

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automatic classification of *Drosophila*

Daniela Maria da Silva Tavares



International Master in Computer Vision

Supervisor: Jorge Manuel de Sousa Basto Vieira

Co-Supervisor: Cristina Alexandra Gonçalves Paula Vieira

Co-supervisor: Miguel Fernando Paiva Velhote Correia

Co-supervisor: Alba Nogueira Rodríguez

August 3, 2023

Automatic classification of *Drosophila*

Daniela Maria da Silva Tavares

International Master in Computer Vision

August 3, 2023

Resumo

Espécies do género *Drosophila*, nomeadamente *D. melanogaster*, têm sido usadas como espécies modelo para elucidar os principais mecanismos da vida, bem como no estudo de diversas doenças, como por exemplo Alzheimer e Parkinson. Na área da Genética, *D. melanogaster* está associada às maiores descobertas. Não é por isso surpreendente que esta espécie seja considerada um organismo modelo. Contudo, o estudo de populações naturais tem sido também essencial para entender a função, estrutura e evolução do genoma, a adaptação das espécies a mudanças climáticas, bem como questões fundamentais da genética populacional como evolução neutral, seleção, recombinação, inversões, demografia, o papel dos elementos transponíveis, entre outras questões fundamentais da Biologia. Muitas destas questões requerem dados de genómica de múltiplos indivíduos da mesma espécie (Poolseq sequencing) para poderem ser respondidas. Assim, a identificação correta da espécie usada é fundamental, embora este não seja um procedimento trivial.

Para facilitar a identificação de espécies de *Drosophila*, nesta dissertação, usaram-se métodos de Processamento de Imagem e Redes Neurais para classificar as espécies, usando fotografias de várias espécies de *Drosophila*, nomeadamente de *D. americana*, *D. melanogaster* e *D. novamexicana*. A metodologia adoptada consiste em duas fases. Na primeira fase foram usadas técnicas de Processamento Digital de Imagem, usando filtros e combinando alguns desses filtros com técnicas de *thresholding*. Na segunda fase foram usados métodos de *Deep Learning*, mais precisamente Redes Neurais Convolucionais. Esta metodologia baseada em filtros é inovadora pois em trabalhos anteriores usaram-se *keypoints* em *Deep Learning*.

Concluiu-se que, embora os resultados das imagens com fundo sejam melhores, que os resultados das imagens sem fundo, estes últimos são mais fidedignos. Comparando as redes neurais usadas, DenseNet-121, EfficientNet-b0, e ResNet-50, e usando as imagens sem fundo como dados de entrada, foi com a primeira destas que se obtiveram os melhores resultados; uma veracidade de 78%, uma precisão de 71%, uma sensibilidade de 72% e uma *F1-score* de 71%. De entre os pré-processamentos feitos, nomeadamente a aplicação de filtros Gaussianos, de mediana, bilateral e da técnica de *unsharp masking*, a que mostrou melhores resultados para as imagens sem fundo foi o filtro Gaussiano. Com este método, aplicado à DenseNet-121, obteve-se uma veracidade e uma precisão ambas de 88%, e uma sensibilidade e *F1-score* ambos de 87%. Estes resultados são superiores aos já citados, em que não foi aplicado nenhum pré-processamento às imagens.

Como trabalho futuro, pretende-se melhorar o programa, para que seja possível classificar um maior número de espécies de *Drosophila*. Também se pretende diminuir o overfitting de alguns dos métodos usados.

Abstract

Drosophila species, namely *D. melanogaster*, have been used as model species to elucidate the principal life mechanisms, as well as the study of several diseases, such as Alzheimer and Parkinson. In genetics, *D. melanogaster* is associated with some of the greatest discoveries. Therefore, it is not surprising that this specie is considered a model organism. However, the study of natural populations has also been essential to understand the role, structure, and evolution of the genome, the adaptation of this species to climatic changes, as well as fundamental questions of populational genetics, such as neutral evolution, selection, recombination, inversions, demography, the role of transposable elements, among other Biology fundamental questions. Many of these questions require genetic data of multiple individuals of the same specie (Poolseq sequencing) to be answered. In this way, the correct identification of the specie being used is critical, although this is not a trivial procedure.

To facilitate the identification of *Drosophila* species, in this thesis, Image Processing and Deep Convolutional Networks were used to classify the species, using data of various *Drosophila* species, namely *D. americana*, *D. melanogaster*, and *D. novamexicana*. This methodology based in filters is innovative, as in the previous works keypoints were used.

We conclude that, whereas the results with images that had background was better than the results with images with no background, the latter results are more trustworthy. Comparing the Neural Networks used, DenseNet-121, EfficientNet-b0, and ResNet-50, and using images with no background as input, the best results were achieved by DenseNet-121; an accuracy of 78%, a precision of 71%, a recall of 72%, and a F1-score of 71%. Among the procedures made, namely the use of Gaussian, median, and bilateral filters, and unsharp masking, the one that provided the best results was the Gaussian filter. Applying this method to DenseNet-121, it was obtained both an accuracy and a precision of 88%, and both a recall and a F1-score of 87%. These results are higher than the ones already cited, where no preprocessing was applied to the images.

As future work, it is aimed to improve the program, so it can be used to classify more *Drosophila* species. It is also intended to diminish the overfitting of some of the methods used.

Acknowledgements

I want to start by thanking Cristina Vieira and Jorge Vieira, from Instituto de Investigação e Inovação em Saúde, i3S, for all the support, and for giving me the opportunity to use the laboratory to obtain the dataset.

I want to thank Professor Miguel Correia, from FEUP, for helping me during this journey, and for all the great ideas and advice to continuously improve my thesis.

I want to thank my co-supervisor Alba Rodríguez for the support, and Hugo López Fernández for the support he gave me, even though he was not my co-supervisor.

I want to thank all professors from all the universities in this Master's. A special thanks go to Professor Lucía Ramos García from Universidad de la Coruña for helping me during the first year of the Master.

Even though I am currently not a student at FCUP (Faculdade de Ciências da Universidade do Porto), I want to thank the whole team from the Mathematics department for always helping me during my undergraduate path, especially Professor José Carlos Santos for everything he has done for me, not only course related, but also with extracurricular activities.

I want to thank Diogo for all the love and support he gave and for always believing I was capable.

I want to thank Cláudia Pinto for being my first friend, and for all the moments we have been through.

Finally, I want to thank my family, my brother, my father, and my mother for everything they have done for me, for giving me all the conditions to handle this phase, giving me the love and support I always needed in the good and bad moments.

Contents

Abbreviations	x
1 Introduction	1
1.1 Importance of <i>Drosophila</i>	1
1.2 Objectives	3
1.3 Thesis Outline	3
2 Related Work	4
3 Background	8
3.1 Computer Vision	8
3.1.1 Thresholding	8
3.1.2 Filtering	9
3.1.3 Lowpass Filters	10
3.1.4 Highpass Filters	11
3.1.5 Nonlinear Filters	12
3.1.6 Peak Signal-to-Noise Ratio	13
3.2 Machine Learning and Deep Learning	13
3.2.1 Types of Learning	14
3.2.2 The Perceptron	14
3.3 Types of Artificial Neural Networks	15
3.3.1 Feed-Forward Networks	15
3.3.2 Recurrent Neural Networks	16
3.3.3 Convolutional Neural Networks	16
3.4 Training Algorithms for ANNs	17
3.4.1 Evolutionary algorithms	17
3.4.2 The Gradient Descent Algorithm	18
3.4.3 The Backpropagation Algorithm	18
3.5 Common Convolutional Networks	20
3.5.1 ResNet	20
3.5.2 EfficientNet	21
3.5.3 DenseNet	22
3.6 Evaluation metrics	23
4 Methodology	25
4.1 Dataset	25
4.2 Processing pipeline	25
4.3 Background Removal	27

4.4	Smoothing	27
4.5	CNNs	28
5	Results and Discussion	30
5.1	Experiments with the original images	30
5.1.1	DenseNet-121	30
5.2	Experiments with images with no background	31
5.2.1	With no preprocessing	31
5.2.2	With preprocessing	35
5.2.3	Global results	39
5.3	Overall Discussion	40
6	Conclusions	42
A	Novikoff's Theorem	43
B	Tutorial to use the program	45
	References	47

List of Figures

1.1	<i>Drosophila</i> phylogeny tree	2
2.1	Dual-channel model (from Yim and Sohn (2017))	7
3.1	Example of a convolution operation, reproduced from Podareanu et al. (2019) . .	10
3.3	Plot of the ideal highpass filter Jobling (2018)	11
3.4	Processing of unsharp masking, reproduced from Gonzalez and Woods (2008) . .	12
3.5	Differences between AI, ML and DL, AI- (2018)	13
3.6	Architecture of a Perceptron (from Rosebrock (2021))	14
3.7	Architecture of an ANN, reproduced from TIBCO	15
3.9	A CNN with 1 hidden layer with 2 nodes, reproduced from TIBCO	19
3.10	ResNet building block, reproduced from He et al. (2015)	20
3.11	ResNet-18 architecture, reproduced from Ramzan et al. (2019)	21
3.12	ResNet-50 architecture, reproduced from Challa and Vaishnav (2020)	21
3.13	Scale each of the parameters separately vs compound scaling method, reproduced from Tan and Le (2019)	22
3.14	Efficient Net basic architecture, reproduced from Tan and Le (2019)	22
3.15	EfficientNet-b0 architecture, reproduced from Tan and Le (2019)	22
3.16	DenseNet approach, reproduced from Hassan et al. (2021)	23
3.17	Example of a DenseNet, reproduced from Badgujar (2021)	23
3.18	DenseNet-121 architecture, reproduced from Radwan (2019)	23
4.1	Selected <i>Drosophila</i> species.	25
4.2	Processing pipeline	26
4.3	<i>D. melanogaster</i> before and after background removal	27
4.4	PSNR variation according to kernel size for Gaussian filter, median filter, and unsharp masking	28
4.5	PSNR variation, according to sigma, for bilateral filtering	28
4.6	Filtered images – with the parameters described in the subsection 4.4	29
5.1	DenseNet-121	30
5.2	Confusion matrix for DenseNet-121	31
5.3	DenseNet-121	32
5.4	Confusion matrix for DenseNet-121	32
5.5	ResNet-50	33
5.6	Confusion matrix for ResNet-50	33
5.7	EfficientNet-b0	34
5.8	Confusion matrix for EfficientNet-b0	34
5.9	Gaussian filter	35

5.10	Confusion matrix for Gaussian filter as preprocessing	36
5.11	Median filter	36
5.12	Confusion matrix for DenseNet-121 with median filtered images	37
5.13	Unsharp masking	37
5.14	Confusion matrix for DenseNet-121 with unsharp masked images	38
5.15	Bilateral filter	38
5.16	Confusion matrix for DenseNet-121 with bilateral filter as preprocessing	39

List of Tables

5.1	DenseNet-121 results	30
5.2	DenseNet-121 results	31
5.3	ResNet-50 results	32
5.4	EfficientNet-b0 results	33
5.5	Model's comparison for images with no background	34
5.6	DenseNet-121 results for Gaussian filtered images	35
5.7	DenseNet-121 results for median filtered images	36
5.8	DenseNet-121 results for unsharped images	37
5.9	DenseNet-121 results for bilateral filtered images	38
5.10	Comparison of the preprocessing techniques	39

List of Algorithms

1	The Perceptron Algorithm	15
2	Evolutionary algorithms general approach	18
3	Gradient Descent	18

Abbreviations

Acc	Accuracy
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
FN	False Negatives
FP	False Positives
LR	Learning Rate
ML	Machine Learning
ReLU	Rectified Linear Unity
S_{xy}	neighborhood centered at the point of coordinates (x, y)
TN	True Negative
TP	True Positives

Chapter 1

Introduction

Drosophila is a very large group of well over 1500 described fly species (Markow, 2006) and has been widely used in Biology and Medicine for several reasons, such as being easy to keep in the lab, its high reproduction, its low mean lifetime – which means that experiments take less time–, and its low cost. Besides that, it shares genes, tissues, and even organs, with mammals (Matthews and Gelbart, 2005).

1.1 Importance of *Drosophila*

Woodworth was the first to suggest that *Drosophila* could be used in genetics¹. However, the first person to really use it was Thomas Hunt Morgan, who discovered a mutation in *D. melanogaster* that leads to white eyes, instead of red. Based on Thomsons' work, Hermann Joseph Muller, a collaborator of his, showed that exposure to X-rays produces genetic mutations. More recently, in 2011, Jules Hoffman and Bruce Beutler, discovered how receptors detect microorganisms and activate our innate immunity. In 2017, Jeffrey Connor Hall, Michael Rosbash, and Michael Warren Young made discoveries about the molecular mechanisms that control the circadian rhythm². Other important discoveries include the discovery of *Notch* gene, by Poulson, that causes a malformation in *Drosophila*'s wings; the discovery of the embryonic development of the nervous system, by Nusslein-Volhand and Wieschaus; the behavioural responses of *Drosophila* in countercurrent assays; and, in 2000, the *Drosophila* full genome sequence, by Craig Venter's team (Stephenson and Metcalfe, 2013).

In today's research, *Drosophila* has a fundamental role. It is a model organism for the study of several diseases, such as Alzheimer, Parkinson, epileptic encephalopathy, Autism spectrum disorders, and obesity, among others. It is also a model organism for genetics and inheritance, embryonic development, organ regeneration, learning, behaviour, and ageing (Yamaguchi and Yamamoto, 2022). *Drosophila* is also used in Bioengineering, for the production of artificial tissues, and in the clinical drug discovery process (Jennings, 2011). The different *Drosophila*

¹<https://www.nobelprize.org/prizes/medicine/1933/morgan/biographical>

²<https://www.nobelprize.org/drosophila/>

species can be distinguished by several morphological features. For instance, *D. americana* has an overall dark body, whereas the closely related *D. novamexicana* displays a derived light body. Besides that, *D. melanogaster* wings are evenly pigmented throughout the wing blade (see Fig. 1.1) in contrast to other species (Massey and Wittkopp, 2016).

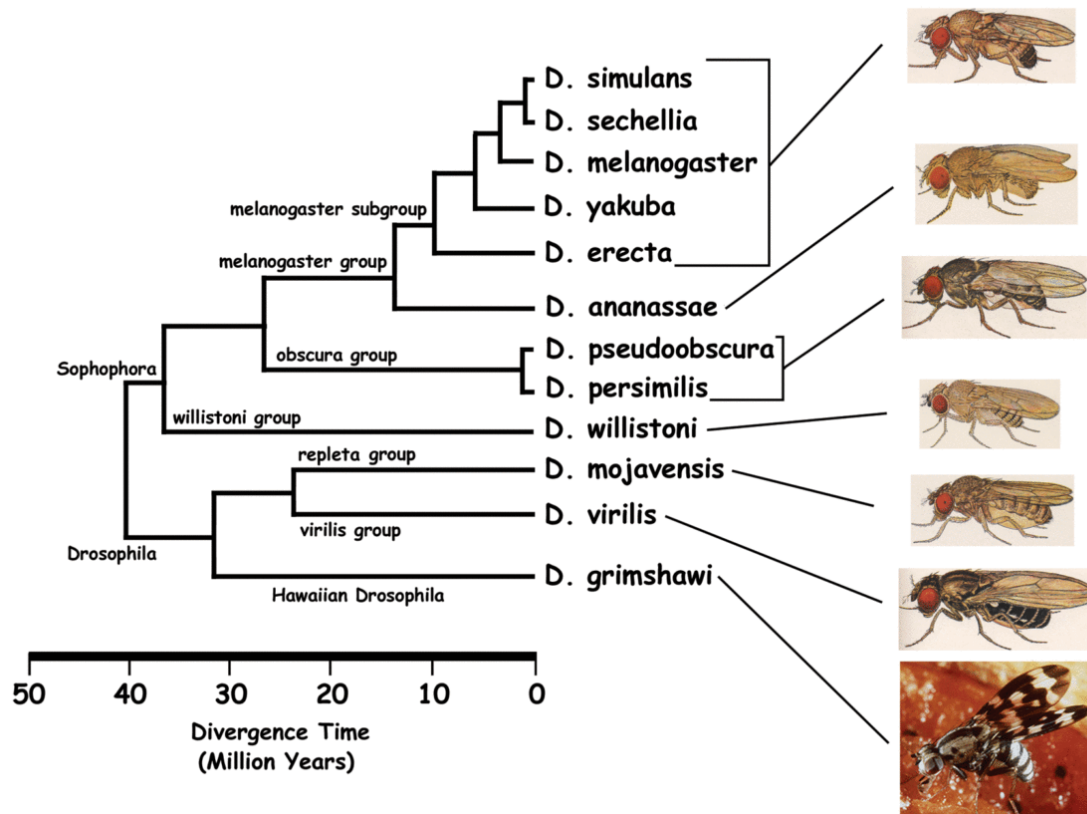


Figure 1.1: *Drosophila* phylogeny³

In this thesis, we are going to compare *Drosophila americana*, *Drosophila novamexicana*, and *Drosophila melanogaster*. *D. americana* and *D. novamexicana* are from virilis group and have been diverging for 40 million years (Reis et al., 2020). *D. melanogaster* is from melanogaster group. Even though the inter-variance is relatively low, as the features are the same in all species (for example, head, abdomen, eyes, and wings), it is possible to distinguish them, as it can be seen by Fig. 1.1. Some features, such as, eye size is an incomplete dominant trait that is largely affected by overall body size (Reis et al., 2020).

A *spatial filter* is an operation where it is chosen a centre pixel, and operations are performed in its neighbourhood, where the size of the neighbourhood is defined. The result of this operation is the output at that point. This process is repeated for every pixel. If all the performed operations are linear, we say that it is a *linear spatial filter*, otherwise we say it is a *nonlinear filter*.

³<https://cre.syr.edu/2018/03/26/national-science-foundation-recommends-funding-to-support-drosophila-evolutionary-phenomics-research-by-the-cre/>

Convolution Neural Networks (CNNs) are consecutive operations of filtering and convolution, where data features are learned. For that to work, some hyperparameters are needed. These include the definition of a loss function, a learning rate, an optimizer, the number of epochs, and the batch size. The loss function is a criterion that measures how far the model at the actual instant is from what is intended. There are various types of losses, depending on the problem at hand; the learning rate is the step size of the parameter to be updated; optimizer is a mathematical method so the function converges to an optimal solution; batch size controls, in the gradient descent algorithm (see 3.4.2), the number of images that get into the model before the weights are updated; finally, the number of epochs controls, in the gradient descent algorithm, and the number of complete passes in the training set.

For example, suppose that we have 4000 images/samples and that we choose a batch size of 4 and 20 as the number of epochs. This means that the model will have $4000/4 = 1000$ batches, each with 4 samples, and consequently that the weights will be uploaded after each batch of 4 samples. It also means that the model will pass through the training dataset 20 times, and a total of $1000 \times 20 = 20000$ during the entire training.

1.2 Objectives

The objective here is to classify different species of *Drosophila*. As *Drosophila* is a small insect, that brings us the difficulty of analysing it. Some species may be easy to differentiate, but others may only be distinguishable by their genitalia (Xia et al., 2018), and, because of that, a dataset was created.

1.3 Thesis Outline

In this chapter the state of the art is reviewed, as well as some concepts of Image Processing. Chapter 2 lays down the background needed for the thesis. Some basic techniques, principles, and concepts are explained, both for Computer Vision, and Machine Learning and Deep Learning. In Chapter 3, we present the method used to solve the problem. In Chapter 4, the results are presented and discussed. Finally, in Chapter 5, we draw conclusions. In Appendix A, it is presented a Theorem that proves that the Perceptron algorithm converges in a finite number of steps, and in Appendix B it is provided a tutorial to use the program created.

Chapter 2

Related Work

Deep Learning has provided us with excellent results both for regression and classification problems. Thereby it is increasingly easy to find Deep Learning models for biological applications. *Drosophila* classification is no exception and that is quite noticeable. The search for the related work grouped by the works that were related with *Drosophila*, and by the works that were not about this theme, but used either filtering as preprocessing, or used a Deep Learning architecture that we also used.

Cao et al. (2020) proposed a method based on transfer learning for insect recognition. The dataset contained 9 species of insects – mythimna separata, rice borer, rice plant hopper, mole cricket, mantis, locust, grass fly, ladybug, and ditch beetle. Firstly, they start by applying between 0 and 5 preprocessing techniques to each image. In total, they have 10 options for preprocessing: flip and transform the image; randomly use one of Gaussian blur, mean blur and median blur to enhance the picture; sharpening; edge detection superimposed on the original image; noise disturbance with Gaussian noise added to the image; rotation or reflection transformation; contrast transformation to change the contrast of the entire image to half or double the original; change RGB into grayscale image and multiply it by alpha to add to the original image; move the pixels to the surrounding area; distort local areas of the image. To expand their dataset, they used a Deep Convolution Generative Adversarial Network (DCGAN)¹. The transfer learning was based on Imagenet’s dataset. The neural networks they used before applying this technique were VGG16, VGG19, InceptionV3, and InceptionV4. They also used in their pipeline various digital image processing techniques, such as Gaussian, mean, and median blur; sharpening; edge detection; noise disturbance (add Gaussian noise to the picture); and contrast transformation. the best result was achieved by VGG19, with an accuracy of 97.39%.

Le et al. (2020) propose a method to predict the landmarks manually, and create a Convolutional Neural Network (CNN), the EB-Net, which builds in a modular way the concept of “Elementary Blocks”, each made up of usual CNN layer types. First, they asked entomologists to position manually the landmarks on digital images. Then, they created their automatic approach. They start by using a custom data augmentation procedure; trained and tested EB-Net, based on a dataset of

¹<https://developer.ibm.com/articles/generative-adversarial-networks-explained/>

different parts of carabids (pronotum, head, and elytra). To evaluate, and improve, the obtained results, they either train the model from scratch, or fine-tune it. Comparing the distances between the manual landmarks and the ones obtained by the automatic method, they concluded that their method can replace the manual landmarking.

Kumar et al. (2022), created an automated pipeline to classify *Drosophila* species based on the wings, as wings are a powerful genetic model for studying cellular and developmental processes. With this in mind, they created MAPPER, which is an ML-based pipeline that quantifies high-dimensional phenotypic signatures, with each dimension quantifying a unique morphological trait of *Drosophila*'s wings. The MAPPER's first module is based on Deep Learning, and segments the images to separate the foreground from the background. They start by training a U-Net, train the last few layers of their model with a Convolutional Neural Network, and finally use K-Nearest Neighbours method to classify the intervein regions. Their results overcome the ones obtained with thresholding, and the active contours method. The second method is a Machine Learning based pixel classifier for the same task. They concluded that MAPPER's automated measurements are statistically equal to manual measurements.

Luo et al. (2022), tried to identify ticks, as these are vitally important to assess threats. However, most approaches are based on taxonomy, and because of that are time-consuming and require expertise skills. The images were obtained from a tick passive surveillance program that receives ticks from public individuals, partnering agencies, or veterinary clinics. It consists of 12000 high-resolution micrographs. All species were molecularly confirmed by a species-specific TaqMan PCR assay, so there is no human error in the distinction. The program distinguishes between *Amblyomma americanum*, *Dermacentor variabilis*, and *Ixodes scapularis*. They applied Transfer Learning to pretrained Convolutional Neural Networks, specifically VGG-16, ResNet-50, InceptionV3, MobileNetV2, and DenseNet-121. The best result was an accuracy of 99.5%, even though all the accuracies were greater than 98%.

Shen et al. (2021) two main objectives were to distinguish between Tephritid and Non-Tephritid fruit flies, as the first are important agricultural pests around the world, and classify between Anastrepha, Ceratitis, Rhagoletis, and Bactrocera, the 4 major species of Tephritid. For the creation of the dataset, they used iNaturalist fruit fly database. They cleaned the data using InceptionV3 neural network, and used K-means clustering to identify between the four species. They obtained 95.44% accuracy for EfficientNet-b0 to identify Tephritid fruit flies, and 89.65% accuracy for classifying representatives of the major tephritid genus.

Using more traditional methods, Loh et al. (2017) identified 16 *Drosophila* species. They created a system to detect and measure keypoints and vein segments from *Drosophila*'s wings. For this purpose, they used image transformations, as well as template matching. The accuracy was 94% in identifying the correct species, and 56% in identifying the correct sex. Badirli et al. (2021), made a more general approach, in the sense that they tried to classify insects, but not just *Drosophila* or ticks. They classified 32848 instances of insects of 1040 described insect species. DNA-based methods have aided in providing additional evidence of separate species, however expertise knowledge is needed, so they created a Bayesian Deep Learning method, based on hier-

archy of species around corresponding genera and uses deep embeddings of images and barcodes together to identify insects at the lowest level of abstraction possible, for the open-set classification of species. They got an accuracy of 96.66% classifying the instantiated species, and an accuracy of 81.39% identifying genera of undescribed species.

[Şaban Öztürk and Akdemir \(2018\)](#), classified histopathological images of cancer vs non-cancer patches. The dataset used was the one of CAMELYON challenge, which consists of 400 whole-slided images for breast cancer detection. It was used a similar approach to the one of this thesis. They preprocess the images in four different ways, (1) no preprocessing, (2) normal preprocessing, that consists in reducing background noise and in enhancing the cells; (3) other normal preprocessing techniques; (4) over preprocessing, that consists in applying a threshold, morphological operations, besides the normal preprocessing. The images are then used as input to AlexNet, which is trained by 500 epochs. The best results were obtained using normal preprocessing. With this study, they found that preprocessing algorithms make training faster and help to achieve better results, compared to not using it. However, excessive preprocessing may not help either.

Also in other biomedical applications, we find [Gielczyk et al. \(2022\)](#) and [Avşar \(2021\)](#). In [Gielczyk et al. \(2022\)](#) they used for preprocessing (1) none; (2) histogram equalization; (3) histogram equalization followed by a Gaussian filter, with a kernel of 5×5 ; (4) histogram equalization followed by a bilateral filter with $\text{diameter}=5$, $\sigma_{\text{color}} = \sigma_{\text{space}} = 5$ as parameters; (5) adaptive masking; (6) adaptive masking followed by histogram equalization and Gaussian blur. The results showed that the best preprocessing method was the last one referred, but that overall preprocessing improves the results. Without preprocessing, accuracy was of 93%, and F1-score was of 91–96% for every class, whereas with just histogram equalization, Precision, Recall, and F1-score raised by 2%. In [Avşar \(2021\)](#), the CNNs used were MobileNetV2 and EfficientNetB0, that were trained with and without the preprocessing techniques. The methods used were contrast limited adaptive histogram (CLAHE), unsharp masking, and Wiener filtering. With this study, they found out that the best results for both models were obtained with Wiener filter, although CLAHE and unsharp masking may improve the performance only with MobileNetV2. It was also concluded that there were not substantial improvements with more than 30 epochs.

Image processing is such a common method that there are groups using it in cultural studies. An example of this is [Kusrini et al. \(2022\)](#). A shadow puppet show of Indonesian culture, “Wayang Kulit” is facing the end of its existence, as there is almost no support from the government, among other factors. As the characters are not publicly available, they needed to create their own dataset with images similar to those used in the show. With this study, it is aimed to recreate the Punakawan puppet show with less costs. The group used filtering as preprocessing –that is classified as Gaussian or non-Gaussian–, VGG-16 as CNN, and KNN method for the classification task. Among all the experiments made, the researchers found that the combination of CLAHE, RGB images (as, in some experiments, only the green channel of the images is used), Gaussian filter, and thresholding was the process with the highest accuracy, 98.75%. It was also found that CLAHE improved the accuracy of green channel images. Besides that, this study found two ways

of reducing image noise, which were Gaussian filter, and median filter.

As the quality of the images used in Deep Learning models is one of the most influential factors for an high accuracy, we could not miss a study that tried to find a solution to obtain good results with low quality images, as for example JPEG images. That's what [Yim and Sohn \(2017\)](#) do. They suggest a generalised architecture of a dual-channel model (Fig. 2.1). The architecture receives

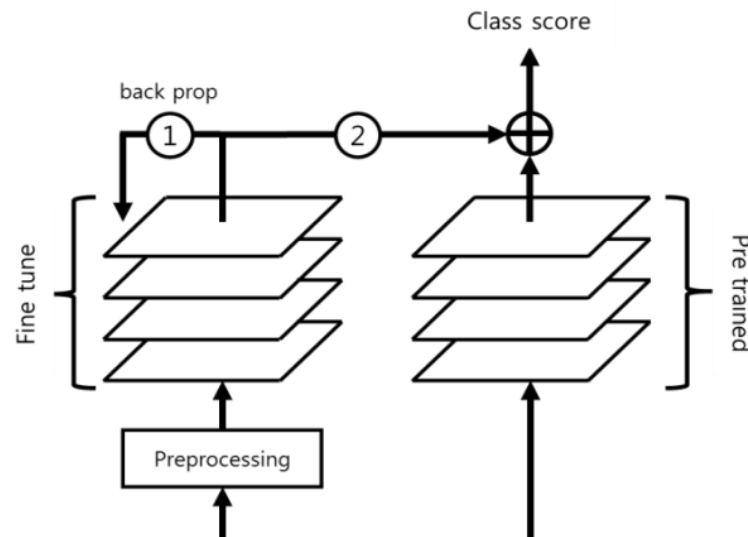


Figure 2.1: Dual-channel model (from [Yim and Sohn \(2017\)](#))

two inputs, the original images and the outline-enhanced images. The denoising methods used were (1) nonlocal filtering; (2) bilateral filtering; (3) total variation. This group obtained different conclusions than the previously discussed research. They found that a preprocessing method does not produce any significant improvement per se, but provides a stable accuracy with any dataset using the dual-channel structure.

Chapter 3

Background

3.1 Computer Vision

Computer Vision is an engineering-related area that tries to extract useful information from images and videos. The manual extraction of features is labour-intensive, expensive, and time-consuming, and the number of extracted features may not be enough for the project at hand. With Computer Vision, this process gets automated, and so, faster. Another advantage is that with technology's help, the datasets can be larger, and so the results can be more accurate, without so much subjectivity from experts in taking features. Below we review some of the most important concepts in this field. The definitions and equations of the following sections were taken from [Gonzalez and Woods \(2008\)](#) and [Mitra \(2001\)](#).

3.1.1 Thresholding

Global thresholding is a technique that divides the image into two components based on the pixel's intensity values. That division is done by choosing an “arbitrary” threshold value, T .

$$g(x,y) = \begin{cases} 1, & f(x,y) \geq T \\ 0, & f(x,y) < T \end{cases}$$

The pixels labeled as 1 are expectedly part of the foreground, and the ones labeled as zero would be part of the background.

Adaptive thresholding is the method where the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions.

Otsu's method gives us the best threshold automatically. The idea of the algorithm is to separate image pixels into foreground and background. We are assuming that the image is grayscale – which means that the image is normalized– and that its intensity distribution is bimodal. The

objective is to minimize the within-class variance, which is the same as maximizing the between-class variance. Otsu method relies on the second choice, and the expression is Eq. 3.1:

$$\sigma_w^2 = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \quad (3.1)$$

where:

q_1 and q_2 are the class probabilities given by

$$q_1(t) = \sum_{i=1}^T P_i \quad \& \quad q_2(t) = \sum_{i=T+1}^I P_i \quad (3.2)$$

the class means are given by

$$\mu_1(t) = \sum_{i=1}^T \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=T+1}^I \frac{iP(i)}{q_2(t)} \quad (3.3)$$

and finally, the individual class variances are given by

$$\sigma_1^2(t) = \sum_{i=1}^T (i - \mu_1(t))^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=T+1}^I (i - \mu_2(t))^2 \frac{P(i)}{q_2(t)} \quad (3.4)$$

3.1.2 Filtering

The concept of filtering comes from Signal Processing, where the method is used to transfer certain frequency components without any distortion and to block other frequency components (Mitra, 2001).

A *lowpass filter* is the one that transfers all the frequency components below a specified frequency ω_c , named *cutoff frequency*, and blocks all frequencies above it. A *highpass filter* does the opposite, so it transfers all frequency components above ω_c , and blocks all the ones below it. The range of frequencies that the filter transfers is called *passband*, and the range of frequencies that is blocked is called *stopband*.

The main concept in the filtering process is the *convolution operation*. This is defined, in images, by Eq. 3.5,

$$(w * f)(x, y) = \sum_{s=0}^a \sum_{t=0}^b w(s, t) f(x - s, y - t) \quad (3.5)$$

where w is a $a \times b$ filter. An example of this operation is given in Fig. 3.1.

One important concept in Image Processing is the one of *edge*. As said in Chapter 1, edge detection techniques were used in this thesis. We define an edge as a pixel where there is an abrupt change in intensity. To detect edges we use first and second-order derivatives because gradients point in the direction with the greatest rate of change. The first-order partial derivative is defined by Eq. 3.6

$$\frac{\partial f}{\partial x} = f(x + 1) - f(x) \quad (3.6)$$

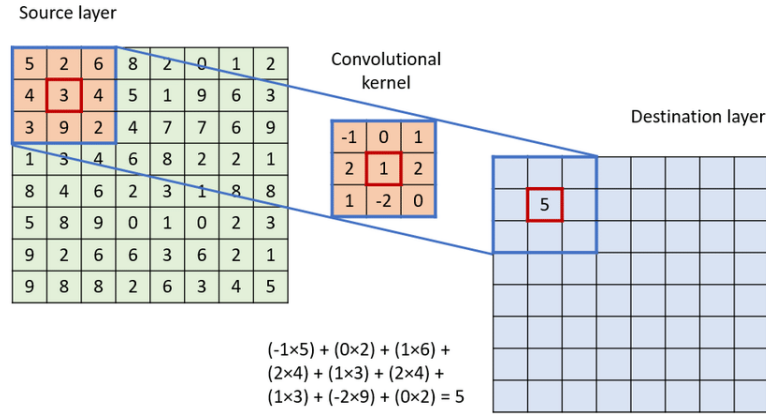


Figure 3.1: Example of a convolution operation, reproduced from Podareanu et al. (2019)

whereas the second-order partial derivative is defined as in Eq. 3.7

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} &= f'(x+1) - f'(x) \\ &= f(x+1) + f(x-1) - 2f(x) \end{aligned} \quad (3.7)$$

The difference in using the first or second-order derivative is in the sharpness, which is usually better with the second-order derivative. So, while we use gradients for first-order derivatives, we use the Laplacian operator to detect the second-order derivatives. Finally, we say that an edge has been detected if its intensity is higher than a threshold T .

3.1.3 Lowpass Filters

3.1.3.1 Ideal Lowpass Filter

The ideal lowpass filter is defined by Eq. 3.8.

$$H(\omega) = \begin{cases} 1 & , \omega \leq \omega_c \\ 0 & , \text{otherwise} \end{cases} \quad (3.8)$$

where ω is the frequency. The respective plot is illustrated in Fig. 3.2.

Lowpass filters smooth the image, including the edges, giving to the image a blurring effect. An example of lowpass filter is the Gaussian filter, whose kernel is given by Eq. 3.9.

$$w(x, y) = \frac{1}{\sqrt{2\pi\sigma_x\sigma_y}} e^{-\left(\frac{x-\mu_x}{2\sigma_x^2} + \frac{y-\mu_y}{2\sigma_y^2}\right)} \quad (3.9)$$

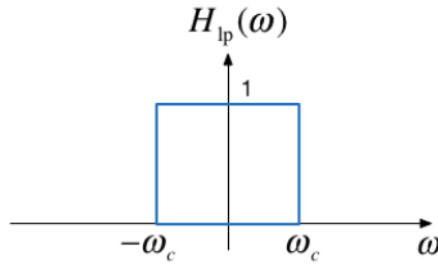


Figure 3.2: Plot of the ideal lowpass filter Jobling (2018)

3.1.4 Highpass Filters

3.1.4.1 Ideal Highpass Filter

The ideal highpass filter is defined by Eq. 3.10

$$H(w) = \begin{cases} 1 & , \omega > \omega_c \\ 0 & , \text{otherwise} \end{cases} \quad (3.10)$$

where ω is the frequency, and the respective plot is illustrated in Fig. 3.3.

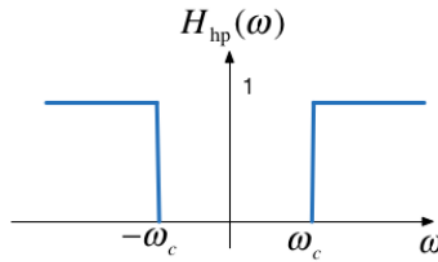


Figure 3.3: Plot of the ideal highpass filter Jobling (2018)

A highpass filter enhances the edges of an image. A technique for sharpening images is unsharp masking, and it is given by the following process:

1. Blur the image
2. Subtract the blurred image from the original one (this difference is called *mask*)
3. Calculate the sum between the original image and the weighted mask

The general equation for this procedure is given by equation Eq. 3.11

$$g(x,y) = f(x,y) + kg_{mask}(x,y) \quad (3.11)$$

where $f(x, y)$ is the original image, and g_{mask} is its mask. Do just note that blurring an image is the same as applying a lowpass filter. When doing $f(x,y) - kf_{blurred}(x,y)$ we obtain just the edges, as

the other pixels get 0-valued. We can then say that it is a highpass filter. Finally, when summing $f(x,y)$ and $f(x,y) - f_{blurred}(x,y)$, we obtain $2f(x,y) - kf_{blurred}(x,y)$. So, the 0-valued pixels that were near the edges, get the value $2f(x,y) - kf_{blurred}(x,y)$. This process is illustrated in Fig. 3.4.

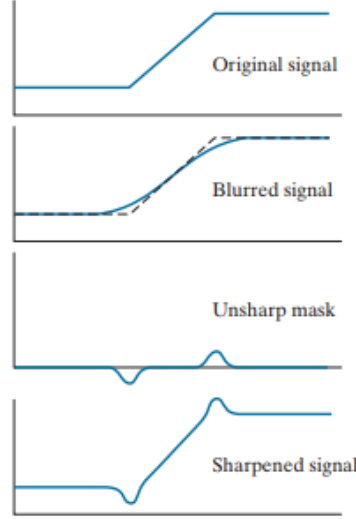


Figure 3.4: Processing of unsharp masking, reproduced from Gonzalez and Woods (2008)

3.1.5 Nonlinear Filters

3.1.5.1 Median Filter

Median filter replaces the center pixel value by the median value of a predefined neighbourhood. It is quite useful for salt-and-pepper noise, which is manifested by black and white dots over the image.

3.1.5.2 Bilateral Filter

Most of the times, sharpening filters smooth too much the areas where frequencies are low. If we want to avoid this, a good solution is bilateral filter. It is given by equations Eq. 3.12 and Eq. 3.13. Let I be a grayscale image, and p be the pixel location. We define the Bilateral Filter (BF) as in Eq. 3.12

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in \mathbb{S}} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) I_q \quad (3.12)$$

where W_p is a normalisation factor given by Eq. 3.13

$$W_p = \sum_{q \in \mathbb{S}} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) \quad (3.13)$$

σ_r and σ_s are measures of the amount of filtering for an image I ; G_{σ_s} is the spatial Gaussian that decreases the influence of distance pixels, and G_{σ_r} is the range (Gaussian that decreases the influence of pixels q with an intensity value different from I_p).

3.1.6 Peak Signal-to-Noise Ratio

The Peak Signal-to-Noise Ratio (PSNR) block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image¹. It is computed using Eq. 3.14.

$$PSNR = 10 \log_{10} \left(\frac{R^2}{MSE} \right) \quad (3.14)$$

The higher the PSNR, the better is the image after being processed. MSE stands for Mean Squared Error, between the two images that we are analyzing.

3.2 Machine Learning and Deep Learning

Artificial Intelligence (AI) is the ability of a machine to do human tasks². Machine Learning (ML) is the area of AI where the learning algorithms are created³, and Deep Learning (DL) is the area of ML “that involves training artificial neural networks with large amounts of data to recognize patterns and make predictions or decisions” (Hector Martinez, 2023). These concepts are Fig. 3.5.

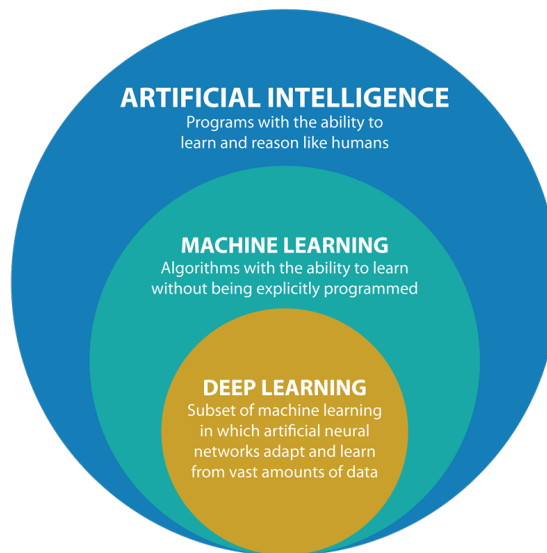


Figure 3.5: Differences between AI, ML and DL, AI- (2018)

¹<https://www.mathworks.com/help/vision/ref/psnr.html>

²<https://www.britannica.com/technology/artificial-intelligence>

³<https://www.ibm.com/topics/machine-learning>

3.2.1 Types of Learning

Unsupervised Learning In this type of learning all information is extracted from the data itself, we do not know what will be the output. This means that the output is part of just the pseudo-labelled set.

Supervised Learning In this type of learning, the data is already labelled and we know that the output will be in the input set of labels.

Semi-Supervised Learning In Semi-Supervised Learning, we do have just part of the data labelled. Then, during the training, according to a rule that is established by the person that is making the system, data is matched and the system creates new labels (also known as *pseudo-labels*). The output may be part of the input set of the labels or part of the pseudo-labels).

3.2.2 The Perceptron

Perceptron is the simplest type of network, actually being a linear binary classifier. An input is given, and the respective output is either 1 or -1, according to the prediction correctness (Algorithm 1). Its architecture is shown in Fig. 3.6.

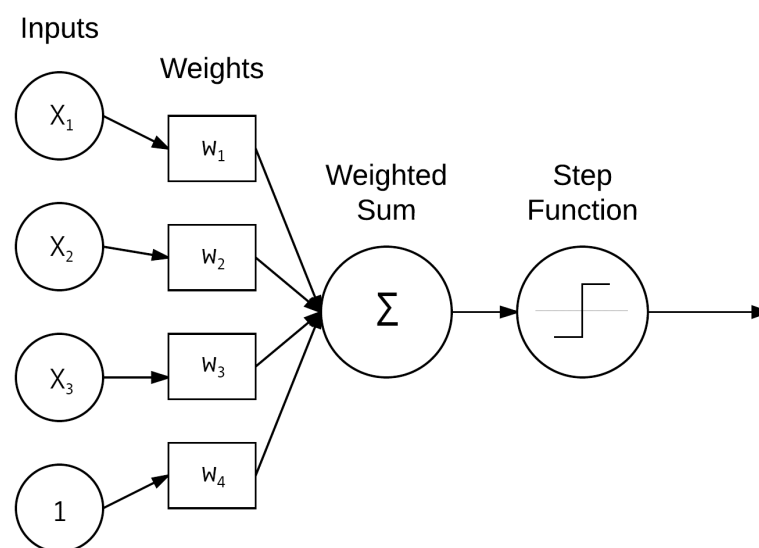


Figure 3.6: Architecture of a Perceptron (from [Rosebrock \(2021\)](#))

Algorithm 1 The Perceptron Algorithm

Input: An input set \mathbb{S} of tuples $\{(x_i, y_i)\}$ where $x_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$ is a label, and η is a positive scalar

Output: w and b so that when $w \cdot x + b > 0$, it's true that y_i is 1, otherwise it is -1

1: **Initialise:** $w = 0$, $b = 0$, $k = 0$, and $R = \max_i \|x_i\|$

2: **if** there's a mistake **then**

3: $k = k + 1$

4: $w = w + \eta y_i x_i$

5: $b = b + \eta y_i$

6: **if** there are no more errors **then**

7: **algorithm finishes**

8: **end if**

9: **end if**

3.3 Types of Artificial Neural Networks

3.3.1 Feed-Forward Networks

It is a system based on the human brain, where neurons are called *nodes*, and synapses are called *connections* (Fig. 3.7). A Feed-Forward Network is theoretically an agglomeration of perceptrons, although, as in real life most applications are non-linear, quite often the neurons are actually sigmoid-neurons instead of perceptrons⁴.

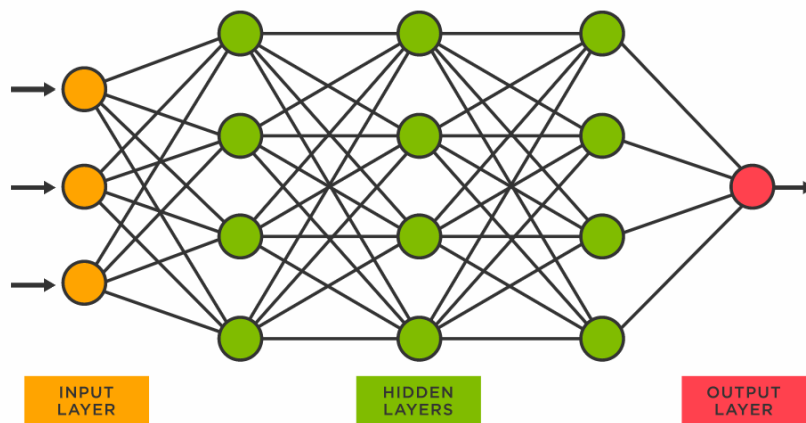


Figure 3.7: Architecture of an ANN, reproduced from TIBCO

Input is given, and with that, the respective output is computed through the formula

$$y = f(w^T \cdot x + b)$$

⁴<https://blog.adxy.dev/what-is-a-neural-network-visualising-understanding-a-neural-network-in-depth>

where y is the output, w is the vector of the weights connections, and b is a vector formed by the biases. Finally, f is a non-linear function, also known as *activation function*. Non-linearity is important in these models because it adds complexity to the model, which makes it more adaptable to new inputs.

Some well-known activation functions are the ones below:

- ReLU: $f(x) = \max(0, x)$
- Sigmoid/ Logistic activation function: $f(x) = \frac{1}{1+e^{-x}}$
- Tanh: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Softmax: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$, where x_i is the value of the i th-class of the model

3.3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a variant of CNNs that deal with sequential data. This means that their inputs and outputs are not independent of each other, as it happens for feed-forward networks (Fig. 3.8). Also, while feed-forward networks have weights across each node, RNNs share the same parameters within each layer of the network, which is saying that RNN architectures do have cycles, whereas feed-forward networks do not. RNNs are used, for example, in language translation, speech recognition, and video [Schmidt \(2019\)](#).

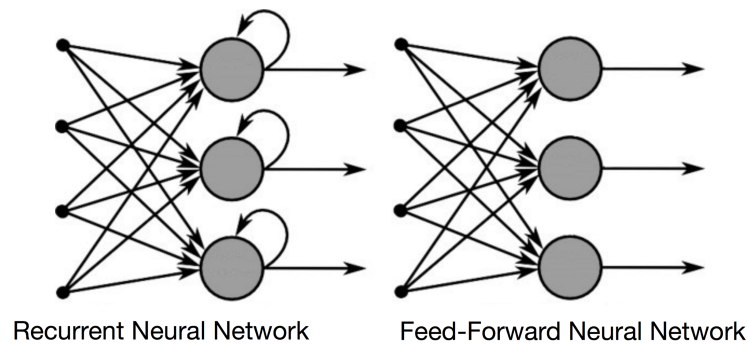


Figure 3.8: RNN vs Feed-Forward networks [Jeans \(2019\)](#)

3.3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are the method most often used in DL. Even though their architecture is quite similar to ANNs, they are mainly used in Computer Vision tasks, as they are proven to be highly effective in solving problems that use images as data input.

The different layers of a CNN are the following:

- Convolutional Layer: it is where the input image is convolved with a convolutional filter (also called a *kernel*). Some parameters of the convolutions are the following:
 - Padding: to add extra pixels 0-valued around the boundary of the image
 - Stride: when we want to downsample, we move our window more than one element at a time, skipping intermediate locations. It is the number of rows and columns traversed per slide.
- Pooling Layer: the maps between layers are sub-sampled, i.e. large-size feature maps are shrunk to create smaller feature maps, so we get just the most important information. In this way, there are fewer parameters to learn. The most common pooling functions are Alzubaidi et al. (2021):
 - Global Pooling: each channel in the feature map is reduced to a single value. The most common operations in global pooling are
 1. taking the maximum value (global max pooling)
 2. taking the minimum value (global min pooling)
 3. taking the average value (global average pooling - GAP)
 - Average pooling: compute the average value for each patch (typically 2×2) on the feature map
- Batch normalization: converts interlayer outputs into a standard format, called normalizing. This makes the model to be trained more efficiently.
- Fully Connected Layer: is usually at the end of the CNN. In this layer, each node is connected to the others. The output is the final result from this layer.

3.4 Training Algorithms for ANNs

3.4.1 Evolutionary algorithms

Evolutionary algorithms are unsupervised learning algorithms based on the theory of evolution. These algorithms have three fundamental aspects⁵:

- Population-based: all possible solutions from the algorithms are in one population. Encoding the solution in the population, it will make crossover and mutation much easier
- Fitness-oriented: the fitness function is used to evaluate partial solutions, to know how much the solution is right
- Variation-driven: solutions will vary when changing the solution space.

There are many types of evolutionary algorithms, but in general they have the structure of Algorithm 2.

⁵https://optimization.cbe.cornell.edu/index.php?title=Evolutionary_algorithms

Algorithm 2 Evolutionary algorithms general approach

Require: A population**Output:** A solution (in case it exists)

- 1: **Randomly generate the initial population of individuals (first generation)**
 - 2: **Evaluate the fitness of each individual in that population with the preferred fitness function**
 - 3: **while** Optimal solution is not found **do**
 - 4: Select the parents (best-fit individuals) for reproduction
 - 5: Breed new individuals through crossover and random mutation, giving “birth” to the next generation
 - 6: Use the fitness function to gauge the individual fitness of the new individuals
 - 7: Replace least-fit population with new individuals
 - 8: **end while**
-

3.4.2 The Gradient Descent Algorithm

Gradient Descent (Algorithm 3) is an optimization algorithm with the objective of minimizing the objective loss function used to train the model. For that, the gradient is computed, and the method reaches for a solution in its opposite direction.

Algorithm 3 Gradient Descent

Require: Loss function L , Maximum Number of Steps**Input:** Random value x_i for each parameter i **Output:** Optimal solution

- 1: **Take the gradient of L**
 - 2: **Plug x_i into the gradient**
 - 3: **while** Step size does not converge to 0 or Maximum Number of Steps is not reached **do**
 - 4: **Compute the step size:** $Step\ size = Slope \times LR$
 - 5: **Compute the new parameters:** $New\ Parameters = Old\ Parameters - Step\ size$
 - 6: **end while**
-

3.4.3 The Backpropagation Algorithm

The backpropagation algorithm is constituted of two phases, a forward pass and a backward pass. In the forward pass, the input is propagated through the NN, whereas in the backward pass, we start from the last layer, and from there, we compute the optimal parameter for each layer using the chain rule. These algorithms are computed in parallel with the Gradient Descent algorithm.

In the forward pass, we propagate the data in the input layer. This input is propagated through the hidden layer, measures the networks' predictions from the output layer, and computes the network error based on the predictions that the network has made. At this time, we enter in the backward process. In the backward pass, we start by propagating the error from the output layer until reaching the input layer, passing through the hidden layers. The process of propagating the network error from the output layer to the input layer, through the chain rule, is called *backward*

propagation, or simply *backpropagation*. The backpropagation algorithm is the set of steps used to update network weights to reduce the network's error.

For example, suppose there is one entry for the input, one hidden layer with two nodes with an activation function (the same), and one output. Let w_1, w_2, w_3 , and w_4 be the weights of the NN, and b_1, b_2 , and b_3 be its biases, as in Fig. 3.9. Also, let $f(x)$ be our activation function. We want these parameters to be optimal, so we get the curve that best fits our data.

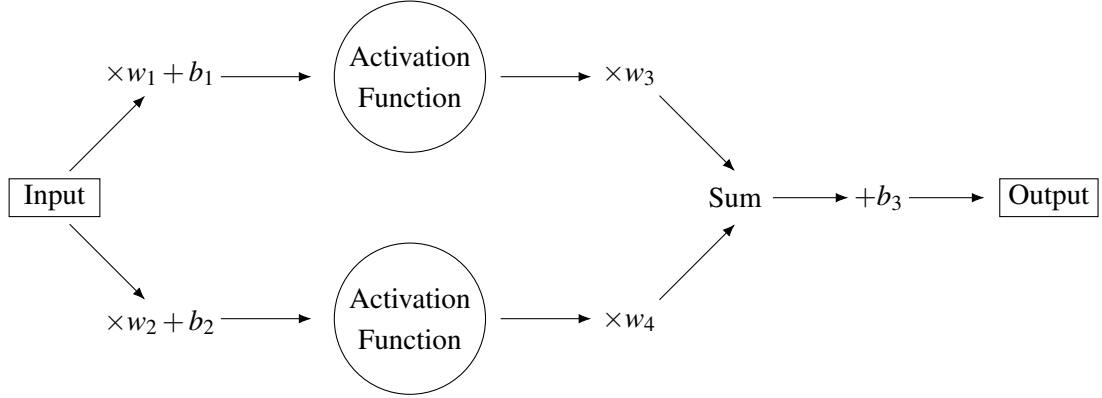


Figure 3.9: A CNN with 1 hidden layer with 2 nodes, reproduced from **TIBCO**

We start by initializing biases to 0, and we randomly choose values from a normal distribution for the weights.

From the forward propagation step, we know that our output is going to be given by $sum + b_3$, where sum is $f(Input_i \times w_1 + b_1)w_3 + f(Input_i \times w_2 + b_2)w_4 + b_3$. Then, we compute the Sum of the Squared Residuals (SSR). A *residual* is given by the difference of the predicted value from the observed value. SSR is given by equation Eq. 3.15.

$$SSR = \sum_{i=0}^n (Observed - Predicted)^2 \quad (3.15)$$

As we want to know how b_3 varies, and we know that b_3 is related to SSR through the predicted values, using the chain rule, we have Eq. 3.16.

$$\frac{dSSR}{db_3} = \frac{dSSR}{dPredicted} \times \frac{dPredicted}{db_3} \quad (3.16)$$

$$= \sum_i -2(Observed - Predicted) \times 1 \quad (3.17)$$

$$= \sum_i -2(Observed - Predicted) \quad (3.18)$$

Now we want to find the optimal value for w_3 and w_4 . So, we compute $\frac{dSSR}{dw_3}$ and $\frac{dSSR}{dw_4}$.

$$\frac{dSSR}{dw_3} = \frac{dSSR}{dPredicted} \times \frac{dPredicted}{dw_3} \quad (3.19)$$

We had already computed $\frac{dSSR}{dP_{Predicted}}$ in Eq. 3.16, we know that Eq. 3.19 is equal to $\sum_i -2(Observe - Predicted) \times f(Input \times w_1 + b_1)$. Analogously, we know that $\frac{dSSR}{dw_4}$ is equal to $\sum_i -2(Observe - Predicted) \times f(Input \times w_2 + b_2)$.

With the help of the chain rule, we also compute $\frac{dSSR}{dw_1}$, $\frac{dSSR}{db_1}$, $\frac{dSSR}{dw_2}$, and $\frac{dSSR}{db_2}$.

As we had initially chosen the values for each of the parameters, all we have to do now is to do an initial forward pass and apply the Gradient Descent algorithm to each one of the parameters to get the optimal values.

3.5 Common Convolutional Networks

3.5.1 ResNet

When CNNs were starting to gain popularity, there was the idea that the more layers the model had, the better it would perform. However, that is not always true, as there is a vanishing gradient problem. Vanishing gradients happen when doing backpropagation, because when doing the partial derivatives successively, the gradient starts getting too small, and the accuracy tends to infinity. With that in mind, ResNet was created [He et al. \(2015\)](#). With ResNet, instead of stacking more layers, we do an identity map (so, we do not learn any new features) and sum that output to the output of the previous layer. With this architecture, also called *residual blocks*, we do not lose so much information and the architecture does not get as deep. Apart from that, ResNet is quite similar to the structure explained in the previous chapter.

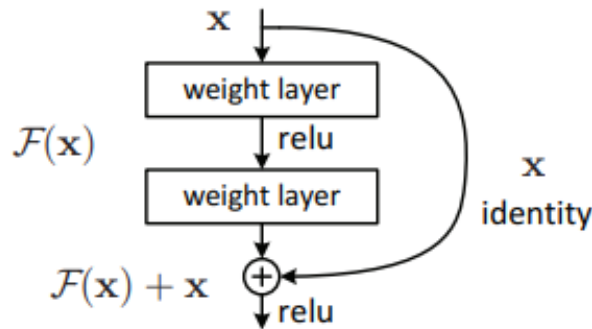


Figure 3.10: ResNet building block, reproduced from [He et al. \(2015\)](#)

3.5.1.1 ResNet-50

ResNet-50 is a 50-layer CNN, 48 convolutional layers, 1 max pooling layer, and 1 average pooling layer.

The main difference from the previous ResNets is that it has a bottleneck. A bottleneck is a 1×1 convolution. It reduces the number of parameters and multiplications, making the procedure faster.

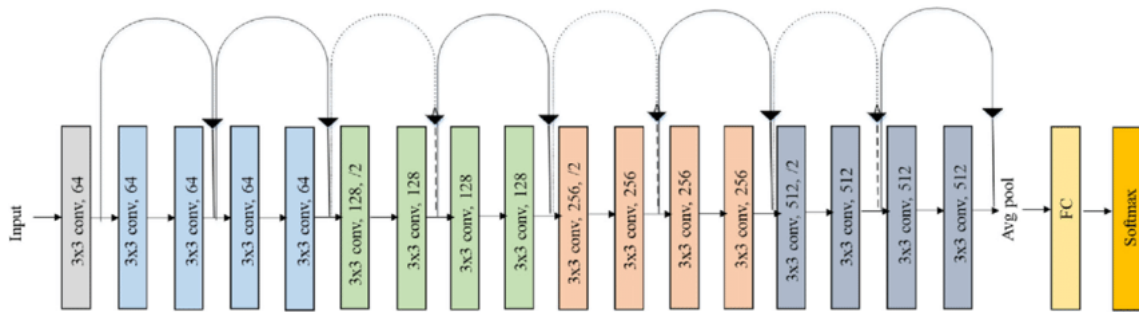


Figure 3.11: ResNet-18 architecture, reproduced from Ramzan et al. (2019)

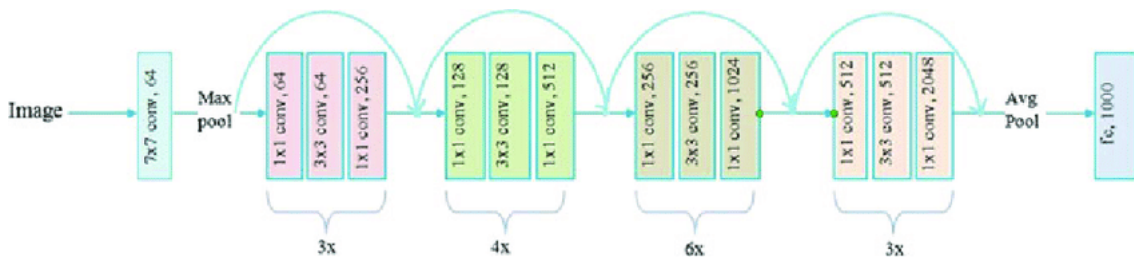


Figure 3.12: ResNet-50 architecture, reproduced from Challa and Vaishnav (2020)

According to Fig. 3.7, the input layer here is a convolutional layer (the 7×7 convolution layer) that accepts as input an RGB image. The hidden layers are all up to the average pooling layer, and the output layer is the FC layer.

3.5.2 EfficientNet

EfficientNet was constructed so there could be a way to scale up a CNN without the network getting too deep, thus getting high accuracy and efficiency. Usually, to overcome that, we try to adjust the height, width, depth, and resolution by trial and error and just adjusting one of the parameters each time. Here it was created a method, called *compound scaling method* (Fig. 3.13, so instead of randomly scaling up width, depth or resolution, compound scaling uniformly scales each dimension with a certain fixed set of scaling coefficients⁶.

3.5.2.1 EfficientNet-B0

According to the figure Fig. 3.15, the respective layers in Fig. 3.7 are the following: the input layer is the first 3×3 convolution, hidden layers are all the layers until the last MBConv block. A Mobile Inverted Residual Bottleneck Convolutional (MVConv) block is a series of operations such as depthwise convolutions, pointwise convolutions, and squeeze-and-excitation operations. These operations are aimed at reducing computational complexity while maintaining a high level of representation power. The output layer in EfficientNet-b0 consists of a combination of an FC layer and a softmax function.

⁶<https://medium.com/mllearning-ai/understanding-efficientnet-the-most-powerful-cnn-architecture-eaeb40386fad>

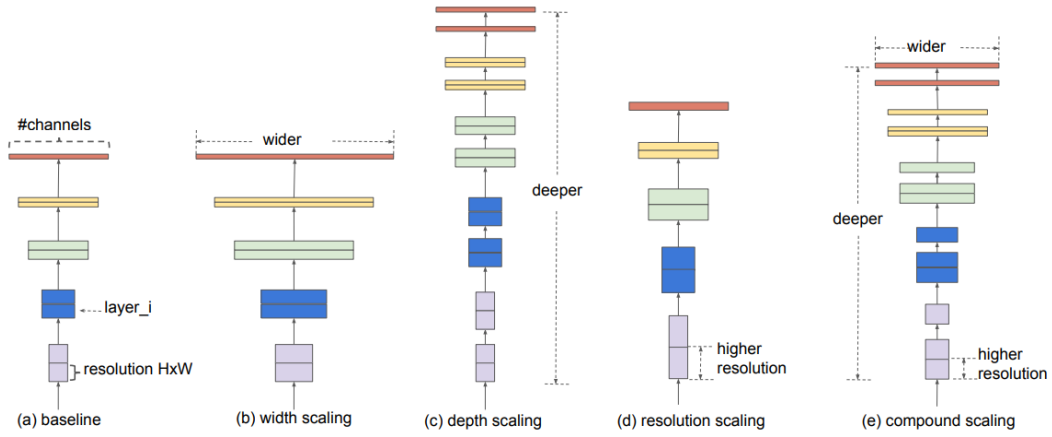


Figure 3.13: Scale each of the parameters separately vs compound scaling method, reproduced from Tan and Le (2019)



Figure 3.14: Efficient Net basic architecture, reproduced from Tan and Le (2019)

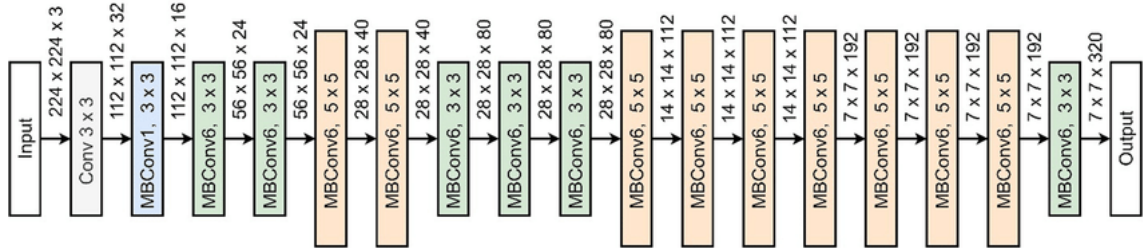


Figure 3.15: EfficientNet-b0 architecture, reproduced from Tan and Le (2019)

3.5.3 DenseNet

As well as ResNets, DenseNets were created to overcome the problem of vanishing gradients. However, instead of shortcuts as in ResNets, here all the layers are connected (Fig. 3.16). In other words, in ResNet we were summing the output of the previous layer with the actual output. In DenseNet all outputs from previous layers are used for the output of the actual layer.

3.5.3.1 DenseNet-121

Comparing with the figure Fig. 3.7, the input layer of DenseNet-121 is a convolution 7×7 , the hidden layers are all layers up to the last pooling layer. Finally, the output layer is a FC layer.

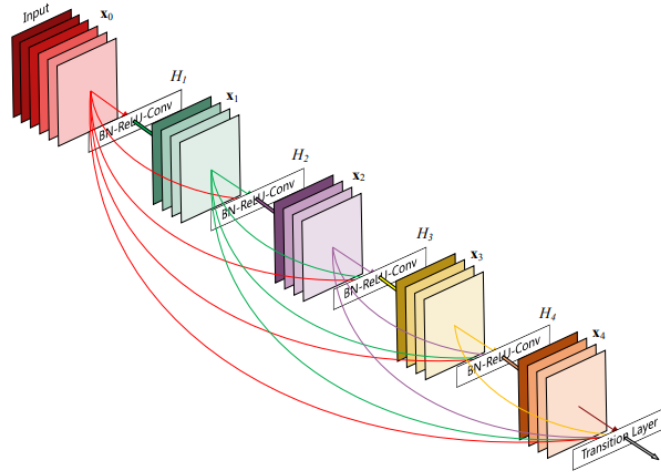


Figure 3.16: DenseNet approach, reproduced from Hassan et al. (2021)

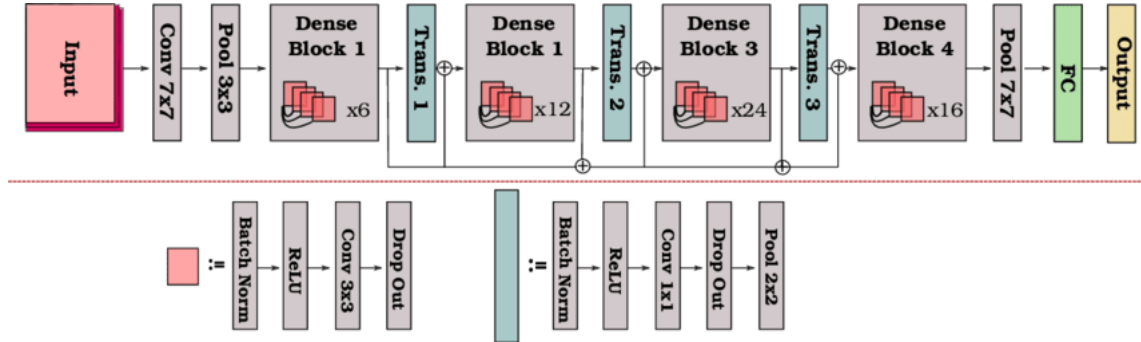


Figure 3.17: Example of a DenseNet, reproduced from Badgujar (2021)

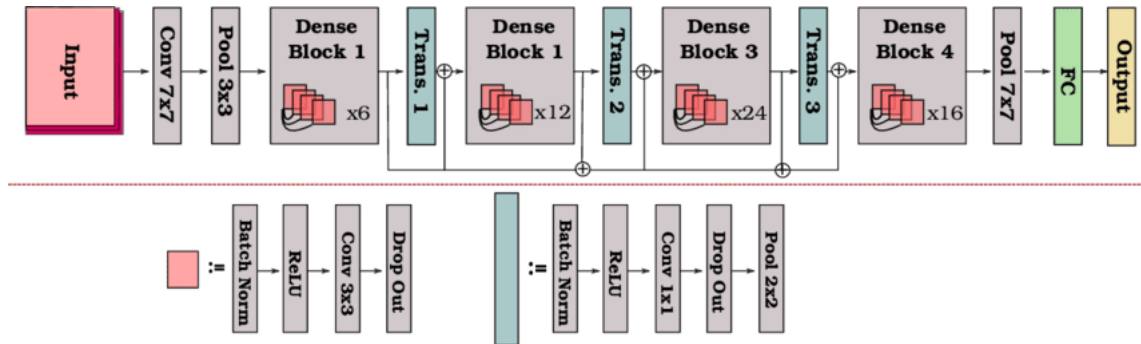


Figure 3.18: DenseNet-121 architecture, reproduced from Radwan (2019)

3.6 Evaluation metrics

With the dataset tested, we need to evaluate the method used. There are different metrics, and whereas some can be interpreted together, there are some that are better used in specific cases.

All the most-known ways to evaluate a dataset are based on True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). TPs and TNs are well-classified data, as

being part of the class, or not, respectively. FPs are the data that was classified by the system as being Positive but actually is not. FNs are the data that were misclassified as being Negative. These numbers can be agglomerated in a matrix called *confusion matrix*, where the principal diagonal of the (squared) matrix corresponds to the well-classified data.

Some of the most well-known metrics are the following:

- Accuracy: computes the number of correct predicted classes in relation to the total number of samples

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: computes the fraction of positive patterns that are correctly predicted by all predicted patterns in a positive class

$$Precision = \frac{TP}{TP + FP}$$

- Recall (also known as Sensitivity): computes the fraction of positive patterns that are correctly classified

$$Recall = \frac{TP}{TP + FN}$$

- F1-score: computes the harmonic mean between Precision and Recall

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

- True Positive Rate (TPR): computes the probability of a false alarm

$$TPR = \frac{TP}{TP + FN}$$

Chapter 4

Methodology

4.1 Dataset

In the first approach, a subset from Drosophoto's website¹ has been used. On one hand, it contains numerous species, however there are very few images in each class (not more than 20 images in a given class). Therefore, as a second approach, images were taken in the i3S lab, and the image resolution was varying, even though it was always relatively similar. The images were taken with three different states of contrast: low, medium, and high. In total, there are 275 images of size 1600×1200 (weight and height, respectively). From these, 89 are classified as *D. melanogaster* – Fig. 4.1a – 50 images with low contrast, 20 with medium contrast, and 20 with high contrast); 94 are of *D. novamexicana* – Fig. 4.1b– 83 with low contrast, and 9 with medium contrast); lastly, there are 91 images of *D. americana* – Fig. 4.1c – 80 images with low contrast, and 10 images with medium contrast.



(a) *D. melanogaster*



(b) *D. novamexicana*



(c) *D. americana*

Figure 4.1: Selected *Drosophila* species.

4.2 Processing pipeline

The created pipeline (Fig. 4.2) consists of three phases, a first one where images are preprocessed, a second one where images are trained and tested through a CNN, and a last one where the desired

¹<https://drosophoto.com/>

images are classified. To the user, it is possible to just preprocess the images; to do preprocessing and train one the possible models (DenseNet-121, EfficientNet-b0, and ResNet-50); to preprocess, train and test a model, and classify other images that are not in the dataset (this means, external images); and just the classification – just note that in this last case, the user has to train and test the model at least once.

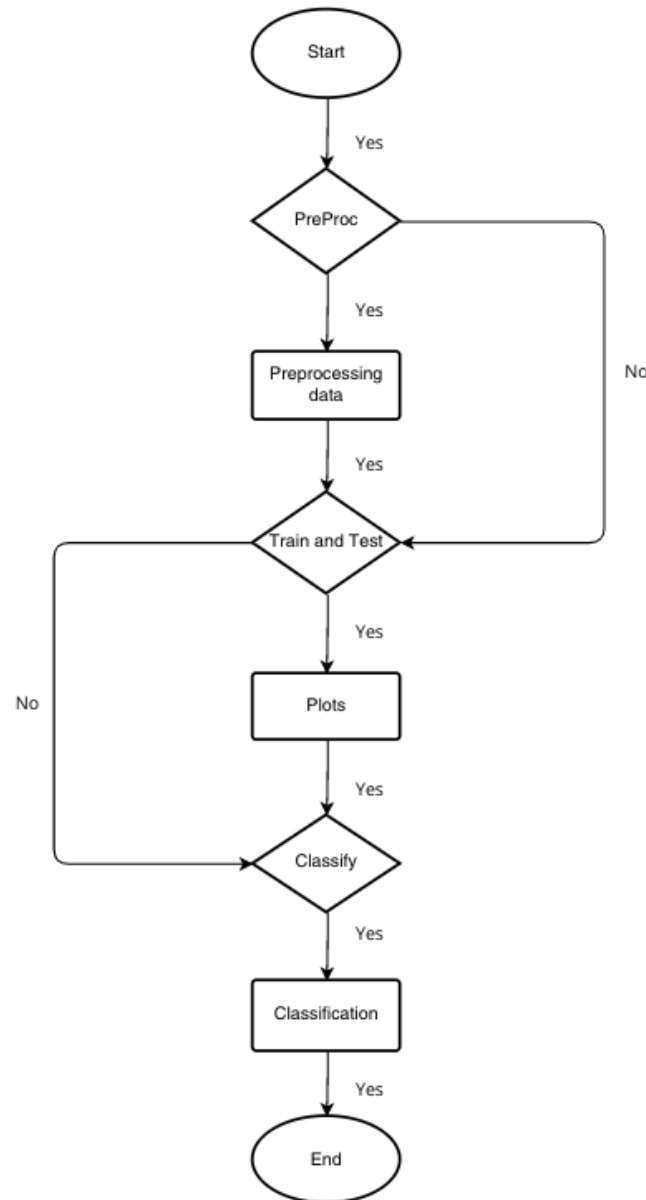


Figure 4.2: Processing pipeline

As the name suggests, the first phase is named *Preprocessing*, and consists in using techniques such as thresholding and smoothing. The second part is *Training and Testing* and is where the model learns its parameters (weights and biases). Training is done using the backpropagation algorithm. Notice that before images are used as input for NNs, data augmentation is done. Data

augmentation is a method to have more images using linear transformation and is used when the dataset is relatively small. In this case, the operations made were a resize of 256×256 , a centre crop of 224×224 , conversion of the image to a tensor, and lastly a normalisation according to ImageNet parameters. In this phase, there is a sub-phase where we plot the variations in accuracy and losses, and is where we plot some metrics, namely the Precision, Recall, and F1-score.

The last phase is the classification per se of the image the user provides. It is worth noticing that the problem at hand is of supervised learning type, as the data is all labelled before the training.

4.3 Background Removal

Thresholding is a segmentation procedure. This means that we are interested in analyzing in our image just an element in particular, which is in the foreground. For that purpose, we used the *rembg* application², which is written in Python. An example of its output can be seen in Fig. 4.3.



(a) Original image



(b) Example of an image with the background removed by the application *rembg*

Figure 4.3: *D. melanogaster* before and after background removal

4.4 Smoothing

The objective of using smoothing filters is to reduce the noise from the image we are analyzing. Depending on the filter used the result differs. The only linear filter used was the Gaussian filter, whereas the remaining ones (median and bilateral) are nonlinear.

To decide the best kernel to use in the Gaussian filter, median filter, and unsharp masking, we computed PSNR, for each filter, for several values (Fig. 4.4). With the kernel size that provided us with the highest PSNR, we compute the respective contrast. For the bilateral filter, the process was the same, although the parameter that was varying was the *sigmaColor* (and consequently the *sigmaSpace*, as it is recommended to use the same value for both, for simplicity³).

²<https://github.com/danielgatis/rembg>

³https://docs.opencv.org/4.7.0/d4/d86/group__imgproc__filter.html#ga9d7064d478c95d60003cf839430737ed

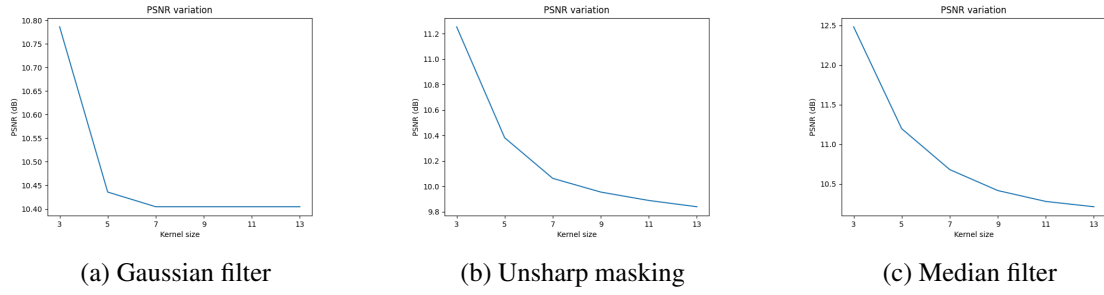


Figure 4.4: PSNR variation according to kernel size for Gaussian filter, median filter, and unsharp masking

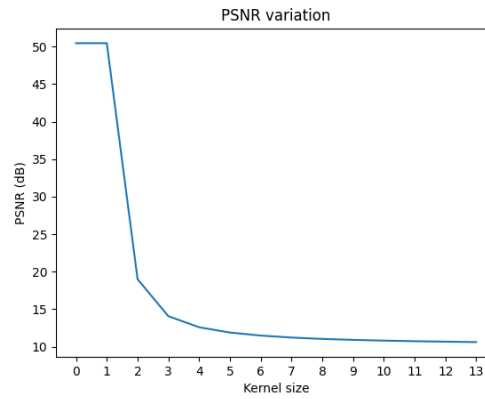


Figure 4.5: PSNR variation, according to sigma, for bilateral filtering

According to the described method to be used, we decided to use a kernel size of 3×3 , and a sigmaX (which is Gaussian kernel standard deviation in X direction) of 1. sigmaY value was the default value, which is the same value used for sigmaX. Lastly, the borderType value used was also the default one, which is 0. The borderType is the pixel extrapolation method. As unsharp masking is the sum between the mask and the original image, and as the filter used was a Gaussian filter, the parameters for the unsharp masking method were the same as the ones of the Gaussian filter.

The kernel size used for the median filter, which is its only parameter, was of size 3×3 .

For the bilateral filter, the diameter of each pixel neighborhood used was 7, the sigmaColor (which is the filter sigma in the color space) was of 0, as well as sigmaSpace (which is the filter sigma in the coordinate space). The borderType used (this is, the border mode used to extrapolate pixels outside of the image) was the default value.

4.5 CNNs

For this thesis, three different CNNs were trained: EfficientNet-b0, ResNet-50, and DenseNet-121, with the first dataset that was presented in section 4.1. Afterwards, we decided to create our

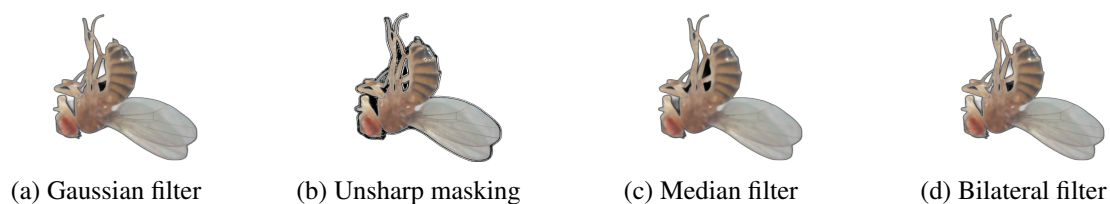


Figure 4.6: Filtered images – with the parameters described in the subsection 4.4

own dataset instead, although the images were trained only in ResNet-50, EfficientNet-b0, and DenseNet-121 as these were the ones that had better results with the first dataset.

To train the model, it was used Adam's optimizer and Cross-Entropy loss function. We used a LR and a weight decay both of 10^{-4} . *Weight decay* is a regularization term that is added to weights. It is useful because it prevents CNN from overfitting. All experiments were made for 40 epochs. Finally, we used a batch size of 16. Then, before splitting the dataset, we applied

The dataset was split into train and test sets, with percentages of 70% and 30%, respectively, meaning that 83 images were used for testing. Then, the results with and without preprocessing were compared. The main advantage of Convolutional Neural Networks is that it automatically detects the features to be learned without human supervision, besides being able to learn extensive amounts of data.

Chapter 5

Results and Discussion

5.1 Experiments with the original images

5.1.1 DenseNet-121

When feeding the raw images as input, DenseNet-121 obtains a train accuracy of 100%, a train loss of 0.012, and a test accuracy of 98.80%. Results with DenseNet-121 are astonishing (Table 5.1). At a first glance, we would say that this is the perfect CNN for this dataset.

Species	Precision	Recall	F1
<i>D. americana</i>	100%	97%	98%
<i>D. melanogaster</i>	100%	100%	100%
<i>D. novamexicana</i>	96%	100%	98%

Table 5.1: DenseNet-121 results

However, these results cannot be taken seriously, as experiments made in the lab prove that the images were being classified based in their background, instead of being classified by the features that are important to really learn to distinguish the different species. Because of this, we decided not doing anything that involved these results, and, as this happened with every species, the results involving the two other CNNs are not presented.

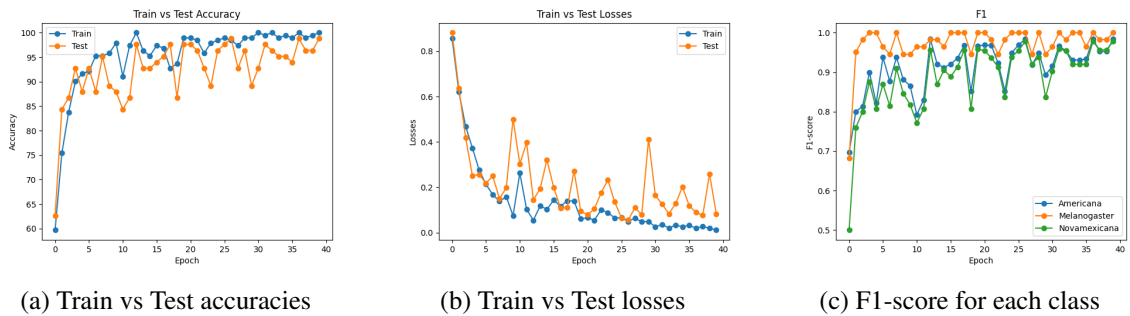


Figure 5.1: DenseNet-121

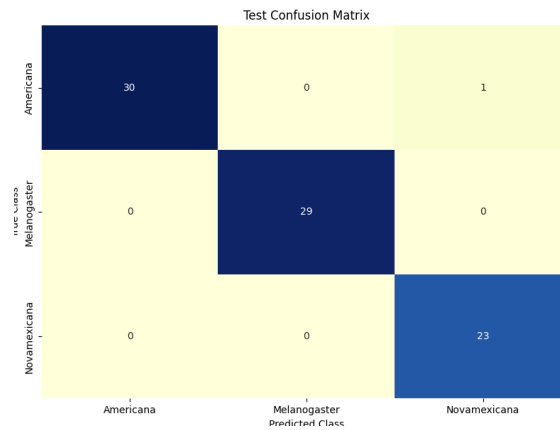


Figure 5.2: Confusion matrix for DenseNet-121

5.2 Experiments with images with no background

5.2.1 With no preprocessing

5.2.1.1 DenseNet-121

In the images with no background, the training accuracy was 99.48%, train loss was 0.03, and the test accuracy was 71.08%. Overall, with the images with no background as input, DenseNet-121 provides good results (Table 5.2). The worst test measures are *D. americana* precision, and *D. novamexicana* recall, as we can see in Table 5.2.

Species	Precision	Recall	F1
<i>D. americana</i>	49%	83%	61%
<i>D. melanogaster</i>	86%	77%	81%
<i>D. novamexicana</i>	100%	55%	71%

Table 5.2: DenseNet-121 results

Also, according to figures Fig. 5.3a and Fig. 5.3b, we can conclude that there is overfitting, as at the beginning of both train and test accuracies and losses are very similar, but around the fifth epoch, the CNN starts having discrepancies between train and test results. This means that either the patterns learned in the training phase were wrong, the patterns learned were not enough, or we would need more data. About the F1-score, Fig. 5.3c, the highest grade is for *D. melanogaster*, and the worst is for *D. americana*. However, it is interesting to notice that this score gets higher, and lower, for all three species approximately at the same time.

We can see by the respective confusion matrix, Fig. 5.4, why *D. americana* is the one with the lowest F1-score. All misclassified images of the other two species were predicted as being *D. americana*. This is actually something that was not expected, as the other two species are much more similar between them compared to *D. melanogaster*. Also, these images had no background, which means that the problem was about some feature that was not the best for this classification.

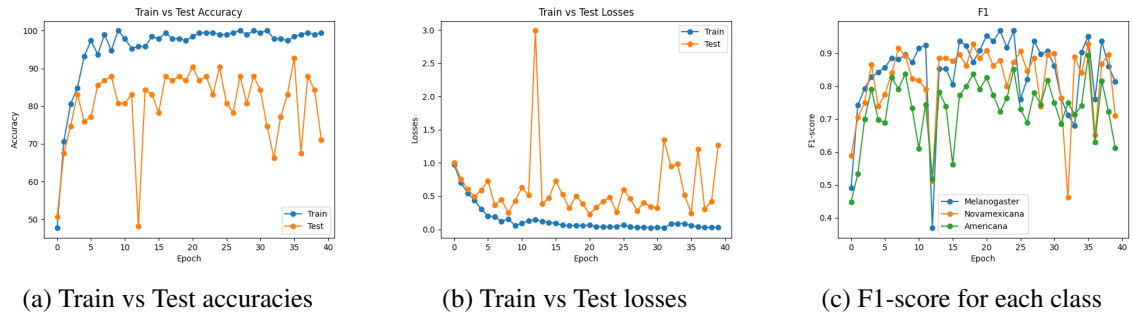


Figure 5.3: DenseNet-121

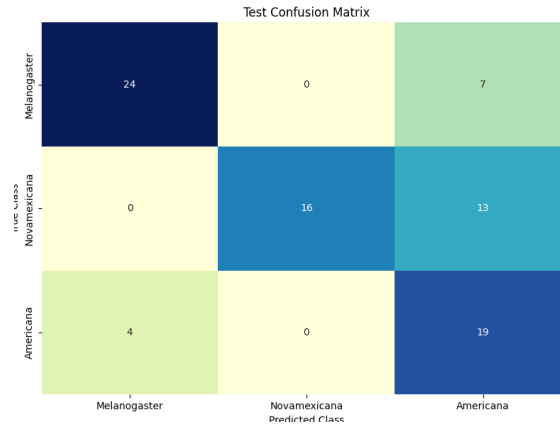


Figure 5.4: Confusion matrix for DenseNet-121

5.2.1.2 ResNet-50

With the images with no background, the training accuracy was 98.95%, train loss was 0.02, and the test accuracy was 72.29%. An interesting fact about this experiment is that all the scores are approximately uniform (Table 5.3). This means that, for this CNN, all the species were relatively similar, so even though it seems that the system suffers from overfitting, it actually may not be the case.

Species	Precision	Recall	F1
<i>D. americana</i>	64%	78%	71%
<i>D. melanogaster</i>	70%	68%	69%
<i>D. novamexicana</i>	84%	72%	78%

Table 5.3: ResNet-50 results

Another interesting point is that, as it happened with the original images, F1-score converges to 100% in all classes, as can be seen in figures Fig. 5.5b and Fig. 5.5c. This means that with the “experience”, the system actually learned to distinguish the three species correctly.

Surprisingly, the CNN seems to have been more overfitting with the images with no background (Fig. 5.5a and Fig. 5.5b). This fact had already occurred with the original image with

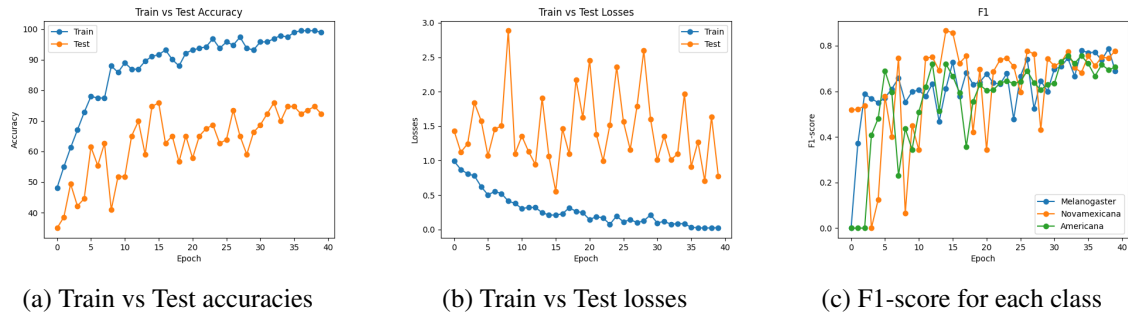


Figure 5.5: ResNet-50

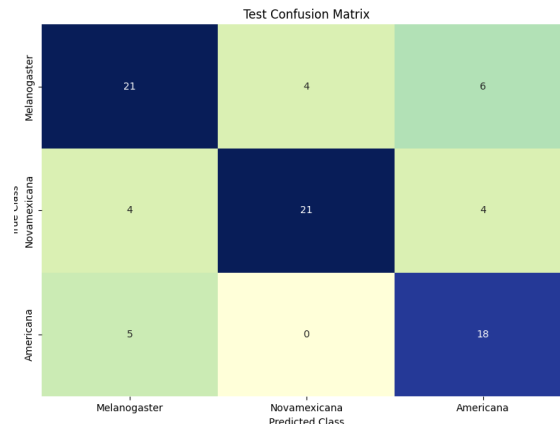


Figure 5.6: Confusion matrix for ResNet-50

ResNet-50. This may be happening either because of the dataset or because of the CNN architecture. We believe this is due to the second option, as this did not happen with DenseNet-121. According to what was said in the previous chapter, ResNet-50 sums the actual output with the output of some layers before. This means that if some of these outputs are not as good, the actual output gets affected. To change this issue, maybe trying CNNs that do not have these skip connections provides better results. Another option would be to change the locations of these skip connections so that the output of the sum has “more actual” features.

With the images with no background, the training accuracy was 93.72%, train loss was 0.20, and the test accuracy was 57.83%.

Species	Precision	Recall	F1
<i>D. americana</i>	46%	70%	55%
<i>D. melanogaster</i>	66%	61%	63%
<i>D. novamexicana</i>	68%	45%	54%

Table 5.4: EfficientNet-b0 results

5.2.1.3 EfficientNet-b0

Clearly, in this experiment, EfficientNet-b0 suffers from overfitting. It can be seen in several ways, such as:

- the astonishing difference between train and test accuracy, seen in Fig. 5.7a
- the test loss gets worse along the epochs, whereas the opposite happens for train loss, see Fig. 5.7b
- F1-score is never higher than 70% for any class, see Fig. 5.7c
- the confusion matrix, see Fig. 5.8, has FPs in every class

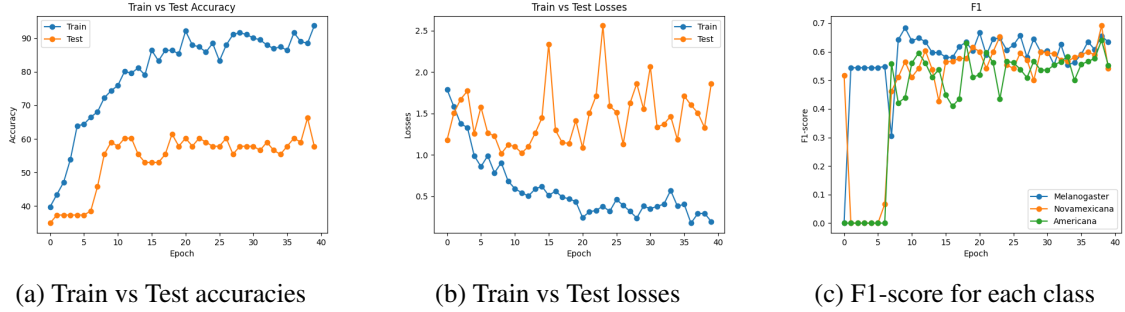


Figure 5.7: EfficientNet-b0

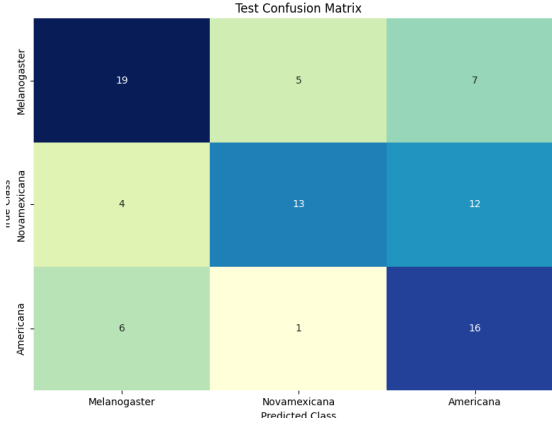


Figure 5.8: Confusion matrix for EfficientNet-b0

Model	Accuracy	Precision	Recall	F1
DenseNet-121	78%	71%	72%	71%
ResNet-50	72%	73%	73%	72%
EfficientNet-b0	58%	60%	59%	58%

Table 5.5: Model's comparison for images with no background

The results from DenseNet-121 and ResNet-50 are very similar, with the exception of accuracy, which is higher in DenseNet-121. EfficientNet-b0 provides very poor results in all parameters. These results are in line with the previous results, as DenseNet-121 had always the best results and EfficientNet-b0 was the worst. From now on, we will just apply the filters to DenseNet-121, as it is the NN that provided the best global scores, to compare which filter(s) provide the best results.

5.2.2 With preprocessing

These experiments were conducted just with DenseNet-121, as it was with this CNN that the best results were obtained.

5.2.2.1 Gaussian filtering

In this experiment, train accuracy was of 100%, the train loss was 0.02, and the test accuracy was of 87.95%. *D. melanogaster* was the species with the best results, as it can be seen in Table 5.6, even though the results for the three species are good.

Species	Precision	Recall	F1
<i>D. americana</i>	86%	78%	82%
<i>D. melanogaster</i>	86%	100%	93%
<i>D. novamexicana</i>	92%	83%	87%

Table 5.6: DenseNet-121 results for Gaussian filtered images

According to Fig. 5.9a and Fig. 5.9b, the model seems to suffer a little overfitting in the first epochs, but it is overcome. According to Table 5.6 and Fig 5.9c F1-scores are all high, with the best being *D. melanogaster* and the worst being *D. americana*. This is saying that *D. melanogaster* is the class with the least FPs, which can be seen in Fig. 5.10.

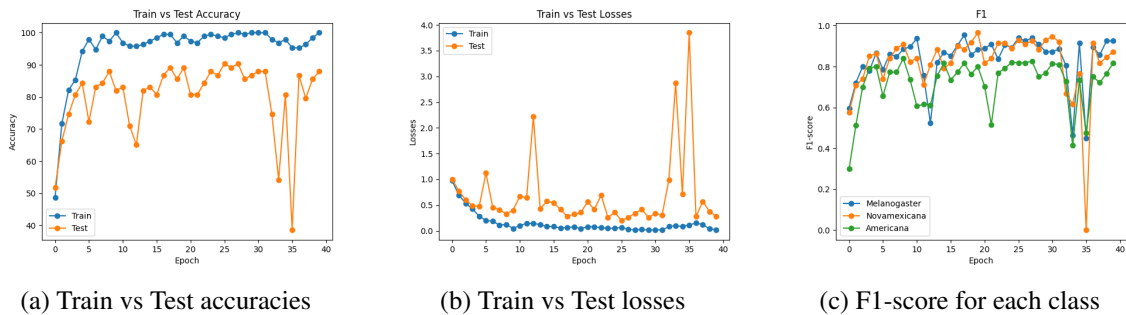


Figure 5.9: Gaussian filter

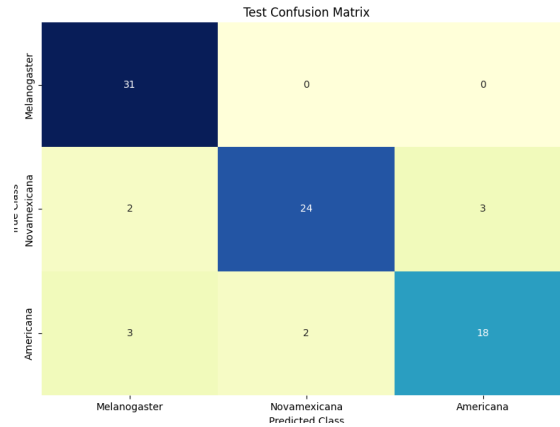


Figure 5.10: Confusion matrix for Gaussian filter as preprocessing

5.2.2.2 Median filtering

In this experiment, train accuracy was 98.95%, train loss was 0.03, and the test accuracy was 60.24%. Once again, *D. melanogaster* is the one with the overall best score, but now by a large margin (Table 5.7).

Species	Precision	Recall	F1
<i>D. americana</i>	41%	83%	55%
<i>D. melanogaster</i>	81%	81%	81%
<i>D. novamexicana</i>	100%	21%	34%

Table 5.7: DenseNet-121 results for median filtered images

With this mean filter, Fig. 5.11a is similar to Fig. 5.9a, however overfitting is never overcome. From Fig. 5.11b we can conclude that either the system does not learn the right features for species identification, or the data is not enough. Fig. 5.12 and Fig. 5.11c tell us that there is some feature that is not being learned, that is what should help to distinguish between *D. americana* and *D. americana*, although this is counterintuitive according to what was said previously.

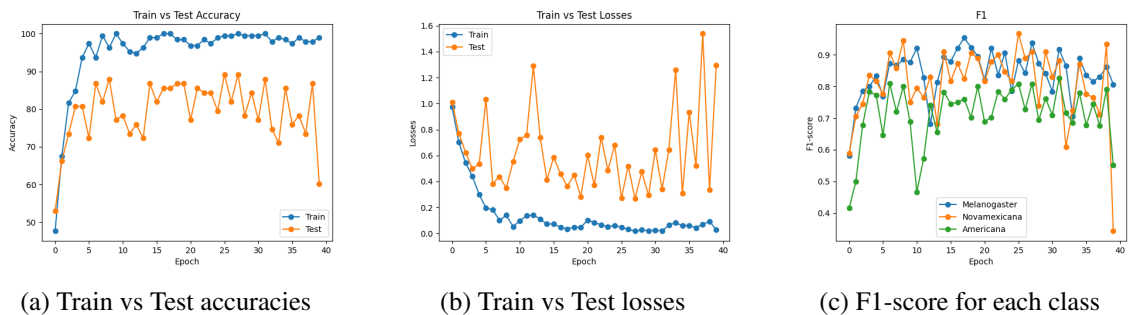


Figure 5.11: Median filter

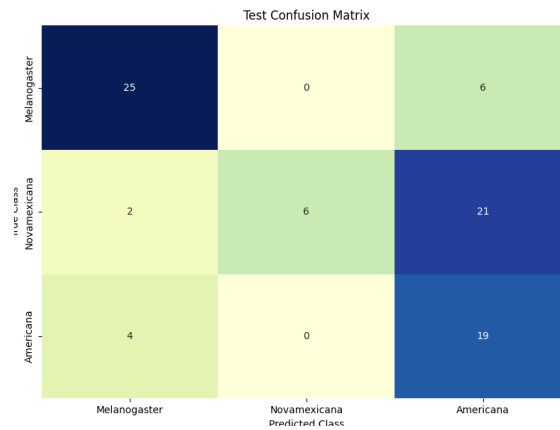


Figure 5.12: Confusion matrix for DenseNet-121 with median filtered images

5.2.2.3 Unsharp masking

In this experiment, train accuracy was 98.95%, train loss was 0.03, and the test accuracy was 85.54%. With unsharp masking, *D. novamexicana* was the species with the best results overall, even though *D. melanogaster*'s recall is higher, according to Table 5.8.

Species	Precision	Recall	F1
<i>D. americana</i>	88%	65%	75%
<i>D. melanogaster</i>	77%	97%	86%
<i>D. novamexicana</i>	96%	90%	93%

Table 5.8: DenseNet-121 results for unsharped images

The plot of 5.13a is what is expected of a well-trained model. The worst about this method is the number of *D. americana* that were classified as being *D. melanogaster*, as it is shown in Fig. 5.13c and Fig. 5.14. Biologically this does not make much sense, as they are not from the same group. However, the only problem can actually be the dataset not being big enough, as it is the problem that is shown in Fig. 5.13b.



Figure 5.13: Unsharp masking

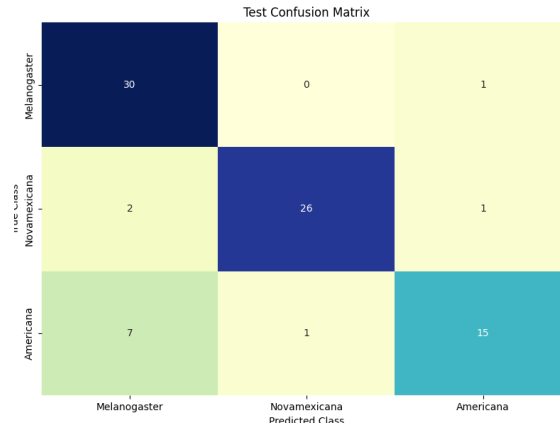


Figure 5.14: Confusion matrix for DenseNet-121 with unsharp masked images

5.2.2.4 Bilateral filtering

In this experiment, train accuracy was 100%, train loss was 0.015, and the test accuracy was 85.54%. The species with the highest overall score is *D.melanogaster*, although none of the classes had a score below 80%, which is very good (Table 5.9).

Species	Precision	Recall	F1
<i>D. americana</i>	78%	78%	78%
<i>D. melanogaster</i>	86%	97%	91%
<i>D. novamexicana</i>	92%	79%	85%

Table 5.9: DenseNet-121 results for bilateral filtered images

Fig. 5.15a shows a bit of overfitting, although the losses in Fig. 5.15b are as expected. This means that there are no big information losses about features. In this way, Fig. 5.15a may be explained by the NN not having learned the right features, or not having the right amount of features. The results in Fig. 5.16 (and in Fig. 5.15c) show that there is not a high amount of misclassified images of one of the species, even though the FPs are distributed along all of the three species, which may mean that the NN would need more training epochs, or more data.

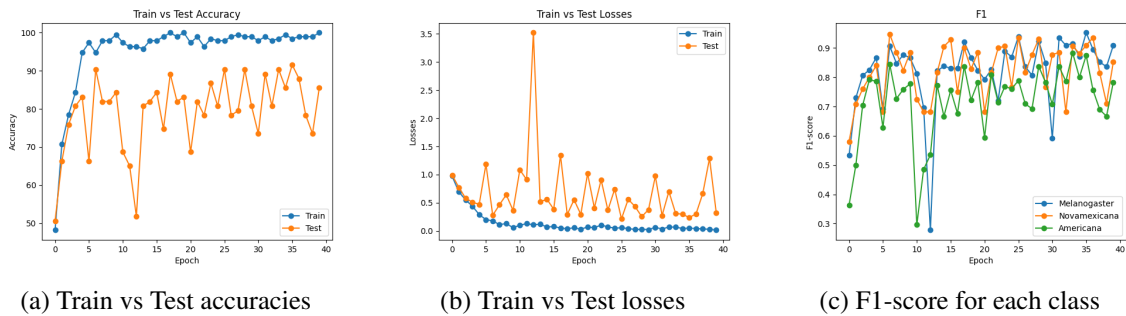


Figure 5.15: Bilateral filter

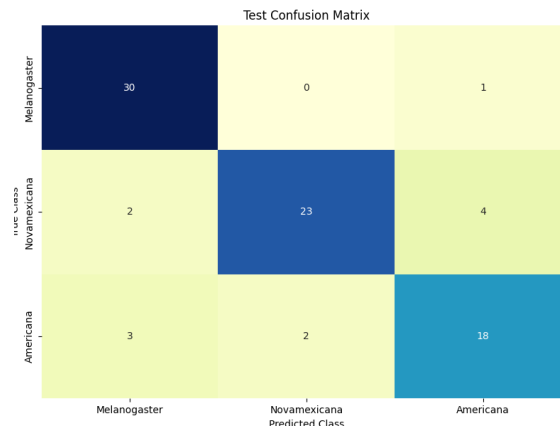


Figure 5.16: Confusion matrix for DenseNet-121 with bilateral filter as preprocessing

5.2.3 Global results

Technique	Accuracy	Precision	Recall	F1
Gaussian filter	88%	88%	87%	87%
Median filter	74%	61%	57%	60%
Usharp masking	86%	87%	84%	85%
Bilateral filter	86%	85%	85%	85%

Table 5.10: Comparison of the preprocessing techniques

The best preprocessing technique was the Gaussian filter, followed by both unsharp masking and bilateral filter, and the median filter lastly. It is worth noting that although the images had noise, the background was homogeneous. This is saying that the noise was not of the salt-and-pepper type, which can explain why the median filter got the worst results.

Gaussian noise is a lowpass filter, and unsharp masking can be considered a highpass filter. As both provide good results, it may mean that the ideal filter would be a bandpass, which is an intermediate type of filter between these two. The bilateral filter enhances the edges (high frequencies), so it may be a plus for learning some features through the contours, such as the size, the perimeter, the area, and the number of details in the wings. The bilateral filter can be thought of as a band-reject filter.

In conclusion, the only thing we can say is that the median is the worst of the preprocessing made, and all of the other preprocessing methods provided relatively similar values. From here we can conclude that the “problem” is actually in the images, because, as in images in Chapter 3 show, they are all relatively similar and mostly black. Besides that, the kernels used were small (3×3), except for the bilateral filter. So, it was expected that at least the Gaussian filter and unsharp masking had similar results.

5.3 Overall Discussion

Our results are in agreement with the results obtained by Şaban Öztürk and Akdemir (2018), that it is better to use a normal processing – removing the background and apply a filter to enhance part of the image – than not using any preprocessing. This is somewhat expectable, as real-world images contain noise, in different quantities, that affect the training, and this influences the image classification task.

Avşar (2021) results are also in agreement with ours when saying that more than 30 epochs do not make much difference. In fact, in the plots of the Results section, the values were mainly stable when achieving 30 epochs. In some cases, the accuracy decreased abruptly, but it could have also happened with more epochs, as that has to do with the features learned, and not with the number of epochs per se.

Shen et al. (2021) achieved an accuracy of 95.44% with EfficientNet-b0. In our case, just with the background removed, we obtained an accuracy of 57.83%. We do not have results with EfficientNet-b0 with preprocessing, due to our methodology, where filters were applied to the CNN that had the best results with no background. However, their dataset contains 674 images of *Anastrepha*, 1463 of *Ceratitis*, 1030 of *Rhagoletis*, and 1061 of *Bactrocera*. We have less than a quarter of the images in this study. We do believe that if we had more images, our results would be better. However, when we removed the background, EfficientNet-b0 was the CNN with the worst result. It may be because it is the only one we tried that changes all the parameters at the same time. Probably, when changing one parameter each time we have more control over it and can change the parameters accordingly.

Luo et al. (2022) obtained accuracies greater than 98% with all CNNs, in particular ResNet-50 and DenseNet-121. With just the background removed, we obtained an accuracy of 72.29% for ResNet-50 and an accuracy of 71.08% for DenseNet-121. Applying filtering operations after the background removal, we obtained accuracies of 87.95% for the Gaussian filter, 60.24% for the median filter, and 85.54% for both unsharp masking and bilateral filter. The main difference in the results of Luo et al. (2022) is that their images are high resolution. In the dataset used for this thesis, we tried to have images of low, medium, and high resolution. That may degrade the results, so we can say that when creating a dataset for insect classification, it is better to have all images with the same level of resolution.

Giełczyk et al. (2022) concluded that adaptive masking followed by histogram equalization and Gaussian blur was the best preprocessing method. Our results partially agree with this, as the Gaussian blur was the one with the best results, having achieved an accuracy of 87.95% for the images with no preprocessing. Their results were better, although the difference is not that much. It can be explained by the fact that they used a 5×5 kernel, while we used a 3×3 kernel, which has less effect. We used a 3×3 kernel as this was the kernel size that provided the highest PSNR. However, as the PSNR decreased smoothly, we could have used actually a 5×5 kernel.

Our results are in agreement with the ones obtained by Kusrini et al. (2022) in the part that the Gaussian filter after applying a threshold is the best technique (our background removal is,

actually, thresholding). It is expectable, as Gaussian noise is an very common type of noise. Also, when doing a threshold, Machine Learning knows where to find the features, and with that, the probability of learning the correct features increases. However, [Kusrini et al. \(2022\)](#) concluded that the median filter is also one of the best, which is opposite to our conclusion. It can be, as already said, that the noise of our images really is not of the salt-and-pepper type.

Chapter 6

Conclusions

The main objectives of this thesis were achieved, to develop a program that distinguishes successfully *Drosophila* species, and compare the results with and without preprocessing the images.

The first conclusion to be taken is that we should always remove the background before the images are used as input for the CNNs, as it may happen that the CNN learns the background features instead of the *Drosophila* features to be classified. With the background removed from the input data, with and without image preprocessing, DenseNet-121 is always the best-performing CNN. This may be related to its architecture, as DenseNets use the features from all layers up to the actual layer, whereas neither EfficientNet or ResNet have that mechanism. Actually, the results of ResNet-50 can be explained by its architecture, because the actual output can have not-so-correct features that are used for the sum in the skip connections, and that can poorly affect the results. The second conclusion is that there is not much difference in using Gaussian filtering, bilateral filtering, or unsharp masking. This can be explained by the arguments given in the previous chapter, that the image is relatively homogeneous, that the kernel sizes are small, and that the frequency range is very limited.

As future work, it is intended to improve the program, so it can classify more *Drosophila* species. It is also intended to diminish the overfitting of some of the methods used.

Appendix A

Novikoff's Theorem

This theorem proves that if we have a data set that is linearly separable, and non-trivial, it converges after a finite number of steps (in this case to converge means that all the points are well-classified). Besides that, it provides us with an upper bound on the time complexity. But before we get into the theorem, some definitions are needed (this appendix was adapted from [University \(2023a\)](#), [Collins \(2023\)](#), and [University \(2023b\)](#)).

Definition 1. A dataset is said *linearly separable* if

$$\forall (\vec{x}, y) \in \mathbb{X} \exists \vec{w}_* \exists \gamma > 0 : \vec{w}_* \cdot \vec{x} > y \geq \gamma$$

Definition 2. A set is said to be *non-trivial* if not all x_i have the same label.

Definition 3. The distance between the hyperplane (w, b) and the nearest point in \mathbb{D} is given by

$$\text{margin}(\mathbb{D}, w, b) = \begin{cases} \min_{(x, y) \in \mathbb{D}} y(w \cdot x + b) & \text{if } w \text{ separates } \mathbb{D} \\ -\infty & \text{otherwise} \end{cases}$$

Definition 4. The *margin of a dataset* is given by

$$\text{margin}(\mathbb{D}) = \sup_{w, b} \text{margin}(\mathbb{D}, w, b)$$

Theorem 1 (Novikoff's Theorem). *Let \mathbb{S} be a set of tuples $\{(x_1, y_1), \dots, (x_n, y_n)\}$ where x_i are vectors in \mathbb{R}^n and y_i is a label, either -1 or 1. If \mathbb{S} is linearly separable and non-trivial, then there exists a vector w_{opt} in \mathbb{R}^n , a scalar b_{opt} , and a positive scalar γ , so that $\|w_{opt}\| = 1$ and*

$$y_i(w_{opt} \cdot x_i + b_{opt}) \geq \gamma$$

If this holds and we use with Perceptron Algorithm to find values of w and b that separates the points, then the number of mistakes made by the algorithm is bounded by R^2/γ^2 .

Proof. Let \hat{w} be the vector in \mathbb{R}^{n+1} which is the vector w augmented by a magnitude of b/R , and \hat{x}_i the vector in \mathbb{R}^{n+1} which is the vector x_i augmented by the magnitude of R . This makes it so that

$w \cdot x + b = \widehat{w}_i - \widehat{x}_i$. One can easily check that the update rule $\widehat{w} = \widehat{w} + \eta y_i x_i$ preserves the updates to w and b in the algorithm.

It will be proved by induction that $k \leq (R/\gamma)^2 \forall k$. However, firstly there will be proved a couple of Lemmas that are going to be useful for the proof.

Lemma 2. $\widehat{w}_k \cdot \widehat{w}_{opt} \geq k\eta\gamma$

Proof. By induction,

$$\begin{aligned} \widehat{w}_k \cdot \widehat{w}_{opt} &= (\widehat{w}_{k-1} + \eta y_i \widehat{x}_i) \cdot \widehat{w}_{opt} \\ &= \widehat{w}_{k-1} \cdot \widehat{w}_{opt} + \eta y_i \widehat{x}_i \cdot \widehat{w}_{opt} \\ &\geq (k-1)\eta\gamma + \eta y_i \widehat{x}_i \cdot \widehat{w}_{opt} \\ &\geq (k-1)\eta\gamma + \eta\gamma \\ &= k\eta\gamma \end{aligned}$$

□

Lemma 3. $\|\widehat{w}_k\|^2 \leq k\eta^2 R^2$

Proof. By induction,

$$\begin{aligned} \|\widehat{w}_k\|^2 &= \|\widehat{w}_{k-1} + \eta y_i x_i\|^2 \\ &= \|\widehat{w}_{k-1}\|^2 + 2\eta y_i \widehat{w}_{k-1} \cdot x_i + \eta^2 \|\widehat{x}_i\|^2 \\ &\leq \|\widehat{w}_{k-1}\|^2 + \eta^2 (\|x_i\|^2 + R^2) \\ &\leq (k-1)\eta^2 R^2 + \eta^2 R^2 \\ &= k\eta^2 R^2 \end{aligned}$$

Do just notice the fact $y_i \widehat{w}_{k-1} \cdot x_i < 0$ is used from step 2 to step 3.

□

In conclusion, to prove the main theorem, we assemble these lemmas:

$$\begin{aligned} k &\leq \frac{\widehat{w}_k \cdot \widehat{w}_{opt}}{\eta\gamma} \\ &\leq \frac{\|\widehat{w}_k\| \|\widehat{w}_{opt}\|}{\eta\gamma} \\ &\leq \frac{\|\widehat{w}_k\|}{\eta\gamma} \\ &\leq \frac{\sqrt{k}\eta R}{\eta\gamma} \\ &= \frac{\sqrt{k}R}{\gamma} \end{aligned}$$

$$\therefore \frac{k}{\sqrt{k}} = \sqrt{k} \leq \frac{R}{\gamma} \implies k \leq \left(\frac{R}{\gamma}\right)^2$$

□

Appendix B

Tutorial to use the program

The code, which is covered under an MIT License, can be found at <https://github.com/DanielaFCUP/DrosophilaV2>.

1. Create a conda environment

- (a) `conda create --name myenv python=3.10`

- (b) `conda activate myenv`

where myenv is the name you want for your environment

2. Install the requirements: `pip install -r /path/to/requirements.txt`

where /path/to/requirements.txt is the path to where the requirements file is located.

3. Change the parameters in the file `conf.yaml`, in case you want, except the outputs parameter

- `optim`: Adam
- `model`: densenet, resnet, or efficientnet
- `epochs`: Any integer greater than 0, even though it is better to be > 30
- `batch`: 16 (it is convenient to be a power of two)
- `lr`: `!!float 5e-4` (it can be any number between 0 and 1)
- `raw`: `'in/'`

4. If you just wish to preprocess your images: `python main.py -c conf/conf.yaml -r preproc -i [image] --preproc [preprocessing method]`

where:

- `image` is the link to the image you want to classify
- the preprocessing methods available are: `skip`, `remove_background`, `bilateral`, `gaussian`, `median`, `unsharp`; `skip` is an option if you do not want to preprocess your images
- `preproc`: it only does image preprocessing

- prepare: it does preprocessing, training and testing, and plots the performance plots and metrics
- 'classify': it classifies the respective image. You must have a trained model first and point to it with the -m flag
- 'full': it does everything from preprocessing to classification

Obs1: The results from training and testing are saved in /out/outputs.txt

Obs2: If you wish to do two followed preprocessing methods, in your second preprocessing you should write the path to the outputs folder in [image]

References

- Brief about ai/ml/dl, 2018. URL <https://harikrishnabhyravi.blogspot.com/2018/10/aimldl.html>. Accessed June 2, 2023.
- Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-dujaili, Ye Duan, Omran Al-Shamma, Jesus Santamaría, Mohammed Abdulraheem Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8, 2021.
- Ercan Avşar. Effects of image preprocessing on the performance of convolutional neural networks for pneumonia detection. In *2021 International Conference on INnovations in Intelligent Systems and Applications (INISTA)*, pages 1–5, 2021. doi: 10.1109/INISTA52262.2021.9548351.
- Sumeet Badgajar. Unpacking densenet to understand and then creating using tensorflow, 2021. URL <https://medium.com/analytics-vidhya/unpacking-densenet-to-understand-and-then-creating-using-tensorflow-670d88aace5c>. Accessed May 30, 2023.
- Sarkhan Badirli, Christine J. Picard, George Mohler, Zeynep Akata, and Murat Dundar. Classifying the unknown: Identification of insects by deep open-set bayesian learning. *bioRxiv*, 2021. doi: 10.1101/2021.09.15.460492. URL <https://www.biorxiv.org/content/early/2021/09/17/2021.09.15.460492>.
- Hugo Bellen, Chao Tong, and Hiroshi Tsuda. 100 years of drosophila research and its impact on vertebrate neuroscience: A history lesson for the future. *Nature reviews. Neuroscience*, 11: 514–22, 04 2010. doi: 10.1038/nrn2839.
- G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- Rémi Cadène, Nicolas Thome, and Matthieu Cord. Master’s thesis: Deep learning for visual recognition, 2016. URL <https://arxiv.org/abs/1610.05567>.
- Juan Caicedo, Sam Cooper, Florian Heigwer, Scott Warchal, Peng Qiu, Csaba Molnar, Aliaksei Vasilevich, Joseph Barry, Harmanjit Bansal, Oren Kraus, Mathias Wawer, Lassi Paavolainen, Markus Herrmann, Mohammad Hossein Rohban, Jane Hung, Holger Hennig, John Concannon, Ian Smith, Paul Clemons, and Anne Carpenter. Data-analysis strategies for image-based cell profiling. *Nature Methods*, 14:849–863, 09 2017. doi: 10.1038/nmeth.4397.
- Xu Cao, Ziyi Wei, Yinjie Gao, and Yingqiu Huo. Recognition of common insect in field based on deep learning. *Journal of Physics: Conference Series*, 1634(1):012034, 09 2020. doi: 10.1088/1742-6596/1634/1/012034. URL <https://doi.org/10.1088/1742-6596/1634/1/012034>.

- Pierre Capy and Patricia Gibert. *Drosophila melanogaster, drosophila simulans: so similar yet so different*. *Genetica*, 120:5–16, 04 2004. doi: 10.1007/978-94-007-0965-2_1.
- Sri Venkata Divya Madhuri Challa and Hemendra Vaishnav. *Weather Categorization Using Fore-ground Subtraction and Deep Transfer Learning*, pages 595–601. 03 2020. ISBN 978-981-15-2042-6. doi: 10.1007/978-981-15-2043-3_64.
- Channabasava Chola, Bibal Benifa Jv, Devanur Guru, Abdullah Y Muaad, Hanumanthappa Davanagere, Mugahed A. Al-antari, and Abdu Gumaiei. Gender identification and classification of drosophila melanogaster flies using machine learning techniques. *Computational and Mathematical Methods in Medicine*, 2022:1–9, 01 2022. doi: 10.1155/2022/4593330.
- François Chollet. *Deep Learning with Python*. Manning, November 2017. ISBN 9781617294433.
- Michael Collins. Convergence proof for the perceptron algorithm, 2023. URL https://www.cise.ufl.edu/~arunava/Teaching/Lectures-CN/perceptron_convergence.pdf. Accessed January 5, 2023.
- Yarin Gal. Uncertainty in deep learning. 2016.
- Agata Gielczyk, Anna Marciniak, Martyna Tarczewska, and Zbigniew Lutowski. Pre-processing methods in chest x-ray image classification. *PloS one*, 17(4):e0265949, 2022. ISSN 1932-6203. doi: 10.1371/journal.pone.0265949. URL <https://europepmc.org/articles/PMC8982897>.
- Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 2008. ISBN 9780131687288 013168728X 9780135052679 013505267X. URL <http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X>.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- Sk Hassan, Arnab Maji, Michał Jasiński, Zbigniew Leonowicz, and Elżbieta Jasińska. Identification of plant-leaf diseases using cnn and transfer-learning approach. *Electronics*, 10:1388, 06 2021. doi: 10.3390/electronics10121388.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL <https://arxiv.org/abs/1512.03385>.
- Hector Martinez. What is deep learning?, 2023. Accessed May 30, 2023. <https://pyimagesearch.com/2021/04/17/what-is-deep-learning/>.
- Nathalie Jeans. How i classified images with recurrent neural networks, 2019. URL <https://medium.com/@nathaliejeans/how-i-classified-images-with-recurrent-neural-networks-28eb4b57fc79>. Accessed June 9, 2023.
- Barbara H. Jennings. *Drosophila – a versatile model in biology & medicine*. *Materials Today*, 14(5):190–195, 2011. ISSN 1369-7021. doi: [https://doi.org/10.1016/S1369-7021\(11\)70113-4](https://doi.org/10.1016/S1369-7021(11)70113-4). URL <https://www.sciencedirect.com/science/article/pii/S1369702111701134>.
- Chris P. Jobling. Eg-247 signals and systems, 2018. URL <https://cpjobling.github.io/eg-247-textbook/introduction/index.html>. Accessed June 8, 2023.

- Florian Jug, Tobias Pietzsch, Stephan Preibisch, and Pavel Tomancak. Bioimage informatics in the context of drosophila research. *Methods*, 68(1):60–73, 2014. ISSN 1046-2023. doi: <https://doi.org/10.1016/j.ymeth.2014.04.004>. URL <https://www.sciencedirect.com/science/article/pii/S1046202314001480>. Drosophila developmental biology methods.
- Sebastian Bas Kanã. Automatic landmark identification in digital images of drosophila wings for improved morphometric analysis. 2019.
- Martin Kapun, Joaquin C B Nunez, María Bogaerts-Márquez, Jesús Murga-Moreno, Margot Paris, Joseph Outten, Marta Coronado-Zamora, Courtney Tern, Omar Rota-Stabelli, and et al. Guerreiro, Maria P García. Drosophila Evolution over Space and Time (DEST): A New Population Genomics Resource. *Molecular Biology and Evolution*, 38(12):5782–5805, 09 2021. ISSN 1537-1719. doi: 10.1093/molbev/msab259. URL <https://doi.org/10.1093/molbev/msab259>.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.
- Nilay Kumar, Francisco Huizar, Keity Farfán-Pira, Pavel Brodskiy, Dharsan Soundarrajan, Marcos Nahmad, and Jeremiah Zartman. Mapper: An open-source, high-dimensional image analysis pipeline unmask differential regulation of drosophila wing features. *Frontiers in Genetics*, 13, 04 2022. doi: 10.3389/fgene.2022.869719.
- Kusrini Kusrini, Muhammad Arif, Arif Yudianto, and Hanif al Fatta. The effect of gaussian filter and data preprocessing on the classification of punakawan puppet images with the convolutional neural network algorithm. *International Journal of Electrical and Computer Engineering*, 12: 3752–3761, 08 2022. doi: 10.11591/ijece.v12i4.pp3752-3761.
- Van-Linh Le, Marie Beurton-Aimar, Akka Zemhari, Alexia Marie, and Nicolas Parisey. Automated landmarking for insects morphometric analysis using deep neural networks. *Ecological Informatics*, 60:101175, 2020. ISSN 1574-9541. doi: <https://doi.org/10.1016/j.ecoinf.2020.101175>. URL <https://www.sciencedirect.com/science/article/pii/S1574954120301254>.
- Sheng Loh, Yoshitaka Ogawa, Sara Kawana, Koichiro Tamura, and Lee Hwee-Kuan. Semi-automated quantitative drosophila wings measurements. *BMC Bioinformatics*, 18, 06 2017. doi: 10.1186/s12859-017-1720-y.
- Chu-Yuan Luo, Patrick Pearson, Guang Xu, and Stephen M. Rich. A computer vision-based approach for tick identification using deep learning models. *Insects*, 13(2), 2022. ISSN 2075-4450. doi: 10.3390/insects13020116. URL <https://www.mdpi.com/2075-4450/13/2/116>.
- Moritz Lürig, Seth Donoughe, Erik Svensson, Arthur Porto, and Masahito Tsuboi. Computer vision, machine learning, and the promise of phenomics in ecology and evolutionary biology. *Frontiers in Ecology and Evolution*, 09:642774, 04 2021. doi: 10.3389/fevo.2021.642774.
- Kapun M, Barrón MG, Staubach F, Obbard DJ, Wiberg RAW, Vieira J, Goubert C, Rota-Stabelli O, Kankare M, Bogaerts-Márquez M, Haudry A, Waidele L, Kozeretska I, Pasyukova EG,

- Loescheke V, Pascual M, Vieira CP, Serga S, Montchamp-Moreau C, Abbott J, Gibert P, Porcelli D, Posnien N, Sánchez-Gracia A, Grath S, Sucena É, Bergland AO, Guerreiro MPG, Onder BS, Argyridou E, Guio L, Schou MF, Deplancke B, Vieira C, Ritchie MG, Zwaan BJ, Tauber E, Orengo DJ, Puerma E, Aguadé M, Schmidt P, Parsch J, Betancourt AJ, Flatt T, and González J. Genomic Analysis of European *Drosophila melanogaster* Populations Reveals Longitudinal Structure, Continent-Wide Selection, and Previously Unknown DNA Viruses. *Molecular Biology and Evolution*, 37(9):2661–2678, 05 2020. ISSN 0737-4038. doi: 10.1093/molbev/msaa120. URL <https://doi.org/10.1093/molbev/msaa120>.
- Therese A. Markow and Patrick M. O’Grady. Chapter 6 - handling wild-caught specimens. In Therese A. Markow and Patrick M. O’Grady, editors, *Drosophila*, pages 182–185. Academic Press, San Diego, 2006. ISBN 978-0-12-473052-6. doi: <https://doi.org/10.1016/B978-012473052-6/50006-8>. URL <https://www.sciencedirect.com/science/article/pii/B9780124730526500068>.
- Therese Ann. Markow. *Drosophila : a guide to species identification and use*, 2006.
- Daniel Luis Simões Marta. Deep learning methods for reinforcement learning. 2016.
- J.H. Massey and P.J. Wittkopp. Chapter two - the genetic basis of pigmentation differences within and between drosophila species. In Virginie Orgogozo, editor, *Genes and Evolution*, volume 119 of *Current Topics in Developmental Biology*, pages 27–61. Academic Press, 2016. doi: <https://doi.org/10.1016/bs.ctdb.2016.03.004>. URL <https://www.sciencedirect.com/science/article/pii/S0070215316300965>.
- Kathleen Matthews and William Gelbart. Research resources for drosophila: the expanding universe. *Nature reviews. Genetics*, 6:179–93, 04 2005. doi: 10.1038/nrg1554.
- S.K. Mitra. *Digital Signal Processing: A Computer-based Approach*. McGraw-Hill international edition electrical engineering series. McGraw-Hill/Irwin, 2001. ISBN 9780072321050. URL <https://books.google.pt/books?id=SjNxQgAACAAJ>.
- Thomas B. Moeslund. *Introduction to Video and Image Processing - Building Real Systems and Applications*. Undergraduate Topics in Computer Science. Springer, 2012. ISBN 978-1-4471-2502-0. doi: 10.1007/978-1-4471-2503-7. URL <https://doi.org/10.1007/978-1-4471-2503-7>.
- Francis Jesmar Montalbo. Automated diagnosis of diverse coffee leaf images through a stage-wise aggregated triple deep convolutional neural network. *Machine Vision and Applications*, 33, 01 2022. doi: 10.1007/s00138-022-01277-y.
- W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019. doi: 10.1073/pnas.1900654116. URL <https://www.pnas.org/doi/abs/10.1073/pnas.1900654116>.
- Francisco Gerardo Medeiros Neto, Ítalo Rodrigues Braga, Matthew Henry Harber, and Iális Cavalcante De Paula. *Drosophila melanogaster* gender classification based on fractal dimension. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 193–200, 2017. doi: 10.1109/SIBGRAPI.2017.32.

- Patrick M O'Grady and Rob DeSalle. Phylogeny of the Genus *Drosophila*. *Genetics*, 209(1):1–25, 01 2018. ISSN 1943-2631. doi: 10.1534/genetics.117.300583. URL <https://doi.org/10.1534/genetics.117.300583>.
- Damian Podareanu, Valeriu Codreanu, Sandra Aigner, Caspar Leeuwen, and Volker Weinberg. Best practice guide - deep learning, 02 2019.
- Noha Radwan. *Leveraging Sparse and Dense Features for Reliable State Estimation in Urban Environments*. PhD thesis, 06 2019.
- Farheen Ramzan, Muhammad Usman Khan, Asim Rehmat, Sajid Iqbal, Tanzila Saba, Amjad Rehman, and Zahid Mehmood. A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks. *Journal of Medical Systems*, 44, 12 2019. doi: 10.1007/s10916-019-1475-2.
- Micael Reis, Gordon Wiegler, Julien Claude, Rodrigo Lata, Britta Horchler, Thuy Ha, Christian Reimer, Cristina Vieira, Jorge Vieira, and Nico Posnien. Multiple loci linked to inversions are associated with eye size variation in species of the *drosophila virilis* phylad, 03 2020.
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.
- Adrian Rosebrock. Implementing the perceptron neural network with python, 2021. URL <https://pyimagesearch.com/2021/05/06/implementing-the-perceptron-neural-network-with-python/>. Accessed December 2, 2022.
- Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *CoRR*, abs/1912.05911, 2019. URL <http://arxiv.org/abs/1912.05911>.
- Yefeng Shen, Md Hossain, Shafin Rahman, and Khandaker Asif Ahmed. Systematics of tephritid fruit flies: A machine learning based pest identification system. page 10400, 06 2021. doi: 10.3390/IECE-10400.
- M.N.H. Siddique and M.O. Tokhi. Training neural networks: backpropagation vs. genetic algorithms. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 4, pages 2673–2678 vol.4, 2001. doi: 10.1109/IJCNN.2001.938792.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL <https://arxiv.org/abs/1409.1556>.
- Anne Sonnenschein, David Vanderzee, William Pitchers, Sudarshan Chari, and Ian Dworkin. An image database of *drosophila melanogaster* wings for phenomic and biometric analysis. *Giga-Science*, 4, 05 2015. doi: 10.1186/s13742-015-0065-6.
- R Stephenson and Neil Metcalfe. *Drosophila melanogaster*: a fly through its history and current use the use of *drosophila melanogaster* in medical and scientific research. *The journal of the Royal College of Physicians of Edinburgh*, 43:70–75, 01 2013.
- Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019. doi: 10.48550/ARXIV.1905.11946. URL <https://arxiv.org/abs/1905.11946>.
- Sergios Theodoridis. *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, Inc., USA, 1st edition, 2015. ISBN 0128015225.

- TIBCO. What is a neural network? URL <https://www.tibco.com/reference-center/what-is-a-neural-network>. Accessed December 2, 2022.
- Seiichi Uchida. Image processing and recognition for biological images. *Development, growth & differentiation*, 55, 04 2013. doi: 10.1111/dgd.12054.
- Oregon State University. On convergence proofs for perceptions, 2023a. URL <https://classes.engr.oregonstate.edu/eecs/spring2021/cs519/extra/novikoff-1962.pdf>. Accessed January 5, 2023.
- Oregon State University. On convergence proofs for perceptions, 2023b. URL <https://classes.engr.oregonstate.edu/eecs/fall2017/cs534/extra/novikoff-1963.pdf>. Accessed January 5, 2023.
- Enrique Varela, E. Ulises Moya-Sanchez, Armando Aguilar-Meléndez, Octavio Castillo Reyes, Eduardo Vázquez-Santacruz, Sebastián Salazar-Colores, and Ulises Cortés. Detection, counting, and classification of visual ganglia columns of drosophila pupae. *Computacion y Sistemas*, 23:391–397, 06 2019. doi: 10.13053/CyS-23-2-3200.
- Denan Xia, Peng Chen, Bing Wang, Jun Zhang, and Chengjun Xie. Insect detection and classification based on an improved convolutional neural network. *Sensors*, 18:4169, 11 2018. doi: 10.3390/s18124169.
- Chengjun Xie, Jie Zhang, Rui Li, Jinyan Li, Peilin Hong, Junfeng Xia, and Peng Chen. Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning. *Computers and Electronics in Agriculture*, 119:123–132, 2015. ISSN 0168-1699. doi: <https://doi.org/10.1016/j.compag.2015.10.015>. URL <https://www.sciencedirect.com/science/article/pii/S0168169915003282>.
- Masamitsu Yamaguchi and Shinya Yamamoto. Role of drosophila in human disease research 2.0. *International Journal of Molecular Sciences*, 23:4203, 04 2022. doi: 10.3390/ijms23084203.
- Jonghwa Yim and Kyung-Ah Sohn. Enhancing the performance of convolutional neural networks on quality degraded datasets. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, 2017. doi: 10.1109/DICTA.2017.8227427.
- Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.
- Şaban Öztürk and Bayram Akdemir. Effects of histopathological image pre-processing on convolutional neural networks. *Procedia Computer Science*, 132:396–403, 2018. ISSN 1877-0509. doi: <https://doi.org/10.1016/j.procs.2018.05.166>. URL <https://www.sciencedirect.com/science/article/pii/S1877050918309001>. International Conference on Computational Intelligence and Data Science.