UNIVERSITY OF BERGEN

Department of Information Science and Media Studies

MASTER THESIS

A multimodal approach for event detection from lifelogs

A study on eating moments detection

Author:
Espen Holst WIik

Supervisor:
Duc Tien Dang Nguyen

June 15, 2020

Abstract

This paper analyzes how personal lifelog data which contains biometric, visual, activity data, can be leveraged to detect points in time where the individual is partaking in an eating activity. To answer this question, three artificial neural network models were introduced. Firstly, a image object detection model trained to detect eating related objects using the YOLO framework. Secondly, a feed-forward neural network (FANN) and a Long-Short-Term-Memory (LSTM) neural network model which attempts to detect ‘eating moments’ in the lifelog data. The results show promise, with F1-score and AUC score of 0.489 and 0.796 for the FANN model, and F1-score of 0.74 and AUC score of 0.835 respectively. However, there are clear rooms for improvement on all models. The models and methods introduced can help individuals monitor their nutrition habits so they are empowered to make healthy lifestyle decisions. Additionally, several methods for streamlining event detection in lifelog data are introduced.

Keywords: ***AI, Deep learning, Food Detection, Lifelogging, Event annotatio, Image Detection, Neural Network***

Acknowledgements

I would like to thank my supervisor Tien for his time and effort into the completion of this master thesis. I would also like to extend my thanks to my fellow students at room 635 for all the help and morla support they provided. In addition, I would like to thank my parents for allowing me to use their home as an home office during the quarentine of 2020.

Table of Contents

List of Figures

List of Graphs and Tables

Tables:

Graphs:

Chapter 1

Introduction

1.1 Introduction

The World Health Organization (WHO) [7] states that “Obesity is one of today’s most blatantly visible - yet most neglected public health problem” and calls obesity a “global pandemic. Obesity has tripled since 1975 and now causes more deaths than undernutrition. It is estimated that over 500 million people are now classified as obese. Not only does obesity affect the global health but also has major economic ramifications [8]. However, obesity is preventable by better diet and exercise regimes [7].

Lifelogging as defined by Dodge and Kitchin [1] is “*a form of pervasive computing, consisting of a unified digital record of the totality of an individual’s experiences, captured multi-modally through digital sensors and stored permanently as a personal multimedia archive.*” Lifelogging can be used to achieve a more healthy lifestyle and reduce change of obesity and the complications that come it. By leveraging the data gathered and stored in lifelogs, historical nutrition data can be presented to the lifelogger and thus, empowering the individual to make healthier decisions. For this to be achieved, the nutrition related data has to be extracted from the rest of the data contained within a lifelog. Deep learning artificial neural networks (ANN) have the ability to detect ‘eating moment’ events [6], given the necessary data to do so. Object detection ANNs can be used to gather such data from images contained in lifelogs to provide the ‘eating moment’ detection models with valuable information to make predictions on. The proposed models and methods in this thesis contribute towards that goal, with the end result of an Long-Short-Term-Memory (LSTM) model with an F1-score of 0.65.

The phenomenon of lifelogging is rising due to the increased accessibility of sensors and wearable technology, mainly through smartphones and fitness trackers e.g. FitBit. Lifelogging has potential to assist people in their everyday lives by inferring or mining knowledge from the data gathered. That means that the potential audience for the methods and technologies applied in this thesis is increase. However, lifelogging is a personal activity and as a result so is the data [4]. Thus, the results achieved with the models in this thesis might not be scalable to other liferloggers. Therefore, this thesis will in addition to the methods and models introduced, use the experience gained from using them to provide suggestions on streamlining event detection in lifelogs.

1.2 Research Questions and Goals:

This thesis is the documentation of an attempt to provide an annotation tool for lifelog data related to historical nutrition consumption of individuals without the use of data gathered by nonintrusive tools. gathering methods. As aforementioned, mining/inference knowledge from a lifelog is challenging for a number for reasons. The proposed solution is also therefore an attempt at addressing those challenges which reveals themselves during the development process and this is reflected in the established research questions

1.2.1 Research Questions:

In this section, the research questions that this thesis attempts to answer will be outlined and discussed.

1. *How can a combination of lifelog data (visual, biometric, semantic) and deep learning classification models be leveraged to provide an individual with historical nutrition data?*
   To answer this research question, in this thesis there will be used several neural network models and other machine learning techniques in an attempt to predict where in a lifelog dataset an ‘eating moment’ is occurring

2. *What types data and technologies should be prioritized for improved streamlining of event detection in lifelogs?*
   This question will be answered through knowledge and experience gained from solving the challenges faced when applying methods and technologies used in the attempt to detect the ‘eating moments’ from research question 1.

1.2.2 Goals

- Create an ‘eating detection’ model with results at a sufficient level
- Provide value to the lifelog dataset used in this thesis
- Introduce methods and models to help other lifeloggers create similar or better models for themselves

- Contribute to the lifelogging scientific field, more specificly to event detection

1.3 Eating moment definition

Working definition of an ‘eating moment’ for the detection task is:
“Any point in time where the lifelogger is engaged in the activity consuming any drink or food.”

1.4 Thesis Outline

- **Chapter 2:** This chapter provides background information on related works, lifelogging and artificial neural networks.
- **Chapter 3:** This chapter includes some observations made during the instantiation of this thesis
- **Chapter 4:** This chapter is a quick introduction to the models and their interconnectedness for reference to the rest of the thesis
- **Chapter 5:** This chapter is a outline and exploration of the dataset used in this thesis
- **Chapter 6:** This chapter details the methods used in modelling of all models created in the making of this thesis. A object detection model as well as two event detection models for lifelog data.
- **Chapter 7:** This chapter consists of an evaluation of the various models from chapter 6
- **Chapter 8:** This chapter is an discussion of the results achieved and relating them to the research questions in section 1.2.1
- **Chapter 9:** This chapter concludes the thesis and consist of a general summarization of the work done throughout.

Chapter 2

Background

This chapter starts with an outline work of others related to this thesis and the research questions it's based on. Moreover, background theory on both Lifelogging and Artificial Neural Networks (ANN) detailed.

2.1 Related works:

This section will address some of the studies and work in the field of lifelogging and 'eating moment' detection that are deemed relevant for this thesis.

2.1.1 Eating Moment Recognition using Heart Rate Responses

Hotta et al. [35] proposes a method of 'eating moment' recognition leveraging bimodal heart rate responses. They use two features, namely short-term heart rate increase and long term heart rate increase. Heart rate is shown to increase moderately during consumption of food and to a larger degree post consumption. They achieved a F1-score of 0.56 on detecting 'eating moments'. In order to reduce temporal noise, they remove heart rate data of periods where the user is walking or performing similar activities. The 'eating moment' definition used in the paper differs greatly from the ones used in this paper as Hotte et.al don't include the consumption of beverages.

2.1.2 Analyzing First-Person Stories based on Socializing, Eating and Sedentary Patterns

Herruzo et. al. [2] propose the LAP dataset, where they tackle the problem of analyzing socializing, eating and sedentary lifestyle patterns by recognizing the lifelogger's activities. In total, they gathered 45k images from 4 different people in consecutive days and labeled them with a certain value for each of the 3 targeted aspects (socializing, eating and sedentary). Their results, using both conventional machine learning and ANN, had a F-1 score of 0.64. Moreover, most of their errors came from misclassification of *Eating* patterns. This task is similar to the one in this paper in that both wish to identify eating moments. Where the two differ is in that the task in this paper allows for the use of additional types of data outside of images. In addition, the task in this thesis is also only centered around eating i.e. a binary classification, compared to Herruzo et al. task which is a multinomial classification problem.

2.1.3 A Baseline Search Engine for Personal Life Archives

Zhou et al. [3] propose in their paper, a method for event detection using multi-modal information from time, location and concepts with ad-hoc pre-designed rules. In their paper they introduce a system which allows for the user to retrieve basic moments from their lifelog, reliably and efficient. Depending on the type of information in the lifelog, different techniques for feature extraction are required. Wearable camera information can be turned into features for extraction using computer vision systems. They also highlight the importance of privacy and security when dealing with lifelog data. Zhou et. al.'s research differs from the research resulting in this paper in that 'eating moment' detection is narrower in its scope and more focused on that single concept. Zhou et. al. search engine could be leveraged to retrieve the 'eating moments' detected using the models outlined in this paper to provide users with ease of access to said moments.

2.1.5 A Deep Learning based food recognition system for lifelog images

Nguyen et. al. proposes in their paper [6] a deep learning based system applied to food recognition in lifelog images. This type of system can further be leveraged to achieve such systems outlined previously e.g. calorie calculator, nutritional intake tracker etc. The system proposed is a 3-part system consisting of (i) 'Eating Moments Detection', (ii) 'Images Enhancement' and (iii) 'Food Recognition' as shown in figure 2.1 below.

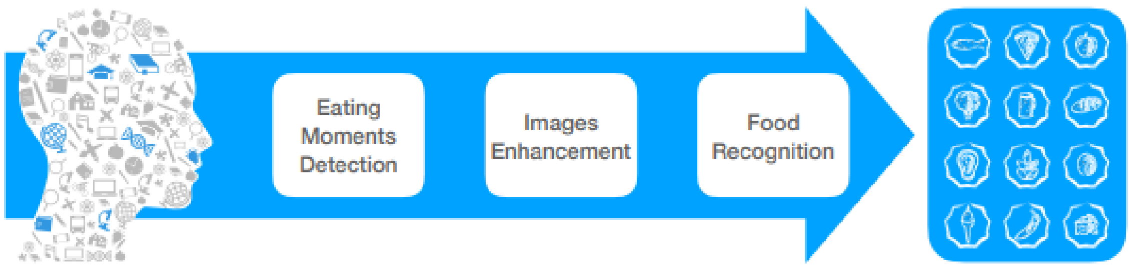


Figure 2.1: Nguyen et al. proposed system [6]

In regards to eating moments detection, the paper proposes to use a variation on Zhou et al. [2] to accomplish this task. This thesis is inspired by the work of Nguyen et al. who in their paper introduce a deep learning food recognition system intended for use on lifelog images.

Therefore this thesis will explore the eating moments detection part of the system seen in figure 2.1.4.

2.1.6 Deep Learning-Based Food Calorie Estimation Method in Dietary Assessment.

Liang et al. [42] estimate calories food using deep learning. In order to estimate calories, Liang et al. method required two photos; one from a side view, and one from a top view. GrabCut algorithm was used get the contour of the food, before volume of food was estimated. When said process was accomplished, the method could estimate each food’s calories. The results had a mess error score of ±20% for the most food groups, but some were as high as ±33.5%.

2.2 Lifelogging

This subchapter provides background information and theory on lifelogging as a process and all of that which it entails.

2.2.1 Lifelog Introduction

The lifelog, a term used to describe the actual data gathered, has also been referred to as a “black box” of a person’s life. Lifelogging, refers to the gathering, storing and using the lifelog. The amount of data generated and stored in the “black box” are heterogeneous and tends to be of a large volume. The data gathering process is accomplished through wearable sensing technology such as cameras and fitness trackers. Apart from this, other complementary data can be gathered e.g. computer usage, music listening history. All data combined provides rich contextual information that has the potential to leveraged into valuable tools for information extraction from a lifeloggers life [4]. The goal of this background information on lifelogging is to provide an overview of key concept and insight into three topics relevant to the research project:

1. What goes into the gathering, storing and enhancing the data in a lifelog?
2. What are some potential use-cases and/or applications made possible through lifelogging?
3. What are common challenges and methods for event detection and information retrieval of a lifelog dataset?

2.2.2 Creating a lifelog

Data and data sources:

The data gathered from lifelogging is commonly mundane and repetitive. It consists of heterogeneous data gathered from the lifeloggers everyday life; eating breakfast, working at the office, commuting etc. This separates lifelogging from the ordinary documentation of our lives we are so used to seeing on social media, which more typically show highlights of a person’s day/week/year e.g. pictures from birthday parties or vacations.

Lifelogging should be a passive process in which the lifelogger gathers data from their everyday lives, without the lifelogger having to initiate any process or interact with equipment on a frequent basis. This is important so that experiences are not tainted by requiring the lifelogger to initiate any tedious process [4].

A key point in the definition of lifelogging by Dodge and Kitchin [1], is "*digital record of the totality of an individual's experiences"*. Data gathering methods to accomplish this feat are not available. That said, a lifelogger should strive for that goal. Thus far, lifelogging has been focused on the visual aspect and rightfully so as it's deemed to be the data type of most interest when gathering detailed data on life activities. However, there are many other opportunistic sources of data available for gathering, many of which has been made readily available through smartphones and other recent innovations in modern sensing technology. Following is a summary of an overview of various lifelogging data gathering tools as described by Gurrin et. al. in their paper [4].

- **Passive Visual Capture:** Refers to cameras that capture video or pictures and is, as mentioned, the most essential data source for lifeloggers. The camera should be wearable and capture images continuously e.g. a head-mounted camera that snaps a picture every 30 seconds. The camera should have enough battery life to avoid having to charge it mid-day and miss out on data. Examples of such tools are the OMG Autographer and the SenseCam. This data gathering can be accomplished in mostly a nonintrusive way, given the right equipment. However, there is something to be said on whether or not wearing a camera can affect the wearer's experience, particularly when engaging in social activities with others.

- **Personal Biometrics:** Refers to passive monitoring of metrics related to the human body e.g. heart rate, caloric output, distance traveled etc. as well as sleep duration and quality. These types of data gathering has in recent times been adopted into the mainstream through products like FitBit smartwatch and the Lark wristband.

- **Mobile Device Context:** Refers to the use of one's smartphone to passively gather contextual data for the user. Examples of such data can be GPS location, sensing other phones/people nearby through Bluetooth, and recording audio. With the smartphone industry innovating at the rate it is, one can only expect more sensing opportunities are on the horizon. Recently, there has been an emergence of early lifelog applications made available for download e.g. Moves and Saga

- **Communication Activities:** Refers to passively logging of communication with others i.e. texts, emails, phone calls etc. There exist various tools for this data gathering process.

- **Data Creation/Access Activities:** Refers to passively logging of activities (not communication) carried out on computers, phones, tablets etc. There also exist tools for this process e.g. Stuff-I've-Seen. Examples of data are web browsing history, user input and screenshots.

- **Active Capture of Life's Activities:** Refers to the either directly or indirectly logging activity. This data source is not passive, as it is initiated by the user. Examples of data are blog posts, Facebook/Twitter status updates etc. One can make an argument that this data source is not lifelogging due to it not being passive in nature.

With a good understanding of possible data to gather and how to gather them, a lifelogger can choose which lifelogging data gathering tools to use that would best suit them. Even with the use of all these data sources the "black box" would not hold the totality of a individual's experience. More data sources increase the variance of data in the lifelog and thus, the challenge of storing and organizing also increases.

2.2.3 Storing and organizing the lifelog

The primary data types that a lifelogger should aim to store are visual data, personal biometrics, human activity and information access [38]. Those types of data are gathered from many different types of sources, listed above, together with the possibility of deriving secondary data from the initial data. When the data is gathered to form a lifelog the data is normalized, and data variety is removed. Accomplishing this can be challenging, but also necessary for automatic analysis and information retrieval.
Moreover, it is important that the data gathered throughout a day adheres to the same time of day, typically in minute-based units. [38]. This process is referred to as data 'alignment metadata'.
Depending on various types of data the lifelogger gatherers, the size of said data can vary a lot. While it is trivial to store data such as biometric data, image and video data may pose challenges depending on quality and frequency due to it naturally having larger file size. Cost of storage is also relevant, regardless of either it is stored locally or in a cloud service [4].
User-identifiable content should be removed, either by deleting data or altering it to satisfactory degree. The dataset should also be password protected and all access to the lifelog should be logged to further increase protection [38].

In regard to structure, it is recommended to store the data in a hierarchically i.e. days on the highest level sorted by a chronological order, followed by hours and minutes on the lower levels [38]

2.2.4 Making the lifelog useful

Making the lifelog useful i.e. organizing it such, raises a new set of challenges. In lifelogs, there exist no generally accepted unit for which information retrieval methods can be extracted. The unit used for retrieval is very dependent on the use-case.
Another challenge that arises when a large quantity of the data originates from sensors, is that data exists in a form that is not searchable using established information retrieval techniques. Without addressing these challenges, a lifelog would be of little value to its user [4].

Based on Cohen and Conway [5] model of episodic memory, Doherty et al. as in [4] as identified several baseline principles for a useful lifelog:

- **Event detection**: The segmentation of raw unprocessed data generated by a lifelogger's data gathering tools into relevant units to provide a basic atomic unit for retrieval. Generally, the unit has been "events" i.e. a temporally related sequence of a lifelog. An example of an event can be "eating moments". Another possible unit that has received some attention is a summarization or aggregate of the data. This unit is common with biometric data e.g. how many steps a user has taken or how many calories have been burned in a day.

- **Annotating events (or other units):** Events should be annotated with meaningful semantics. An event could for instance be 'working out' and annotated with semantics regarding what type of workout e.g. 'strength' or 'cardio'. A necessary step to generate meaningful data for analysis or other use-cases.

- **Access and retrieval:** Refers to the support of appropriate access and retrieval methods. Due to the potential file size of a lifelog, the common browsing method would be very inefficient. Lifelogs requires different access and retrieval forms such as questions answering, summarization, narrative generating etc.

- **Multimodal Interaction:** Refers to the support of access and retrieval on multiple devices such as PC, mobile and newer technologies like smart glasses.

2.2.5 Lifelog Use-case and Applications

With a basic understanding of what goes into the creation of a functional and valuable lifelog now established. Next, a brief look at some potential use-cases and applications for lifelogging. As mentioned previously in this section, lifelogging offers potential to infer knowledge and mine information regarding how the logger lives his/her life. This section is a summary of some selected potential applications outlined by Gurring at al. in [4].

Self-Monitoring of activities:
Information obtained through the observation of self gives feedback and help with reaching goals and staying on "the right path" in life as the logger deems it. Lifelogging can make this activity more accessible. For example, as this research project aims to contribute to, the monitoring of eating. A type of monitoring very popular even before technological aids. At the current stage the logger has to manually note when and what has been eaten. To provide any automation in this process can streamline the activity. Monitoring eating will help keep users accountable for not just how much they are eating (breakfast, dinner, snacks, etc.) but also what types of food that is being consumed.
Smoking is an activity many of its users are either attempting to quit or to cut down. Monitoring one's smoking habits can help loggers track how many they smoke each day and if they have smoked more or less than previous days/weeks/months.

Memory assistance
Memory assistance was one of the early use cases for lifelogging, particularly visual lifelogging. It can be a very valuable tool for people struggling with short-term memory loss, dementia or Alzheimer's. "This exploited the well-known phenomenon of Proustian Moments as described in Stix (2011), where a trigger of some kind – a smell, sound, image, object, etc. – causes a spontaneous recollection of something from our past" [4]. By providing lifeloggers with knowledge of their day, a application as such could help with assisting their memory which lives up lifelog's nickname of "black box".

Population-based applications
There are also lifelogging applications that could benefit the population at large. This is referred to as population-based lifelogging applications and is where lifelogs are combined which allows for potential inferring of knowledge on a wider scale. A example of this is within a workplace environment in which knowledge could be inferred or mined from logs

consisting of worker activities. On an even wider scale, aggregated lifelogs could be used to monitor biometrics of an entire city.

2.2.6 Lifelog and nutrition

A lifelog can be leveraged to provide the user with historical data on user's overall health and self-care actions in a detailed manner, depending on various data gathered and the use of said data. A lifelog containing images automatically captured throughout the lifeloggers day is not at all uncommon and records a huge amount of visual data. In theory, this data can be used to track nutritional intake of the user via e.g. a automatic calculator of food consumption or a personal food recommender. In addition, it can also be leveraged to identify moments of physical activity. Both can be used to give users a general score on their overall health and prompt users to take actions to increase said score. However, such huge amount of raw data has its fair share of obvious challenges. Therefore, it is important to generate units of retrieval to make nutrition tracking applications more feasible.
Monitoring the nutrition habits of a person is an established mechanism typically used in the health domain for several medical conditions such as obesity, hypertension and diabetes [6]. This mechanism can and should be further taken advantage of by a wider array of people. Preventing individuals from contracting diseases such as diabetes through good nutrition and well-being is a lot more valuable than helping those who already suffer from them, at least from a global health perspective. A system that can leverage this data efficiently has the possibility to replace traditional methods of monitoring nutritional intake on a personal level. These methods rely on subjects to manually fill out questionnaires, which are error prone provide biased information and thus, can be difficult to work with [9].

2.3 Artificial Neural Networks

2.3.1 Deep Learning Introduction

The goal of machine learning is to transform input data into meaningful output data through algorithms and statistical methods that are trained from sample data [11]. Deep learning is a sub-field of machine learning. It differs from other machine learning methods in its network architecture [12].

Classification Neural Network

Neural networks can learn primarily in one of two ways. Supervised and unsupervised learning [13]. In supervised learning one utilizes training datasets with corresponding target data to learn in order to make predictions. A classification algorithm attempts to predict which of its given classes a input belongs to e.g. “is this a picture of a dog or a cat?”. A classification problem where the output can belong to one of two classes is known as a two-class or binary classification problem. If there are more than two possible outcomes the problem is known as a multiclass classification problem. Most classification models use a continuous value that represents the probability of a given input belonging to each class. Let’s say a model predicts that a moment in a lifelog has a 0.8 chance of containing the activity of “eating” and thus a 0.2 chance of not containing the activity of “eating”. Then the model would label the moment as “eating” since it’s more likely than not “eating”. This type of machine learning problem differs from a regression problem, where the output is a continuous value e.g. “the price of stocks at apple for 2021 will increase by 3%” [13].

Data for Neural Networks

In general, neural networks and machine learning models use tensors as data structure. A tensor is a container of, in most cases, numerical data. A tensor has three key attributes, namely shape, number of axes (rank) and data type.

- Rank: refers to the number of axes e.g. a tensor consisting of [1,2,3] has the rank of 1 and a tensor consisting of [[1,2,3],[4,5,6]] has the rank of two.
- Shape: is a tuple of integers that describes the number of dimensions the tensor has along their axes.
- Data type: refers to the type of data contained in the tensor e.g. float64.

To explain this concept further, 4 different tensors are provided as examples of some of the various types of tensors:

- **Scalar (0D tensor):** [5]

Scalar is a tensor consisting of only one element within a container. Scalar has an empty shape, ( ).

- **Vectors (1D tensor):** [1, 5, 2]

Vectors are one dimensional tensors consisting of one container with multiple elements. This tensor will have the shape of (n_Elements,). For the vector above the shape is (3,).

- **Matrices (2D tensor):** [[1, 5, 2],
  [3, 7, 12]]

Matrices are an array of vectors. Matrices have multiple containers within one outer container. This can visualized as a grid of columns and rows. The matrix in the examples has the shape of (3, 2). For a traditional feed forward neural network whether you have more than one feature this is the type tensor used for the input data and output data.

- **3D tensor(s):** [[[1, 5, 2],
  [3, 7, 12]],
  [[11, 9, 6],
  [6, 4, 8]]]

3D tensors (or higher) are matrices within another or multiple containers. These types of tensors can be visualized as a cube of elements. The examples above is of a 3D tensor with the shape of (2, 2, 3) i.e. one container consisting of two containers, consisting of two additional containers with 3 elements in each. 3D tensors are used as input data in most convolutional neural networks, and also in some Long-Short-Term-Memory models [14, p.31].
It is also primarily recommended that the numerical data in models are between 0 and 1

In order to train and evaluate an ANN model, the dataset intended for training a model needs to be split into 2. One training-set used for training of the model and one evaluation-set used for evaluation. The evaluation-set will not be fed into the neural network during training so the neural network will be evaluated on not-seen-before data. The split is commonly 80% for training and 20% for evaluation [14].

2.3.2 Architecture of a Deep Learning Neural Network

The architecture of a deep learning model consists of three different types of layers as shown in figure 1. i) The input layer which holds the data for the model ii) the hidden layer which sits between the other two. Most networks have multiple instances of this layer and are typically fully connected except for convolutional neural networks. iii) Finally, the output layer which outputs the network's prediction. In a traditional feed forward network, the input is sent from layer to layer as seen in figure 1. [15]. The number of layers in a model is known as model's *depth*. Layers consist of nodes and it is within these nodes that computation occurs. Nodes have related *weights* which contain the information learned. The weights are updated in the process of learning when the network is exposed to training data through optimization. The network aims at finding the optimal value for each layer/node in order to achieve reliable output predictions [14.]

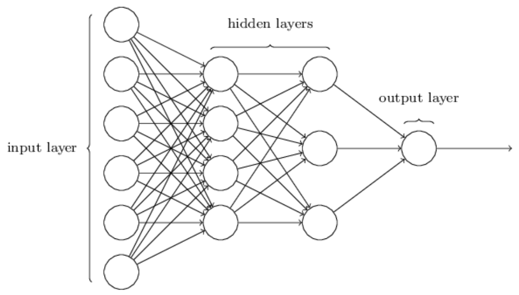


Figure 2.2 The layers that make up the architecture of a neural network [15].

Loss function

A loss function defines what metrics of measurements will be used when evaluating the network's success rate. The measurement is referred to as loss score, which is calculated after each input is processed. A network aims minimizing this quantum via training. There are numerous loss functions and picking the right one for the model at hand is important to ensure the network behaves as intended [14].

Activation functions

Activation functions are layer-specific parameters (hyperparameters) that modifies the output value of a node. Different types of activation functions are suitable for different problems and layer types [14].

Optimization

Optimization of a model is done through *gradient descent*. Gradient is the derivative of multidimensional inputs, for example tensor or matrices. The goal of the gradient descent is to find the global minimum of the loss value. To find a minimum for any given point, the optimizer will calculate the derivative in each point and attempt to move closer to where the derivative of function f is equal to 0. Figure 2 shows this step by step in a linear 1D loss curve. The jump from point to point is determined by the learning rate. In order to prevent the optimizer to find the local minimum as opposed to the global minimum, techniques such as momentum is used. Momentum tracks the velocity of the gradient by tracking not only the current gradient value but also the previous updates. It can be thought of as a ball rolling the down the loss curve [14].

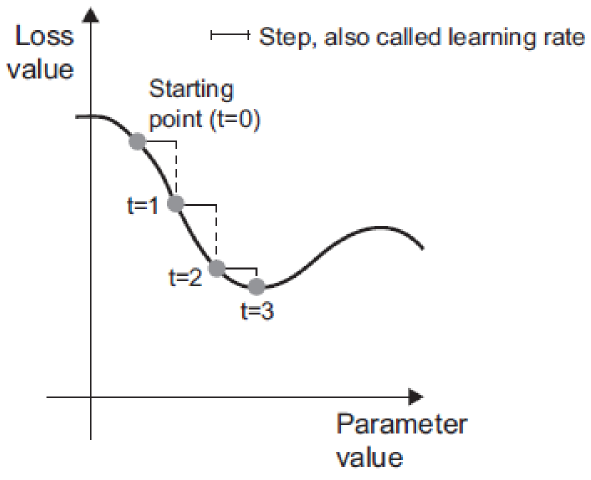


Figure 2.3: Describes the steps of the gradient descent [14].

Anatomy of a neural network

Figure 3. illustrates the way in which different components of a neural network interact. As aforementioned the input is sent from layer to layer before the network outputs a prediction. The loss function will in turn compare said prediction to the targets and calculate a loss score. The optimizer will then, based on loss score, determine how and how much the weights of nodes will be updated [14].

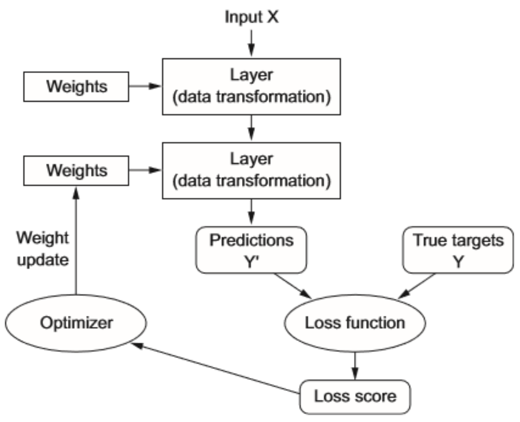


Figure 2.4. Relationships between the various components in a deep learning NN [14].

Neural networks will outperform most machine learning models. However, there are some requisites. i) Neural networks require large datasets to reliable produce good results. ii)

When dealing with large amounts of data comes the requirement of strong computational power to process it as well as storage capacity to hold said data. Given enough amount of data, computational cost must also be taken into account. iii) The data needs to be appropriate to the problem the network is designed to solve [14].

2.3.3 RNN and LSTM

Recurrent neural network (RNN)

Unlike conventional feed forward neural networks, RNNs have the ability to process sequential data by maintaining a state that is based on previously processed data. In essence, RNNs have an internal loop as shown in figure 4. RNNs take sequence data as input and loops over the data while maintaining a state containing information related to what the network has previously seen. The state is added to the input in the processing of data, so the network has input regarding both present and past [14].

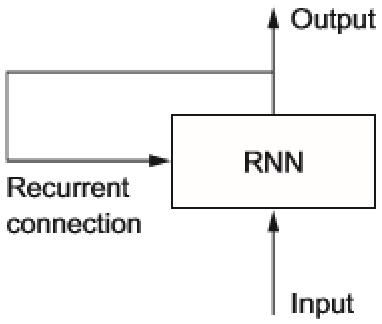


Figure 2.5: The internal loop of a RNN [14]

In practice, RNNs are seldomly used as they are viewed as too simplistic too for real life use cases. [14, p.202].

The vanishing gradient problem is main reason that RNNs underperforms. The model will eventually become untrainable if exposed to long data sequences. The updates to the parameters from the gradient gradually shrink to a point where the updates are so small that training no longer has any meaningful impact on the parameters. The exploding gradient problem is the opposite of the vanishing gradient problem, meaning that the gradient will exponentially increase its influence on the neuron updates. This has the effect of the model turning unstable and not able to learn. LSTM models are a variant on the recurrent models that are more applicable and outperforms RNNs [14].

Long-Short-Term-Memory (LSTM)

LSTM is a variant on the RNN model. Similar to RNNs, LSTM have recurrent connections i.e. connections from previous neuron activations from inputted data are used as context for

the model's current prediction. LSTMs, however, have a different formulation and architecture that is designed in a manner which addresses the vanishing and exploding gradient problem [16]. This has been a major factor in the model's popularity.
Unlike an FANN model, LSTM model's input and output data has be in a 3D array [16]. The three dimensions are

- Sample: A row in the dataset or one sequence i.e. a data point.
- Time Step: One time step is equivalent of one point of observation consisting of n_sample(s).
- Features: Refers to different types of observations or variables contained in each sample.

A LSTM's memory cells (nodes i.e. computational unit) consist of *weights* and *gates.*
Weights: There are different weights in the cell for different operations, as well as a internal state:

- Input Weights: Weights input of current time step
- output Weights: Weights output of previous time step
- Internal State: Calculates output for this time step.

Gates:

- Forget Gate: Responsible for discarding irrelevant data from cell
- Input Gate: Responsible for which values from input used in updating memory state
- Output Gate: Decides what to output based on current time step input and memory of cell.

It is the gates and a consistent data flow that ensures that the cells are stable, therefore avoiding an exploding or vanishing gradient [16].

LSTM models will fall in under one of four sequence prediction model categories [16]. Those categories are:

- One-to-one: Takes one input sample and produces one output
- One-to-many: Takes one input sample and produces multiple output values
- Many-to-one: Takes multiple input samples and produces one output values
- Many-to-many: Takes multiple input samples and produces multiple output values

2.3.4 Convolutional Neural Network

Convolutional Neural Networks (CNN) are primarily used in object detection in images and does not vary too much from other neural networks other than it can be tailored to pattern recognition. This pattern detection is what makes CNNs so useful for image analysis [17].

Convolutional Neural Networks Architecture

CNNs, similarly to other neural networks, consist of weighted neurons that optimize themselves through learning. However, in a CNN the neurons are organized into three dimensions, namely height, width and depth. The neurons are not fully connected i.e. they are only connected to other neurons in close proximity to itself [17].

There are 3 different types of layers in a CNN. The convolutional layers, pooling layers and fully connected layers.

The convolutional layer

The convolutional layer is the basis of a CNN and is what makes this technology so applicable to image detection [17]. Similar to other layers, convolutional layers receive some input, transforms that input and outputs the transformed data as input to the next layer in the network.

The input images that are fed into the convnet has a height, width and depth, thus all pixels in the image has a position. The convolutional layers use learnable kernels (also known as filters or feature detectors) who’s job is to identify edges, curves, color, circles etc. In layers deeper in the network, they can detect things like fur, feathers, scales and even deeper layers will be able to detect even sophisticated objects like entire dogs and cats. Kernels are often small in size, typically 3x3 or similar (meaning they cover 3x3 pixels), but through its course the layer will convolve the kernel across the input’s spatial dimensionality in its entirety. As the filter moves across the input, a scalar/dot product is calculated. The kernel ‘fires’ when it comes across the specific feature it’s looking for and that is referred to as ‘activations’. The different layers have different kernels that search for different features. When the kernel activates it will map that feature on what is called a feature-map. A feature map basically translates to a summation of all the various features detected by the various kernels in various layers.

Commonly, ANNs are fully connected which makes inputs such as images too large to train in an efficient manner. CNNs, as mentioned, have neurons that are organized to small regions of the input data, this is referred to as receptive field size [17].

Hyperparameters to optimize output:

- A “**stride**” determines how much overlapping between the kernel/filters there will be. Stride set to 1 means heavy overlapping.
- “**zero-padding**” is simply adding a padding to the image in order for all the pixels in a image to be processed in the convolutional network.

The pooling layer

The pooling layer works like a zoom function on the feature-map, gradually reducing the size and thus also reducing the numbers of parameters. Max-pooling layers is the most common technique used.The fully connected layer at the end of the CNN will look at all the features found in the convolutional layers and predict a outcome. It is more similar to that of a traditional neural network.

The fully connected layer

The fully connected layer will perform the same task as a standard neural network, which is explained in the deep learning background theory section. It will make a prediction based on the findings of the convolutional and pooling layer

2.3.5 Evaluating classification neural networks

Evaluation on NNs used the test portion of the dataset to evaluate its performance. Different neural networks require different metrics of evaluating its performance. Following are some the popular ways of evaluation for classification neural networks.
Before starting on evaluation methods it’s important to have a understand of classification between TP, TN, FP and FN where TP = true positive, TN = true negative, FP = false positive and FN + false negative. In order to accomplish this, each classification will be explained using the detection of eating moments as example.
TP = a moment has been correctly labeled as an eating moment
TN = a moment has been correctly labeled as NOT an eating moment.
FP = a moment has been incorrectly labeled as an eating moment.
FN = a moment has been incorrectly labeled as NOT an eating moment.

Accuracy: *“approximates how effective the algorithm is by showing the probability of the true value of the class label; in other words, it assesses the overall effectiveness of the algorithm*” [18]. In other words; total number of accurate predictions divided by the total number of predictions. The formula for accuracy is:

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

Precision: "*estimates the predictive value of a label, either positive or negative, depending on the class for which it is calculated; in other words, it assesses the predictive power of the algorithm*" [18]. In other words; total number of accurate predictions divided by total predicted positive. The formula for precision is:

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

Recall / True Positive Rate / Sensitivity: "*approximates the probability of the positive label being true; in other words, it assesses the effectiveness of the algorithm on a single class*" [18]. In other words; total number of accurate positive predictions divided by total actual positive. The formula for recall is:

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

Specificity: is a metric similar to recall, but for a negative prediction to be true. In other words, total number of accurate negative predictions divided by total actual negative. The formula for specificity is:

$$Specificity = \frac{TN}{TN + FP}$$

False Positive Rate (FPR): Measures the ratio that exists between FP and total number of negatives. The formula for FPR is:

$$FPR = 1 - Specificity$$

F1 Score: Is a harmonic combination of precision and recall. It is used when the balance between precision and recall is important [18]. The formula is

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

AUC-ROC curve (AUROC): stands for 'area under the receiver operating characteristics' and has become a very popular evaluation metric for classification problems in machine learning. However, it is not a metric as the previous evaluation metrics are, but a graphical representation. In short, AUROC indicates how well a classifier can distinguish between classes [33].

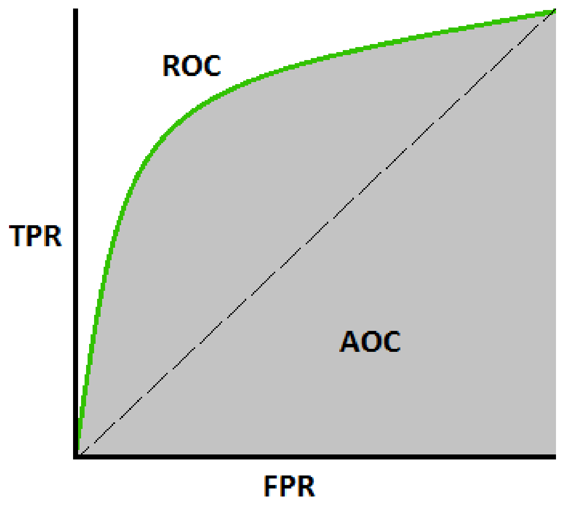
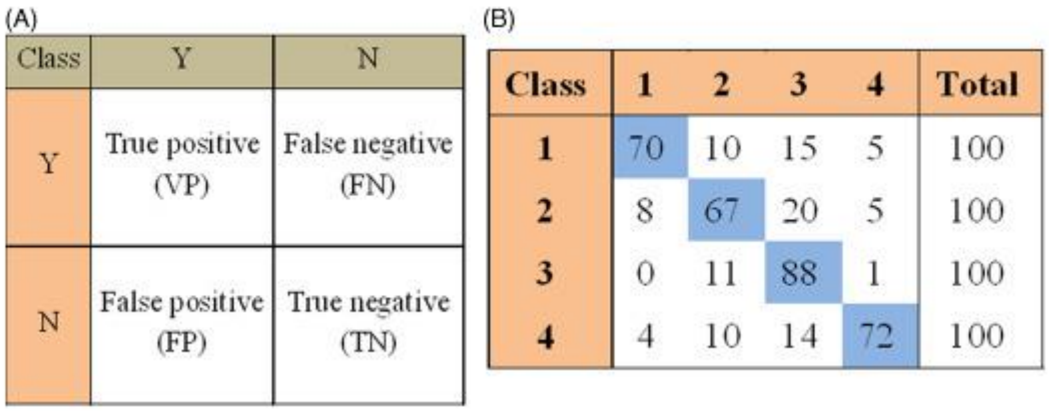


Figure 2.6: A visualization of an AUROC [44]

As seen in figure 2.6 AUROC are visualized using a graph. On the graphs y-axis, TPR (True Positive Rate) is plotted and on the x-axis FPR (False Positive Rate) is plotted. The dotted line diagonal line indicates where the true positive rate is equal to the false positive rate. The ROC curve, visualized by the green line, is a result of plotting TPR against FPR. The area under the curve (AUC) visualizes the performance of a classifier. AUC can be calculated to a metric e.g. a classifier with AUC-score of 0.7 will be able to distinguish between the positive class and negative class 70% of the time. The AUROC is also a very useful tool when comparing different classifiers [33].

Confusion Matrix: is an evaluation tool used to visualize performance of prediction models. Confusion matrices aims to visualize the correctness of classifications by computing TP, TN, FN and FP. Figure 2.7 displays two confusion matrices. Matrix (A) is a binary classification problem in which one can identify the distribution of TP, TN, FN and FP. The predictions are on the x-axis and ground truth is along the y-axis


(A)

Class	Y	N
Y	True positive (VP)	False negative (FN)
N	False positive (FP)	True negative (TN)

(B)

Class	1	2	3	4	Total
1	70	10	15	5	100
2	8	67	20	5	100
3	0	11	88	1	100
4	4	10	14	72	100

Figure 2.7: Two confusion matrices. Matrix (A) is from a binary classification and matrix (B) is from a multi-class classification [43]

For matrix (B), a multiclass problem, the same principles apply with ground truth on the Y-axis and predictions on the X-axis. A multiclass confusion matrix displays the distribution of TP, FP, FN and FP classes. Confusion matrices provides an intuitive and quick understanding of the performance of the model in its entirety, as well as the individual classes. [41].

Chapter 3

Observations

This subsection was included to shed some light on some of the observations made related to the thesis. It consist of a deeper look at human compared to machine intelligence and

3.1 An ‘eating moment’s’ complex context.

When comparing machine intelligence to human intelligence many parallels and analogies are drawn from AI to the human brain e.g. how a computer can “learn” or “understand” various concepts. However, even NN (which is designed to mimic learning similar to the human brain) differs in multiple ways to the biologic concepts learning, thinking and understanding. There are also differences computer cognitive abilities and human mental abilities. One such difference where humans outperforms computers is visual recognition of complex contexts [40]. With that in mind, a ‘eating moment’ type of detection task might be trivial to most humans, but can be very difficult for a computer to accomplish due to the complexity and variance of the activity.
With the working definition of an ‘eating moment’ being related to the consumption of any food or drink, it is obvious that such moments will contain a great deal of variation. From eating a quick snack, to a larger meal, and also including drinking water, all of which contain variation in relatable types of objects.

3.2 Scalability

A ‘one-fits-all’ type model will likely not be achievable in this master thesis as the data generated in lifelogging will vary from person to person. In addition, sensing equipment and data storage methods are not standardized.
Moreover, individual’s nutritional habits can vary greatly. For example, different types of food e.g. fast food compared to home cooked meals can change the optimal methods used in detecting the event. Also, the cultural differences in countries/regions introduce various other challenges related to image detection such as food and surroundings have different esthetics. For example, western culture predominantly uses knives and forks to eat, while some Asian countries use chopsticks. Additionally, biometric data is individual i.e. they will vary from person to person. Temporal and spatial variables can also vary from individual to individual.

That being said, each lifelogger, who uses sufficient amount of equipment, generates enough data to train a customized model for him/her.

Chapter 4

Method Architecture

This subchapter consist of a short introduction of the models that was applied in this thesis and their interconnectedness.

The exploration on how the combination of structured and visual data can be exploited for event detection and data annotation in a lifelog. The annotation provides the lifelog with a unit for retrieval, which in turn generates increased value to the lifelog [lifelog value kilde]. The eating detection results are achieved through various models as shown in fig x. below. Part one is tasked with object detection in the lifelog images. This task is accomplished using a custom version of the YOLOv3 CNN on the images captured and stored in the lifelog. The objects detected in said pictures are then annotated to the lifelog data for future use.
Part two consist of a deep learning model for detecting eating moments in the annotated lifelog dataset. The model is trained on data such as the annotated objects, original data found in the lifelog, as well as feature engineered data as features. The eating moments detected are then, similarly to objects detected, annotated to the lifelog data.

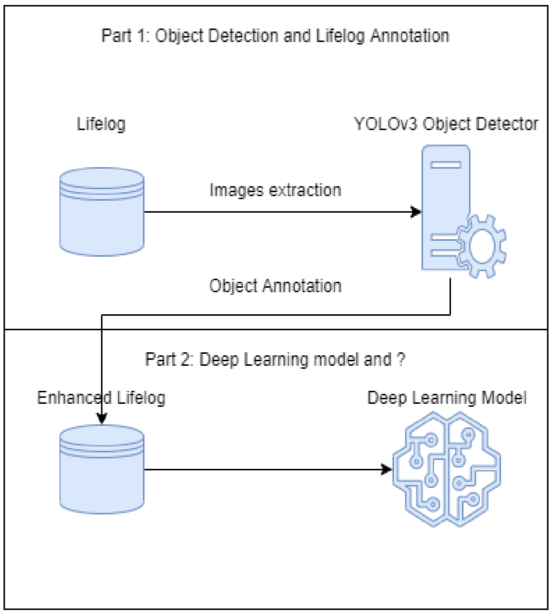


Figure 4.1: An overview of the proposed ANN models and data annotation methods, and their interconnectedness

The models are based of Zhout et al. [3] and Nguyen et al. [6] approaches and suggestions to event detection in lifelogs.

Chapter 5

Data Exploration

Before progress can be made on applying the methods to attempt to solve the ‘eating moment’ detection task. It is important to understand the data available for this thesis. This section is dedicated to the exploration of the NTCIR-14 Lifelog dataset.

5.1 NTCIR-14 Lifelog Dataset

The NTCIR-14 Lifelog dataset is gathered in a time span of 42 days from an individual lifelogger. The data is categorized in 4 groups, namely multimedia, biometrics, human activity and computer usage [20]. Following is a detailed description of said dataset, as well as some observations for context of the data’s relevance to the ‘eating moment’ detection task.

Multimedia:

- Using the OMG Autographer the two lifeloggers captured 2 images per minute throughout the day. The image capturing starts when the lifeloggers starts his/her day and lasts until he/she goes to sleep. There are roughly 1500 images gathered from each day.
- Music listening history

Vision is one of the primary senses humans use when observing and interacting with the world around them. Images thus becomes a very important source of data gathering when attempting to collect data on the entirety of an individual's experience. The dataset consists of approximate 1500 images for each day, which equates to ~63000 images.
The quality of data varies from image to image. Lack of lighting, fast movement of camera or objects in view can all lead to blurry images. The angle of which the images are captured from are from the chest area of the lifelogger. This leads to scenarios where objects are blocking the camera from seeing what the lifelogger sees. In addition, a person may turn their head and not their torso, thus the camera will not capture what the lifelogger sees. In other words, the camera is not the end all be all tool for capturing the visual data that the lifelogger is exposed to.
Music listening history, although may prove to be valuable data for other projects, carries little relevance to an eating detection prediction model.

Biometrics:

- Biometric data tracked continuously 24 hours 7 days a week. Included data are heart rate, calorie burn and steps.
- Blood sugar levels are tracked every fifteen minutes.
-

Human activity:

- Semantic locations visited
- Physical activities
- Diet log consisting of manually logging of photos of food

Two activities were annotated in the dataset, namely transport and walking. Transport is referring to whenever the user uses any means of transportation e.g. bus, car. Walking refers to any moment where the user was walking.

Computer usage:
Tracking the lifeloggers activity on computers. Monitoring user input and
screen recording using ASR screen recorder. The screen recorder collects data on a minute to minute basis. The data gathered from screen recordings is filtered in order to protect privacy rights.

The dataset has also been approved by a ethics committee for all research relevant to the research project.

The lifelog provided was in a CSV structure and consisted of rows and columns. Rows are data points where each one represents a minute of the lifeloger’s day and columns are the various types of data gathered for the different data points.

5.2 ‘Eating moment’ labels

In order to accomplish the task of detecting eating moments, data labels on whether eating moment was occurring or not, were required. The labels provided were fully in line with the NTCIR-14 dataset.
The eating annotations are naturally very important for this research project, as it is required to train a classification model and would be a very time consuming and tedious task which had already been accomplished. The task was outsourced and was accomplished through

manual labor by individuals that were shown images which they annotated. Humans are, however, prone to errors i.e. the annotations are not 100% correct.

5.3 Challenges and Variations in the Lifelog Dataset

As previously mentioned in this section, the dataset was recorded in the span of 42 days. However, the 'eating moment' labels were not available for the same period of time. The labels are of utmost importance and thus the available data to work with for this task is less expensive than the NTCIR dataset. The labels are available for 28 of the days and there exists a gap of 5 days (after the first 10), making the dataset non-continuous
During this time span the lifelogger travels on an international trip. This is not ideal, in terms of achieving optimal results, as trips have the tendency to upset a person's daily routines.

The necessary data e.g. labels, semantic data, image file names etc. were not provided in one complete dataset. Instead, they were provided in several files. This meant that a considerable amount of work had to be performed in order to end up with the final dataset which includes all the necessary and exclusion of days/rows where labels were not available. This process was a large undertaking and was performed using data frames in pandas, the python library.
Pandas has the functionality to merge data frames together, but in order to do so, a 'key' variable is required. Depending on what data the various files contained, several such keys had to made/altered. Some could be merged on the 'Image_path' variable and others on 'minute_id'.

Chapter 6

Method

This chapter will detail the process of developing the models used in obtaining eating moments detection in the dataset. As mentioned earlier, the models are difficult to apply to other individuals Lifelog or data, thus this chapter is of high importance in order provide detailed insight into the development such that it can be replicated. This chapter is divided into two main parts consisting of the two main components of the models outlined in the architecture chapter.
Firstly, the modelling of a custom YOLOv3 object detector is detailed. This modelling process is different compared to the other models due to the framework (darknet) its built on and the image type data used for training it.
Second, this chapter will detail the development of two NN models, one FANN model and one LSTM model. It will include all steps taken, from the preprocessing stage to the modelling and training stages. For each stage, techniques and methods of the development are outlined, as well as reasoning for applying them.

6.1 Darknet and YOLO

Darknet is an OS neural network framework developed by Joseph Chet Redmon on which YOLO is run [21].
You Only Look Once (YOLO) was selected for use in detecting eating related objects in this thesis for the accessibility of the framework and its computational strengths
YOLOv3 (version 3) is a CNN state-of-the-art real time object detection model. On higher-end graphic cards the system can process images at 30 frames per second with a mAP of 57,9% on COCO (common objects in context) test-dev [23]
YOLO’s accuracy is on par with other object detection systems, but its speed is significantly faster. The system also offers the capability to trade speed for higher accuracy without requiring a retraining of the model. YOLO differs from other detections systems in that the entire image is processed at once, compared to convolution that occurs in a traditional CNN, as can be seen in figure 6.1. The YOLO detection system first resizes the image given as input to a 448 x 448 dimensions divides the image into regions/grids and creates bounding boxes where the model detects an object [22].

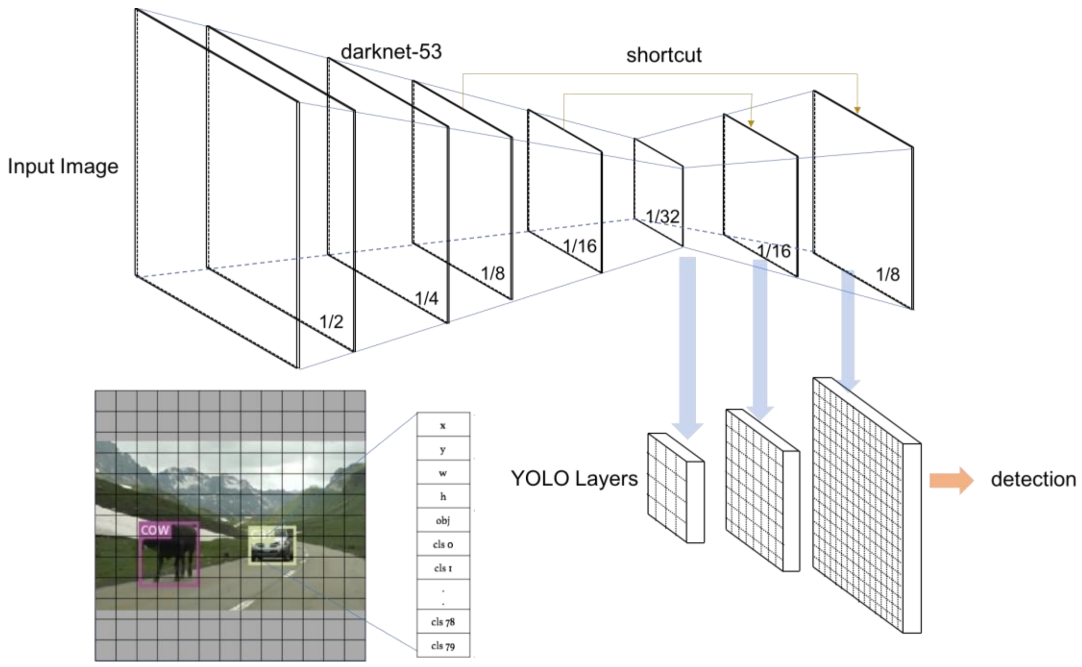
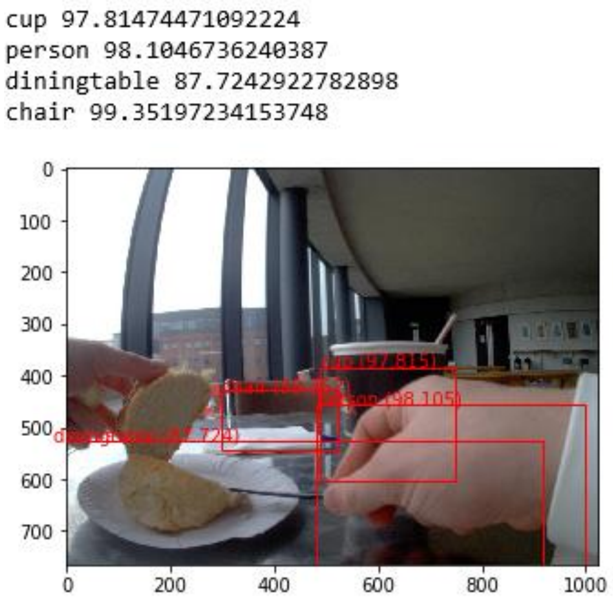


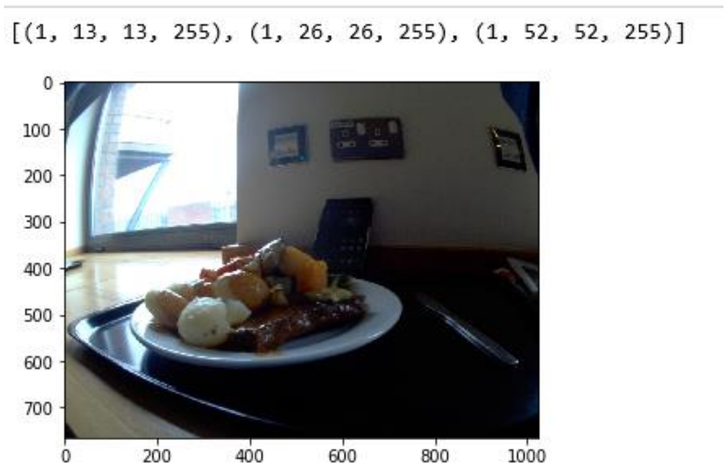
Figure 6.1: Shows the YOLOv3 architecture

The speed of YOLO is one of its greatest features. The base network can run 45 frames per second with a Titan X GPU, but with tradeoffs in accuracy for more speed can run up to 150 frames per second [23]. This ability makes the model well suited for image detection in lifelog images, as a lifelog will contain and generate a large volume of images that will need to be processed. Another argument for the use of YOLO is that the processing power required when using detection models that are more intensive is not readily available, nor cheap.
ImageNet is a database, on which YOLO's preconfigured weights were trained, described as "*a large-scale ontology of images*" [24].

Using the YOLOv3 pre-trained weights yield some promising results as seen in figure 6.2 However, in fig x. no objects have been detected, even though the image contains valuable information to detect an eating moment, easily visible to a human eye. These results conclude that the standard YOLOv3 object detector is not an adequate tool for eating moment detection, despite showing promise in some scenarios/images. Therefore, a retraining of YOLOv3 was undertaken, in an attempt to achieve better and more relevant results in regard to object detection.



```
cup 97.81474471092224
person 98.1046736240387
diningtable 87.7242922782898
chair 99.35197234153748
```



```
[(1, 13, 13, 255), (1, 26, 26, 255), (1, 52, 52, 255)]
```

Figure 6.2 Shows two images of 'eating moments' extracted from the lifelog.

6.2 Re-training YOLOv3

As YOLOv3 is originally designed to operate on Linux system, AlexeyAB's YOLOv3 Github repository was applied for training of the custom YOLOv3 convolutional neural network The images used for the retraining are taken directly from the NTCIR lifelog dataset. This is to provide YOLOv3 with data that accurately captures the world seen from a lifelog camera.

6.2.1 Class Selection

Retraining YOLOv3 requires that the various types objects that will be detected, be listed in a txt-file referred to as classes. Deciding on what classes the retrained version of YOLOv3 will attempt to detect is a important decision. Retraining YOLOv3 is a relative lightweight processing task. Nevertheless, with limited processing capabilities and the whole task of retraining being considerably time consuming; this is the sort of task one wants to get right within few iterations.
Some requirements for a class to be included in the class file are:

- Related to eating in a somewhat unique fashion as
- Various instantiation of the class has to be similar i.e. share similar shapes and colors.
- Consistent in many eating moments

After reviewing many pictures from the lifelog the following classes have been selected for detection in the retraining of YOLOv3:

Fork & Knife: These are two separate classes but have both been included for the same reasons. Namely that various knives and forks are very similar in both shape and color, they are used often by the lifelog during eating and they appear somewhat seldom outside of eating moments. The photos annotated for retraining consist of knives and forks from the various eating moments in the lifelog, i.e. a knife and work will be in close proximity to the camera and often in the hands of the lifelogger. This means there could be a higher chance of detection when the lifelogger is using knives and forks for eating, and a smaller chance for detection if the items appear in a different context.

A challenge with these two classes is that fork and knife, while in the hands of the lifelogger, are often in motion, which means that the object often appear blurry on photos. In addition, the objects are long and thing, thus larger parts of the data contained in the labeling are background data and not the fork/knife itself (depending on angle).

Plate: Somewhat similar to the knife and fork classes in that plates are also often similar in shape and color. That said, this class varies more than the aforementioned i.e. plates are often round and white but can also be squared and black. A challenge with the ‘plate’ class is that the view of the plate will be partial as there will be food on it during eating. Also, plates will look different to each other depending on the angle i.e. the shape of a plate in a picture taken from a horizontal point of view will be different from one taken from a top-down point of view.

Cup: This class was included to capture data on the moments when the lifelogger is consuming beverage from a cup. Cups shape are mostly consistent and recognizable.

Spoon: A late addition to the class selection. Was included for the same rationale as the knife and fork, although it appears less often in eating moments images. It could still prove to be a valuable detection.

6.2.2 Data Labeling

The custom YOLOv3 object detector training requires labeled images i.e. images used for training have a corresponding text file, which includes what relevant objects are in the image and their cartesian coordinates (xy). This is a task that must be completed manually. To make said task less time consuming, a labeling tool customized for YOLOv3, named Yolo_label has been applied. Yolo_label automatically creates a txt file to each image that has been labeled. The manually labeling is a simple task consisting of selecting the class you want to label and dragging a label box around the object(s) in the image, as shown in figure 6.3. below

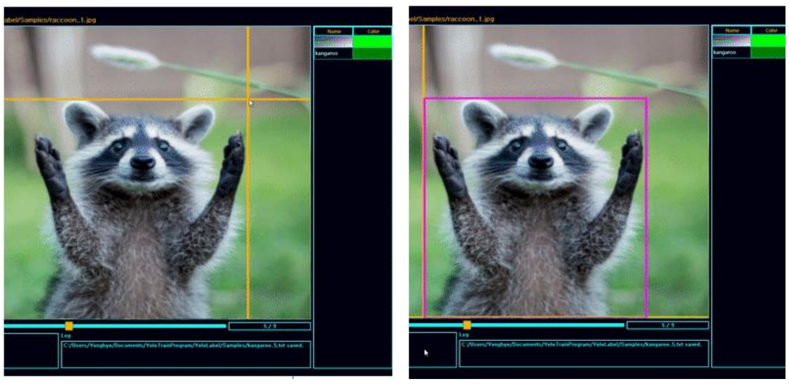


Figure 6.3: Shows how YOLO_label was applied in the image labeling process.

As aforementioned, the images used as data for the retraining are taken from manually extracted eating moments in the lifelog dataset. This means that there will be many very similar images. In order to avoid overfitting on the retrained YOLOv3 model, but at the same time give the model sufficient data, some measures have been taken when labeling the images. In the labeling process, no two labels should be the same. Some labeling boxes only include a partial object as to increase the model's ability to detect objects even though they are partially blocked. Other labeling boxes are of the same object, but with different lighting and/or angles. For the knife and work class this works very well, as they change position in the image all the time as they are being used.

6.2.3 Training a customized YOLOv3 Model

YOLOv3 1st Iteration

With the classes selected and images for each class labeled with, the actual training could commence

The processing of data for retraining was accomplished through an external processing via google colab, which allows for 12 hours session using GPU. The training was completed after 3 sessions of 12 hours.

After a quick and dirty evaluation of the first iteration using lifelog images, the results showed that the model would not be able to adequately detect objects in images required to accomplish the end goal of detecting eating moments. In fact, they were not good enough to warrant a evaluation. Therefore, a 2nd iteration was required.

Yolov3 2nd Iteration

The best way to improve any deep learning model is to acquire a larger quantity of training data [14, p104]. The model was trained from scratch on newly acquired images from Google Open Images (GOI) in addition to the image dataset used to train the first iteration of the model.

Google Open Images is a large scaled database of images containing various labeled objects. During this projects, version 6 of was released which contained several new objects, some of which were relevant to the retraining of YOLOv3.
Using OIDv4 ToolKit to retrieve pre-annotated / bounding boxes images of the classes: Spoon, fork, plate and coffee cup. The annotation to the images, located in a separate txt file, was not in YOLOv3 format, thus they had to be altered. This process was somewhat challenging as the bounding boxed used an alternate XY coordinate system and converting them. In addition, the class ID had to be changed from e.g. "Plate" to its corresponding class number ID.
The actual training was conducted identical to the 1st iteration. As can be seen in figure 6.4 below, the model now was able to perform detections on images.

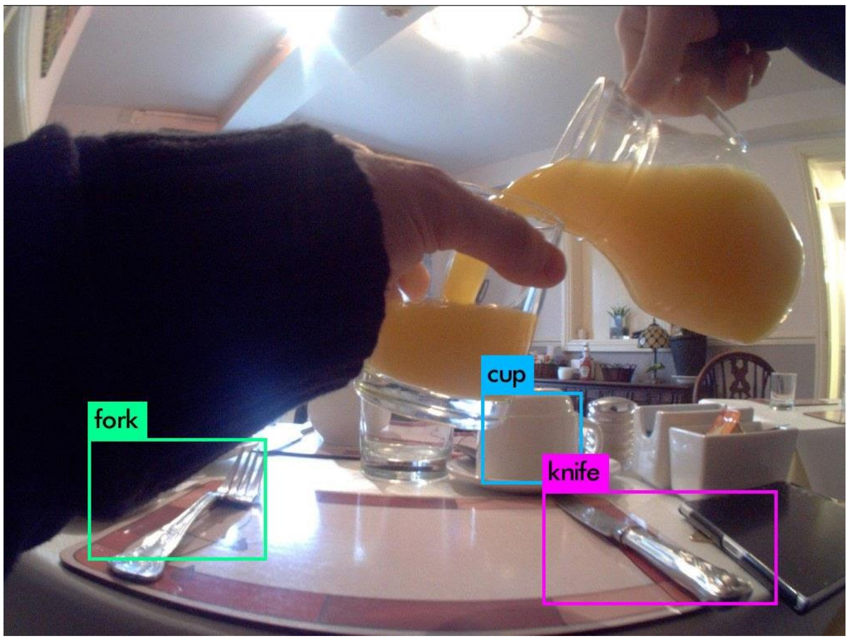


Figure 6.4 Detections on a lifelog image of a cup, fork and knife made by the re-trained YOLOv3 model

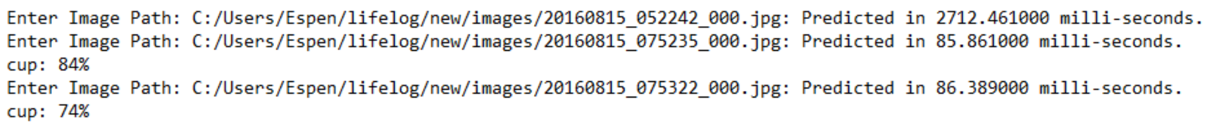
No elaborate evaluation of the model was performed at this stage. Instead, its performance was deemed adequate based on the detections it was able to make on images from lifelog 'eating moments'.

6.2.4 Annotate the Lifelog with Detection Image Detection Scores

With the customized YOLOv3 object detector working at a satisfactory level, the process of annotation the lifelog could start. As detailed in the data exploration section, the
Using of the retrained YOLOv3 the lifelog dataset was expanded upon by adding data in the form of a new features related to each object the object detection model was trained to

detect. Where objects were detected these features were given values that reflect how confident the model was in the accuracy of the detection e.g. the model detects a fork with 80% accuracy, thus the value is set to 80. No detection of any object or no image available at a given moment will have value of 0.0.

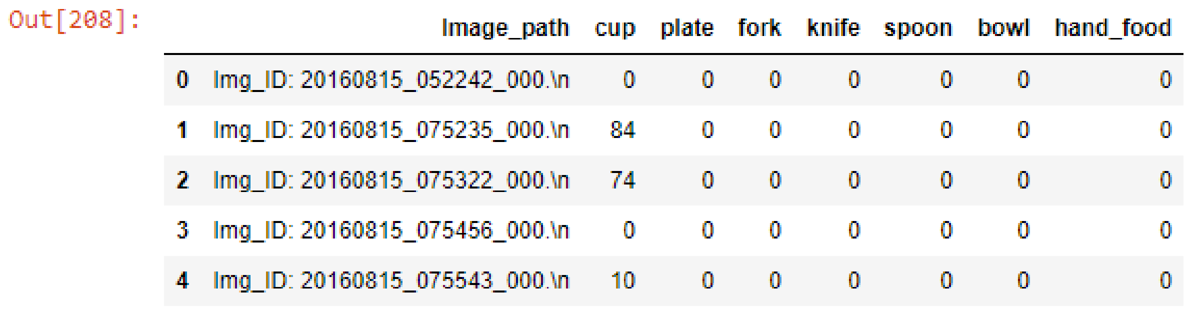
The process of combining the results provided by the YOLO object detection model with the lifelog, in order to annotate the dataset, is detailed here:
When the customized YOLOv3 object detector had processed the images of the lifelog, the detection results were provided in a 'result' text file that included a prefix for each image, as well as the image's file location + names, detection speed and any corresponding detection YOLO made on that image as can be seen in figure 6.5 below.



```
Enter Image Path: C:/Users/Espen/lifelog/new/images/20160815_052242_000.jpg: Predicted in 2712.461000 milli-seconds.
Enter Image Path: C:/Users/Espen/lifelog/new/images/20160815_075235_000.jpg: Predicted in 85.861000 milli-seconds.
cup: 84%
Enter Image Path: C:/Users/Espen/lifelog/new/images/20160815_075322_000.jpg: Predicted in 86.389000 milli-seconds.
cup: 74%
```

Figure 6.5: Snippet of YOLOv3 results extracted from the result.txt file

In order to get these results into the lifelog dataset, the result file structure had to be altered so it could match and merge with the lifelog. The detection speed and file location information was removed and replaced with the prefix "Img_ID:" using a python script, leaving image names + detections. The file was then transferred through another script that would store each filename in a pandas dataframe with additional columns for each of the objects that could potentially be detected. Then the script would then add values of the detections to its corresponding image resulting in the data frame shown figure 6.6 below.



Out[208]:

	Image_path	cup	plate	fork	knife	spoon	bowl	hand_food
0	Img_ID: 20160815_052242_000.\n	0	0	0	0	0	0	0
1	Img_ID: 20160815_075235_000.\n	84	0	0	0	0	0	0
2	Img_ID: 20160815_075322_000.\n	74	0	0	0	0	0	0
3	Img_ID: 20160815_075456_000.\n	0	0	0	0	0	0	0
4	Img_ID: 20160815_075543_000.\n	10	0	0	0	0	0	0

Figure 6.6: The data frame containing image IDs and detection scores

In order to merge the resulting data frame with the lifelog data frame, the contents in Image_path column had to be altered so it only contained the image file names and the same had to be accomplished in the lifelog data frame. Accomplishing this enabled the two data frames to be merged This resulted into the lifelog being annotated with the detections made by the YOLO object detection model and thus, the lifelog now contains numerical data leveraged from the lifelog images.

Now the detection and annotation part of the completed, work on the deep learning model for eating moments detection could commence.

6.3 Data preprocessing

Data preprocessing is an important stage of NN development. It entails the process of preparing input and output data before feeding them into a NN [14]. In this stage, several preprocessing techniques will be applied

6.3.1 Feature selection and engineering

This subchapter will describe how the data pre-processing processes was conducted. All tasks were completed using the python library Pandas, unless otherwise specified.

Features:
"*A feature is an individual measurable property of the process being observed*" [25]. Multiple features are often used in ML algorithms, ranging from tens to hundreds as the machine learning field has expanded and become more complex over the years [25]. A good feature is a feature that allows one to solve the problem at hand more elegantly using fewer resources (e.g. computing power) and with far less data [16].

Feature selection:
Feature selection (dimensionality reduction) is the process of selecting the most relevant features for the task at hand [25]. The aim is to select a sub dataset derived from the original dataset capable of providing a machine learning model with the necessary data to complete its classification task, while simultaneously avoiding the inclusion of features that either hinder or simply just slow down the process without improving results.
In the case of working with a lifelog this is a very important process. A lifelog is in an ideal world supposed to contain data which describes the totality of an individual's experience [1]. With such large volume and variance of data the, a large part of the data will not be relevant for various types of event detection tasks, and if included would only slow down the process and result in an unnecessarily complex model.
Following is a list and reasoning over the features that have been and have not been selected for use in the deep learning model tasked with detection of eating moments. This

list represents the features that were included, not in the finalized models, but up to the experimentation phase.

- **Heart rate:**
  Heart rate can increase after or during eating a meal, particularly a carb heavy meal [26]. However, heart rate can also be indicator of other activities such as stress, exercise etc. and thus can be difficult to differentiate [34]

- **Objects detected:**
  An obvious feature to include. The visual data was assumed to be the most important feature for the deep learning model. More on this later.

- **Steps:** Assumption that it is highly unlikely a person is moving while eating. That said, the average person is stationary large parts of the day. That means that there would be correlation between eating moments and (lack of) steps taken in a given time frame. But, since humans are mostly stationary beings, lack of steps in a moment would be prevalent in large parts throughout a day.

- **Blood sugars**: levels are tracked every 15 minutes in NTCIR-14 dataset and may be a good indicator of when someone has had a meal or a drink. But, tracking blood sugar would likely not be relevant for most lifeloggers as it is a
  it is also intrusive to track, which conflicts with lifelogging data gathering

Feature engineering

Feature engineering is closely related to feature selection. It is the process of creating new features for selection in order to increase performance in a machine learning model [16]. When applying deep learning it is important to get the best usage of your data as possible. The lifelog's given format was not optimal and therefore, by applying feature engineering, one can enhance the dataset to get more valuable data.
The data structure contained in the lifelog was suboptimal for detecting eating moments and was therefore improved upon through manipulation for a more optimal dataset for this given task. Following is a list and reasoning over the features that been engineered to include in the deep learning training phase.

- **Time of day**
  A time of day feature was engineered using an already inc place time feature. It was created so that models could correlate ‘eating moments’ with a general time of day, compared to a specific time value i.e. ‘morning’ instead of ‘0743’.

- **Sleeping:**
  A feature engineered feature that adds to the human activity category in the dataset. Since the eating moments in the dataset are not a labeled activity i.e. activity = 0, and the same goes for sleeping, this feature has been engineered in order to help the model differentiate and increase correlation between eating moments and activity = 0 or “NaN”. Said feature was engineered using the date/time features and image_path variable. Since the lifelogger turns off their OMG Autographer during sleep / nighttime the image_path variable during sleeping hours is set to “NaN”. Combining this with time data; all timestamps activity feature where no images are linked, and time is between 01.00 am and 06.00 am previously set to “NaN” were now transferred to “sleeping”. This feature might not be 100% accurate as there might be moments where the user is up late or early and the OMG Autographer, but for some reason the image or image path is missing. This would lead to said moment would also have activity set to “sleeping”. However, this is likely a safe assumption to make and any errors potentially caused to the dataset would be very minor.

6.3.2 Data Normalization:

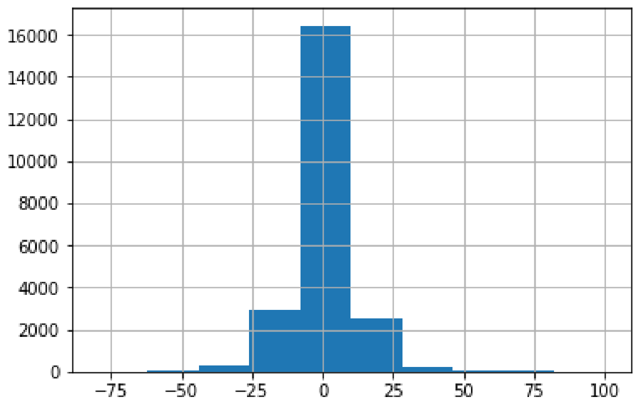
Data normalization (aka. feature scaling) refers to the process of scaling the data to fit within new ranges [27]. For instance, the Min-Max technique where the data is fitted within a predefined boundary e.g. from 0 to 1. This is a required process when dealing with features that have different ranges. It is a common practice to have all data ranges fit between -1 to 1. Depending on the type and distribution of data, different techniques are used

Heart: When dealing with features that are normally distributed i.e. most instances of said feature hover around the mean value of all instances, it is recommended [28] to use the standardization scaling technique commonly referred to as Z-score normalization. This standardization transformation takes the mean value of the feature to 0 and the standard deviation to 1.

The heart feature in the lifelog dataset is, as shown in graph.x, below a prime candidate for this type of scaling, which was the reasoning for applying it. However, graph.x also indicates that there exist outliers in the data. Outliers can be troublesome and refers to data points that contain values that deviate in a significant manner to the mean value. This can cause the majority of the data to become indistinguishable from one another [28]. To combat the outlier problem, the values that were deemed necessary were reduced/increased before applying the Z-score normalization. These values were most likely a result of faulty readings by equipment or some error storing the data.

```
In [7]:  df['heart'].diff().hist()

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x21980b25be0>
```



Graph 6.1: Showcasing 'heart' feature's distribution where mean value is set to 0. The instances are primarily in and around said mean value as expected.

Image Detection data: The lifelog image detection data was normalized using the Min-Max technique which was a trivial process.

6.3.3 Data vectorization

As previously discussed, all inputs for neural networks should be in the form of tensors consisting of numerical data. The process of transferring data such as text and images into tensors and numerical data is referred to as data vectorization or data encoding [14].

One-hot-encoded features: The one-hot-encoding technique is a common form of data encoding. This technique transformers all unique values to separate features and provides the values 0 or 1, depending on if that feature is present or not, in a data point. This technique was applied to, the ‘activity’ and ‘location’ feature. To provide an example; the ‘activity’ feature had ¾ unique values, each of which were given their own column in the dataset named accordingly. Where the user’s activity was previously set to ‘walking’ would now be set to 1 in a ‘walking’ column. The one-hot encoding technique was applied to both the activity and semantic location features.

6.3.4 Dealing With Missing Feature Values

Incomplete datasets are very common when dealing with real world data sources [28], such as lifelog data. Missing values in NN must filled out, both for training of a network and in test/prediction data. One common practice is to simply to replace the missing values with a ‘0’. The model, during training, will learn to ignore those values [14].
In the lifelog dataset there existed various missing or NaN values on different features. Instances of this varies a lot, but some examples are heart values, semantic locations, missing images e.g. during sleeping etc. which results in NaN value on detection features. All features that were one-hot encoded (activities, locations) took care of itself as missing values did simply just not get its own separate column. The ‘heart’ feature that was standardized the missing values were set to 0 i.e. the mean value. The same was done for the detection images, which would mean that none of the objects were detected at that point in time.

6.3.5 Dataset shrinking

The dataset, as previously mentioned, in terms of rows and columns are structured so that 1 row = 1-minute. 1-minute av data would in all likelihood not contain enough information for any neural network model to perform a detection on. Reasonings for this are images can be blurry/missing (meaning low detection opportunities), missing values or simply the data at that point in time revealed little to indicate an ‘eating moment’ was taking place, . Instead, in the spirit of attempting to improve potential results, it would be beneficial to reshape the data to one input representing a longer period of time, compared to one minute. To accomplish this, pandas’ data frame were leveraged and 15 rows were turned into one, and the mean values of features were calculated. To elaborate further, this would mean that if 10/15 rows where the one hot encoded feature home was set to 1 and the other 5 set to home = 0, the shrunken dataset home feature will have a value of 0.666. This also required some altering to the target feature i.e. the binary column that indicates eating or not eating. Where this column had the value of > 0.1333 were set to 1 and values below that were set to 0. In practical terms, this would equate to 15 rows shrunken that had 2 or more eating moments would be set to ‘eating’ and 1 or less row would be set to ‘not eating’.

The final dataset is a matrix consisted of 1611 rows and 34 columns where rows represents 15 minutes of time and the columns consist of features/variables. A detailed summary of the dataset is provided in the table below for reference.

Description	**Value**	**Normalization/Encoding**
Rows	1611	NaN
Columns	34	NaN
Shape	(1611, 34)	NaN
Data Type:	float64	NaN
Rows where Eating = True	130	NaN
Rows where Eating = False	1481	NaN
Visual Detection features	cup, plate, fork, knife, spoon	Min-Max
Biometric features	Heart	Z-score
Semantic features:	activity, location	one-hot-encoding

Table 6.1: A summary of the dataset used in training of ANN models

In addition, many semantic locations were removed from the dataset due to being deemed insignificant, mostly because many were highly infrequent. Too many features would impact the training of a model in a negative way, since it would unnecessarily complex the data. Further alteration would be made to the dataset during model experimentation.

Finally, the dataset was split into 4 parts as shown in figure 6.7. This is a necessary step in order for ANN models to be able to train and evaluate models. X_train and Y_train are used in training, while X_test and Y_test are used for evaluation of the model. X_train and Y_train make up 70% of the dataset and thus, X and Y test make up the remaining 30%. A conventional split is 80/20 but due to the need for a larger amount of data required for evaluation 70/30 was chosen.

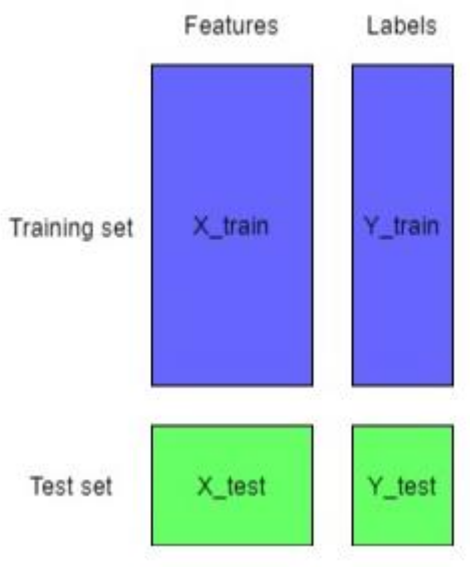


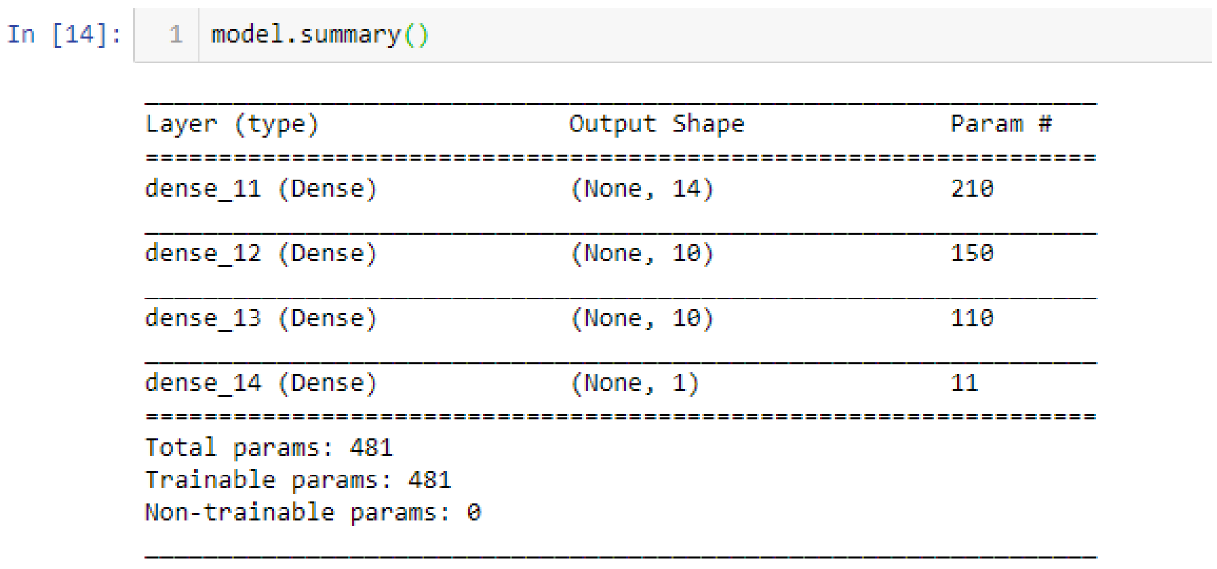
Figure 6.7: A visualization of the splitting of the dataset for training

6.4 Feedforward Artificial Neural Network Modelling

The first model to attempt to tackle the 'eating moment' detection task is a Feedforward Artificial Neural Network (FANN). The reasoning for the creation of this model is that a FANN model is relatively simple to model and sets a benchmark for result comparison with the other model.

6.4.1 Modelling Network Architecture

The hyperparameters used in the finalized model was selected through a mixture of experimentation and decision based on related academic work of others. The process of arriving at the architecture and hyperparameters summarized in figure 6.8 are outlined in this and the following section.



```
In [14]:  model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 14)                210
_________________________________________________________________
dense_12 (Dense)             (None, 10)                150
_________________________________________________________________
dense_13 (Dense)             (None, 10)                110
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 11
=================================================================
Total params: 481
Trainable params: 481
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.8: A summary of the finalized FANN model used to predict eating moments.

As figure 6.8 above shows the FANN model consist of 4 layers of which the 2nd and 3rd layers (dense 12 & 13) are hidden layers and the 1st (dense_11) and 4th (dense_14) layers are input and output layer respectively. All layers are dense layers, which is a regular deep learning layer that allows neurons to receive input from previous layer (or input) and outputs a transferred tensor computed using two linear operations - a dot product and addition [14] The input layer has 14 neurons, equivalent to the number of features in the input data. The output/final layer has only one neuron due to the task being a binary classification problem i.e. output is either 0 or 1.

The first 3 layers (input & hidden) leverage Rectified Linear Unit (ReLu) as their activation function. ReLu is good for keeping the neurons in check by hindering the occurrence of the vanishing gradient problem. The output layer uses a sigmoid activation function that outputs a probability of the input being eating moment or not an eating moment. Sigmoid is highly popular with binary classification problems [29].

Binary cross-entropy was applied as loss function for the model. It is widely regarded as the optimal loss function when dealing with a binary classification problem that requires output in the form of probability [30]. For optimizer the model employs Adam, which is a momentum-based optimizer. Adam is specifically designed for neural networks and its core strength lies in providing each parameter with an individual learning rate [31].

6.4.2 Training and experimentation

With a smaller dataset leveraged for this model, the training time is on the shorter side. This allowed for some result-based testing and experimentation with various hyperparameters.

Number of epochs (determines the number of times the entire dataset should be processed during learning) should be set as high as possible [14]. Due to the shrinking process applied to the dataset in preprocessing the size of dataset is now significantly smaller, which requires a lower number of epochs. For this model the number of epochs were initially set to 100. However, loss was continuously going down at a decent pace, therefore the model was trained over 3 sessions with 100 epochs each which equates to a total of 300 epochs.

Batch size (number of inputs processed before model parameters are updates) should fairly small as the dataset is also fairly small [14]. Thus, the model trains with batch size of 32, which is regarded as a standard value.

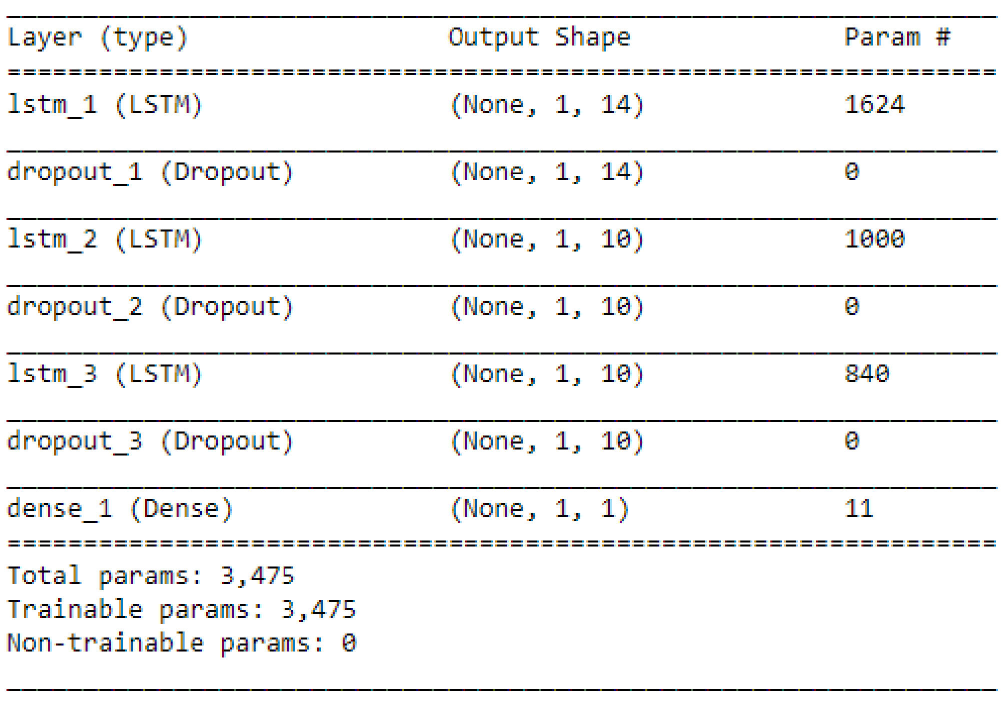
Some experimentation was also done on excluding and including various features. The features used in the finalized model were image detection, heart, activity and some semantic location features. This group of features provided the best results. Further reasoning and theories on this subject are to be explained in the discussion chapter of this paper.
This concludes the modelling process of the FANN model.

6.5 Long-Short-Term-Memory Artificial Neural Network Modelling

The LSTM model, detailed in the following subchapter, was selected for tackling the 'eating moment detection' task due to the sequential nature of the data. The lifelog data, as previously stated, is stored on a minute by minute basis in a sequential structure. A LSTM model has the ability to take advantage of this fact, by using its previous inputs to make a more informed prediction.

6.5.1 Modelling Network Architecture

The LSTM modelling was a very similar process compared to the FANN model.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 1, 14)             1624      
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 14)             0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 1, 10)             1000      
_________________________________________________________________
dropout_2 (Dropout)          (None, 1, 10)             0         
_________________________________________________________________
lstm_3 (LSTM)                (None, 1, 10)             840       
_________________________________________________________________
dropout_3 (Dropout)          (None, 1, 10)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 1, 1)              11        
=================================================================
Total params: 3,475
Trainable params: 3,475
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.9: Model summary of the LSTM model.

As can be seen in figure 6.9 the LSTM model uses the same hyperparameters as the FANN model in terms of number of nodes. In addition, the final layer is identical to the one previously used in the FANN model, as both models' output should be in the same format (binary classification). Optimizer set to 'adam' and loss set to 'binary cross entropy' is also the same. Where the model differs, besides from using LSTM layers compared to Dense, is in the inclusion of 'dropout' layers. Dropout layers can be used to combat a model from overfitting. Overfitting is a more prevalent issue with LSTM models compared to FANN models, and they are highly recommended [16].

Despite having a mostly similar hyperparameters to the FANN model, one can see by comparing figure 6.8 to figure 6.9 that the number of parameters in the LSTM model is much higher. This is due to fact that LSTM models contain parameters that take consider what previous timesteps' output was.

This LSTM model is a many-to-many type model. That entails that for each timestep in the input sequence, the model will produce an output. The 'many-to-many' type model was selected for this task due to its ability to produce predictions using multiple features and its frequency of outputs [16].

The 'many-to-one' type model was selected for this task for several reasons. First of all, in order for any model to make detections based on lifelog data, several features are required. This alone ruled out both the 'one-to-one' and 'one-to-many' model. Leaving 'many-to-one' and 'many-to-many' as the only candidates left.

In order to compare the performances of the two models, the data input should ideally be the same. And with the shrunken data, a ‘many-to-many’ model is the only viable candidate. The alternative would be to use the dataset in the state before shrinking it and feeding a ‘many-to-one’ model with equivalent 15 rows compared to 1 as input.

6.52 Training the LSTM model

Before training could commence, additional data transformations had to be done. In sectioned x.x it was outlined that LSTM models need the input and target data to be in a three-dimensional array (samples, n_timesteps, features). The dataset had already been shrunken and thus, one sample would contain sufficient amount of data for detection. Therefore, reshaping of the data input was set to (1, 1127, 14) i.e. 1 sample/row for each of the 1127 timesteps with 14 features. Target / y_train was reshaped to (1, 1127, 1).

In terms of features, this model was trained on the same ones as the previously mentioned FANN model, namely image detection, heart, activity and some semantic location features.

Compared to the FANN model, the LSTM model’s loss went down much faster and stopped dropping a lot sooner. In order to avoid overfitting, the training was accomplished using only 30 epochs.

6.6 Data post-processing

The act of shrinking the dataset also means that the prediction outputs has to be transformed into output that matches the lifelog dataset in its unshrunken form. This was accomlished by adding a key variable to the dataset before shrinking it. The key variable value was mutlipled by 15 in the shirnking process and divided by 15 in the unshrinking process. Then row’s correct ID was calulated using that number. Then the annotation could be merged with the lifelog dataset on the ID variable.

Chapter 7

Results

The results outlined in this chapter are from the YOLOv3 object detection model, as well as the two neural network models (FANN & LSTM). Both the FANN and LSTM models were used to attempt to solve the same problem; to detect eating moments in a lifelog dataset. The two models represent two different approaches to solve said problem. The YOLOv3 model has different evaluation methods compared to the other models, as it is not a binary classification model, but a multiclass one.
Towards the end, a quantitative comparison of the two ANN models' performance is also included.

7.1 YOLOv3 Object Detection Model Results

This subchapter will entail the conduction of an evaluation on the YOLOv3 object detection model.
The YOLOv3 model was evaluated using manually annotated images from the lifelog, constructed using same methods as in section 6.1.3. Each class was given 25 images, each of which did not contain any of the other classes. Threshold for detection was set +50% i.e. the YOLOv3 model had to have a confidence level above 50% in order for the detection appearing in the results. 25 additional classes were added which contained none of the classes in order to test the model on non-eating moments as well. It should be mentioned that the photos were hand-picked from the lifelog dataset. The images were in focus and the objects were in a relatively close proximity to the camera.

	Precision	**Recall**	**F1-score**
None	0.309	0.840	0.452
Fork	0.875	0.560	0.683
Knife	0.857	0.480	0.615
Soon	0.696	0.640	0.649

Plate	1.000	0.480	0.649
Cup	0.850	0.680	0.756

Table 7.1: Precision, recall and F1-score for all the YOLOv3 model's classes

Table 7.1 displays the YOLOv3 results with the use of evaluation metrics.
Recall indicates what portion of actual positives were correctly predicted. The 'none' metrics can be misleading, as each time no objects were not detected in any of the 125 images, this class would get +1 to false positive. A recall score of 0.84 means that no false positives were predicted in 84% of the images and is a good indicator of how often the model would predict objects that weren't there. Recall also indicates that the model struggles more to detect the classes knife and plate, less so with fork, and is adequate at detecting cups and spoons when the objects are in the image.
Precision indicates what portion of the positive predictions were correct. All classes, apart from 'none', had high scores on this metric. That tells us that when the YOLOv3 model predicts and object, there is a high chance that said prediction is correct. In particular the 'plate' class which were never mistaken for any of the others in the evaluation. The model performance is less impressive on the 'spoon' class.
F1- Score is a combination of precision and recall. This metric indicates that the model performs best on the 'cup' class, which with a score of 0.756 and the rest of the object classes hovered around 6.1 to 6.8, with fork being at the lower end.

The confusion matrix of the YOLOv3 model predictions, as can be seen in figure 7.1 which confirms the previous evaluation metrics for the model. The model produces a healthy amount of True positives for each class.

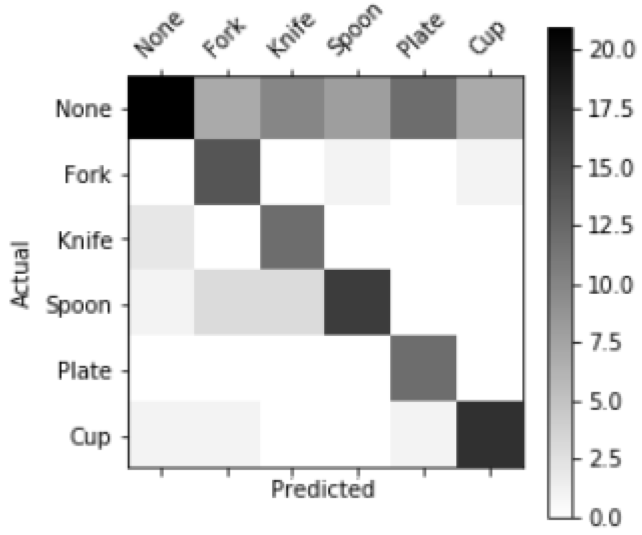


Figure 7.1: A confusion matrix of the YOLOv3 model's classes

The first row on the x-axis indicates false negatives for the classes. One can also see that knife, fork and spoon are mistaken for each other more than the average.

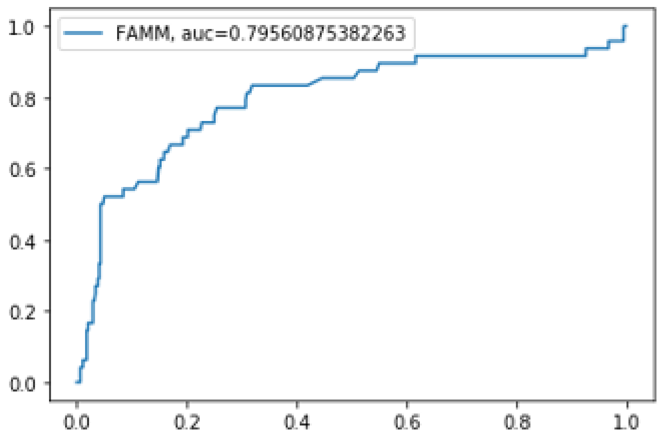
7.2 Feedforward Artificial Neural Network Model Results

In this subchapter, the results of the feed forward neural network model on the test set are outlined.

Table 7.2 below is an overview of the evaluation metrics' scores which are included as indicators of the model's performance

Evaluation metric	**Value**
$Accuracy = \frac{TP + TN}{TP + FP + FN + FP}$	0.93
$Precision = \frac{TP}{TP + FP}$	0.77
$Recall = \frac{TP}{TP + FN}$	0.5
$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$	0.489
AUC-score	0.796
AUC-ROC curve:	Figure 7.2

Table 7.2 Evaluation scores for the FANN model



Graph 7.1: AUROC for the FANN model

Accuracy will naturally be high, as ‘eating moments’ in the lifelog dataset can be identified as an outlier prediction.

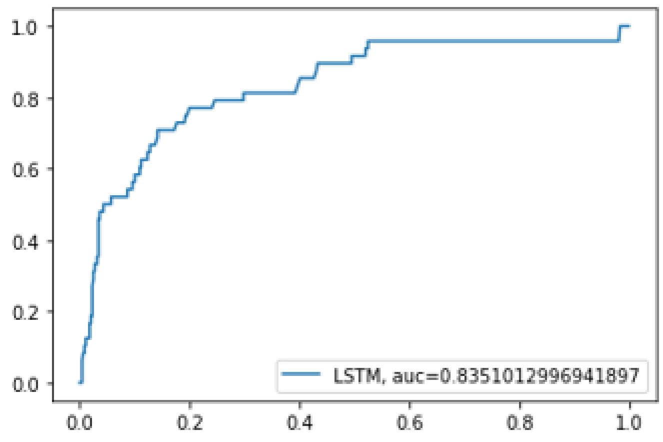
7.3 Long-Short-Term-Memory Artificial Neural Network Model Results

This subchapter details the results of the LSTM model on the test set.

Table 7.3 below is an overview of the evaluation metrics’ scores which are included as indicators of the model’s performance

Evaluation metric	**Value**
$Accuracy = \frac{TP + TN}{TP + FP + FN + FP}$	0.94
$Precision = \frac{TP}{TP + FP}$	0.8125
$Recall = \frac{TP}{TP + FN}$	0.5417
$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$	0.65
AUC score	0.835
AUC-ROC curve:	Figure 7.2

Table 7.3: Evaluation scores for the LSTM model



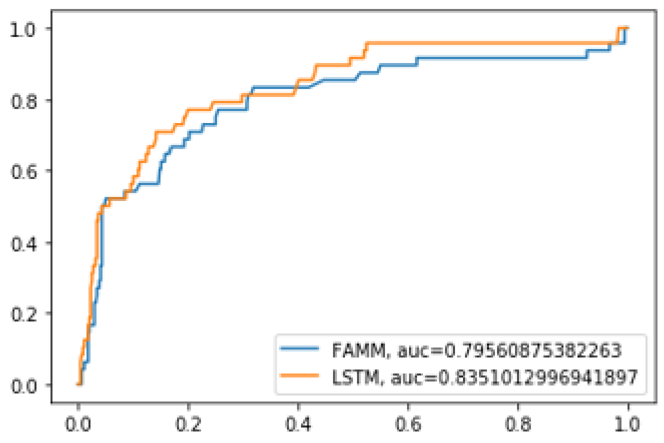
Graph 7.2: AUROC for the LSTM model

7.4 Comparison Between LSTM and FANN models

	Accuracy	**Precision**	**Recall**	**F1-score**	**AUC-score**
FANN	0.93	0.77	0.5	0.489	0.796
LSTM	0.94	0.8125	0.5417	0.65	0.835
Difference	0.01	0.0425	0.0425	0.161	0.039

Table 7.4 Evaluation metrics comparison

The accuracy of both models hovers around 94%. Accuracy is not an adequate or reliable metric of evaluation for classifiers as it is very dependent on the distribution of classes in the dataset [34]. For the task at hand, ‘eating moments’ are much less frequent than any other moments. Had a model not made any ‘eating moment’ predictions, it would still have an accuracy of 90%.



Graph 7.3 AUROC comparison of the LSTM and FANN models

When comparing the two ‘eating moment detection’ models’ predictions there is a 94% of overlap, indicating the models’ share the many of the same strengths and weaknesses.
The LSTM and FANN models’ results are both similar and varied depending on the evaluation metric. Jeni et. al. outlines this in their paper that shows that evaluation metrics can be dramatically skewed in an imbalanced dataset [32]. With a total of 484 rows in the test dataset used for prediction/evaluation, only 48 were labeled as eating moments, which equates to roughly 10%. With this imbalance in distribution, evaluation metrics can be skewed in either directions.
The F-1 score can only be skewed in a ‘positive direction’ i.e. the metric can give the impression of better results than reality [32]. This appears to be the case with the f-1 score when comparing the two models. The LSTM model outperforms the FANN model by 16% in f-1 score, but, if you compare the amount of TP, FN, FP there are only +/-4 in TP/FN and only -1 FP predictions with the LSTM model. The large variance in F-1 score is due to the imbalanced distribution of True Positives and True Negatives.
The AUROC method, however, is mostly unaffected by skew [32]. Taking that into account, it is clear that it is the AUROC and AUC-score which are the best methods for evaluation for this task and are most in line with reality.
the AUROC scores, as seen in graph 7.3, for the models are significantly higher than the F1-score. That indicates that were the models correctly predicts ‘eating moment or ‘non-eating moment’ there is a often a clear margin in prediction scores. This, with the few amount of false positives, indicates that the models are very adept in detecting moments were eating is not occuring. Likely, moments where the lifelogger is sleeping will all have low ‘eating moment’ prediction value as there are no image data and heart rate will be low.

Chapter 8

Discussion

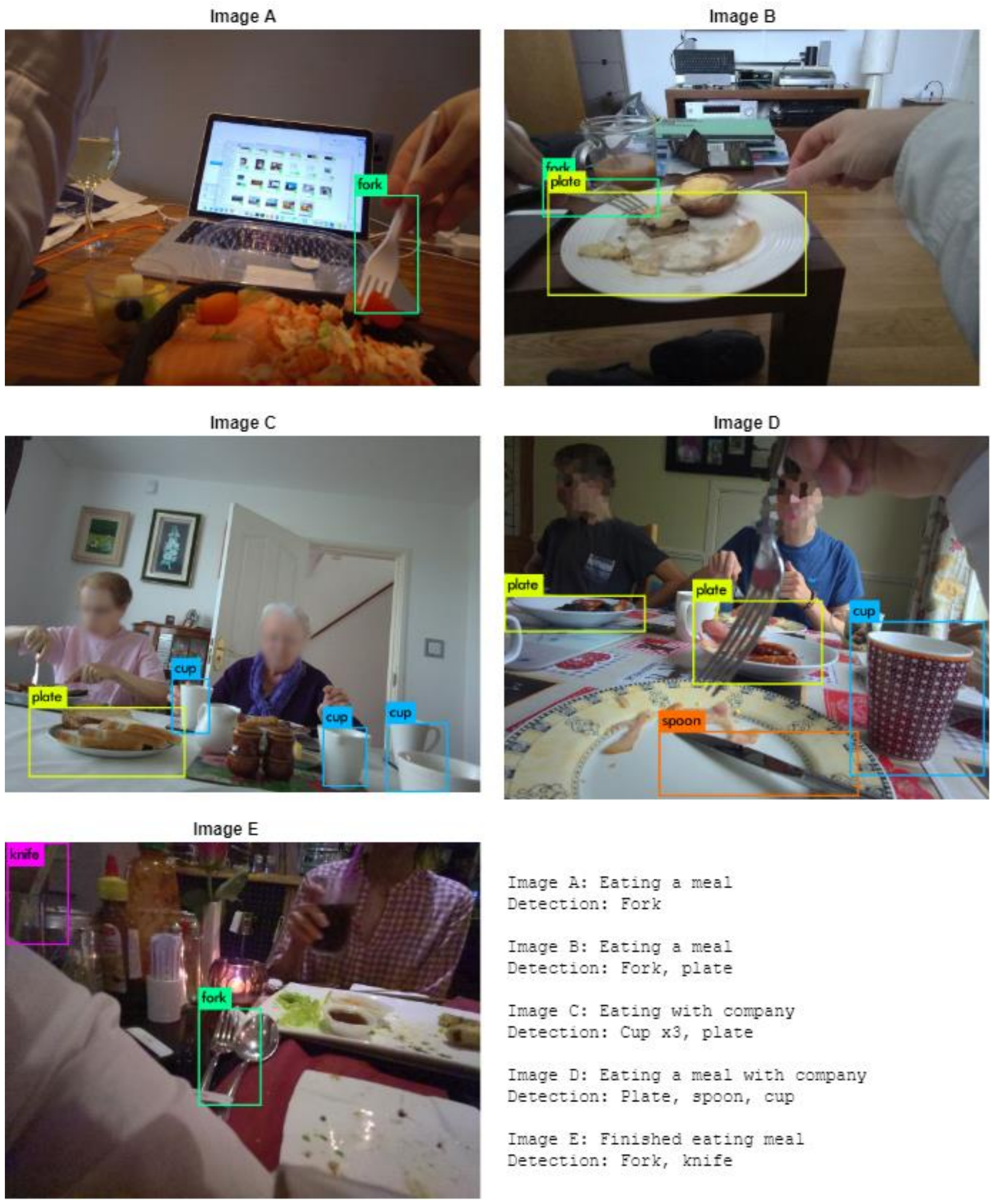
This section will outline the key findings of this research paper and their implications. In addition, the results will be discussed in terms of limitations and interpretations. The chapter also includes the research questions, which will be reiterated and answered. Finally, some recommendations on future work on the YOLO object detection model, 'eating moment detection' NN models and lifelogging research in general.

8.1 Key findings:

This subchapter will discuss the key findings made in this thesis.

8.1.1 Image detection data provide the best features

As alluded to in the previous paragraph, the object detection features appear to be a good indicator for both FANN and LSTM models. A strong correlation of false positive predictions and high object detection scores in the dataset was found, which proves that the models have bias toward the image detection scores when making predictions.
Naturally, there is also a strong correlation between 'eating moment' correct predictions and the image detection scores features. Examples of this are provided in figure 8.1 below which contain images extracted from labeled and predicted 'eating moments'.



```
Image A: Eating a meal
Detection: Fork

Image B: Eating a meal
Detection: Fork, plate

Image C: Eating with company
Detection: Cup x3, plate

Image D: Eating a meal with company
Detection: Plate, spoon, cup

Image E: Finished eating meal
Detection: Fork, knife
```

Figure 8.1: An extract of images taken from various 'eating moments' were several detections were made by the YOLOv3 Model. In order to get the label boxes on the objects, each image had to individually inputted into the YOLOv3 Model

The YOLOv3 model yielded detections on all of the images shown in figure 8.1 though some objects were not detected, while others were predicted as other objects.
Image A and B contain objects (knife and spoon) which were not detected and image D contains a knife predicted as a spoon. The task of distinguishing of cutlery objects was a difficult task for the YOLOv3 model as they are often similar in material and shape, and the results of the YOLOv3 model in the result chapter confirms this. False positive object detections will naturally increase the false positive 'eating moment' predictions. However, in

some cases they can also help increase true positive ‘eating moment’ predictions where the false positive object detection overlaps with an actual ‘eating moment’.

The other features included as input for the FANN and LSTM models, are either situational or have limited impact.

- Heart rate: this feature has slightly higher mean value of (+0.0109) where ‘eating moments are labeled True.
- Activity: these features has a negative correlation of -0.0996 indicating that when the value of any of the activity features are != 1 there is a slightly higher chance of an ‘eating moment’ occurring. As show in ‘Image C’ in figure 8.1 below, ‘eating moments’ occur while the lifelogger is walking i.e. this feature could make such moments even harder to detect.
- Location: The various location features are situational. Many locations are only visited once or twice, which would mean if said location only occurred during the training dataset, it would only negatively influence the predictions on the test dataset. Other locations like ‘home’, ‘work’ and frequently visited cafes/restaurants were decent indicators, but in most cases did not provide enough information to indicate ‘eating moments’ by themselves.

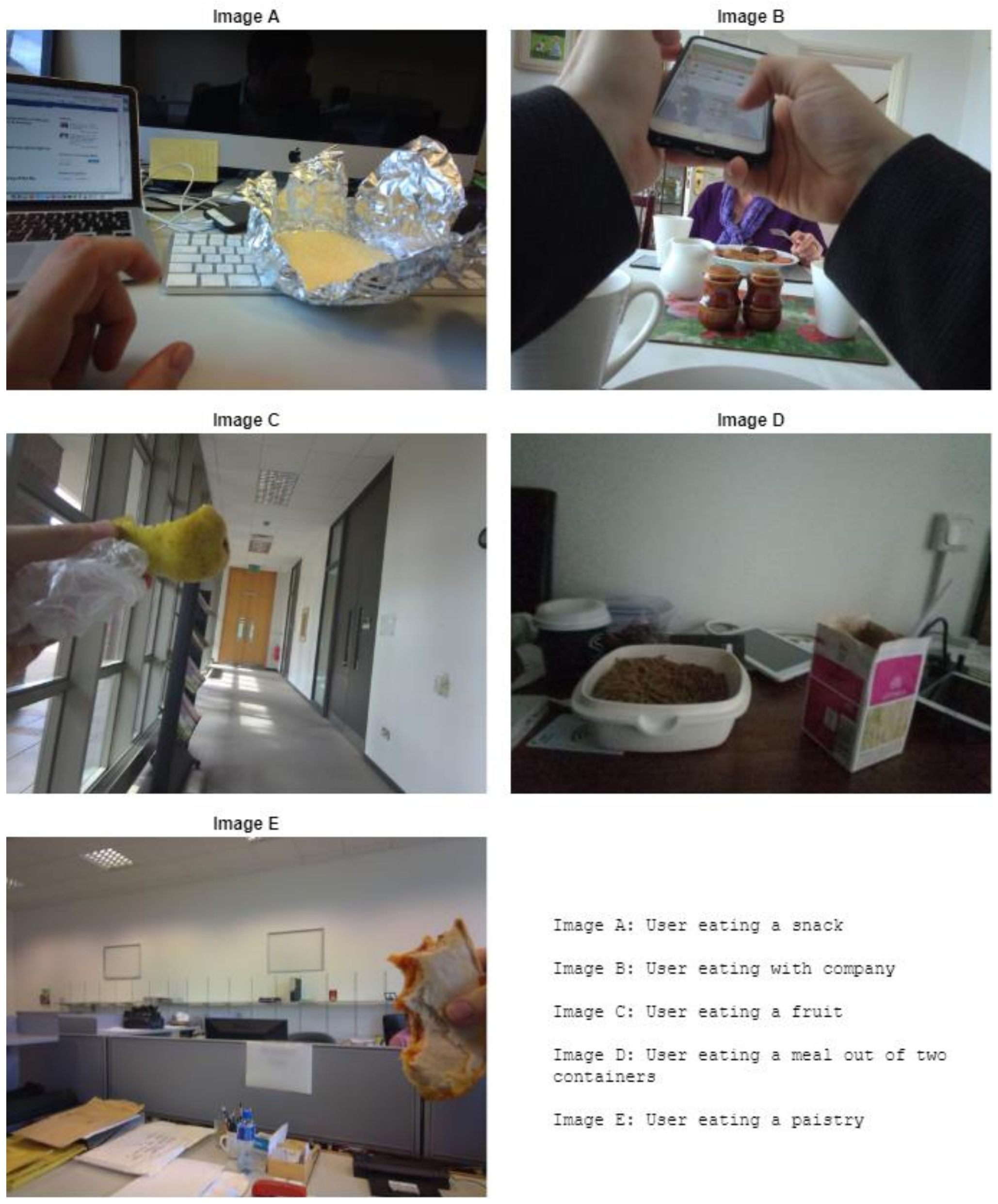
8.1.2 The LSTM Model’s Forecasting Ability Provide Best Results

It can be hard to differentiate the reasons behind the variances in results between the FANN and LSTM models. It is feasible that the variance is due to the different type of NN model or it could be difference in weights / biases obtained by each model during training.
At first glance, the difference between the FANN and LSTM model can be viewed as minor or even insignificant. However, a closer look into the actual predictions reveal a strength of the LSTM model compared to its counterpart. It can ‘read between the lines’ so to speak. Where the FANN model predicted 1,1,0,1 i.e. three ‘eating moments’ and one ‘non-eating moment’ separating them, the LSTM model predicted 1,1,1,1, which was the correct prediction according to the labels. This can be contributed to LSTM models’, in general, ability to use previous input’s predictions as information when calculating its current prediction [16]. In other words, the LSTM models has learned that after a ‘eating moment’ it is more likely that the next moment also contains ‘eating’, and thus made the correct prediction. Even though this isn’t significantly reflected in the evaluation metrics, it could prove to be significant on other datasets.

8.1.3 YOLOv3 model shortcomings potentially reflected in results.

As previously discussed, the image detection data was highly influential in correct predictions of an ‘eating moment’. With that in mind, the YOLO model was only trained to detect various cutlery e.g. plate, fork, knife etc. This would mean that food like fruit, fast food or beverages contained in bottles (to name a few) would not provide any visual detection data for the models. Thus, depending on what types of food and beverages the lifelogger consumed during the time period of which the lifelog data was gathered, evaluation metrics could be inflated or deflated. Had the lifelogger consumed a larger amount of fast food or fruit for an extended period of time, it is safe to assume that the NN models would not perform to the level they currently are.

One example of this can be seen in the figure 8.2 ‘Image C’ below, where the user is eating an pear. This image was extracted from the lifelog where ‘eating moment’ was labeled, but neither the NN models nor the YOLO model made any detections. More examples of similar occurrences are also visualized in figure 8.2



```
Image A: User eating a snack

Image B: User eating with company

Image C: User eating a fruit

Image D: User eating a meal out of two
containers

Image E: User eating a paistry
```

Figure 8.2:

An extract of images taken from various 'eating moments' were no object detections were made by the YOLOv3 Model.

Moreover, false positive, even though few, were on 3 predictions located near to true positives, indicating that meal preparation could be the reasoning for the false positive. Objects that the YOLO model were trained to detect are also frequently used in meal preparation as well as consumption. Differentiating the two using images could be a challenging task in some cases even for humans. Two examples of said occurrence can be seen in figure 8.3.

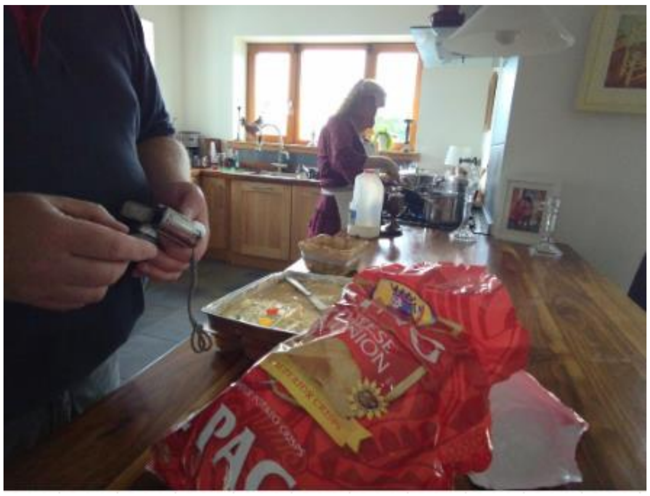


Figure 8.3.

A image extracted from moments which were predicted as ‘eating moments, but labeled as not.

Another possible reason the false positives could be incorrect labeling i.e. that a ‘eating moment’ is in fact taking place, but i labeled as ‘not a eating moment’. That said, it appears that both models are dealing with said problem in manner which provide an acceptable number of false positives. However, this could be a problem for other event detection models where the activity is not an outlier, where false positives are more common.

In addition, the performance of the YOLO model on the objects it was trained to detect have room for improvement. The poorer performance of the ‘knife’ class is not unexpected (table x), as the class had much less training date compared to the others. The similar performance of knife, fork and spoon makes since as they are similar objects, which also makes them hard to distinguish from one another. The model struggles with the detection of the ‘plate’ class. This could be due to food covering the plate, as can be seen in figure 8.4 below.

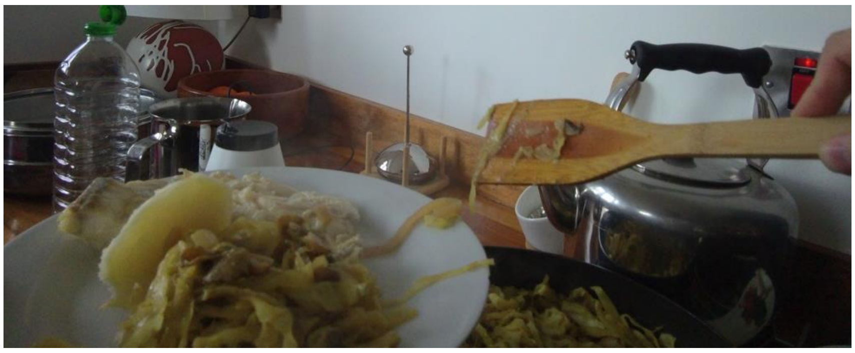


Figure 8.4

A image of a plate, which was not detected by the YOLOv3 model

The training data for plate was also not optimal, as many of the images gathered from Google Open Images were of colorful plates and taken from a top down perspective. This could have affected the models ability to detect plates in lifelog images. Cup was the most successful class, and this is reflected throughout the annotated lifelog. This fact would indicate that a better performing object detector would significantly improve the FANN and LSTM models' probability to detect 'eating moments'

8.1.4 The longer the eating moment, the higher chance of detection

Looking at the prediction list makes it clear that the longer the eating moment last, the higher the chance of detection. This fact was discovered when manually comparing both model's predictions with the labels and with the dataset in its not shrunken form. Most false positives were short eating moments (15 minutes or less) or at the start of a longer' eating moment'. The reasoning for this is that there are more data to indicate that a 'eating moment' is in fact taking place. To elaborate, multiple continues labeled 'eating moments' i.e. (1,1,1), are larger meals where the lifelogger is taking his/her time with it. The same logic can be applied to moments where the lifelogger is drinking coffee, a beverage which is normally consumed at a slower pace. These types of eating activities will provide a greater volume of data that indicates higher probability of a 'eating moment' occurring. Comparing that to a singular 'eating moment' i.e. (0,1,0), could be up to 15 minutes long, but also could be only a few minutes of the user grabbing a snack. In addition, one can assume that larger meals have a higher chance of using any of the objects the YOLO object detection model is trained to detect i.e. fork, knife plate etc. An example of this is provided in figure 8.2 where in 'Image B' potential object detection could have been made on cups and/or plate by the YOLOv3 model, but none were.

8.2 Limitations:

8.2.1 Dataset Sample Size

Sample size of the test dataset is potentially a factor that could influence results. With only 48 labeled 'eating moments' it is not only the distribution that will affect results, but also sample size. The results are dependent on what type of eating moments that were included in the test dataset. Additionally, a larger amount of labeled data would also provide the FANN and LSTM with more data to train on. Additional training data could provide better results, particular so for the LSTM model which overfitted quite rapidly. More training data will help models, in general, to generalize better [14].

8.2.3 Computation power

Computational power has been a bottleneck for deep learning since the field was discovered. Nowadays, smaller NNs are possible to compute using a mid-range laptop. However, NN used in computer vision tasks require a significant larger amount computational power [14, p.20]. YOLOv3 are much faster than traditional CNNs, but are still expensive [23]. This limitation made it difficult to experiment with different classes, training data when training to YOLOv3 model.

8.2.4 Labeled image data

As alluded to in the discussion section 5.3., the lack of available labeled image data has affected the results of the YOLOv3 model, and in turn also affected the ‘eating moment’ detection models. The manual labeling of the YOLOv3 model classes in lifelog images was a cumbersome process. There were many images containing one or more of the classes in the lifelog dataset. However, they had to be manually located in the and there was also an active intent to avoid annotating similar images to avoid overfitting. In addition, none of the images from the time period of which the ‘eating moment’ labeling were available were used as that would compromise the results of training or testing. The ‘Google Open Images’ were also no ideal to train the YOLOv3 model on as they were visually different from the images in the lifelog i.e. angle, shapes, obstructions.

8.3 Implications:

8.3.1 Providing Lifelogger with Historical Nutrition Data through a UI

LifeSeeker, as outlined in section 2.1 is a lifelog search engine. The ‘eating moment’ annotations, now annotated to the lifelog dataset, provided by the LSTM or FANN models can be leveraged as a unit of retrieval [5]. This means that the ‘eating moments’ can be located using LifeSeeker with the use of its GUI, as seen in figure 8.5. With LifeSeeker the user can filter the search to be within a specific time period, location and more.

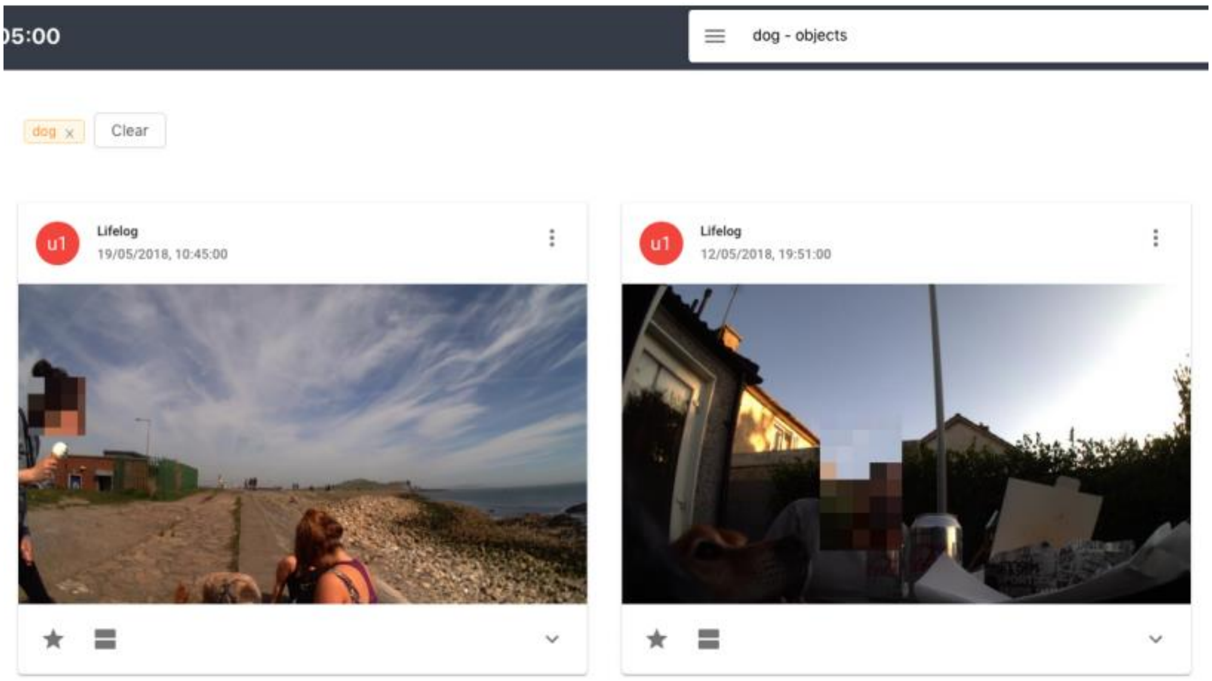


Figure 8.5

LifeSeeker extraction of images to the query of ‘dogs’

The images seen in figure 8.5 are also clickable and grants access to additional contextual information extracted from the lifelog.

In subchapter 3.2, the difficulties of scalability with the types of models created in this paper was observed. Nonetheless, accessibility of AI frameworks like YOLO [22] and WEKA [36] enables a larger population the ability to leverage AI technologies. This entails that it is not unreasonable for lifeloggers to develop and train their own ‘eating moment’ detection models, following methods like the ones outlined in this paper and/or other resources, despite a limited knowledge of the technologies.

- its still somewhat applicable to others

The methods applied in this master thesis can empower lifeloggers with a tool capable of providing great value in form of nutrition-based data and there are many individuals who in some manner could benefit from this. These types of individuals could range from those affected by diseases that require a healthy nutrition, to individuals who want to prevent such diseases from occurring. As discussed in section 2.2.6, obesity is a growing worldwide issue. Individual historical nutrition data could be analyzed by professionals to predict the health status of a person in the future, and from there work maintain or get back to a healthy nutrition. Additionally, increased knowledge about one's nutrition habits could be of interest to athletic performers.

8.4 Future work:

8.4.1 Derive Semantic Location Data to Enrich the Lifelog

There are several ways to derive secondary data from the initial lifelog data which is stored. One type of secondary data that should be derived is semantic location information from Google Knowledge Graph Search API, i.e. Burger King is a type of 'fast food restaurant' [37]. In the subchapter [8.1.2] it was discussed how the 'eating moment' detection models would have low probability of making a detection when the related images did not contain any of the YOLov3 model object classes. With the inferred information from location data, the detection capabilities of the models would undoubtedly increase, as it is safe to assume that longer periods of time spent in a restaurant would entail a 'eating moment'' was taking place. The same would apply to bars, coffee shops and cafeterias.
This inferred information would also be of great benefits to other event detections where the user is visiting locations related to activities e.g. a gym location would mean the individual could be working out.

8.4.2 Create a Lifelog Image Annotated Database

As mentioned at several points in this thesis, lifelog images are different than regular images. They are also one of the key data sources to make a lifelog progressively more useful. Lifelog images should, before being stored, go through processes which protects privacy of individuals who otherwise would be identified in them [38]. Lifelogging generates a decent amount of visual data i.e. 1500 images each day. A good portion of the images gathered for a lifelog are of mundane and everyday events due to the non-intrusive image gathering method. Such images could be hard to acquire otherwise, but can be of value to research attempting to bridge the gap between 'cyberspace' and the physical environment. With that in mind, the introduction of an image database consisting of annotated images is something that would benefit future event detection methods that will rely on images. The images could be annotated with the use of multipurpose object detection models like YOLO [22] and/or Mask R-CNN [39]. For privacy and protection concerns, the database would have to be password protected and only be made available for approved research projects. In addition, all activity should be logged.
The images could be used to train a new CNN or to customize a pre-trained network.

8.4.3 Calorie counter and personal food recommender

By combining the methods and models introduced in this thesis with the work of Liang et al. [42], outlined in subchapter [2.1.5] the ‘eating moments’ located could be further leveraged to produce summary reports of calorie intake. Furthermore, a personal food recommender system could also be developed applied to further increase the value to the lifelogger as suggested in [6].

Chapter 9

Conclusion

This chapter will summarize the achieved goals, contributions and results introduced this thesis.

9.1 Conclusion

In this thesis it was proposed a novel way to detect ‘eating moments’ in a lifelog dataset by leveraging several deep learning technologies and data processing. The best performing ANN model on this task was a LSTM model with an F1-score of 0.65. In addition, using YOLOv3 an object detection model was introduced, which is trained to detects several commonplace objects related to eating.
Through the work introduced in this thesis, it has been revealed that image data are key in detecting complex context events. However, lifelog images are different compared to most images available online, due to the pervasive nature in which they are gathered. Therefore, in order to streamline the creation of event detection models for lifelogs, a image database consisting of lifelog images should be introduced for future research projects.
Furthermore, location data in lifelogs should be expanded upon by inferring data from APIs that are available. This would lead to an increased amount of metadata that will aid in future research and provide value to lifeloggers with more units of retrieval.

Through the use of the models introduced in this thesis, the lifelogger can extract ‘eating moments’ from his/her lifelog which will provide historical nutrition data in form of images. With the use food calorie estimation method [42] additional nutritional information can also be extracted. The models and methods introduced would also allow other lifeloggers to apply them to their lifelog, though likely with poorer results. However, they could be altered to fit the new user’s data.

Literature

[1] Martin Dodge and Rob Kitchin. 2007. ‘Outlines of a World Coming into Existence’: Pervasive Computing and the Ethics of Forgetting. Environ. Plan. B Plan. Des. 34, 3 (June 2007), 431–445. DOI:https://doi.org/10.1068/b32041t

[2] Pedro Herruzo, Laura Portell, Alberto Soto, and Beatriz Remeseiro. 2017. Analyzing First-Person Stories Based on Socializing, Eating and Sedentary Patterns. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 109–119. DOI:https://doi.org/10.1007/978-3-319-70742-6_10

[3] Liting Zhou, Duc Tien Dang-Nguyen, and Cathal Gurrin. 2017. A baseline search engine for personal life archives. In LTA 2017 - Proceedings of the 2nd Workshop on Lifelogging Tools and Applications, co-located with MM 2017, 21–24. DOI:https://doi.org/10.1145/3133202.3133206

[4] Gurrin, C., Smeaton, A. F., & Doherty, A. R. (2014). LifeLogging: Personal Big Data. Foundations and Trends® in Information Retrieval, 8(1), 1–125. https://doi.org/10.1561/1500000033

[5] Helen Williams. Williams, H.L., Conway, M.A. &amp; Cohen, G. (2008). Autobiographical Memory. In G. Cohen &amp; M.A. Conway (eds.), Memory in the Real World (3rd Edition) London: Psychology Press. pp. 21-90. Retrieved May 16, 2019 from https://www.academia.edu/2955232/Williams_H.L._Conway_M.A._and_Cohen_G._2008_._Autobiographical_Memory._In_G._Cohen_and_M.A._Conway_eds._Memory_in_the_Real_World_3rd_Edition_London_Psychology_Press._pp._21-90

[6] Binh T. Nguyen, Duc Tien Dang-Nguyen, Tien X. Dang, Thai Phat, and Cathal Gurrin. 2018. A Deep Learning based Food Recognition System for Lifelog Images. In ICPRAM 2018 - Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods, 657–664. DOI:https://doi.org/10.5220/0006749006570664

[7] WHO | Controlling the global obesity epidemic. Retrieved June 8, 2020 from https://www.who.int/nutrition/topics/obesity/en/

[8] Maximilian Tremmel, Ulf G. Gerdtham, Peter M. Nilsson, and Sanjib Saha. 2017. Economic burden of obesity: A systematic literature review. International Journal of Environmental Research and Public Health 14. DOI:https://doi.org/10.3390/ijerph14040435

[9] Jindong Liu, Edward Johns, Louis Atallah, Claire Pettitt, Benny Lo, Gary Frost, and Guang Zhong Yang. 2012. An intelligent food-intake monitoring system using wearable sensors. In Proceedings - BSN 2012: 9th International Workshop on Wearable and Implantable Body Sensor Networks, 154–160. DOI:https://doi.org/10.1109/BSN.2012.11

[11] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I. Jordan. 2003. An introduction to MCMC for machine learning. Mach. Learn. 50, 1–2 (January 2003), 5–43. DOI:https://doi.org/10.1023/A:1020281327116

[12] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. Nature 521, 436–444. DOI:https://doi.org/10.1038/nature14539

[13] Pedro Strecht, Luís Cruz, Carlos Soares, João Mendes-Moreira, and Rui Abreu. A Comparative Study of Classification and Regression Algorithms for Modelling Students' Academic Performance.

[14] Francois Chollet. 2018. Deep Learning with Python & Keras. Manning Publications Co. DOI:https://doi.org/citeulike-article-id:10054678

[15] Nielsen, M. (n.d.). Neural Networks and Deep Learning. Retrieved from http://neuralnetworksanddeeplearning.com

[16] Jason Brownlee Disclaimer. 2017. Long Short-Term Memory Networks With Python Develop Sequence Prediction Models With Deep Learning.

[17] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211–252. http://doi.org/10.1007/s11263-015-0816-y

[18] 2006. Why Question Machine Learning Evaluation Methods (An Illustrative Review of the Shortcomings of Current Methods). Retrieved June 8, 2020 from https://www.aaai.org/Library/Workshops/2006/ws06-06-003.php

[19] O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. Retrieved from http://arxiv.org/abs/1511.08458

[20] NTCIR14 - Lifelog Data Description . Retrieved June 8, 2020 from http://ntcir-lifelog.computing.dcu.ie/data/index.html

[21] Joseph Redmon (2013-2016). Darknet: Open Source Neural Networks in C. Retrieved June, 8 from http://pjreddie.com/darknet

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. Cvpr (June 2015). Retrieved June 8, 2020 from http://arxiv.org/abs/1506.02640

[23] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. (April 2018). Retrieved June 8, 2020 from http://arxiv.org/abs/1804.02767

[24] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. 248–255. DOI:https://doi.org/10.1109/cvprw.2009.5206848

[25] Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. Comput. Electr. Eng. 40, 1 (January 2014), 16–28. DOI:https://doi.org/10.1016/j.compeleceng.2013.11.024

[26] J. D. Armstrong. 1986. Heart rate as an indicator of activity, metabolic rate, food intake and digestion in pike, Esox lucius. J. Fish Biol. 29, (1986), 207–221. DOI:https://doi.org/10.1111/j.1095-8649.1986.tb05012.x

[27] S.Gopal Krishna Patro and Kishore Kumar sahu. 2015. Normalization: A Preprocessing Stage. IARJSET (March 2015), 20–22. DOI:https://doi.org/10.17148/iarjset.2015.2305

[28] S B Kotsiantis and D Kanellopoulos. 2006. Data preprocessing for supervised learning. Int. J. … 1, 2 (2006), 1–7. DOI:https://doi.org/10.1080/02331931003692557

[29] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. 2018. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. (November 2018). Retrieved June 9, 2020 from http://arxiv.org/abs/1811.03378

[30] Katarzyna Janocha and Wojciech Marian Czarnecki. 2017. On Loss Functions for Deep Neural Networks in Classification. Schedae Informaticae 25, (February 2017), 49–59. Retrieved June 9, 2020 from http://arxiv.org/abs/1702.05659

[31] Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings.

[32] László A. Jeni, Jeffrey F. Cohn, and Fernando De La Torre. 2013. Facing imbalanced data - Recommendations for the use of performance metrics. In Proceedings - 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, ACII 2013, 245–251. DOI:https://doi.org/10.1109/ACII.2013.47

[33] Andrew P. Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit. 30, 7 (July 1997), 1145–1159. DOI:https://doi.org/10.1016/S0031-3203(96)00142-2

[34] Jasmina Dj. Novaković, Alempije Veljović, Siniša S. Ilić, Željko Papić, and Tomović Milica. 2017. Evaluation of Classification Models in Machine Learning | Theory and Applications of Mathematics & Computer Science. Editura Universit♯¿¿Đii " Aurel Vlaicu " Arad. Retrieved June 11, 2020 from https://www.uav.ro/applications/se/journal/index.php/TAMCS/article/view/158

[35] Shinji Hotta, Tatsuya Mori, Daisuke Uchida, Kazuho Maeda, Yoshinori Yaginuma, and Akihiro Inomata. 2017. Eating moment recognition using heart rate responses. In UbiComp/ISWC 2017 - Adjunct Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers, 69–72. DOI:https://doi.org/10.1145/3123024.3123093

[36] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software. ACM SIGKDD Explor. Newsl. 11, 1 (November 2009), 10–18. DOI:https://doi.org/10.1145/1656274.1656278

[37] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 601–610. DOI:https://doi.org/10.1145/2623330.2623623

[38] Duc Tien Dang-Nguyen, Liting Zhou, Rashmi Gupta, Cathal Gurrin, and Michael Riegler. 2017. Building a disclosed lifelog dataset: Challenges, principles and processes. In ACM International Conference Proceeding Series, 1–6. DOI:https://doi.org/10.1145/3095713.3095736

[39] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2020. Mask R-CNN. IEEE Trans. Pattern Anal. Mach. Intell. 42, 2 (February 2020), 386–397. DOI:https://doi.org/10.1109/TPAMI.2018.2844175

[40] Thilo Hagendorff and Katharina Wezel. 2019. 15 challenges for AI: or what AI (currently) can't do. AI Soc. 35, 2 (March 2019), 355–365. DOI:https://doi.org/10.1007/s00146-019-00886-y

[41] Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. Inf. Process. Manag. 45, 4 (July 2009), 427–437. DOI:https://doi.org/10.1016/j.ipm.2009.03.002

[42] Yanchao Liang and Jianhua Li. 2017. Deep Learning-Based Food Calorie Estimation Method in Dietary Assessment. (June 2017). Retrieved June 14, 2020 from http://arxiv.org/abs/1706.04062

[43] Confusion Matrix - an overview | ScienceDirect Topics. Retrieved June 15, 2020 from https://www.sciencedirect.com/topics/engineering/confusion-matrix

[44] Understanding AUC - ROC Curve - Towards Data Science. Retrieved June 15, 2020 from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5