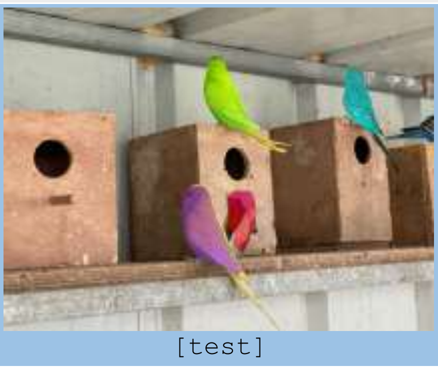
[Blue: 2 trains] [Orange: 1 train]



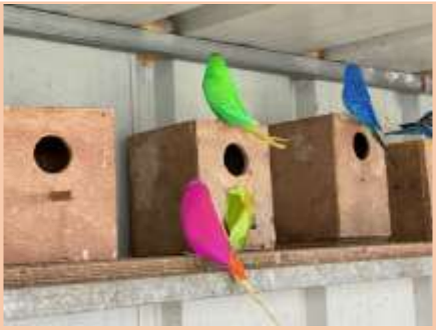
train_Img[1]



train_Img[2]



[test]

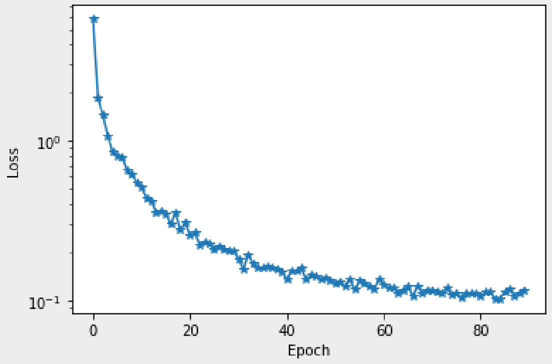


[test]

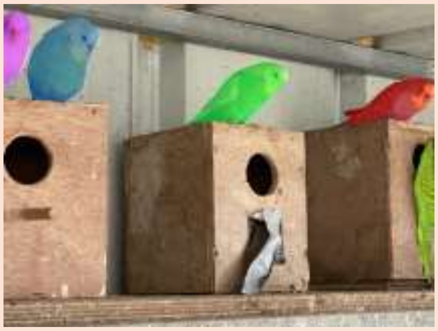
Conclusions: It is clear that the **train** masks should be precise, exactly like we see at the two left images. Now, let us notice that on the case of two training images – they are similar. Another important detail is that the parrots are mostly in a difficult position, i.e. inside the hole of the birdhouse, overlapping, the body is not full, and we cannot always see the tail.

In a very unexpected way, our network is actually giving us good test results and a low loss that goes to zero at the graph. The only problems were: [1] the network could not recognize the tails at the tests [2] In both cases (1 training at orange or 2 trainings at blue) it recognized by mistake a piece of fabric as a bird. When the reason is probably from the training; since we have trained our network that a bird can come out of the bird house.

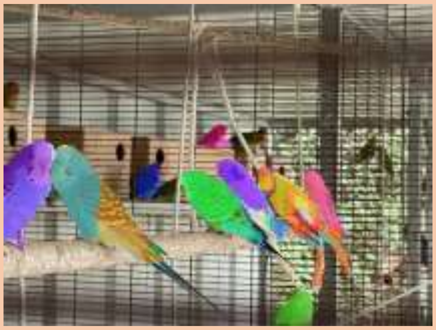
Now, for the same Parameters, but with file1.JSON. We got the next results:



[Orange: 1 train]



train_Img[1]



[test]

Conclusions: Again, the results look amazing by considering the fact that there was only 1 training Image

Let us denote that the parameters from the tests were chosen after a lot of iterations that I have tried.

5 Summary and Conclusions

The main purpose of the project was to find and check a new way of detection by using an artificial intelligence. The results came out surprisingly good, despite the fact that we had a small amount of data to train with.

One of the goals was to check if the neural network is able to detect the objects in spite stuff like: loaded backgrounds, cage bars at the back, objects covering each other, objects in bad self-positions i.e. you can see only a part of the body. And again, the results were pretty good. Our network actually was able to overcome all of the above. (see the image results)

I believe that the next step at the project should be the parrot recognition – and it probably will also work not in a bad way.

6 Sources

- *Stanford course CS-231 .2017. All lectures*

- Article (01/2018) “Mask R-CNN”
  published by *Kaiming He, Georgia Gkioxari, Piotr Doll´ar and Ross Girshick*