Image Classification, Adversarial Attack and Interpretability using Neural Networks Big Data For Official Statistics A.Y. 2020/2021

Luca Scofano
Matricola: 1762509

ABSTRACT

Image classification is used by computers to analyse an image and identify the 'class' the image falls under. (Or a probability of the image being part of a 'class'.) A class is essentially a label, for instance, 'car', 'animal', 'building' and so on. Early image classification relied on raw pixel data. This meant that computers would break down images into individual pixels. The problem is that two pictures of the same thing can look very different. They can have different backgrounds, angles, poses. We need a model that is based on translation invariance, this means that the object should be recognized independently from all of the factors I've listed previously. This made it quite the challenge for computers to correctly 'see' and categorise images.

Deep learning is a type of machine learning; a subset of artificial intelligence (AI) that allows machines to learn from data. Deep learning involves the use of computer systems known as neural networks.

In neural networks, the input filters are passed to hidden layers of nodes. These nodes each process the input and communicate their results to the next layer of nodes. This repeats until it reaches an output layer, and the machine provides its answer(depending on the task).

For this project I decided to use two different models, a classical ***CNN*** *and* ***YOLO*** *(with DarkNet 53 infrastructure).*

The project then divides into two substask, adversarial attack and interpretability.

The first subtask consists in perturbing the initial dataset in order to obtain less accurate results.

The second subtask instead in based on interpreting why and how our models are able to classify and predict the right labels.

INTRODUCTION

Image classification. The ability to use artificial intelligence to identify objects and classify images has been one of the fastest growing areas in AI since Alex Krizhevsky et al. developed the AlexNet convolutional neural network (CNN) that won the 2012 ImageNet challenge. The AlexNet network achieved state-of-the-art performance labeling images from the 14,197,122 ImageNet dataset. Since then, the quantity and diversity of CNN architectures has expanded dramatically across deep CNN architectures to inception and residual CNN networks. Today, CNN's are being used for image classification, medical imaging, facial recognition and identification, sorting your mail, robot vision and coming soon, self-driving cars.

Deep learning excels in recognizing objects in images as it's implemented using 3 or more layers of artificial neural networks where each layer is responsible for extracting one or more feature of the image (more on that later).

To do so, we first need to teach the computer how a cat, a dog, a bird, etc. look like before it being able to recognize a new object. The more cats the computer sees, the better it gets in recognizing cats. This is known as supervised learning. We can carry this task by labeling the images, the computer will start recognizing patterns present in cat pictures that are absent from other ones and will start building its own cognition.

How do we represent images for a computer to understand?
Colors could be represented as RGB values (a combination of red, green and blue ranging from 0 to 255). Computers could then extract the RGB value of each pixel and put the result in an array for interpretation.

Computers are able to perform computations on numbers and is unable to interpret images in the way that we do. We have to somehow convert the images to numbers for the computer to understand.

One of the most popular techniques used in improving the accuracy of image classification is Convolutional Neural Networks (CNNs for short).

Instead of feeding the entire image as an array of numbers, the image is broken up into a number of tiles, the machine then tries to predict what each tile is. Finally, the computer tries to predict what's in the picture based on the prediction of all the tiles. This allows the computer to parallelize the operations and detect the object regardless of where it is located in the image.

As I previously noted, the first model I used is a CNN and the second one is based on YOLO. Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in a image, Yolo architecture is more like FCNN (fully convolutional neural network) and passes the image (nxn) once through the FCNN and output is (mxm) prediction.

Adversarial attack. An adversarial attack consists of subtly modifying an original image in such a way that the changes are almost undetectable to the human eye. The modified image is called an adversarial image, and when submitted to a classifier is misclassified, while the original one is correctly classified. The real-life applications of such attacks can be very serious –for instance, one could modify a traffic sign to be misinterpreted by an autonomous vehicle, and cause an accident. Another example is the potential risk of inappropriate or illegal content being modified so that it;s undetectable by the content moderation algorithms used in popular websites or by police web crawlers.

Interpretability. First, interpretability in machine learning is useful because it can aid in trust. As humans, we may be reluctant to rely on machine learning models for certain critical tasks, e.g., medical diagnosis, unless we know "how they work". Approaches to interpretability that focus on transparency could help mitigate some of these fears.

Perhaps most interestingly, contestability. As we delegate more decision-making to ML models, it becomes important for the people to appeal these decisions. Black-box models provide no such recourse because they don't decompose the decision into anything contestable. This lack of contestability has already led to significant criticism of proprietary recidivism predictors like COMPAS. Approaches to interpretability, which focus on decomposing the model into sub-models or illustrate a chain of reasoning, could help with such appeals.

1. DATASET

Dataset Description. The data set is a collection of images of alphabets from the American Sign Language, separated in 29 folders which represent the various classes.

The training data set contains 87,000 images which are 200x200 pixels. There are 29 classes, of which 26 are for the letters A-Z and 3 classes for SPACE, DELETE and NOTHING. These 3 classes are very helpful in real-time applications, and classification. The test data set contains a mere 29 images, to encourage the use of real-world test images.

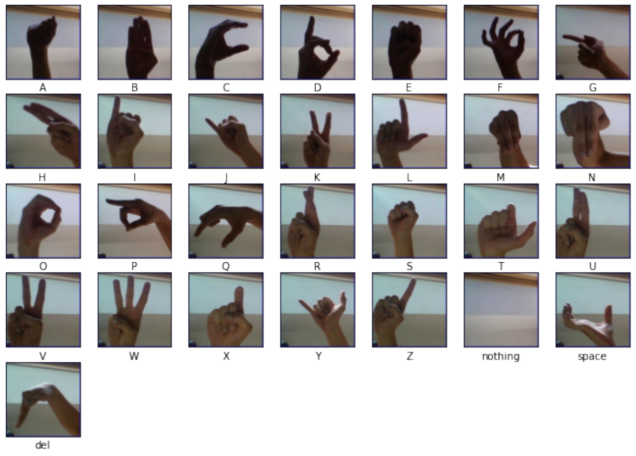

Figure 1: Schematic diagram of underwater optical imaging

Dataset pre-processing. I wanted to perform some data augmentation in order to reduce overfitting and increase generalization. This process modifies the input images while doing different operations. Initially a random rotation is used, afterwards width and height shift is also performed and at the end I applied a random crop to the entire dataset. Is important to notice that this whole process is done only on the *training dataset.*

Batch size. Smaller batch sizes have been empirically shown to have faster convergence to "good" solutions. The downside of using a smaller batch size is that the model is not guaranteed to converge to the global optima. It will bounce around the global optima, this is why I used **128**.

2. MODELS

CNN and DarkNet. I picked two different models in order to compare the performance in terms of accuracy (minimizing the loss function). Since the models use different architectures the accuracy tends to be different, favoring the more complex model, DarkNet, that uses **40m** parameters more during the training phase. The other reason was based on the adversarial attack part, I was curious to see whether both models would under-perform when applying imperceptible noise to the original image.

2.1 DarkNet

The idea behind YOLO is this: There are no classification/detection modules that need to sync with each other and no recurring region proposal loops as in previous 2-stage detectors (like RCNN). It's basically convolutions all the way down (with the occasional maxpool layer). Instead of cropping out areas with high probability for an object and feeding them to a network that finds boxes, a single monolithic network needs to take care of feature extraction, box regression and classification. While previous models had two output layers — one for the class probability distribution and one for box predictions, here a single output layer contains everything in different features.

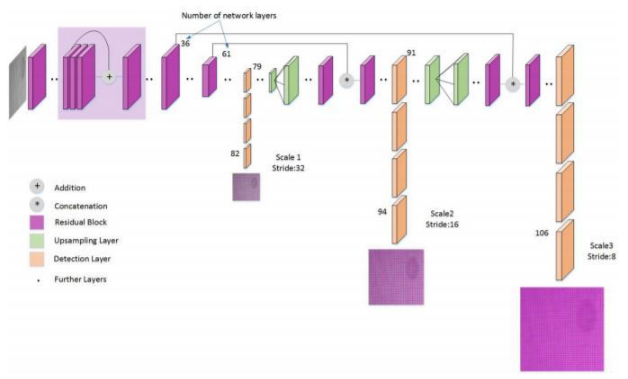

Figure 2: Schematic diagram of underwater optical imaging

Architecture. Inspired by ResNet and FPN (Feature-Pyramid Network) architectures, YOLO-V3 feature extractor, called Darknet-53 (it has 52 convolutions) contains skip connections (like ResNet) and 3 prediction heads (like FPN) — each processing the image at a different spatial compression.

A Feature-Pyramid is a topology developed in 2017 by FAIR (Facebook A.I. Research) in which the feature map gradually decreases in spatial dimension (as is the usual case), but later the feature map expands again and is concatenated with previous feature maps with corresponding sizes. This procedure is repeated, and each concatenated feature map is fed to a separate detection head.

Feature Pyramid Network. FPN topology allows the YOLO-V3 to learn objects at different sizes: The 19x19 detection block has a broader context and a poorer resolution compared with the other detection blocks, so it specializes in detecting large objects, whereas the 76x76 block specializes in detecting small objects. Each of the detection heads has a separate set of anchor scales.

Unlike SSD (Single-Shot Detector) architectures, in which the 38x38 and 76x76 blocks would receive only the high-resolution, partly processed activations from the middle of the feature extractor (the top 2 arrows in the diagram), in FPN architecture those features are concatenated with the low-resolution, fully processed features at the end of the feature extractor.

Results

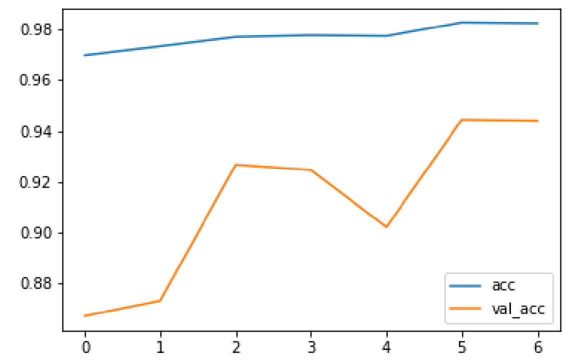

Figure 3: Schematic diagram of underwater optical imaging

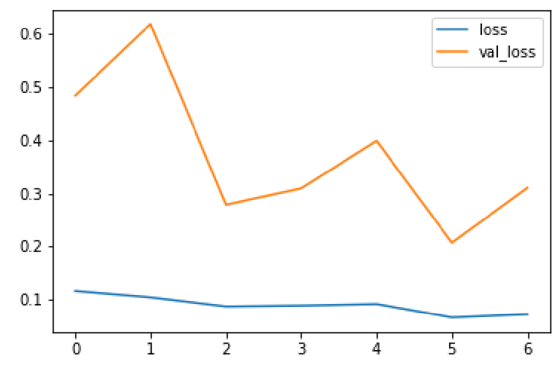

Figure 4: Schematic diagram of underwater optical imaging

Results.

2.2 CNN

This model is a simple CNN with *3* main blocks and a couple of regularization layers. At the end, being a classification task, I added two dense layers in order to flatten the inputs and at the end I wrapped it in a *softmax* activation function.

Main Block. The three main blocks are formed by a convolutional layer that doubles in time each time **16, 32, 64** in order to capture more information on the features. The second element of this main block is a **MaxPool** layer, it's used in order to downsample and at the same time increase the receptive field.

Normalization Layers

Batch normalization. Batch normalization (also known as batch norm) is a method used to make artificial neural networks faster and more stable through normalization of the input layer by re-centering and re-scaling. The discussion on the impact of this method is still on, most times is said that we use batch normalization in order to smooth out the outputs and ease the computation of the gradients through backpropagation.

Dropout. The *Dropout* layer is applied after flattening the outputs, this procedure discards some of the neurons with probability *p*. It's used to prevent overfitting.

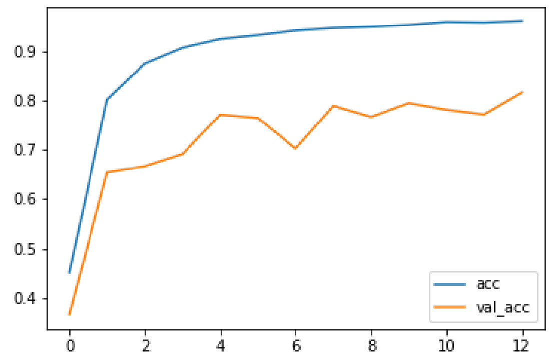

Figure 5: Schematic diagram of underwater optical imaging

2.3 Training

Training. Following there are some of the parameters I used for the training part.

Number of epochs

For the DarkNet I only needed 6 epoch to achieve a really high accuracy on the validation set $\approx 90 percent$. As for the simple CNN I initially used 20 epochs, but the early stopping kicked in at epoch number *30*, since it wasn't really increasing in accuracy.

Early Stopping

Stop training when a monitored metric has stopped improving. Assuming the goal of a training is to minimize the loss. With this, the metric to be monitored would be 'loss', and mode would be 'min', **min delta** was Minimum change in the monitored quantity to qualify as an improvement, i.e. an absolute change of less than min

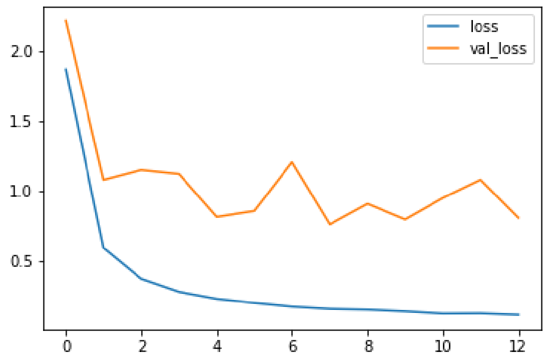

Figure 6: Schematic diagram of underwater optical imaging

delta, will count as no improvement. And the **patience**, Number of epochs with no improvement after which training will be stopped, is equal to **5**.

Learning rate

When training deep neural networks, it is often useful to reduce learning rate as the training progresses. This can be done by using pre-defined learning rate schedules or adaptive learning rate methods.

3. RESULT ANALYSIS

The figure shows one of the predictions our **DarkNet** did, as we can see the actual label and the prediction are the same in this case.

We now consider an entire batch (*128*) of images and make predictions on these. The heatmap shows us that all of the labels where actually labelled correctly, there was no misclassification for this batch.

4. ADVERSARIAL ATTACK

In white box attacks the attacker has access to the model's parameters, while in black box attacks, the attacker has no access to these parameters, i.e., it uses a different model or no model at all to generate adversarial images with the hope that these will transfer to the target model. As for this project the approach I'm going to use is the white-box one.

The aim of non-targeted attacks is to enforce the model to misclassify the adversarial image, while in the targeted attacks the attacker pretends to get the image classified as a specific target class, which is different from the true class.

For this project I used the Fast Gradient Sign Method. This method computes an adversarial image by adding a pixel-wide perturbation of magnitude in the direction of the gradient. This perturbation is computed with a single step, thus is very efficient in terms of computation time.

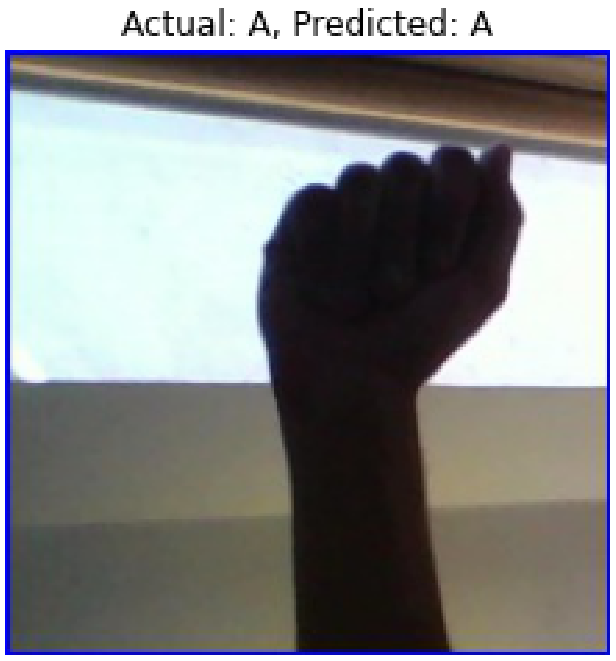

Figure 7: Schematic diagram of underwater optical imaging

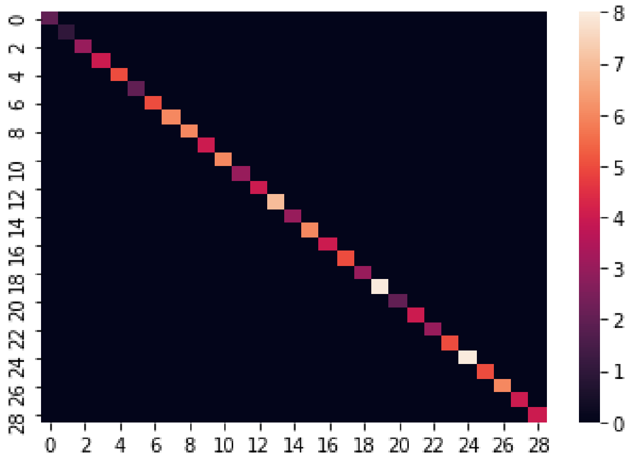

Figure 8: Schematic diagram of underwater optical imaging

Fast Gradient Sign Method

Therefore, the authors focus on producing an adversarial example x' = x + such that x' is classified incorrectly by the neural network. In order to make x' and x produce different outputs, should be bigger than the precision of the features. For example, if each pixel of an image is represented by 8 bits, any information below 1/255 of the dynamic range is discarded. In order to achieve this, the authors use the following scheme for computing :

$$\eta = \epsilon sign(\Delta_x J(\theta, x, y)) \qquad (1)$$

Here, J is the cost function used to train the neural network, represents the parameter of a model, x is the input to the model and y is the target associated with x (for machine learning tasks that have targets). Here, decides the size of the perturbation and sign of every element of the perturbation vector(or matrix/tensor) is decided by the sign of the input gradient at the element. This solution is motivated by linearizing the cost function and solving for the perturbation that maximizes the cost subject to an L constraint. This linear perturbation technique method reliably causes a wide variety of models to misclassify their input. This method does not require an iterative procedure to compute adversarial examples, and thus is much faster than other considered methods.

These following figures represent the noise that I've applied to two different images, in this case $\epsilon = 0.3$

Figure 9: Noise applied to test images

What happens to the accuracy with different values of epsilon?

Following there are two plots that show how an increasing value of ϵ will impact the model

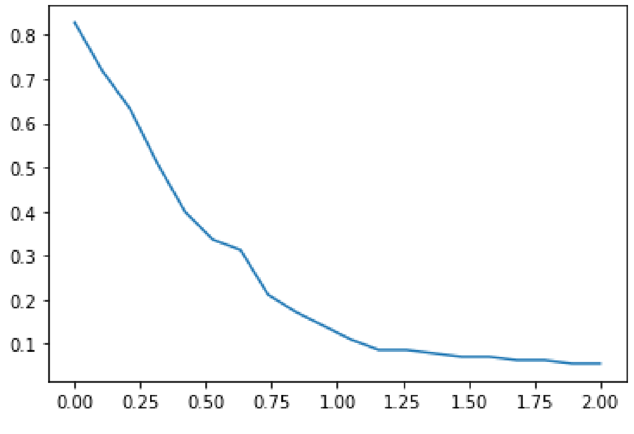

Figure 10: Adversarial attack on CNN

In this case the curve drops quite swiftly, when $\epsilon = 1$ the accuracy is equal to 0.1.

The more complex model tries to keep up with the noisy input, but we also reach an accuracy that is equal to 0.2 when $\epsilon = 1.75$

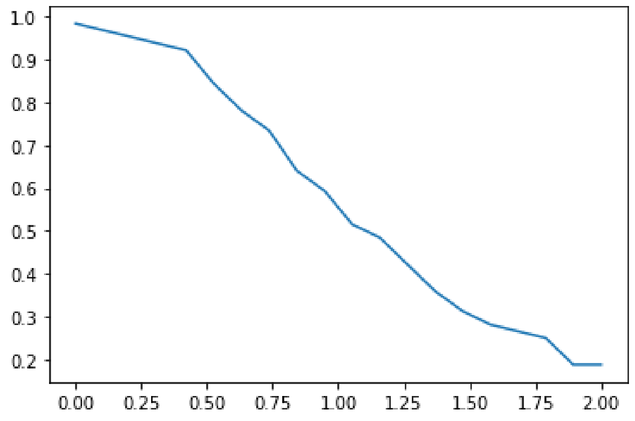

Figure 11: Adversarial attack on DarkNet

5. INTERPRETABILITY

The basic idea is to understand why a machine learning model (deep neural network) predicts that an instance (image) belongs to a certain class. LIME creates explanations by generating a new dataset of random perturbations (with their respective predictions) around the instance being explained and then fitting a weighted local surrogate model. This local model is usually a simpler model with intrinsic interpretability such as a linear regression model.

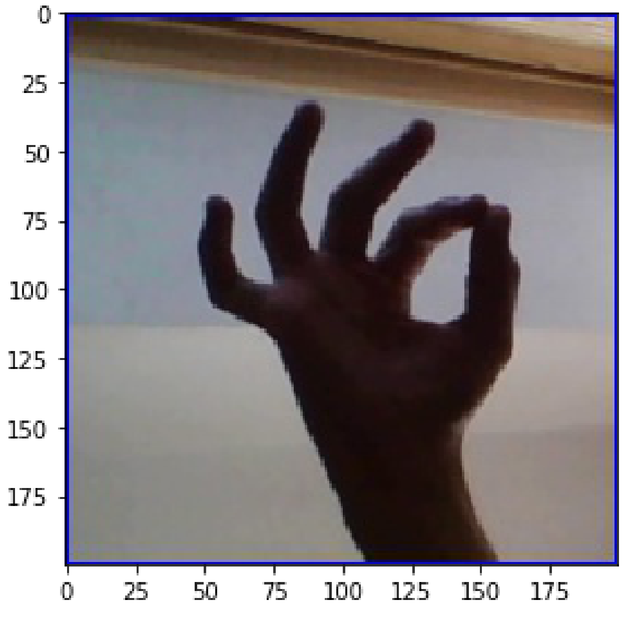

Figure 12: Schematic diagram of underwater optical imaging

Step 1. Create perturbations For the case of image explanations, perturbations will be generated by turning on and off some of the superpixels in the image. Superpixels are generated using the quickshift segmentation algorithm. It can be noted that for the given image, 68 superpixels were generated. The generated superpixels are shown in the image below.

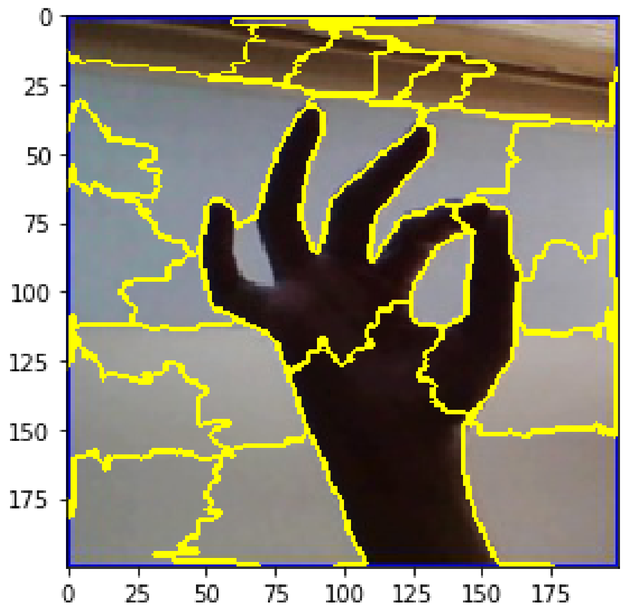

Figure 13: Schematic diagram of underwater optical imaging

For the case of images, LIME generates perturbations by turning on and off some of the super-pixels in the image. The algorithm uses quick-shift segmentation algorithm to compute the super-pixels in the image. In addition, it generates an array of 150 perturbations where each perturbation is a vector with zeros and ones that represent whether the super-pixel is on or off.

In this example, 150 perturbations were used. However, for real life applications, a larger number of perturbations will produce more reliable explanations. Random zeros and ones are generated and shaped as a matrix with perturbations as rows and superpixels as columns. An example of a perturbation (the first one) is show below.

The following script uses the *DarkNet* to predict the class of each of the perturbed images. The shape of the predictions is (150,29) which means that for each of the 150 images, we get the probability of belonging to the 29 classes. From these 29 classes we will use only the Labrador class in further steps since it is the prediction we want to explain.

Now we have everything to fit a linear model using the perturbations as input featuresX and the predictions as output y . However, before we fit a linear model, LIME needs to give more weight (importance) to images that are closer to the image being explained.

We use a distance metric to evaluate how far is each perturbation from the original image. The original image is just a perturbation with all the super-pixels active (all elements in one). Given that the perturbations are multidimensional vectors, the cosine distance is a metric that can be used for this purpose. After the cosine distance has been computed, a kernel function is used to translate such distance to a value between zero and one (a weight). At the end of this process we have a weight (importance) for each perturbation in the dataset.

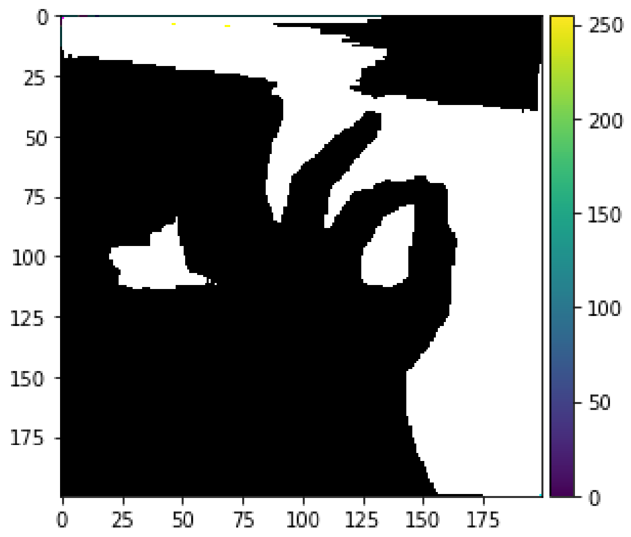

Figure 14: Schematic diagram of underwater optical imaging

A weighed linear regression model is fitted using data from the previous steps (perturbations, predictions and weights). Given that the class that we want to explain is **the letter F**, when fitting the linear model we take from the predictions vector only the column corresponding to the top predicted class. Each coefficients in the linear model corresponds to one superpixel in the segmented image. These coefficients represent how important is each superpixel for the prediction of labrador.

Now we just need to sort the coefficients to figure out which are the supperpixels that have larger coefficients (magnitude) for the prediction of labradors. The identifiers of these top features or superpixels are shown below. Even though here we use the magnitude of the coefficients to determine the most important features, other alternatives such as forward or backward elimination can be used for feature importance selection.

Let's show the most important superpixels defined in the previous step in an image after covering up less relevant superpixels.

This is the final step where we obtain the area of the image that produced the prediction of **the letter F**.

Conclusion and Discussion

Image classification might be considered as a basic task for machine learning but as we saw it can still hide some difficulties, especially if we modify the input (*adversarial attack*). And also it's one of the foundation for more advanced tasks. Additionally we also saw how to *interpret* the output of neural network, that is usually taken for granted. This project could be continued more thoroughly by adding different adversarial attacks and by using more recent networks. Another interesting thing to do would be to use the dataset we picked to perform **image to audio** translation.