UNIVERSITY OF CRETE - DEPARTMENT OF PHYSICS

Machine Learning for Food Authentication

Authentication of honey based on its botanical origin through machine learning

---

Andreas Papanikolaou

Supervisor
Dr. George Tzagkarakis (FORTH-ICS)

Advisory Committee Members
Prof. George Tsironis (Physics Dept., University of Crete)
Dr. Anastasios Kampolis (CheMa Laboratories)

Abstract

Assessing the authenticity of honey is extremely valuable to the modern food market for both the consumer and producer. A way to categorize honey is based on its botanical origin. As the traditional methods require time-consuming and repetitive effort performed by highly specialized personnel, using computer assisted methods, when available, is extremely valuable. To address this problem, in this work we developed two computational pipelines, namely, (a) a statistical one, and (b) a machine learning one. Specifically, the first approach consists of a 2D discrete wavelet decomposition, coupled with statistical fitting of the wavelet coefficients, and a subsequent Support Vector Machine (SVM) classifier. The second pipeline involves the construction and training of Convolutional Neural Network (CNN) architecture. Experimental evaluation on a real dataset reveals that both approaches achieve a comparable classification accuracy for a moderate size of training data.

Contents

1 Introduction

1.1 Motivation

Food fraud is the act of tampering with food products and ingredients for economic gain. This is done through substitution and addition followed by misrepresentation. Such cases have a significant impact on public health and the global economy. For example, the European horse meat scandal of 2013 used DNA analysis to reveal that in some cases nearly 100% of the beef burger was actually horse meat. A more devastating case is that of the 2008 Chinese milk scandal, where milk, infant formula, and other food ingredients had been adulterated with melamine. Out of an estimated 294.000 victims, 54.000 babies were hospitalized and 6 died from getting kidney stones and severe kidney damage. As a result of many such cases, public interest in food authentication has risen and the field is growing rapidly. More specifically, honey as a natural and high value product is subject to adulteration and many types of fraud.

1.2 What is honey?

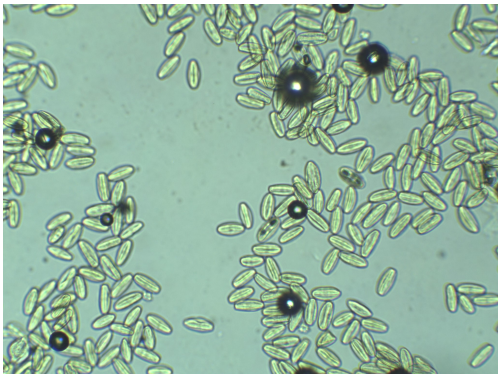
Honey is defined as the natural sweet substance produced by honey bees from nectar of blossoms or from secretions of living parts of plants or excretions of plant sucking insects on the living parts of plants, which honey bees collect, transform, and combine with specific substances of their own. Then store and leave in the honeycomb to ripen and mature [1].

1.3 Adulteration of honey

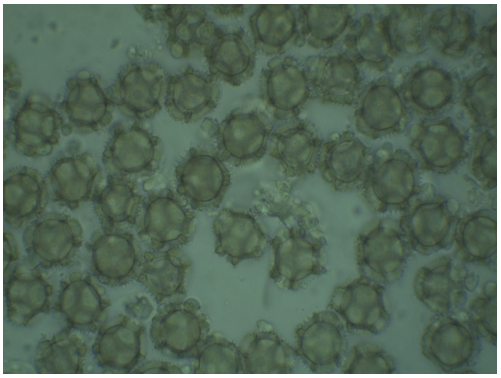
As a high value product, and since production is dominated by nonprofessional beekeepers, adulterated honey has a significant presence in the European market (Technical Round Table on Honey Authentication, JRC-Geel, Belgium 25 January 2018). According to the aforementioned report, honey adulteration can be prioritized into five major classes, namely:

1. Addition of sugar, through inappropriate bee feeding methods and/or from direct addition of sugar/syrup.
2. Mislabeling, with respect to botanical and geographical origin.
3. Resin treatment/ultrafiltration, commonly followed by blending with other types of honey and/or addition of pollen and enzymes to match higher priced types of honey.
4. Bee feeding, although widespread and accepted when necessary, is many times needlessly continued way after nectar flow has started.
5. Immature honey, industrially dried immature honey is sold as mature honey.

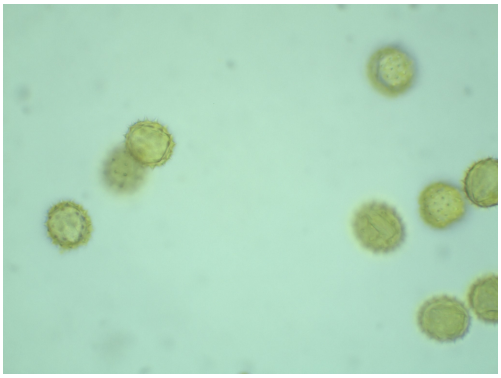
In this work, we focus on the botanical origin of honey. The traditional way to classify honey based on its botanical origin is through pollen investigation under a microscope. This method is carried by first chemically preparing and creating samples of honey and then carefully examining them to determine the types and concentrations of pollen present in the sample. This last step is time-consuming and requires focus, extended knowledge and attention to detail (Figure 1), and must be done by highly specialized scientists. As such, having this step done computationally is extremely valuable.



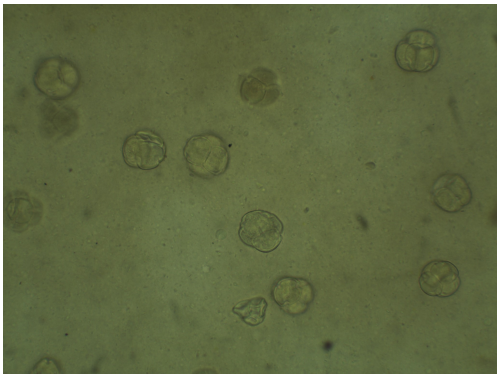
(a) Castanea



(b) Cichorium Intybus (Asteraceae)



(c) Inula Spinoza



(d) Calluna Vulgaris

Figure 1: Microscope pictures used for pollen investigation.

2 Background

2.1 Wavelet Transform

The wavelet transform is a form of signal analysis capable of extracting localized frequency information from a signal.

2.1.1 Fourier Transform

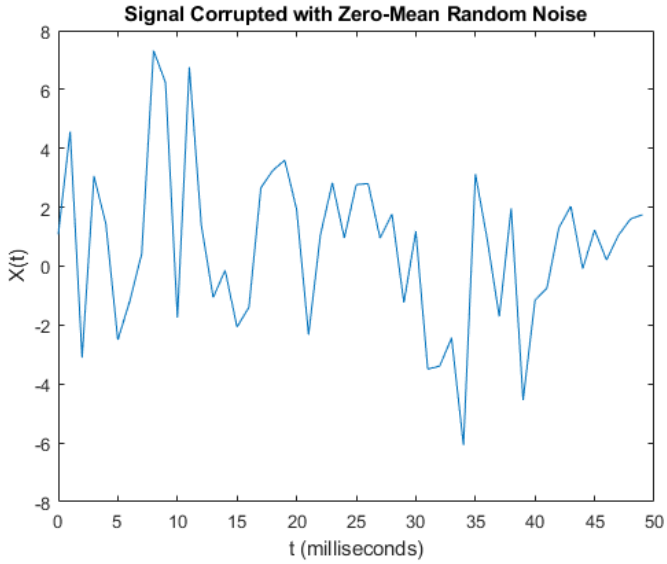
The Fourier transform predates wavelet analysis and offers a way to extract non-localized frequency information. It is vital in many problems of classical and quantum mechanics for calculating a physical system's time evolution determined by its initial and boundary conditions.

2.1.2 1D Continuous Fourier Transform

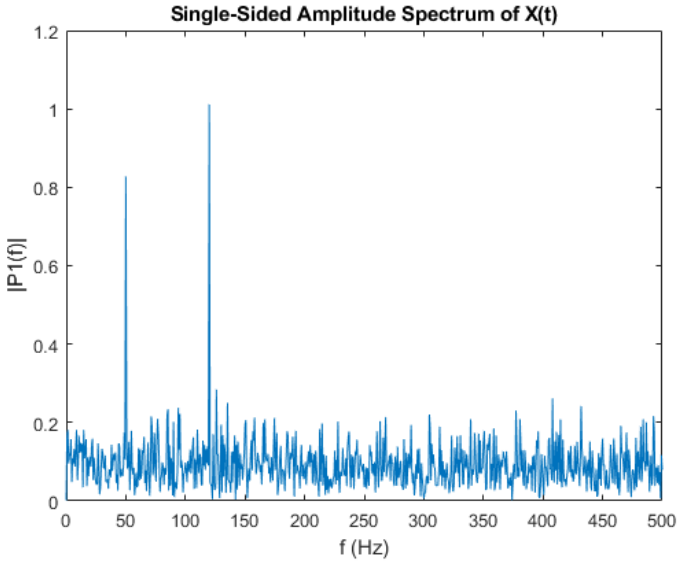
An analog signal function, f, of real values must be square-integrable to be admissible for the Fourier transform, which is defined by,

$$\mathcal{F}\{f(t)\} = F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,dt\ , \tag{2.1}$$

where $F(\omega)$ contains the global frequency content of f.



(a) Signal



(b) Continuous Fourier Transform

Figure 2: Example of 1D Continuous Fourier Transform (from Mathworks).

2.1.3 1D Discrete Fourier Transform

Most signals nowadays are digital, meaning they are the result of sampling an analog signal every T time units. The analog signal f becomes $f[n] = f(nT)|_{n\in\mathbb{Z}}$. The discrete Fourier transform is given by,

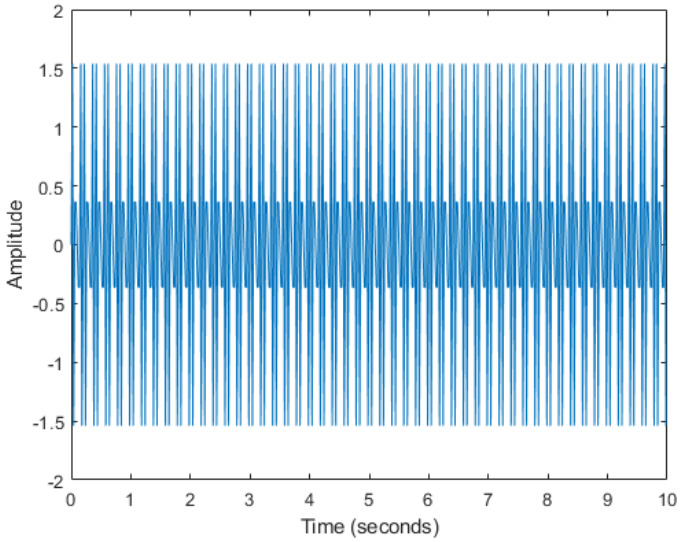
$$F[k] = \sum_{n=0}^{N-1} f[n]e^{-i2\pi kn/N}, k = 0, ..., N-1 \ , \tag{2.2}$$

and its inverse is obtained as follows,

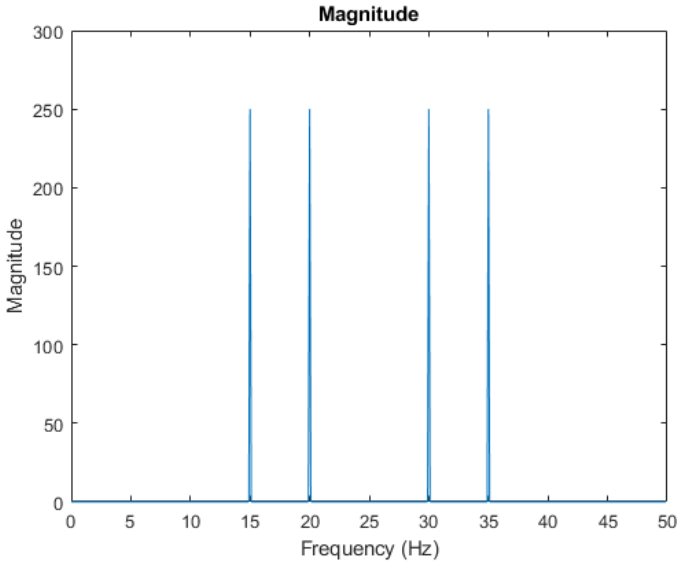
$$f[n] = \frac{1}{N} \sum_{k=0}^{N-1} F[k] e^{i2\pi kn/N}, n = 0, ..., N-1 \ . \tag{2.3}$$

The coefficients $F[k]$ are a measure of how much the signal f contains the frequency $\omega = \frac{2\pi k}{N}$.

For example, Figure 3 and Figure 4 show the signal amplitude over time and its discrete Fourier transform (frequency components) for two signals. Firstly, the sum of two sinusoidal signals of different frequencies, and secondly, the sound of a whale.

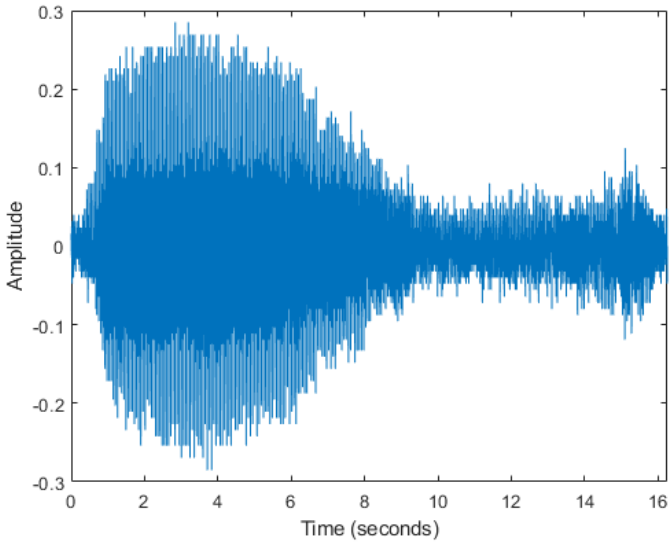


(a) Signal

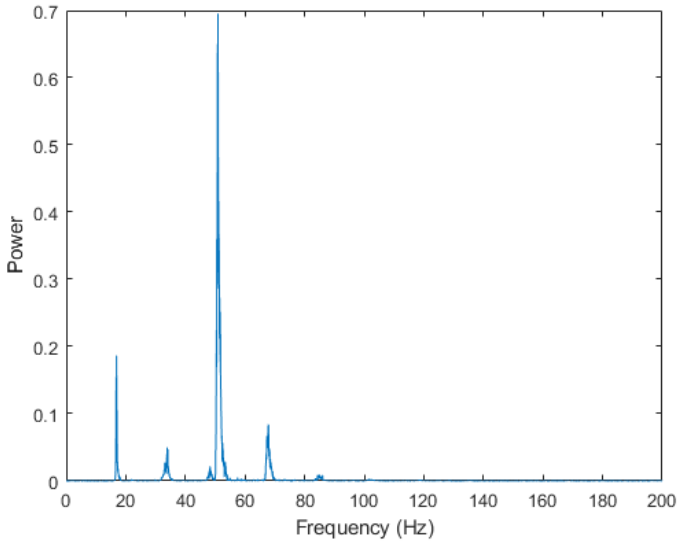


(b) Discrete Fourier Transform

Figure 3: Example of 1D Discrete Fourier Transform of $f(t) = \sin(2\pi 15t) + \sin(2\pi 20t)$ (from Mathworks).



(a) Signal



(b) Discrete Fourier Transform

Figure 4: Example of 1D Discrete Fourier Transform of a blue whale call (from Mathworks).

2.1.4 Short-Time Fourier Transform (STFT)

As we can see, the previous analysis can only reveal the frequency content globally. So the Fourier transform, as is, can only be useful when working with stationary signals. One way of tackling non-stationary signals is by breaking them down to pieces, since sufficiently large pieces can be assumed stationary. We can take a piece of a signal by multiplying it with a window function $g(t-\tau)$ to get $f(t)g(t-\tau)$, with this piece of the signal being centered around τ. We can now apply the Fourier transform on this piece and get a localized representation of the frequency content. By shifting the

window to different locations τ, and then taking its Fourier transform, we obtain the short-time Fourier transform (STFT) (Figure 5), defined by,

$$S_f(\tau, \omega) = \int_{-\infty}^{+\infty} f(t)g(t-\tau)e^{-i2\pi\omega t}\,dt\ . \tag{2.4}$$

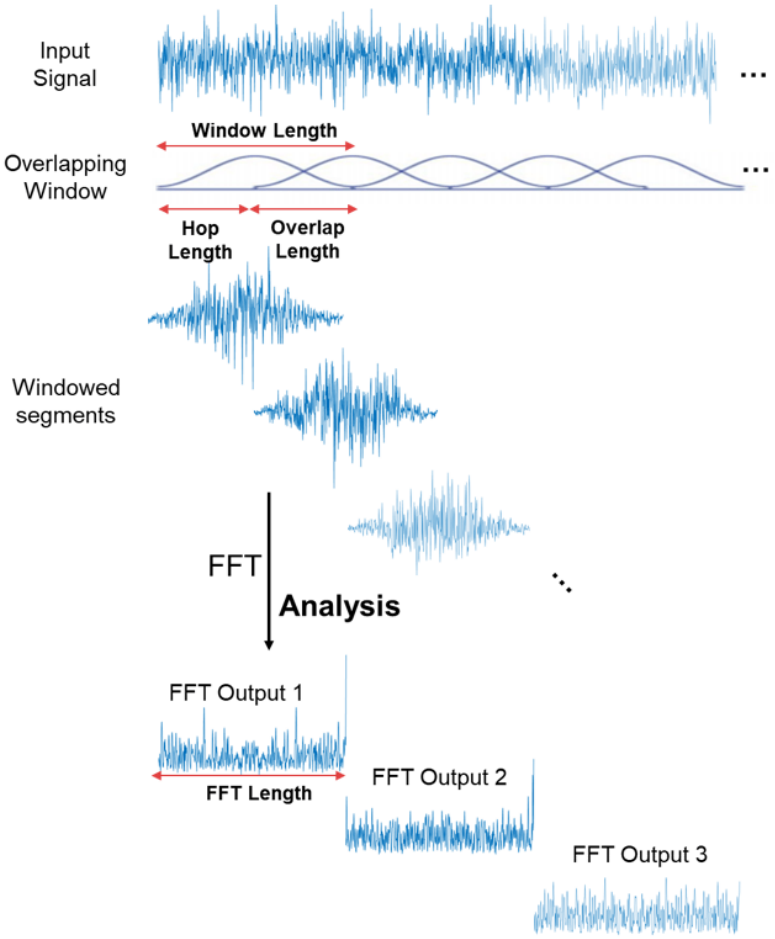


Figure 5: Illustration of STFT analysis (from Mathworks).

The STFT maps a signal $f(t)$ into a two-dimensional function in a time-frequency plane (τ, ω). Each coefficient $S_f(\tau, \omega)$ is a measure of the localized frequency content of the signal. More accurately, $S_f(\tau, \omega)$ tells us how much of the signal f in the neighbourhood of $t = \tau$ is made up of frequencies in the neighbourhood of ω. The time-frequency resolution, or more simply, the size of these two neighbourhoods, is determined entirely by the time-frequency spread of the window $g(t)$. As an illustration, Figure 6 shows the STFT of a signal produced by a voltage controlled oscillator output, controlled by a sinusoid.

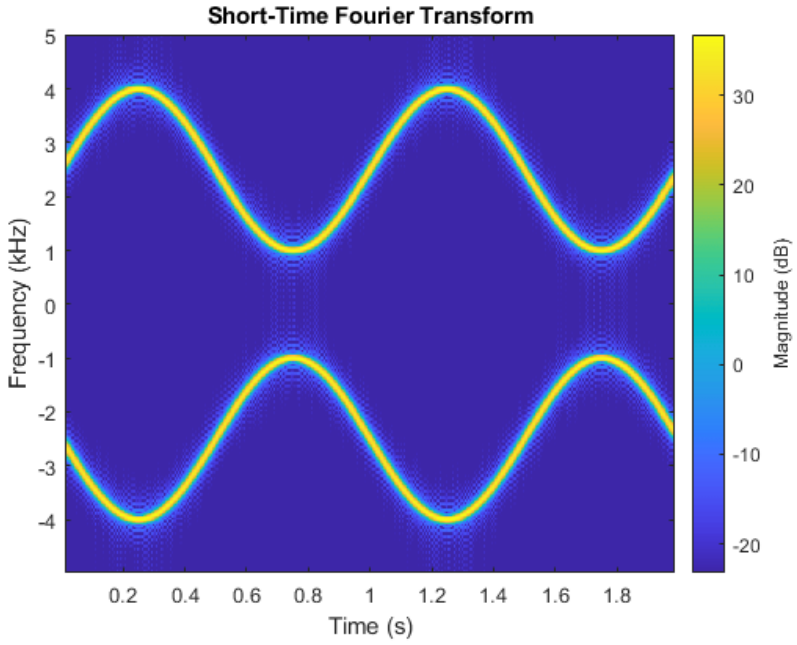


Figure 6: STFT analysis of a voltage controlled oscillator output (from Mathworks).

2.1.5 1D Continuous Wavelet Transform

The limitations of using a window of fixed time-frequency resolution over the whole signal become more apparent when taking into consideration the following. By nature, locating high-frequency components in a signal demands an adequately narrow time-domain window. In contrast, locating low-frequency components demands a sufficiently wide time-domain window to be used. The wavelet transform does exactly that. It uses a flexible time-frequency window that is wider when observing low-frequency components and narrower for the higher frequency ones.

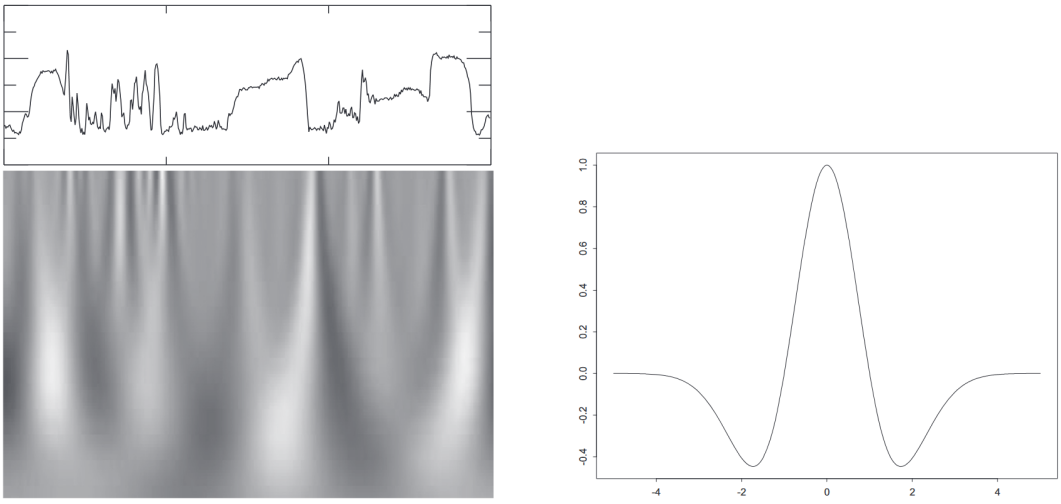
For a given signal function $f(t) \in L_2(\mathbb{R})$ using the Morlet-Grosmman definition [2], the wavelet coefficients are given by,

$$W(\alpha, b) = \frac{1}{\sqrt{\alpha}} \int_{-\infty}^{+\infty} f(t)\psi^* \left(\frac{t-b}{\alpha} \right) dt = f * \bar{\psi}_\alpha(b) \ , \tag{2.5}$$

where $\bar{\psi}_\alpha(t) = \frac{1}{\sqrt{\alpha}}\psi^*(\frac{-t}{\alpha})$, and

- $W(\alpha, b)$ is the wavelet coefficient of $f(t)$,
- $\psi(t)$ is the analyzing wavelet and $\psi^*(t)$ is its complex conjugate,
- $\alpha \in \mathbb{R}^+$ is the scale parameter,
- $b \in \mathbb{R}$ is the position parameter.

Figure 7a shows the wavelet coefficients (bottom) of a signal $f(t)$ (top) using the Mexican Hat wavelet (Figure 7b). Lower positions on the y axis refer to smaller scaling parameter α.



(a) 1D Continuous Wavelet Tranform (b) The Mexican Hat Wavelet

Figure 7: 1D Continuous Wavelet Transform.

At positions of high frequency changes we observe cones of higher wavelet coefficients that extend downwards.

2.1.6 1D Discrete Wavelet Transform

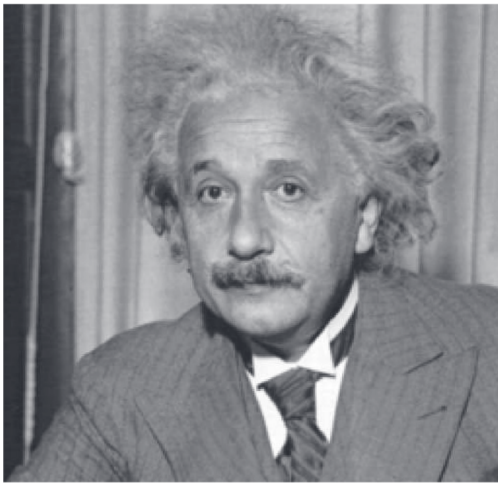
The 1D continuous wavelet transform can be carefully adapted, following Shannon's theorem [3] and Mallat's theory [4], to be applicable on 1D discrete signals. According to Mallat's multiresolution analysis theory, we can create a ladder of approximation subspaces with the embeddings,

$$\ldots \subset V_3 \subset V_2 \subset V_1 \subset V_0 \tag{2.6}$$

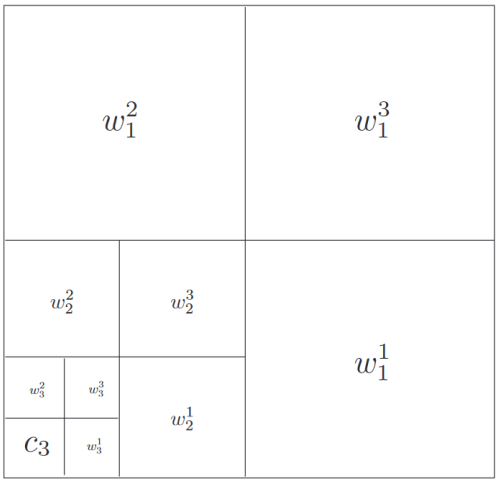
The signal function $f(t)$ can be projected on each level j onto the subspaces V_j and W_j, V_j's orthogonal complement in V_{j-1}. These projections hold the approximation, and detail coefficients accordingly. Moving down the ladder is done by projecting the details onto V_{j+1}, and so on. Projection is done through convolution with the discrete $h(t)$ (low-pass) and $g(t)$ (high-pass) filters, for the wavelet and detail coefficients followed by factor 2 decimation (i.e, subsampling).

2.1.7 2D Discrete Wavelet Transform

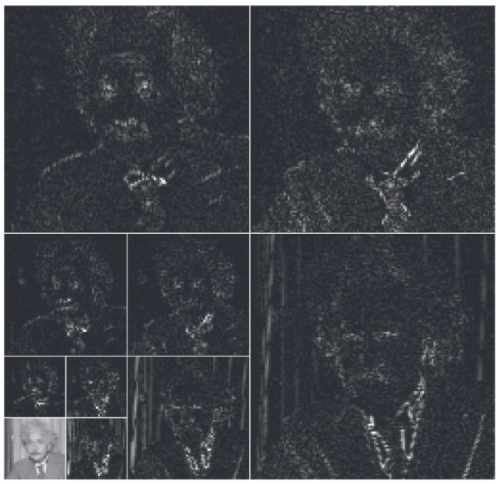
For 2D discrete signals, such as digital images, unlike 1D signals, orientation is important. That is why we need 3 wavelets (filters) one for each dimension and one for orientation. As a result, each level has 3 sets of wavelet coefficients. A horizontal, diagonal and vertical one. Figure 8 depicts the $n = 1, 2, 3$ orientations at each level $m = 1, 2, 3$, as well as the approximation c_3.



(a) Input image



(b) 2D DWT decomposition



(c) 2D DWT of input image

Figure 8: Example of 3-Level 2D Discrete Wavelet Transform.

2.2 Probabilistic Modeling

A probability distribution is a function that relates all possible outcomes of an experiment to their probability of occurrence [5]. For example, the probability distribution of a coin toss experiment is $P(heads) = P(tails) = \frac{1}{2}$. That is because after a sufficiently large number of experiments N, the ratio of heads outcomes over N will be

close to $\frac{1}{2}$. In this case, P is a discrete probability function. A continuous probability function is useful in experiments that count, for instance, the ratio of male to female students $x = \frac{male}{female}$ of a university class. Because the associated random variable is continuous, calculating the probability of occurrence of one of infinite number of outcomes is impossible. It makes more sense to use $P(0.51 < x < 0.52)$ or generally $P(a < x < b)$, and to define a function f such as,

$$P(a < x < b) = \int_a^b f(x)\,dx \ . \tag{2.7}$$

Function f is a probability density function (PDF), which is further used to define other useful functions such as,

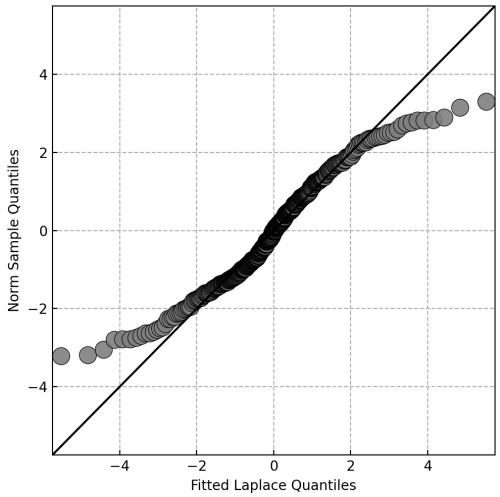
- the cumulative distribution function (CDF), $F(x) = \int_{-\infty}^{x} f(x)\,dx$, that evaluates the probability of getting a value $\leq x$.
- and its inverse, the quantile function, that finds the x such that, with probability q, all outcomes of a random variable X will not exceed x.

2.2.1 Probability distributions - Gaussian vs non-Gaussian

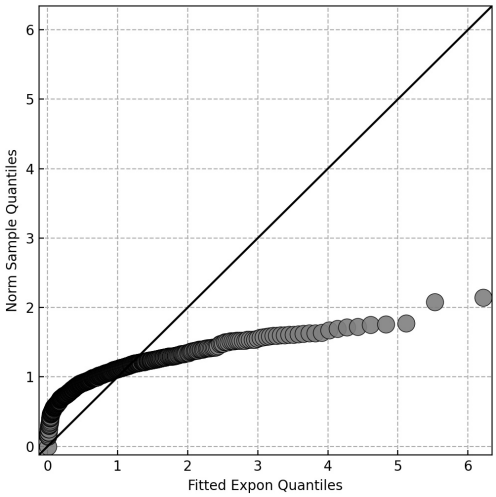
There are many families of probability distributions that are used for modeling a wide range of systems. By far, the most commonly used one is the Gaussian, as it is fairly successful in many applications and typically yields analytical solutions. However, the Gaussian assumption used in a non-Gaussian environment leads to significant performance degradation (ref. [6]). A feature shared among non-Gaussian phenomena is impulsiveness. This behaviour is expressed by the heavy-tail characteristics of such distributions, meaning that their PDFs have a lower rate of decay. Thus the probability of extremely large observations is non-negligible.

2.2.2 Quantile-quantile plots

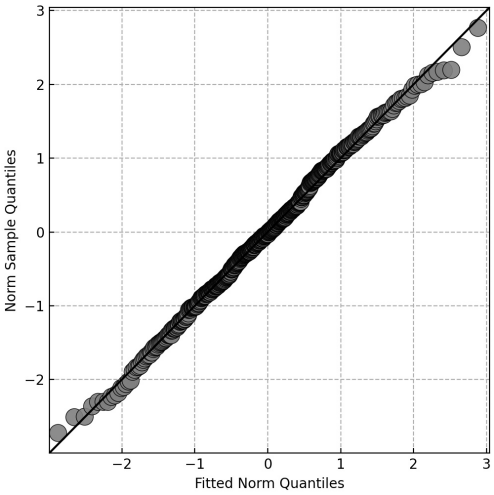
Quantile-quantile (or Q-Q) plots are useful for comparing two probability distributions. They are created by plotting the two distributions quantiles against each other. For example, in Figure 9 the quantiles of a Gaussian distribution (sampled 500 times) are plotted against itself and those of two other distributions (Laplace and Exponential). Notably, a Q-Q plot of two similar distributions will have many points around the $45°$ line, which is exactly what we see at Figure 9c.



(a) Normal vs Laplace



(b) Normal vs Exponential



(c) Normal vs Normal

Figure 9: Comparison of two probability distributions through Q-Q plots.

2.3 Classification Methods

In supervised machine learning, classification models use a set of features as input to predict an instance's class based on previous optimization and fitting of the model based on the data available.

2.3.1 Support Vector Machines

Support Vector Machines (SVMs) (ref. [7]) are one of the many types of models that are used extensively for data classification. A barebones SVM can tackle linear binary classification problems (ref. Figure 10) by finding the widest possible zone (called

large margin classification) that can split our data in two subsets. The zone is fully determined by the instances located on its edges (called the support vectors).

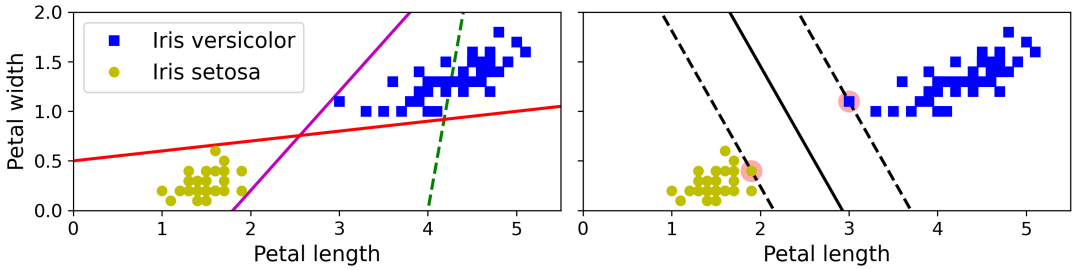


Figure 10: Large margin classification on the Iris dataset.

Although really useful, this type of model breaks down in cases where the classes are not linearly separable, such as in Figure 12, or contain outliers, such as in Figure 11. Thankfully, the model can be adapted to overcome such limitations.

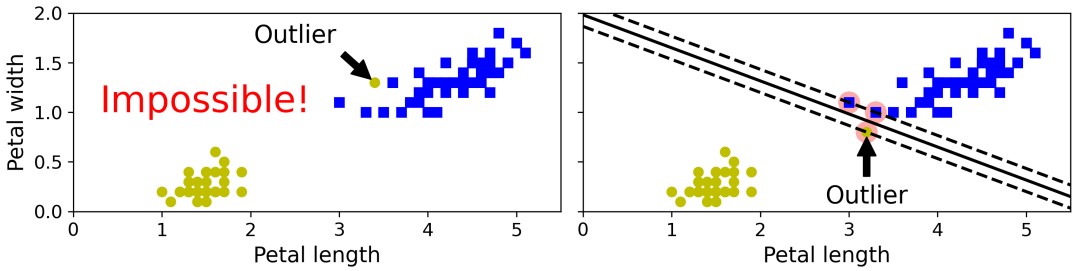


Figure 11: Case with outliers.

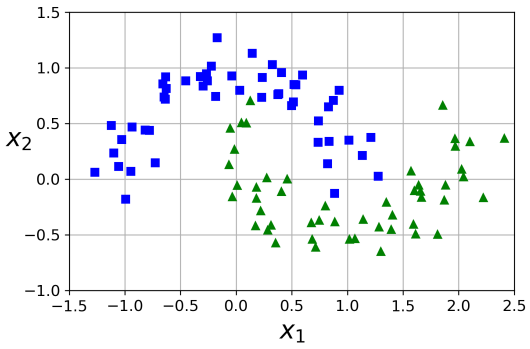


Figure 12: Non-linearly separable data.

2.3.2 Soft Margin Classification

The aforementioned zone can be modified to have soft margins that allows instances to be in it (see Figure 13). In such models, softness can be controlled through a

hyperparameter. Soft margins are less sensitive to outliers and can be used to classify mixed-class data.

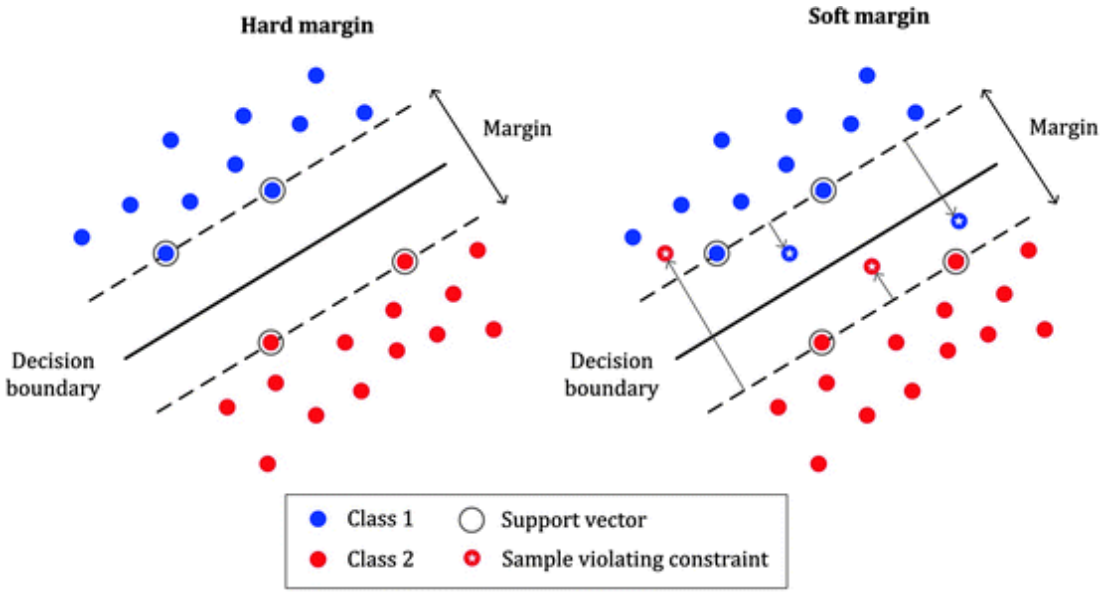


Figure 13: Soft margin classification.

2.3.3 Nonlinear Classification - Polynomial Features

One way to classify nonlinear data is by adding more features, such as polynomial features (ref. Figure 14), expecting that in the new feature space the classes will be better separable. This may not be possible for all cases though. A possible solution is to try adding features of a higher degree, but as the number of features increases so does the computational complexity of the model slowing it down.

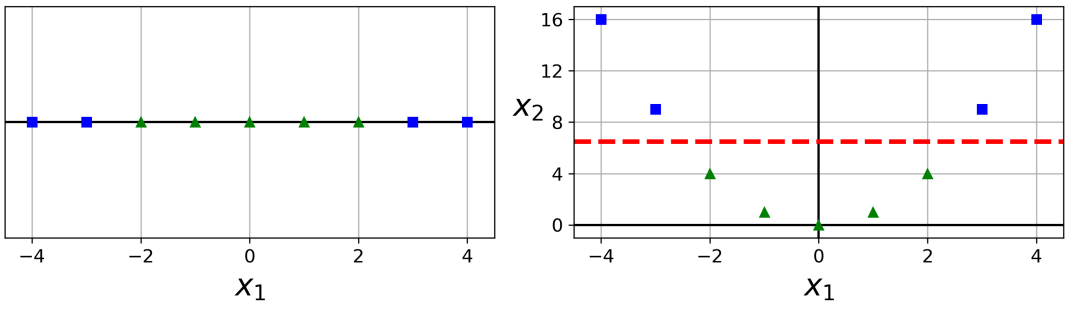


Figure 14: Nonlinear classification using polynomial features.

2.3.4 Nonlinear Classification - Similarity Features

Similarity features are useful metrics of resemblance between an existing feature and a landmark value of that feature. Landmark position and the similarity function can

be chosen freely by the user. Similarly to polynomial features, a large number of landmarks (added features) used can slow down computation.

A commonly used similarity function is the Gaussian Radial Basis (RBF) function, defined by

$$\phi_\gamma(\boldsymbol{x}, \boldsymbol{l}) = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{l}\|^2) \ . \tag{2.8}$$

Figure 15 shows the values of ϕ for two landmarks l_1, l_2, applied on feature x_1. Plotting points of $(\phi(x, l_1), \phi(x, l_2))$ for each instance we see at the right figure that the transformed dataset is now linearly separable.

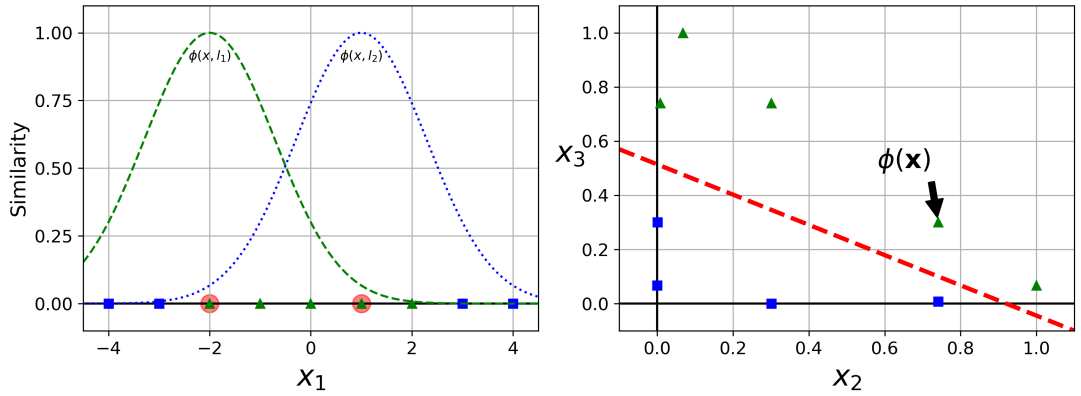


Figure 15: Nonlinear classification using similarity features.

2.3.5 The Kernel trick

The methods presented in Section 2.3.3 and Section 2.3.4 are capable of performing nonlinear classification at the cost of greatly increasing the computational complexity of the model. Thankfully, there is a way to get the same results while avoiding an increase of computational cost. A mathematical technique, called the kernel trick, can be used to substitute a part of the model's fitting calculations. For example, adding 2nd degree polynomial features essentially maps $\boldsymbol{a} = (x_1, x_2)$ to $\phi(x) = (x_1, x_1^2, x_2, x_2^2)$. The fitting algorithm then calculates $\phi(\boldsymbol{a})^T\phi(\boldsymbol{b})$, with $\boldsymbol{a}$ and $\boldsymbol{b}$ being the feature vectors of two instances. The kernel technique simplifies $\phi(\boldsymbol{a})^T\phi(\boldsymbol{b})$ to $K(\boldsymbol{a}, \boldsymbol{b})$. Typical examples of kernels are given below.

- Linear: $K(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T\boldsymbol{b}$
- d-degree polynomial: $K(\boldsymbol{a}, \boldsymbol{b}) = (\gamma\boldsymbol{a}^T\boldsymbol{b} + r)^d$
- Gaussian RBF Linear: $K(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma\|\boldsymbol{a} - \boldsymbol{b}\|^2)$
- Sigmoid: $K(\boldsymbol{a}, \boldsymbol{b}) = \tanh(\gamma\boldsymbol{a}^T\boldsymbol{b} + r)$

2.4 Deep neural networks for classification

Neural networks can be seen as a "black box" capable of simulating very complex functions. Although there is nothing secret or hidden about it, one can not develop an intuitive understanding about its inner workings by simply looking at the weights.

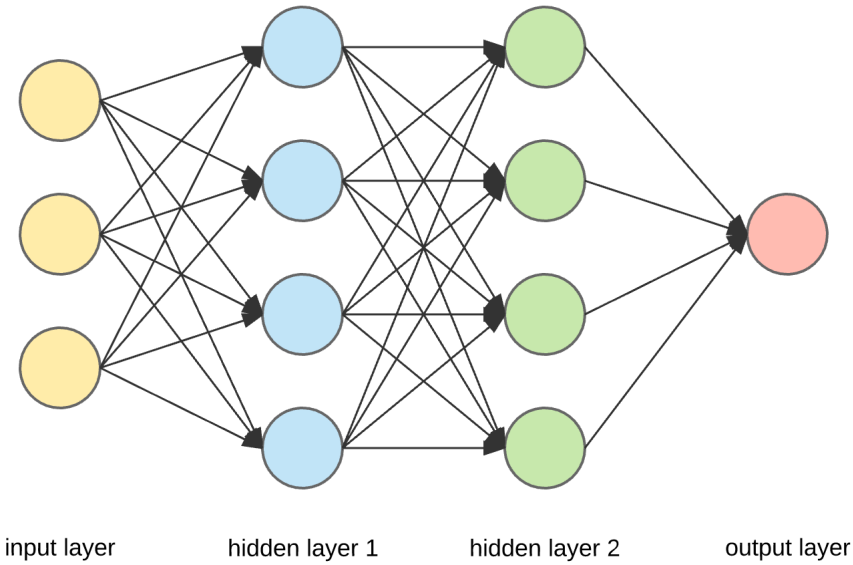


Figure 16: Simple neural network architecture (from towardsdatascience).

In Figure 16, each layer has a number of nodes and each node is connected to other nodes. Each connection has a weight and each node has an activation function as shown in Figure 17. The goal of fitting a neural network is to tune the weights by using the data available.

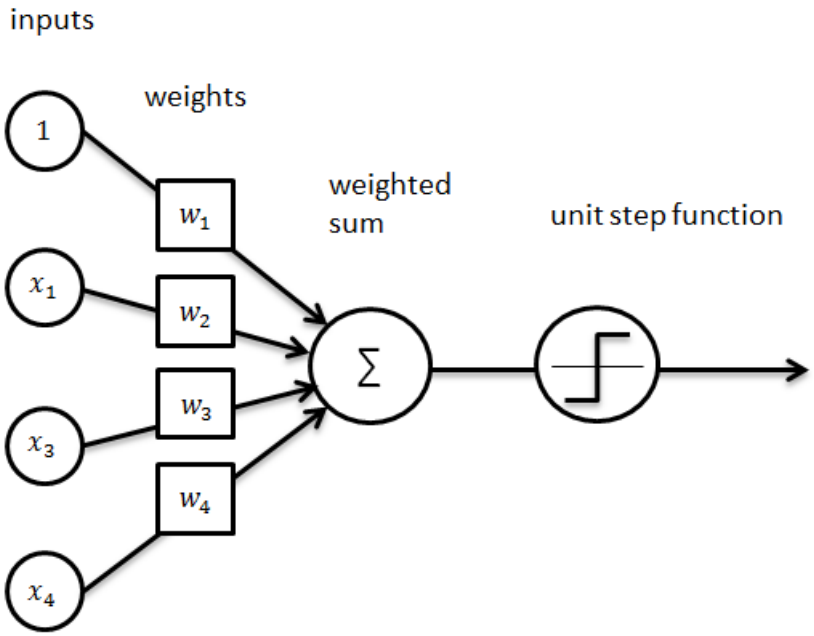


Figure 17: Node diagram.

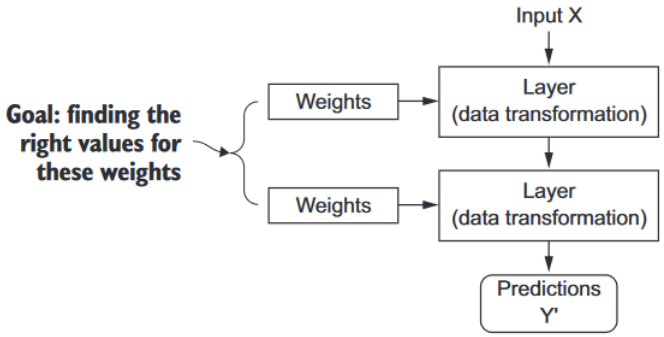


Figure 18: A neural network is parameterized by its weights (ref. [8]) .

To calculate the change of weights we need a metric that compares prediction made by the network for a given input with the true target. To this end, a loss function is employed, which, given two or more outputs, it can quantify their distance. Lower loss values are related with better predictions.

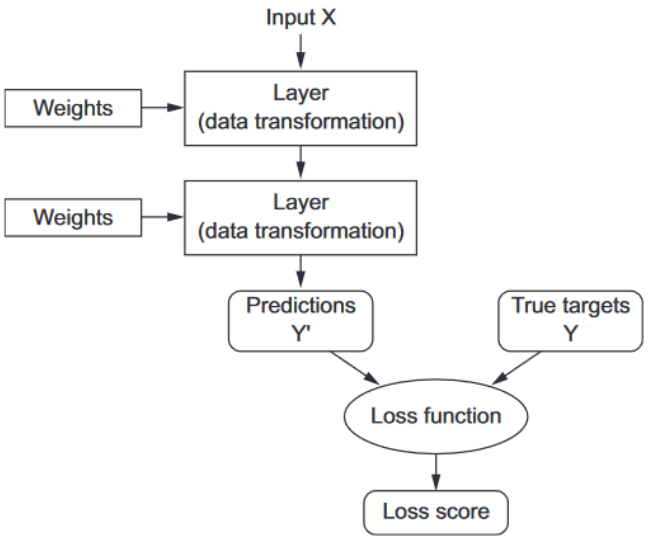


Figure 19: Scoring process (ref. [8]) .

Finally, we need to update the weights in a way that reduces the loss score. For this, an appropriate optimizer is employed.

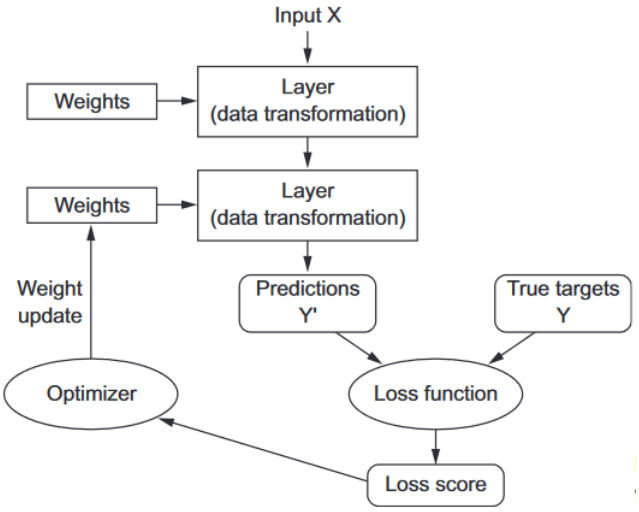


Figure 20: Scoring (ref. [8]).

The optimizer aims at reducing the weights of our model's loss score. Continuing in this manner one can get a model that performs well on both seen and unseen inputs (of similar nature).

2.4.1 Convolutional layers

Figure 16 shows the functionality of dense neural layers, which are useful when dealing with non localized input features. When it comes to images, relations between

neighbouring pixels matter a lot because these relations give rise to patterns. A great way to take advantage of these relations is to use convolutional layers instead of dense ones. A convolutional layer has filters (or kernels) in a similar way a dense layer has nodes. Performing a convolution between an image and a kernel (visualisation) can blurr figure 21, sharpen or even detect edges on an image. It all depends on the kernel's values. During training these values are being changed during training with the goal of extracting the most appropriate patterns for the task at hand.

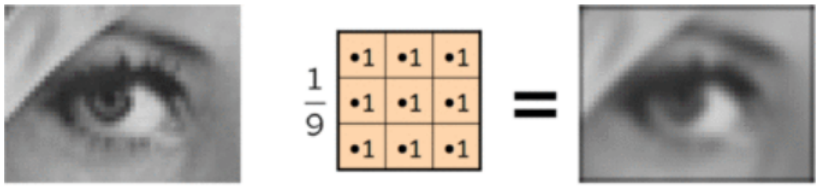


Figure 21: Blurring effect.

For more examples click here.

3 Data Description

Our dataset consists of classified digital images most commonly used during palynological analysis. The images were captured and provided to us by CheMa Laboratories (Korinthos, Greece) using the following method (also implemented in [9]).

A quantity of 5–10 mg of each sample was dispersed in 1 mL of deionized water using a vortex mixer. A total of 0.5 mL of each suspension was spread on a 22×22 mm area of a microscopy slide and was left to dry at 39°C. After preheating a cover plate on the hotplate at the same temperature, a drop of glycerol gelatin was transferred to it. This held the pollen in position while the cover glass was lowered onto the dried sediment. The pollen grains were identified using an Euromex BioBlue optical microscope at $\times 400$ magnification. For the identification of the pollens, the following databases were used: Pollen Atlas, pollen Wiki database, and the pollen library at the CheMa Laboratories. Table 1 contains the class populations in descending order.

Class Name	Class ID	Number of Instances
Castanea sp.	2	109
Thymus capitatus	15	101
Calluna vulgaris	13	101
Inula Spinoza	10	8
Helianthus	7	2
Origanum	11	2
Pinus	12	1
Sonchus	14	1
Hibiscous	9	1
Heliotropium	8	1
Hedera Helix	6	1
Eukalyptus sp.	5	1
Citrus sp.	4	1
Tribulus sp.	17	1

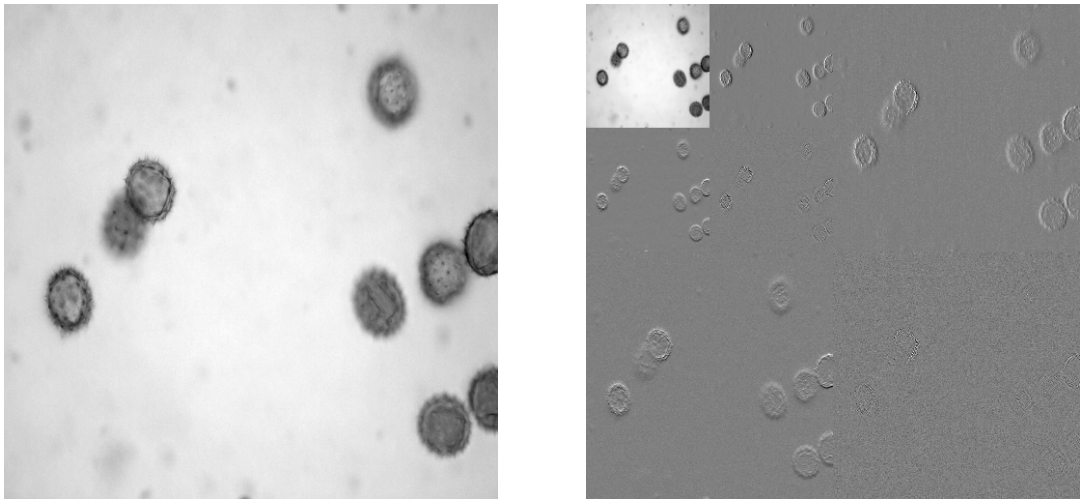
Table 1: Class populations.

4 Experimental Evaluation

4.1 Classification through wavelet analysis and SVMs

This approach improves upon the traditional pollen investigation method of determining the botanical origin of honey by handling the last step of image classification through machine learning. We trained a SVM model on data containing classified images of honey. The pipeline consists of these steps:

1. Wavelet decomposition of the black and white images (Figure 22) using the PyWavelets python package.



(a) Input (b) Output

Figure 22: Two level wavelet decomposition using Daubechie's first order wavelet of an Inula Spinoza honey image.

2. Flattening of each subband to a vector and then fitting of different probability distributions to these vectors to acquire the distribution parameters per subband using Matlab (Table 2).

	3 1 1	3 1 2	3 2 1	3 2 2	3 3 1	3 3 2
0	0.00	31.24	-0.06	29.46	0.00	13.71
1	-0.12	30.67	-0.09	29.47	-0.01	13.99
2	-0.04	24.06	0.04	23.15	0.00	7.76
3	-0.04	42.04	-0.04	40.06	-0.01	14.93
4	-0.07	20.51	0.05	19.90	-0.00	6.96

Table 2: A subsection of the distribution parameters dataset calculated for 3 levels of wavelet decomposition using the db4 wavelet and later fitting of each flattened subband image to a Normal distribution (requiring 2 parameters for complete definition). The code 3 1 1 refers to (Level 3, Orientation 1, Parameter 1).

3. Using each column of the distribution parameters dataset (Table 2) as a feature to train a multiclass SVM model (OneVsOne strategy) using the Scikit-Learn python package. We used the OneVsOne strategy because SMV classifiers scale poorly with the size of the training set.

The training procedure has several hyperparameters that need tuning.
Namely:

- SVM hyperparameters: γ (regulating the RBF) and C (regulating margin softness).
- Wavelet hyperparameters: the levels and wavelet type used for the decomposition.
- Probability distributions used: Normal, Stable, tLocationScale.

For the purposes of this thesis the number of wavelet hyperparameters and probability distributions tested was limited by the RAM memory (16 GB) and computing power (AMD Ryzen 5 2600 Six-Core Processor) available. The 3 level wavelet decomposition of the 364 images takes 15'-20' while the subsequent fitting of the distributions takes: 10' for the Normal distribution, ~ 5 hours for the t-Location Scale distribution and 7-8 hours for the Stable distribution. On the other hand 2000 runs of the SVM training procedure take less than 3'. As such, only 3 levels and the db4 wavelet were used for the decomposition.

The outline of our cross validation workflow is shown at Figure 23. We first keep $\frac{1}{3}$ of the data hidden so we can later compute the confusion matrix and other metrics of our best predictor. To find our predictor we selected a subset of the images containing the most images per class and used n-fold cross validation (CV split in Table 3 refers

to n). This method of evaluation first splits the whole dataset into n random sets of equal size and then using n-1 sets for training and 1 set for testing. We do this so all n sets are used for testing, the result of our evaluation is the mean accuracy and its standard deviation calculated using our n tests. We also probed to estimate the importance of class imbalances in the data by randomly selecting a number of images from the most populated classes to match their numbers to those of the least populated class. Each n-fold CV, and the balancing selection was repeated 100 times to measure the reliability of the resulting mean accuracy through its standard deviation. The results for $\gamma = 0.01$ and $C = 1$ can be seen at Table 3.

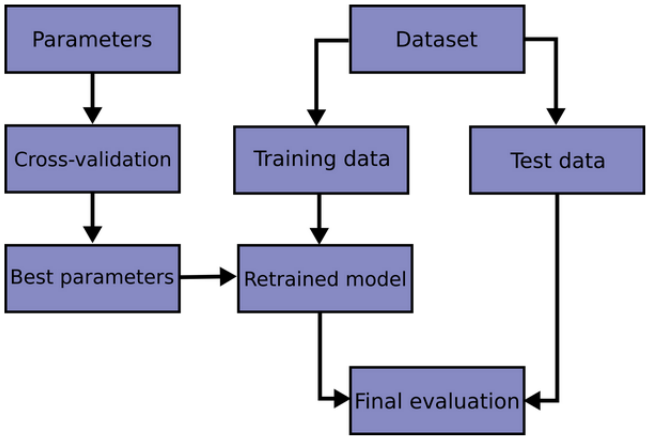


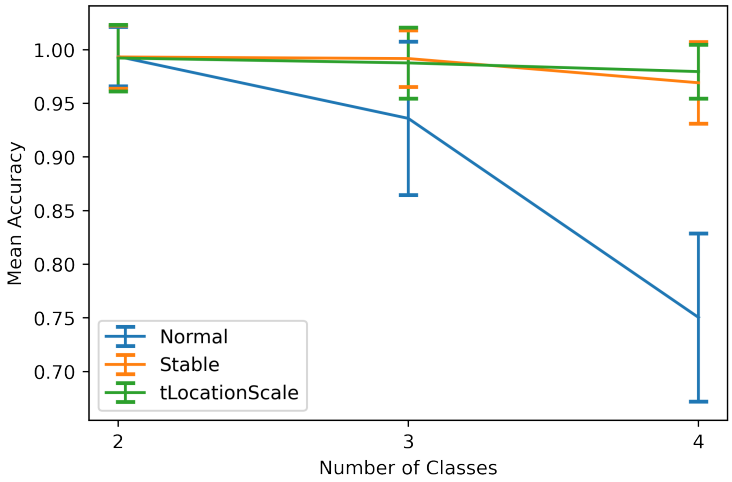
Figure 23: Validation workflow (click here).

Classes	Images per class	Distribution	CV split	Mean Accuracy	σ of Mean Accuracy
2	63	Normal	20	0.9937	0.0277
2	63	tLocationScale	20	0.9922	0.0310
2	63	Stable	20	0.9932	0.0294
3	63	Normal	20	0.9360	0.0715
3	63	tLocationScale	20	0.9877	0.0331
3	63	Stable	20	0.9918	0.0265
4	6	Normal	3	0.7504	0.0783
4	6	tLocationScale	3	0.9796	0.0250
4	6	Stable	3	0.9692	0.0381

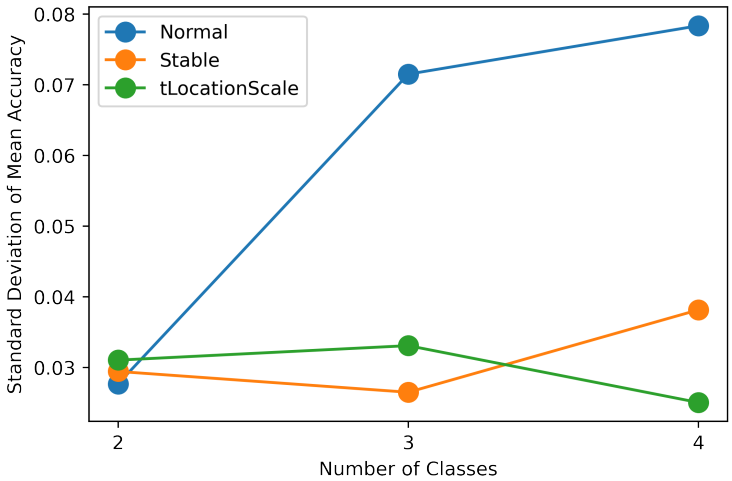
Table 3: Cross validation results as the number of classes increases while keeping the classes balanced. The 2 class evaluation used classes 2 and 15 from Table 1 (Class ID) while the rest used the n most populated classes.

Keep in mind that as the number of classes gets larger the number of splits (CV split) must be lowered because otherwise we would end up with folds of 1-2 images each severely lowering the quality of our evaluation.

Plotting (Figure 24) the mean accuracy and its standard deviation as the number of classes increases helps shows the decreasing trend of the mean accuracy as the number of classes increases most likely due to lower number of images used.
The stable and t-LocationScale distributions show greater accuracy and reliability for all evaluations thus validating the notion that heavy tailed distributions outperform Gaussians.



(a) Mean Accuracy



(b) Standard deviation of mean accuracy

Figure 24: Mean accuracy and its standard deviation as the number of classes used increases.

Lastly, we retrained our 3 class Stable distribution model and used the hidden data to calculate the confusion matrix (Figure 25). As you can see the model scored perfectly, but these results are too good to be true. More data is needed for a reliable evaluation of this method. Considering that the expected accuracy is close to 99% having 1 missclassification on a sample of 100 images is normal.

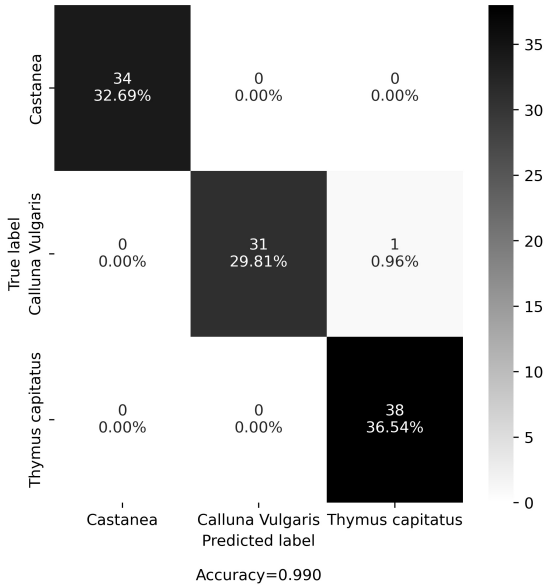


Figure 25: Confusion matrix.

4.2 Classification through Convolutional Neural Networks

This section illustrates the performance of convolutional neural networks for the classification of our image dataset. We emphasize that, due to the dataset size requirements, the analysis is restricted to the top 2 and 3 most populated classes. Furthermore, RAM and VRAM limitations necessitate a prior image resize. Specifically, we first converted the images from .tiff to .jpg (.tiff is not supported by Keras at the moment) and then resized them from 2040×1528 to 256×341 ($\sim 17\%$ of the original size). This comes along with a reduction of the overall visual information available to our model, compared to the wavelet method.

The proposed computational pipeline consists of the following steps:

1. Preprocessing: convertion to .jpg and resizing to 17% of the original size.
2. Rescalling from 1-256 to 0-1 as common practices dictate.

3. (Optional) data augmentation by randomly flipping (vertically and horizontally) and rotating the original images (see Figure 26).
4. Splitting the dataset to 2 sets, for training and testing.
5. Fitting the deep convolutional network. Figure 27 visualises a run of the fitting step.

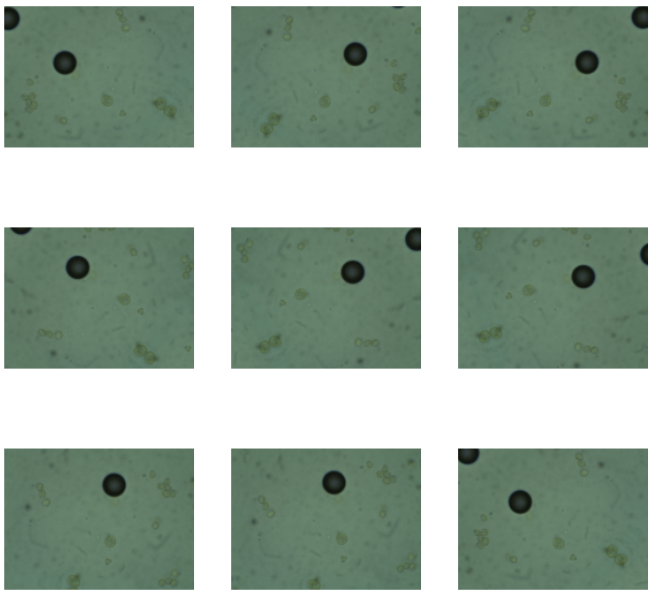
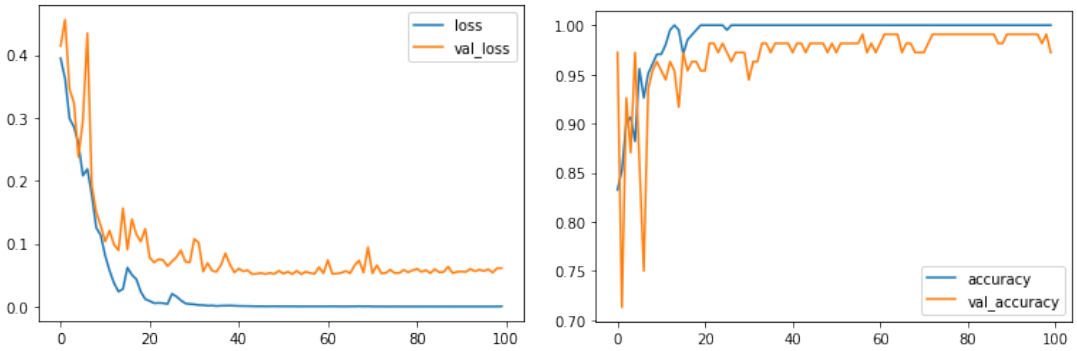


Figure 26: Example of image transformation for data augmentation.



(a) Loss versus epochs.

(b) Accuracy versus epochs.

Figure 27: Loss and accuracy as epochs increase.

The baseline architecture used is shown in Figure 28. The convolutional layer's activation function is ReLU, max pooling size is fixed at (2,2), dropout rate is equal to 0.5 and the dense layer's activation function is softmax. We used the ADAM optimizer with a learning rate of 10^{-3} as implemented by Keras.

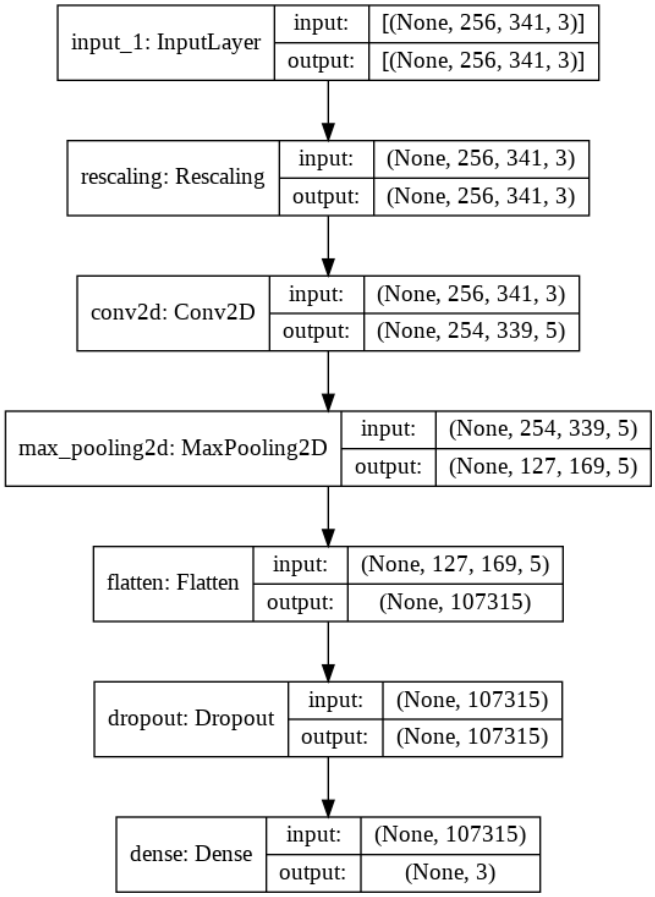


Figure 28: Architecture diagram.

We again followed the validation workflow shown in Figure 23. We kept $\frac{1}{3}$ of the data hidden so we can later compute the confusion matrix and other metrics of our best predictor. With the rest of the data we performed a grid search for up to 200 epochs with early stopping (patience 20, monitoring accuracy) of the following parameters using 3-fold cross validation (using sklearn's GridSearchCV method and Keras).

- Batch size: 8, 16, 32
- Depth: 2 to 5 subsequent pairs of convolutional and max pooling layers.
- With and without data augmentation.

Table 4 shows the top 10 results based on mean accuracy.

Mean Accuracy	σ of Accuracy	Batch size	Depth	Data augmentation
1.000000	0.0000	8	2	False
0.995238	0.0067	16	2	False
0.995169	0.0068	32	4	False
0.995169	0.0068	32	2	False
0.980814	0.0136	32	4	True
0.976052	0.0179	8	3	False
0.966322	0.0181	32	2	True
0.961560	0.0180	32	3	False
0.961422	0.0182	8	5	False
0.956729	0.0003	16	3	False

Table 4: Top 10 grid search results for 3 classes in ascending order of Mean Accuracy.

According to Table 4 the best model has a depth of 2, does not use data augmentation and has a batch size of 8. We repeatedly validated this model (100 times) on the aforementioned $\frac{2}{3}$ of the data using n-fold cross validation (cv split 3) and it achieved a mean accuracy of 0.9833 with a standard deviation of 0.0171. Lastly we trained this model on the $\frac{2}{3}$ with early stopping (patience 20, monitoring accuracy) for a maximum of 200 epochs. We then calculated the confusion matrix (Figure 29) using the hidden $\frac{1}{3}$ of data to see that it performed perfectly. The reason is once again the lack of data to properly validate a model of $\sim 98\%$ accuracy.

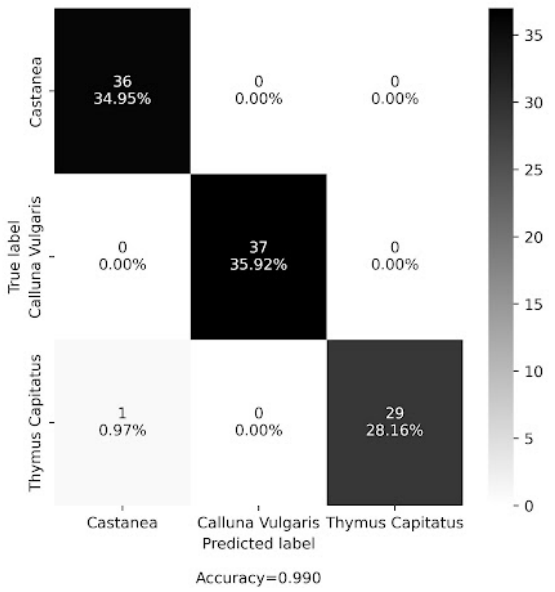
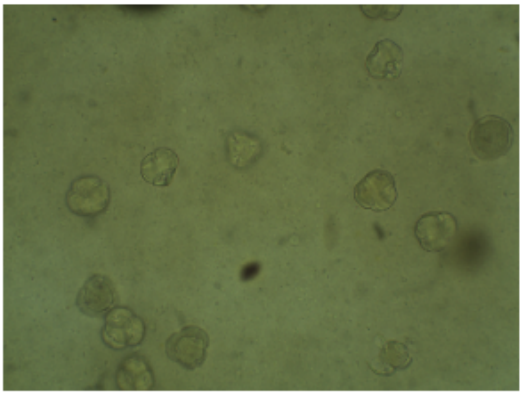


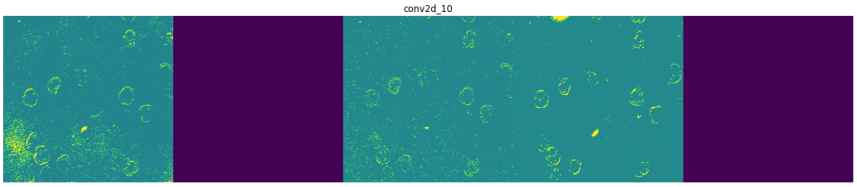
Figure 29: Confusion matrix.

4.2.1 The activations of the convolutional layers

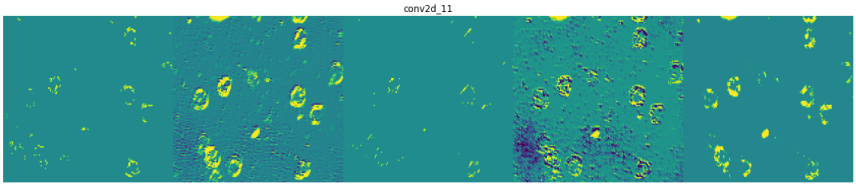
We are able to peek into the network and see the transformation of an image as it passes through the network. We can try to understand what kinds of patterns are being encoded by the filters but they do not necessary have to make sense to us as long as they are good enough for the network.



(a) Input Image

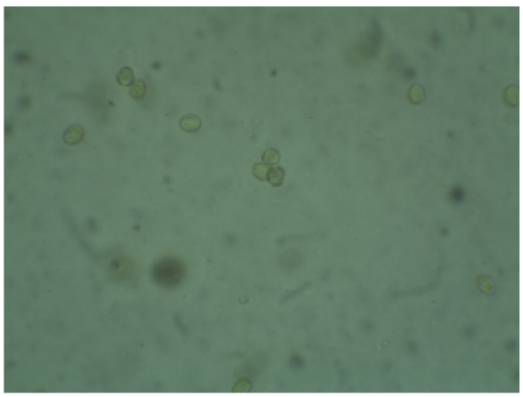


(b) First convolutional layer

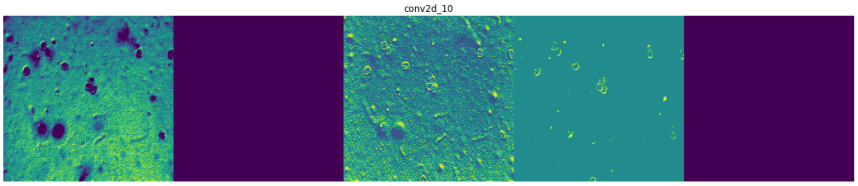


(c) Second convolutional layer

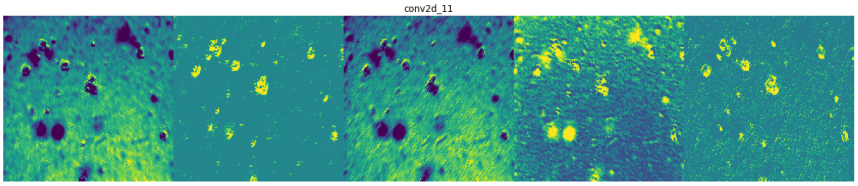
Figure 30: Convolutional layer activations



(a) Input Image

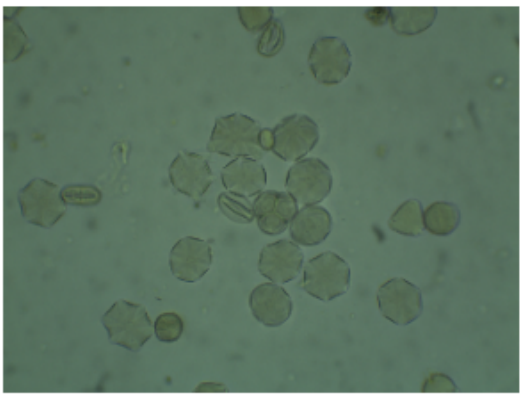


(b) First convolutional layer

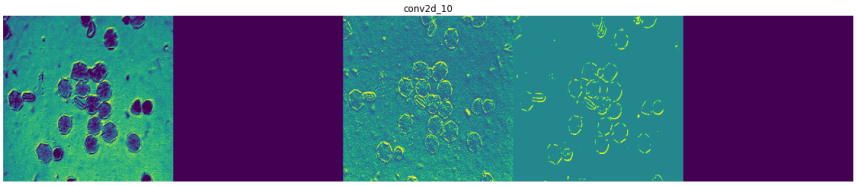


(c) Second convolutional layer

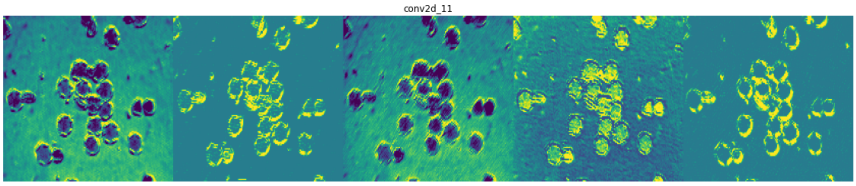
Figure 31: Convolutional layer activations



(a) Input Image



(b) First convolutional layer



(c) Second convolutional layer

Figure 32: Convolutional layer activations

5 Conclusions and Future Work

To conclude, for the given dataset both computational pipelines yielded an equally high accuracy (close to 99%). Regarding the first pipeline, it is important to note the improved modeling capabilities of heavy-tailed distributions for the statistical fitting of the wavelet coefficients. Notably, this pipeline revealed a very promising performance in the case of small-sized datasets. Concerning the second pipeline, despite the small size of the dataset (~ 300 pictures) compared to the 70.000 images in the MNIST dataset (usually used to introduce people to deep learning) it still resulted in a satisfactory accuracy of $\sim 98\%$. We emphasize though that the validation through the confusion matrices and its related metrics is not that insightful due to the size of the dataset.

Despite the promising performance of the two proposed pipelines, there is still space for further improvements.

Regarding the wavelet method, the use of adaptive instead of fixed wavelets may produce better results. Improvement may also come by using multidimensional alpha-stable models instead of the univariate used here. Such multidimensional models will be capable of modeling inter-level and inter-subband correlations between wavelet coefficients.

When it comes to the CNN method, implementing and testing more architectures on a larger dataset will further assess this method's value. For example, is definitely worth trying out multimodal learning using chemical analysis data alongside the images used herein.

Bibliography

[1] P. C. Molan. *Authenticity of honey.* Springer, Boston, MA, 1996. ISBN: 978-1-4612-8426-0.

[2] A. Grossmann, R. Kronland-Martinet, and J. Morlet. "Reading and understanding continuous wavelet transforms". In: *In Wavelets: Time-Frequency Methods and Phase Space* 1 (Jan. 1989), p. 2. DOI: 10.1007/978-3-642-75988-8_1.

[3] C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[4] S. G. Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 11.7 (1989), pp. 674–693. DOI: 10.1109/34.192463.

[5] R. B. Ash. *Basic probability theory*. Mineola, N.Y.: Dover Publications, 2008, 2008. ISBN: 9780486466286.

[6] G. Tzagkarakis and P. Tsakalides. "A statistical approach to texture image retrieval via alpha-stable modeling of wavelet decompositions". In: *5th Intl. Workshop on Image Analysis for Multimedia Interactive Services*. 1997, pp. 21–23.

[7] A. Geron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 1st. O'Reilly Media, Inc., 2017. ISBN: 1491962291.

[8] F. Chollet. *Deep Learning with Python*. Manning Publications Company, 2017. ISBN: 9781617294433. URL: https://books.google.de/books?id=Yo3CAQAACAAJ.

[9] A. Galanis, P. Vardakas, M. Reczko, et al. "Bee foraging preferences, microbiota and pathogens revealed by direct shotgun metagenomics of honey". In: *bioRxiv* (2021). DOI: 10.1101/2021.06.09.447678. eprint: https://www.biorxiv.org/content/early/2021/06/10/2021.06.09.447678.full.pdf. URL: https://www.biorxiv.org/content/early/2021/06/10/2021.06.09.447678.