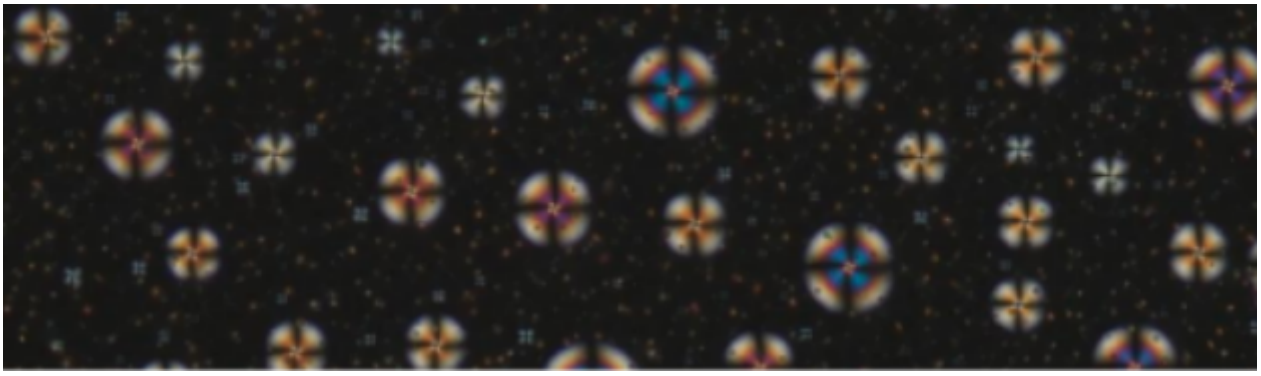
TÉCNICO LISBOA



Video classification of Odors with Convolutional Neural Networks

José Maria Lopes Rafael Ledesma Frazão

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Maria Margarida Campos da Silveira
Susana Isabel Conde Jesus Palma

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Maria Margarida Campos da Silveira
Member of the Committee: Prof. Ana Catarina Fidalgo Barata

February 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First and foremost I would like to express my profound gratitude to Professor Margarida Silveira. Throughout this last year, Professor Margarida was tireless guiding me in my work by sharing her knowledge and experience during our meetings week in and week out.

I would like also to thank the Biomolecular Engineering Group, UCIBIO, REQUIMTE, FCT/UNL for having generated the datasets that were key in the development of this Thesis. A special word goes to Susana Palma and Cecilia Roque that, being part of this group, insisted on following closely my progress as well as being fully available to answer all the questions I had regarding the data and the project in general.

To my family and friends, a very special thanks for all the support and helping me push through the difficult moments .

Abstract

Trying to mimic natural olfactory systems, by achieving the perfect combination between chemical sensing and artificial intelligence, has been an important task among the science community. Over the last decade, e-nose technologies have undergone important developments on their way to be implemented in real life situations such as food monitoring, air quality monitoring, etc. In order for that to be possible, e-noses need to possess a very efficient and highly accurate pattern recognition component. In this work we will investigate how CNN based algorithms can be implemented in these devices in order to help them achieve good results. With deep learning being a hot topic nowadays, and with the advance in computational hardware, plenty of very good architectures have emerged that can achieve outstanding results in pattern recognition tasks. Different architectures are approached, from simple 3 dimension CNNs to an object detection algorithm (YOLOs) and recurrent algorithms capable of handling sequencial data (LSTMs). Since this work focuses solely on the pattern recognition side of e-noses, it will work on top of an already produced dataset. At first, an hybrid gel was exposed to 11 different volatile organic compounds and its reaction to them was recorded in the format of image sequences. The objective is to find the algorithm capable of classifying the image sequences corresponding to the the 11 VOC classes with the highest possible score. On the second task, this gel was exposed to the same compound but with different concentrations in the exposed gas. Thus, the second goal is to find the model best capable of predicting the actual exposed concentration, also based on the recorded image sequences.

Keywords

E-nose, CNN, LSTM, YOLO, VOC, Object Detection, Concentrations

Resumo

Tentar imitar sistemas olfativos naturais, através da combinação perfeita entre sensoriamento quimico e inteligencia artificial, tem sido uma tarefa bastante discutida dentro da comunidade cientifica. Na última década, tecnologias de narizes eletrónicos sofreram avanços importantes para poderem ser utilizados em situações do quotidiano como monitoração da qualidade de comida, da qualidade do ar, etc. Para que isto seja possivel, estes narizes eletrónicos têm de possuir uma componente de *pattern recognition* muito eficiente e precisa. Neste trabalho, será investigado como algoritmos baseasdos em CNNs podem ser implementados nestes dispositivos de forma a ajudá-los a atingir resultados fidedignos. Sendo *deep learning* um tema bastante em voga nos tempos atuais, e com o avanço de *hardware* computacional, muitas arquiteturas capazes de alcançar resultados excepcionais em tarefas de *pattern recognition* têm vindo a aparecer. Diferentes arquiteturas foram abordadas, desde CNNs a 3 dimensões a algoritmos de deteção de objetos (YOLO) e algoritmos recorrentes (LSTMs). Uma vez que este trabalho se foca apenas na parte de *pattern recognition* de um nariz eletrónico, vai se basear num *dataset* já produzido. Primeiramente, um gel hibrido foi exposto a 11 compostos orgânicos voláteis e a sua reação for gravada no formato de sequencia de imagens. O objetivo é encontrar um método capaz de classificar estas sequências de forma fiável. Numa segunda tarefa, o gel foi exposto ao mesmo composto mas com concentrações diferentes. O segundo objetivo é então encontrar um modelo capaz de prever a concentração real exposta ao gel, também baseado nas sequências de imagens gravadas.

Palavras Chave

Nariz Elétronico, CNN, LSTM, YOLO, VOC, Deteção de Objetos, Concentrações

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

BPTT Backpropagation Through Time

BiLSTM Bidirectional LSTM

CNN Convolutional Neural Network

CNN2D 2 Dimensional Convolutional Neural Network

CNN3D 3 Dimensional Convolutional Neural Network

FCL Fully Conneced Layer

GPU Graphical Processing Unit

GRU Gated Recurrent Unit

IoU Intersection over Union

LDA Linear Discriminant Analysis

LSTM Long Short-Term Memory

MAE Mean Absolute Error

mAP Mean Average Precision

ML Machine Learning

MLP Multi-layer Perceptron

MSE Mean Squared Error

NMS Non-Maximum Suppression

OvO One vs One

OvR One vs Rest

PCA Principal Component Analysis

R-CNN Region Based Convolutional Neural Network

RNN Recurrent Neural Network

RPN Region Proposal Network

RoI Region of Interest

SVM Support Vector Machine

TBPTT Truncated Backpropagation Through Time

VOC Volatile Organic Compound

YOLO You Only Look Once

Chapter 1

Introduction

1.1 Motivation

The olfactory sense is considered to be the oldest out of all senses. This is a sense that even bacteria have in order to react and adapt to the chemicals around it. Some of these bacteria were already around ages before creatures with the ability of hearing, touching or seeing started to exist [1]. On a more actual context, lots of different animals, not so much humans, still put too much trust on their smelling sense since it is used to find food, avoid danger, track their mates and their preys, etc.. This sense plays a vital role in their existence and capacity to understand the surrounding environment.

This unique ability of detecting different smells/odors is continuously being a subject of study and plenty of advances have been made in this field. One of the ideas that came out from it was the possible development of some kind of artificial olfactory system able to identify substances in the air. Having the possibility of mimicking this highly complex and powerful system in various situations can be game changing. Electronic based devices are being developed to serve this exact purpose, which are called E-noses. These devices contain an array of sensors that, in contact with Volatile Organic Compounds (VOCs), generate signals that can later be analyzed in order to characterize and identify these same compounds. The odors present in the air are composed by mixtures of these VOCs. E-noses aim to be the perfect combination between chemical sensing, signal processing and pattern recognition inside the brain, in order to achieve high performances in odor recognition tasks.

Several of these devices are already being successfully put into practice. E-noses are already being implemented in several areas such as disease detection [2–4], quality control in laboratories [5], food quality control [6], air quality monitoring [7], security systems [8] and many more. Figure 1.1 shows an E-nose device developed by the German company *Airsense Analytics*. This E-nose has 10 different metal oxides single thick film sensors and can, depending on the application, deliver results from 4 seconds to

some minutes. It can be used in quality control tasks such as checking the rancidity of oils, freshness of food, off odour in packaging materials and many more according to its technical flyer [1].

Figure 1.1: E-nose (PEN3) developed by *Airsense Analytics*

1.2 Objectives

Over the past few decades, the development in the field of Artificial Intelligence (AI) has been astronomical. All the hype around it and perception of its immense potential led loads of people to spend their time and money towards investigation in this field. This, together with the advance in computing hardware, such as Graphical Processing Units (GPUs), enabled fields like Machine Learning (ML) to rise and offer a whole new set of solutions to problems that once seemed unsolvable. The whole idea behind ML is to teach a computer how to perform a certain task without having the need to provide exact instructions every step of the way. Basically ML algorithms learn the underlying relations among data to then make decisions without being programmed to. One of the ways to do that is through the use of Artificial Neural Networks (ANNs) which are based on the biological neural networks that compose animal brains.

These networks later evolved to new sub-fields of ML like Convolutional Neural Networks (CNNs) that managed to tackle a new set of problems with great performances as it will be seen further in the document. CNNs have been proven to be a very successful neural network to process image data. They have shown state-of-the-art results in many image related tasks such as recognition, detection and segmentation. CNNs are a hot topic inside the ML community that is using it for every imaginable task possible. CNNs have shown very good results in problems like Facial Recognition [9], Action Recognition [10], Cancer Detection [11] and many more.

The pattern recognition in a real olfactory system is performed by the brain that takes as inputs the electrical signals generated when VOCs bind to the chemoreceptors in the nose. The brain has a huge

[1] https://airsense.com/sites/default/files/flyer_pen.pdf

pattern recognition capacity as it can easily match stimulus produced by the outside world with data stored in memory. This work focuses on the pattern recognition section of the olfactory system, not really targeting the stimulus acquisition and transducing parts. In an artificial olfactory system, many different methods can work as this pattern recognition element but in this work, CNN based systems will serve this purpose. So, the main objective of this Thesis is to investigate how can these systems be implemented in an E-nose system by making use of their pattern recognition capabilities. CNNs have already been used in E-nose systems to predict odor pleasantness [12] or to identify a specific type of tea [13] for example. However, these works do not use CNNs as feature extractors in image sequences, but in other types of signals. In both these works, the reactions to the odors were captured by Metal-Oxide and/or Quartz Microbalance sensors that record data in only one dimension. Two dimensional CNNs were still used but, in order for that to be possible, several sensor responses were put together into 2D matrices or a response was multiplied by its transpose to again obtain data in two dimensions.

The Biomolecular Engineering Group, UCIBIO, REQUIMTE, FCT/UNL developed sensors (gas-sensitive gel) that generate a distinct optical response according to the VOC they are being exposed to. This optical response can be filmed, and through its pattern variations, in theory, it is possible to identify the exposed VOC (fingerprint). In order to do that, CNN based systems will be applied to these recorded responses (image sequences).

This Thesis can be divided into 2 different parts. The first one consists in trying to predict which VOC the gas-sensitive gel is being exposed to. On the second one, the goal is to predict the concentration of a specific VOC (Acetone) to which the gel is being exposed. For both these experiences, the results will be achieved always resorting to CNN based systems that take as input the recorded reactions of the gel (more specifically, the reaction of the droplets present in this gel). Another goal, besides trying to find the models that work better on solving these tasks, is to check whether the size of these droplets has influence on the results.

1.3 Thesis Outline

This thesis is structured as follows:

Chapter 2 Theory & State of the Art shares some background information and state of the art applications of important models, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) and You Only Look Once (YOLO), that are later on trained and tested in order to achieve the proposed goals.

Chapter 3 Proposed Approach presents the general system architecture used in the experimental part, as well as, some used model architectures and addressed strategies.

Chapter 4 Experiments & Results describes every performed experience, for both proposed tasks,

and presents the associated results.

Chapter 5 Conclusions presents the overview of what was achieved during this work and some suggestions on how to improve this project in the future.

Chapter 2

Theory & State of the Art

2.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network is a special type of Neural Network usually applied to computer vision tasks. CNN based algorithms have been showing great performances in several image related tasks, such as image classification, segmentation or detection, where they are able to produce state-of-the-art results. In order to get the best possible results out of this technology there are many details that need to be established and correctly understood. Details like the network length, kernel size, which activation function to use, which types of layers to use, how to avoid overfitting and much more, are very important details that in the end can make the difference between a successful, high accuracy network and a poor network that cannot learn enough useful information.

Usually, a CNN is formed by 3 main types of layers. These are the Convolutional Layers, the Pooling Layers and the Fully Connected Layers. A typical CNN is normally composed by alternating convolutional and pooling layers, finishing with at least a fully connected one (Figure 2.1).

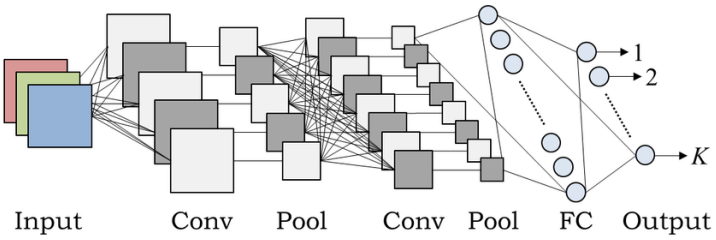


Figure 2.1: Typical CNN architecture (taken from [14])

A convolutional layer is usually composed by several kernels/filters, each one theoretically useful for different analysis of nearby pixels. The fact that the set of weights in a kernel is shared for the whole image is a huge advantage compared to a simple Multi-layer Perceptron (MLP) since the amount of

parameters to train decreases drastically. Besides that, if a certain feature changes its location in the image, it can still be detected given this all round spatial property. The convolution is an operation where the kernel slides horizontally and vertically over the entire image. In each position, the sum of every multiplication of a kernel element by its corresponding element in the image is stored in an output image/feature map (Figure 2.2). The dimensions of this layer's output image will depend on the original image size, on the kernel size, the stride applied on the convolution, padding, etc.. As soon as the convolutions of every kernel are done, a feature map is generated for each one of them.

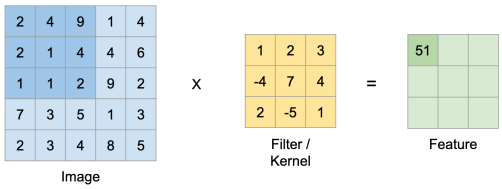


Figure 2.2: Convolution

Normally, every convolutional layer goes hand in hand with an activation function. What this means is that every produced value by the convolutional operation is still subject to a non linear function known as activation function. Without signal activation, the output signal of this layer would merely be a linear function. In this scenario, the neural network would become a simple linear regression model that would not be able to handle complex data and would not perform well enough most of the times. Knowing this, adding non-linearities to the system makes it more robust and capable of tackling more difficult problems. Besides this, it is of the utmost importance for this function to be differentiable since, while applying the backpropagation algorithm to train the network, the gradients of the losses with respect to the weights are computed. Consequently the weights are updated through an optimization technique, usually a Gradient Descent variant. The most popular activation functions nowadays are the Sigmoid, Tanh and ReLu (Figure 2.3). The last one has been showing better performances compared to the others and it is the most used nowadays [15]. It has outgrown the other two by delaying the vanishing gradient problem, which is very much present in the aforementioned. Deeper architectures using this activation function are less prone to suffer from this vanishing gradient problem.

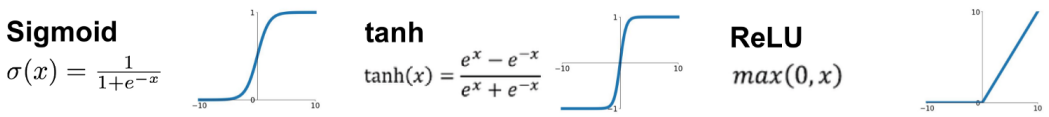


Figure 2.3: Activation Functions: Sigmoid, Tanh, ReLu

The second main type of layers used in CNNs are the Subsampling/Pooling Layers. The main reason for using these types of layers is to reduce the system dimensionality, which consequently reduces the

amount of parameters to train in the process. To do that, it combines the values of a certain region in the feature map into a single value. This operation sums up the information in this region, outputting one single value, according to the used function. This is also done using the sliding window technique. By decreasing the dimensionality of the network, the amount of parameters to train decreases which can help avoid overfitting. Theoretically, any type of function can be used to downsample the input image but in practice almost only the max function (Figure 2.4) is used in these layers since it helps highlighting the most activated features [16].

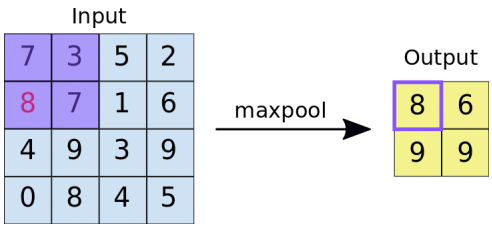


Figure 2.4: Max Pooling Operation

Another also very commonly used layer in CNNs is the Fully Connected Layer. These layers usually appear at the end of the CNN. They work as a traditional MLP where every neuron in the previous layer is connected to every neuron in this layer. Their goal is to take the feature maps outputted by the previous layers, flattened, and predict the final result. In a classification task, the output layer usually has the same amount of neurons as the number of classes. To assign to each class its own probability, this last layer normally takes the softmax function as its activation function. In a regression task, where the goal is not to predict a class but a numeric score, the output layer is normally a single neuron with no activation function.

If one wants for its architecture to work just okay, the aforementioned layers might just be enough. But in order to turn a just okay architecture into a state-of-the-art one, every little tweak and trick on the architecture can be very important. Besides the main layers, sometimes other types of regulatory layers are added into the CNN architecture to optimize its performance such as Batch Normalization and Dropout layers. Usually, before feeding an image into a CNN, the user applies some pre-processing to it. This pre-processing allows for the input data to be standardized which helps massively the learning in the Network. The most commonly used type of batch/data standardization assumes the data follows a Gaussian distribution with zero mean and unit variance, by subtracting its mean to every value and dividing it by its standard deviation. The fact is that normalizing the data should not only be a concern for the data fed into the model but also for the data produced after every transformation. Even if the data is correctly standardized at the input of a certain layer, nothing guarantees that it will also be standardized at its output. Knowing this, back in 2015, a Batch Normalization layer [17] was introduced that can adaptively normalize data during training. Basically this layer keeps track of two additional parameters, moving average of the batch-wise mean and variance observed during training. This kind

of layers helps mitigating the vanishing gradient problem allowing networks to get deeper. Besides that, it also adds a bit of noise to hidden layers, reducing overfitting slightly and helping in the quest for generalization. As for Dropout [18], it is also one of the most effective regularization techniques used to fight against overfitting in a network. It consists on dropping out (setting to zero) random entries of the output feature map of the previous layer (Figure 2.5). The amount of entries set to zero depends on the layer's dropout rate that ranges from 0 to 1, with 0 meaning that none of the entries is set to 0 and 1 meaning that every entry is set to 0. Usually the dropout rate is set between 0.2 and 0.5. For rates closer to 1, the amount of information lost in every training iteration is already quite significant, turning these layers into a bottleneck for Network learning. By setting a neuron to 0, the cost function becomes more sensitive on how neighbouring neurons change their weights during backpropagation, helping the model to generalize. It is important to have in mind that this technique should only be applied during the training cycle. During test time, no units are dropped out, although they need to be scaled down by the the dropout rate factor. This is done to account for the fact that during testing more units are active than during training.

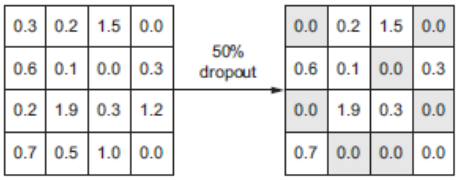


Figure 2.5: Dropout of 0.5 applied to a random 4x4 matrix

So basically CNNs are divided into several learning stages composed of combinations of the various layers just mentioned. The convolutional layers, resorting to their set of filters, extract locally correlated features by applying convolutions to several image slices. Their outputs are subject to an activation function that inserts non-linearites into the system not only helping it in learning abstraction but also making it more robust and capable of solving more complex problems. Downsampling layers help decreasing the system dimensionality and to summarize the previous layers results. The regularization layers mentioned above have as their main goal to help the network get a broader view of the problem in order to avoid overfitting.

Even knowing all this, to build a network capable of solving a certain problem with a high performance is not a straight forward task. Having these notions can help getting a starting point for a Network but the odds of it generating the best possible results at its first try are slim. In order to achieve better and better performances, one needs to get past the exhaustive and slow process of parameter tuning. The problem of this process is that sometimes to train and test one version of a network can take long periods of time and the great amount of parameters to tune does not help.

2.2 Recurrent Neural Network (RNN)

Unlike vanilla ANNs or CNNs, Recurrent Neural Networks (RNNs) are able to capture information over time. These networks can remember the past and base their current decisions on what they previously learned. They are able to do this by keeping a state containing information regarding what they have learned until a certain moment. Between different sequences this state is reset since sequences are single inputs to the network.

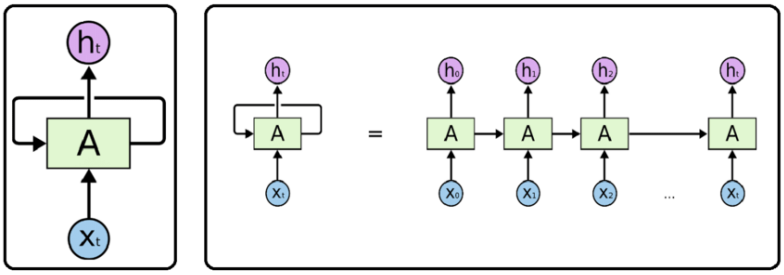


Figure 2.6: Simple RNN architecture (taken from [19])

These networks able to capture the time dynamics of a sequence. So in conclusion, these models can input or output sequences of data that are not independent from each other.

RNNs can be divided into different types. A RNN can either be classified as One to Many, Many to One or Many to Many.

One to Many Networks deal with fixed sized inputs but return sequences of data as their outputs. A common use for these types of networks is Image Captioning. The network takes as an input a single image and produces a sequential caption based on its content.

Many to One architectures take sequences of information as their input and output some fixed sized data. These networks are commonly used in Sentiment Analysis tasks. The RNN takes as its input a given sentence and has to figure out if it is expressing a positive or a negative sentiment. In this work, we will work specifically with these types of networks since the goal is to classify a image sequence into one of the 11 classes/find a concentration value.

Many to Many or Sequence-to-Sequence (seq2seq) networks map input sequences to output sequences. Machine translation is probably the most famous use for this type of architecture. RNNs of this type are able to ”read” a sentence written in Portuguese and return the same sentence written in English, for example.

Due to the completely different architecture of RNNs compared to the previously mentioned ones, the standard method for network training can no longer be used. Due to the recurrent loops present in these types of networks, backpropagation as we know it is not able to update the weights.

To solve this problem, the concept of Backpropagation Through Time (BPTT) emerged as the way

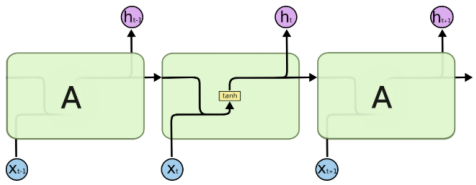


Figure 2.7: Simple RNN Architecure

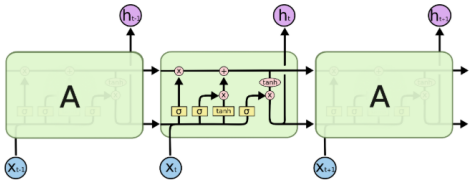


Figure 2.8: Simple LSTM architecture

to train RNNs [20]. On a Recurrent Neural Network, one input is fed to the model per timestep and also one output is predicted. In order to properly train these networks, all timesteps are unrolled. So basically each timestep has one input, a copy of the network and one output. The errors are calculated for each timestep and later accumulated. When all the computations are done, the network is rolled back up and the weights are updated. As the number of timesteps increases, the dependencies between timesteps increase and the number of derivatives calculated also increases which can lead to gradients vanishing or exploding which can ruin training. So probably the main problem when using BPTT is the very high cost of updating a single parameter, which becomes unfeasible for large number of iterations.

To try and fight this problem, a modified version of BPTT called Truncated Backpropagation Through Time (TBPTT) was developed. TBPTT aims to keep the computational benefits of BPTT while relieving the need of backtracking all the way until the first timestep for every timestep [21]. This is done by breaking large sequences into smaller ones that will act independent from each other. Although this can work fairly well in practice, by doing this, the network becomes blind for temporal dependencies that span more than the amount of timesteps of each smaller sequence. For example, if we take a 1000-long sequence and break it into 100 smaller sequences of length 10 each, the network will not be able to learn dependencies between timesteps more than 10 timestep units apart. There is clearly a significant trade-off between the cost of updating a parameter and the amount of timely information lost from the past.

But this was not the only way found to alleviate the vanishing/exploding gradient problem. A new architecture called Long Short-Term Memory LSTM Network allowed to soften this problem and consequently to implement deeper networks capable of dealing with larger sequences.

2.2.1 Long Short-Term Memory (LSTM)

LSTM networks [22] were firstly instated with the goal of overcoming the problem of vanishing and exploding gradients. This new model resembles a typical RNN but where every hidden layer is replaced by a memory cell.

As it is possible to check in Figure 2.8 and 2.7, unlike a RNN that has a single neural network layer in its hidden block, a LSTM has four interacting layers in a special way. The key idea of an LSTM is that it has a state being propagated by every cell block. The state at a certain block contains information learnt

about previous timesteps.

Every block in a LSTM has the capability of changing the information present in this flowing state by resorting to a regulator structure called gate. These gates are usually formed by a sigmoid neural network layers that, together with a pointwise multiplication operation, decide the 'quantity' of information that goes through. A zero value means that no information can go through while a value of one lets all the information pass. By checking Figure 2.8 it is possible to observe that a single LSTM cell uses three gates to control its state.

The first step in a LSTM cell is to decide how much information coming from the previous state is going to be thrown away. For that the LSTM resorts to the so called Forget Gate. This gate takes as input the output generated in the previous timestep (h_{t-1}) and the current timestep input (x_t). According to Equation 2.1, the Forget Gate generates a number between 0 and 1 for each element present in the incoming cell state (S_{t-1}).

$$f_t = \sigma(Weights_f.[h_{t-1}, x_t] + bias_f) \tag{2.1}$$

Having decided which information to keep, it is now necessary to decide which new information should be added to the state. This process is divided in two sub blocks. The first block, composed by a *tanh* layer, generates the candidate values (C_t) to be added to the state. The second block, called Input Gate, works in the same way as the Forget Gate (with a different set of weights and bias) and decides 'how much' of each candidate value is actually added to the state stream. This is all done according to Equations 2.2 and 2.3.

$$C_t = tanh(Weights_c.[h_{t-1}, x_t] + bias_c) \tag{2.2}$$

$$i_t = \sigma(Weights_i.[h_{t-1}, x_t] + bias_i) \tag{2.3}$$

With the forget and update values already calculated it is time to update the cell state. The Equation 2.4 shows how to update the new cell state (S_t).

$$S_t = f_t.S_{t-1} + i_t.C_t \tag{2.4}$$

Now that the cell state is correctly updated, the output can be calculated. As in the state update section, the computation of the cell output is also subdivided in two sections. First, the cell state is subject to a *tanh* layer to push its elements to values between -1 and 1. Secondly, the Output Gate is going to define what values are going to be sent to the output and in which proportions. Once again this gate works in the same fashion as both previously mentioned. The output values can then be calculated

using Equations 2.5 and 2.6.

$$o_t = \sigma(Weights_o.[h_{t-1}, x_t] + bias_o) \tag{2.5}$$

$$h_t = o_t.tanh(S_t) \tag{2.6}$$

The LSTM just described represents the model of the Classic LSTM. From this model, several other models emerged that may or may not get better results than the classical one, depending on the task at hand. From it emerged models like the Peephole LSTM, the Gated Recurrent Unit (GRU), the Multiplicative LSTM and LSTMs with Attention.

The Peephole LSTM only differs from the typical LSTM on the fact that the current cell state also takes place as an input of the three gate layers (Forget, Input and Output). This configuration, for example, showed an increased accuracy in its ability to count and time distances between rare events [23].

The Gated Recurrent Unit network is a simpler version of the traditional LSTMs and, due to that fact, it exhibits a faster learning speed. These networks combine the Forget gate and the Input gate into a new Gate. Besides this, in this architecture, the cell state and the hidden outputs have been merged into a single internal hidden state. Although they have been successfully implemented for simpler seq2seq models or in exotic concepts like Neural GPUs [24], they are usually considered as less powerful versions of the traditional LSTMs due to their counting limitations.

2.3 Object Detection

Object Detection is an area that falls into the Computer Vision field, which deals with the identification and localization of objects in images. This can be done either by drawing a bounding box that contains the object or by selecting every pixel in the object (segmentation).

Before discussing some of the existing object detection algorithms, a couple of very important concepts used in their architectures will first be clarified.

2.3.1 Relevant Concepts

2.3.1.A Intersection over Union (IoU)

Intersection over Union (IoU) is an evaluation metric regularly used in object detection algorithms that tests the accuracy of a proposed bounding box. An object detection model is fed with images on which the objects to be detected are correctly marked and labeled. The positions of these objects in the image are defined by bounding boxes termed ground-truth boxes. In order to test the accuracy of the proposed

bounding boxes, they have to somehow be compared with the ground-truth ones. IoU is a metric used in these situations.

Having two bounding boxes, their IoU score is computed by their area of overlap over the area encompassed by both of them (Figure 2.9).

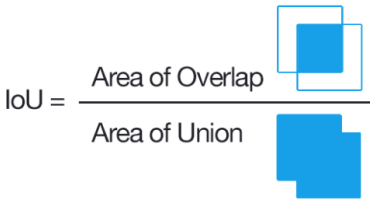


Figure 2.9: IoU score representation

Bounding boxes with low IoU scores probably are not detecting any wanted object so they are discarded. The threshold used to discard a bounding box can be differently set on different object detection methods, but a IoU score greater than 0.5 is normally considered a good prediction.

2.3.1.B Non-Maximum Suppression (NMS)

Non-Maximum Suppression (NMS) is a technique used in object detection algorithms that makes sure that every object is only detected once. Having a list of proposed bounding boxes for a certain class and their IoU scores with the ground-truth boxes, it is possible to apply the NMS algorithm. The box with the highest IoU score is taken as the reference box. Then, it is calculated the IoU score between the remaining boxes and the reference box. The boxes that get a score higher than the threshold are discarded. This means that these boxes are in a very similar position to the reference box and probably are detecting the same object so they are not worth keeping. In theory, with this algorithm, only the proposed bounding box with the highest IoU score with the ground-truth of a certain object is kept. Figure 2.10 shows a good NMS example.

Figure 2.10: NMS application example (taken from [25])

2.3.2 Region Proposals

Back in 2014, the paper regarding Region Based Convolutional Neural Networks (R-CNNs) was released [26] stating to present the first object detection algorithm using CNNs inside its architecture. The first step taken by this method consists on generating about 2000 category-independent region proposals on the input image using a selective search algorithm. For every proposed region, a feature map would be generated resorting to a CNN and later classified by category-specific linear Support Vector Machine (SVM). Besides predicting the presence of an object inside a certain bounding box, the algorithm also predicts four offset values to adjust its size and position in order to increase its precision. Every generated region proposal is first subject to a simple technique called 'Affine Image Warping' in order to compute its fixed size CNN input. The method was called R-CNN since it is the combination between (R)egion proposals and CNN features. The overall scheme of the architecture is represented below in Figure 2.11.

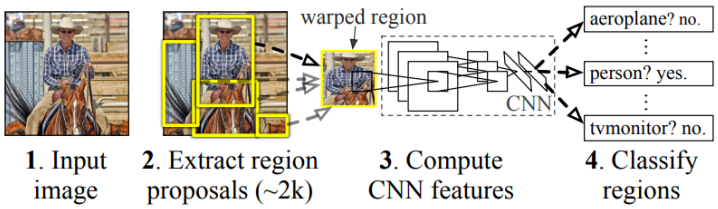


Figure 2.11: R-CNN System Overview [26]

This algorithm, being the first implementation of a region proposal network using a CNN, still had lots of things that could be improved. Since 2000 region proposals per image have to be classified, training takes a huge amount of time. Besides that, this method cannot be implemented in real time situations since it takes several seconds to test an image.

Some of the drawbacks of the R-CNN were solved in a new algorithm called Fast R-CNN [27] published by the same author in the following year. One major improvement on the old architecture was that instead of feeding all 2000 region proposals to the CNN, only the input image is fed into it to generate its convolutional feature map. The fact that 2000 feature maps are not being fed into the CNN speeds up training significantly. The region proposals are then identified from the image feature map. For every region proposal, a Region of Interest (RoI) pooling layer extracts a fixed-length feature vector so that it can be fed into a fully connected layer that branches into two output layers. One produces the probability of an object belonging to a certain class, making use of the Softmax activation function, and the other outputs four real-valued numbers to refine the bounding boxes. The Fast R-CNN architecture is shown in Figure 2.12.

The performance in terms of time between R-CNN and Fast R-CNN can observed on the graphs of Figure 2.13.

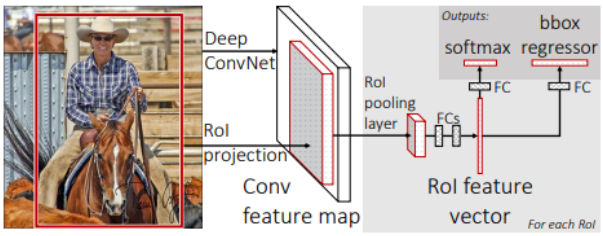


Figure 2.12: Fast R-CNN System Overview [27]

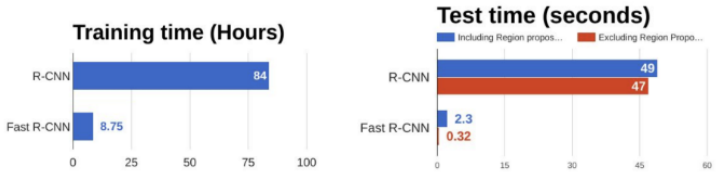


Figure 2.13: Time comparison between R-CNN ad Fast R-CNN on train and test times [27]

From the graphs in Figure 2.13 it is possible to conclude that the changes made on the Fast R-CNN model made it extremely faster both in training and testing time when compared to the R-CNN. Now, the focus should turn to the testing times of the Fast R-CNN model when including and excluding region proposals. When using region proposals, the testing time is almost 7 times higher that when region proposals are being excluded. From this, it is possible to conclude that the region proposal technique became a bottleneck in Fast R-CNN and its significantly affecting its performance.

Since the region proposal technique applied was being computationally expensive, in 2016 was introduced a new network based on the previous one called Faster R-CNN [28] which had as its main focus to diminish the time spent generating region proposals. In order to do so, a Region Proposal Network (RPN) was invented to directly generate region proposals, predict bounding boxes and detect objects. The Faster R-CNN is then a combination between this new RPN and the old Fast R-CNN model. First, the input image is fed to the CNN to produce its feature maps. Then, a 3x3 window slides all over the maps to output feature vectors capable of being fed to two fully connected layers. The region proposals resultant from this process are called anchors. If an anchor box value for ”objectness” is lower than a certain threshold, the box is discarded. ”Objectness” is a measure related to the chance of an object being in a certain box. Avoiding the selective search method and using a RPN instead, accelerates training and testing as well as network performance. The graph for the test time on R-CNN networks is presented in Figure 2.14.

2.3.3 You Only Look Once (YOLO)

While other object detection algorithms need to go through an image several times before being able to detect an object, YOLO [29] only has to look at it once, hence the name You Only Look Once. The

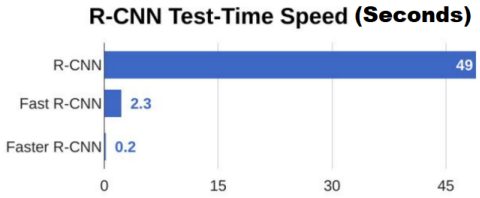


Figure 2.14: R-CNN algorithms Test-Time speed [28]

YOLO developer team reframed object detection as a single regression problem capable of extracting bounding box coordinates and class probabilities straight from image pixels.

At first, an image is divided into a 7x7 grid. If the center of a certain object falls into a specific cell, it becomes responsible for detecting that same object. The problem with this approach is that any grid cell can detect only one object. This means that only a maximum of 49 objects can theoretically be detected and if a cell contains more than one object it will not be able to correctly detect all of them. These problems were solved in more recent versions of YOLO.

Each one of the 7x7 grid cells predicts 2 bounding boxes. Every bounding box is subject to the IoU method to compare them to the ground truth boxes. Boxes with a score higher than a certain are kept and the remaining ones are rid of. The model also outputs 4 numbers that represent the location of the box relative to the bounds of its corresponding cell. To obtain the bounding box predictions and their class probabilities, YOLO uses only a single CNN (Figure 2.15).

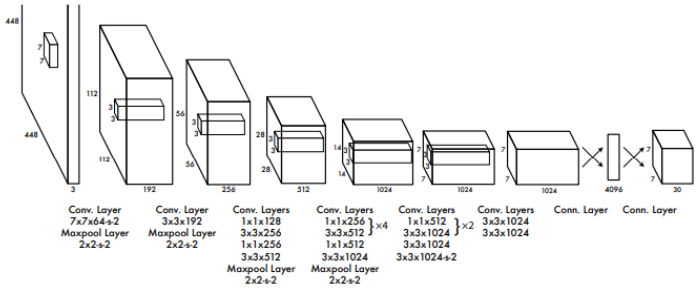


Figure 2.15: Original YOLO architecture [29]

Although the speeds obtained for the first model were incredibly fast, allowing the algorithm to be used in real time applications, the number of localization errors was significant and the network presented a relatively low recall. So to try and achieve a better performance, a new YOLO version was developed (YOLOv2) [30]. In this new architecture, batch normalization layers were added after every convolutional layer and the resolution for detection was increased. But the main introduction was the use of Anchor Boxes which allows multi-object prediction per grid cell. The fully connected layers previously used on YOLO to predict bounding boxes were removed and replaced by anchor boxes. Anchor boxes are a set of predefined bounding boxes with certain values of height and width. They are chosen

according to the typical ground truth boxes present in the dataset to use. These anchor boxes are tiled across the image, and the predicted bounding boxes are basically refinements of these original boxes.

2.4 State of the Art

2.4.1 E-noses

As previously discussed, an e-nose consists on both an hardware and a software component. Drawing a parallelism between these devices and a real biological olfactory system, the hardware component is seen as the olfactory chemoreceptors in the e-nose system while the software component is more closely associated with the brain and its functions. The software part, the one focused in this work, mainly consists on a data processing unit capable of classifying the different scents detected by the hardware component. Choosing the suitable software, according to the used hardware, is very important when looking to produce an efficient and highly accurate device. This section will focus on the different pattern recognition algorithms and techniques used in already successfully tested and implemented e-noses.

Besides the ANN based algorithms regularly used in e-nose devices, there are also other tools and techniques used in e-noses worth mentioning that helped them achieve good results. Three commonly used methods are Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Support Vector Machines (SVMs).

PCA is a famous worldwide used tool implemented with the goal of linearly reducing a dataset dimensionality. This method tries to explain high dimensional data using a smaller amount of variables called the principle components. By reducing the data dimensionality, it can help speed up significantly the training process of a network although sometimes at the cost of loosing some accuracy. It is mostly used in exploratory data analysis and in predictive models. An e-nose with the mission of discriminating different types of Iberian hams [31] used PCA on the data provided by the sensors before feeding it into the implemented neural network, showing great success.

Another widely used technique for dimensionality reduction is LDA. While PCA is an unsupervised tool that does not take into consideration the different classes in a dataset, LDA is a supervised tool aiming to maximize class discrimination taking their labels into account. Also several of e-nose implementations used LDA in their pipeline. An e-nose used for characterization and classification of Barbera wines [32] that used LDA in their data processing architecture was able to achieve almost 100% correct predictions.

SVMs are supervised learning models utilized for both regression and classification problems. Being a supervised learning technique, they are commonly applied to linear and nonlinear binary classification problems. This method computes the best fitting hyperplane (decision boundary) to distinctly classify

data points. Although SVMs are not one of the hottest areas in Machine Learning anymore, with the rise of deep learning, they still prove to be useful in several e-nose applications. A good example was the use of an SVM in an e-nose with the goal of estimating black tea quality [33]. According to the paper, experimental results showed that the SVM models used offered more than 97% accuracy when predicting considerable variations in tea quality.

2.4.2 Convolutional Neural Networks (CNNs)

In recent years, we have been witnessing the birth of several new CNN architectures. What, at the beginning, could be described as simple architectures with few layers, quickly scaled to much more complex architectures, extremely difficult to analyze but with whom came significant accuracy rises. From 1998, when LeNet-5 [34] was introduced, to nowadays plenty of new CNN models emerged like AlexNet [35] in 2012, VGG-16 [36] in 2014, several versions of Inception [37] [38], ResNet-50 [39] in 2015, Xception [40] in 2016, ResNeXt-50 [41] in 2017 and many more.

Some of the most important innovations in the CNN area have come from submissions to the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC), commonly referred to as the ImageNet challenge. ImageNet is a huge dataset containing over 14 million images associated to more than 21 thousand classes, according to their webpage [1]. This annual competition, held between 2010 and 2017, aimed to promote the development of new and better computer vision algorithms and to benchmark the state-of-the-art. The state-of-the-art of image classification algorithms in this section will be based on results achieved with these datasets. This does not mean that if an algorithm performs better than another one in this dataset, it is better overall. Algorithm performances can change significantly depending on which dataset is being used.

The first major milestone regarding CNN networks was recorded in 2012 when the so-called AlexNet reached a top-5 error rate of 15.3% completely outperforming the accuracy of the previous best model (26.2%). AlexNet (Figure 2.16) is considered nowadays a simple CNN architecture, presenting only five consecutive alternated convolutional layers and max-pool layers and three fully connected layers at the end.

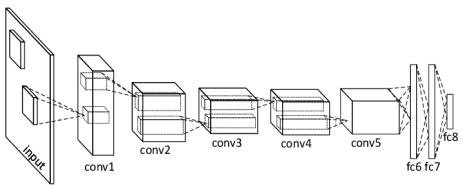


Figure 2.16: AlexNet architecture [35]

Since this milestone, reasearchers kept trying to outperform the AlexNet implementing deeper ar-

[1] http://www.image-net.org/

chitectures. In 2014, the VGG16 model (Figure 2.17) was released which reached a top-5 error rate of 7.3%, reducing by a factor of two the error achieved by the AlexNet model. As suggested by the model's name, this new model was composed of sixteen convolutional layers which represents a huge rise in depth compared to the previous models. The introduction change in filter size for 11x11 to 3x3 played a huge part in the accuracy increase.

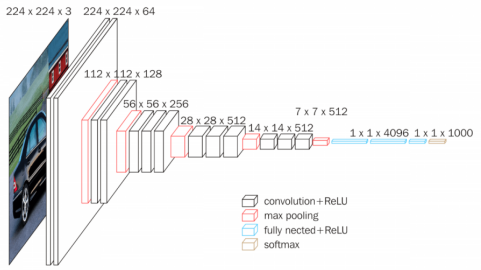


Figure 2.17: VGG16 architecture [36]

In the same year appeared the concept of 'Inception Modules'. While the typical convolutional layers use linear transformations with a nonlinear activation function, such as ReLU, these new modules trained multiple convolutional layers simultaneously and stacked their generated feature maps into an MLP, which also kept the wanted non-linearities in the system. This idea was implemented in a network called GoogLeNet (Inception V1). This system architecture contained several of these new models and was able to reach an error rate of 6.7%. After tuning some parameters and applying a few architecture changes, the same team that developped Inception V1 [37], also launched Inception V2 [38] and Inception V3 [38] that achieved top-5 error rates of 5.6% and 3.58% respectively.

While the trend at the time was to keep increasing network depth in order to achieve better results, some researchers noticed that increasing depth is associated to an increase in error rate due to difficulties in training and optimization in very deep models. It was introduced the concept of 'Residual Learning'. In Residual Learning a connection between the input of one or more convolutional layers and their output is established. This allows for patterns in the input image to be learned in deeper stages of the network. In 2015 a model called ResNet, taking advantage of this principles, got a error score of 3.57%.

The combination of inception blocks with residual blocks gave birth to Inception-ResNet (Inception V4) which outperformed all the models at the time with an error score of 3.08%.

Being ImageNet a huge dataset with millions of images and thousands of different classes, obtaining error rates close to 3% can be considered a huge victory and proof that there are already very powerful algorithms capable of handling real life tasks with great accuracies. Also, since deep learning and computer vision are two very hot topics nowadays, and with different new models being presented regularly, it is very likely that, soon, other architectures emerge and outperform all the other ones.

Chapter 3

Proposed Approach

3.1 Data

3.1.1 Data Acquisition

Since this work is focused on the pattern recognition side of an e-nose it has to work on top of an already produced dataset. The Biomolecular Engineering Group, UCIBIO, REQUIMTE, FCT/UNL developed a gas-sensitive gel which can be used for odor detection [42]. This hybrid developed gel consists of biopolymeric matrices encapsulating IL–liquid crystal self-assembled droplets. Hybrid gels combine the ionic conductivity of ILs with the unique optical properties of LCs, resulting in optoelectrical stimuli-responsive materials. Images of this gel, under a polarized optical microscopy, obtained during exposition to the odors (VOCs), show different dynamic patterns. These patterns can be observed on several droplets inside the gel. The setup used to collect both datasets (11 VOCs and Concentrations Experiments) is presented in Figure 3.1. The produced hybrid gel films were placed inside an in-house-designed glass chamber. This chamber is isolated from ambient air and light, and is fixed under a polarized optical microscope. There are two pumps connected to the chamber, one being in charge of pumping the VOC inside it and the other one in charge of injecting ambient air, getting rid of the VOC inside it. A full cycle starts when one of the pumps injects a gas, containing a certain VOC, into the chamber. Then, after a few seconds of exposure, the second pump pushes ambient air into the chamber to remove the gas and bring the system back to its original state. These actions correspond to the exposure and recovery cycles, respectively.

Every full cycle is recorded in video format by an optical microscope (Zeiss Axio Observer.Z1/7 coupled with an Axiocam 503 color camera) with a 1936x1460 resolution. All of the recorded video samples capture a specific region of the gel containing several droplets that evolve through time creating many dynamic patterns. In Figure 3.2 and Figure 3.3 it is possible to observe a excerpt of the Recovery cycle

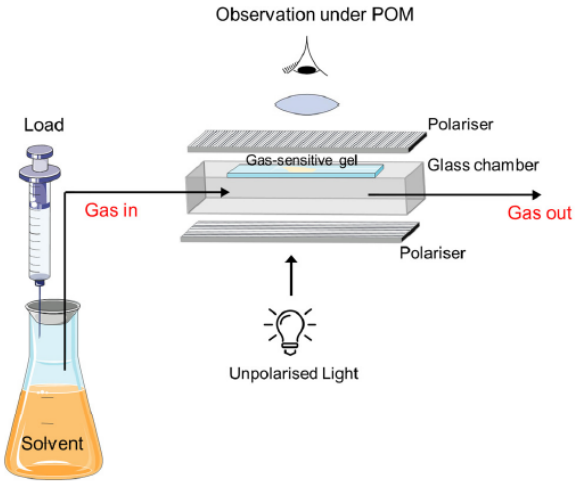


Figure 3.1: Visual representation of the setup used to collect the dataset [42]

and the evolution of a single droplet through an entire cycle in one of the video samples, respectively. As the gas containing the VOC hits the chamber, the droplets evolve in a way that looks like they have disappeared. As soon as the recovery cycle starts and clean air starts being pumped into the chamber, the droplets begin to appear once again spinning back into their original state.

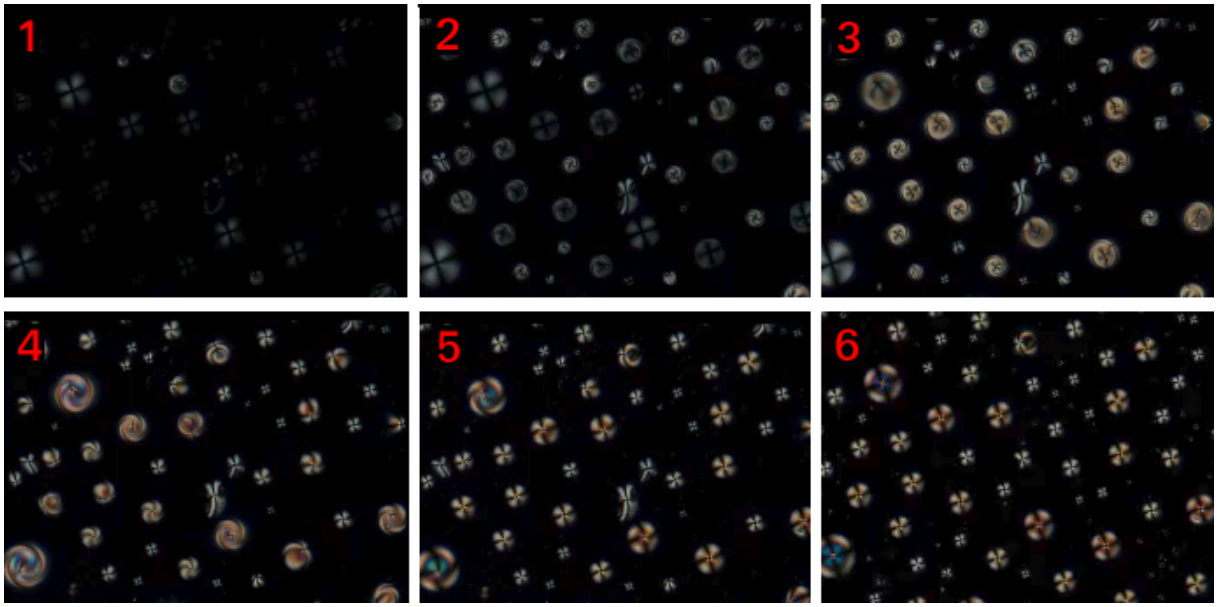


Figure 3.2: Excerpt of a collected Acetone image sequence in recovery cycle

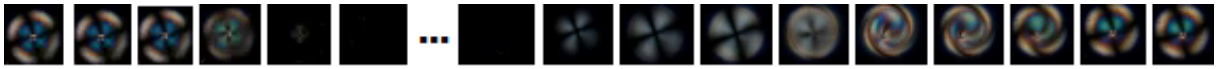


Figure 3.3: Acetone droplet evolution in an entire sequence

3.1.2 Droplet Annotations

All video sequences in both datasets explore the evolution of a certain area within the gel when exposed to a VOC and then exposed again to clean air. Since the data provided is completely raw, with no annotations whatsoever, and the proposed work is focused on the timely evolution of isolated droplets, every droplet in a sequence needs to be labelled. This process can either be executed manually, where every annotation in an image is done resorting to pure human work, or in a more automatic way where a computer program does the labelling itself. Taking into consideration the nature of the data, where in each sequence there are hundreds of droplets in a black background, annotating all of them manually would be a very time consuming task so an automated strategy had to be adopted. Besides that, the high contrast between the actual droplets and the background clearly allows for this type of strategy.

The goal is to identify pixels in an image that belong to the same droplet. Having all the pixels that belong to a droplet it is possible to get the coordinates of the bounding box that contains it.

The first step consisted on transforming an image to a binary one (black and white), using *Otsu's* method which computes the threshold that makes the distinction between foreground (droplets) and background (Figure 3.4 LEFT). Then, and since the droplets are not perfectly delimited and completely filled circles, there are some black areas in them that might difficult this process later on. A process of image dilation is applied to try and close the droplet borders and right after that it is applied a function that fills image holes (Figure 3.4 CENTER). A hole is considered to be a set of pixels with background value that cannot be reached from the edge of the image. The black areas in the droplets already enclosured by white pixels are turned into white. After these, several connected objects are identified, where a connected object is a set of pixels in which a path connecting two pixels in it always exists. To conclude, objects that contain less pixels than a certain threshold are discarded (Figure 3.4 RIGHT). This action removes droplets that are too small to be considered as well as impurities in the gel that got enhanced by this process.

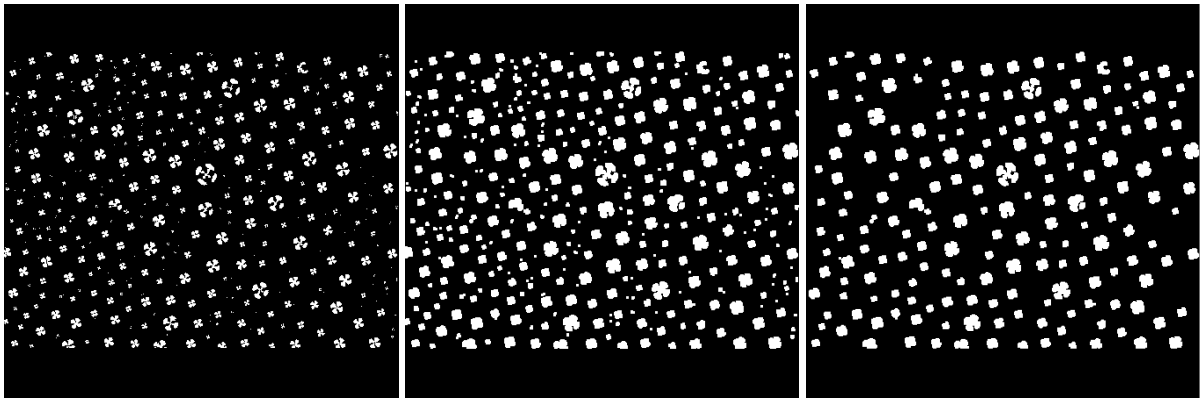


Figure 3.4: Image treatment for annotation: **Left** - B&W Representation **Center** - After image dilation and hole filling **Right** - After discarding small connected objects

Finding the bounding boxes for the droplets is now only one step away since the pixels that belong to each one of them are known. Given that the droplets are basically round, their center is found by computing the mean pixel coordinate and the radius by finding the distance from the center to the farthest pixel. It then becomes straightforward to find the 4 vertices that characterize every bounding box. The outcome of this strategy is presented in Figure 3.5.

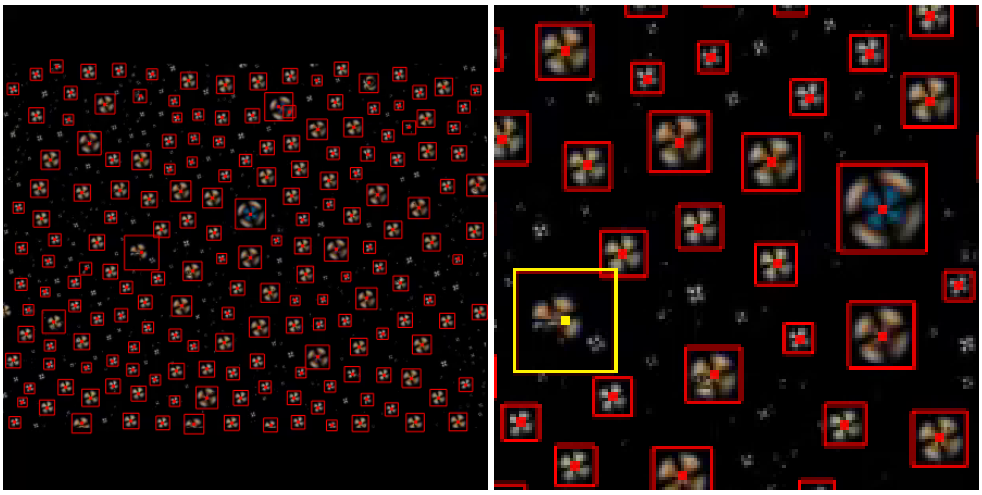


Figure 3.5: End product of automated annotation

It is possible to observe in Figure 3.5 that the majority of the bounding boxes fit very well with the corresponding droplet. Nevertheless, some droplets did not get boxed perfectly which required a manual tuning in order to correct some bounding boxes coordinates. For example, the bounding box highlighted in yellow, represents a perfect case of this situation. This probably happened due to the significantly smaller droplet also inside the bounding box that, when dilated, got attached to the bigger one and corrupted the bounding box coordinates.

Since the droplets stay in the same position during the exposure to the VOC, this process of annotation can basically be done only for the first frame of every video and then the bounding box positions remain the same throughout the video. In some droplets it is possible to observe a very slight change in position over time but these events were ignored due to their rareness.

3.2 General Overview

This Thesis is divided into two parts, each one resorting to a different set of data and different goals, but both with a very similar nature. Both datasets are composed by several image sequences where a large number of droplets can be observed. These droplets and their evolution through time are the subjects of study in this Thesis. Therefore, to extract the actual data from the original datasets, an image processing algorithm had to be developed.

Given that the ultimate goal of this project is to develop a device capable of identifying substances/find their concentrations in the air, according to the visual reaction of small droplets over time, two main tasks have to be performed: **Detection** and **Classification/Regression**. Correctly detecting the droplets is the target for the first task and it is shared for both parts of this Thesis. The idea was to train a model capable of mastering droplet detection and that was able to perform this action on a very short period of time so it could eventually be implemented on a real time system. It is crucial for this model to obtain spot on detections in order to take full advantage of the models that follow. The second task was to develop a system that, given droplet sequences, would be able to make predictions regarding the VOCs present in the air or their concentration. Given the theme of this thesis, both these tasks were performed having CNNs at their core.

In regards to the detection stage of the system, given the nature of the images in the dataset (coloured droplets on a black background), it probably could have been executed with some simple image processing algorithm that would only take into consideration the pixel colours in an image. But then again, since the theme of this work is related to CNNs, this strategy was not adopted. Instead, YOLO was chosen, an object detection algorithm that, as seen in Chapter 2.3.3, has CNNs in its architecture and besides that is known for its real time detections which are ideal for this problem.

Now with respect to the classification part of the system, the idea was to start with a simple and naive approach to the problem that would work as a baseline to more powerful and complex techniques, that in theory, would be able to achieve better results. Thus, the first applied strategy only took into consideration the spatial features of a droplet sequence, completely disregarding its time component. Each frame in the sequences was treated independently, only resorting to two dimensional CNNs. This method, paired up with a simple voting system that would make up for the disregard over the third dimension, offered some results to try and improve on with the strategies that followed. To follow up on this, and given that image sequences can also be considered as 3D images, where the third dimension corresponds to the temporal component, plenty of experiments recurring to three dimensional CNNs were performed. Depending on the results achieved by the original experiences, different paths were taken to try and improve the previous results. Some ensembling techniques were also implemented in order to try to obtain a better predictive performance, by taking advantage of the strong points of different models. In theory, with these methods, it is possible to achieve better results than the ones achieved by each of its constituting models alone. Finally, since the data is comprised of temporal sequences, RNNs had to be tested. LSTM models were used to try and exploit the temporal dimension of the data on a different way from regular three dimensional CNNs. The models used in the Experiments are described in further detail in Section 3.4.

The regression problem, present in the second part of the work, as mentioned, has a lot of similarities to the classification one. The majority of the strategies applied for the first part could also be applied

to this problem with some slight tweaks on the networks, specially on their last layer and cost function. Coming into this stage of the work with already lots of executed tests in the first problem helped to give an idea on which strategies had a highest chance of achieving better results.

Figure 3.6 presents a scheme of the overall system, with both the Detection and Classification/Regression stages.

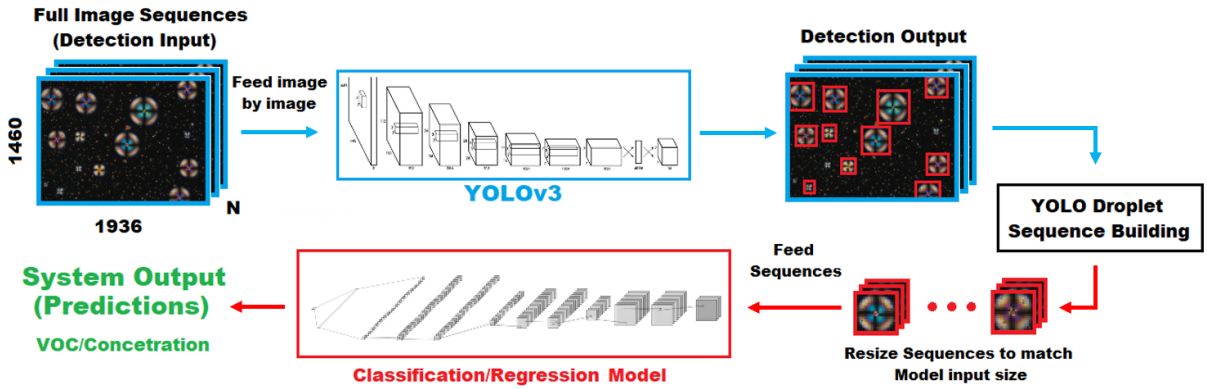


Figure 3.6: Detection and Classification/Regression Approach Overview

3.3 YOLO Droplet Sequence Building

The YOLO network produces bounding boxes around droplets on single images and not on 3D images/image sequences. In both parts of this Thesis, the goal is to produce a result based on an entire droplet sequence so a simple strategy was developed in order to build these sequences from the detections performed on all the frames. The YOLO detections on the first frame could simply extend to every other frame but this could prove to be very error prone. Instead, droplet bounding boxes are predicted for every frame. Since every droplet has a shape very similar to a circle, the bounding boxes, in theory, should all be squares in order to circumscribe them perfectly. Thus, bounding boxes with a width that exceeds the height by more than 20%, or vice-versa, are discarded right away. Another performed inspection is to check for overlapping bounding boxes in the same frame. If two bounding boxes present an IoU score over 0.3, they are considered to be detecting the same droplet so only one can remain. The image is turned into black and white and the bounding box containing more white pixels inside it is kept since it probably contains a bigger portion of the droplet, with the other being discarded. In order to put together bounding boxes from different frames and associate them to the same droplet, it is used once again the IoU metric. Bounding boxes from the first frame are set as references. Then, every bounding box in the remaining frames that achieves an IoU score above 0.5 with one of the reference ones is associated with it, meaning that it is very likely to be circumscribing the same droplet on a different frame. With this strategy, if a certain bounding box on the first frame does not have a very good

set of coordinates, it may not be associated with other ones and a droplet sequence might be missed. However, since the first frame is the one where the droplets are still idle and in a more 'clear' form, it is in theory the frame less prone to substantial errors. After doing this association, sequences that contain very few bounding boxes, less than 25% of the number of frames in a sequence, were also discarded. Finally, the mean coordinates values are computed to obtain the final ones. In order to feed these sequences to certain CNN classification/regression models, the droplets are cropped from every frame, resized to the model's input size, and put together to form a sequence. Every time a model evaluates its performance on the test set using the YOLO detections, this is the approach used to assemble the droplet sequences.

3.4 Models

3.4.1 CNN 2D/3D LeNet-5 Based Model (Baseline Model 1)

This CNN architecture was based on the LeNet-5 [34] model. This LeNet-5 network is made up of 7 layers, where the first four consist of two sets of Convolutional and Sub-Sampling Layers, followed by two fully connected layers and a softmax classifier at the end. This model was used for handwritten character recognition where the input images had a dimension of 32x32 and only one colour channel. The first and second convolutional layers applied 8 and 16 different filters of 5x5 dimension, respectively. The sub-sampling layers applied average pooling on the previous features maps with 2x2 filters with stride 2. The tanh function was the chosen one to serve as activation function for all layers, except the last one. This model, as represented on the original paper, is shown in Figure 3.7.

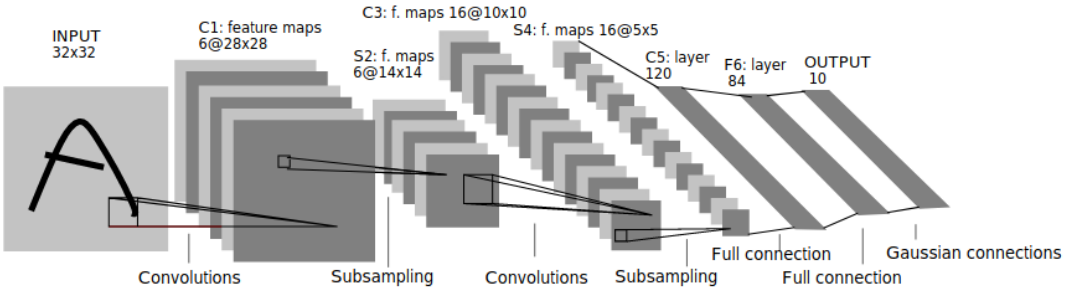


Figure 3.7: LeNet-5 architecture [34]

The proposed model was based on the LeNet-5 one, but some changes and some additions were applied to it. The seven original layers were kept, but other two were introduced, the dropout and batch normalization layers. As mentioned in Section 2, these layers work as regulators in the model to prevent/delay overfitting in the model. According to the author of the Batch Normalization paper [17], these layers should be applied before the Activation Function. As for the Dropout layers, they should

be implemented after the computations of said activations. With this being said, after both convolutional layers a Batch Normalization layer was inserted right before its activation and a Dropout one right after. The same was done after the first fully connected layer. Another applied change was to use ReLU as activation function instead of using tanh. With respect to the last layer, for the experiments where the goal was to classify a sequence among 11 VOCs, a 11 unit one was used paired with the softmax activation function which allows the association of a confidence percentage per VOC. In the experience where the goal is to find the concentration of Acetone exposed to the gel, this layer was replaced by only one neuron with no activation function associated to it in order to predict values on a linear scale. Finally, regarding the parameters like filter dimensions, number of filters, dropout rate and so on, none of them was set permanently since different combinations were tested in order to find the one that produces the best results on the validation set. The input images/sequence dimensions can also vary from experience to experience.

3.4.2 CNN 2D/3D Deeper Model (Baseline Model 2)

In an attempt to try and achieve better results than the previous ones that used the LeNet-5 based model (Section 3.4.1), a new CNN model architecture was developed. This new model was composed by a total of 10 layers. Three sets of two convolutional layers followed by a max pooling layer make up the convolutional section of the network. At the end, after flattening the output of the last max pooling layer, there is only the eleven unit layer paired paired up with a softmax activation function. Every convolutional layer still uses ReLU as its activation function. Besides that, padding was applied in this layers to keep the dimensions of the feature maps unaltered after the convolution. Regarding regularization, only one dropout layer was used after the last max pooling layer. A schematic of a 2D version of this model can be found in Figure 3.8. As mentioned on the previous model, parameters within the layers were not set permanently since there is no indication before hand on which are the better ones. This information only comes with the training.

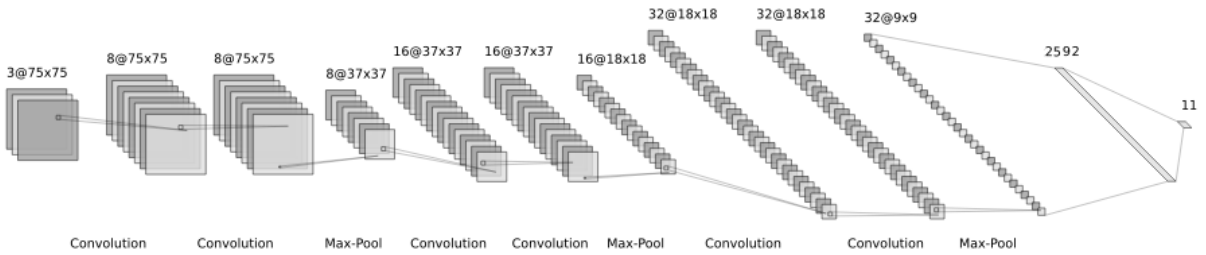


Figure 3.8: CNN2D (Baseline Model 2) architecture

3.4.3 CNN2D + LSTM

As it was already mentioned, given that this problem works on time series data, at any point in the experiments, LSTMs would have to be used given their proven effectiveness on problems with these kinds of data. The type of LSTM applied was of the type Many to One since its input is a sequence of data and the output is a class/VOC or concentration, obtained through the returned values of the last cell. The amount of cells that compose the LSTM is the same as the sequence length. These are image sequences so they are not fed directly to the LSTM cells without any prior pre-processing due to the high complexity of the feature space. Instead, each image goes through a 2 Dimensional Convolutional Neural Network (CNN2D) (same for all timesteps), and the output features are then fed to the respective LSTM cells. The output returned from the last LSTM cell, that is already influenced by every timestep, is connected to either a 11 unit softmax layer or a single neuron, according to which problem it is being applied to. The only regularization used here is a dropout layer between CNNs and the LSTM. The experiences that follow, besides using unidirectional LSTMs, also employ Bidirectional LSTMs (BiLSTMs) where the inputs are also managed from the future to the past. Instead of only one, these LSTMs have two hidden states and they use the combination of both of to produce the end result which can be beneficial given the additional data 'perspective'.

The scheme of an LSTM paired up with 2 dimensional CNNs is represented in Figure 3.9. In this example the LSTM output is connected to a single neuron (Regression problem).

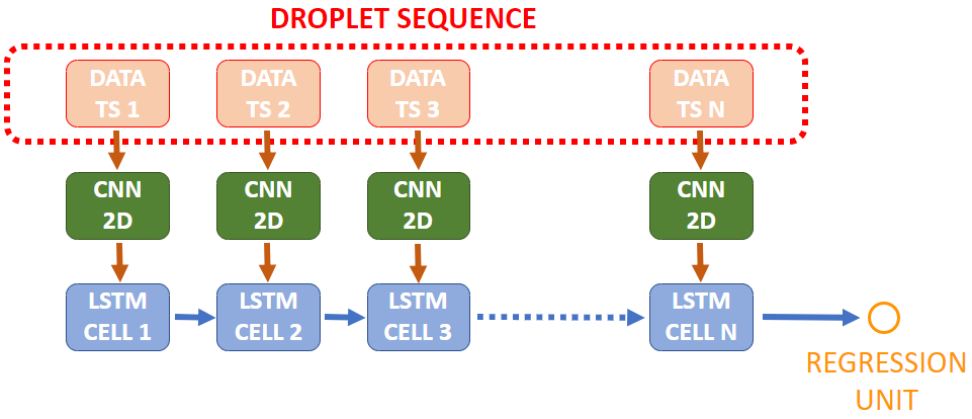


Figure 3.9: LSTM + CNN2D architecture scheme

Chapter 4

Experiments & Results

4.1 VOC Experiment - 11 VOCs

As mentioned previously in this document, the first line of work consists on correctly identifying the VOC present in the gas exposed to the gel out of 11 possible ones. The 11 VOCs selected for this experiment were: Acetone, Acetonitrile, Chloroform, Dichloromethane, Diethylether, Ethanol, Ethylacetate, Heptane, Hexane, Methanol and Toluene. These are VOCs representative of different chemical classes. The chemical structures of every component are represented in Figure 4.1. This just comes as some piece of additional information that will not be taken into consideration during the experiments.

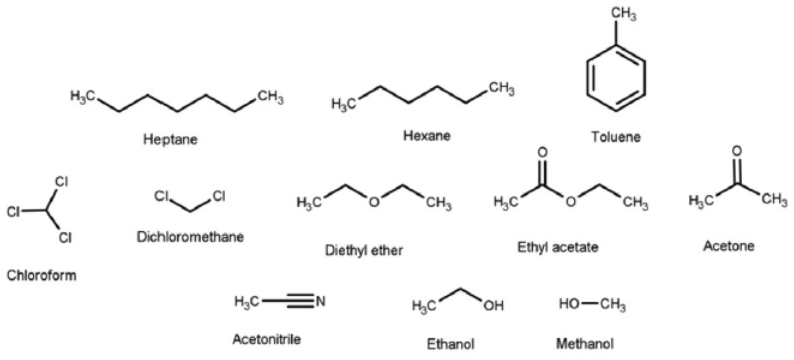


Figure 4.1: Chemical Structures of the 11 VOCs to which the gel was exposed [42]

Since there are 11 distinct VOCs in study, 11 different gels were selected and each one was to be exposed to a different VOC. 11 videos were recorded and in all of them, the respective gel region was exposed 5 times in a row to its respective gas. An exposure here consists on an Exposure Cycle and a Recovery Cycle.

4.1.1 Sequence Splitting

As mentioned before, there are 11 videos, one per VOC, where the gel in each one is exposed to the gas 5 consecutive times. In the following sections, where Neural Networks will be trained and tested, the more data available the better. So in this scenario, each exposure to the gel is considered to be completely independent from the others. The team that acquired the data did not provide any information regarding the time or frames that delimited each exposure so a simple analysis of the data is needed in order to split each sequence. The chosen approach consisted on plotting the sum of every pixel's colour in every frame.

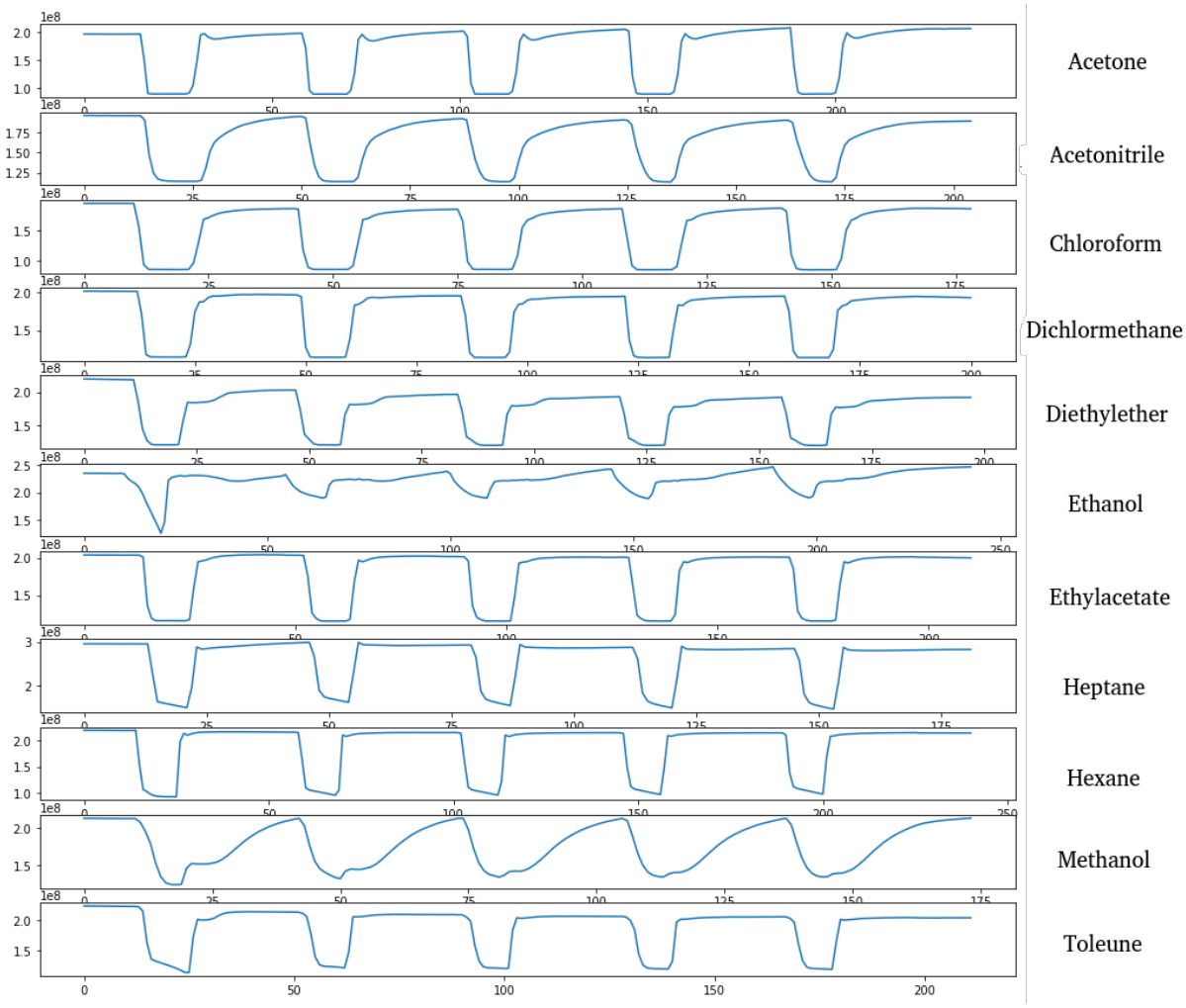


Figure 4.2: Total frame pixel colour sum (3 colour channels) per VOC

From this analysis resulted the 11 plots presented in Figure 4.2. It is easy to confirm that every gel was exposed 5 consecutive times to its respective gas, given the pattern repetitions over time. When the gas hits the chamber, the gel droplets start to change their optical configuration, and the overall level of colour in the images starts to drop. When the recovery cycle begins (chamber gets filled with clean air),

the droplets start evolving back to their idle state making the colour levels rise again and converging to the values pre-exposure.

To measure the length of the sequences in every video, the frame numbers where the sudden drop in colour level occurs were registered. By obtaining the difference between consecutive frames where the colour drop happens it was possible to determine the length of the sequences. Whereas for Acetone, a sequence had a 43 frame length, for Dichloromethane, the sequences had a 36 frame length. The lengths of the sequences per VOC varied in a range from 32 to 44 which is definitely not ideal. Although the exposure times to the gases were the same in all the videos, the sequence lengths change between videos which was caused by the use of different camera settings between acquisitions. For the majority of machine learning approaches the data input size needs to be very well defined so this situation needs to be rectified.

The solution found was to use the smaller registered sequence length (32) for all the sequences which implies a loss in frames for the majority of sequences. This might not be that big of a deal if the discarded frames are the ones at the end of the sequences where the droplets are already very close to their idle state. The exposure cycle and the majority of the recovery cycle are not discarded since it is where it is believed that the most important and significant information is. Besides that, it was considered that every sequence started 3 frames before the registered drop in colour so that the implemented networks could also capture the moment when the gas gets in contact with the droplets.

After splitting the exposures in every video and applying the changes mentioned to standardize all of them, the colour plots per sequence turned out the ones in Figure 4.3.

4.1.2 Droplet Dimensions

Before engaging on the experiments with the already labelled data, it would be interesting to know more about the droplets in hand. This could prove to be beneficial later on when analyzing results or trying to find new solutions to some problems. To this extent, it was made an analysis regarding the droplets dimensions in the 11 gel regions.

Since the dataset contains 11 videos, where in each video a different VOC is exposed to a certain gel region 5 times, there are in total 11 different gel regions to be analyzed. Knowing that the droplets in a certain area are the same throughout the video, it is only necessary to select a frame per video and study the droplets in it. For this purpose, the first frame of each of the 11 videos, where the droplets are still in their idle state, was selected. In Table 4.1 it is possible to observe the Maximum, Minimum, Mean and Median values regarding droplet diameters and in Figure 4.4 the histograms per video also with respect to the droplet diameters. In these datasets, 220 pixels correspond to 100 μm.

From both Table 4.1 and Figure 4.4 it is possible to observe that the droplet sizes distribution fluctuate a between different videos. Some regions are populated mostly with droplets on the low side of the size

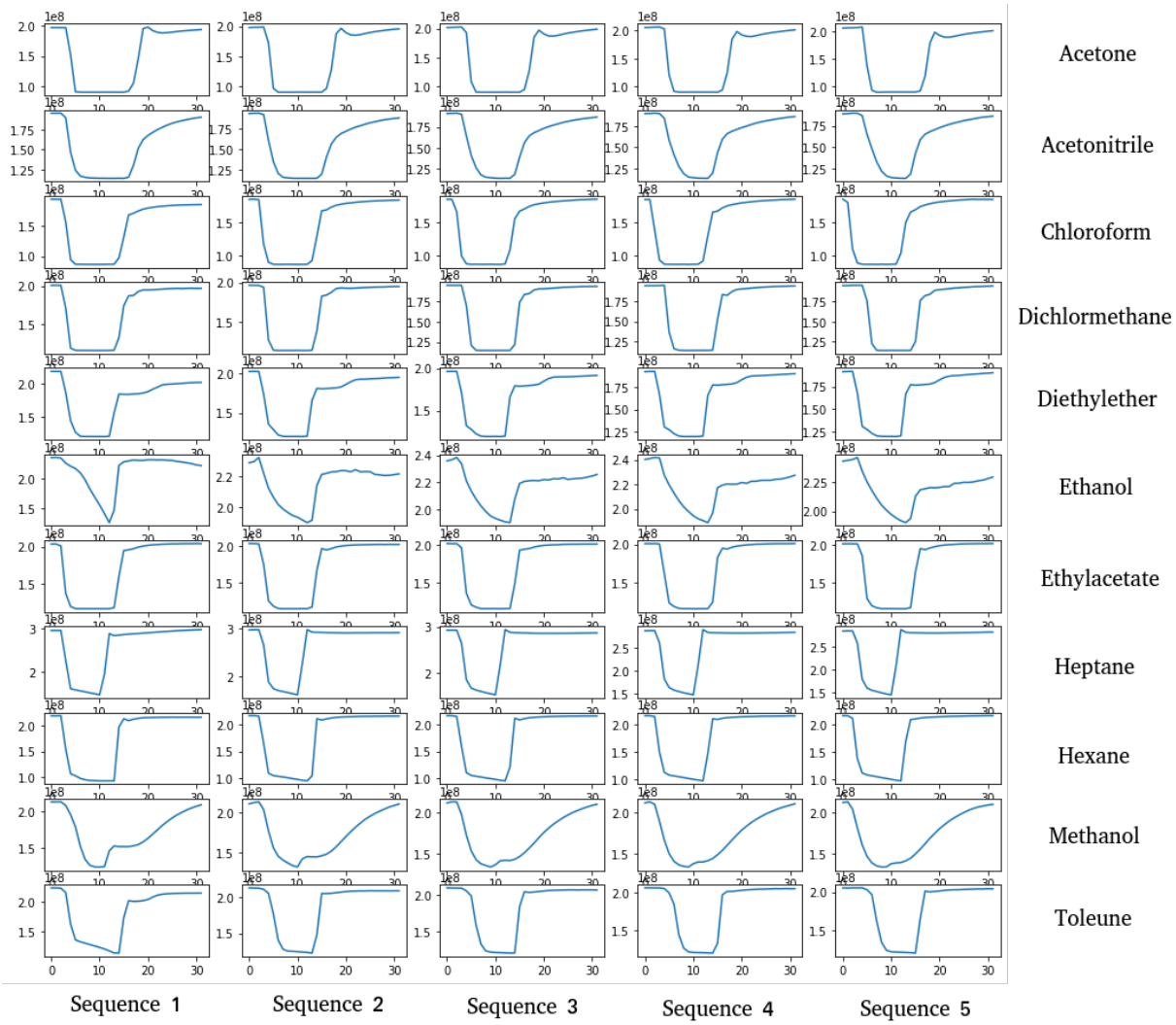


Figure 4.3: Total frame pixel colour sum (3 colour channels) per sequence and VOC

spectrum, like the ones associated to Methanol, Acetonitrile or Ethylacetate, whereas other regions, like the ones associated to Ethanol or Heptane, possess a higher range of droplet sizes. The Biomolecular Engineering Group, that generated the dataset, had a specific interest on the droplet diameters' distribution. Since it is difficult to observe a trend in droplet dimensions by taking a look at the distributions for individual videos, all droplets were analyzed as a whole. Figure 4.5 shows the diameter distribution of all droplets considered in this experiment. It is clear to note that the number of droplet occurrences diminishes with the increase in diameter.

All the histograms represented in this section used bins with 2 μm of width and went from 12 μm to 68 μm in order to cover the entire range of detected diameters.

On another note, the maximum, minimum and mean diameters obtained for all the droplets were 67.23432 μm, 12.72727 μm and 23.18243 μm respectively.

Table 4.1: Droplet Diameter Stats (in μm)

VOC	Max	Min	Mean	Median
Acetone	51.361	13.181	22.213	21.363
Acetonitrile	39.088	12.727	19.968	18.181
Chloroform	39.989	12.952	22.625	22.500
Dichloromethane	40.451	13.181	21.672	20.107
Diethylether	42.499	12.727	22.031	20.454
Ethanol	60.681	13.181	26.781	25.680
Ethylacetate	40.451	12.727	20.900	19.545
Heptane	67.234	19.771	38.220	36.818
Hexane	48.634	13.181	23.419	22.272
Methanol	33.408	13.181	18.524	17.272
Toluene	47.727	13.181	23.385	21.589

4.1.3 Train, Validation and Test Sets

Usually, in supervised learning problems, the available data is split into three sets. The training set is the set of data from which the models actually train with. This is the data fed into the neural networks during the training stage used to adjust their weights and biases. The validation set also takes part in the training stage with the role of performing a frequent evaluation on the model. The idea behind this set of data is to provide an unbiased evaluation on the model performance during training, not only to help fine tune its hyperparameters but also to avoid overfitting. If the accuracy for the training set keeps increasing but the one with respect to the validation set is not able to keep up with it, by staying the same or decreasing, it means the model is starting to learn specific features of the training set instead of more general ones. Although the networks do not learn from the validation set, they are affected by it indirectly so there is the need of a third set of data. This last set, the test one, evaluates the actual predictive skills of the final model on a series of data never 'seen' by it before. Even if a model performs really good on the training and validation sets but its outcome is not satisfactory for unseen data, it cannot be considered a success.

As stated in Section 4.1.1, for each of the 11 VOCs, there are 5 sequences that correspond to 5 exposures of the corresponding VOC. All three sets of data must contain sequences associated to exposures of the 11 VOCs. Besides that, a choice was made to not include droplet sequences from the same exposure in different sets of data. In theory, all exposures of the same VOC share the exact same characteristics so this decision would not seem necessary. However, as it is possible to observe in Figure 4.3, this does not happen in practice. From exposure to exposure there are always some slight variations, even if these are not noticed by looking at the videos at a first sight. Thus, for example, having droplet sequences in the test set from the same exposure of some other droplets that belong to the training set, could produce some misleading results. Information about that exposure would already be learnt during the training phase so the test set would not be considered completely 'unseen'. With

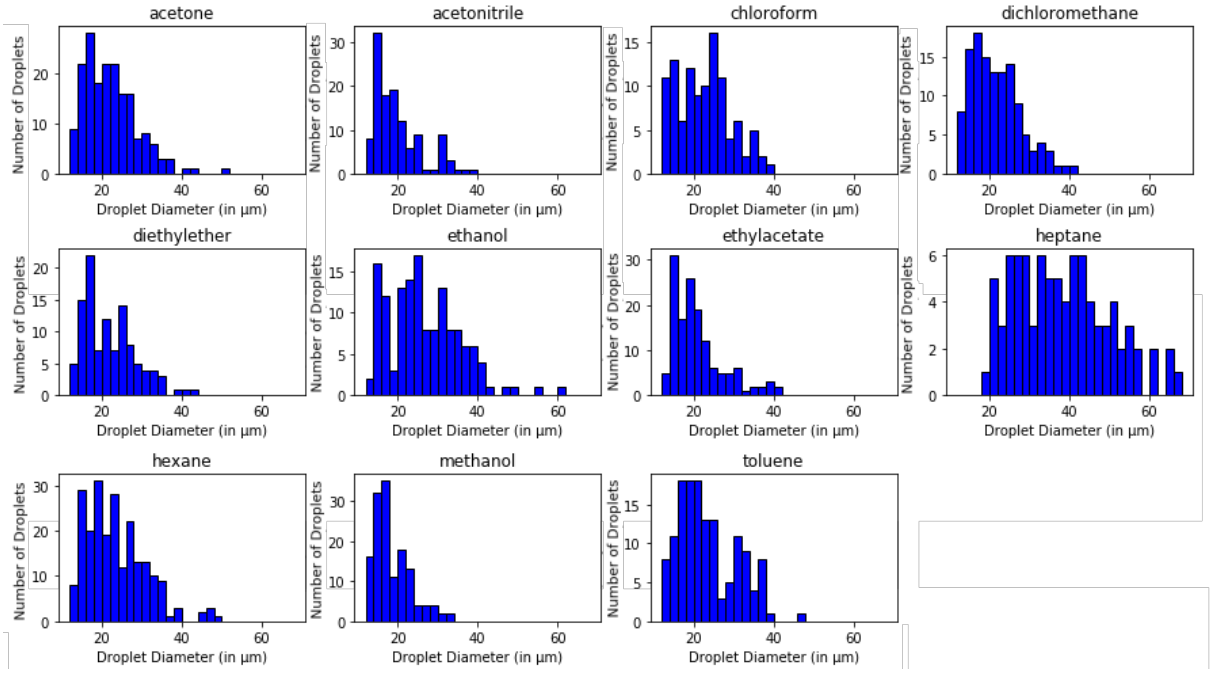


Figure 4.4: Droplet Diameters Distribution (in μm) in the gels that were exposed to each VOC

that said, from the 5 sequences per VOC, the first 3 were selected to be part of the training set, the fourth ones to the validation set and the fifth to the test set.

4.1.4 YOLO - Detection Stage

The YOLO network used to perform the droplet detection was an implementation developed by *wizyoung* made available on *GitHub* [1].

In order to train this model it is required to feed it images associated with a file that includes the coordinates of the bounding boxes that circumscribe the droplets in those same images. These bounding boxes were already taken care of, as seen in Section 3.1.2, when all the videos were annotated. The only tricky detail in all this was to exclude the frames where no droplets were present. As Figure 3.3 shows, while being exposed to a VOC, the droplets may disappear for a little while before coming back to a visible state. This selection was performed for both the training and validation sets. It is also worth mentioning that, since the only goal of the YOLO is to detect droplets, this becomes a monoclass problem where the only class is 'Droplet'.

Having the required data ready, it was necessary to set some parameters to be used in the training process. The parameters used are presented in Table 4.2.

This YOLO implementation offered an already pre-trained set of weights on the COCO dataset. These weights would be loaded into the model at the beginning of the training. Since this is an extensive

[1] https://github.com/wizyoung/YOLOv3_TensorFlow

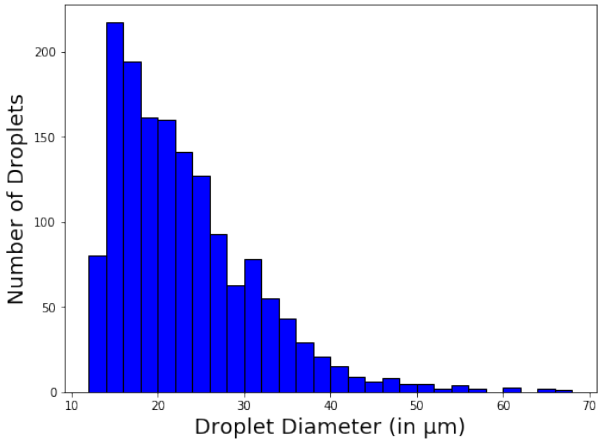


Figure 4.5: Droplet Diameters Distribution Histogram (in μm)

Table 4.2: YOLO parameters during training

Parameters	**Value**
Batch Size	5
Image Resize	416x416
Keep Aspect Ratio	True
Batch Norm Decay	0.99
L2 Weight Decay	0.0005
Optimizer	Momentum
Learning Rates	0.0001, 0.00003, 0.00001
Fine Tuning	Whole Model
NMS Threshold	0.5
mAP Threshold	0.5

dataset with hundreds of thousands of images with labeled objects of around 80 classes, this set of weights could work as a solid starting point for some object detection problems. Only fine tuning some of the top layers could be enough to adjust these weights to a specific problem and would save plenty of time in the process. However, this dataset contains images and classes from the everyday life which have nothing to do with the dataset at hand so it was decided to train the whole model. On another note, the 'Image Resize' and 'Keep Aspect Ratio' parameters refer to the dimension to which the input images are resized before being fed to the model. It was selected a smaller image size, instead of keeping the original one, in order to help with memory constraints and speed training. Due to the complexity of the model and also images sizes, it was impossible to train the model with a batch size higher than 5, which could result in some instability during this stage. The 'Batch Norm Decay', 'L2 Weight Decay', 'Optimizer' and 'Learning Rates' parameters were the ones already set by default. At the beginning, there was no motive to change these parameters and since they ended up producing good results, as it will be seen later, they remained unchanged. The three values for the learning rate correspond to different values that this parameter takes during training, starting from the highest to the lowest. The NMS and Mean

Average Precision (mAP) thresholds are parameters regarding the Intersection over Union operation applied on Non-Maximum Suppression 2.3.1.B and mean Average Precision in the validation process.

In Fig 4.6 it is possible to observe some plots regarding metrics such as Precision, Recall and mAP on both training and validation sets as well as some other ones with respect to class and bounding box losses on the validation set.

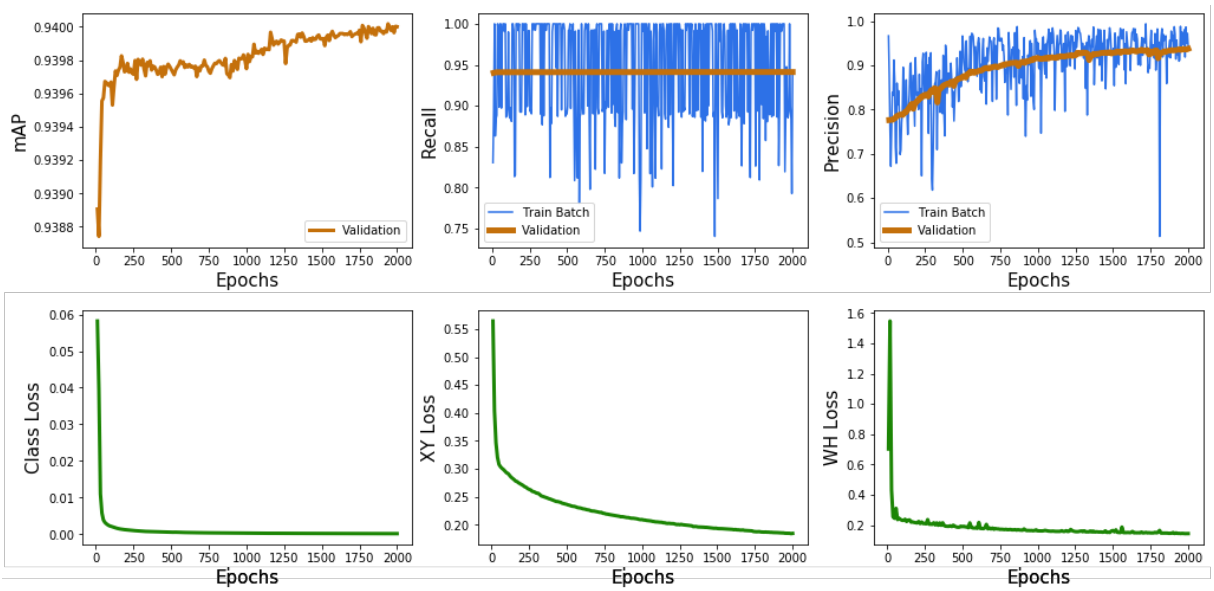


Figure 4.6: YOLO training precision, recall, mAP and losses

After 2000 epochs and around 48 hours, the training of the YOLO model was stopped. It was decided to stop it around this time since, according to the several plots in Figure 4.6, training was already stagnating. Although there were no signs of overfitting, which is clear on the Precision and Recall plots, leaving the model to train for more epochs would not have implied any significant improvement in performance. The time and computational resources that could have been spent at this stage of training would not be worth in relation to the improvements that would be verified. Regarding the upper plots, only the Precision one registered a significant improvement over time on both the training and validation sets. This may mean that the model, as training went by, corrected gradually the False Positives situation by turning them into True Positives. This must be the case since the losses regarding the bounding boxes' center coordinate (XY Loss) and width and height values (WH Loss) decreased over time. That is, the predicted bounding boxes for the validation set overlapped more and more the ground truth ones, increasing the amount of instances that achieved IoU values above the threshold (0.5) that makes the distinction between True or False positives. On another note, it is possible to observe a lot of flickering for training batches' Precision and Recall which has everything to do with the batch size. As mentioned before, due to the batches' small size (5), the weight convergence in the model is not particularly stable.

The training smoothness and high values achieved for mAP, Precision and Recall metrics on the

validation set, even on an early stage, are completely correlated with how the datasets were firstly acquired and later on split in training, validation and test. Since every VOC was only exposed to one gel region, even though the three sets do not have information regarding the same exposures, the same droplets are present in all three of them. Even if the droplets configurations change slightly over time and between exposures, the similarities between them, in different sequences, are still very notorious. This fact led to results that, most likely, would be significantly better than others achieved with a more diverse set of data.

In Figure 4.7 it is presented a frame from the test set, never 'seen' before by the model, and the YOLO detected bounding boxes on top of the ground truth ones.

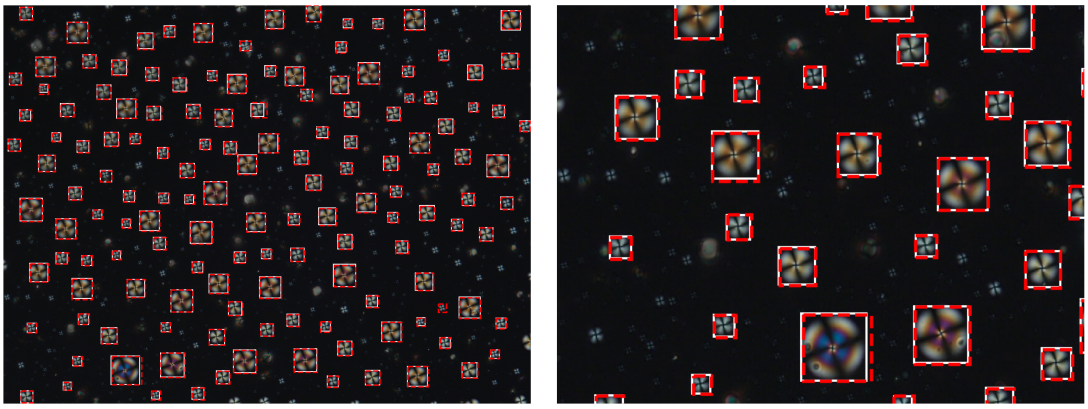


Figure 4.7: Ground truth bounding boxes (White) vs YOLO detected bounding boxes (Red) - **Left**: Whole Frame **Right**: Frame Zoomed In

4.1.5 Classification Stage

4.1.5.A CNN2D (Baseline Model 1) + Voting System

For this first experiment, a very naive and simple method was purposefully executed, where the timely component of a droplet sequence was completely ignored. The idea behind this simple model was to achieve a set of results that, in theory, would not be great and would work as a baseline. The trained model had the goal of associating each frame of a droplet sequence to one of the 11 VOCs, not taking into consideration how a droplet evolved over time, but only its current state at a given stage. Since, when exposed to the VOCs, the droplets evolve to a state where they disappear, the frames where that is verified had to be withdrawn for consideration for the training, validation and testing phases.

The model used to classify the droplets was the one described in Section 3.4.1. This model takes as input an image with fixed dimensions and since there are droplets with very different sizes in the dataset, as observed in the analysis done in Section 4.1.2, they all had to be resized to a specific one. In theory, by looking at the different videos, the bigger droplets are the ones that offer a more detailed

Table 4.3: CNN2D Models Parameters

Parameters	Values
Number of Filters (Conv1, Conv2)	(16,32), (32,64), (64,128)
Filter Dimensions Conv Layers	10x10, 15x15, 20x20
Stride Conv Layers	1
Filter Dimensions Max-Pool Layers	2x2
Stride Max-Pool Layers	2
Dropout Rate	0.2, 0.35, 0.5
Units FCL 1	32
Units FCL 2	11

and clear evolution over time, so they should achieve better results compared with the smaller ones. Taking this assumption into consideration, and in order not to lose quality on these bigger droplets, all of them were resized to 148x148 pixels, which is the dimension of the biggest one detected in the dataset. Until mentioned otherwise, the experiments that follow also use this same droplet resize.

During the training stage, while some parameters changed between models, others remained unaltered. It was decided to use 2x2 filters with stride 2 for both Max-Pooling layers on all models, as well as a 32 unit Fully Connected layer before the last one. Table 4.3 presents all the relevant network parameters. Since there are three parameters that change between models, and 3 values were picked for each one of them, a total of 27 instances of the same model were trained.

All models were trained with the Adam optimizer and Categorical Crossentropy as the loss function. No epoch limit was set for training since early stopping was used. After 5 epochs, if no improvement on the validation loss is registered, training is ceased so the models do not get too adjusted to the training set.

In order to choose the model that outperforms all the other ones, the macro F1 score was chosen. This metric consists on the average F1 score achieved for the 11 classes, that already takes into consideration their Precision and Recall. Table 4.4 presents the Precision, Recall, F1-scores per class for the best model on the validation set. This model had 64 and 128 filters of dimension 10x10 for the first and second convolutional layer. The dropout rate set for this model was 0.35. It was possible to achieve an overall accuracy of 97.4% for the droplets on the validation set. It is important to mention that no voting system was implemented at this stage. Besides that, no class underachieved the others with all of them registering a F1-score higher than 94%.

In order to really evaluate the accuracy of the model, it was tested on the test set. The strategy described in Section 3.3 was used to assemble the droplet sequences. However, since this is an approach where the classification is made frame by frame, the ones that presented a colour level below a certain threshold (a third of the sequence maximum pixel colour sum) were again discarded and not fed to the model. These correspond to the frames where the droplets disappear and the image is almost entirely black. The remaining ones were fed to the trained model and the class with the most votes within a

Table 4.4: Precision, Recall and F1-Score per class on the best CNN2D model (Baseline Model 1) per droplet frame - Validation Set

VOC	PRECISION	RECALL	F1-SCORE
Acetone	0.9661	0.9500	0.9580
Acetonitrile	0.9704	0.9724	0.9714
Chloroform	0.9991	0.9507	0.9743
Dichloromethane	0.9452	0.9501	0.9477
Diethylether	0.9831	0.9784	0.9808
Ethanol	0.9540	0.9862	0.9698
Ethylacetate	0.9332	0.9756	0.9539
Heptane	0.9986	0.9995	0.9990
Hexane	0.9829	0.9871	0.9850
Methanol	0.9930	0.9976	0.9953
Toluene	0.9917	0.9589	0.9750
AVERAGE	0.9743	0.9733	0.9737

Table 4.5: Precision, Recall and F1-Score per class on the best CNN2D model (Baseline Model 1) before and after majority voting - Test Set

	PRECISION		RECALL		F1-SCORE	
VOC	BEFORE	AFTER	BEFORE	AFTER	BEFORE	AFTER
Acetone	0.8795	0.9615	0.7054	0.7911	0.7829	0.8681
Acetonitrile	0.8152	0.8611	0.6948	0.7686	0.7502	0.8122
Chloroform	0.9406	0.9703	0.8100	0.9074	0.8704	0.9378
Dichloromethane	0.4150	0.4241	0.8379	0.8790	0.5551	0.5722
Diethylether	0.6160	0.6776	0.8878	0.9450	0.7273	0.7893
Ethanol	0.7607	0.7676	0.9310	0.9820	0.8373	0.8617
Ethylacetate	0.8492	1.0000	0.0487	0.0423	0.0920	0.0811
Heptane	0.9535	0.9556	0.9795	0.9773	0.9664	0.9663
Hexane	0.9370	0.9747	0.9686	0.9847	0.9525	0.9797
Methanol	0.4808	0.5526	0.9667	1.0000	0.6422	0.7119
Toluene	0.5819	0.5802	0.3667	0.3310	0.4499	0.4215
AVERAGE	0.7481	0.7932	0.7452	0.7826	0.6933	0.7274

sequence was assigned to it.

Table 4.5 presents the Precision, Recall and F1-score values before the application of the majority voting system (all droplet frames independently) and after. Looking back at the results achieved on the validation set where the F1-scores with respect to all 11 classes exceeded 94%, the same model completely under-performed on the test set. Only the Heptane and Hexane classes kept their F1-scores above this value, whereas for Toluene and Ethylacetate not even the 50% mark is reached. For the latter, since there are 11 classes in total, a completely random guess would have a higher chance of getting the correct prediction, which shows the completely inability of this model to distinguish this class from the others. After applying the majority voting system within a sequence it is possible to observe a slight improvement in almost every class, except for the two worst ones already mentioned. In these cases, most of the correctly predicted frames disappear as a result of the majority voting system, hence this

drop. For the remaining classes the opposite phenomenon occurs. In Figure 4.8 it is shown a graphical representation of the confusion matrix regarding the final class assignments of all sequences after the polling. The major take outs from this matrix are that the Ethylacetate and Toluene sequences are most of the time wrongly classified as Dichloromethane. Even though this happens, Dichloromethane sequences are correctly labeled 88% of the times, they are never confused with Ethylacetate and only four times with Toluene. Besides that, Ethylacetate sequences are also labeled as Toluene ones 17% of the times. Looking at the droplets on the these three sequences, the resemblance is clear between them. Not taking into consideration how they evolve over time and only considering their instantaneous states should be the reason behind this mistakes. Overall, the registered accuracies before and after the majority voting system were 72.0% and 75.4%.

One detail worth mentioning is that 70% of the first frames of every sequence were correctly labeled. On the first frame of every sequence, the exposure to the VOCs has not really started so, in theory, it should be impossible to determine the VOC associated to it. The first frame accuracy should be around 9% given the fact that 11 VOCs are being considered. This is a consequence of the fact that every exposure of the same VOC was performed on the same exact gel region, that contains the exact same droplets. The model has learned quite well how to associate certain droplets to certain classes, when the only thing that should matter is how they evolve over time. In a dataset with plenty of exposures of the same VOC to always different gel regions it would be impossible to associate a droplet with a VOC. This is what should happen since the VOCs are not correlated at all with the droplets to which they are being exposed to and they should not have any influence on the final results.

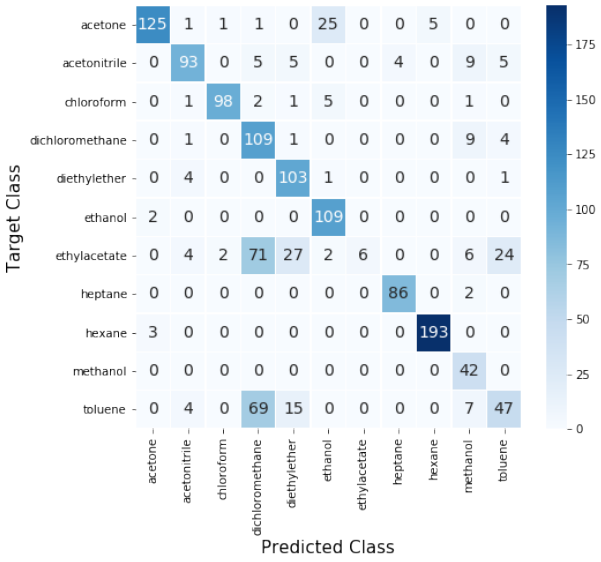


Figure 4.8: Confusion matrix on prediction results after voting - CNN2D (Baseline Model 1) + Voting System - Test Set

At last, an analysis was made on the impact of droplet size on the accuracy since this is also an

important goal of this Thesis. Figure 4.9 shows an histogram with the diameters of the detected droplet sequences as well as the their accuracy. Each bin's color is related to the accuracy obtained for the droplets that fall into that diameter range. The trend is clear, with the increase in diameter there is also an increase in accuracy. For smaller droplets (12μm-22μm), accuracies around 60% were obtained and, by going up in diameter, it is possible to achieve accuracies around 100%. Even though some of the last bins only represent the accuracy of only one or two droplet sequences, by taking a look at the 62 sequences with the highest diameters (42μm-70μm), only two of them were wrongly predicted, resulting in a 97% accuracy.

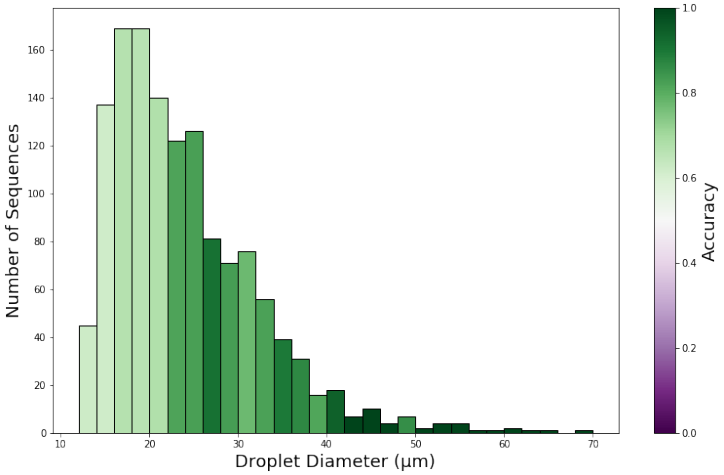


Figure 4.9: Droplet Sequence Accuracy by Diameter - CNN2D (Baseline Model 1) + Voting System - Test Set

4.1.5.B CNN3D (Baseline Model 1)

While on the previous section a 2 dimensional CNN was used to try to associate each frame on a droplet sequence to one of the 11 VOCs, now, the entire sequence is considered for prediction. With this approach, the droplet evolution over time is already taken into consideration by the model. Now, instead of training models with 2D images, 3D images are used where the 3rd dimension is associated to time. Architecturally-wise, there are not significant changes between the 2D and 3D versions. Since now the input data have a 148x148x3x32 dimension, to accommodate this change, both the convolutional and max-pooling layers use kernels that extend to the third dimension. Regarding the fully connected layers, since the data that comes out of the second max-pooling layer is flattened, there is no need for change.

Regarding training itself, again, several combinations of parameters were tried. The goal was to apply the same set of parameters used for the 2 dimensional CNN, and train again 27 instances of the same model (Table 4.3). However, given that the computations, memory required and time needed to train the 3 dimensional CNN were way more expensive than before this was not done. Instead, only 9 models were trained. Table 4.6 shows the parameters used on these models. With respect to the

Table 4.6: CNN3D (Baseline Model 1) Parameters

Parameters	Values
Number of Filters (Conv1, Conv2)	(16,32)
Filter Dimensions Conv Layers	10x10x10, 10x15x15, 10x20x20
Stride Conv Layers	1
Filter Dimensions Max-Pool Layers	2x2x2
Stride Max-Pool Layers	2
Dropout Rate	0.2, 0.35, 0.5
Units FCL 1	32
Units FCL 2	11

Table 4.7: Precision, Recall and F1-Score per class of the best CNN3D model (Baseline Model 1) - Validation Set

VOC	PRECISION	RECALL	F1-SCORE
Acetone	1.0000	1.0000	1.0000
Acetonitrile	0.8181	0.8181	0.8181
Chloroform	1.0000	0.9907	0.9953
Dichloromethane	0.9212	0.9435	0.9322
Diethylether	1.0000	1.0000	1.0000
Ethanol	0.9652	1.0000	0.9823
Ethylacetate	0.9930	1.0000	0.9964
Heptane	1.0000	1.0000	1.0000
Hexane	0.9955	0.9955	0.9955
Methanol	0.9772	0.9555	0.9662
Toluene	0.9185	0.8794	0.8985
AVERAGE	0.9626	0.9620	0.9622

optimizer, loss function and early stopping, everything was kept the same.

The model which achieved the best macro F1-score (96.2%) used 16 and 32 filters of dimension 10x20x20 on the first and second Convolutional layers, respectively, and a 0.2 dropout rate. Table 4.7 presents the metric stats obtained by this model on the predictions made on the validation set. It shows out maximum accuracy when it comes to classify Acetone, Diethylether, Ethanol, Ethylacetate and Heptane sequences and almost perfect accuracies for Chloroform and Hexane.

The results on the test set, obtained through the YOLO detections, as done before, can be checked in Table 4.8 in the form of Precision, Recall and F1-Score or in Figure 4.10 as a confusion matrix. Then again, even though the results on the validation test were next to perfect, the same was not verified for the test set. The first thing that comes to sight, and that was not observed for the 2 dimensional CNN plus majority voting strategy, is that for some classes (Acetone, Heptane and Diethylether), the predictions for their sequences were 100 % correct. Besides that, the first two mentioned VOCs also achieved 100% precision. This shows that this model has achieved the ability of perfectly distinguishing these classes' sequences from all the other ones. On the other hand, the performance on Dichloromethane and Ethylacetate was very poor, not even achieving a 50% recall. The Ethylacetate class is still being confused very often with Dichloromethane but, curiously, with the insertion of the time component in

Table 4.8: Precision, Recall and F1-Score per class of the best CNN3D model (Baseline Model 1) - Test Set

VOC	PRECISION	RECALL	F1-SCORE
Acetone	1.0000	1.0000	1.0000
Acetonitrile	0.5000	0.6611	0.5693
Chloroform	1.0000	0.6759	0.8066
Dichloromethane	0.1582	0.1774	0.1673
Diethylether	0.7801	1.0000	0.8764
Ethanol	0.9478	0.9819	0.9646
Ethylacetate	1.0000	0.3028	0.4648
Heptane	1.0000	1.0000	1.0000
Hexane	0.9609	0.9949	0.9776
Methanol	1.0000	0.9069	0.9512
Toluene	0.3989	0.5177	0.4506
AVERAGE	0.7951	0.7471	0.7480

the data, the Toluene class is not mistaken anymore by Dichloromethane but with Acetonitrile instead. In addition to this, Dichloromethane that once achieved very high recall values, is now being heavily confused for Toluene. Adding the third dimension to the data originated different mixes in predictions but still was not enough to achieve a satisfactory level, given the lack of ability to correctly classify sequences of plenty of classes.

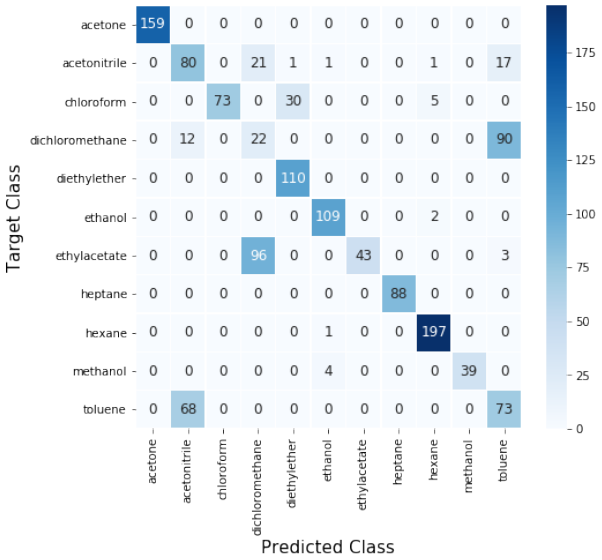


Figure 4.10: Confusion matrix on prediction results - CNN3D (Baseline Model 1)- Test Set

Even though there were significant changes in results among classes, with some achieving better results with the 3D strategy and others worse, as it is possible to verify in Figure 4.11, the same relation between droplet size and accuracy was observed. It is easier to detect the exposed VOC through bigger droplets than smaller ones.

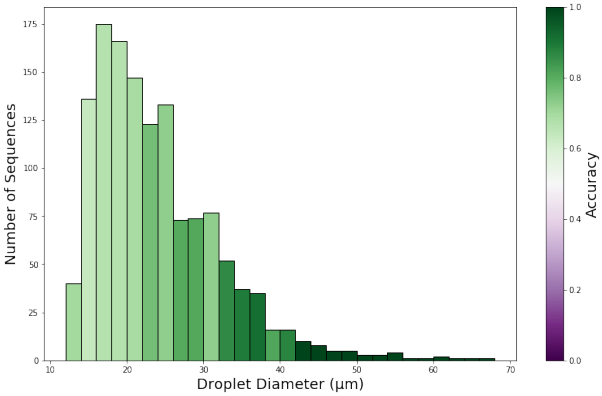


Figure 4.11: Droplet Sequence Accuracy by Diameter - CNN3D (Baseline Model 1) - Test Set

4.1.5.C CNN3D - Recall based weights

Imbalanced data can be an issue in some classification problems. This happens when the classes present in a problem are not equally represented in the dataset. This can lead to a model not picking up enough information about the under-represented classes and then not being able to correctly classify them. To check whether this situation was directly affecting the results, a plot with respect to the recalls achieved on the validation set, for the 11 classes, by the 9 trained models, was made on top of a bar chart representing the number of sequences of each class present in the training set. This plot can be observed in Figure 4.12.

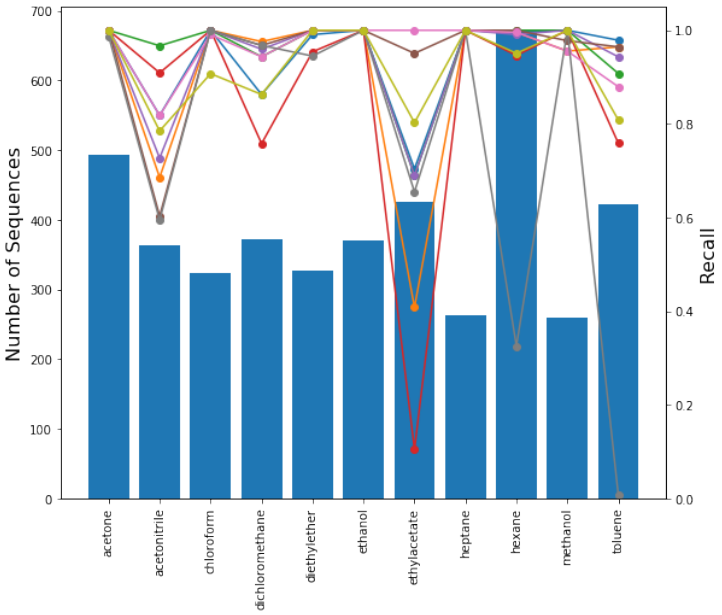


Figure 4.12: Recall vs Number of Sequences (9 trained CNN3D (Baseline Model 1) models)

According to Figure 4.12, although the classes are not balanced, that does not seem to really affect

Table 4.9: Original and Weighted models F1-scores

	Original	Weighted
Model_0	0.9398	0.9371
Model_1	0.9019	0.8983
Model_2	0.9530	0.8741
Model_3	0.8332	0.9053
Model_4	0.9335	0.8699
Model_5	0.9513	0.9198
Model_6	0.9622	0.8211
Model_7	0.7216	NOT TRAINED
Model_8	0.9173	0.7923

the achieved recalls. Classes like Acetone or Hexane, that possess the two highest number of sequences in the dataset, get very accurate predictions by every model instance. The same goes to other classes that do not get that much representation like Heptane and Methanol. So it does not seem like the discrepancy in the number of sequences per class is playing a direct role on the bad performance of some classes.

One of the tactics commonly used to combat the effect of unbalanced datasets on the performance of neural networks is to assign weights to each class. During training, when a model makes a mistake on a class assigned with a higher weight, it is also 'punished' on a more severe way by performing a more substantial adjustment to the model weights. Thus, classes with less representation in the dataset get assigned bigger weights. Given that the imbalance in the data was not a significant issue when it came to class accuracy, this strategy was not applied. Instead, these class weights were applied based on the registered recall values. The idea behind this was to assign higher weights to classes that under-performed, in order to give them more relevance during training and by doing so, increasing their accuracy. The weights were designated to each class according to Equation 4.1. All 9 previously trained 3 Dimensional Convolutional Neural Network (CNN3D) models, except one, were trained again from scratch using these weights. Since one of the original models achieved only a 0.7% recall on the Toluene class, the weights assigned to the classes would be completely disproportional, so that instance of the model was not trained.

$$W_i = \frac{\sum_{j=FirstClass}^{LastClass} Rec_j}{11 * Rec_i}, i \in Classes \tag{4.1}$$

Table 4.9 presents the F1-scores obtained for the same models, in their original implementations and after using weighted classes. Overall the results were very disappointing since only one out of the eight newly trained models achieved a higher F1-score than its original version and it didn't surpass the highest F1-score on the validation set to date.

4.1.5.D Stacking Ensemble

Until this moment, the most consistent model achieved a 96.2% F1-score on the validation set. This value is obtained by computing the average F1-score on the 11 classes. Since this value is only an average, it does not mean that this model outperforms all others in every class. In fact, the model with the best overall F1-score is outperformed by other trained models in some classes. By only picking the best overall model to predict on 'unseen' data, strong qualities from the remaining ones are being put to waste.

Stacking is an ensembling technique where an extra model is trained on the combinations of predictions of 2 or more base models. By doing so, this model is trying to put the skills of all these models together in order to provide predictions with higher accuracies than the ones given by each one of the base models alone. Figure 4.13 presents a schematic of this stacking process.

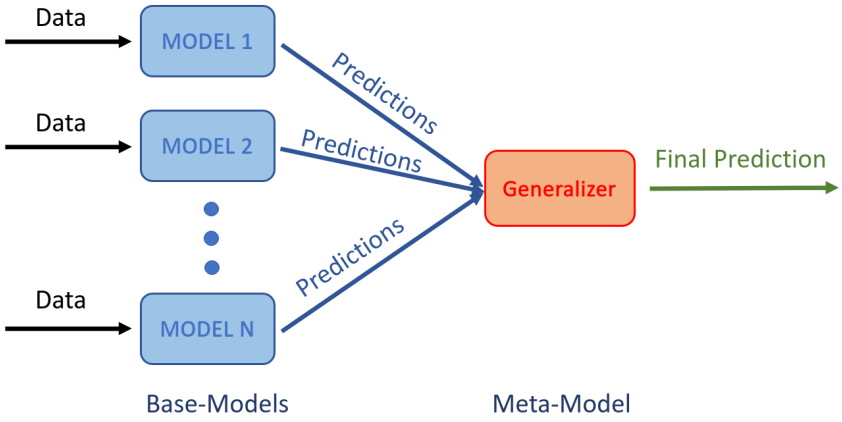


Figure 4.13: Stacking Scheme

The base models used in the stacking process were the 9 originally trained ones (Section 4.1.5.B). Given that these models already achieved very high accuracies for the training set, training the meta model on the predictions for this set of data would not bring any improvements. The ideal situation would be to train the meta model on the predictions obtained on a completely new dataset so the base models would not be biased at all. With this, the meta model would learn how to interpret 'pure' predictions which is what happens when the whole system is fed with the test set. Since the models are completely adapted to the training set and the test set needs to be kept completely unseen until the last stage, the meta-model was trained on the validation set. The *sklearn* library on *Python* offers a wide variety of different classifiers suitable for multiclass problems. Several different classifiers were tested as meta-models (Logistic Regression, Decision Tree, Gaussian Naive Bayes, Support Vector Machine, Multi-layer Perceptron, etc..). All these algorithms were used with their default parameters. Since the validation set is composed of only one exposure per VOC, the cross-validation method was used to estimate the skill of all these models in order to chose the best one to be used on the test set. The validation data was

Table 4.10: Precision, Recall and F1-Score per class resultant of CNN3D (Baseline Model 1) models Stacking Ensemble - Test Set

VOC	PRECISION	RECALL	F1-SCORE
Acetone	1.0000	1.0000	1.0000
Acetonitrile	0.5958	0.9504	0.7324
Chloroform	1.0000	1.0000	1.0000
Dichloromethane	0.3737	0.6209	0.4666
Diethylether	1.0000	1.0000	1.0000
Ethanol	1.0000	0.9459	0.9722
Ethylacetate	1.0000	0.1056	0.1910
Heptane	1.0000	1.0000	1.0000
Hexane	1.0000	1.0000	1.0000
Methanol	1.0000	1.0000	1.0000
Toluene	0.9666	0.8226	0.8888
AVERAGE	0.9033	0.8586	0.8410

split into 5 folds and the metric chosen to evaluate the performance of each classifier was again the macro F1-score. The Gaussian Naive Bayes classifier outperformed all the other ones by achieving an average macro F1-score of 98.4% which was not far from the others that achieved values between 93% and 97%.

The results achieved by the final stacked model (using the Gaussian Naive Bayes classifier as meta-model) on the YOLO detections performed on the test set can be observed in Table 4.10 as well as the corresponding confusion matrix on Figure 4.14.

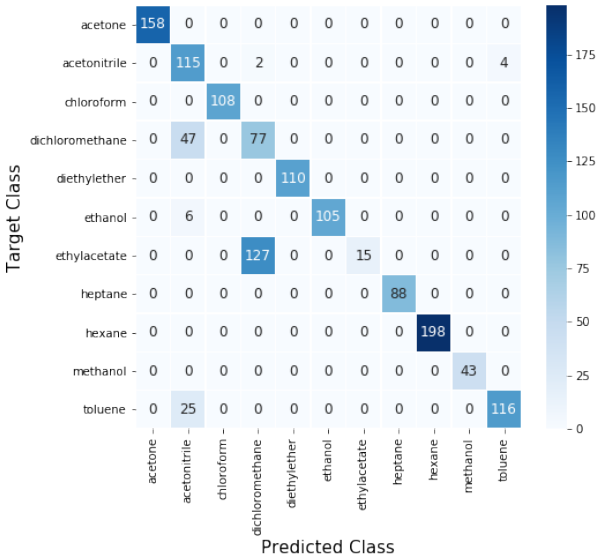


Figure 4.14: Confusion matrix on prediction results of CNN3D (Baseline Model 1) models Stacking Ensemble - Test Set

According to the data in Table 4.10, the macro F1-score achieved with the stacking technique was 84.1%. By using model stacking, it was possible to improve this value by 9.3%, comparing to the one

achieved by the best single model. The classes that already achieved excellent results did not drop in performance and the remaining ones, that still had some room to improve, with exception to Ethylacetate, registered some significant increases in F1-score, that helped boost the overall score. Acetonitrile, Chloroform, Dichloromethane and Toluene were the most influential ones in this overall score boost by improving their respective F1-scores by 16.3%, 19.3%, 30.0% and 43,8%, respectively. On the other hand, the Ethylacetate class registered a 27.38% drop in F1-score due to an extreme incapacity of correctly predicting this classes' sequences. By analyzing this confusion matrix and the one in Figure 4.10 it can be observed that Acetonitrile sequences are no longer being confused for Dichloromethane or Toluene (only 1 time now for each), Chloroform sequences are no longer confused for Diethyleter, now 77 Dichloromethane sequences are correctly predicted instead of only 22 (confusion with Toluene drops to zero and it is now being confused with Acetonitrile), and lots of Toluene sequences that used to get wrongfully classified as Acetonitrile (68) are now being correctly classified.

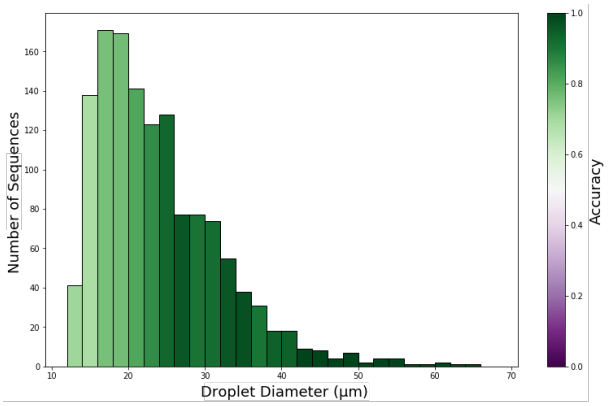


Figure 4.15: Droplet Sequence Accuracy by Diameter - CNN3D (Baseline Model 1) models Stacking Ensemble - Test Set

As for the accuracy by droplet size, Figure 4.15, the trend remained the same as expected but due to the increase in overall performance it is possible to observe higher accuracies for smaller droplets.

4.1.5.E One vs Rest (OvR)

One vs Rest (OvR) and One vs One (OvO) are two techniques used to turn a multiclass classification problem into several binary classification ones. Since there are some algorithms that do not support classification tasks with more than two classes, these strategies allow them to be used by breaking these problems into several binary ones. In OvR, each binary classifier used is associated to one class and it is trained to identify if a certain data sample belongs or not to that class. The data samples associated to that class are labelled positively (1) and the ones associated to any other class are labelled negatively (0). To predict the class of a certain data sample, it is fed to all base models and the one that produces the higher confidence score regarding its class, labels the sample. In this scenario, since there

Table 4.11: OvR models Recall per class on Validation set

	Recall	
	1st Round	**2nd Round**
Acetone	1.0000	-
Acetonitrile	0.8842	0.8181
Chloroform	1.0000	-
Dichloromethane	0.7500	0.7661
Diethylether	1.0000	**-**
Ethanol	1.0000	-
Ethylacetate	0.1126	0.5985
Heptane	1.0000	-
Hexane	1.0000	-
Methanol	1.0000	-
Toluene	0.7021	1.0000

are 11 VOCs/classes, 11 models would have to be trained. In OvO, each binary classifier is in charge of distinguishing data between two classes. Thus, in total, C(C-1)/2 base models are needed, with C being the total number of classes. For a 11 class problem as this one, 55 models would be needed. Again, a data sample is fed to all base models and, at the end, the class predicted the most times from those models is the one assigned to the sample. Even though CNNs support multiclass classification problems, they were used in this context to check if by specializing models in a specific class, the final result would be better. Since for a OvO approach 55 models would have to be trained, only the OvR strategy was executed.

The architecture chosen for the 11 models was based on the 9 originally trained CNN3D models. For each class, a new model was trained with the parameters of the model instance that achieved a better Recall value for that same class. In case of a tie, the Precision/F1-score settled it. Since now every model is solving a binary problem, the chosen loss was changed to the binary crossentropy. Besides that, since now there are only two classes per model and one of them includes 10 of the 11 VOCs, the dataset is completely unbalanced so some weights were assigned to both classes based on their number of sequences. After a 1st round of training where a few models underachieved on the validation set, these same ones were trained again with a new set of parameters [2] to try and improve on the previous values. Table 4.11 presents the recall values achieved on the validation set by the 11 trained models on the 1st round and the 4 on the 2nd round.

By analyzing Table 4.11, it is possible to conclude that, at least for this model architecture and parameters, turning the problem into various binary ones did not really improve the performance of each class. Looking back at the results achieved by the originally trained models in Section 4.1.5.B, there is at least one per class that achieves an equal or higher Recall value also in the validation set. By combining all these models with the detections performed by the YOLO network, it was only possible to achieve a

[2] Table 4.6: Conv Filters: 10x10x10, Dropout: 0.2

74.5 % macro F1-score. Given that the best original CNN3D model achieved a 74.8% F1-score alone (Table 4.8) and the stacked strategy achieved an 84.1% F1-score (Table 4.10), this score obtained with the OvR strategy was pretty disappointing.

4.1.5.F CNN3D (Baseline Model 1) Smaller Input Resize

One of the conclusions taken from the already done experiments was that the accuracy was highly dependent on the size of the droplet. The bigger the droplet, the higher are the chances of making a correct prediction. One of the reasons that could be causing this dependence is that for every model tested until this moment, all droplets were being resized to the size of the biggest droplet. This might be causing smaller droplets to get blurred after the resize and because of that not getting good enough results. Due to this assumption, some new models were trained where all droplets were resized to 75x75 pixels instead of 148x148 pixels. Besides trying to check if the relation between droplet size and accuracy changes, and by taking advantage of the fact that the data size is now about one fourth the size of the original one, lots of new models were trained to try and obtain one with an even higher macro F1-score.

An exhaustive search for an even better model was undergone, by training a total of 50 new CNN3D networks. In some models a third set of convolutional and max-pooling layers was added, for others the third dimension of the convolution kernel was changed, others were trained with max-pooling layers that used 3x3x3 and stride 3 filters instead of the original 2x2x2 and stride 2, etc.. Even though lots of different models were trained and even by performing slight changes on the original network that were not done before, the maximum F1-score achieved on the validation set was 95.0 % which is still 1.2 % shy of the best model achieved with the 148x148 pixel droplets (Table 4.7).

Since until this moment, the best results achieved for the test set were obtained through the use of the stacking system, the same was executed here, but this time with the meta-model making its prediction based on all predictions from the 50 different models. The meta model trained was once again a Gaussian Naive Bayes classifier. Using the detections on the test set made by the YOLO network, this system achieved a 81.6% macro F1-score. It is once again lower than the value achieved for the original ensemble of 84.1%. Figure 4.16 presents the confusion matrix resultant of this system predictions.

Comparing this results with the ones obtained in Section 4.1.5.D, it is possible to observe that they are pretty similar except for the Dichloromethane and Toluene classes. While for the original stacking experiment the Dichloromethane class only got correct 77 sequences, now 109 are correctly predicted. On the other hand, the Toluene class that previously achieved almost perfect results has now taken a serious dip in accuracy. The Ethylacetate class decreased even more its performance. It is the one pulling down massively the overall system results by only getting correct 5 out of 142 sequences.

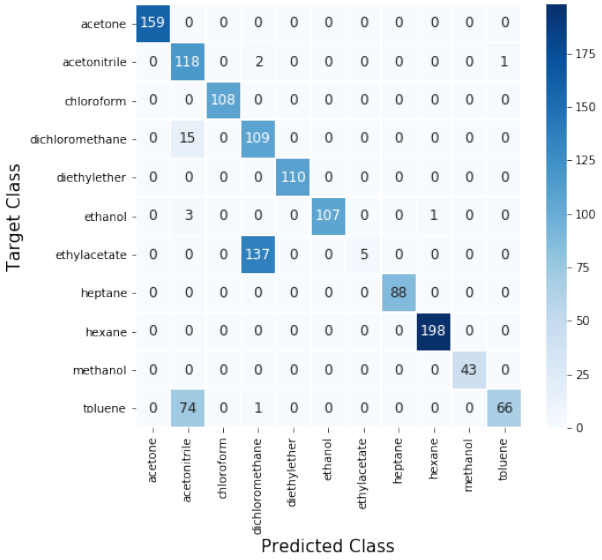


Figure 4.16: Confusion matrix on prediction results of 50 CNN3D (Baseline Model 1) models with smaller input data Stacking Ensemble - Test Set

Figure 4.17 represents once again the accuracy in function of the droplet diameter. Although the trend is not as clear as displayed before on previous histograms, it is still there and the bigger droplets keep producing more accurate results than the smaller ones. Thus, it is believed that the size to which the droplet sequences are resized does not have a major impact on the overall predictions. The differences are not significant but since computationally-wise it is much more beneficial training the models with smaller data, every experiment that follows will resize the droplets to 75x75 pixels.

4.1.5.G CNN2D (Baseline Model 1) + LSTM

This is the first experiment using a LSTM. Since every droplet sequence has 32 frames, the LSTM is composed by 32 cells. Before every cell comes a CNN2D and in this experiment the Baseline Model 1 was used. The data goes through two sets of convolutional and max-pooling layers before being flattened and fed into the LSTM cells. At the final end of the architecture there is a 11 unit softmax layer that produces the predictions based on the values put out by the last cell of the LSTM.

The parameters set for this model are presented in Table 4.12.

Given the variations in some of the parameters and the fact that each parameter conjugation is tested for both a unidirectional and bidirectional LSTM, a total of 12 different instances of the same model were trained. All of them were able to achieve macro F1-scores on the validation set over 92% with the best one achieving a 98.7% which is the best one recorded so far for a single model. This same model was able to achieve a 88.8% score on the predictions made by the YOLO network on the test set. This result again outperformed the best CNN3D model as well as the corresponding stacking ensemble. At this

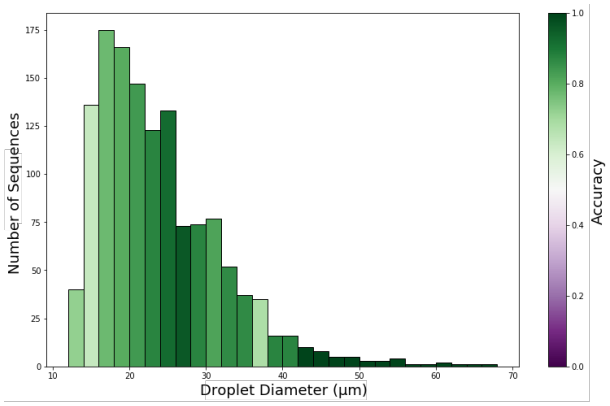


Figure 4.17: Droplet Sequence Accuracy by Diameter - 50 CNN3D (Baseline Model 1) models with smaller input data Stacking Ensemble - Test Set

Table 4.12: CNN2D + LSTM Model Parameters

Parameters	Values
Number of Filters (Conv1, Conv2)	(8, 16)
Filter Dimensions Conv Layers	3x3, 5x5, 7x7
Stride Conv Layers	1
Filter Dimensions Max-Pool Layers	2x2
Stride Max-Pool Layers	2
Dropout Rate	0.5
LSTM Hidden State Dimension	16, 32
Units Softmax Layer	11

moment in time it is already clear to see that the LSTM models are able to produce significantly better results, for this specific problem, than the 3 dimensional CNN ones. It is difficult to find an exact reason for why this LSTM network is able to achieve significantly better results than all the tested CNN3D models. At first, it was thought that this could be related to the number of parameters used in both architectures since the best CNN3D model had 997803 trainable parameters and the best LSTM one, used in this section, had 595803 so there could be more room for the CNN3D model to overfit. However, lots of CNN3D models with a number of parameters around the low hundred thousands were also trained and they simply did not deliver. This led to believe that it was the different way of approaching the third dimension, by these LSTM models, that was more suitable to this specific task, even though it is difficult to be completely sure.

The 12 models were put together into a stacking ensemble. This time, the meta-model that achieved the best macro F1-score for the validation set, when using the cross-validation technique, was the Multi-layer perceptron with a value above 99%. The outcome of this model when applied to the test set data can be found on both Table 4.13 and Figure 4.18.

This system was able to reach a 91.0 % macro F1-score which is an all-time high. The predictions for every class, except for the Ethylacetate, were very close to perfection. This odd class is constantly

Table 4.13: Precision, Recall and F1-Score per class resultant of CNN2D (Baseline Model 1) + LSTM Stacking Ensemble - Test Set

VOC	PRECISION	RECALL	F1-SCORE
Acetone	1.0000	1.0000	1.0000
Acetonitrile	0.8394	0.9504	0.8914
Chloroform	1.0000	1.0000	1.0000
Dichloromethane	0.6185	0.9677	0.7547
Diethylether	0.8503	0.9818	0.9113
Ethanol	0.9909	0.9909	0.9909
Ethylacetate	1.0000	0.3661	0.5360
Heptane	1.0000	1.0000	1.0000
Hexane	1.0000	1.0000	1.0000
Methanol	0.9772	1.0000	0.9885
Toluene	0.9842	0.8865	0.9328
AVERAGE	0.9328	0.9221	0.9096

being mistaken for Dichloromethane and Diethylether, not even getting 50% of its predictions correct. This is a recurrent problem from previous experiments that keeps pushing down the overall score. On a more positive note it was still able to increase its respective recall value by 26.1 %.

By analyzing Figure 4.19, it is possible to conclude that the increase in performance was clearly due to a significant increase in predictive power on the droplets in the mid area of the size spectrum. The smaller droplets were still achieving lower accuracies when compared to the bigger ones and the only class massively under-performing was the Ethylacetate. By taking a look at Figure 4.4, at the Ethylacetate histogram, it is possible to relate these facts. The gel to which the Ethylacetate was exposed to is composed mostly by droplets on the lowest end of the size spectrum. Nevertheless, other VOCs were also exposed to droplets with very small sizes so this cannot be the only reason to cause this phenomenon.

4.1.5.H CNN2D (Baseline Model 2) + LSTM

Before going in detail into the new LSTM experiment, it is worth mentioning that a CNN3D version of Baseline Model 2 was implemented and trained for several sets of parameters as done before for other experiments. The results were pretty disappointing with the best model not even surpassing the 85% F1-score mark on the validation set and with some others that were not capable of making any distinctions between classes, predicting the same VOC for every droplet sequence. Thus, no section was fully dedicated to this implementation.

Now onto this Section's experiment, it consists of the same one presented in Section 4.1.5.G, but instead of using the two dimensional CNN based on the LeNet-5 model (Baseline Model 1) as feature extractor, the one described in Section 3.4.2 was used. Before feeding the data to the actual LSTM, it goes through the three sets of two convolutional layers followed by a max pool layer. Every other aspect

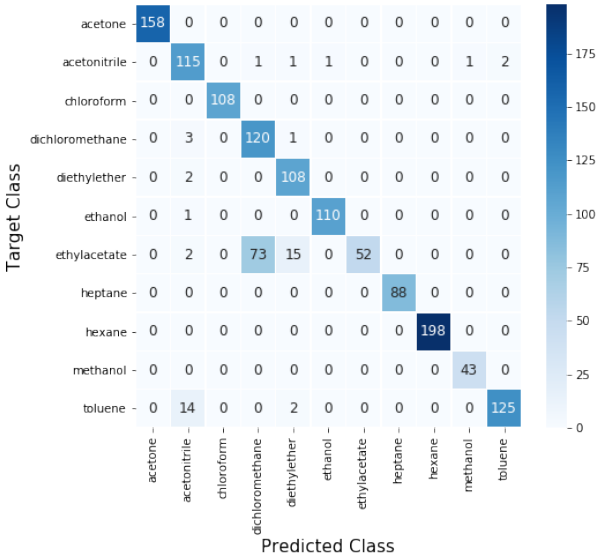


Figure 4.18: Confusion matrix on prediction results of CNN2D (Baseline Model 1) + LSTM models Stacking Ensemble - Test Set

Table 4.14: CNN2D + LSTM Model Parameters

Parameters	Values
Number of Filters (Conv1&2, Conv3&4, Conv5&6)	(8, 16, 32)
Filter Dimensions Conv Layers	3x3, 5x5, 7x7
Stride Conv Layers	1
Filter Dimensions Max-Pool Layers	2x2
Stride Max-Pool Layers	2
Dropout Rate	0.5
LSTM cell units	16, 32
Units Softmax Layer	11

regarding the architecture remained unaltered. The parameters used for this model can be found in Table 4.14.

Only the filter dimensions in the convolutional layers and the number of units per cell in the LSTM are not fixed parameters. Besides that, since every set of parameters is trained again for both an unidirectional and bidirectional LSTM, 12 different instances of the same model were trained.

The model that achieved the best F1-score on the validation set (97.7%) was an unidirectional LSTM with 16 units per cell and used 7x7 filters on the CNN2D convolutional layers. Its performance on the validation set was again almost perfect and for the first time, the performance on the test set surpassed the 90% mark (by a single model). This model when tested with the YOLO detections on the test set, scored a 90.8% F1-score which was again significantly better than the CNN3D ensemble technique and about the same as the ensemble performed on Section 4.1.5.G with the firstly trained LSTMs. Knowing from previous experiments that a single model has been performing worse than the result of the stacking ensemble between different variations of the same model, the same was done here. Again, as in Section

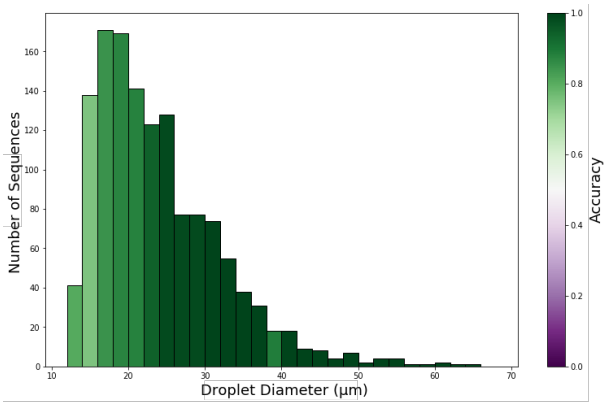


Figure 4.19: Droplet Sequence Accuracy by Diameter - CNN2D (Baseline Model 1) + LSTM models Stacking Ensemble - Test Set

Table 4.15: Precision, Recall and F1-Score per class resultant of CNN2D (Baseline Model 2) + LSTM Stacking Ensemble - Test Set

VOC	PRECISION	RECALL	F1-SCORE
Acetone	1	1	1
Acetonitrile	0.974576271	0.950413223	0.962343096
Chloroform	1	1	1
Dichloromethane	0.616915423	1	0.763076923
Diethylether	0.939130435	0.981818182	0.96
Ethanol	0.990990991	0.990990991	0.990990991
Ethylacetate	1	0.422535211	0.594059406
Heptane	1	1	1
Hexane	1	1	1
Methanol	1	1	1
Toluene	0.972222222	0.992907801	0.98245614
AVERAGE	0.953985031	0.939878674	0.932084232

4.1.5.G, the classifier that achieved the best results on the validation set was the Multi-layer Perceptron. Table 4.15 and Figure 4.20 present the results achieved on the test set using this strategy as well as the corresponding confusion matrix.

As expected, the stacking ensemble method increased even more the overall F1-score achieved for the test set reaching an impressive mark of 93.2%, the highest recorded until this moment, surpassing the 91.0% value achieved on the previous LSTM ensemble. The 16 mistakes on the Toluene class observed in Section 4.1.5.G were no longer a problem. The only problem with this model was that it kept making mistakes when trying to classify Ethylacetate sequences. Even though a slight increase in recall for this class was observed, it was still below 50%. Knowing that this model has performed almost on a perfect level for every other class, only making some rare isolated mistakes on a few of them, increasing even more its performance only depends on increasing the results on the Ethylacetate class.

Regarding the effect of the droplet size on the system accuracy (Figure 4.21), there is not much difference from what was observed on the stacking performed with the previous LSTM version.

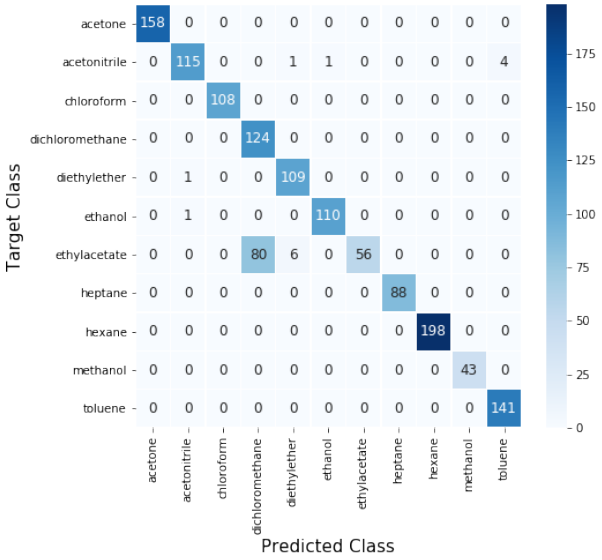


Figure 4.20: Confusion matrix on prediction results of CNN2D (Baseline Model 2) + LSTM models Stacking Ensemble - Test Set

4.1.5.I Result Comparison

In previous sections, some comparisons between the results from different methods were already made. Here, in Table 4.16, the results from all methods are presented together, in the form of F1-Score on both the validation and test sets, to wrap up this experiment. Thus, it can be clearly observed which models produced the best and worst results.

In a first stage, a lot of effort was put into the CNN3D (Baseline Model 1) strategies which ended up not paying off since the best results were not achieved neither with a CNN3D model nor the Baseline Model 1. The CNN2D coupled with an LSTM were by a significant margin the best methods, having their results boosted even more when using the stacking ensemble.

4.2 Concentrations Experiment

While in the first experiment 11 different VOCs were tested, in this second one, Acetone is the only VOC used. The goal of this experiment is to determine the concentration of Acetone present in the gas, therefore the gel was only exposed to several different Acetone concentrations whereas in the first experiment, every VOC was exposed to the gas with a given concentration. 6 different gel regions were chosen resulting in 6 different videos. Each region started by being exposed to a low concentration. For every following exposure, the concentration kept increasing until the maximum possible one was reached. Since it is impossible to expose the gel to every imaginable concentration value, only 11 different Acetone concentrations were produced. The values used can be found below in Table 4.17.

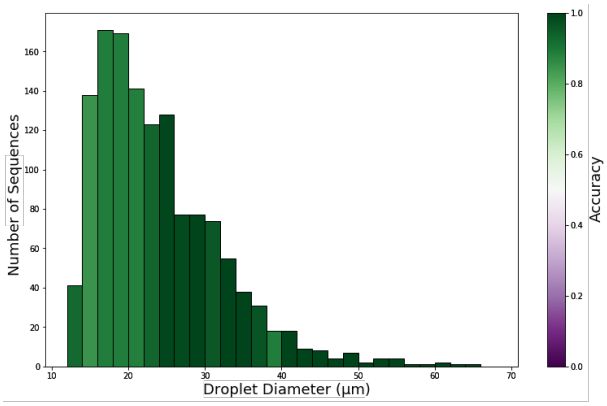


Figure 4.21: Droplet Sequence Accuracy by Diameter - CNN2D (Baseline Model 2) + LSTM models Stacking Ensemble - Test Set

Table 4.16: Validation and Test F1-Scores for different strategies approached in the 11 VOCs experiment

	F1-SCORE	
Strategy	**Validation**	**Test (YOLO)**
CNN2D + Voting System [1]	0.974 [a]	0.727
CNN3D (BM 1) [1]	0.962	0.748
CNN3D (BM 1) – Stacking [1]	0.984	0.841
CNN3D (BM 1) – OvR [1]	-	0.745
CNN3D (BM 1) – Smaller Input [2]	0.950	-
CNN3D (BM 1) – Smaller Input Stacking [2]	0.979	0.816
CNN3D (BM 2) [2]	0.848	-
CNN2D + LSTM (BM 1) [2]	0.987	0.888
CNN2D + LSTM (BM 1) – Stacking [2]	0.991	0.910
CNN2D + LSTM (BM 2) [2]	0.977	0.908
CNN2D + LSTM (BM 2) – Stacking [2]	0.992	**0.932**

[1] Original Input Size: 148x148
[2] Smaller Input Size: 75x75
[a] Voting System not applied

The acquisition method is the same as the one used in the previous experiment except for the fact that for consecutive gas exposures, the VOC concentration changes.

Figure 4.22 presents the relation between the flow rate and VOC concentration. The function that relates both domains was computed using a linear regression with the values present in Table 4.17. This relation (R^2 = 0.998) can be characterized by the equation

$$Conc(flow) = 3.753409 flow + 0.271591 \tag{4.2}$$

4.2.1 Sequence Splitting

Unlike the dataset for the first experiment, the team that acquired the data provided information regarding the frames that delimited the exposures in all the 6 videos. Knowing this information in advance spares

Table 4.17: Acetone concentrations per exposure

Exposure	Flow (SLPM)	VOC concentration %(v/v)
1	0.2	0.94
2	0.28	1.29
3	0.36	1.63
4	0.44	1.96
5	0.52	2.27
6	0.6	2.58
7	0.68	2.87
8	0.76	3.15
9	0.84	3.43
10	0.92	3.69
11	1	3.95

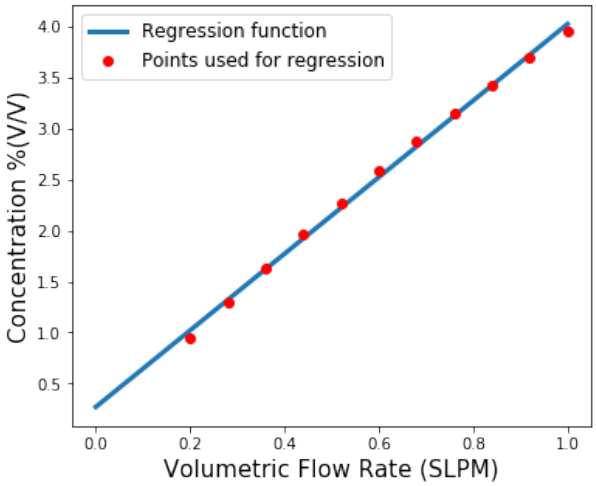


Figure 4.22: Concentration in function of the Volumetric Flow Rate

some work trying to delimit the sequences, as done for the first dataset, and in theory is also more accurate since it was handed over directly by the laboratory team that generated the data. Nevertheless, the analysis regarding the sum of pixel values per frame was still done in order to have a general overview of the data. These plots can be found in Figure 4.23.

From the plots in Figure 4.23 it is possible to observe that with the increase in Acetone concentration, the droplets reaction to the gas is also more significant, since pixel values get further away from the original ones over time.

When the information about the frames that delimit the 11 exposures per video is known, it is straightforward to compute the sequence lengths. The same problem as before arises, the sequence lengths are not all the same. Therefore, the exact same strategy was applied and the first 311 frames (smallest sequence length) were selected per exposure. Even by selecting the shortest sequence length, all sequences were still considerably long. A sequence with 311 frames has almost 10 times the length of a sequence in the first dataset. By the time this analysis was conducted, some experiences were already

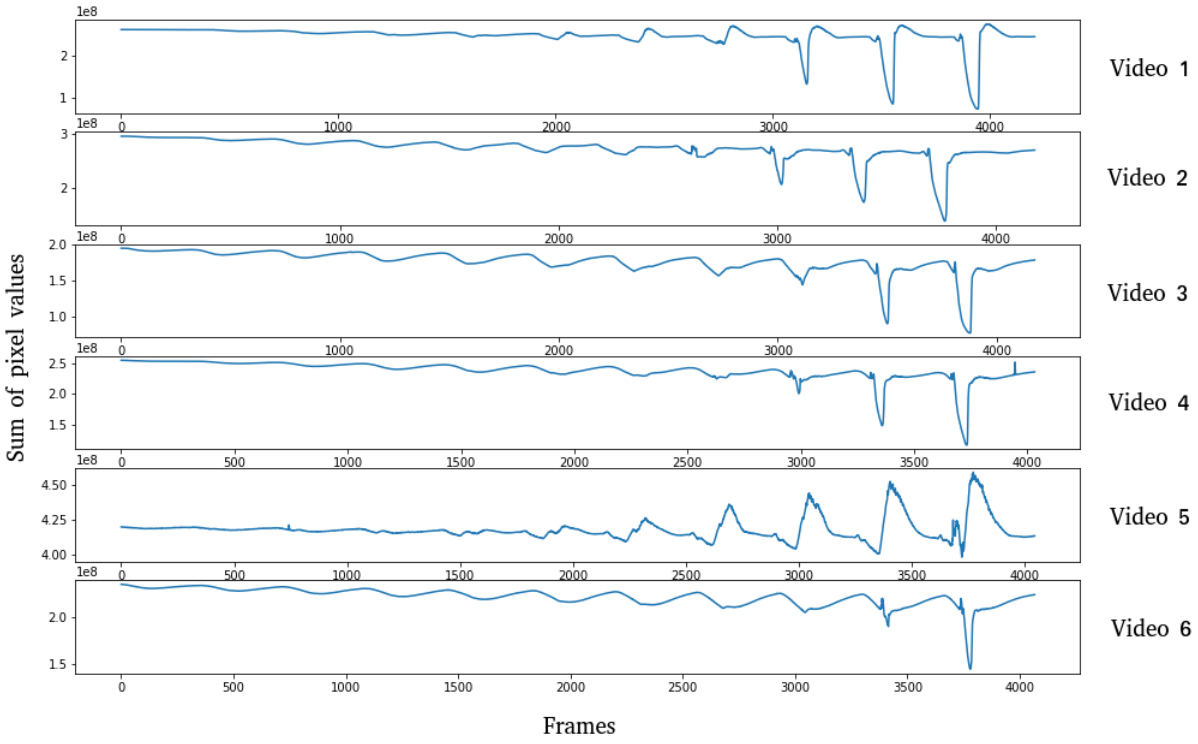


Figure 4.23: Total frame pixel colour sum (3 colour channels) per video

performed with the first dataset, which led to the conclusion that it would be extremely time and memory consuming to run experiences with sequences with 311 frames. A compromise had to be done, reduce sequence length in order to facilitate future model training but not by an amount that would result in a huge loss of information. Since it is difficult to know how much to shorten the sequences, the decision was based on the colour plots already mentioned. The final decision was to reduce the sequences size by a factor of 4 which means that 1 out of every 4 consecutive frames was kept. If the sequences had been reduced by a larger factor, a significant reduction in colour plot definition would be noticed compared to the original ones and that was avoided. With all these transformations, the sequences ended up with a length of 78 frames. After splitting the sequences in all the 6 videos and applying the mentioned changes, the colour plots per sequence turned out the ones in Figure 4.24.

4.2.2 Droplet Dimensions

The analysis done for the first dataset regarding droplet dimensions was also performed for this second one. In Table 4.18 it is possible to observe the Maximum, Minimum, Mean and Median values regarding droplet diameters and in Figure 4.25 the histograms per video also with respect to these diameters.

As it is possible to observe from both Table 4.18 and Figure 4.25 there is again a significant fluctuation in droplet diameters between videos. For instance, the gel region associated with the third video is

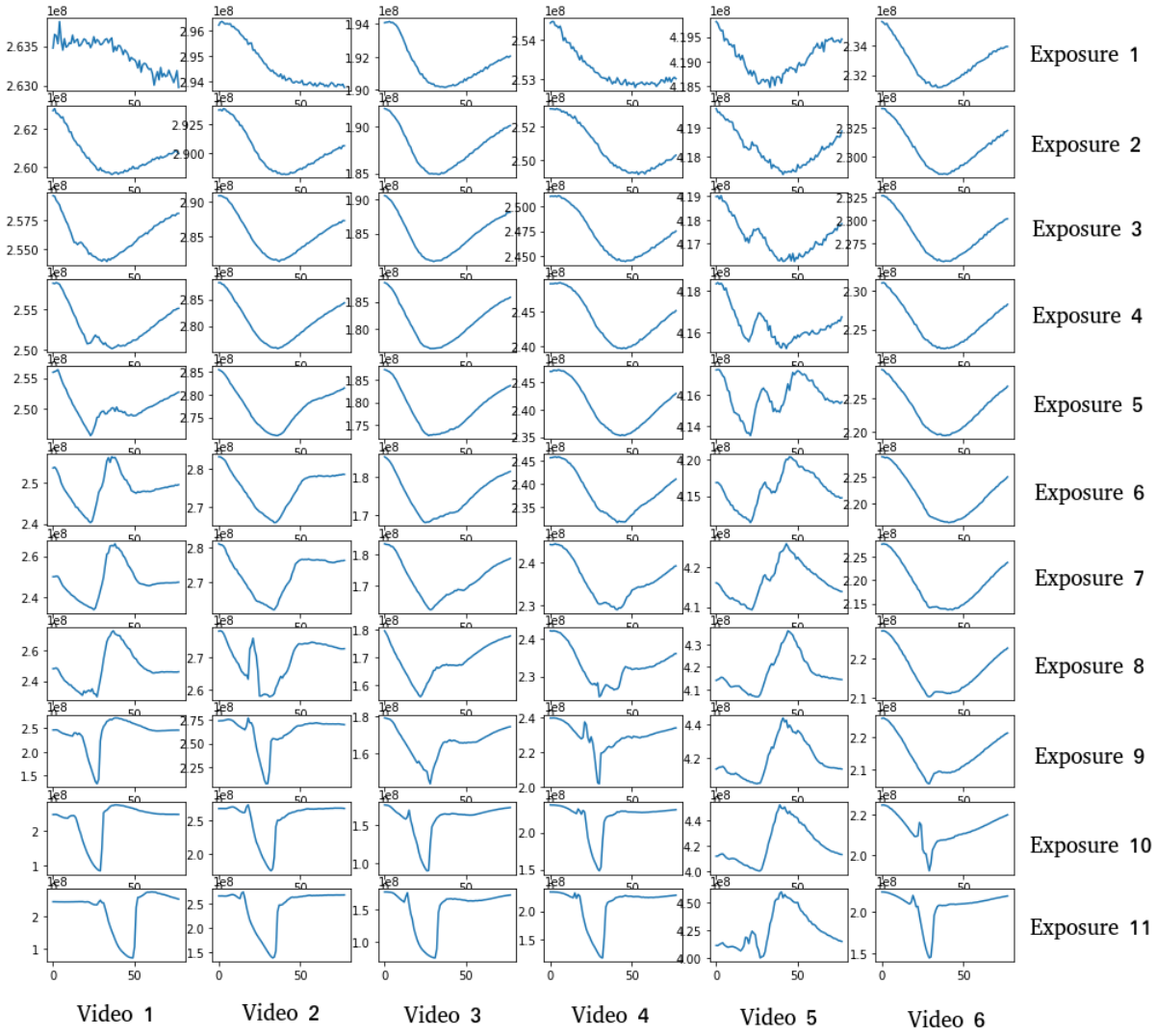


Figure 4.24: Total frame pixel colour sum (3 colour channels) per video and exposure

populated with only droplets in the low side of the size spectrum not having a single one with a diameter exceeding 41 μm. On the other hand, the fifth video records the dynamic evolution of droplets with all kinds diameters, even one that exceeds the 107 μm which is the biggest droplet in the whole dataset by a considerable margin. In accordance with what was done for the first set of data, an histogram was produced in order to visualize the distribution of every labelled droplet in this dataset. This histogram can be found in Figure 4.26. A similar trend to the first dataset is observed with the number of droplets diminishing with the increase in diameter.

All the histograms represented in this section used bins with 3 μm of width and went from 15 μm to 108 μm in order to cover the entire range of detected diameters.

On another note, the maximum, minimum and mean diameters obtained for all the droplets were

Table 4.18: Droplet Diameter Stats (in μm)

Video	Max	Min	Mean	Median
1	58.180043	15.680171	32.353245	31.814935
2	99.089867	16.363636	30.677267	24.997934
3	40.906566	16.818182	24.512981	23.636364
4	60.000000	19.545455	32.087922	31.475217
5	107.718642	17.266745	36.300533	31.590092
6	78.408762	16.121953	32.440184	27.953622

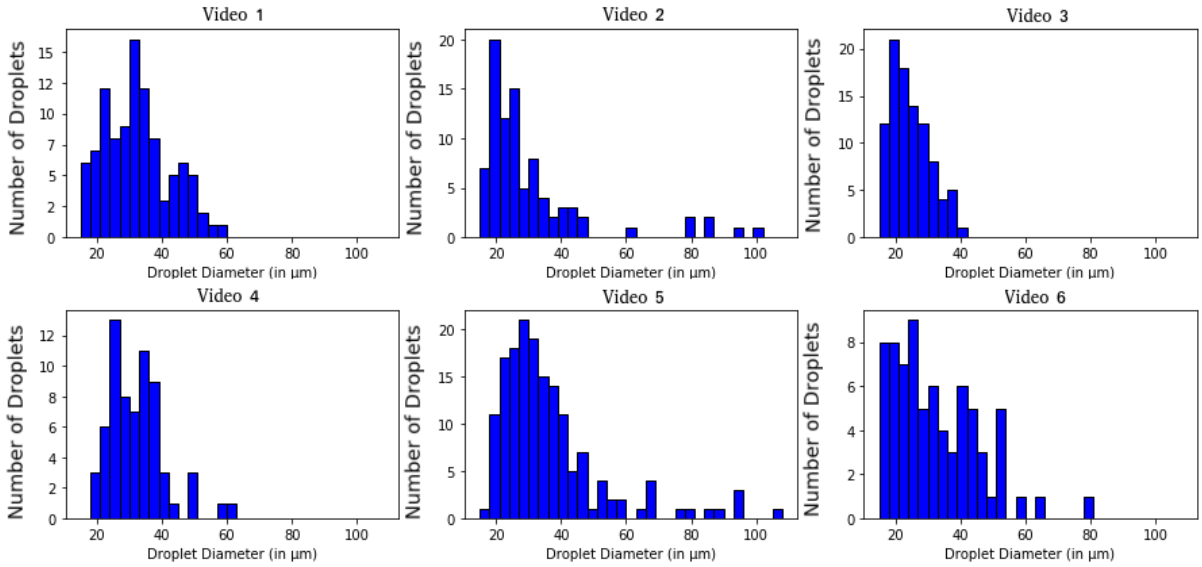


Figure 4.25: Droplet Diameters Distribution (in μm) in the 6 different gels

107.71864 μm, 15.68017 μm and 31.89441 μm respectively.

4.2.3 Train, Validation and Test Sets

As seen previously, the dataset for this experiment is composed of six videos, where in each one of them, a different gel region is exposed consecutively to 11 increasing Acetone concentrations. Four videos were selected to be part of the training set, one for the validation set and the remaining one for the test set. In the previous dataset, exposures of the same VOC were all recorded over the same gel region. Then, in order to have all the classes present in the different sets of data, droplets had to be shared between them which is not ideal. For this dataset, since exposures of all 11 different concentrations were recorded over the six selected gel regions, this problem did not occur and a more 'fair' division of the data was possible.

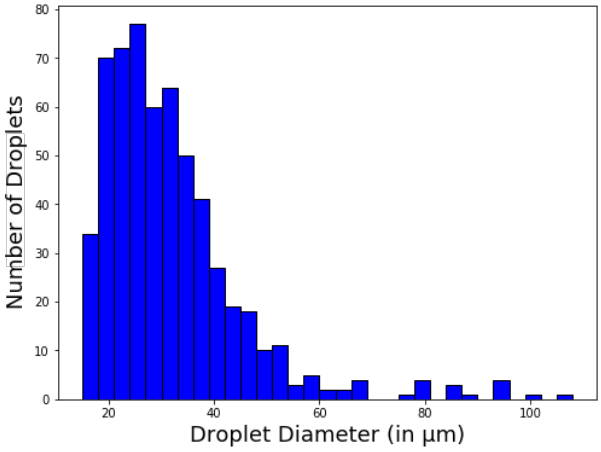


Figure 4.26: Droplet Diameters Distribution Histogram (in μm)

4.2.4 YOLO - Detection Stage

The setup for the training was exactly the same used in Section 4.1.4. Since the previous results were pretty good and there were no major changes regarding datasets, it was decided to keep the same set of parameters. In Figure 4.27, it is possible to observe the plots regarding Precision, Recall, mAP and others related to class and bounding box losses on the validation set.

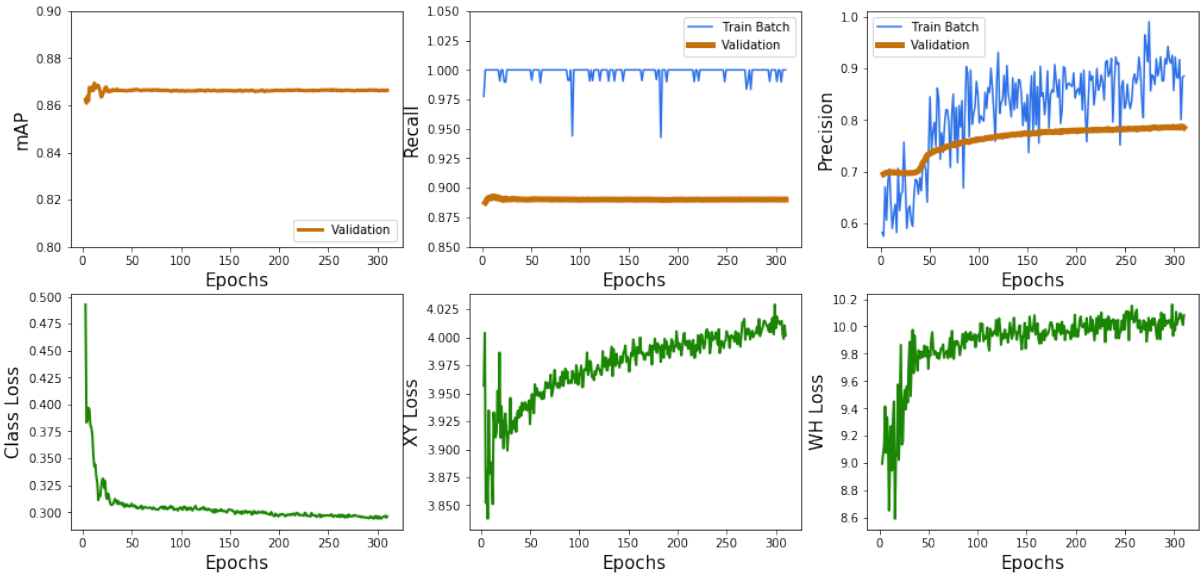


Figure 4.27: YOLO training precision, recall, mAP and losses

Training was stopped after around 310 epochs. The metric and loss plots are not as straightforward as the ones for the previous YOLO training. As it is possible to see from Figure 4.27, as time goes by, losses regarding the bounding box position and dimensions kept going up. This did not match the information given by the Precision plot. For the Precision on the validation set to be increasing over time,

either the number of True Positives was rising or/and the number of False Positives was decreasing. This values are correlated since a detection goes from being a False Positive to a True Positive by surpassing the threshold value of IoU with the ground truth box. In order for this to happen, the accuracy regarding the generated bounding box position and dimensions has to be increasing and not decreasing, as it is hinted by the constant increase in XY and WH losses. Despite these inconsistencies, if the precision kept going up for the validation set, apparently, there would be no reason to stop training. On the other hand, the mAP value, that already took into consideration the Recall and Precision values was very constant. Either way, training was only ceased when the precision value started to stagnate and further training would not have brought any significant improvements. One other thing worth noticing was that, given the fact that no droplets were shared between the training and validation sets, the values for recall and precision on the validation set were always below the ones registered, on average, for the training batches which is what normally happens.

Although there were some misunderstandings regarding the precision and bounding box losses, in order to select the best set of weights for the YOLO model, a test was ran between the ones saved at epoch 12 and at the end. At epoch 12, the highest value for mAP was recorded as well as the lowest ones for XY and WH losses. The test consisted on checking which model achieved a better average IoU against the ground truth boxes on 11 images (1 random frame from every different concentration exposure). Although both registered values were very high, around 92 %, the model at epoch 12 edged it with a small margin of 0.8 %, becoming the chosen one for the experiences ahead.

Figure 4.28 shows the YOLO detections on an image from the test set, where the predicted bounding boxes appear on top of the ground truth ones.

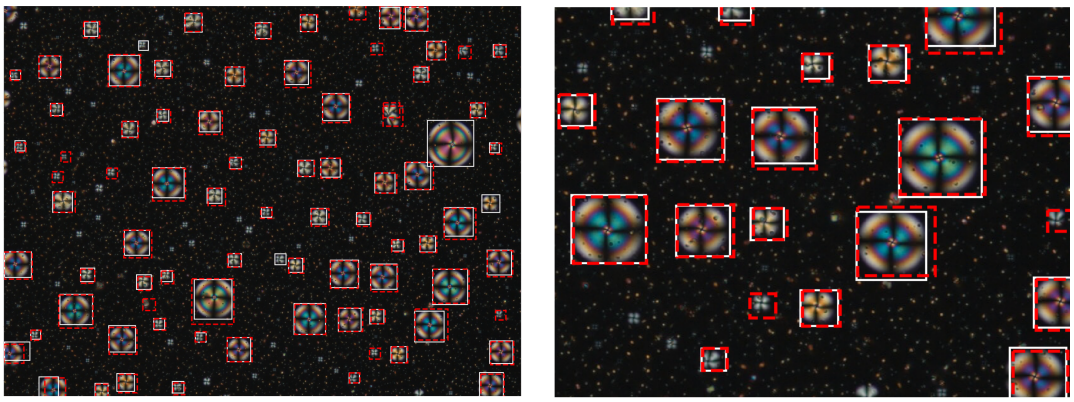


Figure 4.28: Ground truth bounding boxes (White) vs YOLO detected bounding boxes (Red) - **Left**: Whole Frame **Right**: Frame Zoomed In

Table 4.19: CNN3D Models Parameters

Parameters	Values
Number of Filters (Conv1, Conv2)	(16,32), (32,64)
Filter Dimensions Conv Layers	10x10x10, 15x15x15, 15x10x10, 20x20x20, 20x15x15,20x10x10
Stride Conv Layers	1
Filter Dimensions Max-Pool Layers	2x2x2
Stride Max-Pool Layers	2
Dropout Rate	0.5
Units FCL 0	Flattened dimensions of 2nd Max-Pool Layer Output
Units FCL 1	32
Units FCL 2	11

4.2.5 Regression Stage

As seen in the 11 VOCs section of this thesis, even though bigger droplets are more likely to hand out better results, it is not necessary to resize all of them to the biggest dimensions in order to achieve significantly better results. So for the dataset used in this section, where the biggest observed droplet has a 107.7 μm diameter (237x237 pixels), every droplet was resized to 100x100 pixels in an attempt to find a compromise between not losing too many properties on the bigger droplets and not blurring too much the smaller ones. Besides that, the training times for the used models also benefit a lot from this. The sequence lengths were kept as 78 as discussed in Section 4.2.1. Regarding the models trained and tested in this section, their last layer that used to be a 11 unit softmax one is now only composed by a single neuron not coupled with any activation layer. Since this is now a regression problem, this change was crucial so the networks were able to produce linear results. 'Adam' was still the chosen optimizer but the loss function had to be changed, again due to the problem involving a regression task The chosen one was Mean Squared Error (MSE).

Every droplet was exposed to a gas containing a certain concentration of Acetone. Being the goal of this problem to develop networks capable of identifying the concentration associated to a certain droplet sequence, these concentrations should be the target of every data sample. Instead, due to a small mistake, the targets were set to be the volumetric flow rates of Acetone (Table 4.17). This is not problematic at all since both the concentrations and the flow rates are basically related in a linear way as shown in Figure 4.22.

4.2.5.A CNN3D (Baseline Model 1)

On the first experiment on this new dataset, again the CNN3D described in Section 3.4.1 was used to start with. This model was trained for different sets of parameters (Table 4.19), making a total of 12 different instances of the same model.

The model that was able to achieve the best validation loss (4.41×10^{-3}) used 16 and 32 filters of

dimension 15x15x15 on the first and second convolutional layers. This corresponded to a Mean Absolute Error (MAE) of 0.0546. Figure 4.29 presents the predictions of this best model on the validation set, as well as the mean prediction per exposure and the mean absolute error also per exposure.

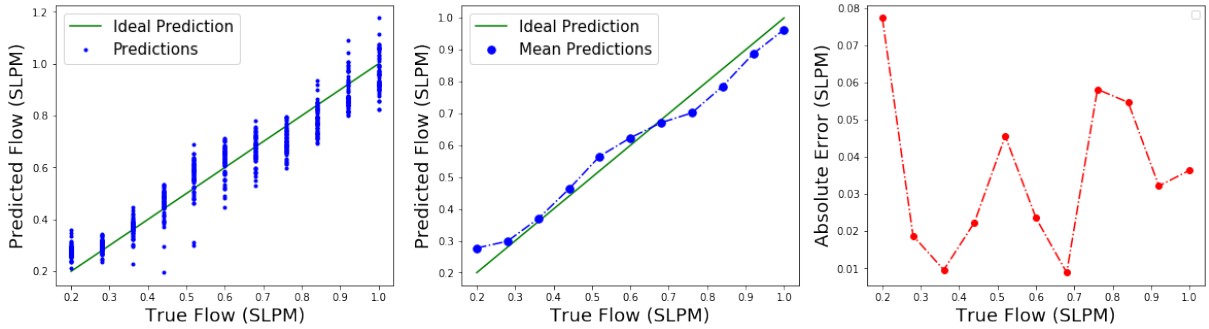


Figure 4.29: CNN3D (Baseline Model 1) validation set predictions

Even though there is a significant fluctuation on the droplet predictions for the same exposure, by computing the mean prediction on the droplets within the same exposure, it is possible to achieve consistent results. The highest absolute error on a single exposure (around 0.08) was recorded for the exposure with the lower concentration of Acetone.

As done in the 11 VOCs experiment, the best model was tested together with the YOLO detections on the test set. The test sequences were acquired in the exact same way as before. The results achieved on the test set are presented in Figure 4.30. As it is possible to observe, compared to the validation results, the predictions on the droplet sequences within the same exposures suffer from an even higher variance. Even with a high variance, if the predictions were more centered on the real value, a good overall prediction would be possible to achieve. What happens is that most predictions are either above (for low concentrations) or below (for higher concentrations) the true values. On this set a MAE of 0.0993 was registered which saw an increase of 82% in regards to the error registered on the validation set.

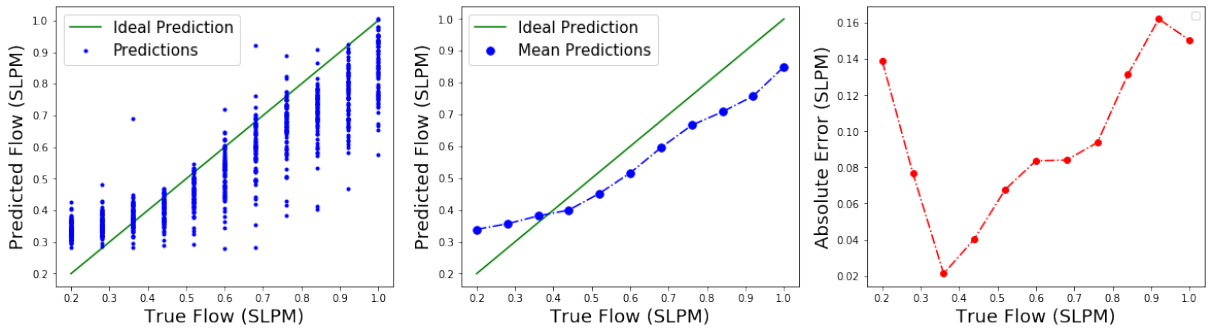


Figure 4.30: CNN3D (Baseline Model 1) test set predictions

Since one of the goals of this Thesis is to check the influence that the droplets' dimensions have on the results, an analysis regarding the performance according to the droplet sizes was made. Figure

Table 4.20: CNN3D (Baseline Model 2) Parameters

Parameters	Values
Number of Filters (Conv1&2, Conv3&4, Conv5&6)	(8, 16, 32)
Filter Dimensions Conv Layers	2x2x2, 3x3x3, 5x5x5
Stride Conv Layers	1
Filter Dimensions Max-Pool Layers	2x2x2
Stride Max-Pool Layers	2
Dropout Rate	0.5

4.31 presents that said analysis. Whereas in the first part of the work, where it was clear that bigger droplets were able to deliver more accurate results, for this experiment no clear trends are observed. The 'medium' sized droplets appear to produce lower errors than the smaller and bigger ones. The bigger droplets, with diameters comprehended between 70μm and 80μm, which were the ones expected to produce better results went in the complete opposite direction. Even though they are present in a significant fewer quantity, they registered a mean absolute error just above 0.2 which is massive considering the orders of magnitude worked within this problem.

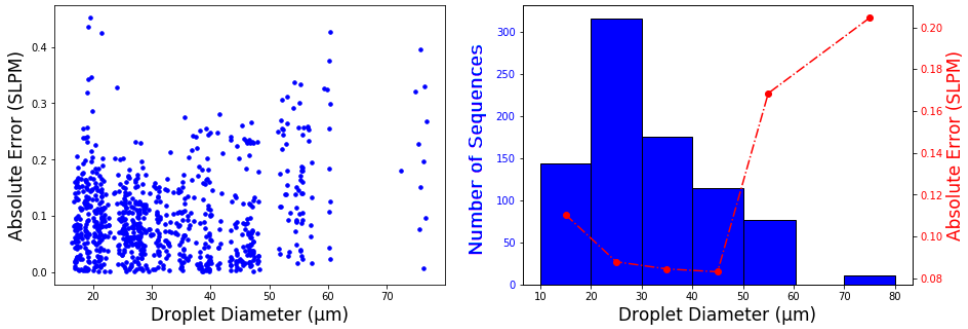


Figure 4.31: CNN3D (Baseline Model 1) test set errors in function of the droplets diameters

4.2.5.B CNN3D (Baseline Model 2)

The CNN3D version of the model described in Section 3.4.2 was used to try and achieve better results than the ones achieved with the Baseline Model 1 architecture. A total of 3 models were trained with this architecture. The parameters used for all three are presented in Table 4.20.

The model that used 3x3x3 filters on the convolutional layers was the one able to attain the lower validation loss (2.68×10^{-3}). The corresponding MAE achieved for this set was 0.0418 which represents a decrease in 23.4% when compared to the same value registered on the same set of data by the CNN3D Baseline Model 1. The predictions made by this model on the validation set are represented in Figure 4.32.

Given the significant drop in MAE achieved by this model, it lead to believe that it would also produce better results on the test set due to its better generalization capacities. What really happened was

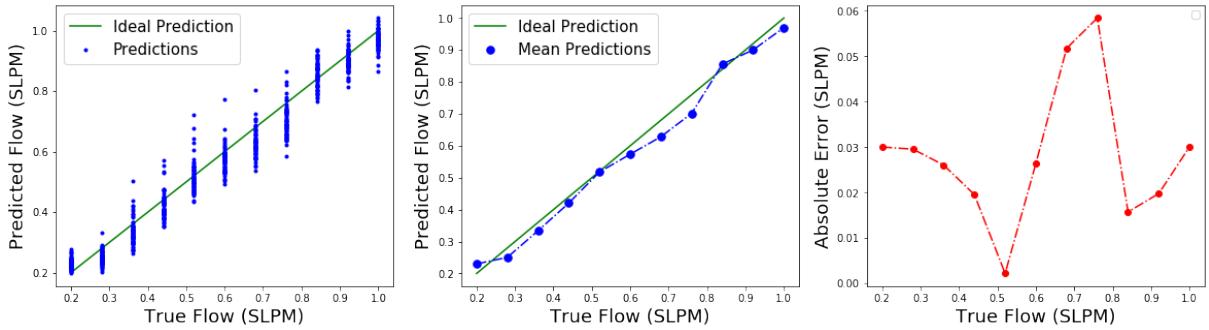


Figure 4.32: CNN3D (Baseline Model 2) validation set predictions

an increase in this value for the test set, although a very slight one, with the predictions averaging an absolute error of 0.1038. The prediction plots on this set are represented in Figure 4.33. The trend regarding the mean predictions and the reference values is very similar to the one observed before. In regards to the size influence on the predictions, it remained inconclusive since the plots relating the error with the droplet diameters were very similar to the ones displayed in Figure 4.31.

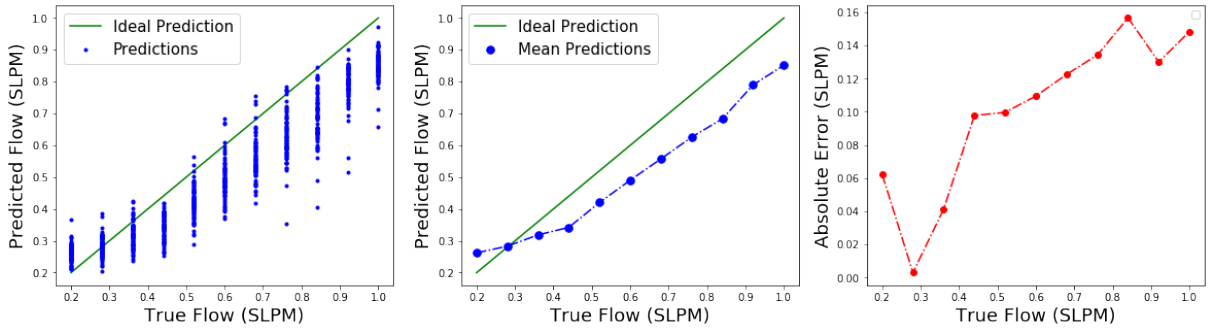


Figure 4.33: CNN3D (Baseline Model 1) test set predictions

4.2.5.C CNN2D (Baseline Model 1) + LSTM

On the first part of the work, even though the results achieved with the 3 dimensional CNNs were satisfactory, the really good ones were obtained through the use of LSTMs. Hence, when approaching this problem, a higher faith was put upon this type of networks. As done for the 1st part of the work, a LSTM system was firstly tried coupled with a (Baseline Model 1) CNN2D, used as a feature extractor for every droplet frame in the sequence. As usual, the model was tested for different sets of parameters (4 this time). All 4 models were set to be bidirectional LSTMs. These parameters can be found in Table 4.21.

Out of the 4 trained models, the model that better performed on the validation set was the one using 3x3 kernels on the convolutional layers. It was able to achieve a MSE and a MAE of 5.42×10^{-3} and

Table 4.21: CNN2D (Baseline Model 1)+ LSTM Model Parameters

Parameters	Values
Number of Filters (Conv1, Conv2)	(8, 16)
Filter Dimensions Conv Layers	3x3, 5x5, 7x7, 10x10
Stride Conv Layers	1
Filter Dimensions Max-Pool Layers	2x2
Stride Max-Pool Layers	2
Dropout Rate	0.5
LSTM cell units	32

0.0553, respectively. Compared to the best performing CNN3D model already trained (Section 4.2.5.B), the errors on the validation set were higher which was not expected due to the reasons mentioned earlier. The predictions made on this set are presented in Figure 4.34. It can be observed that even if there is a high variance in the predictions within the same exposure, the errors achieved on the mean predictions are relatively low given the very uniform 'mirror effect' around the true values.

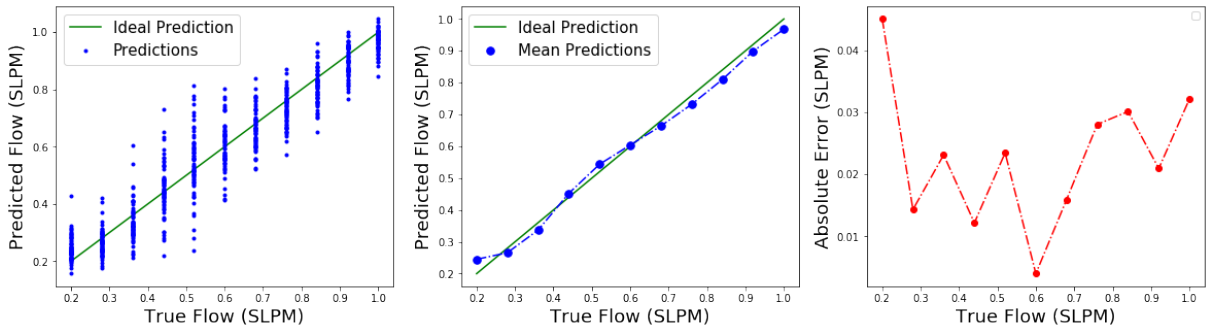


Figure 4.34: CNN2D (Baseline Model 1) + BiLSTM validation set predictions

Again, to really test the model on unseen data, the predictions made by the YOLO network on the test set were fed to the model. Figure 4.35 presents the results. The first thing that stands out is the enormous variance in predictions of the same flow/concentration. Besides that, given the fact that these guesses were not, by any means, centered on the real values, not even the average predictions were accurate. For the higher flow/concentration values, the MAE surpassed the 0.2 mark which is not reasonable at all given the flow/concentration range. A MAE of 0.1638 was registered which is, by far, the worst obtained result out of the 3 different experiments made until this moment. In addition to this, the droplet size still does not seem to have any meaningful influence on the performance.

4.2.5.D CNN2D (Baseline Model 2) + LSTM

Even though the previously tested LSTM model completely underachieved on the test set when compared to both the CNN3D models, the LSTM paired with the CNN2D (Baseline Model 2) was the model that achieved the best results on the first part of the work so it was always worth a try. The parameters

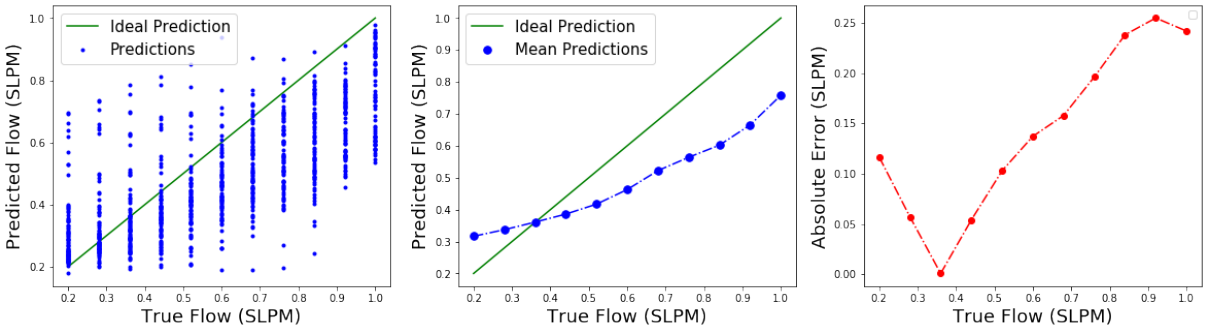


Figure 4.35: CNN2D (Baseline Model 1) + BiLSTM test set predictions

Table 4.22: CNN2D + LSTM Model Parameters

Parameters	Values
Number of Filters (Conv1&2, Conv3&4, Conv5&6)	(8, 16, 32)
Filter Dimensions Conv Layers	3x3, 5x5, 7x7
Stride Conv Layers	1
Filter Dimensions Max-Pool Layers	2x2
Stride Max-Pool Layers	2
Dropout Rate	0.5
LSTM cell units	16, 32

used on this network are presented in Table 4.22. These parameters were only tested for BiLSTMs.

Out of the 6 trained models, the one that was able to achieve better results on the validation set used 3x3 filters on the Convolutional layers and 32 units per LSTM cell. It got to MSE and MAE values of 4.77×10^{-3} and 0.0514. Even though these values are better than the ones registered for the previous LSTM implementation with Baseline Model 1, they are still below the mark set by the CNN3D (Baseline Model 2). Even if the predictions present a significant absolute error, the average ones are able to 'follow' really closely the true values, only exceeding the 0.04 MAE per flow/concentration on the lowest one. The predictions on the validation set are shown in Figure 4.36.

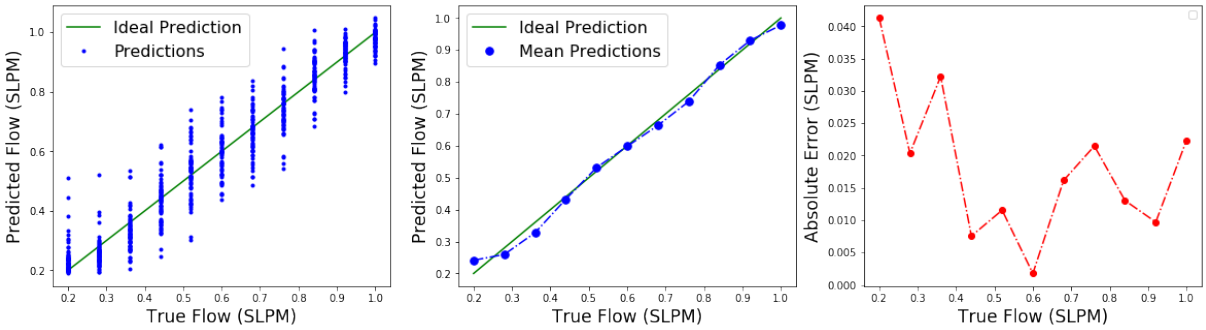


Figure 4.36: CNN2D (Baseline Model 2) + BiLSTM validation set predictions

When testing the model on the YOLO sequences built from the test set, the results were once again

very disappointing and very similar to the ones obtained for the previous LSTM experiment. The predictions are very much identical to the previous ones so the use of a different architecture, preceding the LSTM, did not seem to make any significant difference. This results are presented in Figure 4.37. The mean absolute error achieved on this set was 0.1684 which is mostly influenced by the huge amount of droplet sequences associated with incredibly low predictions. Once again, no conclusive information could be taken from the analysis on the impact of droplet size on the performance.

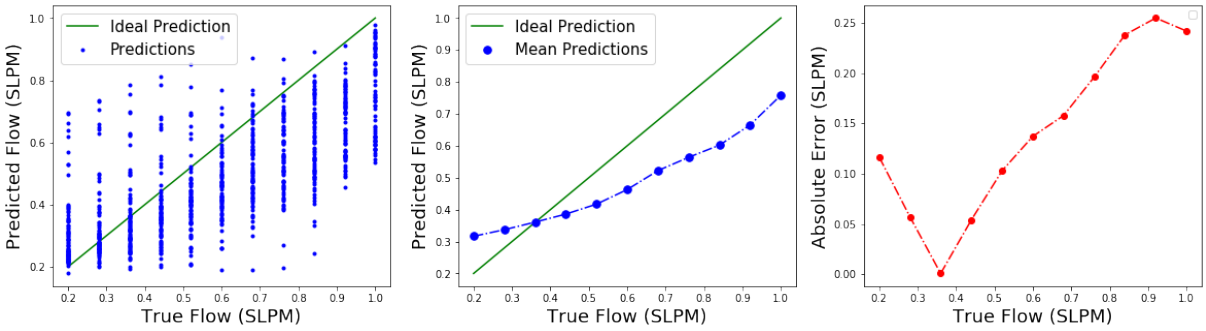


Figure 4.37: CNN2D (Baseline Model 2) + BiLSTM test set predictions

4.2.5.E Dataset Swap

One of the decisions made, coming into this problem, was not to perform any kind of cross validation in order to be able to experiment with different model architectures without the need of spending huge amounts of time on training. On the other hand, this implies a lesser level of confidence on the performance of the different models since only one evaluation is performed on each model. If by any circumstance, the data on the test set happens to have features significantly different from the data in the train and validation sets, the results might come out influenced by a high chance factor. Having extensive datasets is the most obvious solution to this problem since the data is more likely to cover a higher range of possibilities, decreasing the chance of already trained models facing new data with completely different characteristics.

In this problem, as mentioned before, out of the 6 recorded videos, 5 were selected for the training stage of the models (4 for training and 1 for validation) and the remaining one for testing. This cannot be considered a extensive dataset, very far from it, so the bad results registered on the previous sections may be related to this fact and not to the inability of the different models to solve this task. To check whether this was or not the case, a new distribution of the videos across the different sets of data (train, validation and test) was randomly made, only making sure that the video that was previously selected for the test set was not selected again.

Two different networks were tested before for both the CNN3D and CNN2D + LSTM architectures. In each architecture, both Baseline Model 1 and Baseline Model 2 were used. The models that best

performed on the validation set of each of these architectures were trained and tested again for these new sets of data. These are the models represented in Section 4.2.5.B and 4.2.5.D, coincidentally both associated to Baseline Model 2. Figure 4.38 presents the predictions of both these models on the validation set. The CNN3D model once again obtained better results than the LSTM, achieving a MAE of 0.0489 while the latter was only able to achieve 0.0605. Comparing these plots with the ones computed from the predictions on the original validation set, there is not a significant difference with both models also being able to achieve pretty accurate average predictions on this new set.

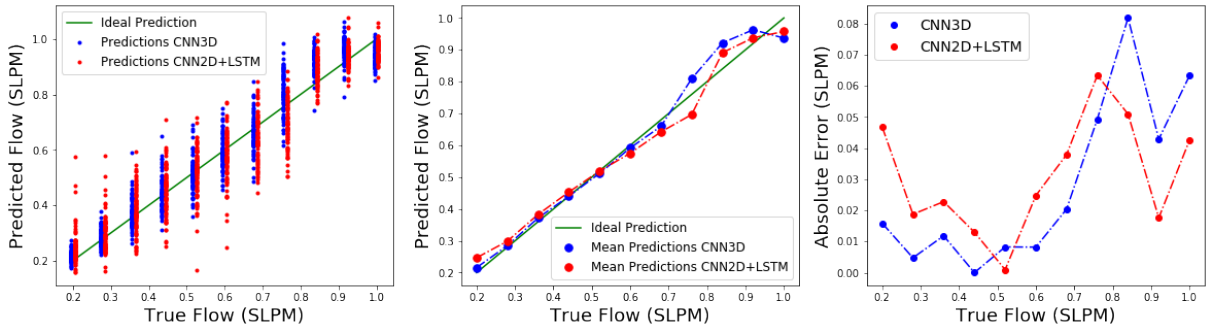


Figure 4.38: CNN3D and CNN2D+LSTM new validation set predictions

The predictions performed on the YOLO detections on the test set are presented in Figure 4.39. While for the original test set the predictions presented very high error values, especially for high concentrations, now, that does not seem to happen. The CNN3D model fulfilled the expectations of performing better than the LSTM model, given the results on the validation set, and registered a MAE of 0.0490. This value equals the one obtained on the validation set which is impressive given that this set of data was completely unseen by the models. Besides that, it is by far the lowest registered error on the test set of all the previous experiments which helps to prove the point discussed earlier in this section. As for the LSTM model, it only managed to achieve a 0.0711 average absolute error which, even though it represents an increase of 45% on the CNN3D experiment, it is also better than all the previous experiments. The main difference found on these results, when compared to the original ones, besides the obvious increase in performance is also the significant decrease in variance of the values predicted by both models, specially by the LSTM. While before the predictions seemed to be a little bit all over the place, now they are definitely more compact and more centered on the true flow values.

Regarding the influence in performance of the droplet sizes, with better results also came more conclusive results. Figure 4.40 presents these results. For both models, with the increase in droplet diameter it is registered a decrease in concentration error with an exception for the bigger droplets on the LSTM model, for which the reason is unknown.

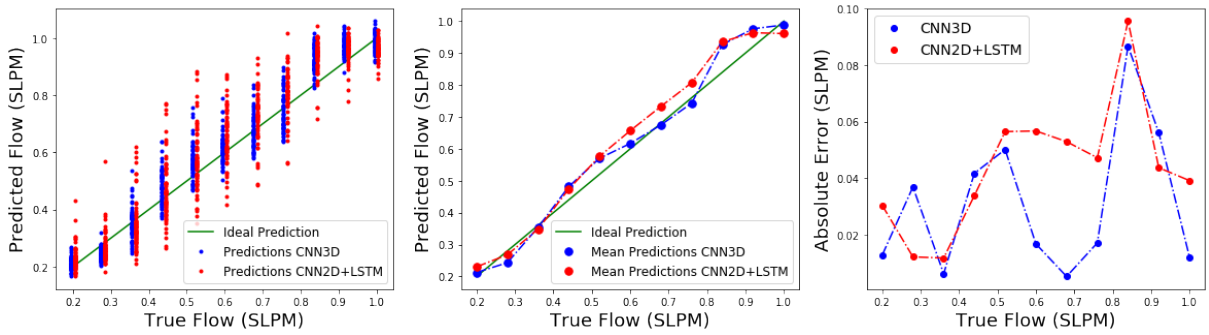


Figure 4.39: CNN3D and CNN2D+LSTM new test set predictions

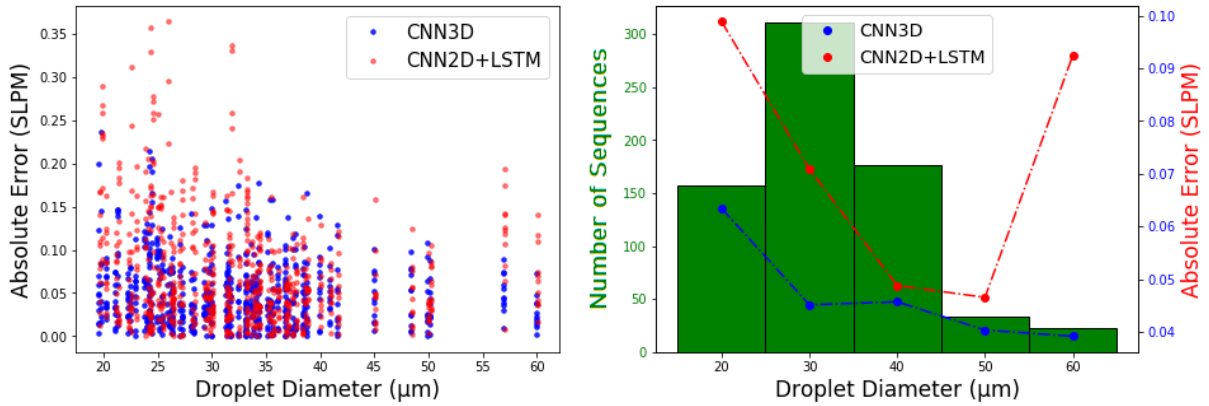


Figure 4.40: CNN3D and CNN2D+LSTM new test set errors in function of droplets diameters

4.2.5.F Result Comparison

Again, to wrap up the experiment, all results were put together. This time the results are presented in the form of the mean absolute error and can be observed in Table 4.23. As previously mentioned, for this experiment, the three dimensional CNNs were able to achieve significantly better results than the LSTMs on the test set. Despite the impact of the new data distribution on the final results, the CNN3D network still managed to pull ahead and produce better results than the LSTM.

Table 4.23: Validation and Test MAE for different strategies approached in the Concentrations experiment

	MAE	
Strategy	**Validation**	**Test (YOLO)**
CNN3D (BM 1)	0.0546	0.0993
CNN3D (BM 2)	0.0418	0.1038
CNN2D + LSTM (BM1)	0.0553	0.1638
CNN2D + LSTM (BM2)	0.0514	0.1684
CNN3D (BM 2) [1]	0.0489	**0.0490**
CNN2D + LSTM (BM2) [1]	0.0605	0.0711

[1] Redistributed Dataset

Chapter 5

Conclusions

In this work, the use of CNN based systems was explored in two different E-nose applications, one with the goal of detecting the substance exposed to the gel and the other with the goal of predicting the concentration of Acetone in the exposed gas. It was observed on both tasks that these systems were able to accomplish the objectives with satisfactory results.

In the 11 VOCs experiment, firstly the focus was around the CNN3D (Baseline Model 1) and trying to make it work by exploring many different strategies and parameter variations. Although the results overall were not bad, a few classes were constantly underachieving no matter what strategy was used so the focus had to be turned to another direction. A second deeper model (Baseline Model 2) was developed and LSTM networks were tested. After a few tests it was soon realized that the results produced by the LSTM models were significantly better than all the ones achieved by the 3 dimensional CNNs. The reason behind this was probably the different way on which the LSTMs approach the temporal component of the data, which should be more suitable to this specific data/problem. By putting together several of these models under a stacking ensemble, close to perfection predictions on all classes were achieved, except for one (Ethylacetate) that missed about half of the times. Regarding the influence of the droplet diameters on the models' performance, it was acknowledged that the bigger droplets are capable of producing more accurate results on a more regular basis. Since relying on a single droplet to predict the VOC might be a bit risky, if all the detected droplets contributed to the final prediction, and by assigning a bigger weight to the bigger droplets, it seems that it should be possible to achieve very consistent results

For the Concentrations experiment, curiously, the model that achieved better performance on the 11 VOCs experiment, was not the one that obtained the best one here. For both the original and changed data distribution, the (Baseline Model 2) CNN3D was the model that achieved the lowest error values. In this experience, the impact that a different distribution in data, in a small dataset, can have in regards to the final results was noticed. While the predictions on the original test set were all over the place,

mostly below the true values, on the second distribution the predictions were way more accurate and centered on the real values. Given this, it is very unlikely to get accurate results based on the predictions of singular droplets given the volatility presented on their predictions. Regarding the influence of the droplet size on this task, it is not 100% clear but it seems that the bigger ones are also able of giving more accurate predictions. This is mostly based on the results obtained on the last experience, with the changed sets, since the results on the original ones were disappointing and no assertive conclusions could have been made from them. So again, in order to get better results, bigger droplets should have a higher influence on the final prediction.

In terms of the YOLO networks used, they were always capable of identifying the droplets in images with great precision. During the testing phase, when some results were below expectations on the test set, they were compared with the ones obtained with the ground truth test sequences to check if the YOLO detections were having a big impact on the predictions. The ground truth results were always very similar to the ones obtained through YOLO detections so it can be concluded that this network was definitely a good choice to perform the droplet detection.

Although the tested models were able to produce good results on both problems, it is very unlikely that they would be able to achieve also good ones on a completely new set of data. This applies more directly to the 11 VOCs problem. As already mentioned, the fact that every exposure of the same VOC was done on the exactly same gel, with the exact same droplets, biased every model trained on this data. The experiment of Section 4.1.5.A, CNN2D + Voting System, provides very insightful information in this regard. On the validation set a F1-score of 97.4% was obtained and on the test set, after the voting, 72.7%. If, for example, a stacking ensemble had been applied to several of this 2 dimensional CNNs, the results achieved would be even higher. Every trained model was picking up a lot of information regarding the droplets themselves instead of the effect caused by the VOCs on them. Of course this also affects the models that take into consideration the time component (CNN3D and LSTMs) and probably because of that it is possible to achieve such good results.

For the concentrations problem, the best model would also probably not perform very well on a new set of data. This is due to the fact that there is a small amount of data samples to train the models. As seen in the sections before the change in set distributions, for that specific test set, the results were far from good but when the sets were redistributed, there was a significant improvement on the results for that new test set. These experiences might have helped to show which model would be able to perform better in a live/real situation, but in order for that to happen it needs, firstly, to be trained on a bigger set of data.

5.1 Future Work

In terms of future work, the most important thing is, definitely, to generate more data for both problems. Without data that can simulate a broad range of real life situations, even if the results on that data are astounding, when the model is put into action with 'unseen' information it might not perform as expected. As seen before the datasets are too short and limited. For the 11 VOC problem, exposures of the same VOC should be performed in as many different gels as possible in order to cover droplets of every types and sizes. By doing this, the models will not be able to associate a droplet itself to a specific VOC and, in theory, they will capture better their evolution through time which is what is directly influenced by the VOCs. It should also be given special attention to the recording settings so that there is not again a significant discrepancy between different exposures as happened with the frame rate, for example, which led to sequences that had a different number of frames even though they all lasted the same amount of time. The same basically also goes for the dataset of the Concentrations problem where the more data the better. If possible, droplets with diameters above 50 μm (for example) should be since, at least from what was able to be concluded, bigger droplets are able to generate more distinctive patterns that help distinguish different compounds. If that is not possible to achieve, it should be used a higher threshold on the moment of detecting the droplets so that the smaller ones are not taken into consideration.

Regarding the models and techniques used, there are always more things that can be tried like different architectures or experiment with more parameters. One idea would be, if enough data is available, to train models based on droplet sizes so that it is not needed to perform those big resizes before feeding the data into the models that can distort it. By doing so it would also be straightforward to conclude if the size really affects the predictions by comparing the performances of the different size models. This could not be performed with the current datasets since there is only a small number of bigger droplets and they would not be enough to train a model of their own.

Bibliography

[1] C. Sarafoleanu, C. Mella, M. Georgescu, and C. Perederco, "The importance of the olfactory sense in the human behavior and evolution," *Journal of medicine and life*, vol. 2, no. 2, p. 196, 2009.

[2] Y. Y. Broza, P. Mochalski, V. Ruzsanyi, A. Amann, and H. Haick, "Hybrid volatolomics and disease detection," *Angewandte Chemie International Edition*, vol. 54, no. 38, pp. 11 036–11 048, 2015.

[3] M. K. Nakhleh, H. Amal, R. Jeries, Y. Y. Broza, M. Aboud, A. Gharra, H. Ivgi, S. Khatib, S. Badarneh, L. Har-Shai *et al.*, "Diagnosis and classification of 17 diseases from 1404 subjects via pattern analysis of exhaled molecules," *ACS nano*, vol. 11, no. 1, pp. 112–125, 2016.

[4] G. Peng, U. Tisch, O. Adams, M. Hakim, N. Shehada, Y. Y. Broza, S. Billan, R. Abdah-Bortnyak, A. Kuten, and H. Haick, "Diagnosing lung cancer in exhaled breath using gold nanoparticles," *Nature nanotechnology*, vol. 4, no. 10, p. 669, 2009.

[5] D. R. Wijaya, R. Sarno, E. Zulaika, and S. I. Sabila, "Development of mobile electronic nose for beef quality monitoring," *Procedia Computer Science*, vol. 124, pp. 728–735, 2017.

[6] A. Loutfi, S. Coradeschi, G. K. Mani, P. Shankar, and J. B. B. Rayappan, "Electronic noses for food quality: A review," *Journal of Food Engineering*, vol. 144, pp. 103–111, 2015.

[7] L. S. Sironi, L. Eusebio, L. Capelli, M. Remondino, and R. Del Rosso, "Use of an electronic nose for indoor air quality monitoring," *CHEMICAL ENGINEERING*, vol. 40, 2014.

[8] M. Z. Iskandarani, "A novel odor key technique for security applications using electronic nose system," *American Journal of Applied Sciences*, vol. 7, no. 8, p. 1118, 2010.

[9] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[10] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 3551–3558.

[11] F. A. Spanhol, L. S. Oliveira, C. Petitjean, and L. Heutte, “Breast cancer histopathological image classification using convolutional neural networks,” in *2016 international joint conference on neural networks (IJCNN)*. IEEE, 2016, pp. 2560–2567.

[12] D. Wu, D. Luo, K.-Y. Wong, and K. Hung, “POP-CNN: Predicting odor pleasantness with convolutional neural network,” *IEEE Sensors Journal*, vol. 19, no. 23, pp. 11 337–11 345, 2019.

[13] S. Xu, X. Sun, H. Lu, and Q. Zhang, “Detection of type, blended ratio, and mixed ratio of pu’er tea by using electronic nose and visible/near infrared spectrometer,” *Sensors*, vol. 19, no. 10, p. 2359, 2019.

[14] A. Hidaka and T. Kurita, “Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks,” in *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, vol. 2017. The ISCIE Symposium on Stochastic Systems Theory and Its Applications, 2017, pp. 160–167.

[15] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” *arXiv preprint arXiv:1811.03378*, 2018.

[16] D. Scherer, A. Müller, and S. Behnke, “Evaluation of pooling operations in convolutional architectures for object recognition,” in *International conference on artificial neural networks*. Springer, 2010, pp. 92–101.

[17] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[19] C. Temizel, C. H. Canbaz, O. Saracoglu, D. Putra, A. Baser, T. Erfando, S. Krishna, and L. Saputelli, “Production forecasting in shale reservoirs using LSTM method in deep learning.” Unconventional Resources Technology Conference (URTEC), 2020.

[20] J. Guo, “Backpropagation through time,” *Unpubl. ms., Harbin Institute of Technology*, vol. 40, 2013.

[21] C. Tallec and Y. Ollivier, “Unbiasing truncated backpropagation through time,” *arXiv preprint arXiv:1705.08209*, 2017.

[22] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] F. A. Gers and J. Schmidhuber, “Recurrent nets that time and count,” in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3. IEEE, 2000, pp. 189–194.

[24] Łukasz Kaiser and I. Sutskever, “Neural GPUs learn algorithms,” 2015.

[25] J. C. de Goma, R. J. Bautista, M. A. J. Eviota, and V. P. Lopena, “Detecting red-light runners (RLR) and speeding violation through video capture,” in *2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*, 2020, pp. 774–778.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[27] R. Girshick, “Fast R-CNN,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[28] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[30] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[31] J. Santos, M. Garcıa, M. Aleixandre, M. Horrillo, J. Gutierrez, I. Sayago, M. Fernandez, and L. Ares, “Electronic nose for the identification of pig feeding and ripening time in iberian hams,” *Meat science*, vol. 66, no. 3, pp. 727–732, 2004.

[32] S. Buratti, S. Benedetti, M. Scampicchio, and E. Pangerod, “Characterization and classification of italian barbera wines by using an electronic nose and an amperometric electronic tongue,” *Analytica Chimica Acta*, vol. 525, no. 1, pp. 133–139, 2004.

[33] P. Saha, S. Ghorai, B. Tudu, R. Bandyopadhyay, and N. Bhattacharyya, “Multi-class support vector machine for quality estimation of black tea using electronic nose,” in *2012 Sixth International Conference on Sensing Technology (ICST)*. IEEE, 2012, pp. 571–576.

[34] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[36] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[39] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[40] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[41] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” 2016.

[42] C. Esteves, G. M. Santos, C. Alves, S. I. Palma, A. R. Porteira, H. M. Costa, V. D. Alves, B. M. M. Faustino, I. Ferreira, H. Gamboa *et al.*, “Effect of film thickness in gelatin hybrid gels for artificial olfaction,” *Materials Today Bio*, vol. 1, p. 100002, 2019.