

Davis Guntars Klavins

Meal type classification using a neural network model trained on chewing audio data

Master's thesis in Electronics Systems Design and Innovation

Supervisor: Dag Roar Hjelme

Co-supervisor: Salman Ijaz Siddiqui

June 2022

Davis Guntars Klavins

Meal type classification using a neural network model trained on chewing audio data

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Dag Roar Hjelme
Co-supervisor: Salman Ijaz Siddiqui
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Master's thesis
TFE4580 - Electronic Systems Design and Innovation

Meal type classification using a neural network model trained on chewing audio data

Davis Guntars Klavins

Supervisor:
Dag Roar Hjelme

Co-Supervisor:
Salman Ijaz Siddiqui

Trondheim, June 13th, 2022



NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology, Mathematics and Electrical Engineering
DEPARTMENT OF ELECTRONIC SYSTEMS

Acknowledgement

This study and thesis was made possible by the involvement and efforts of many special people to whom I'd like to dedicate this section to.

The main credits for achievements in this study go out to my co-supervisor Salman Ijaz Siddiqui and the talented people at the Artificial Pancreas Trondheim, who provided the initial spark for the project, and the momentum during times where the progress was on a path to a halt.

I'd like to thank my friends and colleagues A. Isifan and E.L Gundersen for the insightful academic discussions as well as the camaraderie and great times during our collective 5 years at the NTNU.

And last but certainly not the least, my gratitude and love goes out to my family who put me in the position of being able to attend university in the first place, as well as the immense support and encouragement to keep going. My academic journey, let alone contribution to this study, would not be possible without their help.

Contents

Preface	iii
List of Figures	ix
List of Tables	x
Nomenclature	xi
Abstract	xiii
1 Introduction	1
1.1 About the study	1
1.1.1 Goals	1
1.2 Literature review	2
1.2.1 Physiological metrics	2
1.2.2 Direct motion sensors	3
1.2.3 Food classification	4
1.3 Contribution of the study	4
2 Theory	6
2.1 Sound characteristics of food	6
2.2 Pre-Processing	6
2.2.1 Decimation	6
2.2.2 Fourier transform	7
2.2.3 Power Spectral Density	7
2.2.4 Spectrogram	7
2.3 Neural networks	9
2.3.1 Brief history and basic principles	9
2.3.2 Neural network layers	9
2.3.3 Activation functions	10
2.3.4 Loss function	12
2.3.5 Training procedure	13
2.3.6 Optimizers	14
2.3.7 Training parameters and overfitting	15
2.3.8 Neural network datasets	16
2.3.9 Deep Neural Networks	16
2.3.10 Convolutional Neural Networks	17
2.3.11 Evaluation metrics	19
2.3.12 Feature selection	20

3	Data acquisition	22
3.1	Data considerations	22
3.2	Equipment and setup	22
3.2.1	Recording equipment	22
3.2.2	Meals	24
3.3	Dataset collection protocol	24
3.4	Resulting dataset	25
4	Method description and implementation	26
4.1	Tools	26
4.2	Data collection	26
4.3	Pre-processing of the raw audio recordings	26
4.4	Dataset creation and architecture	27
4.4.1	PSD feature dataset	27
4.4.2	Mel spectrogram dataset	29
4.5	PSD Meal type classifier	33
4.5.1	Network model	34
4.5.2	Dataset labeling	34
4.5.3	Feature selection	35
4.5.4	Training, validation and testing split	35
4.5.5	Training the neural network	36
4.5.6	Testing the neural network	36
4.6	Chew detector (CD)	36
4.6.1	Network model	37
4.6.2	Dataset initialization and pre-processing	38
4.6.3	Training, validation and testing split	39
4.6.4	Training the neural network	40
4.6.5	Testing the Neural Network	41
4.7	Mel meal type classifier (MMT)	42
4.7.1	Network model	43
4.7.2	Dataset pre-processing	43
4.7.3	Training, validation and testing split	43
4.7.4	Training the Neural Network	44
4.7.5	Testing the neural network	45
4.8	3-class meal type classifier (3C)	46
4.8.1	Network model	46
4.8.2	Dataset pre-processing	46
4.8.3	Training, validation and testing split	47
4.8.4	Training the Neural Network	47
4.8.5	Testing the neural network	48
4.9	Testing procedures	48
4.9.1	System and Dataset test - PSD meal type classifier	49
4.9.2	System test - Chew detector (CD)	49

4.9.3	System test - Mel meal type classifier (MMT)	49
4.9.4	System test - 3-class meal type classifier (3C)	50
4.9.5	Dataset test - Mel spectrogram features	50
5	Results and observations	52
5.1	Results - PSD Meal type classifier test	52
5.2	Results - Chew detector system test	53
5.3	Results - Mel meal type classifier system test	55
5.4	Results - 3-class meal type classifier system test	58
5.5	Training curves and testing split evaluation	59
5.6	Results - Mel feature dataset test	61
6	Discussion	63
6.1	The PSD meal type classifier	63
6.1.1	Why PSD features don't work well for meal classification?	63
6.2	Chew detector	64
6.2.1	Why is precision low?	65
6.2.2	Improving the chewing detector	66
6.3	Mel meal type detector	66
6.3.1	Why does the MMT perform slightly better for oats recordings?	67
6.3.2	Possible improvements for meal type classification	68
6.4	3-Class meal type classifier	69
6.4.1	Comparing 3-class classifier to the CD + MMT combination	69
6.5	Meeting the goals for the study	70
6.6	Datasets, data acquisition and labeling	70
6.6.1	Mel spectrogram dataset	70
6.6.2	Data acquisition and it's issues	72
6.6.3	Ideas for dataset improvement	73
6.6.4	Choice of recording labeling	74
7	Conclusion	75
8	Future work	76
9	Bibliography	77
	Appendix A Dataset metadata	80
	Appendix B Source code & Cited studies	80

List of Figures

1	Concise illustration of the working principles of a spectrogram (a) and the resulting spectrogram (b) [Roberts]	8
2	Visual illustration of conversion to mel scale [Roberts]	8
3	Perceptron [Guesmi u. a., 2018]	9
4	Diagram of a fully-connected neural network [Scanlon]	10
5	Softmax function [Basta]	11
6	ReLU function [Sultan]	11
7	Illustration of underfitted, overfitted and properly trained model [Hoffman] (a), Illustration of using loss curves during training to avoid sub-optimal network performance (b)	15
8	Convolutional neural network architecture [Saha]	17
9	Convolution layer [Lendave]	18
10	Feature levels in a CNN [Nvidia]	18
11	Max-pooling layer [PapersWithCode]	19
12	Example of confusion matrix for binary class problem	19
13	Data collection equipment: Laptop, Roland Octa-capture DAC and custom built housing for 4 cup style microphones.	22
14	Oats meal [BareBra] (a), Salad Meal [Matinfo] (b)	24
15	Protocol for data collection	24
16	PSD feature matrix of one recording. Entire recording segmented in to M segments, and N PSD based features extracted from each segment.	27
17	Excerpt from Audacity of a labeled region in the recording	29
18	Single sample in the training dataset, created by applying a Mel spectrogram to a segment of the recording selected by the manual chew labels	30
19	Segment length statistics for the different meal type labels.	31
20	Single recording in the testing dataset, which consists of periodically selected segments from the recording with Mel spectrogram applied to each. Each segment corresponds to the same type of Mel spectrogram segment as in training dataset, shown in Figure 18.	32
21	Neural network model of the PSD meal type classifier)	34
22	Overview of the dataset for the PSD meal type classification system	35
23	Neural network architecture for the chew detection system. Layer dimensions correspond to input sample dimension of 128x33, thus for other inputs tested the dimensions will vary. (k=kernel size, s=stride)	37
24	Overview of the dataset structure for the chew detection system .	40

25	Neural network architecture for the Mel meal type classification system. Dimension numbers correspond to input sample dimension of 128x33, thus for other inputs tested the dimensions will vary. (k=kernel size, s=stride)	43
26	Overview of the dataset structure for the Mel meal type classification system	44
27	Neural network architecture for the 3-class meal type classification system. Dimension numbers correspond to input sample dimension of 128x33, thus for other inputs tested the dimensions will vary. (k=kernel size, s=stride)	46
28	Overview of the dataset structure for the 3-class meal type classification system	47
29	Typical prediction result for the PSD meal type classification system when using datasets with segment length of 0.45, 10, 20, 30 and 60 seconds. Predicted labels in blue, True labels in orange.	52
30	Typical prediction result for the PSD meal type classification system when using dataset with segment length of 1 second. Predicted labels in blue, True labels in orange.	52
31	Chew detector predicted and true labels for two testing recordings: oats meal (top), salad meal (bottom). The green labels indicate the predictions, which were either class “silence” or “chew”. The red labels are the true labels, and are provided for reference.	54
32	Predicted and true labels for two recordings: oats meal (top), salad meal (bottom). Predictions are made using MMT system with CD predicted chew labels. The blue labels indicate the predictions, which were either class “silence”, “salad” or “oats”. The true labels in red show two classes “silence” or “chew”. For the true labels all of the chews are of the meal type indicated by the title of the recording.	55
33	3C system predicted and true labels for two recordings: oats meal (top), salad meal (bottom). The yellow labels indicate the predictions, which were either class “silence”, “salad” or “oats”. The true labels in red show two classes “silence” or “chew”. For the true labels all of the chews are of the meal type indicated by the title of the recording.	58
34	Confusion matrix for the testing split of the best achieved CD (a), MMT (b) and 3C (c) system model	59
35	Training loss and accuracy curves of the best achieved CD (a), MMT (b) and 3C (c) system model	60
36	Comparison between PSD spectrogram, Mel spectrogram and Audacity reference of a single chew occurrence. All 3 windows capture the same length and region of the audio recording.	64

37	Section of the recording and true labels in Audacity (top), and a plot of CD predicted (green) labels and true (red) labels of the corresponding recording section (bottom)	65
38	Excerpt of non-pre-processed salad meal recording spectrogram in Audacity (a). Spectral centroid of all manual chew label segments (also on non-pre-processed recordings) (b)	72
39	Chewing spectrogram of a cookie (left) and a carrot (right) from the [Bi u. a., 2016]	73

List of Tables

1	Summary of microphone placements for recording sessions. Grayed out channels were not used for this study. Only data from channel 1 was used for this study. [Klavins, 2022]	23
2	Description of the PSD based, power features that were used to create the feature matrix in Figure 16	28
3	List of PSD feature dataset configurations used in this study	29
4	Training datasets used for Mel spectrogram based systems in this study	31
5	Testing datasets used in this study. N corresponds to the number of segments in each recording	33
6	Results from the chew detector system test 4.9.2. Recordings marked with * have true labels available. For this test the network is trained and tested with the 350ms segment length dataset (128x33).	53
7	Results from the Mel meal type classifier system test 4.9.3. Recordings marked with * have true labels available. For this test the network is trained on the 350ms segment length dataset (128x33).	55
8	Results from the 3-class meal type classifier system test 4.9.4. Recordings marked with * have true labels available. For this test the network is trained on the 350ms segment length dataset (128x33).	58
9	Results table for the Mel spectrogram dataset test 4.9.5	61
10	Information about the recordings which were collected with the data acquisition methods described in section 3. Meal type hard=salad and soft=oats.	80

Nomenclature

3C	3 Class meal type classifier
ANN	Artificial neural network
APT	Artificial Pancreas Trondheim
BGL	Blood glucose level
CD	Chewing detector
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DT	Decision tree
EMG	Electromyography
FFT	Fast fourier transform
FN	False negative
FP	False positive
GD	Gradient descent
HMM	Hidden Markov model
LDA	Linear discriminant analysis
MMT	Mel meal type classifier
SFTF	Short time fourier transform
SGD	Stochastic gradient descent
SVM	Support vector machine
TN	True negative
TP	True positive

Abstract

Diabetes is one of the most prevalent conditions in the world. Millions of patients depend on invasive and inconvenient treatments to maintain daily function. Research in alternative treatments has led to conception of the term *artificial pancreas*, a fully autonomous system which replaces the faulty insulin production capabilities of the patients pancreas. One among the numerous considerations for realizing such system, is autonomous dietary monitoring. Appropriate injection of insulin requires early information about meal onset, and accurate dosage is dependent on information about the nutritional contents of the food. This study is one of the numerous ongoing research projects by the Artificial Pancreas Trondheim research group, and covers the investigation into approaches for meal type classification.

The meal type classification systems developed in this study are based on neural network models trained on data extracted from audio recordings. A dataset consisting of 20 audio recordings, capturing the chewing sounds during consumption of salad and oat meals, is used to extract Power spectrum density (PSD) and Mel spectrogram features. The initial approach uses a fully connected neural network model, trained on the PSD features to classify the two meal types. Although this approach manages to correctly identify the meal region in the recordings, it fails to reliably identify the meal type.

The second approach implements 3 systems using a convolutional neural network, trained on Mel spectrogram chewing segments. Two of the systems, the chew detector and Mel meal type classifier, are combined to provide meal type classification. This approach correctly classifies meal types for all 20 testing recordings and achieves average prediction ratio of 90.5%. The third system, which combines the previous two, performs similarly, achieving 90.3% prediction ratio. Additional testing of the dataset configurations indicates that the optimal segment length for the dataset is around 350 to 450ms. Although the testing conditions were controlled and recordings confined to only two meal types, this study proves the feasibility of meal type classification and provides a working implementation with a very good performance.

1 Introduction

1.1 About the study

Investigation into dietary monitoring is part of continued effort by the Artificial Pancreas Trondheim (APT) research group to develop a fully autonomous device to replace the faulty insulin regulation capabilities of a diabetes patients pancreas. Dietary monitoring plays an integral role in realizing the artificial pancreas. A system which can provide information about meal onset and meal contents is crucial for accurate calculation of insulin dosage.

The initial pilot study by K.Kölle [Kölle, 2019] investigated the feasibility of using bowel sounds for early meal detection. Compared to conventional methods of blood glucose level based meal detection with detection delay of up to 40 minutes, her support vector machine (SVM) system based on power spectrum density (PSD) features, managed to reduce detection time to 10 minutes. Continuing her work, Viljar. B [Bliksvær, 2021] attempted to improve the previous system by augmenting the bowel sound dataset with EKG measurement features. In the masters project study by the author [Klavins, 2022], which was built on the results of the previous two studies, an SVM model was trained using combination of bowel and chewing sound PSD features. It was found that for the SVM meal detection system, the chewing features yielded significantly better results.

For this study a slightly different dietary monitoring avenue is investigated. This master's thesis will cover the development and implementation of a meal type classification system. The main purpose of such system would be to provide nutritional information which could be used as a guideline for calculation of insulin dosage and other parameters for the artificial pancreas system. Like in the previous studies, this study aims for a non-invasive, audio based machine learning system. More specifically, the meal type classification system will be implemented using convolutional and fully-connected neural network models with a dataset created from audio recordings that capture chewing sounds during the meal.

1.1.1 Goals

The primary goal for this study is to develop a meal type classification system which reliably provides correct information about the meal type in a recording. Reliability requirements for a system depend on its application. For regular dietary monitoring, lower reliability is permitted, however as an integral component of a medical device, the reliability must be held to much higher standards. For application in the artificial pancreas, our system would need to deliver as close to 100% classification accuracy and reliability as possible.

The testing data in this study consists of 20 audio recordings and two different meal types. In one sense the accuracy and reliability can be measured by the number of recordings whose meal type is correctly identified. If testing dataset consists of 20 recordings, we'd expect all 20 to have a correctly classified meal type.

In addition to that, our system will not evaluate the entire meal as a whole. Instead the recordings will be periodically segmented, and each segment will be evaluated separately, and the aggregate of all the separate predictions will be used to assign a predicted meal type to the entire recording. This necessarily introduces the need for a threshold of acceptable prediction confidence. The metric for decision in this study is prediction ratio, which gives the prediction confidence for a given meal type in a recording. It can be argued, that as long as the aggregate prediction ratio score is $> 50\%$ for the correct class, the meal will be classified correctly, however a medical system with a prediction confidence of 50.1% doesn't sound very reliable.

For this study the main priority will be achieving 20/20 correct meal type classification. The prediction confidence threshold however, is a subject of debate. For this study, no specific prediction confidence goal will be set. Instead the aim is to achieve as high prediction ratio as possible. For a practical artificial pancreas system the acceptable threshold for prediction confidence would need to be evaluated by physiology and diabetes experts.

1.2 Literature review

Chewing and meal detection systems aren't a novelty. For over a decade a considerable amount of research has been done investigating methods for detecting dietary patterns and food intake. In general these methods can be condensed to two types: one, directly sensing food consumption mechanisms, and two, exploiting physiological changes as the result of food consumption.

1.2.1 Physiological metrics

The most notable dietary monitoring approaches based on physiological changes involve ECG analysis and bowel sound detection. ECG method involves analyzing heart rate patterns and trends before, during and after food consumption. More specifically, ECG can help to identify blood glucose levels (BGL) by exploiting the connection between BGL and heart rate [Cordeiro u. a., 2021], [Porumb u. a., 2020]. This type of research is favoured due to the availability of ECG in commercial devices and the potential to provide diabetic patients with a non-invasive alternative to existing BGL monitoring methods. Further on, as intestinal movements and bowel activity are linked to food consumption, the detection of abdominal sounds has also been investigated [Kölle, 2019], [Wang u. a., 2022a].

1.2.2 Direct motion sensors

Physically, food consumption consists of mastication which processes food into more easily digestible chunks by chewing, followed by swallowing. There are multitude of suggested approaches which exploit these two consumption mechanisms. For chewing specifically, the survey paper ,[Selamat u. Ali, 2020], presents a broad compilation of the chewing based dietary monitoring methods from the past decade. To summarize, most modern dietary monitoring methods are based on sensors (as opposed to manual logging). The single-sensing methods, such as acoustic, piezoelectric sensors, electromyography (EMG) and accelerometers, use a single sensor to capture a specific defining signal. There are also multi-sensing approaches which aggregate multiple sensors. The following few paragraphs will briefly touch on these different methods and their performance.

Acoustic sensors Ranging from in-ear monitors and hearing aids to neck-band throat microphones, they are used for detection of chewing sounds either through air or bone conduction. Main challenge with this approach is balancing high-quality signal captures and user comfort when wearing the sensors, as well as mitigating impact of background noise. Accuracy-wise performance of acoustically based detection systems is in the mid 80% to mid 90% range.

Piezoelectric sensors Piezoelectric sensors measure deformation in the piezoelectric material. This mechanism can be used to directly detect movement of the jaw or chewing muscles. These sensors reside under patients ear with the help of a medical tape, or they can be built into wearables, such glasses, to measure temporalis (chewing) muscle. Piezoelectric-based detection systems have been used mainly for chew and chewing pattern detection, however they have also been used in food-type (liquid/solid) detection as well as chewing behaviour analysis. This sensor provides the advantage of being simple, non-obtrusive and power-efficient, however other daily activities can affect the measurements and identification of liquid consumption is a challenge. For the mentioned tasks the accuracy of most piezo-electric systems is mid 90%.

EMG sensors Electromyogram sensors work by detecting the electrical signals produced by muscle movement. Like the piezoelectric sensors, these have been employed in wearable devices for a direct contact with the skin above chewing muscles (temporalis specifically). For one study, the combination of vibration sensor with the wearable EMG device provided a robust detection with low artifacts for non-chewing activities. They were also able to classify food types based on their texture. This approach provides simple, unobtrusive design and few artifacts. Main challenge with EMG is providing electrodes with

a direct and reliable contact with the skin, while remaining unobtrusive. They are also limited to detecting food types based on texture. Although there are fewer studies on EMG sensors, the examined systems achieve accuracy of mid 90% (with few exceptions).

Multi-sensor systems Multi-sensor systems aggregate multiple sensors in order to detect chewing or a combination of additional factors, such as swallowing, bite, and motion sensing. For example, Automatic Ingestion System (AIM) consists of 3 sensing modalities: piezoelectric, accelerometer and hand-to-mouth gesture sensors. Each sensor is placed in a separate suitable location on the body. The other, so called SPLENDID multi-sensor system employs a photoplethysmogram (PPG) sensor (measure blood pulse using light emitting diode and photosensor) and in-ear housing microphone. Chewing detection performance for these systems is on par with the single-sensor systems, with accuracy scores in the 80% to 90% range.

1.2.3 Food classification

As mentioned, the methodologies discussed above are a summary of the [Selamat u. Ali, 2020] survey study. Among the compiled studies, the 3 main goals were: food intake detection, chew count estimation and meal type classification.

Regarding our primary goal for this study, meal type classification, all of the examined studies in the survey performed classification on multiple (typically around 8 different) food types, some including liquids. Food characteristic (hard, soft, crispy, etc) variety varied from study to study however most studies include foods to cover, at least partially, the the spectrum of sound characteristics. Collectively, the studies investigate broad spectrum of classification models, such as Hidden Markov Model (HMM), Bayes Naive & Fisher discriminant classifier, Linear discriminant analysis (LDA), Decision tree (DT) and Artificial neural network (ANN). Food type classification accuracy for all methods ranges from 46% to high 90%, with the majority towards the higher end (around 80% on avg.)

1.3 Contribution of the study

This study was initiated with the main goal of developing a system for classifying different food types using audio recordings of meals. As noted in the introduction, meal onset detection was already familiar from previous APT research projects, so the initial approach for this study was to apply the same methods for the purpose of meal type classification. Beyond that, the path was

yet not completely mapped out, and many aspects of the implementation and testing that are presented in this thesis were adopted along the way.

By the end of the study, in addition to accomplishing the main goal of developing a meal classification system, other notable contributions to the topic were made. Primarily, two types of systems were implemented; a meal type classification system with a complimentary chewing detector, as well as a combined alternative. As such, the importance of a well working chewing detection system for this approach was discovered. The study also presents the effect of segment length choice on classification performance, for the particular system implementation that is used. It also addresses the different challenges and issues which were discovered through the process, and are likely to be relevant for future studies.

2 Theory

2.1 Sound characteristics of food

Basis for data in this study are the sound producing mechanisms of mastication. Study by C.Dacremont [Dacremont, 1995] analyzed characteristic frequencies of various foods during mastication, and showed that for particularly crispy foods frequencies of 5000Hz are reached. Crunchy foods, such as carrots, exhibit characteristic frequencies between 1250 and 2000Hz.

2.2 Pre-Processing

This subsection covers the theory behind the different data pre-processing methods used for feature extraction and dataset preparation.

2.2.1 Decimation

Decimation refers to the two-step process of filtering and downsampling a signal. Benefits of this operation include reduced file size as well as removal of redundant information. The filtering step applies an anti-aliasing filter which removes unwanted frequencies, and the downsampling step effectively reduces the sampling frequency of the signal.

Anti-aliasing filter Prior to downsampling, the signal must be treated with an anti-aliasing filter. Purpose of this step is to remove any frequency content that will exceed Nyquist limit after downsampling. If anti-alias filter isn't applied, the signal will get corrupted by frequencies above Nyquist limit, which after downsampling will be transformed to a different frequency within the Nyquist range, effectively adding false information. In order to filter correctly, the Nyquist equation is used to calculate the cut-off frequency for the filter. The cutoff frequency corresponds to half of the sampling frequency:

$$F_s = 2 \cdot F_{max} \quad (1)$$

In practice, the anti-aliasing filter is typically a low-pass filter with cut-off frequency given by F_{max} in the equation above.

Downsampling Downsampling is the second step in decimation. This process changes the working sampling frequency of the signal by picking every M 'th signal sample, resulting in a signal which has the same playback duration, but less samples. For a correct playback of the downsampled signal, the new sampling rate is used. The variable M is determined by the old and new sampling frequencies: $M = \frac{F_{s.old}}{F_{s.new}}$

2.2.2 Fourier transform

Perhaps the most important tool for signal analysis and digital signal processing is the Fourier transform. In short, the mathematical operation transforms the signal from a time domain to a frequency domain, revealing the frequency content of the signal. The amplitude of the spectrum can be visualised where the x-axis becomes a range of frequencies with y-axis giving magnitude of each frequency. The Fourier transform is originally an operation for continuous signals, however the digital version, the Discrete Fourier transform (DFT), is often implemented with the efficient Fast Fourier transform (FFT) algorithm. The DFT is given in the equation below:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-jk\frac{2\pi}{N}n} \quad k = 0, 1, 2 \dots N - 1 \quad (2)$$

2.2.3 Power Spectral Density

A useful tool for analyzing the spectrum of a signal is the Power spectral density (PSD). This operation transforms the time signal to the frequency domain and distributes the signal power relative to the frequency components of the spectrum. Equation 3 calculates the PSD by squaring the absolute value of the Fourier transform of a discrete signal $x(n)$ and dividing by the number of frequency bins.

$$PSD(k) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n)e^{-jk\frac{2\pi}{N}n} \right|^2 \quad k = 0, 1, 2 \dots N - 1 \quad (3)$$

2.2.4 Spectrogram

An often employed method for audio signal analysis is the spectrogram. Being a slight variation of the short-time fourier transform (STFT), the spectrogram essentially stacks multiple FFT windows along the time dimension showing how the frequency contents change over time. It is visualised in plot where x-axis represents time, y-axis is the frequency range and the intensity of the frequency components is mapped to a color-gradient. Usually, to improve visualisation of a wide dynamic range signal, the y-axis (frequency) is converted to logarithmic scale and the color-axis (intensity) is displayed in dB. Figure 1 presents a visual summary of the spectrogram creation process as well as the resulting spectrogram. [Roberts]

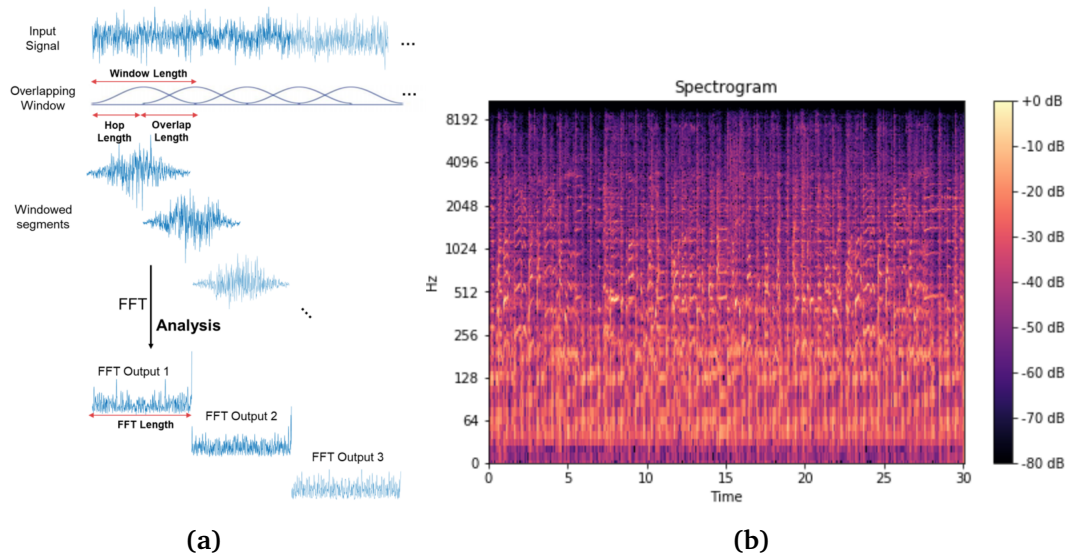


Figure 1: Concise illustration of the working principles of a spectrogram (a) and the resulting spectrogram (b) [Roberts]

Mel Spectrogram As mentioned above, the y-axis can receive a logarithmic transformation. An alternative transformation exists, which maps the frequency range so that it reflects the distances between the frequencies in a way that the human ear would perceive as being equidistant in pitch. This is perhaps illustrated best visually in Figure 2, where the line indicates the transformation between a logarithmic and Mel scale.

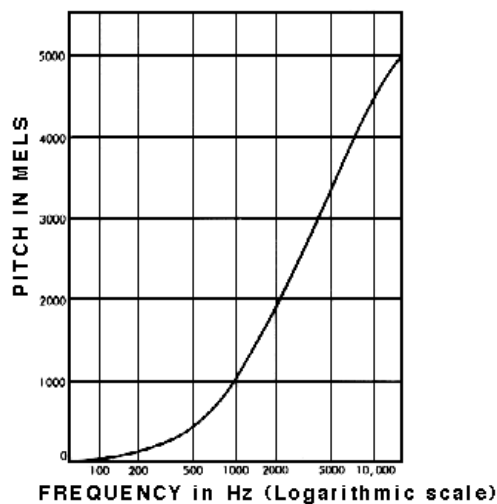


Figure 2: Visual illustration of conversion to mel scale [Roberts]

2.3 Neural networks

At the core of the meal type classification systems in this study is a neural network model. The basic concept and working principles are explained in this subsection.

2.3.1 Brief history and basic principles

The idea of computational neural network stems from biological neural networks, such as those comprising parts of the human brain, which are able to express complex dynamics by means of relatively simple mechanics that can be reduced to elementary arithmetic operations. Building upon the the early research by W.S McCulloch, W.H Pitts and F. Rosenblatt in 40s and 60s, the field of machine learning has, with the advent of modern computing, been able to fully realize the concepts proposed by these early pioneers.

At the base of modern neural networks is the Perceptron, originally proposed by Rosenblatt in the 60s. With what initially was just an algorithm, the Perceptron has gone to define the fully-connected networks with millions of nodes that we think of today.

The following operations describe the perceptron algorithm, however they also apply to the modern fully-connected neural networks. The operational unit of the perceptron can be thought of as a node defined by multiple parameters: inputs, weights, bias and an output. The node receives some number of real valued inputs. Importance of each input signal is manually regulated by multiplying it with some weight value. The weighted inputs, along with a bias value, are then summed together. The result is then evaluated by the *activation function*, which essentially performs thresholding using a defined function. Output of the activation function is the output of the node. Figure 3 illustrates the perceptron.

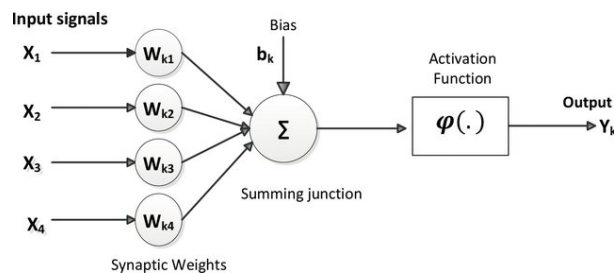


Figure 3: Perceptron [Guesmi u. a., 2018]

2.3.2 Neural network layers

The simplest kind of modern neural networks are made up of such perceptron-like nodes. In general the network consists of 3 parts: *input layer*, *hidden layer*

and the *output layer*.

The *input layer* is a column of nodes, where each node takes one input value and propagates it through the aforementioned mathematical operations, producing an output. After the input layer, the next column of nodes, aka. layer, is a *hidden layer*. Outputs of the input layer are connected to inputs of nodes in the hidden layer. An architecture where the output of each node is connected to all input nodes in the next layer is a *fully-connected network*. One can, and often does, add multiple such hidden layers, before adding the final layer, the *output layer*. An important characteristic of the output layer is that it needs to, in some way, represent the desired output options. Usually this is done by simply having as many output nodes as there are classes in the dataset. Visually, such fully-connected network, is presented in Figure 4

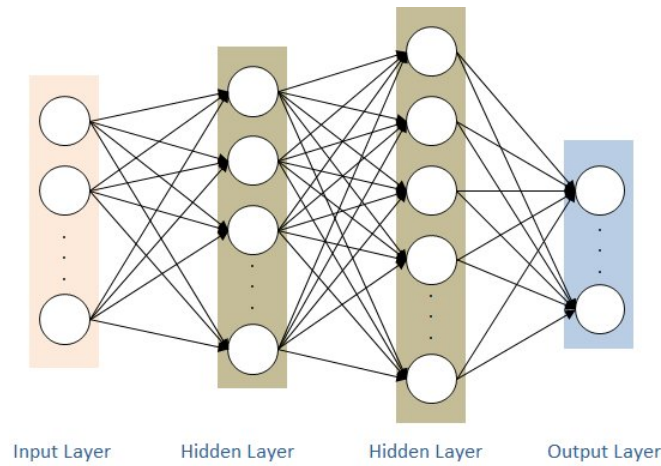


Figure 4: Diagram of a fully-connected neural network [Scanlon]

2.3.3 Activation functions

As briefly mentioned for the perceptron, the activation function is used to get outputs from network nodes. Depending on the application of the network and input data, different activation functions are suitable. The following activation functions are best suited for the NN application in this study. [Sharma]

Softmax

$$\text{softmax}(X)_i = \frac{e^{x_i}}{\sum_{i=1}^K e^{x_i}} \quad (4)$$

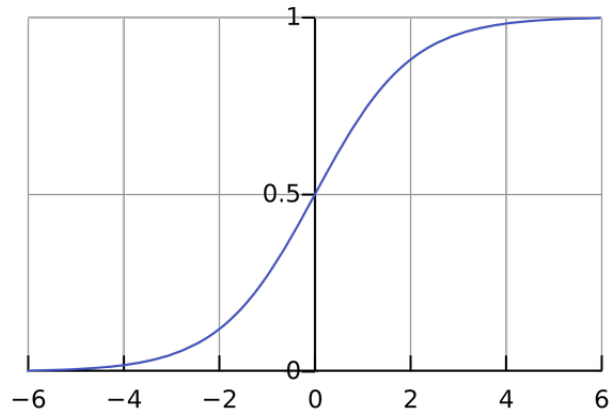


Figure 5: Softmax function [Basta]

Softmax is a function that maps a list of numbers to a list of probabilities, such that all individual probabilities sum up to 1. This can be interpreted from the equation 4, where probability of value x_i is the exponential fraction of the sum of all $i = 0, \dots, K$ node output value exponents. Softmax is well suited as the activation for the output (final) layer for a multi-class classifier, and expects the output node count to be equal to the number of classes.

ReLU

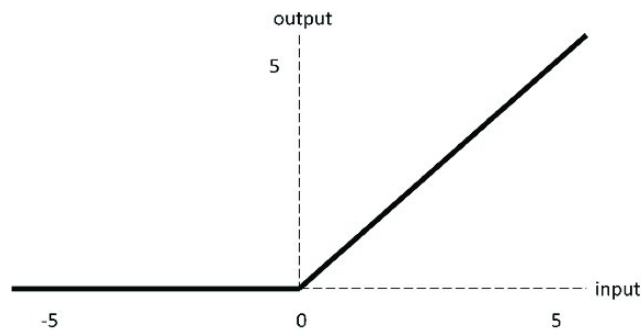


Figure 6: ReLU function [Sultan]

Rectified Linear Unit (ReLU) is currently a very popular hidden layer activation function, which is mainly due to its desirable properties. In order for the network to approximate non-linear patterns, it requires a non-linear activation function. At first glance ReLU might appear to be linear, however its behavior for negative values provides its overall non-linearity property. Due to its unbounded growth, ReLU avoids vanishing gradients and is thus a great combination with gradient-based optimization algorithms.

2.3.4 Loss function

A *loss* or *cost* function is a metric that is used by the network during the training process to assess the performance. Loss function creates some measure between the expected/desired output (true label) and the actual output (predicted label).

Considering a single input sample x_n , belonging to the class y_n (true label), the neural network will return a value, o_n (predicted label), representing the class it decides the sample x_n belongs to. The cost function measures the error between y_n and o_n . A particular cost function, which will be utilized in this study is the Cross-entropy.

Cross-entropy [Brownlee, a] Cross-entropy is a concept stemming from information theory and describes the average number of bits/nats required to represent an event coming from one distribution instead of another.

The first information theory concept, expressed by equation 5, is the information of an event. Probable events contain little information and vice versa.

$$h(x) = -\log(P(x)) = \frac{1}{\log(P(x))} \quad (5)$$

Further on, *Entropy*, is defined as the amount of bits/nats required to represent the information of an event. For an event x_i from a distribution P, the entropy is given as sum of products between information of the event $h(x_i)$ and the probability of that event $P(x_i)$ (eq. 6).

$$H(X) = -\sum_{i=1}^N P(x_i)h(x_i) = -\sum_{i=1}^N P(x_i)\log(P(x_i)) \quad (6)$$

Finally to arrive at cross-entropy we define two probability distributions. Distribution P describes the target distribution (for true labels), and O describes the predicted distribution (for predicted labels). The cross-entropy captures the difference between the target and predicted distributions, and based on this it captures the difference between predicted and true labels. Cross-entropy is given in equation 7.

$$H(P, O) = -\sum_{i=1}^N P(x_i)\log(O(x_i)) \quad (7)$$

Cross-entropy loss functions apply to both binary and multi-class classification problems. Because the implementations vary slightly to accommodate the number of output classes, they are differentiated by name:

- Binary cross-entropy

- Categorical cross-entropy

Since cross-entropy relies on probabilities, it nicely complements a softmax output activation function.

2.3.5 Training procedure

The neural network training procedure follows an iterative process, where the entire training dataset is processed for set number of cycles, called *epochs*. For each epoch the training dataset is subdivided in batches of a defined size (batch size), and each batch gets forward- and back-propagated, thus training the network.

Forward- and Back-propagation Giving the network input values and propagating them through the network is called *forward-propagation*. Outputs from the forward-propagation are then evaluated by the loss function giving the amount of error.

The important part of the network is its ability to improve its performance over multiple epochs, in other words “learn”. To cause the network to learn, the different network parameters (weights and biases) need to be adjusted. While its technically entirely possible to apply some elbow grease and adjust the millions of variables manually, a more rational approach is to automate this process. This is done through the combination of an *optimizer* and an algorithm called the *back-propagation*.

As established above, the loss function gives a measure for the error. Its main use is as a guideline for optimizing (training) the network. Thinking of the loss function in terms of a shape on a graph, it necessarily has a minimum, where the error is the smallest. Obtaining the minimum is usually done by computing the derivative (or *gradient* for multi-dimensional functions) and setting that to zero. For neural networks these computations end up very complex, making numerical solutions the only viable option. One such numerical method is the *gradient descent* (eq. 8).

$$W_t = W_{t-1} - \epsilon \Delta W_{t-1} = W_{t-1} - \epsilon \frac{dC(Y, O)}{dW} \quad (8)$$

The new weights, W_t , are determined by the previous weight, W_{t-1} , minus the gradient of the cost function with respect to the previous weights. The epsilon is defined as the *learning rate*, a value between 0 and 1, which determines how big the jump in the direction of gradient descent will be. The general act of updating weights and biases is called *optimization*, where this specific type of optimization is *gradient descent*.

The back-propagation algorithm is used to compute the gradient part of equation 8. Due to the network architecture, computation of error gradients

with respect to a given layers weights depends on error gradients of its subsequent layers. Thus the algorithm starts with error from the output layer works its way back to the input. Back-propagation algorithm defines an efficient method for computation of, as the name suggests, backwards propagating error gradient.

2.3.6 Optimizers

Gradient descent (and its variants) The optimizer that's used for this study is a derivative of gradient descent, which is why it can be beneficial to understand it beforehand. There are 3 main variations of gradient descent (GD): *Stochastic gradient descent* (SGD), *Mini-batch stochastic gradient descent* and regular GD. Recall the description of batches in subsection 2.3.5. Essentially, batch size determines the type of GD that is used. Quick summary of the three:

1. Stochastic gradient descent (SGD):
 - (a) Performs optimization based on a randomly (stochastically) selected single sample from the training dataset
 - (b) Very fast, but the optimization is very noisy.
2. Mini-batch stochastic gradient descent:
 - (a) Performs optimization based on a randomly selected part (batch) of the training dataset
 - (b) Go-to method for most neural networks, as it strikes balance between smooth optimization and speed.
3. Gradient descent:
 - (a) Performs optimization based on the entire training dataset
 - (b) Slow, due to having to compute gradients for all training samples. This is especially true for networks with large datasets. Upside is a smoother approach towards the cost function minimum.

Adam Adam is an optimizer derived from stochastic gradient descent and was developed to incorporate benefits of AdaGrad and RMSProp optimization algorithms. Without getting too technical, here is how it improves upon the gradient descent.

Adam implements individual learning rates for each parameter (weight), where these rates are calculated using first (mean) and second (uncentered variance) moments of recent gradient magnitudes. These properties help dealing with sparse gradients and make Adam a great option for computer vision applications. [Brownlee, b]

2.3.7 Training parameters and overfitting

Training the network is usually a very simple process, initialized with a handful of function calls in Python. All the difficult operations are neatly tucked behind the api calls of machine learning libraries such as Tensorflow. The user input is however required for picking the training parameters: learning rate of optimizer, number of epochs and batch size. Epochs in particular set the amount of training that the network gets. With too little training, the performance will be sub-optimal, however too much training can result in overfitting.

Model overfitting This machine learning phenomenon refers to a situation where excessive optimization causes the model to learn the training inputs too well, to the point where the performance on similar but unseen data suffers. This is best visualized by the Figure 7a.

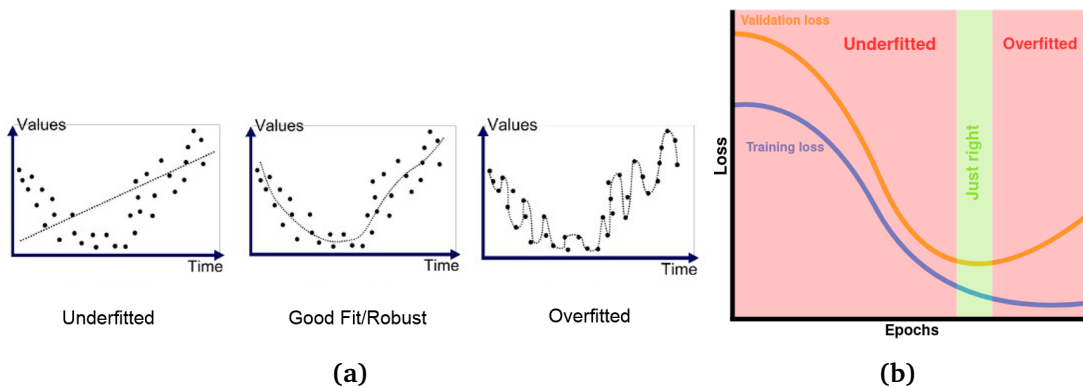


Figure 7: Illustration of underfitted, overfitted and properly trained model [Hoffman]
 (a), Illustration of using loss curves during training to avoid sub-optimal network performance (b)

There are multiple ways of avoiding overfitting of a model. The simplest and the most general method is by tracking the loss curves during the training. A good training method will always involve the use of separate dataset samples for training, validation and testing. During training the network is optimized using training data. By using separate validation data after each epoch to test the network, the validation loss can be used as a guide to know when the training has reached an optimal point. Initially, as the model improves, the loss for both training and validation data will decrease. Once the validation loss plateaus, the model is optimally trained and training should be stopped. After this point the loss will likely start to increase. A typical training loss curve can be seen in Figure 7b.

Regularization can also be used to reduce overfitting. By using a so-called *dropout* layer, the model will disable random nodes for each training epoch.

This forces the network to generalize the connections and patterns, leading to better performance on unseen data.

Dropout layers and loss curves are the main tools that were used to achieve optimized models in this study.

2.3.8 Neural network datasets

When discussing overfitting in the previous Section 2.3.7, 3 types of datasets are mentioned: training, validation and testing. This section will briefly explain the purpose of the 3 datasets and how they are obtained.

In reality neural networks are trained and tested with data originating from the same dataset, however the individual samples cannot be the same for training and testing. This separation is achieved by splitting the entire dataset into different parts. For the simplest case this means a training and testing split, however it is common to also include a validation split.

Training split is used mainly for training. This will usually be the bulk of the entire dataset to maximize learning. It is allowed to test the model using training data however these tests aren't considered valid since the exact samples have already been used to optimize the network during training.

For valid testing a separate testing split is used. This is a part of the dataset that has been excluded from training and is thus unseen by the network. However since the testing samples are from the same dataset, they represent the same type of data. When used for testing they provide a realistic indication for real-world performance.

Additionally, a part of the dataset may be used as a validation split. Validation samples are used during training to validate the intermediate performance after each training epoch. The validation data are never used to train the model, but act as a mini testing data. This is also the data that's used to calculate the validation loss and look out for overfitting during training.

2.3.9 Deep Neural Networks

A Deep Neural Network (DNN) is an umbrella term for neural network architectures with more than one hidden layer. DNNs typically describe feed-forward network architectures, for which the inputs propagate through the network in only one direction (towards the final output). Just for reference, Recurrent neural networks (RNN) aren't a subset of DNNs, due to their ability to have nodes that have access to previous inputs (memory). A Convolutional neural network however, is a subset of DNN.

2.3.10 Convolutional Neural Networks

Convolutional neural networks (CNN) are a variation of DNNs, where the first few layers typically contain *convolution* and *pooling* filters, which extract specific features from the input data. These are followed by a number of fully-connected (FC) layers to provide the final output. Lets examine the architecture in more detail. Figure 8 will provide a reference.

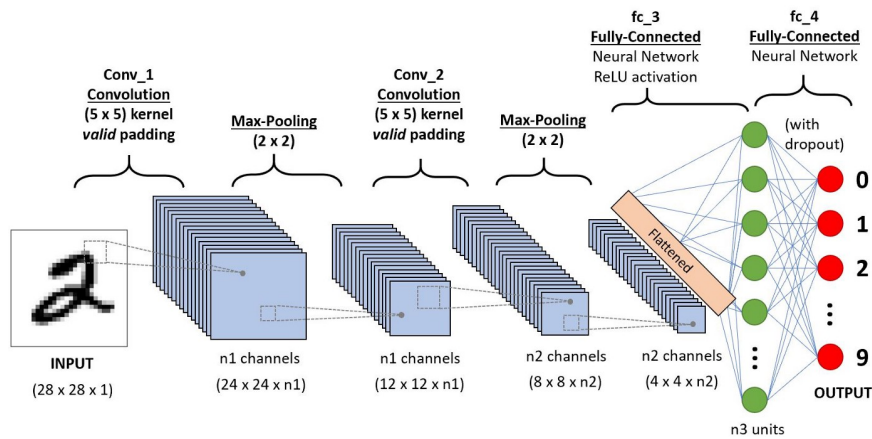


Figure 8: Convolutional neural network architecture [Saha]

CNNs solve two main issues that regular FC-networks face when working with images. First issue is the size of inputs. In order to provide image as an input to a FC-network, the image needs to be flattened, resulting in the number of inputs corresponding to the number of pixels in the image. For modern image resolutions, network node and parameter amounts exceed reasonable processing capacity. The second issue is FC-networks ability to recognize local pixel correlations and extract image features and patterns. Both issues are solved by the combination of correlation and pooling layers, which reduce the image dimensionality as well as extract characteristic image features.

Convolution layer The input layer of a CNN is typically a convolution layer. It takes a whole image as the input and applies a spatial convolution filter over the whole image. The filter is a kernel with a predefined size (eg. 3x3 window) which moves along the pixels with a specified stride, essentially performing dot-product between the kernel and the pixels it selects. For each step, the resulting dot-product is a single value corresponding to single pixel in the output. Each convolution layer is made up of multiple such filters with each filter, by the end of the training session, optimized to extract some specific feature. Each filter in the layer has its own output. The convolution operation is shown in Figure 9

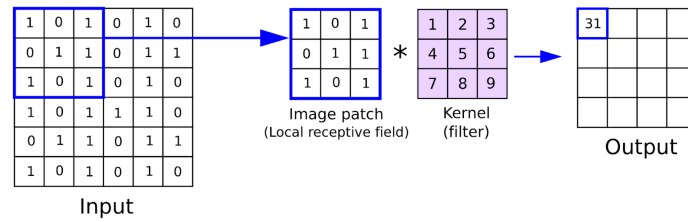


Figure 9: Convolution layer [Lendave]

Using the filters, the convolution operation extracts specific feature elements from the input training images. The first features that are extracted are low-level, meaning simple edges, lines and gradients. Subsequent convolution layers construct progressively higher level features using the lower level features, as illustratively depicted in Figure 10

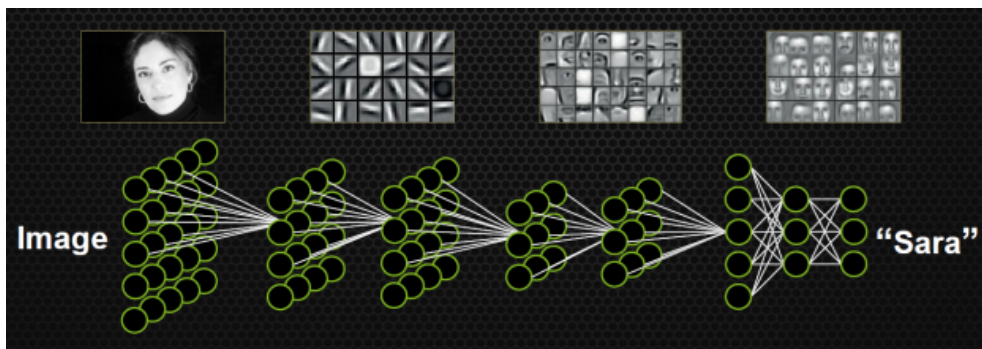


Figure 10: Feature levels in a CNN [Nvidia]

Pooling layer One or multiple convolutional layers are typically followed by a pooling layer. There are two main types of pooling: *max-pooling* and *average-pooling*. Analogous to the convolution layer, the pooling layer performs spatial image filtering using a kernel with specific stride. The difference is that the operation performed on the selected pixels is either max-value (max-pooling) or averaging (average-pooling). Figure 11 shows the max-pooling operation.

In general, pooling layers are used for dimensionality reduction which in turn reduces computational cost of the subsequent layers. Additionally, the pooling layer can help the network to learn the features in a spatially and rotationally invariant way. Max-pooling specifically has the added benefit of de-noising the features, which is why in most cases its considered better than average-pooling.

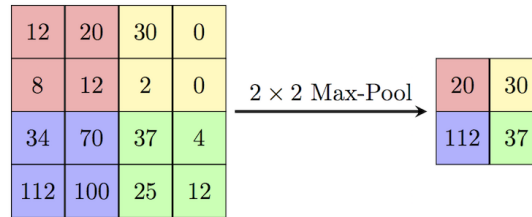


Figure 11: Max-pooling layer [PapersWithCode]

Fully-connected layer The extracted features from the convolution and pooling layers are flattened to 1D vector and used as input for fully-connected layers. The final FC layer provides the network output and consists of the same number of nodes as classes.

2.3.11 Evaluation metrics

The following metrics are the main methods of neural network evaluation used in this study.

Confusion matrix One of the most basic and intuitive ways of summarizing predictions of an NN model is to use a so called confusion matrix. In its simplest form for a binary classifier, it's a 2×2 matrix, with each cell containing quantity of one of the 4 possible classification outcomes: true positive (TP), true negative (TN) and false positive (FP), false negative (FN). The desired and correct outcomes are true positive/negative predictions. Example of a confusion matrix is shown in Figure 12.

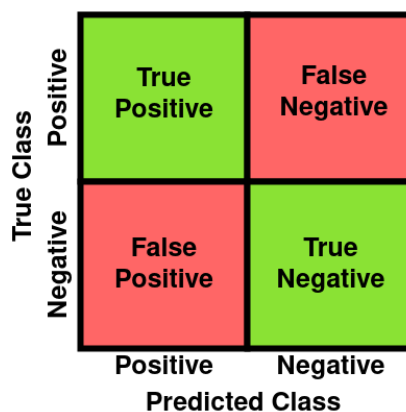


Figure 12: Example of confusion matrix for binary class problem

Accuracy, Precision and Recall The confusion matrix can be interpreted as is, however the values can also be used to calculate more intuitive statistics.

- **Accuracy:** Value between 0 and 1, representing how many of the total predictions are correct.

$$\frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Number of all correct predictions}}{\text{Total number of predictions}} \quad (9)$$

- **Precision:** Value between 0 and 1, representing how many of the predicted positives are actual positives.

$$\frac{TP}{TP + FP} = \frac{\text{Number of positive predictions which are actually positive}}{\text{Number of positive predictions}} \quad (10)$$

- **Recall:** Value between 0 and 1, representing how many of all actual positive class samples have been detected.

$$\frac{TP}{TP + FN} = \frac{\text{Number of positive predictions which are actually positive}}{\text{Total number of actual positive class samples}} \quad (11)$$

Prediction ratio is a decision and evaluation metric that gives the ratio/percentage of how many of the total predictions belong to one particular class. This metric doesn't require true labels to compute and thus can be used to make majority-based classification decisions in this study. Following equation expresses the ratio:

$$\text{Prediction ratio (\%)} = \frac{\text{Class 1 predictions}}{\text{Class 1 predictions} + \text{Class 2 predictions}} \cdot 100 \quad (12)$$

This metric was conceived by the author during the study as an intuitive way of evaluating predictions, however same metric may already exist academically under some other name.

2.3.12 Feature selection

Feature selection is a method used to eliminate less-relevant information from the dataset. The samples, which comprise the dataset, are made up of certain metrics which contain some information about the data, called features. Before training the network, it may be beneficial to remove the features which are less conducive to improving the neural networks performance.

Feature selection uses some metric to determine the relevance of features. For the case of supervised learning, where each sample has a pre-determined class label, this metric typically evaluates some correlation between the sample and the label.

SelectKBest using Chi-squared metric [SkLearn] For this study the feature selection is implemented using Python's SkLearn library, which provides the SelectKBest feature selection method. This method picks K best features based on the Chi-squared metric.

3 Data acquisition

The dataset collection was performed as a part of the thesis by the author and his colleague. What follows is a description of the contents of the dataset and the methods of its conception.

3.1 Data considerations

Prior to creating a data collection protocol, the data content requirements had to be made clear. Although the audio recordings were created primarily for this study, some consideration went into creating a protocol that would also suit parallel APT research. These aspects of the recording protocol were unnecessary for the purpose of this study, however they did not interfere with the quality of data that was required here. With that said, the following data content requirements were necessary for investigating specifically the objective of this study, the meal type classification.

- The dataset consists of at least two meal types with distinct sound characteristics.
- For simplicity, each recording is confined to only one meal type
- Also for simplicity, all environmental influence should be kept to minimum.

3.2 Equipment and setup

3.2.1 Recording equipment

The audio recording setup consists of 4 Knowle's [SPM0687LR5H-1] microphones, connected to a laptop through a Roland Octa-capture sound card. Recording software was Reaper audio workstation. Each microphone has a custom 3D printed cup-like housing which helps to attach the unit to the test subject with a double-sided tape. The hardware setup is illustrated in Figure 13

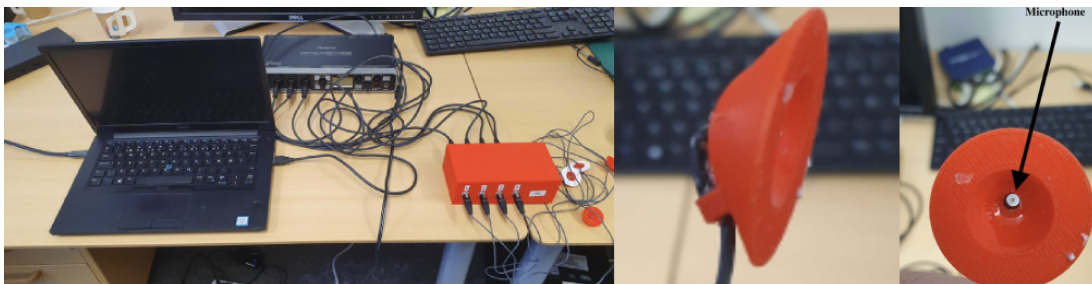


Figure 13: Data collection equipment: Laptop, Roland Octa-capture DAC and custom built housing for 4 cup style microphones.

Each microphone has its designated channel for capturing one of the 4 areas of interest, as shown in Table 1. Thus, each recording session produces 4 audio files.


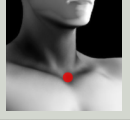


Mic/channel	Target sound	Location	
1	Chewing sounds	Approx. 3cm under the right ear	
2	Swallowing sounds	Above the center of the collar bone	
3	Right bowel sounds	Right lower abdomen	
4	Left bowel sounds	Left lower abdomen	

Table 1: Summary of microphone placements for recording sessions. Grayed out channels were not used for this study. Only data from channel 1 was used for this study. [Klavins, 2022]

Note: Since chewing sounds are the primary focus, only recordings from channel 1 were used in this study. Recordings from the other channels are used in different APT research.

The equipment that's used and the microphone locations are identical to the ones used in the meal onset detection study by the author [Klavins, 2022].

3.2.2 Meals



(a)



(b)

Figure 14: Oats meal [BareBra] (a), Salad Meal [Matinfo] (b)

In order to control for portion size, the meals were chosen in a pre-packaged, single-portion format. The oats meal consists of quinoa, oats and date pieces in a porridge-like mixture with hot water. The salad meal consists of iceberg lettuce, red cabbage, pasta and salad dressing. Note that the chicken in the salad was not consumed.

These meals were chosen based on their sound characteristics. While no objective analysis influenced the decision, it was determined based on personal experience, that the oats and salad were sonically distinct enough.

3.3 Dataset collection protocol

As mentioned earlier, the protocol was designed to suit not only this meal type classification study, but also other APT studies. This mainly entails extended silence regions, beyond what is conceptually required for this study. The final protocol that was settled on is summarized in Figure 15

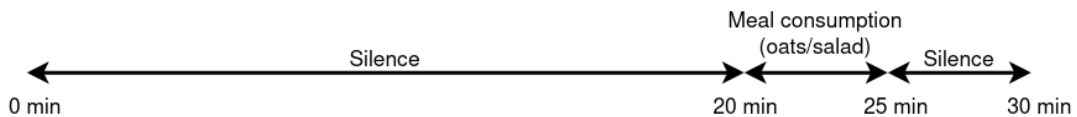


Figure 15: Protocol for data collection

The procedure for a single recording session is as follows:

1. Prior to recording the subject takes a comfortable sitting position. One of the two meals is chosen and prepared for consumption (remove packaging, add hot water, etc) to make it easy for the subject and to avoid unnecessary noise.

2. Once the recording starts the subject is instructed to remain in a sitting position and make as little noise as possible.
3. At the 20 minute mark the subject starts consuming the meal. Subject consumes the meal within about 5 minutes.
4. After the meal is consumed, the subject remains sitting silently for additional 5 minutes.

3.4 Resulting dataset

The recordings were conducted on two healthy male subjects (author and his colleague) within a span of a week. Each day of the recording session each subject made at least 1 pair of recordings, one for each meal type. Total of 20 recording sessions were made, with 10 recordings per test subject, of which 5 were oat meals and another 5 were salad meals. For a complete list of recordings refer to Appendix A

4 Method description and implementation

This section covers all development and implementation information for all 4 systems covered in this study. This includes pre-processing and creation of datasets as well as detailed information about each system. The 4 systems that are implemented are: PSD meal type classifier, Chew detector, Mel meal type classifier and 3-class meal type classifier.

Appendix B contains the source code of the systems implemented in this study. The cited APT studies ([Kölle, 2019], [Klavins, 2022] and [Bliksvær, 2021]) which may not be available online are also included.

4.1 Tools

The following digital tools were used to achieve the results in this study:

- Audacity - Free open source audio manipulation application.
 - Labeling of chews and inspection of the recordings
- Python
 - Used to implement practically everything in the study.
- Main Python libraries
 - TensorFlow - Neural network models and all algorithms surrounding training, testing and evaluation
 - Soundfile - Loading and saving audio files.
 - Matplotlib - Plots and diagrams
 - Pandas - Dataset storage and manipulation
 - Librosa - Mel spectrogram dataset creation and audio recording analysis
 - Sklearn - PSD feature dataset creation, dataset scaling and other useful mathematical operations

4.2 Data collection

4.3 Pre-processing of the raw audio recordings

Prior to extracting the PSD and Mel features to create the datasets, the “raw” audio recordings are pre-processed. The soundcard and the setup is set to operate at a sampling frequency of 48000Hz. Considering that most sounds associated with food consumption are in 4000Hz frequency range, this sampling frequency

is superfluous and the recordings were decimated to 8000Hz, to comply with 4000Hz Nyquist limit. The 4000Hz frequency was chosen based partially on the findings covered in the theory Section 2.1, as well as the data analysis which will be covered in the discussion Section 6.6.2, which showed that for our recordings most of the frequencies were below 2kHz. 4000Hz seemed like a safe compromise. The decimation also uses the SkLearn's cheby1 low-pass Chebyshev filter with cut-off frequency of 4000Hz as the anti-aliasing filter, however most low-pass filter types will work for this purpose.

4.4 Dataset creation and architecture

For the purposes of this study a *Dataset* refers to a collection of samples in the form of some matrix that are used for training, validating and testing the different systems. Datasets are constructed using two different methods, first of which is based on PSD features, and is the exact same as conceptualized in [Kölle, 2019] as well as used in [Klavins, 2022] (authors masters project) and [Bliksvær, 2021]. The second method, involving Mel spectrogram, was conceptualized during this study.

4.4.1 PSD feature dataset

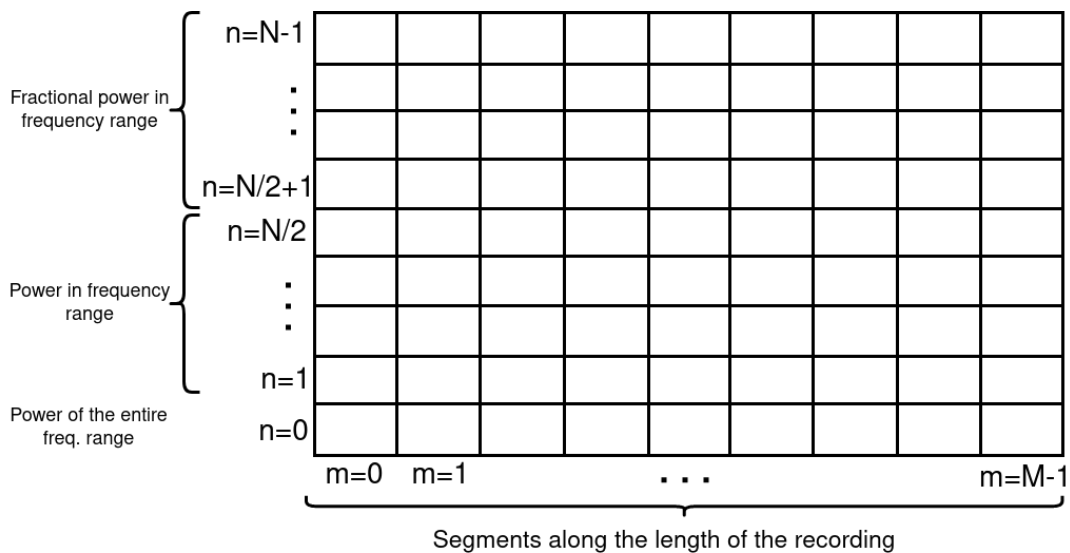


Figure 16: PSD feature matrix of one recording. Entire recording segmented in to M segments, and N PSD based features extracted from each segment.

Motivation for using PSD features As this study is in a sense a continuation of authors masters project [Klavins, 2022], naturally the first approach for dataset creation was the method that had successfully worked in that project.

This specific feature set was introduced by the APT pilot study on early meal detection [Kölle, 2019], and has been subsequently used in similar APT studies with proven results. Originally created to capture characteristics of bowel sounds, the method extracted the total power as well as power and fractional power from 100Hz bands in the 0-2000Hz range. The PSD based, power features were chosen based on the bowel sound analysis which showed increase in the signal power for frequencies below 1000Hz when bowel sounds occurred. The total power captures the overall signal strength information, while the power in the frequency bands allows to differentiate between bowel sounds and environmental noise. From experience, including total power as well as power in frequency bands contains valuable information about the dynamics of the signal. The same type of feature extraction method was successfully utilized in the meal onset detection study using chewing sounds [Klavins, 2022].

Creating the PSD feature dataset A single PSD feature matrix, such as shown in Figure 16, is extracted from each of the 20 recordings. Each pre-processed recording is periodically segmented into segments of defined length and shift, resulting in M segments. The resulting feature matrix is scaled with SkLearn’s MinMaxScaler method, which transforms values for each feature row between 0 and 1. The scaling helps stabilizing the models training process avoiding excessive weight fluctuations due to large error gradients, leading to a better trained model.

For recordings with 8000Hz sampling frequency, the following N features are extracted from each segment:

Feature No.	Description
0	Power in the 0-4000Hz frequency range (total power in that signal segment)
1 to $N/2$	Power in B Hz wide bands, collectively spanning the 0-4000Hz range.
$N/2+1$ to $N-1$	Fractional power in B Hz wide bands, collectively spanning the 0-4000Hz range. Obtained by dividing features no. 1 to $N/2$ by feature no. 0.

Table 2: Description of the PSD based, power features that were used to create the feature matrix in Figure 16

Segment length and shift is customizable. For this study segment lengths of 0.45, 1, 10, 20, 30 and 60 seconds were used. The shift for all segment length configurations is half of the segment length. The previously mentioned study on chewing sounds used only the segment lengths of 10-60s, which worked well for meal onset detection. For this study, in addition to those segment lengths, we also include 0.45s and 1s segments. Motivation for this is the fact that chewing occurrences are shorter than one second (confirmed later in Figure 19). The smaller segment length features could capture more/other details than the

longer 10-60s segment features, and thus potentially work better for meal type classification.

The Frequency bin size is 25Hz for all but one segment length configuration. For 0.45s segments, feature extraction causes numerical errors. For this segment length, the smallest error-free bin size of 80Hz was used. In general, the frequency bins were made as small as possible to include as much frequency detail as possible, however the chosen values were on the low-end constrained by numerical errors in the feature extraction process. Summary of the different PSD feature datasets used in this study is shown in Table 3.

Segment length	Shift	Frequency bin	Frequency range	Resulting shape NxM
0.45 seconds	0.225 seconds	50Hz	0-4000Hz	(161xM)
1 second	0.5 seconds	25Hz	0-4000Hz	(321xM)
10 seconds	5 seconds	25Hz	0-4000Hz	(321xM)
20 seconds	10 seconds	25Hz	0-4000Hz	(321xM)
30 seconds	15 seconds	25Hz	0-4000Hz	(321xM)
60 seconds	30 seconds	25Hz	0-4000Hz	(321xM)

Table 3: List of PSD feature dataset configurations used in this study

4.4.2 Mel spectrogram dataset

Motivation After failure to achieve acceptable results using multiple configurations of the PSD-based features, a new approach was devised. Unlike PSD feature matrices, which use periodically segmented recordings for both training and testing, the new, Mel spectrogram based dataset uses individually extracted chewing segments for training, and periodically segmented recordings for testing.

The use of Mel spectrogram segments was inspired by another concurrent APT research project, which implements the system from [Wang u. a., 2022b].

Labeling

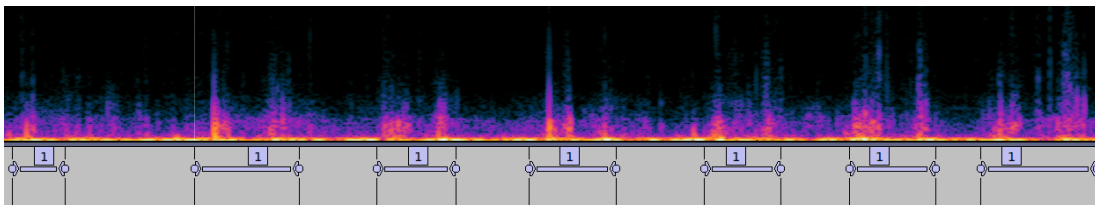


Figure 17: Excerpt from Audacity of a labeled region in the recording

Each individual chew was manually labeled for each of the chosen training recordings. In total 5 of the 20 recordings were labeled , 2 salad and 3 oat

meal recordings (for more details sect. 6.6.4). Labeling was conducted using Audacity, an audio manipulation software, which provides a method for easily marking regions of the recording and exporting the beginning and end time of each marked region to a text file. Each region, aka. *true label*, captures exactly the duration of a chew occurrence, which is characterised by two peaks, one for biting down and the other for relaxing the jaw. Short section of a labeled recording is shown in Figure 17.

Training dataset (Mel spectrogram from individual chew segments)

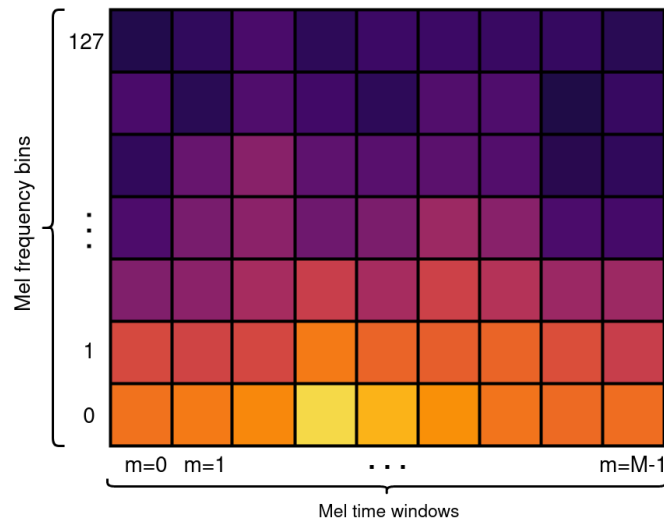


Figure 18: Single sample in the training dataset, created by applying a Mel spectrogram to a segment of the recording selected by the manual chew labels

After the recordings were labeled, Mel spectrogram segments could be created using the labeled regions. To do this, a segment, starting at beginning of the label and ending at $start + segmentlength$, is selected from the signal. A Mel spectrogram is then applied to the segment resulting in a $128 \times M$ matrix, where 128 is the number of frequency bins, and M is the number of time windows within the labeled segment. The resulting Mel spectrogram segment is shown in Figure 18. The same procedure is repeated to create a Mel spectrogram segments for each manual chewing label, which together comprise the training dataset for all 3 Mel spectrogram based systems.

There are 3 parameters which make up the Mel spectrogram segment: Segment length, Mel window length and Mel shift.

Choice of segment length is based on the chew length analysis performed on all manually created chewing labels, the results of which are presented in Figure 19. The median chew length for both oats and salad chews is roughly 350ms, which is chosen as the segment length of the primary dataset. Three other datasets are created, using segment lengths of 100ms, 450ms and 500ms. The

4 datasets will be compared in a test to indicate the optimal segment length. It would be ideal to create segments with the exact length of each individual chew label, however the neural network expects same dimensions for all inputs, hence one-size-fits-all segment length is chosen for this approach.

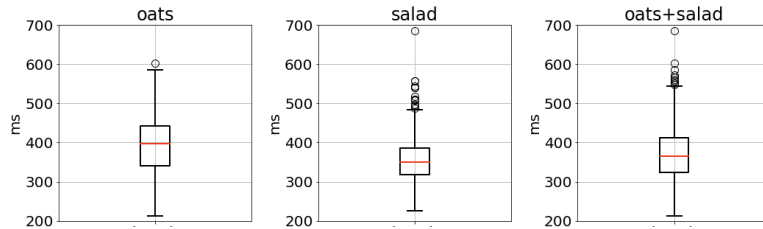


Figure 19: Segment length statistics for the different meal type labels.

Two other parameters are used: *Mel window length* and *Mel window shift*. Both are given in milliseconds and determine the resolution of the Mel spectrogram. For example a 100ms long segment with Mel window length of 30ms and shift of 10ms will produce 8 time windows within that segment. All 4 datasets used the same Mel window length of 30ms and Mel shift of 10ms, a choice which is further discussed in Section 6.6.1. The 4 datasets used in this study are summarized in Table 4.

Dataset Dimensions	Segment length	Mel window length	Mel shift
128x8	100ms	30ms	10ms
128x33	350ms	30ms	10ms
128x43	450ms	30ms	10ms
128x48	500ms	30ms	10ms

Table 4: Training datasets used for Mel spectrogram based systems in this study

Testing dataset (Mel spectrogram of periodically segmented recordings)

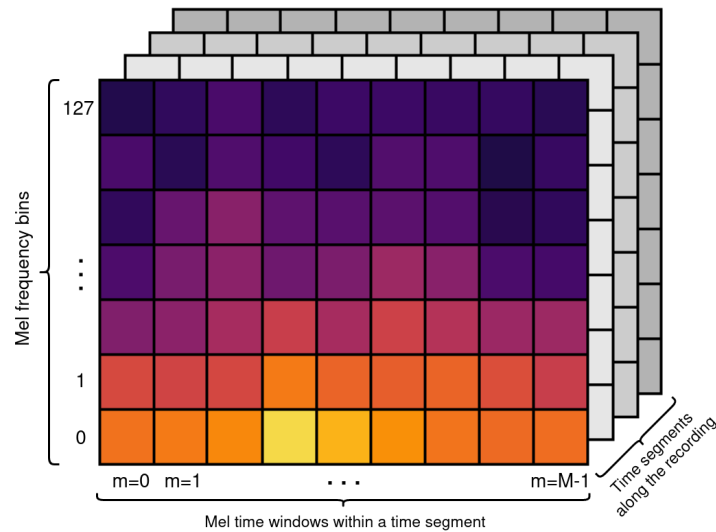


Figure 20: Single recording in the testing dataset, which consists of periodically selected segments from the recording with Mel spectrogram applied to each. Each segment corresponds to the same type of Mel spectrogram segment as in training dataset, shown in Figure 18.

In a practical application the system would not have the luxury of having input samples in the form of segments that only and exactly capture each chewing occurrence, such as is the case for the training dataset. For a live use case the input would be an audio stream from a microphone or a recording of some length. One approach is to create the segments periodically along the length of the recording. Testing dataset is created according to this format.

The neural networks used in this study expect that dimensions of input samples are equal, hence the same $128 \times M$ matrix needs to serve as input for testing. Here's what's different for the testing dataset: Instead of extracting Mel spectrogram from manual chew labels, the entire recording is periodically segmented and Mel spectrogram is extracted from each segment. For this, in addition to segment length, Mel window length and Mel shift, a new parameter is required: the *Segment shift*. Segment shift defines the relative displacement between the segments, given in milliseconds. This results in a $N \times 128 \times M$ matrix for the entire recording, with N being the number of segments spanning the recording. Choice of segment shift value is discussed in Section 6.6.1

Each training dataset from Table 4 has its corresponding testing dataset. Testing datasets are summarized in Table 5.

Sample dimensions	Segment length	Segment shift	Mel window length	Mel shift
Nx128x8	100ms	50ms	30ms	10ms
Nx128x33	350ms	50ms	30ms	10ms
Nx128x43	450ms	50ms	30ms	10ms
Nx128x48	500ms	50ms	30ms	10ms

Table 5: Testing datasets used in this study. N corresponds to the number of segments in each recording

Note: Segmenting the entire recording in such manner yields files of 2GB a piece. Majority of the recording contains superfluous amounts of silence for our purposes. Thus for storage considerations only a part of the recordings are segmented. For simplicity, the contents of the testing dataset will be referred to as “segmented recordings”, but in practice it means that each segmented recording only contains a part of the segments in the 30 minute-or-so long recordings. For chew detector and 3-class meal type classifier, a region from 15 minutes to the end of the recording is used. This translates to 5 minutes of silence prior to and after the meal region, as well as the meal region itself. The included 5 minute silence parts aren’t actually needed to test a meal type classification scenario, since as noted previously, its already possible to isolate the meal region and only focus on that. These parts are included out of curiosity, to see how these systems, which can identify regions of silence, perform outside of the meal region.

The Mel meal type classification system only classifies chews, and as such doesn’t require any silence regions for testing. Therefore for this system the testing dataset contains only segments from the meal region.

4.5 PSD Meal type classifier

This system was the first attempt at creating a meal type classifier using the meal type audio recordings. This system was based on the PSD feature dataset which was successfully used during authors master’s project [Klavins, 2022] to implement a meal onset detection system. Although that project utilizes an SVM model, it was thought that re-using the features and coupling with the more advanced neural network would be sufficient for meal type classification. This subsection covers the first attempt at implementing meal type classification system.

4.5.1 Network model

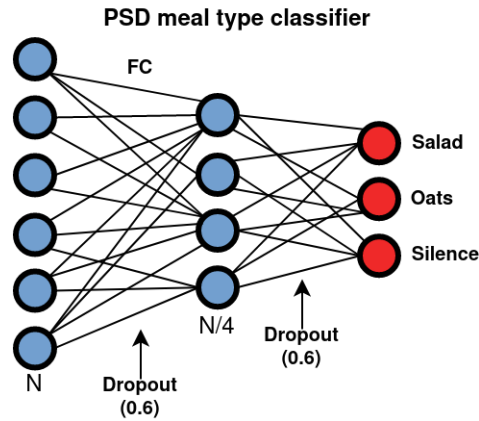
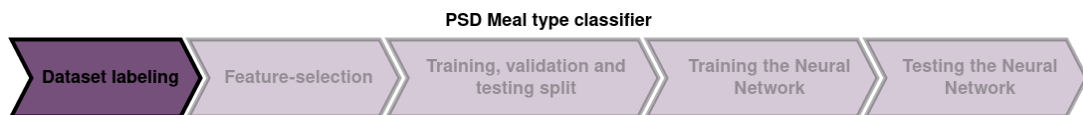


Figure 21: Neural network model of the PSD meal type classifier)

Although many different layer combinations and configurations were tested, the architecture given in Figure 21, works as good as nearly any other. Adding more layers doesn't improve the performance, so in favor of picking a computationally light model, this model was settled on.

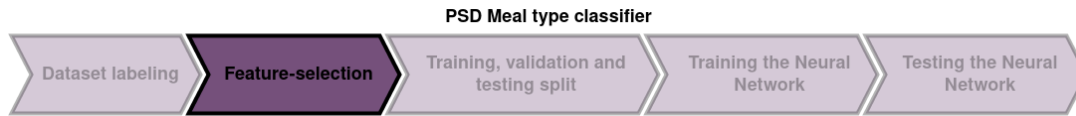
Model consists of two fully connected layers. Input dimension of the first layer corresponds to the number of feature bins for each input segment, N . The second layer contains 1/4th of the inputs of the first layer. Each layer is accompanied with a dropout layer with dropout rate of 0.6. The dropout is rather aggressive to prevent excessive over-training, which happens within very few epochs. The network produces 3 output probabilities, one for each class: "salad", "oats" and "silence".

4.5.2 Dataset labeling



As a first step, the system creates labels for each of the recordings in the dataset. Each recording is a matrix such as in Figure 16, with N features and M time segments. The labels are $M \times 1$ arrays with each element corresponding to the class that specific segment belongs to. Labels of classes "silence", "oats" and "salad", get the values 0, 1 and 2 respectively. Segments of the entire meal consumption region are marked as either "oats" or "salad", depending on the meal that's consumed. The rest of the segments are marked as "silence".

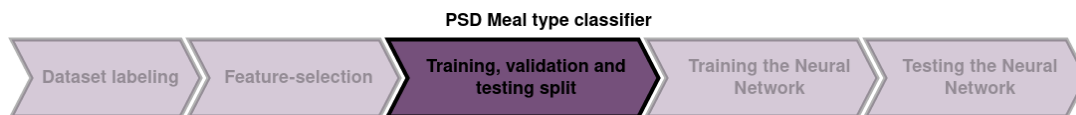
4.5.3 Feature selection



The entire dataset consists of PSD features as detailed in Section 4.4.1. Recall that each segmented recording is a $N \times M$ matrix where N corresponds to the number of features.

For this system implementation feature selection is performed on the dataset. This process picks the K best of the N features. Feature selection is performed using the *SelectKBest* method with Chi-squared metric, from SkLearn's library. The threshold is configurable, but for the final model is set to select 30% of the best features. While the influence of the feature selection in general is quite minimal, multiple threshold values were briefly tested and the 30% was found to give the least false positive predictions across the different dataset configurations. The feature selection then returns the indexes of features to keep, and the rest are discarded from the dataset.

4.5.4 Training, validation and testing split



This system divides the dataset of 20 recordings into training, validation and testing splits. First, 20% (4 recordings) of the dataset is set aside as testing split, and the remaining 80% is further split 80/20 (13 and 3 recordings) into training and validation splits. The resulting dataset is shown in Figure 22. Note that the actual percentages are more crude due to them being rounded to include whole recordings. Also the recordings for each split are chosen randomly each time the splitting operation is performed.

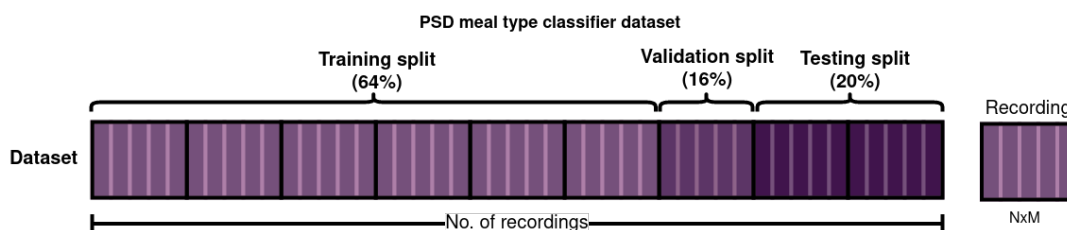
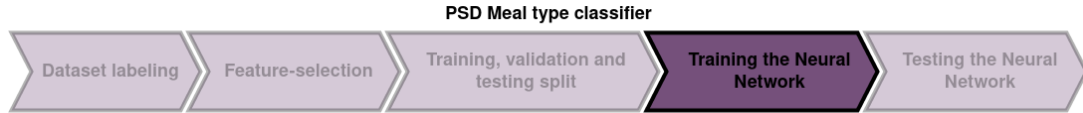


Figure 22: Overview of the dataset for the PSD meal type classification system

4.5.5 Training the neural network

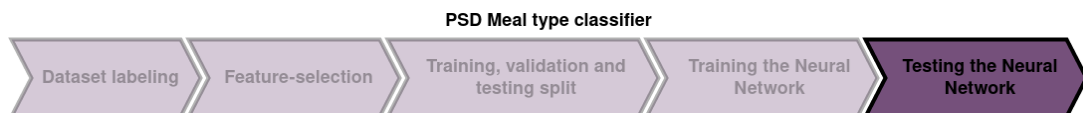


The training is handled by the built in methods of Tensorflow library, which require little work on the users part. There are however some training parameters the user can configure to tune the network. The final model was trained using the following parameters:

- Loss function: Binary categorical crossentropy
- Optimizer: Adam, with learning rate of 0.001
- Batch size: 1024
- Epochs: 7

Note as a technical detail, that although the dataset is thought of as a collection of separate recordings, the network trains only on per-segment basis. This means that with the training split, which contains 14 segmented recordings, the network takes only one Nx1 segment at a time.

4.5.6 Testing the neural network



To test the network the segmented recordings in the testing split are used. These are 4 randomly chosen recordings and they are tested separately. Each test gives an Mx1 array of labels corresponding to each segment of the test recording.

4.6 Chew detector (CD)

The chew detector is one of the 3 systems implemented with Mel spectrogram dataset. It attempts to identify whether or not a segment of the recording contains a chew. This system could be used on its own for dietary monitoring, however its main goal is to augment the Mel meal type classification system, which will be detailed in a later subsection.

4.6.1 Network model

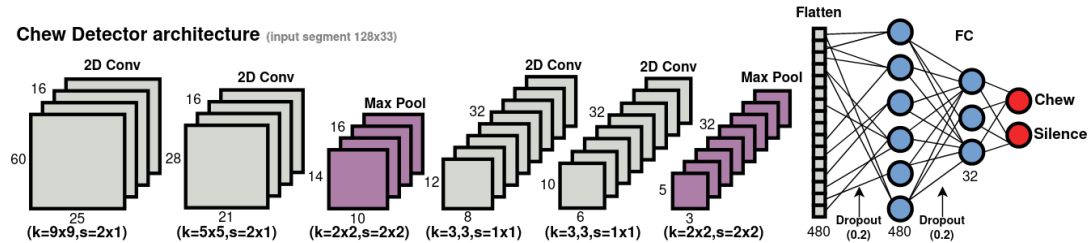


Figure 23: Neural network architecture for the chew detection system. Layer dimensions correspond to input sample dimension of 128x33, thus for other inputs tested the dimensions will vary. (k=kernel size, s=stride)

Figure 23 illustrates the neural network architecture of the chew detection system. This architecture was arrived at through testing many different combinations and orders of pooling and convolution layers, different kernel, stride and filter size configurations, as well as number of fully connected layers and node amounts. Although not very methodically, the process involved picking one specific parameter, testing random values/combinations and narrowing down to a value/combination that works the best. The architecture in Figure 23 is not meant to represent the best possible model for this type of system. In fact, multiple, slightly different configurations lead to similar results, and the final chosen values were among the ones which gave the best results. Obtaining the best possible model and detailing the process is beyond the scope of this study.

As will be detailed later in the study, the Mel meal type classifier and the 3-class meal type classifier, was found to perform optimally using this same architecture. Also note that the dimensions of the layers in the figure correspond to an input sample of 128x33. Other input sample dimensions will produce different layer dimensions.

The classifier is a convolutional neural network (CNN) with two conv-conv-pool (CCP) layers and subsequent fully-connected layers. Note that none of the layers have enabled padding, meaning that the convolution operations will also reduce the dimensionality.

First pair of convolutional layers are equipped with 16 feature filters, and stride of 2x1. This stride allows to reduce the dimensionality of the frequency dimension (128) while keeping the time dimension (M) intact. This was done because the Mel spectrogram segments are rectangular, and application of uneven stride effectively squares off the dimensions. This prevents the shorter time dimension of the filters from becoming too narrow, thus helping with feature learning. The second pair of CCP layers increases the filter amount to 32,

and has a kernel of 3x3 and stride of 1x1 for both.

Each pair of CCP layers is complete with a max-pooling layer. Both are identical with 2x2 kernel and 2x2 stride.

After the two CCP pairs, the features are flattened and followed by two fully-connected (FC) layers. First FC layer is the same dimension as the output of the flattening layer. The second FC layer has 32 nodes, with the intuition being that each filter gets its own node. Additionally, two dropout layers are placed before and after the first FC layer, both with dropout rate of 0.2. Finally the two output nodes are activated by a softmax.

4.6.2 Dataset initialization and pre-processing



The first step in the chain of events for chew detection system is loading in and pre-processing the dataset.

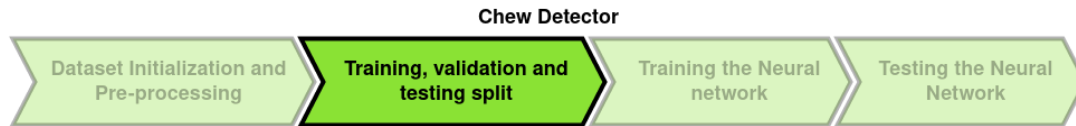
Loading the dataset also automatically loads the labels for each Mel segment in the dataset, thus no explicit labeling is required. Each segment belongs to one of two classes, “oats” or “salad”. The Mel training and testing datasets are loaded in separate dataframes (Pandas containers for data).

As for pre-processing, this is not to be confused with the pre-processing that was performed on the raw audio recordings to down-sample them. When creating the dataset no special scaling is applied. Due to gradient operations in the NN it’s advised to scale the datasets prior to use in training. This will result in better learning performance. The following scaling operations are performed:

1. Logarithmic scaling of the dataset to narrow the dynamic range of the values, by applying $20 \cdot \log_{10}(x)$ operation. This also makes the dataset samples visually presentable in a plot.
2. Sklearn’s MinMaxScaler method is applied to scale values between 0 and 1.

MinMaxScaler first analyzes the entire training dataset, effectively learning the max and min values in the dataset. Subsequently, it transforms the entire training dataset to values between 0 and 1. The same MinMax scaler that’s been initialized with training dataset is also applied to the testing dataset. The idea here is to scale the values such that they are suitable for NN, but also preserve the amplitude differences between individual samples in the datasets. If scaling was performed on samples individually, the amplitude differences would be lost.

4.6.3 Training, validation and testing split



For the chew detection system the creation of training, validation and test split goes through a two step process. First step is augmenting the training dataset with “silence” class segments. Second step is performing the actual split.

Augmenting training dataset with “silence” class segments Chew detector outputs one of two classes: “chew” or “silence” (non-chew). The Mel spectrogram training dataset is comprised only of segments from the “chew” class, hence the dataset requires additional segments of the “silence” class. These silence segments are actually taken randomly from the testing dataset. To avoid mixing the samples from the recordings, the silence segments are taken from the same recordings that were manually labeled and used to create the training segments. Note that each segmented recording in the testing dataset contains two types of “silence” segments: ones that are located outside of the meal region, and ones which are part of the meal region, but don’t contain any chews.

Following process augments the training dataset:

1. Select a segmented recording from the testing dataset. The recording must be chosen among those that were used to create the training dataset.
2. For each recording find all the segments which do not overlap with any labeled chew segments of the corresponding recording. Since none of these found segments overlap with a chewing label, they are technically from “silence” class.
3. From all these chosen “silence” segments, randomly pick a set number of samples. This number is chosen by the user. Equal number of meal region and non-meal region silence segments are chosen.
4. Finally, add the chosen “silence” segments to the training dataset

For all of the testing done in this study, the augmentation chose 3 times as many “silence” class segments as there were “chew” segments. Other ratios, such as 1:1, 1:2 were tested, however 1:3 “chew” to “silence” segment ratio was found to work best. Ratio of 1:4 wasn’t possible as the recording started to run out of “silence” segments to pick from.

Splitting the training dataset After the training dataset has been augmented with segments from “silence” class, it is split into a training, validation and testing splits, with each being 76%, 19% and 5% of the training dataset respectively. These seemingly arbitrary percentages were arrived at by firstly setting aside 5% to be the testing split, and the remaining 95% were split into 80% training and 20% validation split, which is along the lines of how the splits are conventionally created for machine learning applications.

Note: The two terms *testing split* and *testing dataset* will be used throughout this thesis, and its important to know the difference. Testing split contains individual segments created from the manual chew labels. Testing dataset contains all 20 recordings which have been segmented periodically (thus chronologically), and do not necessarily contain precise chewing occurrence, such as in the training split. In order to avoid confusion, Figure 24 shows the different datasets and splits.

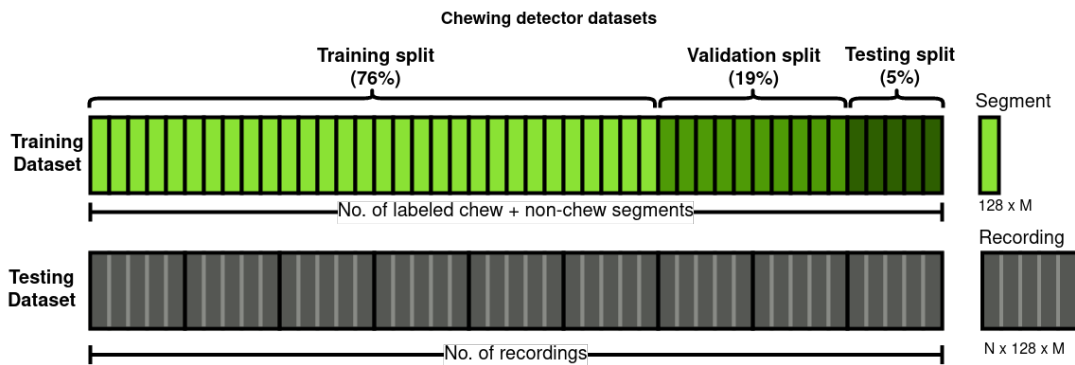


Figure 24: Overview of the dataset structure for the chew detection system

All “chew” class segments in the testing split precisely capture a chew. Results from the testing split can’t be used to indicate real world performance, however it provides an insight of how well the network learned the training dataset data. The actual testing dataset, which consists of entire segmented recordings, is not split into anything, and is saved for the actual testing part.

4.6.4 Training the neural network



Training of the neural network is a very straight forward process, thanks to the convenient Tensorflow library, which does all of the hard work. Some user

input is however required for choosing the training parameters as well as the network architecture. The architecture is once again shown in the Figure 23, and the following training parameters are used:

- Loss function: Sparse categorical cross-entropy
- Optimizer: Adam, with learning rate of 0.0001
- Batch size: 128
- Epochs: Dynamic epochs using Early stopping with validation loss delta of 0.0001 and patience of 30 epochs. This configuration allows the training to stop automatically once the condition is met.

Multiple different values for learning rate, batch size and early stopping parameters were briefly tested before the values given above were chosen. The final chosen values do not necessarily represent the best choice, however they were chosen because they worked just fine for producing a good model. These values are included purely to give more context to the whole training procedure, however in reality, these precise values aren't very significant to the performance. Slightly different values might work just as well.

The early stopping mechanism allows the training to terminate once the change in validation loss across the last 30 epochs doesn't change more than 0.0001. This means that once the validation loss has reached a plateau, further training is omitted to avoid overfitting, such as described in theory Section 2.3.7.

4.6.5 Testing the Neural Network



The network is tested in two ways: using the testing split, and using the testing dataset.

Testing split As described in subsection 4.6.3, the testing split is derived from the training dataset. A part of the training dataset is set aside to be used for rudimentary testing after the training is done. Again, each sample is a Mel spectrogram segment containing either a chew or silence (non-chew). Once the test is performed the results can be displayed in a confusion matrix. Again, this test isn't a good indication of how well it detects chews in a segmented recording such as the testing dataset. The results from this test were mainly used during

development of network architecture for a simple and fast indication of models performance.

Testing dataset The testing dataset contains Mel spectrogram segments which are selected from a part of a recording in a periodic manner. In order to reduce computational cost and storage requirements, only a part of the whole recording is used, specifically from 15 minutes to the end of the recording (as mentioned in the note in Section 4.4.2).

If it's not already clear, it should be noted that for the chew detector, as well as for the next two systems, the recordings which were used to create training dataset, were also used to create the testing dataset. In general this is a big no-no as a machine learning practice. However in this case, even though the same recordings are used, the resulting samples in the datasets differ. This is because while the training dataset contains segments which capture only and exactly a chew, the testing dataset segments the recordings periodically, meaning that each segment will not necessarily capture the exact chew. In either case, it will be evident upon viewing the results in the next section, that the performance on the recordings that were used in training is on par with the rest.

Finally, the testing procedure itself using the testing dataset is as follows. Each $N \times 128 \times M$ testing recording is evaluated by the trained model. The output from the test is a $N \times 1$ array, where each element is a predicted label, corresponding to a $128 \times M$ segment of the recording. The resulting prediction labels were also saved to a file, which are imported by the Mel meal type classifier and used to remove non-chewing segments.

This part only describes the technical execution of using testing dataset for model evaluation. The methodical tests which will be presented in the results section are described by later in this section (4.9).

4.7 Mel meal type classifier (MMT)

Main goal of the meal type classification system is to differentiate between the food types that are consumed in the different recordings in our dataset. The meals consist of either oats or salad, which are the two classes this system will attempt to identify. In addition to classifying any given input segment, regardless of whether or not its a chew, the system will be augmented with the chewing detector to exclude the non-chew segments and improve MMT's classification ability. This will be discussed in the 4.7.5 section, where it comes into effect.

4.7.1 Network model

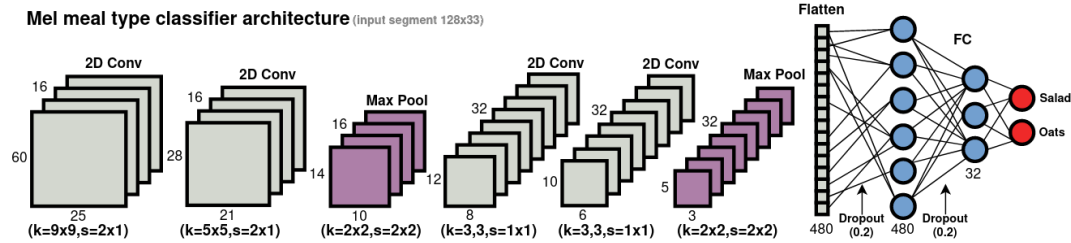


Figure 25: Neural network architecture for the Mel meal type classification system. Dimension numbers correspond to input sample dimension of 128x33, thus for other inputs tested the dimensions will vary. (k=kernel size, s=stride)

If this architecture looks very similar to the one used for chew detection system, then that's because it is in fact the same network architecture. Although initially the architecture was different than for the CD system, through the process of testing models with numerous layer, filter, stride, kernel configurations, the same architecture turned out to perform best. As was already mentioned for the CD system, obtaining the best possible network architecture for this system and detailing the entire process is beyond the scope of the study. The process for testing different network architectures follows the same process that was used for the CD system.

4.7.2 Dataset pre-processing



The dataset is first pre-processed in the same way that it is for the chew detector. In fact the same dataset is used both for MMT and CD systems. In order to keep this short, please refer back to 4.6.2 for in-depth method details.

4.7.3 Training, validation and testing split



After pre-processing, the training dataset is split into a training, validation and testing splits in a similar way to the CD system, described in 4.6.3. The difference here is that the training dataset isn't augmented with "silence" class prior to splitting. This isn't required as all the segments in the training dataset are chewing segments belonging to "oats" or "salad" classes. Thus this part of the system only performs splitting of the training dataset.

The splitting operation separates the training dataset in the same proportions as for the CD dataset. That is 76% for training, 19% for validation and 5% for testing. The resulting datasets are shown in Figure 26

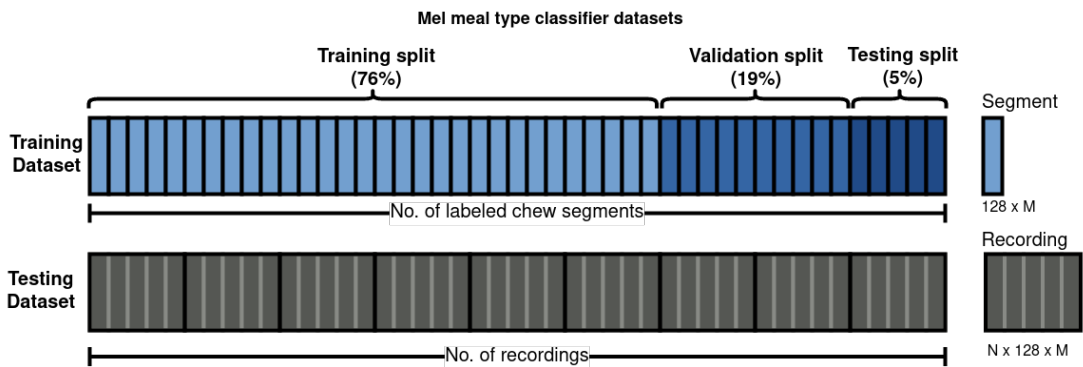


Figure 26: Overview of the dataset structure for the Mel meal type classification system

4.7.4 Training the Neural Network



Training the network here is identical to the way the CD was trained (section 4.6.4). That is to say that Tensorflow manages this part. The following training parameters are used for the MMT system training:

- Loss function: Sparse categorical cross-entropy
- Optimizer: Adam, with learning rate of 0.0001
- Batch size: 128
- Epochs: Dynamic epochs using Early stopping with validation loss delta of 0.001 and patience of 15 epochs.

Just like for the CD system, these chosen values are not meant to represent the best possible choice, however from the brief testing that was done with

different values, these seemed to perform optimally. The patience and delta for early stopping is slightly different than for the CD system, which seemed to allow the training to stop at a better point, where the model was slightly more optimal than with CD values.

4.7.5 Testing the neural network



The network is tested in two ways: using the testing split, and using the testing dataset.

Testing split The testing split is used after the training to test the networks performance on the labeled chew segments. The result of this test is a confusion matrix, and it gives an indication of how well the network has learned the training dataset. Again, the main purpose of the testing split was to give a simple and quick indication of the models performance during the development of the network architecture. The results from this test aren't very insightful when it comes to the performance in a realistic scenario, which is what the testing dataset is used for.

Testing dataset Once again the testing dataset is used to benchmark meal classification performance in a realistic scenario, one in which each segment doesn't necessarily precisely capture a single chewing event. Recall that the testing dataset consists of 20 segmented recordings. Each recording is tested individually. The test is performed twice: First on all segments in the recording, and second, only on the segments which have been identified as chews by the chew detection system.

The first test is straight forward. Each recording in the testing dataset is fed to the network, producing an array where each element is a class label corresponding to that segment in the recording.

For the second test, the predicted chew labels are loaded from a file which were created by the CD system. These labels are applied to the same MMT meal type prediction array as in first test. This effectively nulls all the segments which aren't identified as chews by the CD. Consequently all the segments that are identified as chews, retain their predicted label as either "salad" or "oats". Prediction ratio is used as the primary metric.

4.8 3-class meal type classifier (3C)

The 4th system in this study is one which combines the MMT and CD into one system. Although this system is somewhat redundant in the sense that the two previous systems are combined to produce the same result, it was initiated as a part of the study to provide an alternative avenue, at a point where the two system approach didn't work properly. Full description and evaluation of this system is mainly included to give a broader overview of the things that were attempted in this study. There are however some benefits for this system which are discussed later in Section 6.4.

4.8.1 Network model

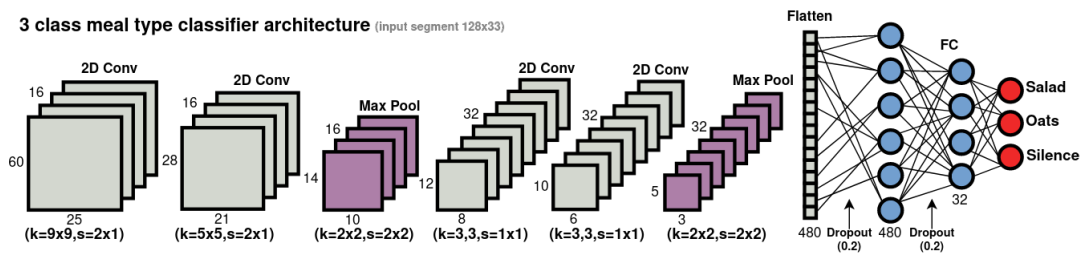


Figure 27: Neural network architecture for the 3-class meal type classification system. Dimension numbers correspond to input sample dimension of 128x33, thus for other inputs tested the dimensions will vary. (k =kernel size, s =stride)

The network architecture is once again the same. Each system, including the 3C, initially started with their own network architecture, however after many hours of tuning the network, all 3 systems ended up with the same architecture. For completion, the model is shown in Figure 27

4.8.2 Dataset pre-processing



This system uses the same dataset as the previous two systems, that is, a collection of chewing segments for training dataset, and periodically segmented recordings for the testing dataset. As such, the pre-processing step is also the same, so for more details please refer back to Section 4.6.2.

4.8.3 Training, validation and testing split



The training dataset is augmented and split exactly like for the chewing detection system, described in Section 4.6.3. The training dataset thus contains segments of 3 classes: “oats”, “salad” (both chews) and “silence”. The splitting operation once again separates the training dataset in the same proportions, shown in Figure 28

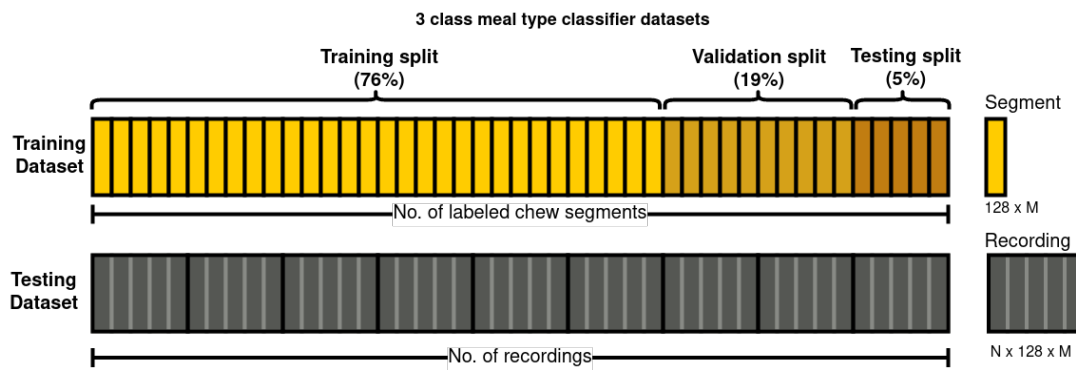


Figure 28: Overview of the dataset structure for the 3-class meal type classification system

4.8.4 Training the Neural Network



Training procedure is no different than for the CD and MMT systems. The following training configuration was used:

- Loss function: Sparse categorical cross-entropy
- Optimizer: Adam, with learning rate of 0.0001
- Batch size: 128
- Epochs: Dynamic epochs using Early stopping with validation loss delta of 0.001 and patience of 30 epochs.

All parameters are the same as for CD and MMT systems, except for delta and patience for early stopping. Again, this change optimizes the point at which the training is stopped, resulting in a more optimal model.

4.8.5 Testing the neural network



The training procedure is essentially the same as for the CD and MMT systems. The testing split is once again mainly used as an indication of networks performance during the development of the architecture. For evaluation of performance in a more realistic scenario, the testing dataset recordings are used. For each testing recording the network outputs an array of predicted labels for each segment of the recording, the same way as for the CD and MMT systems. Unlike the MMT system, there is no need to apply CD predicted chewing labels, as this system already identifies the silence segments.

4.9 Testing procedures

This subsection describes the different testing procedures that will be used to obtain results which will be presented in the results section. Two main tests are performed; System tests, which evaluate the performance of each system based on the described metrics, and dataset test, which will compare the performance of the 4 different Mel spectrogram datasets in order to identify the segment length parameter which gives the optimal results.

Note: Best achieved model The following few paragraphs detail the testing procedures for the different systems. In any case, unless indicated otherwise, each system is using the best achieved neural network model. The measure for obtaining the best model was prediction ratio. To obtain the best model, the system cycle is repeated until a model, which performs the best out of the bunch, is found. That model is saved and used for these tests. This process can take many attempts, as it depends on two randomized factors: 1, the random selection of segments for training, validation and testing splits, and 2, the randomized Tensorflow model initialization. The second factor is out of our control, however the first factor is deliberately randomized. This is mainly to avoid performance bias due to the selected training segments. Downside of this randomization is that the process for finding the best model takes time.

4.9.1 System and Dataset test - PSD meal type classifier

The PSD meal type classifier is the only system in this study which used PSD based features. This was also the first approach at meal type classification. The numbers will be provided in the results section, however it will be mentioned already here that for the purpose of meal type classification this approach worked very poorly.

Creating any objective test of this system is another issue. As the results section will explain, the classifier performed nearly identically on all the different segment length datasets that were tested (from Table 3). In all test cases the system has a bias towards one class or the other, depending on the randomly selected training, validation and testing split recordings, as well as the randomized initialization that Tensorflow does to its network models. Due to this fact, no specific test will be performed on this system. Instead the results directly from the test in Section 4.5.6 will be presented.

4.9.2 System test - Chew detector (CD)

The chew detection system was designed to be used in combination with the meal type classifier. This system test will provide indication of the performance in relation to the true labels which were created manually, as well as the performance when used in combination with the Mel meal type classifier.

Testing procedure follows the methodology of “testing dataset” in Section 4.6.5. For this test the 350ms segment length dataset (128x33) is evaluated using the best achieved CD model. Only 5 of the recordings have corresponding true chewing labels. These recordings will be used to assess the chew detection performance based on accuracy, precision and recall metrics. In addition that, the predicted labels from all 20 recordings will be combined with the best achieved MMT model to provide average prediction ratio score.

The results for this test are given in Table 6

4.9.3 System test - Mel meal type classifier (MMT)

Creating a meaningful performance metric solely for the MMT system is a difficult task. This is because it is a 2 class classifier for a dataset which actually has 3 classes, the 3rd being “silence”. Hence the segments which don’t contain any type of chew, will still be forced to the class closest to either “oats” or “salad”. In practice this means that these “silence” segments will be classified as mostly “oats”.

In order to properly test the MMT system, the “silence” segments need to be removed. Two options are available: using predictions from the CD or using the true (manual) chew labels. The results section includes performance results for both options. CD prediction labels for the 20 testing dataset recordings were

created using the best achieved CD model trained on the 350ms segment length dataset (128x33). The MMT system for this test also uses the best achieved model, trained on the same 350ms segment length dataset.

5 of the 20 recordings in the testing dataset, have corresponding true labels, and will be used to gauge performance for a scenario with “ideal” chewing labels. Additionally, the accuracy, precision and recall metrics will be calculated using the true and predicted labels for the 5 recordings. Results are shown in Table 7

4.9.4 System test - 3-class meal type classifier (3C)

The 3 class meal type classifier (3C) essentially combines the MMT and CD in one system. Since its using the same network model and dataset, the testing procedure will be similar to the two separate systems. All 20 testing dataset recordings will be assessed using the best achieved 3C network model, trained on the 350ms segment length dataset. Performance for all 20 testing recordings will be measured using prediction ratio, as well as the accuracy, precision and recall measures for the 5 recordings with true labels.

4.9.5 Dataset test - Mel spectrogram features

The 3 Mel spectrogram based systems all used the same type of dataset for training and testing. Recall Tables 4 and 5, which contain the 4 datasets which will be benchmarked in this test.

Chew detector and Mel meal type classifier The chewing detector and Mel meal type classifier are meant to be used in tandem, and will thus be tested as such. That said, their collective performance will depend on their individual performance, due to the fact that they are trained separately. In order to test this properly the following procedure will be used:

One of the systems will be trained on the control dataset, which is chosen as the 350ms segment length dataset (128x33). That system will be trained once and remain unchanged during the testing of the other system. The other system will perform 3 test runs for each of the 4 dataset configurations (total 12 runs). For each test run, to evaluate the performance, the systems will be combined and the resulting prediction ratio from the MMT will be used as a metric. This test is performed twice, such that each system is tested on the 4 datasets, while the other is the control.

The intention is to test the impact of each dataset type on CD and MMT separately, as its not guaranteed that both will perform best on the same type of dataset. Also note that because the dataset splitting operation randomly chooses the segments for each split, the performance from run to run may fluctuate by a considerable amount. This randomization can't be completely

removed as part of it is due to the way Tensorflow initializes the models. Also, having additional randomization from the dataset splitting operation eliminates bias that a certain choice of training segments may produce. Therefore our tests include 3 runs for each dataset, such that this randomization can be averaged, and a somewhat realistic performance impression is gained.

3 Class meal type classifier The 3 class meal type classifier doesn't rely on other systems to perform meal type classification, thus the above described testing procedure is simplified. For this system the same 3 runs will be performed on each of the 4 datasets.

5 Results and observations

5.1 Results - PSD Meal type classifier test

The results from testing the PSD dataset configurations from Table 3, can be summarized in the two Figures below.

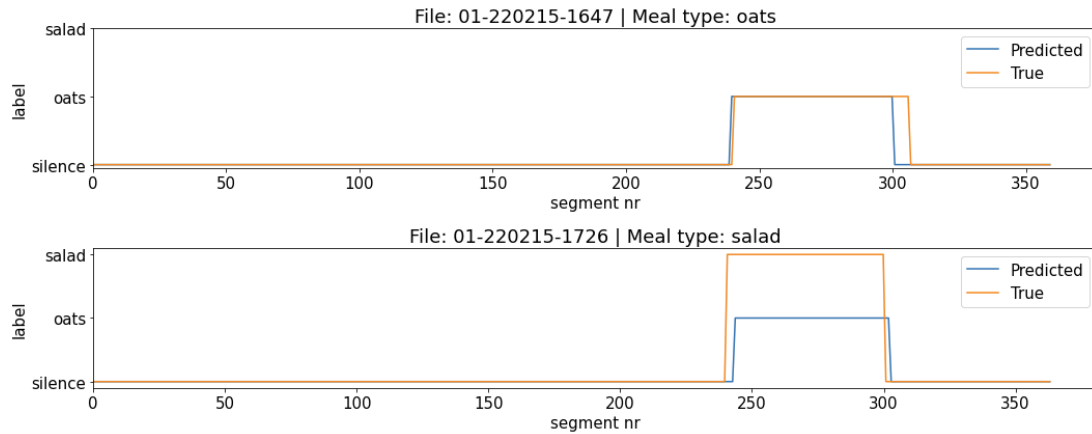


Figure 29: Typical prediction result for the PSD meal type classification system when using datasets with segment length of 0.45, 10, 20, 30 and 60 seconds. Predicted labels in blue, True labels in orange.

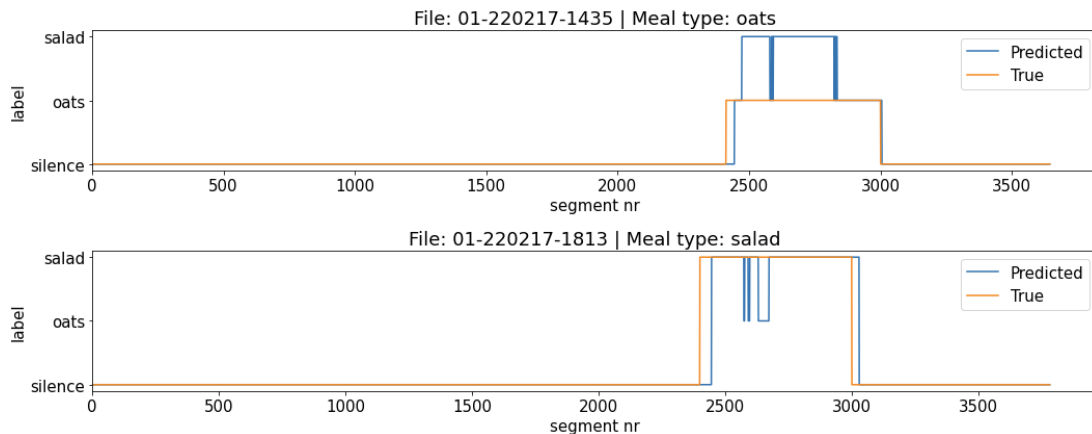


Figure 30: Typical prediction result for the PSD meal type classification system when using dataset with segment length of 1 second. Predicted labels in blue, True labels in orange.

Figure 29 represents the typical result when training the system using PSD features with segment length of 0.45, 10, 20, 30 and 60 seconds. The system is always biased towards one class or the other, leading to all predictions being

of the same class. Which class gets biased is random and changes each time the system is trained. This randomization is caused by the dataset splitting operation (which is intentional), as well as the randomized Tensorflow model initialization. Although randomization is responsible for the changing bias, the bias itself is a flaw of this system.

A slightly better result is obtained when the system is trained using PSD features with segment length of 1 second, as shown in Figure 30. The bias is reduced, allowing few segments during the meal time to be classified correctly, however majority of the predictions are still subject to a bias. This dataset configuration was tested (ran) multiple times and the best observed case achieved accuracy in the mid 60% range, for all 4 test recordings. All in all, the 1 second segment length dataset performed slightly better than others, however the results were still very variable from run to run. None of the runs managed to correctly classify the meal types of all 4 testing recordings.

Despite the meal type classification flaws, this system excels at meal onset detection. All of the tested dataset configurations managed to correctly detect meal regions for all the tested recordings. This isn't surprising considering that the same type of dataset was used in [Klavins, 2022] to detect meal onset.

5.2 Results - Chew detector system test

Recording	Meal Type	Accuracy	Precision	Recall	MMT Prediction Ratio (CD)	MMT Prediction Ratio (True)
01-220215-1528*	oats	0.973	0.290	0.763	95.2%	97.8%
01-220215-1601*	salad	0.948	0.331	0.902	86.7%	94.9%
01-220215-1647*	oats	0.979	0.271	0.669	89.5%	94.5%
01-220215-1726*	salad	0.974	0.408	0.876	89.5%	97.2%
01-220216-1238*	oats	0.976	0.331	0.840	98.0%	100.0%
Avg. 5 recs.	-	0.970	0.332	0.837	91.8%	96.9%
Total 20 recordings	-	-	-	-	90.5%	-

Table 6: Results from the chew detector system test 4.9.2. Recordings marked with * have true labels available. For this test the network is trained and tested with the 350ms segment length dataset (128x33).



Figure 31: Chew detector predicted and true labels for two testing recordings: oats meal (top), salad meal (bottom). The green labels indicate the predictions, which were either class “silence” or “chew”. The red labels are the true labels, and are provided for reference.

Table 6 contains the results from the system test in Section 4.9.2. For this test the dataset with 350ms segment length is used. The test evaluates 5 of the 20 recordings in the testing dataset which have true labels available. The 3 performance metrics, the accuracy, precision and recall, are calculated for the predicted labels relative to the true labels. The 6th column gives the CD predicted label performance on the best achieved MMT system model. For comparison, the MMT performance using the true labels is given in column 7. The average prediction ratio, obtained using predicted labels of all 20 recordings is given on the bottom row.

The accuracy, precision and recall metrics measure systems ability to correctly detect chewing segments. During system development it was observed that the most important metrics, which translate directly to MMT performance, was the precision and recall. These are observed slightly higher for the salad meals. Curiously, the MMT prediction ratio is higher for the oats meals. The most lacking aspect of the performance is the precision, which averaged for all 5 meals to 0.332.

In combination with MMT, which is its intended purpose, the CD system performs very well overall. The average prediction ratio for the 5 recordings is 91.8%, which on average is 5.1% less than the true labels achieved for the same recordings. Additionally, the prediction ratio of the MMT using CD predicted chew labels is 90.5% when averaged for all 20 testing recordings.

Figure 31 shows the true and predicted labels for the first 2 meals in the table. Each spike in the plot corresponds to prediction label of a segment in the test recording. Recall that the segmented testing recordings contain only a part of the entire recording, specifically 15 minutes to the end of the recording. This corresponds to roughly 5 minutes of silence, then 5 minutes of meal consumption, followed by another 5 minutes of silence. The predictions also reflect these regions of the recording. Few false positive chew detections are made outside of the meal region, which is an issue solved by a meal onset detector such as in Klavins [2022], or even by using the PSD meal type classifier, which as we’ve seen works well for this purpose. This figure is included mainly to give a visual sense of the predictions that the CD system provides.

5.3 Results - Mel meal type classifier system test

Recording (ch-yymmdd-time)	Predicted Class (No chewing labels)	Prediction ratio (CD chewing labels)	Prediction ratio (True chewing labels)
01-220215-1528 (oats)*	98.3% (oats)	95.2% (oats)	97.8%
01-220215-1601 (salad)*	71.3% (oats)	86.7% (salad)	94.9%
01-220215-1647 (oats)*	96.1% (oats)	89.5% (oats)	94.5%
01-220215-1726 (salad)*	71.5% (oats)	89.5% (salad)	97.2%
01-220216-1238 (oats)*	99.2% (oats)	98.0% (oats)	100.0%
01-220216-1315 (salad)	70.4% (oats)	84.9% (salad)	-
01-220216-1404 (oats)	94.6% (oats)	85.9% (oats)	-
01-220216-1438 (salad)	57.4% (oats)	84.1% (salad)	-
01-220217-1435 (oats)	97.9% (oats)	94.6% (oats)	-
01-220217-1506 (salad)	55.6% (oats)	90.5% (salad)	-
01-220217-1541 (oats)	92.7% (oats)	80.1% (oats)	-
01-220217-1612 (salad)	58.4% (salad)	91.5% (salad)	-
01-220217-1707 (oats)	98.5% (oats)	94.8% (oats)	-
01-220217-1740 (oats)	94.3% (oats)	88.8% (oats)	-
01-220217-1813 (salad)	68.5% (oats)	85.0% (salad)	-
01-220217-1848 (salad)	65.4% (oats)	85.8% (salad)	-
01-220218-1424 (oats)	94.6% (oats)	93.3% (oats)	-
01-220218-1458 (salad)	73.5% (salad)	98.0% (salad)	-
01-220218-1542 (oats)	99.0% (oats)	96.1% (oats)	-
01-220218-1615 (salad)	54.9% (oats)	97.4% (salad)	-
Avg. 5 labeled recs.	70.2% \pm 38.0%	91.8% \pm 4.2%	96.9% \pm 2.2% (o=97.4%, s=96.1%)
Avg. 15 unlabeled recs.	69.7% \pm 27.6%	90.1% \pm 5.4%	-
Avg. all 20 recs.	69.8% \pm 29.43% (o=96.5%, s=43.2%)	90.5% \pm 5.2% (o=91.6%, s=89.3%)	-
Correct predictions	12/20	20/20	5/5
* Labeled recordings	Accuracy	Precision	Recall
5 CD labeled recordings	0.916	0.558	0.840
5 true labeled recordings	0.998	0.976	0.978

Table 7: Results from the Mel meal type classifier system test 4.9.3. Recordings marked with * have true labels available. For this test the network is trained on the 350ms segment length dataset (128x33).

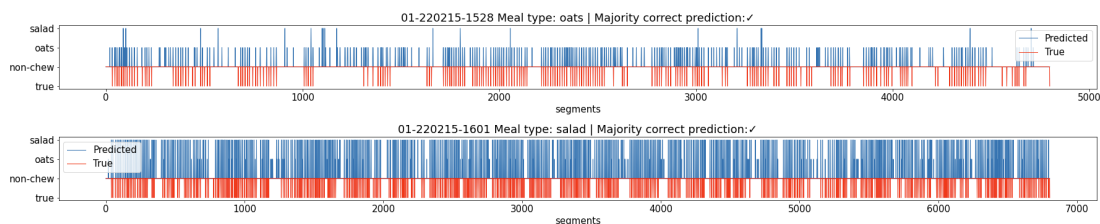


Figure 32: Predicted and true labels for two recordings: oats meal (top), salad meal (bottom). Predictions are made using MMT system with CD predicted chew labels. The blue labels indicate the predictions, which were either class “silence”, “salad” or “oats”. The true labels in red show two classes “silence” or “chew”. For the true labels all of the chews are of the meal type indicated by the title of the recording.

Table 7 shows the results obtained from the test in Section 4.9.3. For this test the dataset with 350ms segment length is used.

Recall the earlier statement about using the same recordings to create the training and testing dataset in Section 4.6.5. That claim can be verified with these results. The first 5 testing recordings, marked with *, were created by periodically segmenting the same recordings that were used to extract the individual chew segments for the training dataset. In most cases this is seen as a flawed way of testing, however the argument proposed here is that by segmenting the recording periodically, such as is the case for the testing dataset, it changes the individual segments enough for this to not be a concern. Comparing averages, 91.8% and 90.1%, for the 5 marked and 15 unmarked recordings respectively, the difference is 1.7% with 4% standard deviation. The marked testing recordings do perform slightly better, however due to the small difference it was concluded that the labeled recordings have practically no significant advantage in terms of performance. For transparency, the performance averages of all unmarked testing recordings are also provided.

Notice that the results for the first 5 recordings, as well as the 5 and 20 recording average prediction ratios are identical to results Table 6 for the CD system test. This is no coincidence, as the same best achieved CD system model was used to generate the chew labels for this MMT system test.

The the important metric for meal type classification is the prediction ratio. This metric gives the percentage of segments in the predictions that are predicted as either “oats” or “salad” class, and can be used for the final decision of meal type of the recording. Relying on this metric, the meal types in all 20 recordings are predicted correctly, with average prediction ratio of 90.5% and 5.2% standard deviation. This comes to avg. 91.6% for oats recordings, and 89.3% for salad recordings.

The second column contains the prediction ratios for the MMT system without applying any kind of chewing labels. This was included to show the necessity of appropriate chew detector, as without one the performance is considerably worse. For the 20 testing recordings, the oats recordings reach prediction average of 96.6% while salad recordings only 43.2%, leading to only 12/20 correctly classified recordings. As mentioned earlier, this is caused by the fact that all of the “silence” class segments are classified to the closest of the two meal type classes, which is “oats”.

The fourth column shows the prediction ratios for the first 5 recordings using the true chewing labels. This means that all of the MMT predictions are made only on segments that contain exactly a chew, which in theory allows to judge pure MMT performance. For the 5 tested recordings, the average prediction ratio was 96.9%, which is a 5.1% increase compared to using the CD predicted chew labels. This reveals two things: one, the pure MMT meal classification ability is very close to perfect, and two, the discrepancy between the prediction ratio for oats (97.4%) and salad (96.1%) is caused by the MMT. Additionally,

the accuracy, precision and recall metrics measure the meal type classification abilities for the 5 labeled recordings. The chewing segments can be selected using either the CD predictions or the true chewing labels. The results table includes these metrics for both. When using the true labels, all 3 metrics are very close to a perfect score of 1. The CD labels also have high, but slightly lower score for accuracy and recall. Precision however, is significantly lower for the CD predicted labels. This is likely correlated to the precision results seen for chewing detector test in Table 6.

Figure 32 shows the prediction and true labels for the 2 first recordings in the results table. Recall that for MMT system, the testing dataset was created using only the meal region part of the recording, thus in comparison to the analogous Figure 31 for the CD system, the silence parts aren't present here. Once again, this figure is included only to provide a visual representation of the predictions that the system makes.

5.4 Results - 3-class meal type classifier system test

Recording (ch-yymmdd-time)	Prediction ratio
01-220215-1528 (oats)*	94.5% (oats)
01-220215-1601 (salad)*	85.4% (salad)
01-220215-1647 (oats)*	94.7% (oats)
01-220215-1726 (salad)*	84.6% (salad)
01-220216-1238 (oats)*	95.3% (oats)
01-220216-1315 (salad)	85.0% (salad)
01-220216-1404 (oats)	81.3% (oats)
01-220216-1438 (salad)	85.1% (salad)
01-220217-1435 (oats)	97.1% (oats)
01-220217-1506 (salad)	87.3% (salad)
01-220217-1541 (oats)	83.0% (oats)
01-220217-1612 (salad)	92.3% (salad)
01-220217-1707 (oats)	92.0% (oats)
01-220217-1740 (oats)	90.2% (oats)
01-220217-1813 (salad)	91.4% (salad)
01-220217-1848 (salad)	86.0% (salad)
01-220218-1424 (oats)	91.6% (oats)
01-220218-1458 (salad)	97.8% (salad)
01-220218-1542 (oats)	96.1% (oats)
01-220218-1615 (salad)	94.9% (salad)
All 20 recordings	avg. 90.3% \pm 5.1% (o=91.6%, s=89.0%)
5 labeled recordings	avg. 90.9% \pm 5.4%
15 unlabeled recordings	avg. 90.1% \pm 5.2%
Correct predictions	20/20
Accuracy (5 labeled)	0.970
Precision (5 labeled)	0.556
Recall (5 labeled)	0.877

Table 8: Results from the 3-class meal type classifier system test 4.9.4. Recordings marked with * have true labels available. For this test the network is trained on the 350ms segment length dataset (128x33).



Figure 33: 3C system predicted and true labels for two recordings: oats meal (top), salad meal (bottom). The yellow labels indicate the predictions, which were either class “silence”, “salad” or “oats”. The true labels in red show two classes “silence” or “chew”. For the true labels all of the chews are of the meal type indicated by the title of the recording.

Concluding the system testing part, the results of the 3-class meal type classifier (3C) system test from Section 4.9.4, are presented in Table 8. For this test the dataset with 350ms segment length is used.

The performance is nearly identical to the MMT system. The 3C correctly classified all 20 testing recordings with average prediction ratio of 90.3% and standard deviation of 5.1%.

Once again, its possible to compare the performance between the recordings marked with * and those without. For the 3C system this difference is even smaller than for the MMT system.

The accuracy, precision and recall metrics also closely resemble the results from the MMT system. Precision is still lacking behind despite the system not relying on external chewing detector to remove silence segments.

Like for the previous two systems, the Figure 33 provides a visual representation of the predictions made by the 3C system. Like for the CD system, all testing dataset recordings capture only from the 15 minutes to the end of the original recording. This looks very similar to the CD system predictions in figure 31, which makes sense considering that the performance of the 3C system and the MMT system using CD chew labels is practically equal.

5.5 Training curves and testing split evaluation

This section includes the training loss and accuracy curves for the best achieved system models of the CD, MMT and 3C systems, as well as the testing split confusion matrices. These results are included to satiate the potential curiosity of the reader, and provide more transparency about the results.

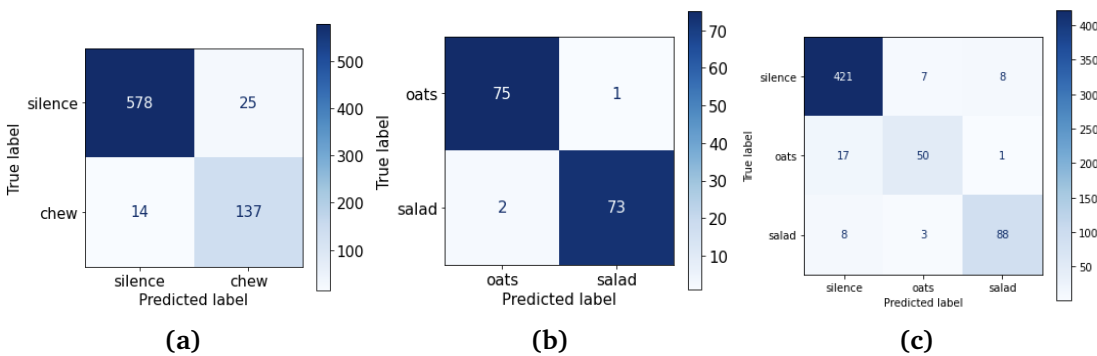


Figure 34: Confusion matrix for the testing split of the best achieved CD (a), MMT (b) and 3C (c) system model

Although the testing split implementation was covered in the methodology section, the actual use for it in the study was limited. The testing split was mostly used for quick and general evaluation of performance during earlier development of the neural network architecture. When it came to fine tuning

the system, it was found that testing split didn't reflect the performance of the testing dataset. In certain cases the performance would improve on testing split, where it would suffer for the testing dataset. Since optimizing for the realistic data scenario is the goal, from that point into the development only the testing dataset was used to assess the performance. The testing split confusion matrices for the CD, MMT and 3C systems are shown in the Figure 34.

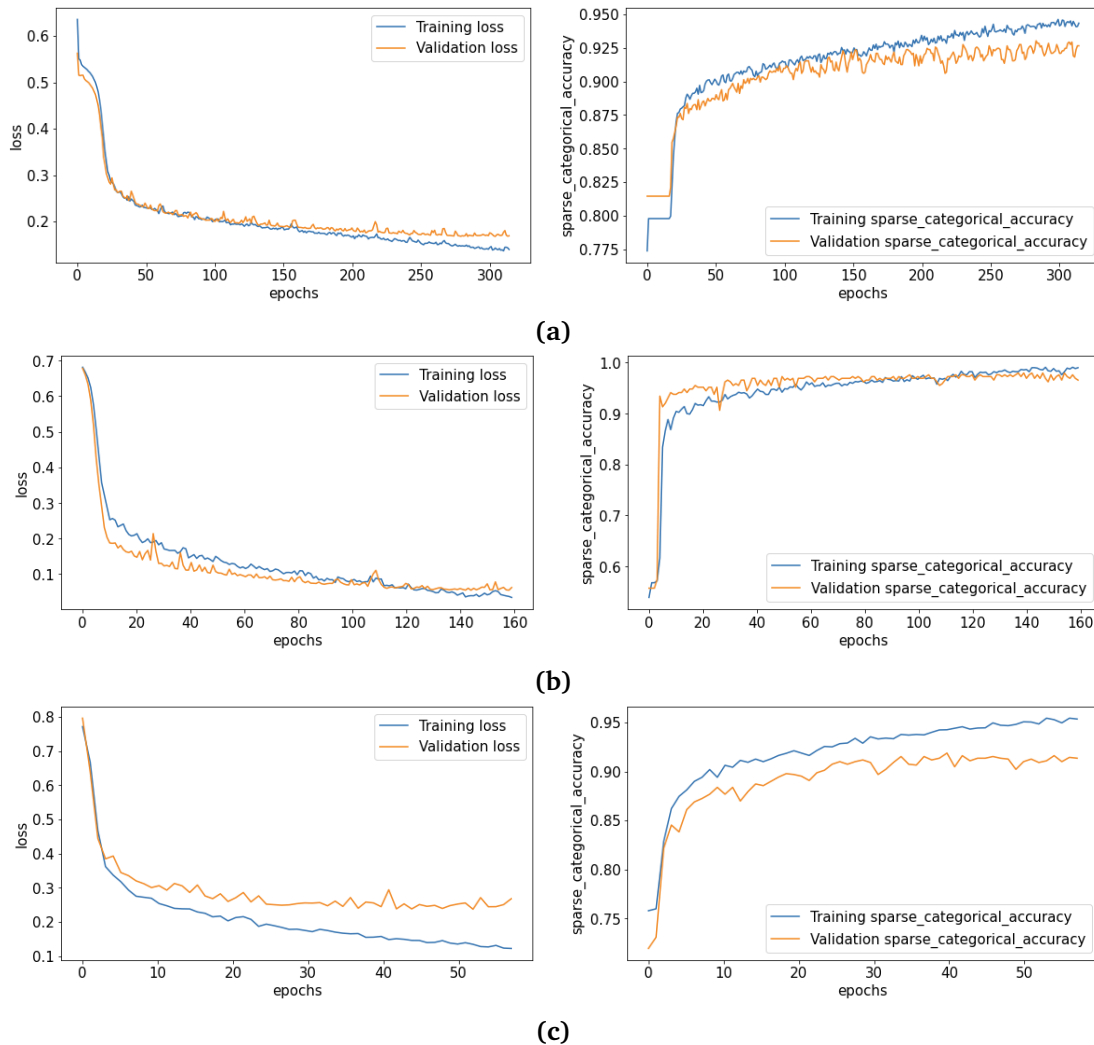


Figure 35: Training loss and accuracy curves of the best achieved CD (a), MMT (b) and 3C (c) system model

The training loss and accuracy curves for training and validation splits of the best achieved models for the 3 systems are given in Figure 35. As mentioned in the methodology section, the number of epochs is controlled dynamically by Early stopping. This terminates further training once the validation loss reaches a plateau, avoiding potential overfitting.

5.6 Results - Mel feature dataset test

Testing system	Control System	Testing Dataset	Run	No. correct recordings	Prediction Ratio (correct class)	Prediction Ratio Avg.
Chewing Detector (CD)	128x33 (MMT)	128x8 (100ms)	1	20/20	81.7%	83.2% ± 1.7%
			2	20/20	85.1%	
			3	20/20	82.8%	
	128x33 (MMT)	128x33 (350ms)	1	20/20	90.1%	88.8% ± 1.2%
			2	20/20	88.4%	
			3	20/20	87.8%	
	128x33 (MMT)	128x43 (450ms)	1	20/20	90.4%	87.9% ± 2.7%
			2	20/20	86.8%	
			3	20/20	89.8%	
	128x33 (MMT)	128x48 (500ms)	1	20/20	88.0%	87.9% ± 0.8%
			2	20/20	85.1%	
			3	20/20	89.3%	
Mel Meal Type Classifier (MMT)	128x33 (CD)	128x8 (100ms)	1	20/20	84.5%	84.2% ± 0.4%
			2	20/20	83.8%	
			3	20/20	84.2%	
	128x33 (CD)	128x33 (350ms)	1	20/20	85.9%	87.0% ± 1.2%
			2	20/20	87.0%	
			3	20/20	88.2%	
	128x33 (CD)	128x43 (450ms)	1	20/20	87.9%	87.6% ± 0.9%
			2	20/20	88.4%	
			3	20/20	86.6%	
	128x33 (CD)	128x48 (500ms)	1	20/20	87.7%	86.5% ± 1.1%
			2	20/20	85.6%	
			3	20/20	86.2%	
3-Class Meal Type Classifier (3C)	None	128x8 (100ms)	1	20/20	75.8%	76.9% ± 1.6%
			2	20/20	76.2%	
			3	20/20	78.7%	
	None	128x33 (350ms)	1	20/20	87.0%	86.7% ± 1.0%
			2	20/20	87.5%	
			3	20/20	85.5%	
	None	128x43 (450ms)	1	20/20	88.9%	89.2% ± 0.9%
			2	20/20	90.2%	
			3	20/20	88.6%	
	None	128x48 (500ms)	1	20/20	89.3%	87.8% ± 1.9%
			2	20/20	85.6%	
			3	20/20	88.4%	

Table 9: Results table for the Mel spectrogram dataset test 4.9.5

The table above summarizes the results from the Mel spectrogram dataset test in subsection 4.9.5. All 3 systems that use Mel spectrogram dataset were tested. The test evaluated the performance across the 4 different Mel spectrogram datasets from Tables 4 and 5. The differing factor being the segment length, this test would indicate which segment lengths perform better for this particular type of Mel spectrogram based systems. The metric for evaluation is prediction ratio.

For the chew detection system, the smallest, 100ms segment length dataset achieved average prediction ratio of 83.2%, with the best run reaching 85.1%. The datasets with segment lengths of 350, 450 and 500ms achieved prediction ratios of 88.8%, 87.9% and 87.9% respectively, and all performed considerably better than 100ms. When considering their standard variations, the prediction ratio difference between the 3 longest segment length datasets isn't very significant, however the 350ms dataset seems to perform the best.

Mel meal type classifier has very similar results. The 100ms dataset achieved prediction ratio of 84.2%, while the 350ms, 450ms and 500ms segment length datasets achieved 87.0%, 87.6% and 86.5%, respectively. The 3 longer segment length datasets once again outperform the smaller, 100ms dataset, by a considerable margin. Overall, the best performing dataset for the MMT system is the 450ms segment length dataset.

Finally, for the 3C dataset test, the results are very similar to the MMT system. The 100ms segment length dataset is once again the worst performing out of the 4. The 3 remaining datasets have similar performance in the 86-89% range. Between them, the best performance of 89.2% was achieved by the 450ms dataset. The 500ms dataset performed slightly worse, with 87.8%, however it managed to edge out the 350ms dataset, which achieved 86.7%.

Overall, the 450ms segment length works best for the MMT and 3C systems, while the CD system performs better with 350ms segment length.

6 Discussion

6.1 The PSD meal type classifier

The first approach for meal type classification was a fully connected neural network trained on PSD feature dataset. As observed in the results Section 5.1, the meal detection performance was very poor. In total, 5 different PSD feature configurations with varying segment lengths (0.45, 1, 10, 20, 30 and 60 seconds) were tested. All resulting predictions were plagued with a strong bias for predicting only one of the two meal type classes. The features with 1 sec. segment length showed slightly better performance, however these results were still unreliable and worse than would be acceptable. The reason for the slight performance improvement for features with 1 sec. segment length is unknown. It could be argued that, compared to the 10+ sec. segment length features, the 1 sec. segment length is closer to the length of a single chew which works better for meal type classification. This theory is contradicted by the fact that the even smaller, 0.45 second, segment length features performed identically to the 10+ second segment length features. Recall Figure 19, which indicates that the individual chews are around 350ms long. If capturing a chew more precisely helped the PSD meal type classifier, then the 0.45 second features would be expected to perform similar to the 1 second segments.

Although unintentionally, the PSD meal type classifier did work very well for meal onset detection. In all cases the system managed to correctly differentiate the meal region from silence, as evidenced by the Figures 29 and 30. This was the case for all the test runs. While this feature is needed for the grand scheme of realizing an artificial pancreas system, the same result was already achieved using SVMs in the previous work done by the author [Klavins, 2022].

6.1.1 Why PSD features don't work well for meal classification?

After seeing these results, it is then curious why the PSD features fail to provide sufficient information for meal type classification, when they work well enough for meal region detection. Two hypothesis are presented.

Firstly, the neural network is only as good as its training data. For PSD meal type classifier the training data consists of multiple whole-length recordings which are segmented periodically, and each segment consisting of multiple features. The collection of training segments capture everything that happens in the recording. Non-chewing events, such as silence, random noises and swallowing events are also present. The labeling which is used for training, marks the entire meal region as the specific meal type, meaning that all the non-chewing events there within are also labeled as that meal type. Human analysis of the recordings reveals that largely chewing and swallowing events are the only auditory cues which perceivably distinguish the meal types, with a strong

emphasis on chewing events. Putting all this together, its possible that the poor performance is influenced by the non-chewing events, which don't really contribute to differentiating between the meal types, but nevertheless have been marked as part of one or the other meal type class.

The second observation, which is more likely to contribute to the poor meal type classification performance, is the information that the PSD features capture. Observe Figure 36, which shows the comparison between PSD and Mel spectrogram features, as well as a reference spectrogram of the the chewing event in Audacity. Although the PSD segment in the figure isn't the same as the PSD based feature matrices (Fig. 16) used for training, it provides an insight in the type of information that the PSD based features are able to extract. The PSD spectrogram has little resemblance to the reference spectrogram. The same observation was made for other randomly selected segments of the recording. Sometimes the PSD features would show significant peaks where there is little auditory activity, and vice versa. This isn't to suggest that the PSD based features are useless. Rather, this indicates that for the system that which was used in this study, the PSD features do not capture the appropriate details, which can cause poor meal type classification performance. The Mel spectrogram captures the chewing events much more accurately.

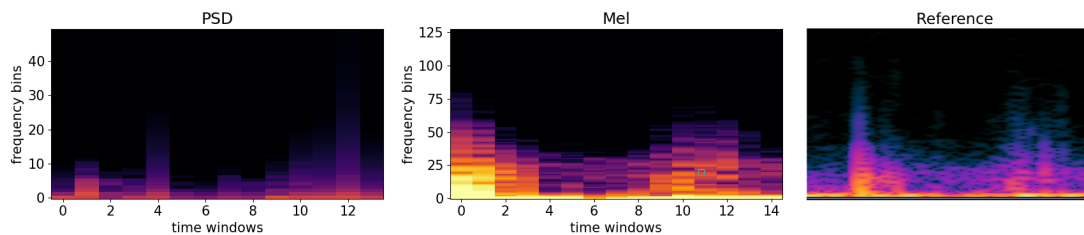


Figure 36: Comparison between PSD spectrogram, Mel spectrogram and Audacity reference of a single chew occurrence. All 3 windows capture the same length and region of the audio recording.

6.2 Chew detector

The CD system was designed to complement the MMT system. Unlike the PSD meal type classifier, both the CD and MMT are using Mel features. Not only is there a change from PSD features to Mel spectrogram, but also the way that the segments are extracted. Instead of using periodically selected segments, the training dataset consists of Mel spectrogram segments which only, and more or less precisely, capture a single chew occurrence.

Although the objective of the CD system is to assist the MMT system, and thus its performance is ultimately measured in conjunction with MMT, its individual objective is to as closely as possible resemble the true labels, which were created manually for 5 of the 20 recordings. To measure this ability the

accuracy, precision and recall metrics were created using the predicted and true labels for the 5 recordings. Judging by these metrics, the overall chew detection performance is good, with the exception of one issue, low precision. The accuracy and recall for the detected chews is relatively high (0.970 and 0.837 respectively), meaning that most of the actual, true labeled, chews are detected. However the precision is somewhat low (0.332), meaning that there are a lot of predicted chews which aren't actual chews.

6.2.1 Why is precision low?

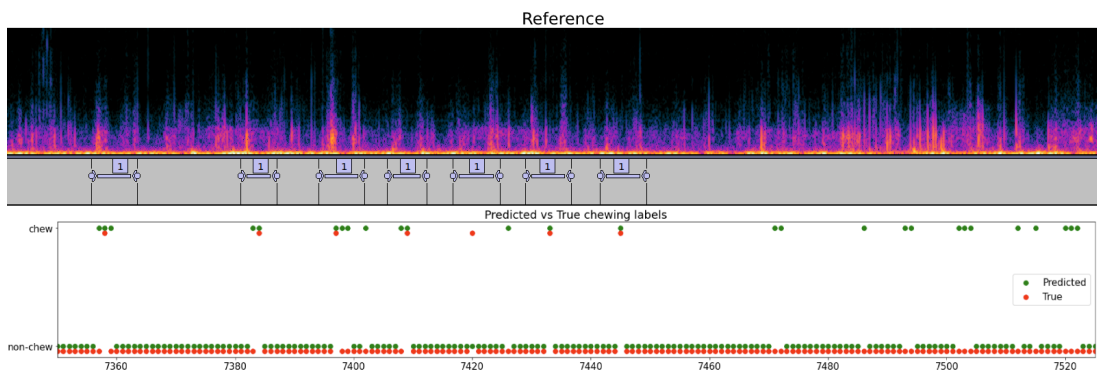


Figure 37: Section of the recording and true labels in Audacity (top), and a plot of CD predicted (green) labels and true (red) labels of the corresponding recording section (bottom)

Figure 37 shows a snippet of the predicted (green) and true (red) segment labels, as well as the corresponding spectrogram along with the true labels from Audacity. This figure captures the two aspects of the low precision problem.

Firstly, notice the left side of the plot. For many of the true labels there are multiple predicted labels scattered in close proximity. The likely cause for this phenomenon lies in structure of the testing dataset. Since the testing recordings are periodically segmented with a segment shift of 50ms, there are multiple segments which capture the same, roughly 350ms long chew. If this is the case, the CD system technically doesn't make a mistake, but since the true labels are only assigned for a single segment, this is seen as a fault by the metrics. As this is a technical issue, it could be fixed by creating true labels which mark multiple consecutive segments which include the same chew.

The second issue that's causing low precision is apparent on the right hand side of the plot. Notice how the region doesn't contain any true chews, yet the CD detects multiple segments all the same. By cross-referencing the region with the reference spectrogram, as well as listening to the recording, it turns out that the region contains only non-chewing sounds, such as swallowing and tongue

movements. This seems to be a quite considerable source of false-positive detections. A possible solution would be to introduce training segments which capture swallowing and miscellaneous sonic elements. It's uncertain how well this would work, considering that the training dataset already contains non-chewing segments extracted from within the meal region during the training dataset augmentation step in Section 4.6.4. This in all likelihood means that some of the miscellaneous sounds have already been included in training. Including these more deliberately and methodically could however potentially help reduce these false positives.

6.2.2 Improving the chewing detector

One of the issues that's been highlighted by the Figure 37 is the false positive prediction of non-chewing noises such as swallowing and miscellaneous mouth sounds. If this is already an issue for our very controlled recording protocol, it is likely that for a dataset which also includes environmental noise, the performance impact would be even more detrimental. As will be detailed later in Section 6.6.2, certain flaws were discovered in the data acquisition equipment which might have affected the overall performance results in this study, including chewing detection. All these factors suggest that relying on audio cues alone to detect chews might be unreliable.

Literature review section has listed multiple methods of dietary monitoring, specifically through detection of chews. One of the least intrusive approaches is using a piezoelectric sensor attached to the temporalis muscle, directly capturing the muscle flexion during mastication. Combining the current audio data with measurements from a piezoelectric sensor could very likely eliminate the problem of false positives. Each prediction label made by the Mel spectrogram data could be cross-referenced by flexion pattern detected by the sensor to ensure that the detections are indeed chews.

6.3 Mel meal type detector

Taking the MMT system results (Table 7) at face value, the performance is very good. In a real world application the meal type classifier would make the final decision based on the prediction ratio, which essentially picks the most common class among the predicted class labels. With this metric the system achieves correct classification for all 20 meals with a very good average prediction ratio of 90.5%. These results are achieved in combination with the predicted chew labels from the CD system.

When using the true chewing labels instead of the CD predictions, the influence of incorrectly predicted CD chew labels is eliminated from the MMT performance. For this configuration, as shown in column 4 of the results Table 7, the MMT achieves average prediction ratio of 96.9%, for the 5 labeled meals.

Note that this is the result for the 5 testing recordings which were also used to create the training dataset, but as discussed in the results section, there is a very small difference in performance between the 5 labeled and 15 unlabeled recordings. Therefore it seems appropriate to conclude that if the CD chewing labels were more accurate, all 20 recordings could see a similar average prediction ratio as for the 5 labeled recordings.

The prediction ratio results for the individual classes follow the same trend, both when true and CD predicted chewing labels are used. That is, oat recordings have a slightly higher prediction ratio than salad, regardless of whether the true or CD labels are used.

The accuracy, precision and recall metrics for the 5 labeled recordings (bottom of the table) confirm that the largest source of error for the MMT system is false positive CD chewing labels. All metrics achieve nearly perfect scores when the true labels are used, thus the primary focus for further improvement of meal type classification should be correct detection of chew segments by the CD system.

As a sidenote, it was already mentioned multiple times that the testing recordings contained only a part of the entire recording. For MMT system tests this was only the meal region. It is assumed that in a live testing scenario a valid meal region would be provided by a meal onset detection system, such as in [Klavins, 2022].

6.3.1 Why does the MMT perform slightly better for oats recordings?

This might be a nitpick considering that the majority of the errors in MMT system are caused by incorrectly selected chewing segments, however as results have shown, there remains a slight performance advantage for recordings with oat meals. The influence of CD chew labels for this issue has also been ruled out by using true labels, which means that this is an inherent issue of the MMT system.

While its not certain as to what causes this slight bias, here are a few facts which may help to find an answer. The initial suspicion was that this was caused by class imbalance. This however was quickly dispelled by comparing the ratios of the two classes in the training dataset: 1302 oats segments, and 1712 salad segments. Considering that more training examples should lead to a better performance, this fact seems to contradict the observations where oats meals are predicted slightly better. Also, this discrepancy shouldn't be caused by influence of non-chewing features during training, as the training dataset consists solely of segments which, more or less exactly, capture only a single chew occurrence.

The more likely explanation is provided when examining the nature of mastication. The principal mechanism that allows the MMT system to classify meal types is the frequency content of the spectrogram. The idea for choosing salad and oats as the food types was that their characteristic sound is different. Oats

sound more muffled and bassy, while salad is crispier and louder. However, what happens once the salad gets chewed, or perhaps is stale and no longer crispy? In that case the salad will be less distinguishable from the oats meal. This was also observed when listening to the recording while creating the manual labels. The intensity and sound characteristic of salad chews becomes more similar to oats as the test subject proceeds to chew the salad. This explanation also conforms with the prediction labels shown in Figure 32. Notice that groupings of predicted labels are followed by slightly larger gaps. These gaps correspond to the test subject swallowing the food and taking another mouthful. Each grouping of frequent predictions is then a period where test subject masticates the mouthful. Also notice that for the salad meal, the last few predicted labels of each grouping tend to be predicted as oats, which agrees with the theory that mastication of food will over time change the sound characteristic of food, leading to false predictions.

Recall from Section 3.2.2 that the salad meal also included some pasta, which was consumed together with salad. Well prepared pasta, as was the case, isn't crispy like salad and produces sound characteristics more similar to oats meal, which could also cause the few salad segments to be predicted as oats.

6.3.2 Possible improvements for meal type classification

Although the MMT system performed very well when provided with accurate chewing labels, the system has not yet reached its full potential. What we've established is that for two, quite distinct meals, the performance is very good. The next logical step is challenging the system by introducing larger variety of meals, and perhaps meals which have similar sound characteristics. By adding more meal types the classifier is very likely to decrease in performance. As the current system stands, the Mel spectrogram based training dataset is enough, however for multiple meal types this might become the limiting factor. Here are some ideas which might need to be considered for when the system is expanded.

Firstly, a possible differentiating property which could be exploited is the difference in chew occurrence frequency between the meal types. While creating manual labels it was observed that oat meals had considerably fewer chewing occurrences than salad meals. This isn't due to oat meals being shorter and therefore having fewer chews, as all recordings had nearly equivalent meal durations. Each oats meal had roughly 250 chewing labels while salad had double. Somehow adding this property into training dataset could potentially help distinguish between meal types.

After the recordings were labeled some analysis was performed on the statistics of the labels (figure 19). This analysis reveals another property which could potentially be used in meal type classification, the chew length difference. As evidenced from the figure, salad meals tend to have shorter chewing occur-

rences than salad. While this difference is small, and perhaps statistically not significant enough for these specific food types, it might be more significant once larger food variety is introduced. A challenge with implementing both suggestions is that they require a reliable chew detector.

6.4 3-Class meal type classifier

Finally, lets review the 3-class meal type classifier. As it has been mentioned many times, this system essentially combines the CD and MMT systems, and judging from the results in Table 8, the performance is also practically the same. The average prediction ratio for all 20 testing recordings is 90.3%, which is only 0.02% different than the MMT system and isn't statistically significant. It also exhibits the same slight class bias towards oat meals, with 91.6% prediction ratio for oats meals and 89.0% for salad. This behaviour has the same explanation as for the MMT. This begs the question, why even have a 3-class classifier? Was it all worth the effort?

6.4.1 Comparing 3-class classifier to the CD + MMT combination

When paying attention purely to the results, both methods for meal type classification gave essentially the same results. Not only they used the same training and testing datasets, but also the same network architecture turned out to work best for both. While this wasn't expected initially, it probably should have. In reality, the 3C system was created at a point in the study when the CD+MMT systems didn't work. Although those problems were largely centered around technical implementation, and are now solved, the 3C system persists.

Now lets look at the differences and advantages each approach has. For one, the 3C system has clear advantage in a practical implementation. This should come as no surprise as combining two separate systems not only takes more software resources, but also separate configuration for each system, which creates more complexity for the same result.

Advantage that two separate systems have is configurability. While for a simple two class meal type classifier the simple models are enough, in a more realistic application scenario, with many different food types and environmental noise in datasets, a much more complex system would likely be required. It is therefore not certain that the same network architecture for both CD and MMT would suffice. More likely than not, the architecture would need to be more closely tailored to their individual tasks. This almost certainly becomes the case if additional and different dataset features are introduced. In that sense, perfecting two individual systems, and combining them only once they are complete, is an easier task than perfecting a single system.

6.5 Meeting the goals for the study

The primary goals for this study were outlined in Section 1.1.1. Performance requirements in relation to application in an artificial pancreas system were briefly discussed, where it was concluded that the focus in this study should be on correctly classifying the meal type of all 20 testing recordings while aiming for the highest possible prediction confidence.

This study implemented two main meal type classification systems, and both performed nearly equally. Both managed to correctly identify the meal type of all 20 testing recordings, with average prediction ratio of $\approx 90\%$. In terms of the goals that were set prior to the study, this is a very good result. As noted in the aforementioned section, the acceptable prediction confidence, in this case measured by prediction ratio, would need to be assessed by doctors or experts in the field. This value would need to be evaluated in conjunction with the artificial pancreas (AP) as a whole. Perhaps the final AP implementation will feature a different security/safety mechanism for ensuring reliability and accuracy of predictions. Either way, for the simple case of two meal types, this system delivered average prediction confidence of 90.5%, which is close to what author guesses would be determined to be acceptable for application in AP system.

It should also be mentioned that the current meal type classification implementation is very simplistic and still considerable ways off from what would likely be required for a finalized AP system. It's therefore difficult to assess performance requirements in that context.

6.6 Datasets, data acquisition and labeling

In this section we'll analyze the choices of parameter values for creating the Mel spectrogram datasets, as well as cover some problems that were discovered during data acquisition.

6.6.1 Mel spectrogram dataset

Segment length The general conclusion from the Mel dataset test results in Table 9, is that longer segment length features work generally better. Initially the 100ms segment length dataset (128x8) was used only for CD system. The idea was that having a short segment length of 100ms would capture only the initial down-bite peak of the chewing occurrences, which is usually the strongest portion of the chew and shows up best on the spectrogram, thus working better for chew detection than longer segments. In reality, results show that, although the performance with the shorter segment length was decent, it was exceeded by the segments with length 350ms, 450ms and 500ms. Comparing these numbers to Figure 19, its apparent that these segment lengths allow to capture nearly the entire chew. This was also the intention for the three longer

segment length datasets, as the idea was for the MMT and 3C neural networks to learn the spectrogram patterns of the entire chew and use them to separate the two meal types. However as it turned out, the longer segment length datasets were superior also for chew detection.

Between the 3 longer segment length datasets (350, 450 and 500ms), the best were the 350 and 450ms. Again, basing off of chew length analysis in Figure 19, the two segment lengths best capture the individual chews. During labeling, it was observed that in certain instances the true chew labels were less than 400ms apart from one and other. Thus it's likely that for the 500ms segments, more than a single chew was captured, negatively impacting the performance. This leads to the conclusion that for the Mel spectrogram systems that were implemented in this study, the best segment lengths to use are 350ms and 450ms.

Segment shift Segment shift is a parameter only present in the testing dataset. It decides the periodicity of the overlapping segments. For all 4 datasets a shift of 50ms was used. Although not documented in the results, early tests quickly showed problems with larger shifts. The main issue being that larger shifts make too large jumps, meaning that capturing a good segment of a chewing occurrence is less likely. Larger shift values cause more segments to contain partial chews, which make classification process more difficult.

The short segment shift doesn't come without downsides. The first obvious downside is increased file size, as more segments are required to cover the chosen length of the recording. Second downside is indirectly mentioned earlier in discussion Section 6.2, where the the issue of low precision is covered. There it was mentioned that each true chewing label is surrounded by a small group of consecutively predicted chewing labels. The likely reason for this is that a small segment shift was used. As the shift gets smaller, more consecutive segments capture the same chew, only slightly shifted with respect to one and other. This isn't a big issue as the possible workaround suggested adapting the true labels to also include the consecutive segments which capture the same chew.

In conclusion, for optimal performance on the methods used in this study, it's best to use shorter shift, with the drawback of larger data storage requirements and slightly more complicated implementation to avoid incorrect accuracy, precision and recall metrics.

Mel window length and shift The two Mel spectrogram parameters which are present for all Mel spectrogram datasets are the Mel window length and Mel shift. During the early stages of Mel-based system implementation, different values for these parameters were tried. A conclusion was reached that more detailed spectrograms yield better results. This meant that primarily Mel window length had to be as small as possible. Too small values however, would

produce numerical errors for spectrogram. Thus a suitable value of 30ms, which was close to the limit, was chosen for all the datasets. The Mel window length could be even further reduced, however this would come at the cost of reduced frequency bins, which arguably contain more important sound characteristic information. How much the number of frequency bins affects the performance remains to be tested.

The Mel window shift could go much lower, however was set to a value of 10ms. Any lower values would only smear/stretch the spectrogram and not provide any more details. Additionally the spectrograms would be larger storage-wise, which was already a problem for computers with lower RAM capacities.

6.6.2 Data acquisition and it's issues

The neural network can only be as good as its input data. Although thus far it has not been mentioned, the quality of audio recordings has been a considerable sticking point during the development of the study. The issue that was discovered early on, was poor sound quality in the recordings. After some testing it was confirmed to be caused by the microphone cup housing, shown in Figure 13. The housing is attached to the patient by a double sided tape ring which effectively creates isolated pressure area inside the microphone cup. The microphones were designed for previous APT studies to capture bowel sounds by placement on the stomach region, and the pressure isolation helps pick up the low amplitude bowel sounds. This also causes a low-pass filtering and resonance effect on the audio, which works well for capturing the low-frequency (up to 2kHz) bowel sounds. However for the purpose of capturing chewing sounds, which as denoted in theory section may reach 5kHz, the setup causes loss of important high-frequency information.

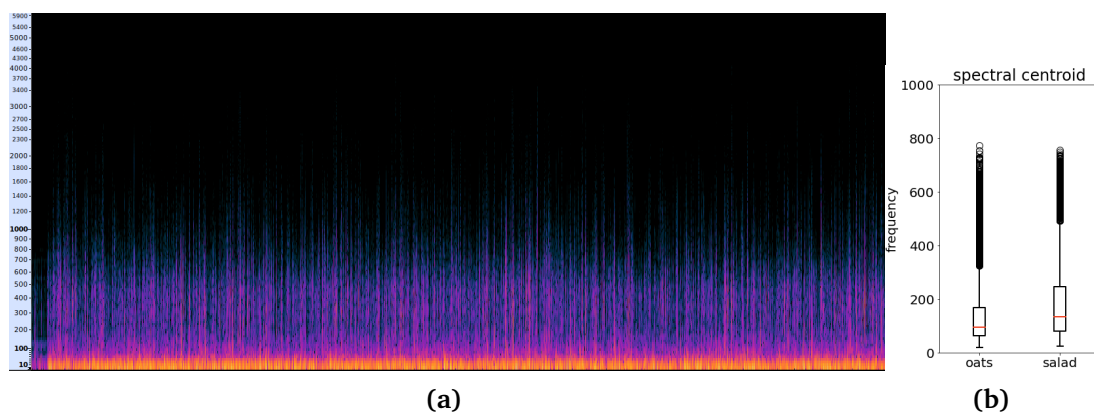


Figure 38: Excerpt of non-pre-processed salad meal recording spectrogram in Audacity (a). Spectral centroid of all manual chew label segments (also on non-pre-processed recordings) (b)

Figure 38 shows analysis that was performed on the raw audio recordings. The spectrogram clearly shows most of the sonic information being confined below 1kHz, with faint peaks up to 2kHz. The figure on the right shows the spectral centroid calculated from all manually labeled chewing segments. The centroid essentially highlights the center of mass of frequencies in the signal. These are clearly in the sub 1000Hz range. By manually listening to the recordings its clear that high-frequency information has been cut, by what sounds like a muffling effect, caused by the microphone setup.

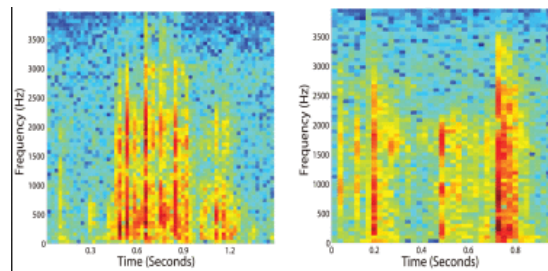


Figure 39: Chewing spectrogram of a cookie (left) and a carrot (right) from the [Bi u. a., 2016]

For comparison, the spectrograms in Figure 39 are taken from a different meal classification study ([Bi u. a., 2016]), and show spectrograms of chews for two different foods (cookie and carrot). Unfortunately it might be hard to see due to poor image quality, but the frequency axis indicates that the chewing sounds display clear and strong activity all the way to 2kHz. The recording setup for this study used two direct skin contact microphones on both sides of the neck/throat. *“The throat microphone converts vibration signals from the skin surface to acoustic signals rather than picking up sound wave pressure as most common microphones do.”*

Unfortunately it wasn’t possible to change the recording setup for our study to something that works better for capturing chewing sounds. Although the systems worked very well with the data they were given, no reason comes to mind why using a better setup which captures higher frequency sounds wouldn’t lead to even better classification performance. This would be even more important for future tests classifying larger variety of foods, where the distinction would likely be in the small high frequency details. Participants of this study have not investigated possible replacements for the current microphones/equipment, thus no suggestions are made on that part. This decision should be influenced by the opinion of an audio/acoustics and physiology expert.

6.6.3 Ideas for dataset improvement

A few things which also relate to dataset improvement were already mentioned in Section 6.3.2, such as adding chew occurrence frequency and chew length

metrics.

Moreover, due to periodicity of the segments in the Mel spectrogram testing datasets, the chew occurrences are captured multiple times, where the same chew is contained in consecutive segments, with only a slight shift in the spectrogram. For the neural network to classify these slightly shifted and/or partial chews, it requires the ability to identify the chew elements in a spatially invariant way. Although pooling layers already help the network reduce spatial variance by downsampling the images, another common practice is to augment the training dataset with distorted versions of existing samples, adding more data diversity and helping achieve spatial invariance. One way to add variety for this type of study could be to shift the chew segment forwards or backwards in time by a small amount. This way, instead of one segment, each manual chewing label will produce possibly 3 segments: backwards shifted, forwards shifted and no shift. The shift would need to be small enough to not overlap with other chewing labels potentially accidentally including another chewing occurrence.

6.6.4 Choice of recording labeling

It feels necessary to briefly comment on the choices that were involved for labeling the recordings. Only 5 of the 20 recordings were manually labeled, thus the entire system was trained on data from only 5 meals. Initially only 2 recordings of each meal type were chosen, however in order to somewhat balance the datasets with roughly equal amounts of oats and salad class segments, a third oats recording was labeled. This is a very laborious process, thus initially the focus was to get the system to work with smaller amount of samples, and create further labels if performance was lacking. This never came to be as the system performed very well with training segments from only 5 recordings. At this point, labeling more recordings would likely yield diminishing returns for performance.

7 Conclusion

This study was initiated to investigate and provide an implementation of a meal type classifier. The initial approach with PSD features, was based on its success for meal onset detection in the previous work by author. This implementation utilized a 3 layer neural network trained on PSD-based features from periodically segmented audio recordings, however it was unsuccessful. The resulting system inadvertently excelled at meal onset detection, however the meal type classification performance was very unreliable and biased towards one class. The tests found the different PSD feature segment lengths to be inconsequential, and that for this specific system implementation the PSD features are inappropriate.

The subsequent approach involved adopting a whole different type of features based on Mel spectrogram, as well as altering the training method by training on individually segmented chewing occurrences created from manually labeled recordings. This approach spawned 3 different systems: chew detector (CD), Mel meal type classifier (MMT) and the 3-class meal type classifier (3C), the latter of which combines the first two. Although initially the models started out with different network architectures and training parameters, the final implementations of all 3 systems were very similar.

Testing was performed using Mel spectrograms extracted from segments of periodically segmented recordings. The CD and MMT systems are designed to operate together, and were tested as such, achieving average prediction ratio of 90.5% and correct meal type prediction for all 20 testing recordings. Additionally the MMT system was tested with the true, manually created, chewing labels for the 5 labeled recordings, and compared to the performance of CD predicted chewing labels. The true labels achieved avg. prediction ratio of 96.9% compared to 91.8% by the CD chewing labels. This discrepancy was caused by the relatively poorer precision score of the CD system. The test highlights the importance of using a good chewing detector in conjunction with the MMT system. The 3C system achieved very similar performance to the CD+MMT approach, both in terms of avg. prediction ratio, which was 90.3%, as well as the accuracy, precision and recall metrics.

Finally, different configurations of the Mel spectrogram dataset were tested. This test involved comparing the avg. prediction ratios of datasets created with 4 different segment lengths: 100ms, 350ms, 450ms and 500ms. Although the CD and MMT were tested separately, both of their tests indicate that the two longer segment lengths were superior to the shorter. The 3C system also confirms the shorter segment length of being a poorer performer. In conclusion, this test indicates that, at least for the approach in this study, the 350ms and 450ms segment length Mel spectrograms perform best.

8 Future work

Many things have already been mentioned in the discussion section suggesting possible avenues for future endeavours. In this section these ideas are more structurally composed.

- Augmenting the chew detection system with data from a piezoelectric sensor on the temporalis muscle (Section 6.2.2).
- Extending and challenging present meal type classification systems by introducing larger variety of food types and environmental, everyday noises in the recordings (Section 6.3.2).
- Introducing additional features in the dataset which could help improve classification. As previously mentioned, chew occurrence frequency and segment length are two possibilities (Section 6.3.2).
- Finding a better method of capturing chewing sounds. The method/equipment used in this study was likely limiting the performance (Section 6.6.2).
- Augment the dataset with distorted training samples to improve the spatial invariance of the neural network model (Section 6.6.3).

9 Bibliography

- [BareBra] BAREBRA: *Supergrot*. <https://www.barebra.no/products/supergrot-kanel-puffet-quinoa/>
- [Basta] BASTA, Nikola: *The Differences between Sigmoid and Softmax Activation Functions*. <https://medium.com/arteos-ai/the-differences-between-sigmoid-and-softmax-activation-function-12adee8cf322>
- [Bi u. a. 2016] BI, Yin ; LV, Mingsong ; SONG, Chen ; XU, Wenyao ; GUAN, Nan ; YI, Wang: *AutoDietary: A Wearable Acoustic Sensor System for Food Intake Recognition in Daily Life*. In: *IEEE Sensors Journal* 16 (2016), Nr. 3, S. 806–816. <http://dx.doi.org/10.1109/JSEN.2015.2469095>. – DOI 10.1109/JSEN.2015.2469095
- [Bliksvær 2021] BLIKSVÆR, Viljar G.: *Sensor fusion of sound and ECG for improved meal detection (Available at Appendix B)*. Trondheim, Norway, Norwegian University of Science and Technology, Diplomarbeit, 2021
- [Brownlee a] BROWNLEE, Jason: *A Gentle Introduction to Cross-Entropy for Machine Learning*. <https://machinelearningmastery.com/cross-entropy-for-machine-learning/>
- [Brownlee b] BROWNLEE, Jason: *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>
- [Cordeiro u. a. 2021] CORDEIRO, Renato ; KARIMIAN, Nima ; PARK, Younghee: *Hyperglycemia Identification Using ECG in Deep Learning Era*. In: *Sensors* 21 (2021), Nr. 18, S. 6263
- [Dacremont 1995] DACREMONT, C: *Spectral composition of eating sounds generated by crispy, crunchy and crackly foods*. In: *Journal of texture studies* 26 (1995), Nr. 1, S. 27–43
- [Guesmi u. a. 2018] GUESMI, Latifa ; FATHALLAH, Habib ; MENIF, Mourad: *Modulation format recognition using artificial neural networks for the next generation optical networks*. In: *Advanced Applications for Artificial Neural Networks*. IntechOpen, 2018, S. 11
- [Hoffman] HOFFMAN, Ken: *Machine Learning: How to Prevent Overfitting*. <https://medium.com/swlh/machine-learning-how-to-prevent-overfitting-fdf759cc00a9>

- [Klavins 2022] KLAVINS, Davis: *Combining machine learning and acoustics for early meal detection (Available at Appendix B)*. Trondheim, Norway, Norwegian University of Science and Technology, Diplomarbeit, 2022
- [Kölle 2019] KÖLLE, Konstanze: *Feasibility of early meal detection based on abdominal sound (Available at Appendix B)*. Trondheim, Norway, Norwegian University of Science and Technology, Diplomarbeit, 2019
- [Lendave] LENDAVE, Vijaysinh: *What Is A Convolutional Layer*. <https://analyticsindiamag.com/what-is-a-convolutional-layer/>
- [Matinfo] MATINFO: *Chicken Salad*. <https://produkter.matinfo.no/BAMA-INDUSTRI-AS/Kyllingsalat/07023026063111>
- [Nvidia] NVIDIA: *Artificial Neural Network*. <https://developer.nvidia.com/discover/artificial-neural-network>
- [PapersWithCode] PAPERSWITHCODE: *Max Pooling explained*. <https://paperswithcode.com/method/max-pooling>
- [Porumb u. a. 2020] PORUMB, Mihaela ; STRANGES, Saverio ; PESCAPÈ, Antonio ; PECCHIA, Leandro: Precision medicine and artificial intelligence: a pilot study on deep learning for hypoglycemic events detection based on ECG. In: *Scientific reports* 10 (2020), Nr. 1, S. 1–16
- [Roberts] ROBERTS, Leland: *Understanding the Mel Spectrogram*. <https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53>
- [Saha] SAHA, Sumit: *A Comprehensive Guide to Convolutional Neural Networks*. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [Scanlon] SCANLON, Maria: *Semantic Annotation of Aerial Images using Deep Learning, Transfer Learning, and Synthetic Training Data*. <https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1>
- [Selamat u. Ali 2020] SELAMAT, Nur A. ; ALI, Sawal Hamid M.: Automatic Food Intake Monitoring Based on Chewing Activity: A Survey. In: *IEEE Access* 8 (2020), S. 48846–48869. <http://dx.doi.org/10.1109/ACCESS.2020.2978260>. – DOI 10.1109/ACCESS.2020.2978260
- [Sharma] SHARMA, Avinash: *Understanding Activation Functions in Neural Networks*. <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>

- [SkLearn] SKLEARN: *SkLearn feature selection: selectkbest*. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html
- [SPM0687LR5H-1] *Datasheet of Knowle's SPM0687LR5H-1 microphone*. https://media.digikey.com/pdf/Data%20Sheets/Knowles%20Acoustics%20PDFs/SPM0687LR5H-1_DS.pdf
- [Sultan] SULTAN, Hossam H.: *Multi-Classification of Brain Tumor Images Using Deep Neural Network*. https://www.researchgate.net/publication/333411007_Multi-Classification_of_Brain_Tumor_Images_Using_Deep_Neural_Network
- [Wang u. a. 2022a] WANG, Ning ; TESTA, Alison ; MARSHALL, Barry J.: Development of a bowel sound detector adapted to demonstrate the effect of food intake. In: *BioMedical Engineering OnLine* 21 (2022), Nr. 1, S. 1–12
- [Wang u. a. 2022b] WANG, Ning ; TESTA, Alison ; MARSHALL, Barry J.: Development of a bowel sound detector adapted to demonstrate the effect of food intake. In: *BioMedical Engineering OnLine* 21 (2022), Nr. 1, S. 1–12

Appendix A Dataset metadata: Information about each recording

filename	ch	date (yymmdd)	time	meal start (min)	meal end (min)	type	subject	recLen (ms)
01-220215-1528	1	220215	1528	20	24	soft	1	1799986.25
01-220215-1601	1	220215	1601	20	25.67	hard	1	1861293
01-220215-1647	1	220215	1647	20	25.5	soft	2	1800290.25
01-220215-1726	1	220215	1726	20	25	hard	2	1820855.625
01-220216-1238	1	220216	1238	20	25	soft	1	1816157
01-220216-1315	1	220216	1315	20	25.4	hard	1	1848151.625
01-220216-1404	1	220216	1404	20	25	soft	2	1800957
01-220216-1438	1	220216	1438	20	25.37	hard	2	1833959.625
01-220217-1435	1	220217	1435	20.1	25	soft	1	1822514.25
01-220217-1506	1	220217	1506	20	25	hard	1	1801895.625
01-220217-1541	1	220217	1541	20	25	soft	1	1801618.25
01-220217-1612	1	220217	1612	23.5	28.25	hard	1	2102269
01-220217-1707	1	220217	1707	20	25	soft	2	1803415.625
01-220217-1740	1	220217	1740	20	25	soft	2	1848498.25
01-220217-1813	1	220217	1813	20	25	hard	2	1891693
01-220217-1848	1	220217	1848	20	25	hard	2	1806605
01-220218-1424	1	220218	1424	20	25	soft	2	1800781
01-220218-1458	1	220218	1458	20.5	26	hard	2	1863581
01-220218-1542	1	220218	1542	20	25	soft	1	1820541
01-220218-1615	1	220218	1615	20	25	hard	1	1818023.625

Table 10: Information about the recordings which were collected with the data acquisition methods described in section 3. Meal type hard=salad and soft=oats.

Appendix B Source code & Cited studies

For those with access to the thesis accessories, there will be an included zip file containing all the source code of the system implementations in this study. This zip file doesn't contain the datasets or the raw recordings as the file sizes exceed 100GB. The raw recordings can be obtained from the APT. The datasets can be created from the raw-recordings using the code included in the "Pre-processing" folder.

Zip file contents (Folders):

- Audacity Files:
 - Contains the audacity files with manual labels (.aup3). These labels have been exported to the .txt files
- FoodTypeDetection:

- Contains Python Notebook (.ipynb) files, with the implementations of the 4 systems discussed in this study
- Pre-Processing:
 - Contains Python and Python Notebook files for pre-processing the raw audio recordings as well as performing the feature extraction
- Saves:
 - Contains the saved best achieved models of the 3 Mel spectrogram systems.
- Studies:
 - Copies of the studies [Kölle, 2019], [Klavins, 2022] and [Bliksvær, 2021], which are possibly not available online.

