
**BRNO UNIVERSITY OF TECHNOLOGY**
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**FACULTY OF INFORMATION TECHNOLOGY**
**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**
**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**


# AUTOMATIC SPEECH DETECTION FOR VHF CHANNEL

**AUTOMATICKÁ SEGMENTACE ŘEČI PRO VHF KANÁL**

**BACHELOR'S THESIS**
**BAKALÁŘSKÁ PRÁCE**


**AUTHOR** **MÁRIA NOVÁKOVÁ**
**AUTOR PRÁCE**

**SUPERVISOR** **Ing. IGOR SZŐKE, Ph.D.**
**VEDOUCÍ PRÁCE**



**BRNO 2023**

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

# Bachelor's Thesis Assignment

140508

| | |
|---|---|
| Institut: | Department of Computer Graphics and Multimedia (UPGM) |
| Student: | **Nováková Mária** |
| Programme: | Information Technology |
| Specialization: | Information Technology |
| Title: | **Automatic Speech Detection for VHF Channel** |
| Category: | Signal Processing |
| Academic year: | 2022/23 |

Assignment:

1. Get familiar with very high frequency channel (VHF), artificial neural networks (NN), and data augmentation.
2. Propose a NN architecture for automatic voice detection and push-to-talk (PTT) detection on provided data.
3. Augment the training data, train and evaluate the NN.
4. Experiment with various approaches and NN architectures. Track the accuracy gains.
5. Draw a conclusion and propose further approaches.
6. Create A2 poster or ~30 seconds long video presenting your work.

Literature:

- Park, D.S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E.D., Le, Q.V. (2019) SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. Proc. Interspeech 2019, 2613-2617
- Pellegrini, T., Farinas, J., Delpech, E., Lancelot, F. (2019) The Airbus Air Traffic Control Speech Recognition 2018 Challenge: Towards ATC Automatic Transcription and Call Sign Detection. Proc. Interspeech 2019, 2993-2997
- ATCO2 project, http://atco2.org
- According to supervisor's recommendation.

Requirements for the semestral defence:
Steps 1 - 3 of the assignment.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

| | |
|---|---|
| Supervisor: | **Szőke Igor, Ing., Ph.D.** |
| Head of Department: | Černocký Jan, prof. Dr. Ing. |
| Beginning of work: | 1.11.2022 |
| Submission deadline: | 10.5.2023 |
| Approval date: | 31.10.2022 |

# Abstract

A noisy environment in air traffic communication is an unavoidable problem. The communication between the control tower and the pilot should be the most reliable and effective. That is why voice activity detection is crucial for recognising the start of the speech segment of the communicants for automated systems. The speakers take turns providing information by pressing the push-to-talk button. To detect voice activity, various approaches are used. Even though these methods are effective, machine learning can easily outshine them. Neural networks are widely used in voice activity detection as well as in other areas. Properly trained models are efficient and adaptable. In this thesis, a solution for voice activity detection together with push-to-talk detection is proposed. Proposed models are evaluated and compared. The adaptation of the GPVAD approach is discussed and compared to the proposed models. Neural networks will have their chance to once again prove that they are suitable for any task.



# Abstrakt

Výskyt hluku a šumu v pozadí audio leteckej komunikácie je problémom, ktorému denne čelia operanti riadenia letovej prevádzky. Aby bola zaistená bezpečná letecká preprava, komunikácia medzi vežou a lietatlom musí byť čo najefektívnejšia. Hlavnú rolu vo vylepšovaní kvality komunikácie hrá detekcia hlasovej aktivity. Správna detekcia reči je nevyhnutá pre rozpoznanie začiatku komunikácie pre systémy. Začiatok komunikácie začína stlačením tlačítka push-to-talk pomocou rádiového systému. Na rozpoznávanie reči existujú rôzne prístupy a implementácie. Za pomoci neurónových sietí sa dá detekcia reči upresniť. Výhodou používania umelej inteligencie je jej adaptácia na nové podnety. Táto práca ponúka riešenie na detekciu reči a push-to-talk udalostí v leteckej komunikácií. Navrhnuté riešenia budú evaluované a porovnané. Na záver, dostupná implementácia GPVAD je prepracovaná na riešenie tohto problému. Strojové učenie má zas a znova príležitosť predviesť svoje schopnosti.



# Keywords

voice activity detection, push-to-talk, very high-frequency channel, python, air traffic control, artificial neural networks, deep learning, convolutional neural networks, convolutional recurrent neural networks, data augmentation, annotation

# Kľúčové slová

detekcia hlasovej aktivity, push-to-talk, veľmi vysokofrekvenčný kanál, python, riadenie letovej prevádzky, umelé neurónové siete, hlboké učenie, konvolučné neurónové siete, konvolučné rekurentné neurónové siete, augmentácia dát, anotácia


# Reference

NOVÁKOVÁ, Mária. *Automatic Speech Detection for VHF Channel*. Brno, 2023. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Igor Szőke, Ph.D.

# Rozšírený abstrakt

Riadenie letovej prevádzky je služba, ktorá poskytuje všetky dôležité operácie na to, aby chod vzdušnej prepravy bol plynulý. Táto služba je poskytovaná pracovníkmi v kontrolných vežiach, ktorí majú na starosti správnu koordináciu pilotov v lietadlách. Poskytujú inštrukcie, aby nedošlo ku kolíziam alebo inému nebezpečenstvu.

Táto služba by nebola zrealizovateľná bez efektívnej a rýchlej leteckej komunikácie s pilotmi. Veža s lietadlom komunikuje pomocou rádiových systémov na veľmi vysokých frekvenciách. Tento spôsob komunikácie je najpoužívanejší a funguje na základe sieti vysielačov a prijímačov. Vyhradené frekvenčné pásmo pre túto komunikáciu je od 118.000 do 136.975 Megahertzov. Pilot si v lietadle naladí správnu frekvenciu na cieľového komunikanta.

Metóda komunikácie cez tieto rádiové systémy je efektívna a pomáha zabezpečovať bezpečnosť leteckej premávky.

Aby bolo možné cez tieto systémy správne komunikovať, je potrebné s nimi vedieť detekovať začiatok komunikácie. Začiatok komunikácie v leteckej doprave je typicky charakterizovaný ako stlačenie push-to-talk tlačítkam, ktoré povolí komunikantovi zdieľať správy.

Stlačením tlačítka sa komunikujúci dostáva do prenosového režimu a môže začať rozprávať. V rovnakom čase je príjemca zbavený možnosti komunikovať a môže správu iba prijímuť. Po pustení tlačítka sa znova systém dostáva do stavu čakania na príjem správy.

Existujú rôzne metódy a prístupy k detekcií rečových segmentov. Niektoré z týchto metód dokážu detekovať reč aj vo veľmi zašumenom prostredí.

Hlavným cieľom je s pomocou ATCO2 projektu, ktorý sa zaoberá zbieraním a spracovaním leteckej komunikácie, je vytvoriť detektor reči a push-to-talk signálu. Tento detektor môže dopomôcť k správnej segmentácií a spracovaniu leteckej komunikácie. Schopnosti takéhoto detektoru môžu byť vylepšené za pomoci strojového učenia.

Neurónové siete predvádzajú svoje schopnosti v rôznych odvetviach a je bežne zaužívané ich využívať aj pri detekcií rečovej aktivity. Pozostávajú z navzájom prepojených vrstiev, ktoré transformujú vstup na požadovaný výstup. Na to, aby sa pomocou umelej inteligencie dali detekovať udalosti v leteckej komunikácií je nevyhnutné mať veľa rôznych dát, na ktorých sa môže sieť učiť.

Dátová sada na riešenie tohoto problému bola poskytnutá a vytvorená ATCO2 projektom. Nahrávky v dátovej sade obsahujú 4 hodiny leteckej komunikácie z rôznych letísk.

Poskytnutá sada obsahuje zároveň aj manuálne prepisy rečových udalostí. Tieto prepisy dokážu zefektívniť proces trénovania neurónovej siete.

Proces trénovania neurónových sietí sa dá zjednodušene vyjadriť ako riešenie optimalizačného problému. Na to, aby takéto siete boli schopné vykonávať, čo sa od nich žiada je potrebné, aby si správne nastavili svoje parametre. Tieto parametre sú vypočítané na základe stratovej funkcie, ktorá vypočíta, aký veľký je rozdiel medzi predpoveďou siete a očakávaným výstupom.

Očakávaným výstupom v kontexte tejto práce bude prestavovať predikcia rečového segmentu alebo push-to-talk udalosti. Kedže očakávané hodnoty pre detekciu reči boli poskytnuté v ATCO2 datasete, jediným problémom ostáva správne označiť push-to-talk udalosti.

Algorithmov na detekciu tejto udalosti je veľmi málo. Kedže sa push-to-talk udalosť v tomto kontexte bude definovať ako náhla silná intenzita signálu, ktorá trvá po dobu 100 milisekúnd, je možné pre túto udalosť definovať dočasný detektor, ktorý poslúži k automatizácií procesu vytvárania očakávaných výstupov.

Navrhnutý algorithmus, ktorý spočíva v hľadaní lokálnych extrémov, dokáže efektívne detekovať udalosti. Spôsob detekcie udalosti je založený na hľadaní extrémov a posudzovaní,

či spĺňajú jednotlivé podmienky. Podmienka, ktorá kontroluje prechod cez nulovú hodnotu po dobu 100 milisekúnd a ďalšia pomienka, ktorá dodatočne kontroluje či je lokalizovaný extrém skutočne extrémom vo svojom najbližšom okolí, vylučujú prebytočné extrémy, ktoré push-to-talk udalosti nereprezentujú.

Akonáhle sú dáta spracované, je možné začať navrchovať architektúry sietí. V tejto práci sa používajú dve vlastne navrhnuté architektúry, ktoré sú trénované na detekciu rečových segmentov a push-to-talk udalostí. Ich zmysel spočíva aj v tom, že ďalším cieľom tejto práce bolo nadobudnúť širší obzor o téme neurónových sietí.

Tieto siete su trénované na datasete, ktorý je rôzne rozdelený, aby spľňoval očakávaný cieľ. Tri experimenty pre jednoduchú doprednú neurónovu sieť a jeden pre konvolúčnu neurónovú sieť budú vysvetlené a porovnané.

Jednoduchá neurónová sieť, bola trénovaná jednotlivo na detekciu rečovej aktivity, push-to-talk udalosti a ich kombinácií. Prvý experiment spočíval v trénovaní neurónovej siete na detekciu iba rečovej aktivity. Bol trénovaný približne na 25 minutách a po piatich iteráciách vedel model správne predpovedať rečové segmenty s 72% úspešnosťou. Druhý experiment spočíval v trénovaní siete na push-to-talk detekciu a s datasetom o pätnástich minútach vedel správne detekovať udalosti s 96.6% správnosťou. Posledným experimentom bolo trénovanie siete na obe úlohy naraz, trénovanie skončilo po troch iteráciach na vyvolanie regularizačného prvku "skoré ukončenie". Výsledky tohto trénovania sú 64.88% pre detekciu reči a 94.10% pre detekciu push-to-talk udalosti.

Posledný experiment bol trénovaný na dátovej sate s distribúciou medzi jednotlivými triedami 50% a 50% pre rečovú aktivitu a z celého datasetu 20% reprezentovalo push-to-talk udalosť. Dataset bol o veľkosti približne 25 minút.

Konvolučná neurónová sieť bola trénovaná iba na detekciu oboch udalostí naraz na rovnakom datasete ako tretí predstavený experiemnt.

Výsledok trénovania je 96.88% pre detekciu push-to-talk a 81.99% pre detekciu hlasovej aktivity.

Kedže proces trénovania neurónovej siete je veľmi časovo a výpočetne náročná udalosť, predtrénovaná sieť GPVAD bude adaptovaná, aby dokázala popri detekcií rečových segmentov detekovať aj push-to-talk udalosti. GPVAD trénovanie na push-to-talk udalosť bolo vykonané na pol hodine nahrávok a výsledkom je 82% úspešnosť. Dolaďovanie neurónovej siete skončilo s 66% úspešnosťou.

Tieto siete budú do istej miery porovnané a vyhodnotené. Kedže GPVAD neurónová sieť, bola natrénovaná na tisíckach hodín audia, nie je možné ich priamo porovnať s výkonom vlastne navrhnutých architektúr. GPVAD sa trénoval na odlišnom datasete za pomoci predtrénovaných váh na detekciu push-to-talk a nakoniec bola pridaná nová vrstva, aby súčasne model predpovedal rečové segmenty a push-to-talk udalosti.

# Automatic Speech Detection for VHF Channel

## Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Igor Szőke Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Mária Nováková
May 10, 2023


## Acknowledgements

First, I would like to express my sincerest thanks to my supervisor, Ing. Igor Szőke Ph.D., for his guidance, advice, and valuable remarks throughout my bachelor's thesis. His feedback has been essential in shaping my research and helping me navigate the challenges of this work. I am grateful for his encouragement to try working with neural networks. Thanks to this experience, I have developed a keen interest in machine learning.

Next, I would like to thank my whole family for their understanding and loving support throughout my academic path. I would also like to thank all my friends and school colleagues, especially Diana Maxima Držíková and Michal Vaňo, who have been my biggest mental support and have made my three years at FIT truly memorable.

# Contents

# Chapter 1

# Introduction

Air traffic control (ATC) is an important service provided by control towers which direct aircraft traffic. The proper directions are needed in order to evade collisions, organize the traffic flow and provide useful information for pilots. To ensure that voice communication is clear and quick, a reliable method needs to be applied. This process can be quite challenging due to an inevitable noisy environment on the pilot side.

Some of the most used systems in ATC are very high-frequency radio systems. These systems allow pilots and control towers to communicate in a certain frequency range. When a participant in the communication wants to communicate, a simple press of a push-to-talk button sets the radio system to transmission mode. By default, the radio system is in receiving mode. Various reliable methods are used for voice activity detection, and many other new approaches can outshine them.

The main motivation for this thesis sources from the ATCO2 project, which aims at developing a platform where air-traffic control data are collected, organized and pre-processed. The main goal of this thesis is to improve the automatic detection of push-to-talk in order to enhance the process of generating segments from captured communication. This could provide an advantage for machines to be able more quickly and accurately process the data.

This work will implement automatic detection of voice activity and push-to-talk with the usage of machine learning. In the process of creating a tool which could correctly detect the speech segments and push-to-talk events, two architectures are proposed and experimented with. The first proposed architecture is a simple feed-forward neural network whose main purpose was to get acquainted with machine learning. The second proposed architecture is a two-dimensional convolutional neural network, which provides an upgrade from the first architecture by using convolutional filters which are more suitable for voice activity detection and feature detection overall.

These models will be trained to detect speech segments and push-to-talk events on the audio recordings provided by the ATCO2 project. The dataset contains files with labels for speech and non-speech segments, which are used to provide supervision during the training of neural networks.

The labels for push-to-talk were not provided in the dataset and there are no public implementations for push-to-talk detection, so an algorithm was proposed. This algorithm called *Push-to-talk Event Extraction and Keying* (PEEK) can correctly detect push-to-talk events and has very high accuracy.

After training and evaluation of the proposed neural networks, the GPVAD implementation was adapted to be fine-tuned on the push-to-talk signals dataset. The GPVAD neural network can with high accuracy detect speech and non-speech segments, and knowledge of

this model will be utilised. The GPVAD architecture is trained to detect the push-to-talk by training the whole model and provides very accurate results.

The proposed models perform well on the ATCO2 dataset, as they are capable of detecting speech segments and push-to-talk events with high accuracy.

In the first chapter, it will be explained what neural networks are and how are they trained. The various architectures are described and discussed. The second chapter provides an overview of the main terms used in this thesis, such as voice activity detection and push-to-talk. Later, the data augmentation techniques and creation of the dataset will be discussed. Finally, the two remaining chapters will explain the process of training the neural networks. The models will be evaluated and compared.

# Chapter 2

# Neural Networks

Just as the human brain processes information and learns from experience, *Artificial Neural Networks* (ANNs) can be thought of as a digital brain that emulates this process to recognize patterns and make predictions, making them a powerful tool for solving complex problems. In this chapter, ANNs and their role in machine learning will be examined. The process of designing and training ANNs will be discussed, as well as the various types and architectures explored. The focus of this chapter will be on the practical applications of ANNs, with a particular emphasis on their use in feed-forward neural networks, convolutional neural networks, and convolutional recurrent neural networks, as they are the core network types used in this work.

## 2.1 Feed-Forward Neural Networks

*Feed-forward neural networks* (FNNs) [14, 3, 16], also *called multi-layer perceptrons* (MLPs), are a type of artificial neural network in which nodes do not form loops. The nodes receive data through input nodes, which flow through *hidden layers* and exit through output nodes. The information in the network is only passed forward, and no links exist that allow information to flow back from the output node.

The FNN approximates functions by calculating classifiers using the formula $y = f^*(x)$, where the input $x$ is assigned to its corresponding output $y$. For instance, in a classification problem, $f$ could be defined as assigning a category label to the input being classified. The network establishes a mapping function:

$$y = f(x; \theta) \tag{2.1}$$

- $\mathbf{x}$ is the input feature vector,
- $\mathbf{y}$ is the network's output vector,
- $\theta$ represents the network's learned parameters.

The network's goal is to learn the optimal $\theta$ values that result in $f$ approximating the target function $f^*$ as accurately as possible.

In a simplified form, the feed-forward neural network can be represented as a *single-layer perceptron* 2.1. The single-layer perceptron multiplies input with weights as they enter the layer, and the computed values are summed together. If the sum is above a certain threshold, typically above zero, the output value is usually converging to one. If the sum falls below the threshold, the output is usually converging to minus one. This behaviour can be found in *Sigmoid* function. This feed-forward architecture is often used for classification, and machine learning algorithms can be integrated into it.

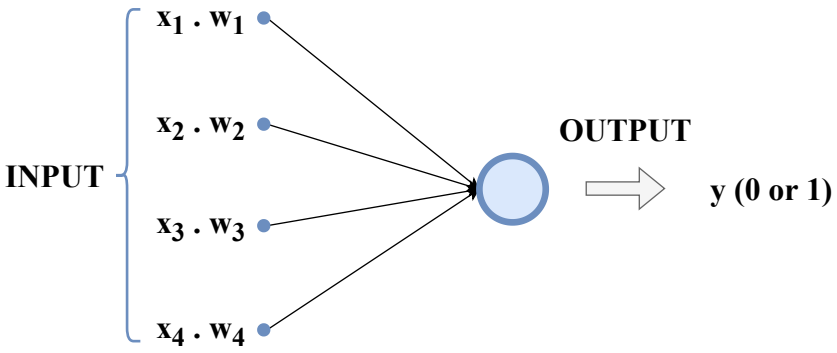


**Figure 2.1: Single-layer perceptron:** computational model that can learn to make decisions based on input data by adjusting weights associated with each input and applying an activation function to produce a single output.

### 2.1.1 Layers of FNNs

Feed-forward neural networks consist of an input layer, one or more hidden layers, and an output layer as shown in figure 2.2. The neurons in the input layer receive input and pass it on to the other layers of the network. The number of neurons in the input layer should match the number of features or attributes in the dataset.

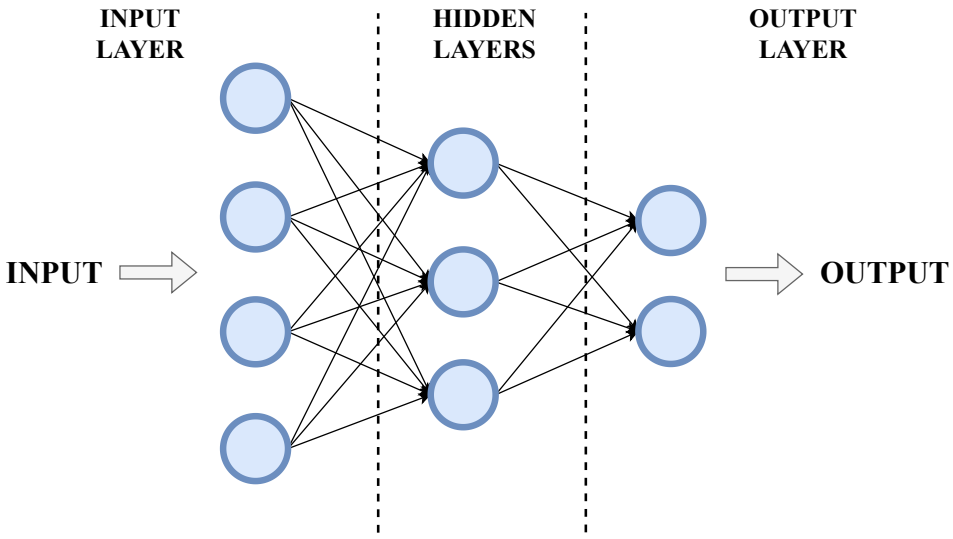


**Figure 2.2:** Basic FNN structure.

*The hidden layers* are placed between the input and output layers and contain several neurons that transform the input before passing it to the next layer. Each neuron in a layer is connected to the neurons in the previous layer via a weight, which measures their strength or magnitude. The neuron transforms its input, $x$:

$$f(x; w, b) = w^T \cdot x + b \tag{2.2}$$

- $\mathbf{w}$ is a vector of input weights,
- $\mathbf{b}$ is the bias.

The weights of the neurons are constantly updated during the learning phase to improve the accuracy of predictions.

*The output layer* represents the forecasted feature, and each neuron in the output layer is connected to the neurons in the last hidden layer. Artificial neurons in feed-forward networks are adapted from biological neurons and function in two ways: creating weighted input sums and activating the sums to make them "normal"[1].

Each *hidden layer* applies an *affine transformation* to its input:

$$a^n = W^n \cdot z^{n-1} + b^n \tag{2.3}$$

- $\mathbf{z}^{n-1}$ is the output vector of the previous hidden layer serving as the input of the current $n$-th layer,
- $\mathbf{W}^n$ is a matrix of weights assigned to the inputs,
- $\mathbf{b}^n$ is a vector of biases assigned to the hidden units,
- $\mathbf{a}^n$ is a vector of output activations.

The output of each hidden layer is transformed using a nonlinear, differentiable *activation function* $h$:

$$z^n = h\left(a^n\right) \tag{2.4}$$

- $\mathbf{z}^n$ is the final output of the layer.

Neurons in a neural network make decisions based on the *activation function*, which determines whether a *linear* or *nonlinear* decision should be made. The activation function also helps to prevent the cascading effect from increasing neuron outputs. This work will specifically use three activation functions (see figure 2.3): Sigmoid, ReLU, and LeakyReLU. These activation functions are used for regression problems.

[1] After creating the weighted input sums, the neuron applies an activation function to the sum in order to map the output to a desired range or state. This allows for nonlinear transformations of the input data and enables the network to learn complex patterns and relationships between inputs and outputs.

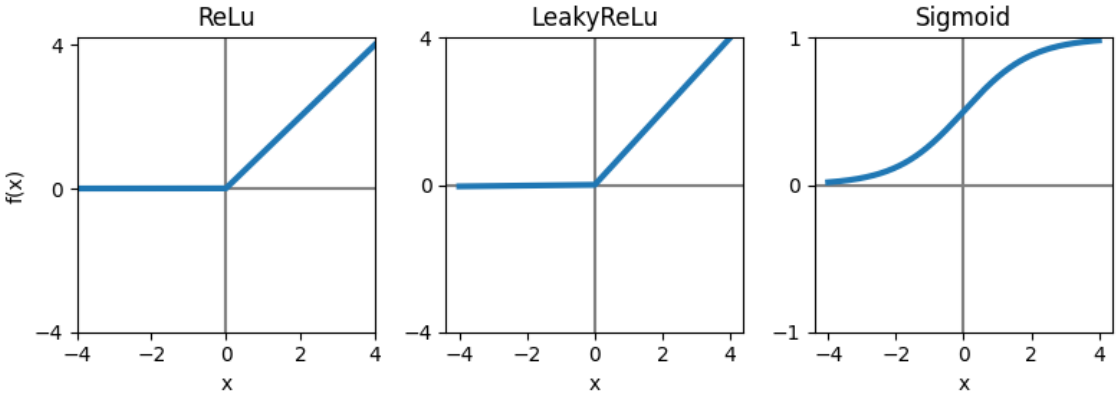


**Figure 2.3:** Graphic representation of activation functions: Rectified Linear Unit (ReLU), LeakyReLU and Logistic Sigmoid (Sigmoid) activation functions.

In FNN, the choice of activation function is important and depends on various factors. The activation function used in the output layer (such as logistic *Sigmoid*) is often different from the one used in the hidden layers. The *rectified linear unit* (ReLU) has become a popular choice for hidden layers, due to its simplicity and effectiveness in preventing the *vanishing gradient problem*[2]. *ReLU* only allows positive values to flow through, while negative values are mapped to 0. *LeakyReLU* is a variant of ReLU that addresses the "dying ReLU" problem[3] by allowing a small gradient to flow when the input is negative. *Sigmoid* is another popular activation function that squashes the input into a range between 0 and 1, making it useful for *binary classification* problems.

Lastly, in the thesis, another activation function is used for the classification problem. *Softmax* activation function is commonly used on output layers to produce a probability distribution over K classes. This probability distribution is useful in *multi-class classification* problems.

It is crucial for the activation function to be nonlinear, allowing the network to perform complex transformations beyond affine transformations. This enables the network to bend and fold the input feature space to model class boundaries in complex classification problems.

[2]When the gradients in the early layers of a deep neural network become very small during backpropagation, leading to poor convergence or no convergence at all. This issue can be mitigated by using activation functions like ReLU or other optimization techniques.

[3]It happens when the gradient of the function becomes zero or very close to zero for certain inputs, causing the neuron to "die" and no longer contribute to the network's output.

### 2.1.2 The Process of Training the FNN

The training process of the neural network starts with proper data preparation, which will be discussed in chapter 4. After preparing data, the training can start [26].

For simplicity, training methods can be divided into three types [15]:

- **Supervised learning** is when the neural network is provided true values which are excepted from the trained model.
- **Semi-Supervised learning** uses labelled as well as unlabelled data to train the neural network.
- **Unsupervised learning** does not require labels and therefore is not provided with any feedback during training. For unsupervised learning, a method such as clustering is used.

In this work, only supervised learning is implemented. Both push-to-talk and voice activity detection is trained with labels.

Supervised learning can be further divided into two types:

- **Classification** is a process when the neural network provides a prediction of the class to which the input data should belong.
- **Regression** models predict the specific value as an output.

In this thesis, the classification method will be used.

As it was mentioned earlier, the neural network consists of neurons. When training from scratch, the weights and biases are initialised randomly. During the training process, these values change based on provided parameters. These parameters are called *hyperparameters*. They are the most essential in the training process. The most crucial parameters are *learning rate, loss function* and *optimizer*.

The difference between the predicted output and true value is computed by the chosen loss function. There are various loss functions used for classification problems, but one of the most common is *CrossEntropy* 2.5. Every training process explained in chapter 6 uses this loss function.

$$H_p(q) = -\frac{1}{N} \cdot \sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right) \tag{2.5}$$

After computing the difference between ground truth and predicted value, it is necessary to optimize weights to assure that the next prediction is closer to ground truth. The predicted output is computed in the *forward pass* by applying weights and activation functions to the input.

The modification of the weights and biases is done by the optimizer call. The optimizer is initialised with a *learning rate*. The learning rate represents the step size at each iteration. It scales down the computed gradients computed during the *backward pass* call. The backward pass is done by *backpropagation* algorithm.

Some of the most common optimizers are *Stochastic Gradient Descent* (SGD)[4] or *Adaptive Moment Estimation* (Adam)[5], *Adaptive Moment Estimation with weight decay* (AdamW)[6].

SGD is a simple optimization algorithm for training neural networks that update weights based on the gradients. Adam and AdamW are extensions of SGD that can adaptively adjust learning rates and introduce weight decay to improve training performance.

The outcome of the training process depends on other important parameters [29] such as *Batch size*, and number of *Epochs*.

*Batch size*, encapsulates input to a vector of batch size. These chunks can make the model more generalised.

*Epoch* represents a single training iteration of all batches in both forward and backward passes. Setting the number of epochs too high could lead to *overfitting*. This would restrain the model from learning how to generalise the data, but rather perfectly fit the training data.

To prevent overfitting of the model, *Regularisation techniques* are applied. Regularisation techniques such as *dropout* and *batch normalisation* do not only prevent overfitting, but can also improve the accuracy of the model. The dropout method drops a number of neurons in the hidden layers. Randomly setting to zero some of the neurons forces other neurons to learn to recognise features. The batch normalisation method normalises the input layer to range from zero to one to speed up training.

## 2.2 Convolutional Neural Networks

*Convolutional Neural Networks* (CNNs) [14, 3] are a type of neural network that is well-suited for processing and analysing data that has a grid-like structure, such as images, audio, and time-series data. Unlike FNNs, CNNs have connections that are sparse and shared. These connections reduce the number of parameters in the network and make it possible to process large amounts of data more efficiently.

The basic idea behind CNNs is to use a set of learnable filters, also called kernels or weights, that slide over the input and compute *dot products*[7] between the weights and the local regions of the input. This operation is known as *convolution* and produces a *feature map* that highlights certain features or patterns in the input.

[4] see `pytorch.org/docs/stable/generated/torch.optim.SGD.html`

[5] see `pytorch.org/docs/stable/generated/torch.optim.Adam.html`

[6] see `pytorch.org/docs/stable/generated/torch.optim.AdamW.html`

[7] The mathematical operation that takes two vectors of equal dimensionality and returns a single scalar value.

The convolution operation can be represented as follows:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n) \tag{2.6}$$

- **S** is the output feature map,
- **I** is the input,
- **K** is the filter or kernel.

Each neuron in the feature map is connected to a small patch of the input and shares the same weights, which are learned during training. This weight sharing reduces the number of parameters in the network and allows CNNs to learn translation-invariant features.

### 2.2.1 Layers of CNNs

CNNs are composed of multiple layers that are arranged in a specific order, for example, shown in figure 2.4.

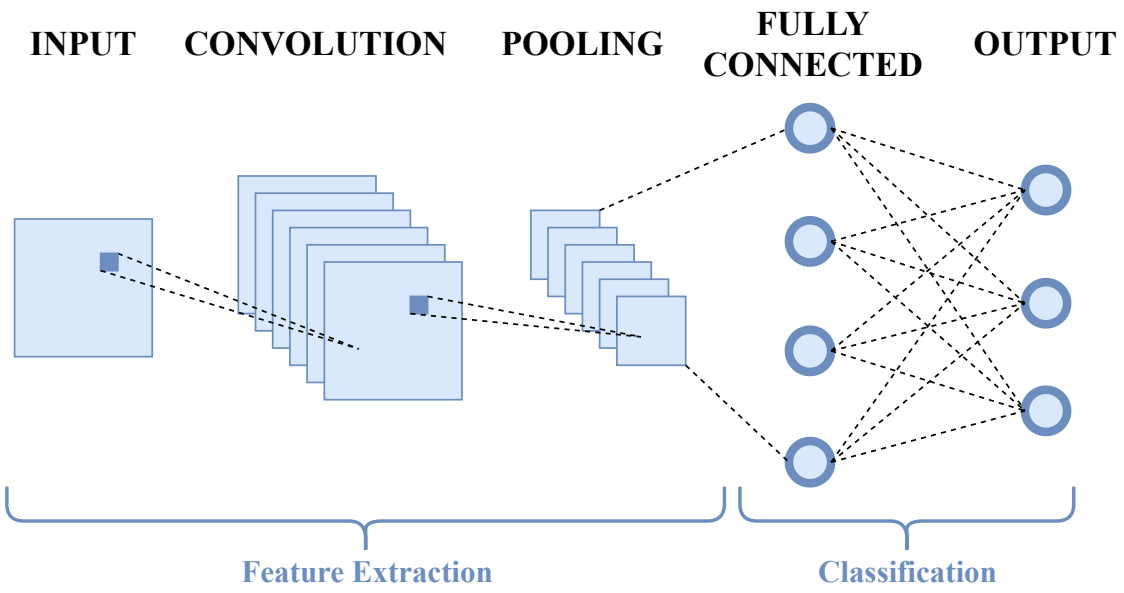


**Figure 2.4:** An illustration of the regular CNN for signal processing.

*Convolutional layers* are the core building blocks of CNNs. They apply several *filters* or *kernels* that slide over the input and perform a convolution operation. Each filter produces a *feature map* or activation map, which represents the presence or absence of specific patterns or features in the input. The output of a convolutional layer is fed to a non-linear activation function (such as ReLU), which introduces non-linearity into the network and enables it to learn more complex patterns. The output is then passed to the next layer, which could be another convolutional layer or a pooling layer.

*Pooling layers* are used to reduce the spatial dimension of the *feature maps* produced by the convolutional layers. They do this by *downsampling* the input in a specific window and returning a single output value. The most common type of pooling is *max pooling*,

where the maximum value in the pooling window is selected as the output (see figure 2.5). Max pooling helps to reduce the number of parameters in the network and also makes the network more robust to small translations in the input.

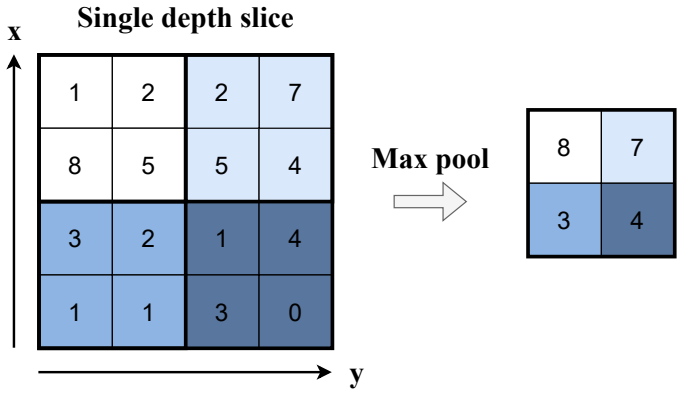


**Figure 2.5:** The process of downsampling an input, using 2×2 filters and moving the filter by a distance of 2 pixels each time.

*Fully connected layers* are used to connect all the neurons in one layer to all the neurons in the next layer. They are commonly used as the last layers in CNNs to classify the input into different classes.

## 2.3 Convolutional Recurrent Neural Networks

*Convolutional Recurrent Neural Networks* (CRNNs) [24] combines the power of both convolutional and recurrent layers (see figure 2.6). The input is filtered by the convolutional layers, which then create feature maps which bring attention to particular patterns and features. These feature maps are then given to the recurrent layers, which keep a hidden state that is updated at each time step and so captures the temporal dependencies.

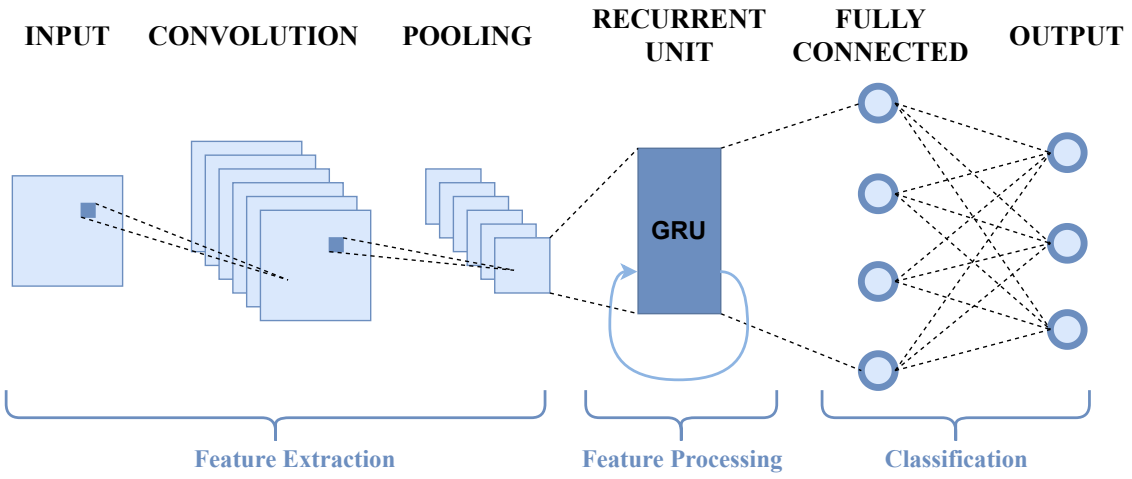


**Figure 2.6:** The architecture of simple CRNN.

### 2.3.1 Layers of CRNNs

*Convolutional layers* in CRNNs are similar to those in CNNs. They apply filters that slide over the input and produce feature maps. The output of the convolutional layers is fed to the recurrent layers.

*Recurrent layers* model sequential or time-series data, with an emphasis on data order. In contrast to feedforward and convolutional layers, which process each input independently and do not have any memory of previous inputs, recurrent layers maintain a hidden state that summarizes the previous inputs and influences the processing of the current input.

Utilising a feedback loop to link the layer's output to its input is the basic concept behind recurrent layers. This feedback loop allows the layer to pass information from one time step to the next and maintain a memory of the past inputs. In combination with convolutional layers, it allows CRNNs to handle both spatial and temporal information in data such as video or audio.

There are several variants of recurrent layers, including *Long Short-Term Memory* (LSTM) and *Gated Recurrent Unit* (GRU) layers, which use more complex architectures to better handle the flow of information through the network and mitigate the vanishing gradient problem. GRU is part of a CRNN architecture, that will be discussed in chapter 6.

A GRU [6, 19] unit typically consists of two gates: a reset gate and an update gate. The reset gate determines how much of the previous hidden state to forget, while the update gate determines how much of the new information to keep. The output of the GRU unit is a combination of the previous hidden state and the new candidate state that is computed based on the input and the reset gate.

The mathematical formulation of a GRU unit can be expressed as follows (see figure 2.7):

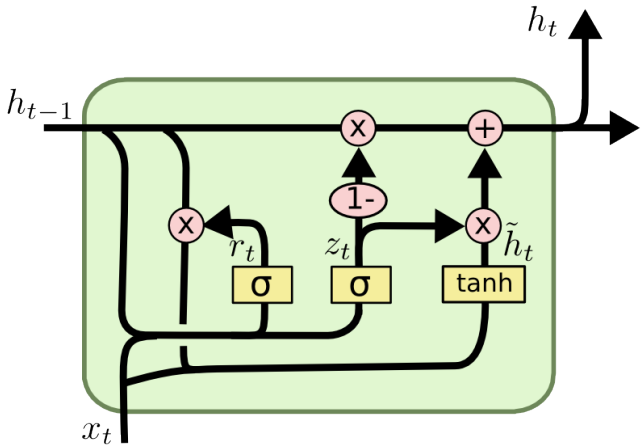


**Figure 2.7:** Graphic representation of a GRU cell. The image was obtained from [19].

1. First, we compute the *update gate* $z_t$ and the *reset gate* $r_t$ using the input $x_t$ and the previous hidden state $h_{t-1}$,

$$\begin{aligned} z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \\ r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \end{aligned} \tag{2.7}$$

   where $\sigma$ is the *sigmoid activation function*, $W_z$ and $W_r$ are weight matrices.

2. Next, we compute the *candidate state* $\tilde{h_t}$ using the input $x_t$ and the *reset gate* $r_t$,

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \tag{2.8}$$

   where $*$ represents the element-wise multiplication.

3. Finally, we compute the *new hidden state* $h_t$ as a *linear interpolation* between the *previous hidden state* $h_{t-1}$ and the *candidate state* $\tilde{h_t}$, weighted by the update gate $z_t$,

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{2.9}$$

   where $*$ again denotes element-wise multiplication.

With state-of-the-art performance for many sequential modelling tasks, the GRU is currently one of the most popular and widely used CRNN variations.

*Fully connected layers* in CRNNs are used to connect all the neurons in one layer to all the neurons in the next layer. They are commonly used as the last layers in CRNNs to classify the input into different classes.

# Chapter 3

# Voice Activity Detection and Push-to-talk in Air Traffic Communication

Effective communication is crucial for Air Traffic Control operations. To ensure quality and flight safety, advanced signal processing techniques are necessary. This chapter focuses on two techniques: Voice Activity Detection and Push-to-talk in Air Traffic Communication. Section 3.1 provides an overview of Air Traffic Control operations and how to execute them safely and timely. Section 3.2 discusses the concept of Voice Activity Detection. It explains why this process is necessary and how speech segments are separated from non-speech segments in a communication channel. Finally, section 3.3, introduces the concept of Push-to-talk, which is used to control speech transmission between two endpoints.

## 3.1 Air Traffic Control

Air traffic control (ATC) is a system of technologies and procedures that guide the movement of aircraft in the air and on the ground. ATC is in charge of maintaining aircraft separation, delivering clearances and instructions to pilots, as well as monitoring and directing traffic flow to prevent collisions [7].

The major goal of ATC is to ensure the security of aircraft, passengers, and crew, as well as to avoid collisions with other aircraft and obstructions. ATC controllers work in control towers or in en-route centres to monitor the movement of aircraft and to communicate with pilots using radar, radio, and computer systems.

### 3.1.1 Air Traffic Communication

Communication between pilots and controllers during different phases of flight is an essential element of air traffic control. This communication follows established protocols and phraseology, including standardized language developed by the International Civil Aviation Organization (ICAO)[1]. To guarantee clear communication and reduce misunderstandings and mistakes, pilots and controllers get trained to use standardised phraseology and adhere to established procedures [11, ch. 1, pp. 18-19].

[1] see https://www.icao.int/Pages/default.aspx

Standard phraseology is used in different types of communication, including radio communication, which is the primary method of communication between pilots and controllers [11, ch. 2, pp. 30-31]. Radio communication requires the use of specific tools and techniques, such as selecting the right frequency to connect with air traffic control and delivering necessary information in a simple and straightforward way.

### 3.1.2 Very High-Frequency Channel

According to the FAA's "Air Traffic Control" guide [13, p. 573], VHF (Very High Frequency) is a band of radio frequencies between 118.000 and 136.975 MHz that is used for short-range communication in air traffic control. VHF is the primary communication method used in ATC and is used for communication between pilots and air traffic controllers on the ground and in the air.

VHF communication is accomplished using a network of ground-based radio transmitters and receivers that are located at airports and other ATC facilities. Each facility has one or more VHF channels assigned to it for communication with pilots. Pilots tune their VHF radios to the appropriate channel for the facility they are communicating with.

When communicating on a VHF channel [13, pp. 53-54], pilots and air traffic controllers use standard phraseology to exchange information about the aircraft's position, altitude, heading, and intentions. This information is used by air traffic controllers to safely and efficiently manage air traffic in the vicinity of the airport or in the airspace sector they are responsible for.

VHF is an essential component of air traffic control, providing reliable and efficient communication between pilots and controllers and helping to ensure the safety of the flying public.

## 3.2 Voice Activity Detection

Voice Activity Detection (VAD) is a process that's main goal involves identifying speech regions in a noisy audio signal using a feature vector $x$ and deciding between two hypotheses 3.1 of speech and noise [21]. As a result, each audio frame from the source will be evaluated against:

$$\begin{aligned} H_0 &: \quad x = n, \\ H_1 &: \quad x = n + s \end{aligned} \tag{3.1}$$

- $H_0$ hypothesis suggests that the current frame solely contains noise or/and other non-speech signals $n$,
- $H_1$ hypothesis indicates that the current frame consists of a speech signal $s$ and potential background noise signals.

Whether a frame contains speech or noise can be determined by choosing the hypothesis with the highest probability given the current frame's features $x$, using a method called maximum a posteriori classification, expressed using Bayes' theorem:

$$\text{VAD}\,(x) = \begin{cases} \text{non-speech} & \frac{P(x|H_0)\cdot P(H_0)}{P(x)} > \frac{P(x|H_1)\cdot P(H_1)}{P(x)}, \\ \text{speech} & \text{else.} \end{cases}$$

When performing voice activity detection, it is common to break down the process into three distinct stages, with each stage playing a crucial role in determining whether speech is present in an audio signal. These stages typically include:

1. **Feature extraction**, where relevant features are extracted from the audio signal. It is important to take into consideration the approach used for modelling voice activity detection because various methods may require different features;

2. **Decision module** where a decision is made about whether the audio signal contains speech or noise based on the extracted features;

3. **Decision smoothing**, to deal with VAD decision errors that may occur due to high levels of background noise. It can help reduce the number of false positive or negative decisions by making the VAD decision less erratic and more dependable.

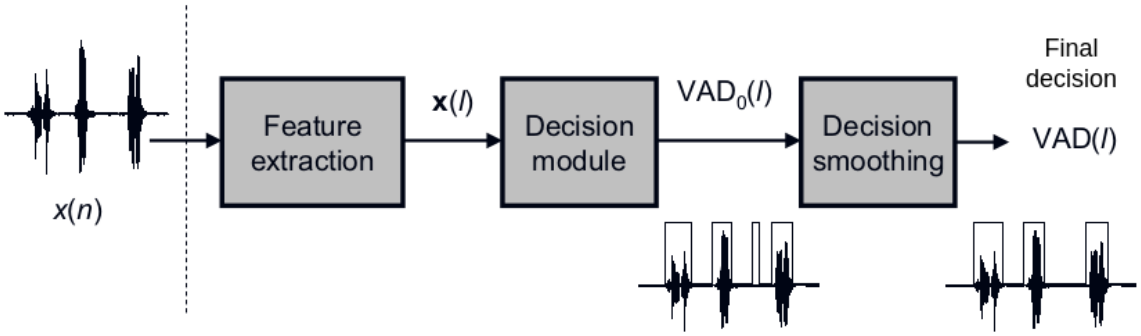


**Figure 3.1:** A graphical representation showing the components and their interconnections of a VAD system. The figure was adapted from [21].

VAD is an important preprocessing step in larger speech-processing systems, such as speech recognition, speaker recognition, and speech enhancement. One of the advantages of VAD is that it can reduce the computational cost of speech processing algorithms by removing non-speech segments from the audio signal.

However, VAD can be difficult to implement, especially in loud or acoustically challenging environments in general. Speech and other sounds, such as background noise or music, may be difficult to distinguish, especially when using traditional VAD methods. Machine learning-based approaches may

### 3.2.1 Traditional Voice Activity Detection Approaches

Traditional VAD Approaches date back to the 1970s, and the field has a long history [31]. In general, traditional VAD methods involve analysing various characteristics of the input signal and using a heuristic or specific algorithm to make an accurate decision. These characteristics typically include energy levels, spectral information, zero-crossing rate, and long-term spectral divergence [28, 30]. The most essential traditional VAD methods include:

- **Energy-Based Method**, which uses the energy level of the speech signal as the primary feature to detect speech activity. The basic idea is that speech signals have a higher energy level than non-speech signals such as background noise or silence.

- **Zero-Crossing Rate Method's** evaluation is based on the number of times the audio signal crosses the zero-axis in a given time frame. The concept is that speech signals have a higher ZCR than non-speech signals, because speech signals contain more rapid changes in amplitude due to the vocal cords vibrating, while non-speech signals, such as background noise or silence, have a more consistent amplitude.
- **Spectral-Based Method** uses spectral characteristics of speech and non-speech signals to distinguish between them. This method analyses the power spectral density (PSD) of a signal to extract features that can be used to classify the signal as speech or non-speech.

The choice of which VAD method to use may vary depending on the specific context, such as available resources, desired level of accuracy, or system performance requirements.

### 3.2.2 Machine Learning Voice Activity Detection Approaches

Modern approaches to VAD incorporate machine learning techniques [1, 29], which have become increasingly popular and have begun to replace traditional VAD methods.

Machine learning-based VAD methods involve using statistical and machine learning techniques to learn patterns in audio data that are indicative of speech or non-speech, and then using these patterns to classify new audio samples.

Some common machine learning-based approaches for VAD include:

- **Supervised Learning Methods** for VAD represent training a machine learning model on labelled data, where the labels denote whether a given audio segment contains speech or not. The model learns to recognize patterns in the audio that are associated with speech, such as the presence of voiced sounds, energy in a particular frequency range, or changes in spectral content over time. The model can then be used to classify new, unlabelled audio segments as speech or non-speech.
- **Unsupervised Learning Methods** for VAD involve using clustering techniques to segment the audio into speech and non-speech clusters without relying on labelled data. For example, k-means clustering can be used to group similar segments of audio together based on their spectral properties, or Gaussian mixture models can be used to fit a mixture of Gaussian distributions to the audio data, with each component corresponding to either speech or non-speech.

*Deep Learning Methods* for VAD involves training deep neural networks (DNNs) on raw audio signals or spectrogram features to perform VAD. Convolutional Neural Networks (CNNs) and Convolutional Recurrent Neural Networks (CRNNs) are commonly used for VAD using deep learning techniques. In a CNN, filters are applied to small segments of the audio to extract local features, which are then pooled and combined to form higher-level representations. In a CRNN, the network maintains a memory of previous time steps and uses this information to make predictions about the current segment of audio.

The exact approach used can depend on factors like the nature of the audio data, the required accuracy and speed of the VAD, and the resources available for model training and deployment.

### 3.2.3 Voice Activity Detection and Air Traffic Control

Air Traffic Control systems relies heavily on VAD, because it helps to filter out background noise and detect when pilots are speaking. There may be several pilots and air traffic controllers communicating on the same frequency in an ATC environment, which can cause interference and confusion. By identifying speech activity and muting the audio when there is silence, VAD can help reduce these problems.

In emergency situations, it is important for ATC to quickly identify and prioritize communication from pilots. VAD can help to automatically detect when a pilot is speaking and alert the air traffic controller to the urgency of the situation, enabling them to respond quickly and appropriately.

## 3.3 Push-To-Talk

*Two-way radio systems*[2] receive and transmit signals through very high-frequency waves [27]. To transmit audio, a push-to-talk switch needs to be pressed. By pressing and holding, the communicant can start to speak. Depending on the specific system, this event can produce a "click" sound. Without pressing the switch, the communication channel is in receiving mode.

The start of the push-to-talk event begins with pressing the button. This action sends an electrical signal to the device, which activates the microphone and broadcasts the audio as an analogue signal. The signal can be in the real-time heard by the listener on another headset. The push-to-talk event can be identified as a sudden change in voltage. Figure 3.2 shows audio with a noticeable push-to-talk event. The red square represents the duration of this event that was selected for this thesis. The duration of the PTT event is approximately 100 ms long.

There are little to no public implementations on how to detect push-to-talk events, so comparison with different approaches is not possible. The push-to-talk duration is at least $100ms$. The position of the peak is usually at the beginning of the frame sequence.

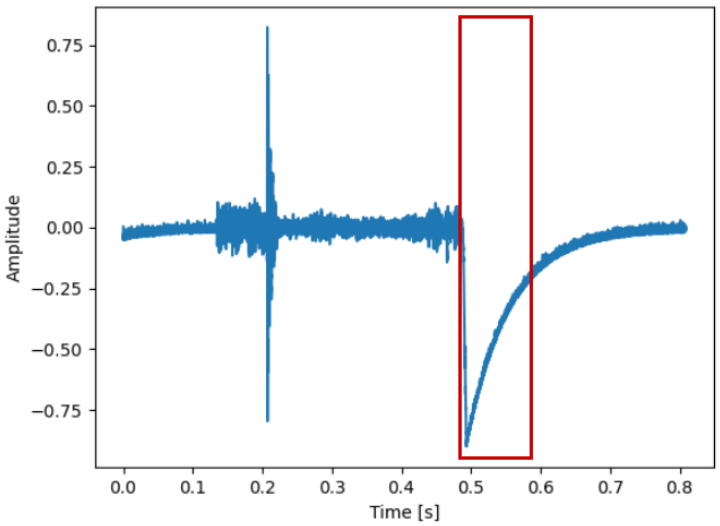


**Figure 3.2:** Push-to-talk signal in audio recording.

[2]Short-range communication systems that allow multiple parties to communicate using radio waves.

### 3.3.1 Push-To-Talk and Air Traffic Control

When a pilot activates their PTT switch to transmit a message, it can serve as an indication to the VAD system that a new speaker has started speaking. This information can then be used to identify different speakers and help ATC to reduce confusion and interference on the frequency. Push-to-talk switches can fail, which could disable transmitting messages.

The pressed PTT button or switch activates the aircraft's radio transmitter, which allows the pilot to transmit their message to the ATC tower or other aircraft. Simultaneously, the aircraft's receiver is muted, which prevents the incoming transmission from being heard by the pilot.

When the button is released, the transmitter is turned off, and the receiver is unmuted, which allows further communication.

In aviation communication systems, PTT is usually activated through a button or switch located on the control wheel or yoke of the aircraft. This allows the pilot to keep their hands on the controls while transmitting their message. Some aircraft may also have additional PTT buttons or switches located on the instrument panel or elsewhere in the cockpit for use by other crew members.

# Chapter 4

# Used Datasets and Data Augmentation

An overview of signal processing techniques and a description of the ATCO2[1] project can both be found in this chapter. It will be discussed in more detail later how the dataset for Push-To-Talk was assembled and labelled.

Data preparation for neural network training is a challenging process. An unbalanced dataset, inaccurate labels, or poor-quality data are a few of the factors that might make neural network training fail. Providing a precise dataset is needed to avoid overfitting or underfitting the neural network and to obtain desired results.

## 4.1 ATCO2 Dataset

*ATCO2* project[2] concentrates on developing a platform where voice communication can be collected, organized and pre-processed [33]. The data is collected through very high-frequency radio receivers that are owned by a community of volunteers.

The data processing pipeline consists of:

1. speech pre-processing tools such as segmentation or volume adjustment
2. diarization which is used to split audio by speakers
3. automatic speech recognition
4. English language detection
5. speaker role detection, which can be ATCO or pilot
6. labelling of call signs, commands and values

This pipeline was used to produce the dataset that was provided for this thesis. The provided dataset contains four hours of ATC speech with manual transcripts. The dataset was built for the development and evaluation of *Automatic Speech Recognition* and *Natural Language Processing* for the ATC domain. The audio files in this dataset are sampled at 16,000kHz.

[1] see https://www.atco2.org/
[2] see https://www.atco2.org/

The structure of the dataset is explained in the following figure:

```
ATCO_TEST_SET_4H
└── release
    ├── LISTS
    │   └── accepted.list
    ├── DATA
    │   ├── LSGS_SION_Ground_Control_121_7MHz_20210503_122624.wav
    │   ├── LSGS_SION_Ground_Control_121_7MHz_20210503_122624.segm
    │   ├── LSGS_SION_Ground_Control_121_7MHz_20210503_122624.cnet
    │   └── ...
    └── DATA_nonEN
        └── ...
```

**Figure 4.1:** Tree structure of provided data folder.

As it is shown in the example of files in the folder (figure 4.1) the most important files in this dataset are wav files.

Other important files are `CNET` files and `SEGM` files. `CNET` files contain annotation for every word that was detected in the audio using the *Kaldi confusion network* which is an automatic transcript, and `SEGM` files contain speech segments. Nearly every audio was annotated, and approximately 66% of audio was accepted as correctly annotated by the human re-checker.

## 4.2 Data Pre-Processing

Signals are converted into spectrograms and then fed into a neural network. Spectrograms are visual representations of signal strength, or "loudness" over a time period and frequency. High energy levels and low energy levels are easy to distinguish visually. To enhance spectrogram features, it is conventional to apply MelBank filters [18]. Mel spectrogram is created by converting the spectrum into Mel Scale.

- a spectrum is created by transforming the signal from time domain[3] to the frequency domain using the Fourier Transform mathematical formula [17]:

$$F(\omega) = \int_{-\infty}^{\infty} f(x) \cdot e^{-i\omega x} dx \tag{4.1}$$

- Mel Scale is a non-linear representation of human perception of sound. The frequencies are converted using a formula:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{4.2}$$

  where $m$ represents Mel and $f$ represents frequency [20].

[3] Continuous signal into discrete frequencies.

The neural network will get a glimpse of how humans perceive sound by transforming the original signal into a Mel Spectrogram. The process of generating a Mel Spectrogram is shown in figure 4.2.

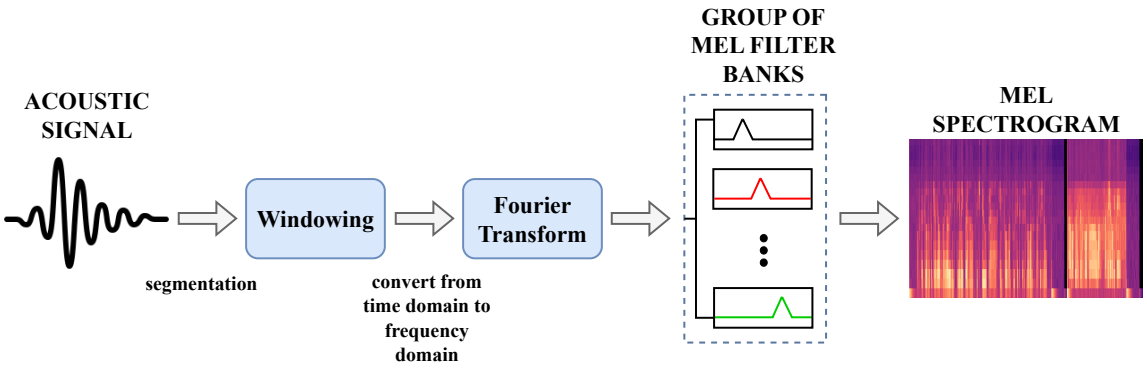


**Figure 4.2:** Pipeline to generate Mel spectrogram.

Mel Spectrograms were generated differently for each implementation (table 4.1).

| | Window (ms) | Overlap (ms) | Mels | Window Function | Post-Log |
|---|---|---|---|---|---|
| **ENN** | 25 | 10 | 20 | Hamming | no |
| **GPVAD** | 40 | 20 | 64 | Hamming | yes |

**Table 4.1:** Table shows key differences in pre-processing signal.

Experimental Neural Networks (ENN) window duration and overlap duration are shorter compared to GPVAD's [9, 10] parameters (table 4.2). GPVAD's log Mel Spectrogram can present the characteristics in the low- and high-frequency regions with greater detail than the non-logarithmized Mel Spectrogram [23].

| | Detecting Short-Lived Features | Better Frequency Resolution |
|---|---|---|
| **ENN** | yes | no |
| **GPVAD** | no | yes |

**Table 4.2:** Table shows key differences in pre-processing signal.

It is hard to determine which parameters are better due to opposition advantages and disadvantages [4]. Each of the pre-processed signals is shown in figures 4.4 and 4.3. Signal pre-processing is a crucial part of preparing the data to feed the neural network. Mel Spectrograms have been shown to be very beneficial.

Using the MFCC[4] as input is another strategy. MFCC is an extension of the Mel Spectrogram. The Logarithm of Mel spectrogram magnitudes undergoes the process of Discrete Cosine Transform and coefficients are created [2].

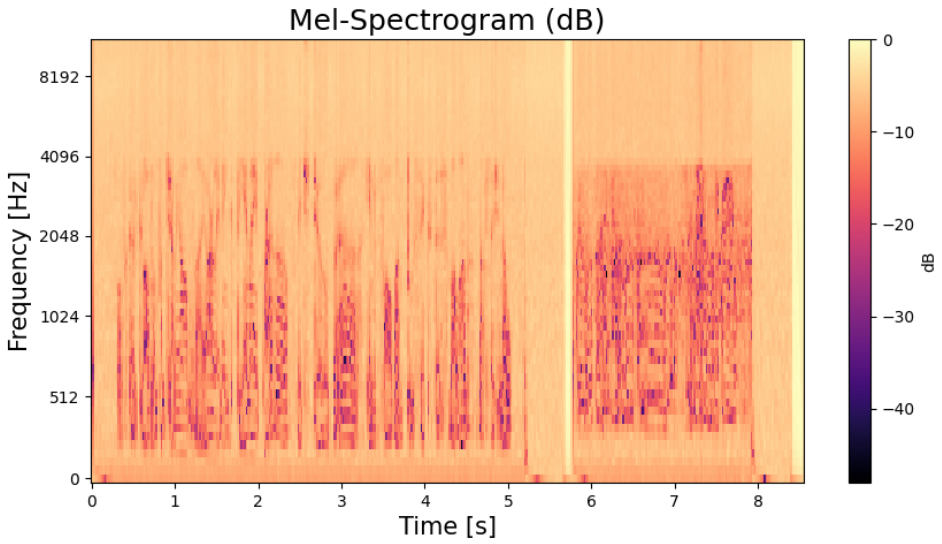


**Figure 4.3:** Mel spectrogram input for GPVAD implementation.

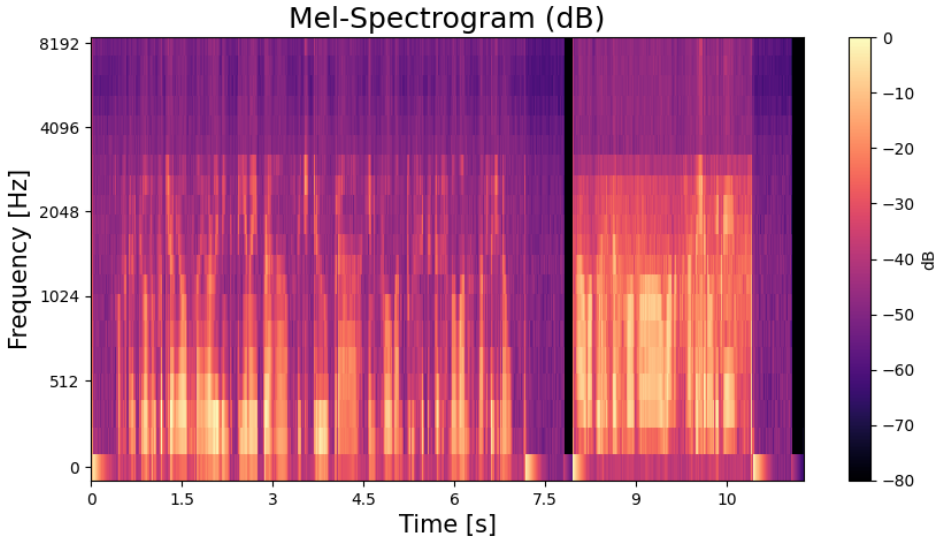


**Figure 4.4:** Mel spectrogram input for ENN implementation.

[4]Mel Frequency Cepstral Coefficients.

## 4.3 Dataset for Push-To-Talk

a balanced dataset is necessary for effective neural network training, as was described in the section 4.5. ATCO2 provides audio files which contain recordings that vary in duration. Almost every audio has a minimum of one push-to-talk (PTT) signal.

As discussed in section 3.3, the push-to-talk signal is represented by temporarily muting the transmitter. Push-to-talk is most of the time easily identified in the recordings (see figure 4.5).

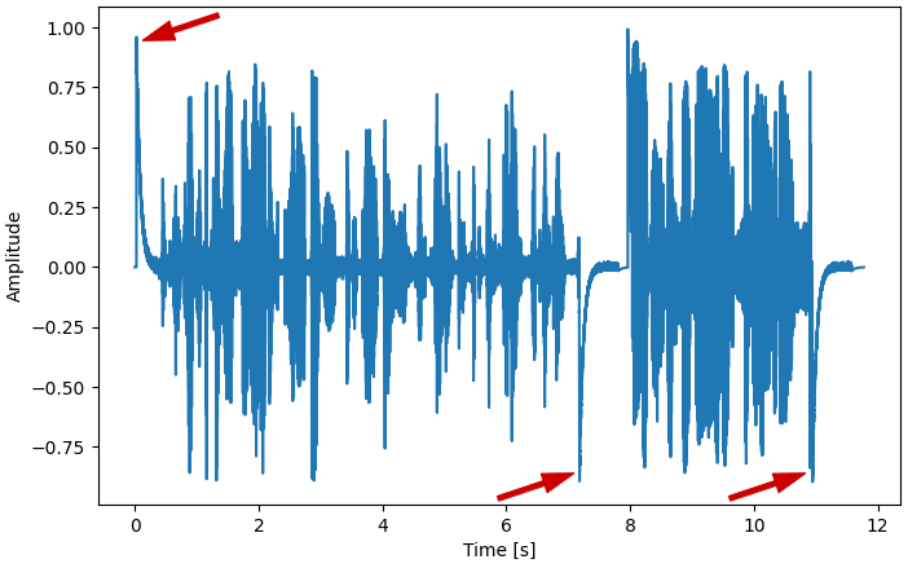


**Figure 4.5:** Image of audio with three push-to-talk events highlighted by the red arrows.

In this dataset, the PTT signal could be classified into three categories (figure 4.6):

- **A clear PTT signal** has the biggest representation in the dataset. PTT of this kind is easily recognized by algorithms or by humans.
- The second most represented category is **noisy PTT signals**. It makes sense that noise-free recordings are frequently challenging to get through air traffic control. Usually, there is excessive noise in the audio, especially on the pilot side. PTT events can be detected despite background noise by using filters that reduce noisy backgrounds.
- The last category represents **double-pressed or multiple-time pressed** PTT signals. This can occur by mistake or by certain configurations of the technology.

a logical distribution between non-PTT and PTT events is necessary when training a neural network to detect push-to-talk signals. The audio's speech and non-speaking segments can both be classified as non-PTT events. Using the PEEK algorithm (see 4.4) audio segments were extracted and one by one manually checked. These segments were used in combination with segments that did not contain PTT events. By randomly selecting from each group, a dataset was created.

In the process of creating the dataset, localised push-to-talk events were produced from each audio in the ATCO2 dataset by the PEEK algorithm. Each detected event was

manually checked and evaluated if it should be stored in the dataset. This way, over, 3500 push-to-talk events were checked and accepted by hand.

In summary, the process of creating a dataset that should be used in order to train a model to detect PTT events was quite challenging. In order to reduce False Alarms, it was important to include in the dataset recordings which did not contain any PTT signal.

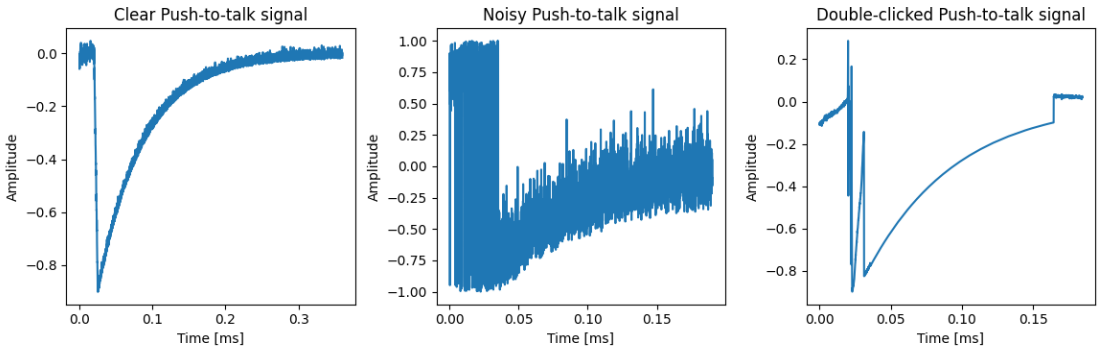


**Figure 4.6:** Types of PTT signal.

The dataset is containing two types of recordings. The first type represents audio files with no push-to-talk event. The second type contains recordings in which at least one push-to-talk event occurred. These recordings contain from one to five push-to-talk events.

## 4.4 Data Labelling

This section can be divided into two parts: labelling voice activity (see 4.4.1) and push-to-talk (see 4.4.2).

Labelling data is the most challenging part of creating a dataset. To supervise the training of the neural network, high-quality labels are necessary. The main goal is to create labels with a very low error rate. The process of creating labels for voice activity was easier in comparison with labels for push-to-talk.

### 4.4.1 Labelling Voice Activity

Annotations for voice activity detection were included in the ATCO2 dataset. As stated in section 4.1 `CNET` files contained transcriptions to text for each word in the communication. The transcriptions were generated by *Hybrid Automatic Speech Recognition*. Labels from these files were used while creating labels for the ENN neural network training, considering more precise speech segments. `SEGM` files contain segments of speech which are more suitable for final comparison between different approaches.

Labels were generated from processing `CNET` files, which contain timestamps for each word. Each frame in Mel Spectrum was a label assigned. Missing segments from files were substituted by placing non-speech labels in their place.

In figure 4.7 the difference between `CNET` files and `SEGM` files is shown.

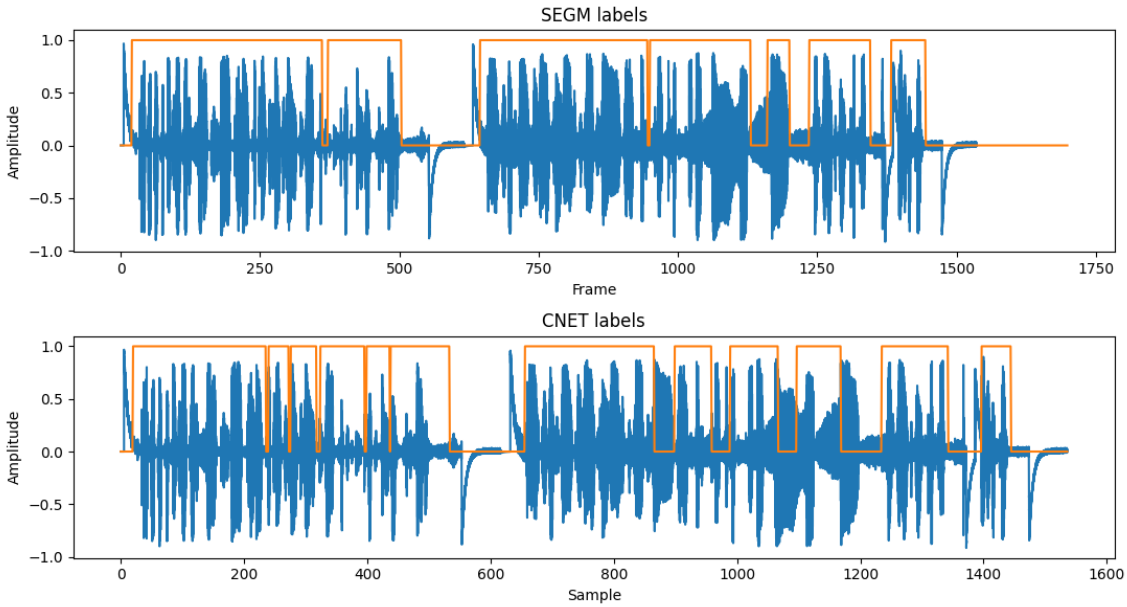


**Figure 4.7:** Comparison between two available labels from files. Label one represents speech and label zero represents non-speech.

Labels for GPVAD implementation were generated from the output of the pre-trained model. This will be further explained in chapter 5.

### 4.4.2 Labelling Push-To-Talk

The ATCO2 dataset does not contain PTT labels. It would be possible to go over each audio in the dataset and label it in some programs like Audacity, but this process would be very time-consuming. To automatize the process of labelling, various approaches were tried. Every one of them was concentrating on sudden changes in signal energy. It is important to mention that the duration of the PTT varies. To generalize the PTT event, a threshold of 100 ms was selected. This threshold represents a minimal duration of the event.

In the first attempt, the power of each frame was considered. By analysing the values of the spectrogram, the potential start of the peak was selected, and its neighbourhood was examined. The basic idea of this approach was to examine the sudden drop and smooth increase of signal in an adjacent area to a potential peak.

**Algorithm 1:** PTT detection algorithm based on analysing spectrogram.

**Input:** Signal
**Output:** Labels for PTT

```
examination = False;
final evaluation = False;
foreach frame in audio do
    if lower energy in the frame then
        potential frame selected;
        examination = True;
    end if
    if examination then
        examine the current frame for a smooth increase
        if examinating for x frames then
            examination = False;
            final evaluation = True;
        end if
    end if
    if final evaluation then
        check sudden drop;
        check smooth increase;
        check the power of the potential frame;
        if every check succesful then
            add a potential frame to the PTT array;
        end if
        final evaluation = False;
    end if
end foreach
foreach element in PTT array do
    previous two frames, the selected frame, and seven following frames are
     labelled as PTT event;
end foreach
```

This method seem promising at first because it could detect peaks with certain negative amplitude. Despite that, its overall performance had shown to be really poor. It was not generic enough to capture each PTT event and label it correctly because of the specific values. After examination, most of the PTT events were missed. On the other hand, the percentage of false alarms was close to zero, This algorithm was not sufficient to use as an automatic label method.

After bad results and a realisation that the push-to-talk event is not only the negative amplitude, the strategy changed. The second attempt was similar to the first attempt. It was still necessary to revolve around changes in amplitude. The signal is examined for both positive and negative peaks. Using Python library functions `find_peaks`[5] and `find_peaks_cwt`[6] to find peaks was unsuccessful. Peaks were not correctly recognised. The outcomes were not applicable to the final detection.

[5] see https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html
[6] see https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks_cwt.html

One appeared more promising than the others after multiple attempts of finding satisfactory peak detection algorithms. Local maxima might be easily found throughout the signal by using a peak detection method called *peakdet*[7] converted from Matlab to Python by Eli Billauer [5]. As the delta value 0.2 was chosen as it seemed the most optimal value.

The push-to-talk event starts with high negative or positive amplitude, after which the upcoming $100ms$ does not flip over zero. The position of the peak is usually at the beginning of the corresponding number of frames.

Conditions for elimination were used once the peaks had been retrieved. When the signal did not flip over zero for at least 100 milliseconds and the peak was a local extreme in the adjacent area, it can be classified as a PTT event. The additional search for local extreme in the neighbourhood[8] excludes peaks that were not excluded in the duration test. This test is especially useful for double-pressed PTT signals.

The PTT labelling was greatly automated thanks to this algorithm. Both ENN and GPVAD implementations rely on labels produced by this technique.

**Algorithm 2:** PTT detection algorithm (PEEK) based on peak detection in a signal.

**Input:** Signal
**Output:** Labels for PTT

```
positive peaks, negative peaks = peakdet (signal);
foreach peak in negative peaks do
    if peak is negative for at least 100 ms then
        keep;
    end if
    if peak is local extreme in its negative neighbourhood then
        keep;
    end if
end foreach
foreach peak in positive peaks do
    if peak is positive for at least 100 ms then
        keep;
    end if
    if peak is local extreme in its positive neighbourhood then
        keep;
    end if
end foreach
```

The final comparison between these two approaches was evaluated on 1000 recordings from the PTT dataset (see 4.3) which contained various numbers of PTT events.

[7] see http://billauer.co.il/blog/2009/01/peakdet-matlab-octave/

[8] The interval starts at the first flip over zero sample before the peak. It ends at the last sample after the peak before flipping over zero.

[8] http://billauer.co.il/blog/2009/01/peakdet-matlab-octave/

Table 4.3 shows the difference in causing false alarms and missed hits with each approach.

| | Missed Push-Po-Talk (%) | False Alarms | Accuracy (%) |
|---|---|---|---|
| **Algorithm 1** | 81.30 | 8 | 18.70 |
| **Algorithm 2** | 0.30 | 0 | 99.70 |

**Table 4.3:** In the dataset, there were 988 PTT events.

As is shown in table 4.3 the second algorithm is far better than the first one. It correctly identified most of the PTT events and didn't cause any false alarms. On the other hand, the first algorithm performed poorly in detecting PTT events and caused 8 false alarms. The second algorithm's performance is more than sufficient for generating labels automatically. The dataset consisted of manually checked push-to-talk events.

The length of the PTT event varies. This is because some of the frames that are considered to have a PTT peak in them could have this peak at the end of the frame, which could cause mislabelling. PTT event lasts for approximately 100ms. This is due to prevent another mislabelling of silence for the PTT event. This topic and experiments with labelling will be further explained in chapter 5.

PTT labels generated from the second algorithm are shown in figure 4.8.

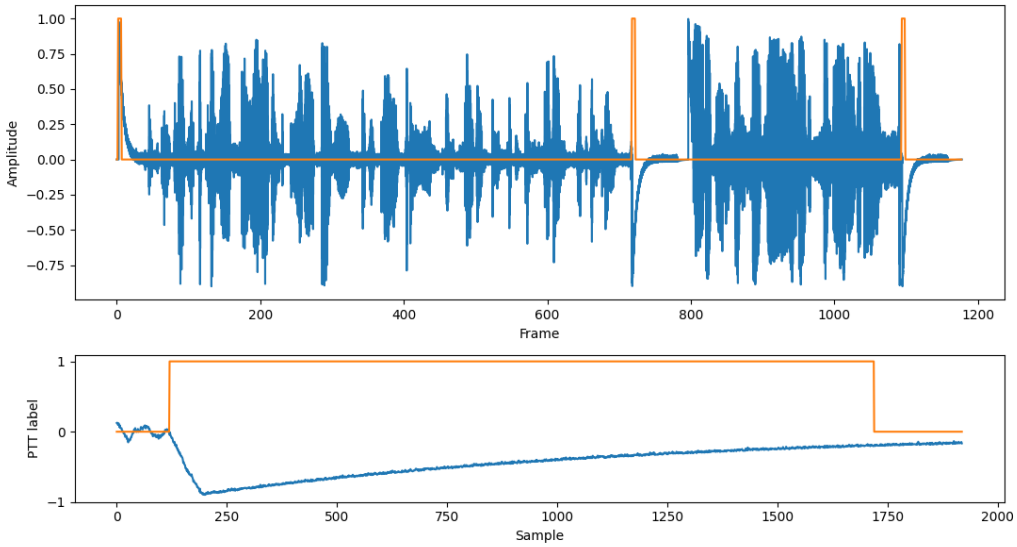


**Figure 4.8:** The upper graph shows a whole image with three PTT events. The bottom graph is zoomed to one of the labelled push-to-talk events.

During training the neural network on multitask detection, it is important that labels do not cause a conflict. The push-to-talk event can be only possible when the non-speech frame is present in the second class. Training on datasets with wrong labels would lead to a deterioration of accuracy.

## 4.5 Augmentation Techniques

Employing augmentation can boost neural networks' capacity for generalisation. Augmentation is a popular preprocessing technique used in the neural network training process. The dataset becomes more robust and varied, which can aid in the training process and improve the classification accuracy of new incoming data. The application of time-domain modification is the method used most frequently to enhance signal data [32]. Some of the most common techniques are [22]:

- **Time Stretching** is a method when the audio is slowed down or sped up while keeping the pitch unchanged.
- **Window Slicing** method refers to randomly selecting a time range which is downsampled or upsampled. Other time ranges remain untouched.
- **Magnitude Slicing** refers to a method during which the magnitude of the signal is modified. It is done by multiplying the original magnitude of the signal at a certain position with another selected magnitude.
- **Noise Injection** is another very commonly used method in signal augmentation. It is done by adding noise to certain signal parts. The labels for these parts remain unchanged.

Other techniques such as normalising the signal or noise filtering can improve the quality of the dataset. Figure 4.9 shows various signal changes after applying some augmentation techniques.

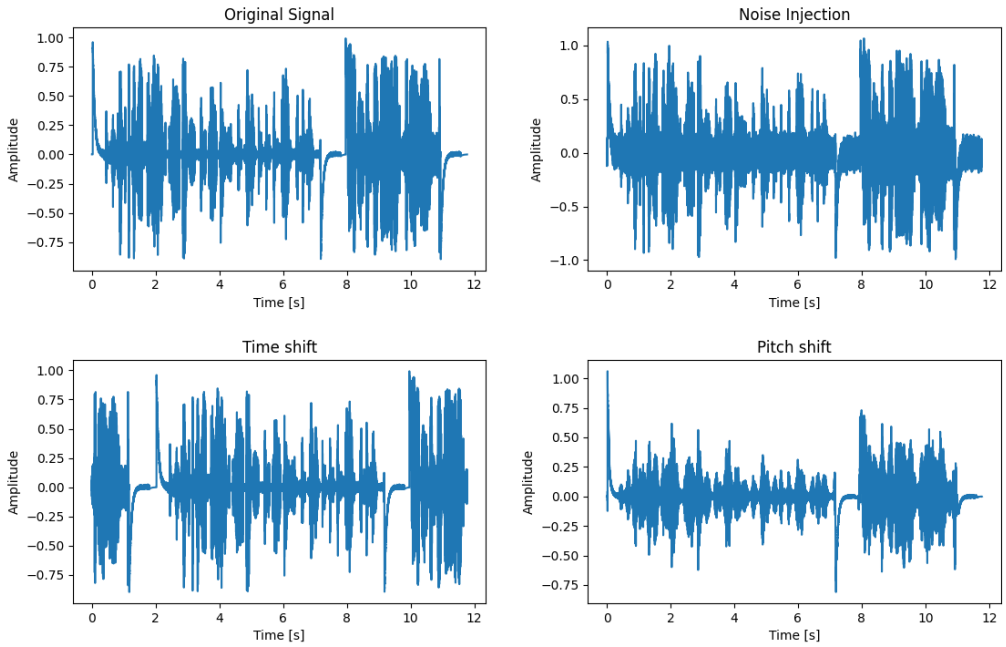


**Figure 4.9:** Augmentation techniques.

This chapter provided an overview of the used datasets in this implementation. The main source of audio files was the dataset by the ATCO2 project.

The ATCO2 dataset contains hours of air traffic communication. Ground truth labels for speech detection were provided in the files of the dataset and used to generate labels for training the models.

The dataset did not contain labels for push-to-talk so an algorithm was proposed. The first algorithm was not efficient enough to detect the push-to-talk events. Another approach was proposed and used for final PTT detection. The PEEK algorithm can correctly detect push-to-talk events even in noisy environments.

Finally, augmentation techniques are introduced and explained.

# Chapter 5

# Proposed solution and Implementation

This chapter will provide an overview of the thesis's goal, implementation pipelines, used technologies, and selected architectures for neural networks.

## 5.1 Definition of the Task

The primary goal of this thesis is to construct a neural network that is capable of both simultaneous predictions of speech and non-speech segments, as well as detection of the PTT event. The process of getting predictions for two or more tasks from a neural network is called multitask learning [8].

The capacity of doing the task with a smaller number of models is one of this strategy's main advantages. The ability to make predictions while multitasking comes naturally to humans. Analysing an object and determining its characteristics simultaneously is done without much thought. Such things as predicting two outcomes together can be learnt by neural networks. The learning process could be enhanced due to this strategy, considering training with more context. It is quite simple, the more data provided, the better generalization and learning of the neural network. But there are also downsides to this strategy. It could be challenging to simultaneously train a neural network for two tasks if the objectives are incompatible.

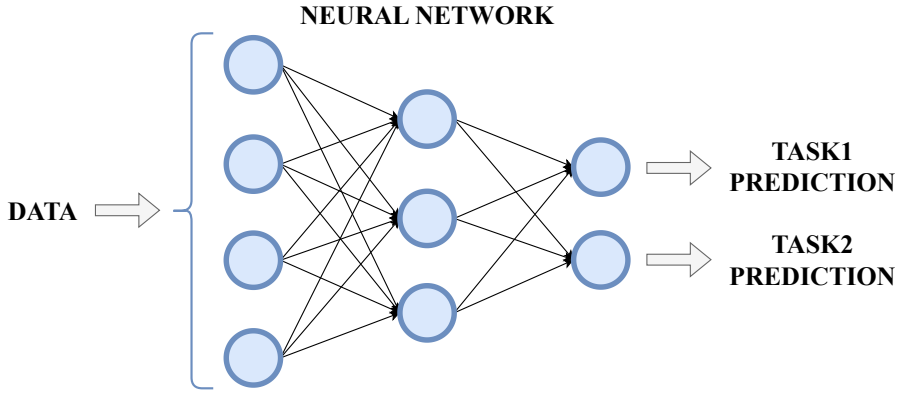


**Figure 5.1:** Abstract visual example of a multitask neural network.

In the next section 5.3 it will be talked about creating such architecture and the adaptation process of existing implementations will be explained.

## 5.2 Used Technologies

The program is implemented in the programming language Python. Python has various libraries which can be used in the process of training and validating the neural network. This program is using library `PyTorch`[1] which is commonly used in this field. It contains various methods that lead to efficient work with neural networks.

For audio processing, the most frequently used libraries were `librosa`[2], `numpy`[3] and `scipy`[4]. Each one contains various functions which help to work with the audio.

## 5.3 Selected Architectures

The architecture of each neural network is presented and discussed in this section. Each neural network consists of connected layers, and each time data passes through a layer, an activation function is applied. This process was explained in chapter 2. How supervised training operates is demonstrated in figure 5.2. There are several steps in the training process. To sum all it up, pre-processed data are fed into a neural network and the model modifies its weights based on the difference between ground truth and predicted output.

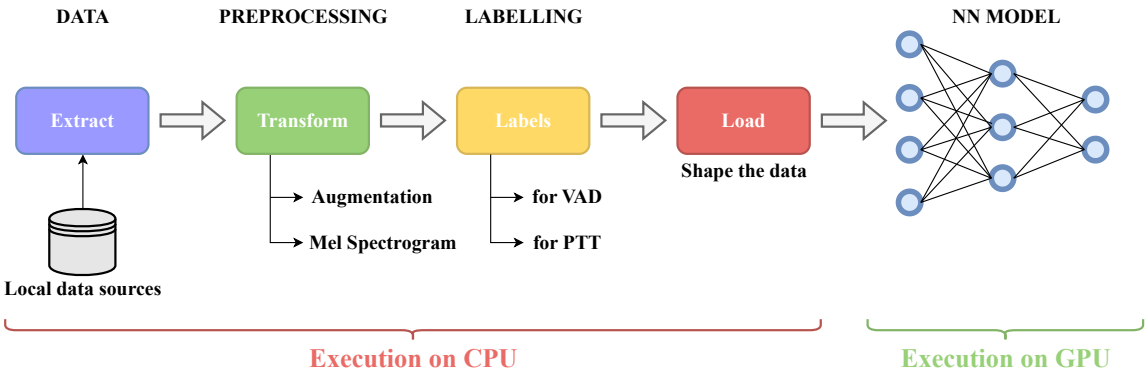


**Figure 5.2:** Abstract visual example of a multitask neural network.

The first experimental neural network is a basic feed-forward neural network with two hidden layers. To further understand the neural network, this step was necessary.

### Proposed Architectures for Initial Experiments

The implementation was centred on experimenting with various neural network designs. Architecture refers to the design of the model. A model can be represented in a number of units and their pattern connection.

To get acquainted with the neural networks, the first proposed architecture was rather simple. It was a basic feed-forward model. This model was made up of two hidden layers.

[1] see https://pytorch.org/
[2] see https://librosa.org/doc/latest/index.html
[3] see https://numpy.org/
[4] see https://scipy.org/

As seen in figure 5.3, the model was quite basic, which was convenient for grasping knowledge about neural networks.

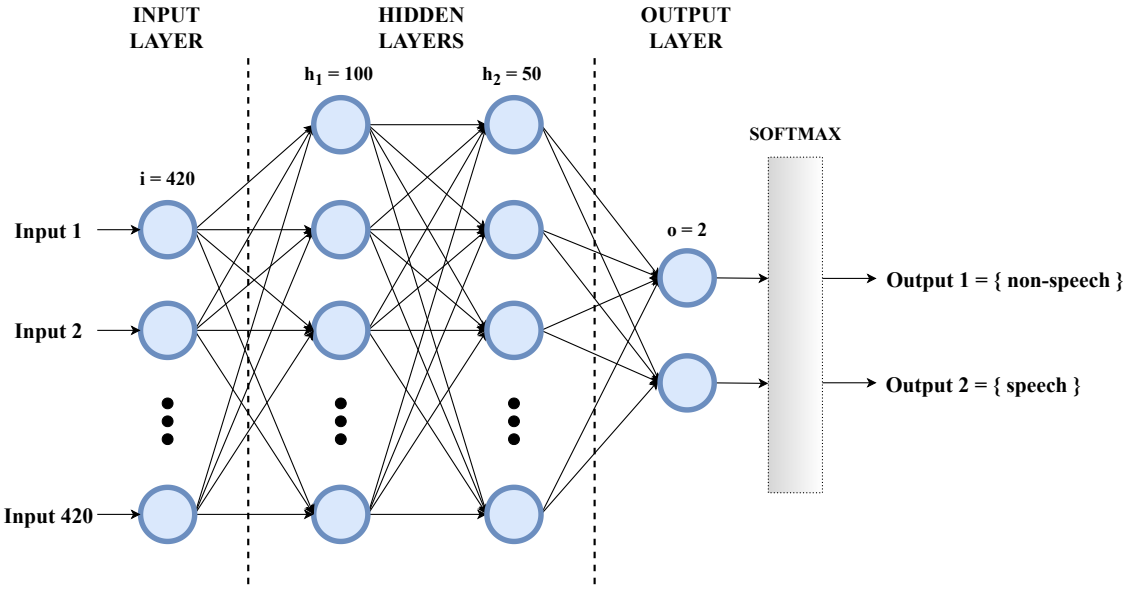


**Figure 5.3:** Architecture of a simple neural network

The model was trained only to distinguish between speech and non-speech frames before constructing a proper algorithm for push-to-talk recognition. A collection of frames represents the input to this neural network. Each frame generated during the Mel Spectrogram creation process represents a set of frequencies. Frames could represent different meanings in different locations. That is why it was necessary to provide further context. Adjacent frames have been joined together to produce more informative surrounds. As a result, input to the neural network is a sequence of 21 frames (210 ms) rather than a single frame (10 ms). This is also due to the fact that one phoneme is at least 100 ms long 3.2. A group of frames with 20 frequencies each after removing one dimension created an input array of length 420 (figure 5.3).

ReLu was used as an activation function. It was chosen because it is one of the most popular and used activation functions due to its non-linearity and robustness to vanishing gradients. It is also computationally undemanding. On the final layer, the Softmax function was applied. The result was a probability for each class.

After implementing the PEEK algorithm, multitasking was added to the neural network. It has the same architecture as the previous model, but with an additional output layer for PTT detection (figure 5.3).

To further understand various types of neural networks, another model was created. Because the neural network is basically fed by visual representation[5] of audio, it is clear that adding convolutional layers could improve performance.

The second experimental model has two convolutional layers. The input shape changed as well. The input was not flattened as it was in the FNN approach but kept in a two-dimensional shape. This means that now neural network works with image-like input. Input is a two-dimensional array of size $21 \times 20$. An explanation of this choice was explained earlier in the text. This neural network is shown in figure 5.4.

[5]Spectrogram.

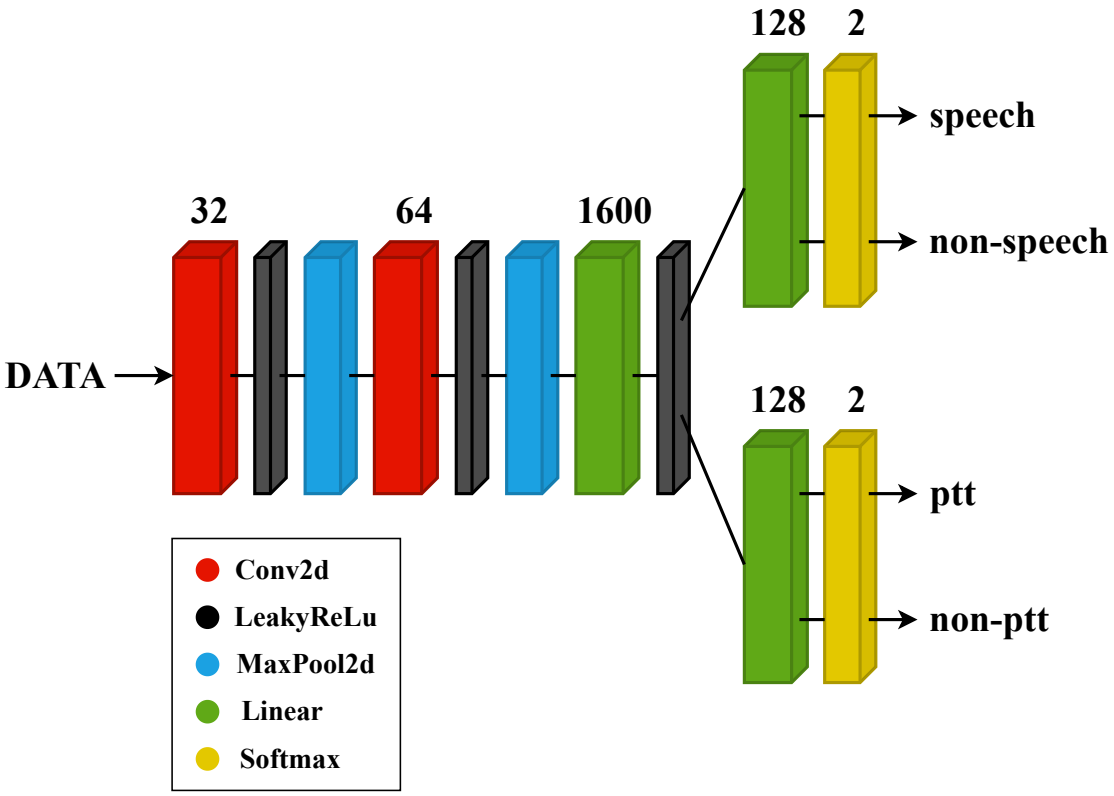


**Figure 5.4:** Architecture of a convolutional neural network with two-dimensional input.

The first convolutional layers contain 32 convolutional filters and the second layer contains 64 filters. Between those two layers, the activation function ReLu is applied and the max-pooling layer is provided. Max pooling is one type of pooling operation which takes responsibility for highlighting the significant features as well as reducing spatial dimensions.

The creation of model architectures was done by combining learnt knowledge, experimenting and testing the NN models. Proposed models are quite simple and not too complex to learn. Simple models can often outperform complex models. The training process and experiments with those architectures will be further explained in chapter 6. Various architectures can change the accuracy of detecting voice activity and push-to-talk events.

### Adaptation of GPVAD

Because neural networks need tons of data to properly learn and generalize to new data, it is conventional to use pre-trained models. It is possible to use some knowledge of previously trained models. The choice of those neural networks depends only on the specific problem. In this implementation, the GPVAD approach was adapted [9, 10].

GPVAD[6] uses weak labelled supervision on clip level, which can outperform strongly supervised VAD during the evaluation of real-world data or noisy datasets.

GPVAD's performance is not significantly better than traditional approaches[7], but trained models do a good job of detecting speech in noisy environments. This characteristic is very useful in terms of detecting speech in air traffic communication, which is

[6] General purpose voice activity detection.
[7] For example models which used Hidden Markov Model as frame-level labels.

an environment with high background noise. The teacher-student approach used in this implementation consists of two main steps:

- Training the teacher model with weak labels on clip level. This means that the knowledge of the teacher is to inspect the audio as a whole.
- Using knowledge of the teacher model in the process of training the student model. The teacher provides guidance for the student model on unlabelled data on the frame level.

In this thesis, the CRNN model will be adapted. The adapted CRNN model consists of various layers and additional functions, as shown in figure 5.5.

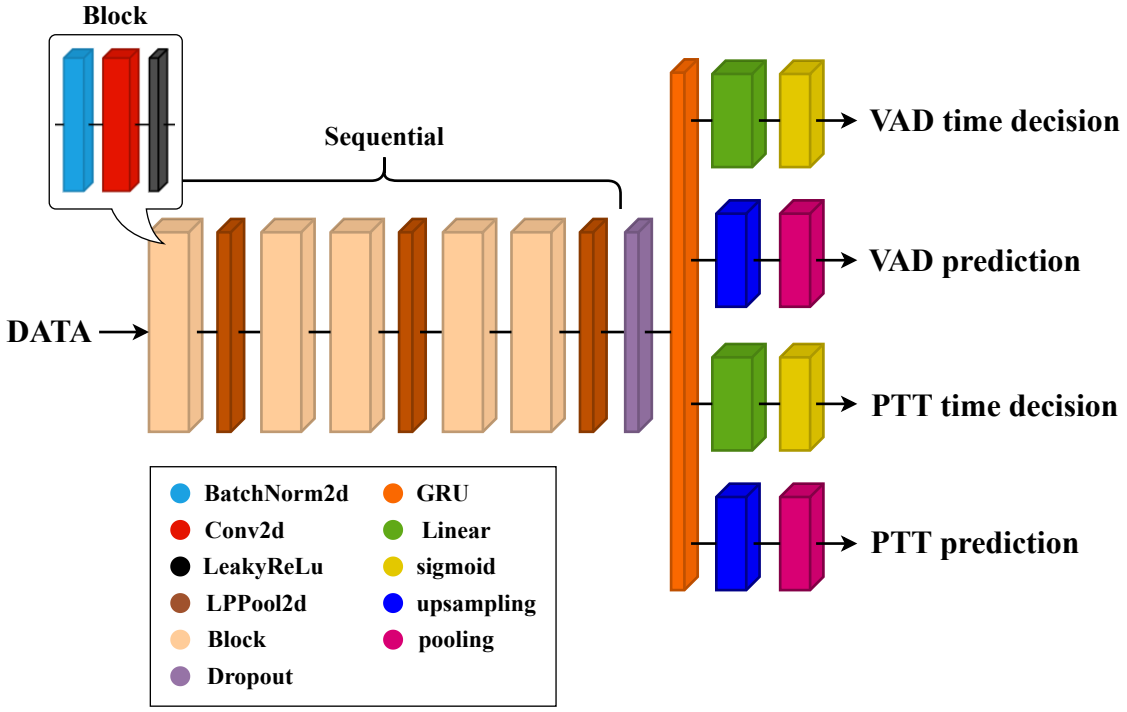


**Figure 5.5:** Adapted CRNN model from GPVAD implementation. The time decision represents a frame-level prediction for the frame being a certain class. The overall prediction outputs represent the probability of each event happening in the audio.

CRNN architecture has many two-dimensional convolutional layers as well as pooling layers and a recursive unit. Convolutional layers in combination with the recursive unit have proven themselves useful in supervised training for voice activity detection. The architecture is the same as before the adaptation, but now it contains one additional output layer for PTT detection. The adaptation of this model consisted of adding another layer to the output layer, which served the purpose of PTT detection.

Input to this model is a Mel spectrogram as well, but with different parameters, as shown in section 4.2. The output of this neural network is a time decision for each frame of the audio for voice activity and push-to-talk events. Time decision for voice activity represents a two-dimensional array that contains the probability of each speech and non-speech event determined individually for each class. The neural network provides time decisions for PTT

as well. The clip output represents the overall probability of each event occurring in the audio.

The process of extracting features remained unchanged. On the other hand, the process of creating labels was slightly modified. Voice activity labels are generated from predictions of a pre-trained neural network. PTT frame labels are created by using the PEEK algorithm, which was explained in subsection 4.4.2. These frames labels are nearly 100% accurate (see 4.3). The clip level was created by using a linear pooling function on frame-level labels.

The training process, which will be further explained in chapter 6 needed adjustments too. The data loading process was modified to load these new labels. The loss function which was computed on the frame level was modified to compute loss for each VAD and PTT output correspondingly.

GPVAD approach contained lots of implementation mistakes. These mistakes did not influence the overall performance of the model, but because of them, the adaptation was quite complicated. The main problems were incorrectly named files and missing files in some directories. Repairing the implementation needed knowledge of the whole project. The overall performance is sufficient, but according to the manual some things did not work, and it was necessary to re-implement them. Problems and resolutions will be further discussed in chapter 6.

This chapter provided an overview of the used models in the thesis. Two architectures of neural networks were introduced. The models are suited for voice activity and push-to-talk detection. The models were proposed in order to get familiar with machine learning.

Finally, an introduction to GPVAD was provided. GPVAD is a neural network which is capable of detecting speech even in a noisy environment. This model is adapted to match proposed experiments.

# Chapter 6

# Experiments and Evaluations

This chapter provides a complete overview of all experiments. The training process of the neural networks will be described, and the distribution of the dataset as well. It will discuss used evaluation metrics. Finally, a comparison between models will be provided.

## 6.1 Evaluation Metrics

Evaluation metrics are utilised in two stages in a basic classification problem: training and testing.

Metrics are used to enhance the classification algorithm during the training stage. They can be used as discriminators to determine and select optimal solutions that can predict future evaluations more accurately. Metrics are used in the testing phase to evaluate the overall performance and effectiveness of the classifier on unseen data.

Each stage requires a dataset containing previously unseen data. Unseen data are commonly divided into test and validation datasets. After training, a test dataset is used to evaluate the model's performance.

Computing a loss function and accuracy on a validation dataset are common techniques for evaluating the progress of neural network learning. The validation dataset contains data that was not included in the training set in order to test whether the model is improving in its predictions for previously unknown data. If the validation loss value starts increasing, it indicates that the model is starting to overfit the training data and that training should be interrupted.

Following the training process, the model is evaluated on the test dataset. It is determined by how well the model predicts and how general it is.

Some of the most common evaluation metrics [12] used for confirming whether a model proves to be accurate are:

- **Accuracy** computation is the most commonly used method for determining model correctness. It is a ratio of the correct predictions or named as True Positive (TP) and True Negative (TN) over all evaluated instances, meaning over False Positive (FP) and False Negative (FN) predictions as well 6.1.

$$\frac{TP + TN}{TP + FP + TN + FN} \tag{6.1}$$

- **Precision** (P) value is representing a ratio between actual or True Positive (TP) values and all predicted positive values from the model, including False Positive (FP) predictions 6.2.

$$P = \frac{TP}{TP + FP} \tag{6.2}$$

- **Recall** (R) is a value obtained by calculating how many True Positives (TP) were predicted by the model. False Negative (FP) values represent incorrect negative predictions 6.3.

$$R = \frac{TP}{TP + FN} \tag{6.3}$$

- **F1 score** represents a mean value of precision and recall 6.4.

$$F1 = 2 \cdot \frac{PR}{P + R} \tag{6.4}$$

- **Confusion matrix** represents a summary of each True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN) prediction on currently evaluated data (table 6.1).

| | **True Positive Class** | **True Negative Class** |
|---|---|---|
| **Predicted Positive Class** | TP | FP |
| **Predicted Negative Class** | FN | TN |

**Table 6.1:** Confusion matrix structure.

- **AUC-ROC** method reflects the overall performance of a classifier. AUC or the area under a ROC[1] curve is a value in a range [0.5,1.0] where 0.5 corresponds to a random classifier and 1.0 to a perfect classifier. The ROC curve is the trade-off between True Positives (TPR) and False Positives (FPR) (figure 6.1).

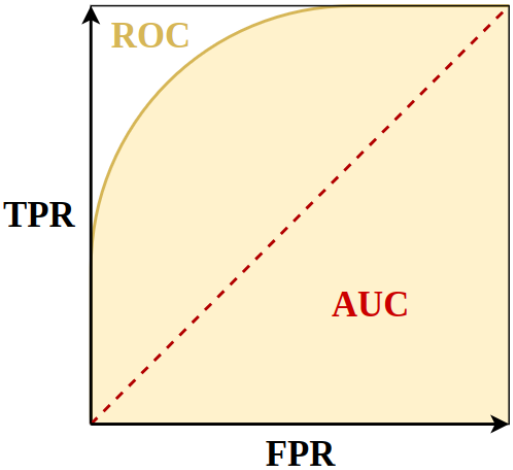


**Figure 6.1:** AUC-ROC curve.

[1] Receiver operating characteristic.

## 6.2 Training and Validation of Selected Models

The training process of the neural network can be really challenging task. Various factors affect the training process. A careful selection of hyperparameters needs to be done. Dataset robustness is also a huge factor to determine whether the training was successful or not. Generally speaking the more data the better performance of the model. High-quality labels are crucial in supervised training.

The process of training the neural networks in this thesis consisted of data pre-processing, which includes feature extraction and label generation.

This section is divided into two parts. The first part (see 6.2.1) will talk about the training and validation process of proposed experimental models. These models will then be evaluated and compared with each other. The second part (see 6.2.2) will talk about the training process of GPVAD reimplementation. It will include an overview of the model's performance without adding another layer for push-to-talk detection. Later, the fine-tuning process will be explained.

Finally, the accuracy of these models will be compared in section 6.3

All the training process was done on GPU. The dataset was divided into three parts. Training data represented 70% of the dataset, and the testing and validation set both represented 15%.

In the unaltered ATCO2 dataset, the class distribution is 65% for speech and 35% for non-speech. This is not necessarily the best balance of the two classes, so it needed to be altered. The specific numbers can be found in the following table:

| | **Speech (%)** | **Non-Speech (%)** |
|---|---|---|
| **Training Data** | 47.60 | 22.40 |
| **Testing Data** | 10.20 | 4.80 |
| **Validation Data** | 10.20 | 4.80 |

**Table 6.2:** Speech and non-speech frames distribution in the original ATCO2 dataset without changes.

In the ATCO2 dataset, PTT frames have significantly lower representation than other classes (table 6.3).

| | **Speech (%)** | **Non-Speech (%)** | **Push-To-Talk (%)** |
|---|---|---|---|
| **ATCO2 Dataset** | 66.35 | 32.81 | 0.83 |

**Table 6.3:** Speech, non-speech and push-to-talk frames distribution in the original ATCO2 dataset without changes.

### 6.2.1 Training and Validation of Experimental Models

It is necessary to define some concepts first. FNN is an abbreviation for feed-forward neural network, which was proposed in section 5.3. CNN is a neural network that processes input frames as a two-dimensional array (see 5.4). The models require a proper dataset, on which they can be learnt.

After the neural network's architecture has been established and the data has been pre-processed and labelled, the training phase can officially start. It is vital to note that this procedure takes quite a while because an ideal solution is rarely found on the first try. Experimenting and analysing neural networks is time-consuming.

#### Feed-Forward Neural Network

The simple feed-forward model is trained first. There are three experiments in total:

- **Experiment A**: only on Voice activity detection (see 6.2.1).
- **Experiment B**: only on Push-to-talk detection (see 6.2.1).
- **Experiment C**: on both Push-to-talk and voice activity detection (see 6.2.1).

Selected hyperparameters are shown in the following table:

| | **LR** | **LF** | **Optimizer** | **BS** | **Epochs** |
|---|---|---|---|---|---|
| **FNN** | 0.001 | Cross Entropy | Adam | 32 | 5 |

**Table 6.4:** Hyperparameters for FNN training. LR – learning rate, LF – loss function, BS – Batch Size.

To use regularization of the training as well, an *Early Stop* mechanism was added to the process. The Early Stop is initialised with a minimal difference of 0.15 and patience 10. This method focuses on terminating training before overfitting. Overfitting is a frequent problem in the training of neural networks. This happens when the model has learned to predict the training data perfectly but is unable to predict the new, unknown data. This indicates that the model failed to generalise the data.

To experiment with neural networks, the model was first trained solely to predict speech activity.

Experiment A, which concentrated on VAD was trained on approximately 90,000 frames with distribution between speech and non-speech was 50% for each class. For experiment B, the number of frames was reduced to approximately 40,000 with 50% for push-to-talk and 50% for non-push-to-talk. Finally, for experiment C, the number of data frames is approximately 50,000.

Each evaluation is done by comparing frame to frame. This leads to a more precise evaluation on a frame level.

### Experiment A

For experiment A, the dataset was limited to around 90,000 frames with a distribution of 50% for each class. The frames were divided into training, testing and validation data by recordings to avoid providing the neural network by mistake in neighbouring frames. The distributed frames were shuffled.

The accuracy of this neural network after one epoch is 77.2% which is pretty sufficient for the first try. Precision and Recall of this neural network is around 80% for each.

Despite the fact that the architecture was simple, and the dataset was relatively small, the results were satisfactory. Speech is labelled with value one and non-speech with value zero. The training was done in one epoch.

Validation of the neural network on an unknown recording resulted in 88.9% accuracy. This accuracy was calculated by comparing the labels from `CNET` file and an index of softmax output from the neural network with higher (figure 6.2).

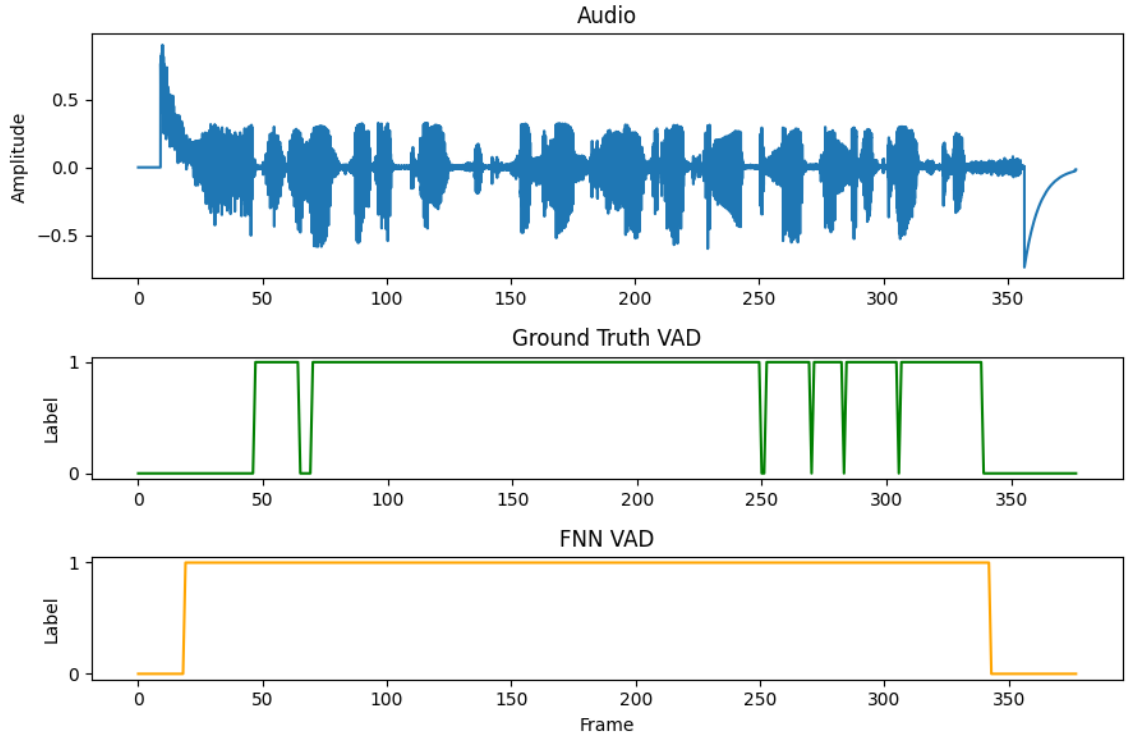


**Figure 6.2:** Experiment A: Validation of unknown audio on voice activity detection.

### Experiment B

Experiment B dwells in training the FNN only on push-to-talk detection, without distinguishing speech and non-speech frames. The distribution between push-to-talk and non-push-to-talk classes is 50% for each. Non-push-to-talk frames were randomly selected from a set of speech and non-speech frames to match the length of the PTT set. This may not be the perfect distribution considering that the validation data will never contain a recording in which 50% is a push-to-talk event, but for this experiment, it was not taken into consideration. Modification of the distribution will be taken into consideration in section 6.2.2. The hyperparameters for training remained the same as were in the first experiment. The accuracy of this neural network on the test dataset is 96.6%.

Validating neural network on unknown recording resulted in 99.3% (figure 6.3). Despite a few false alarms, the overall performance is decent.

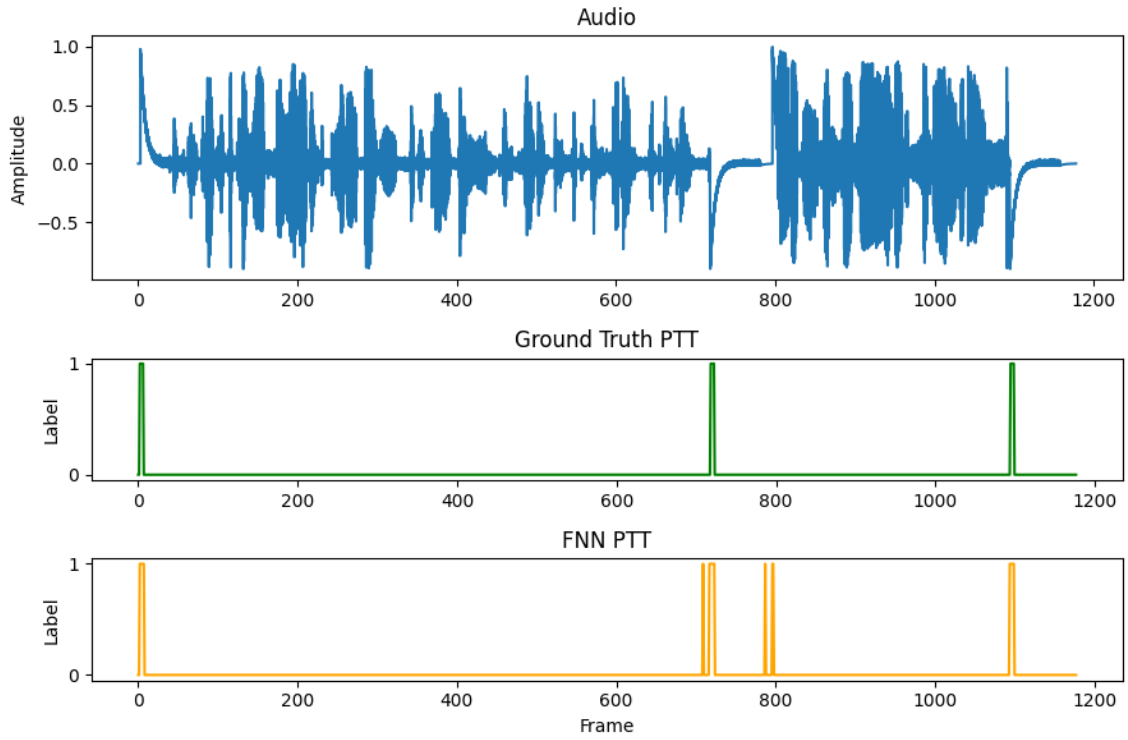


**Figure 6.3:** Experiment B: Validation of unknown audio on push-to-talk detection.

### Experiment C

For experiment C, another output layer was added to the neural network to detect both push-to-talk and voice activity at the same time.

Every PTT frame was stored, and speech and non-speech frames were chosen randomly to match the length of the PTT frames. Because the final evaluation is done on audio recordings which contain more non-push-to-talk frames, the distribution of the classes was altered. The ratio between speech and non-speech frames remained the same. Push-to-talk frames represented 20% of the dataset.

This distribution is probably not the most balanced, but for now, it is sufficient. Training the neural network resulted in a 94.10% accuracy for push-to-talk detection and 64.88% for voice activity detection. The early stop regularisation element was triggered during the third epoch. Validating the whole audio with PTT events ended up as a partial success because of one false alarm for push-to-talk (figure 6.4). FNN with 99.15% accuracy predicted locations of push-to-talk events and with 74.26% accuracy predicted speech segments on unknown audio. This accuracy is discussable due to marking nearly everything in the signal as a speech.

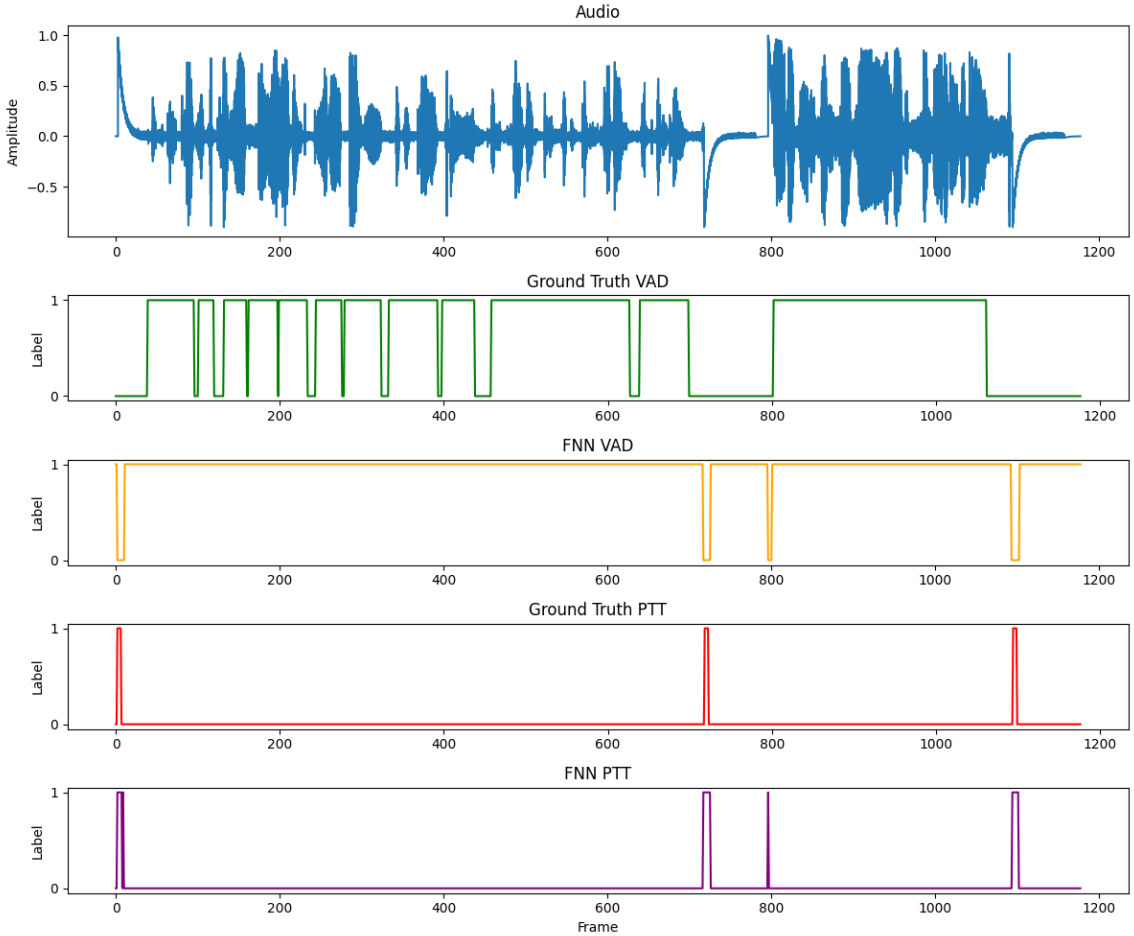


**Figure 6.4:** Experiment C: Validation of unknown audio on voice activity and push-to-talk detection.

For this experiment, the ROC curve was calculated on each task 6.5. A Receiver Operating Characteristic Curve (ROC) is a common method for summarising classifier performance across a range of true positive (TP) and false positive (FP) error rates [25].

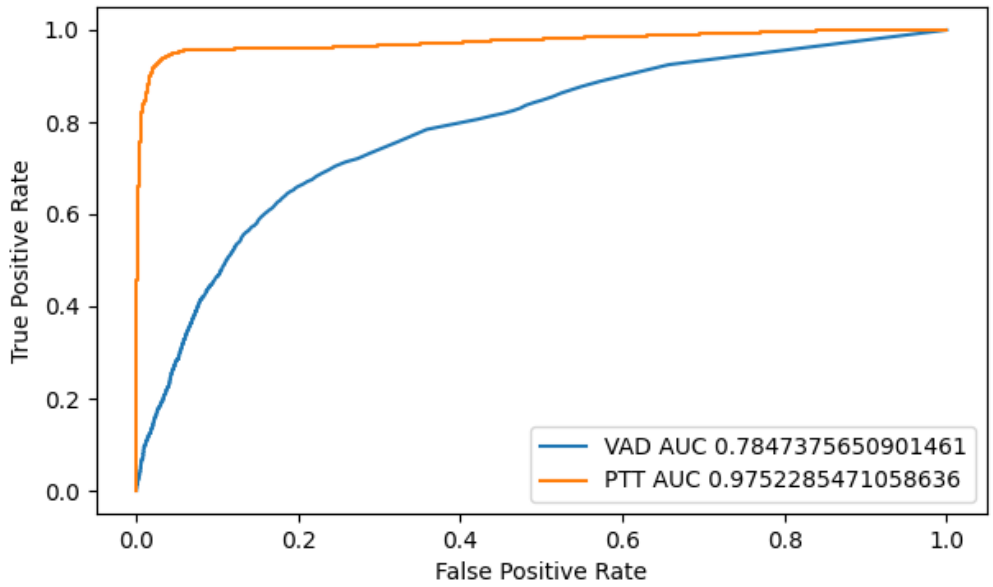


**Figure 6.5:** AUC-ROC evaluation.

In summary, FNN performed with **77.21%** on voice activity detection (Experiment A), **96.60%** on push-to-talk (Experiment B) and **94.10%** for push-to-talk and **64.88%** on multitasking (Experiment C).

The models were trained approximately for one hour and used almost 1 GiB of memory on the GPU.

Because this feed-forward architecture served only as a stepping stone, it was unnecessary to continue with experiments on this architecture.

### Convolutional Neural Network

The second experimental neural network is a convolutional neural network fed by signal frames kept in a two-dimensional array. This neural network could possess an advantage over the FNN thanks to its convolutional layers.

This CNN model is trained only on multitasking for two epochs. The training process was done on the same dataset, as it was used for experiment C (see 6.2.1).

The hyperparameters are presented in the following table:

| | **LR** | **LF** | **Optimizer** | **BS** | **Epochs** |
|---|---|---|---|---|---|
| **FNN** | 0.001 | Cross Entropy | Adam | 32 | 5 |

**Table 6.5:** Hyperparameters for CNN training. LR – learning rate, LF – loss function, BS – Batch Size.

After testing the neural network on a test dataset, the accuracy of the CNN model is 96.80% for push-to-talk detection and 82.00% for voice activity detection. Validating the model on the unknown audio resulted in 99.40% accuracy for push-to-talk and 87.59% for voice activity (figure 6.6).

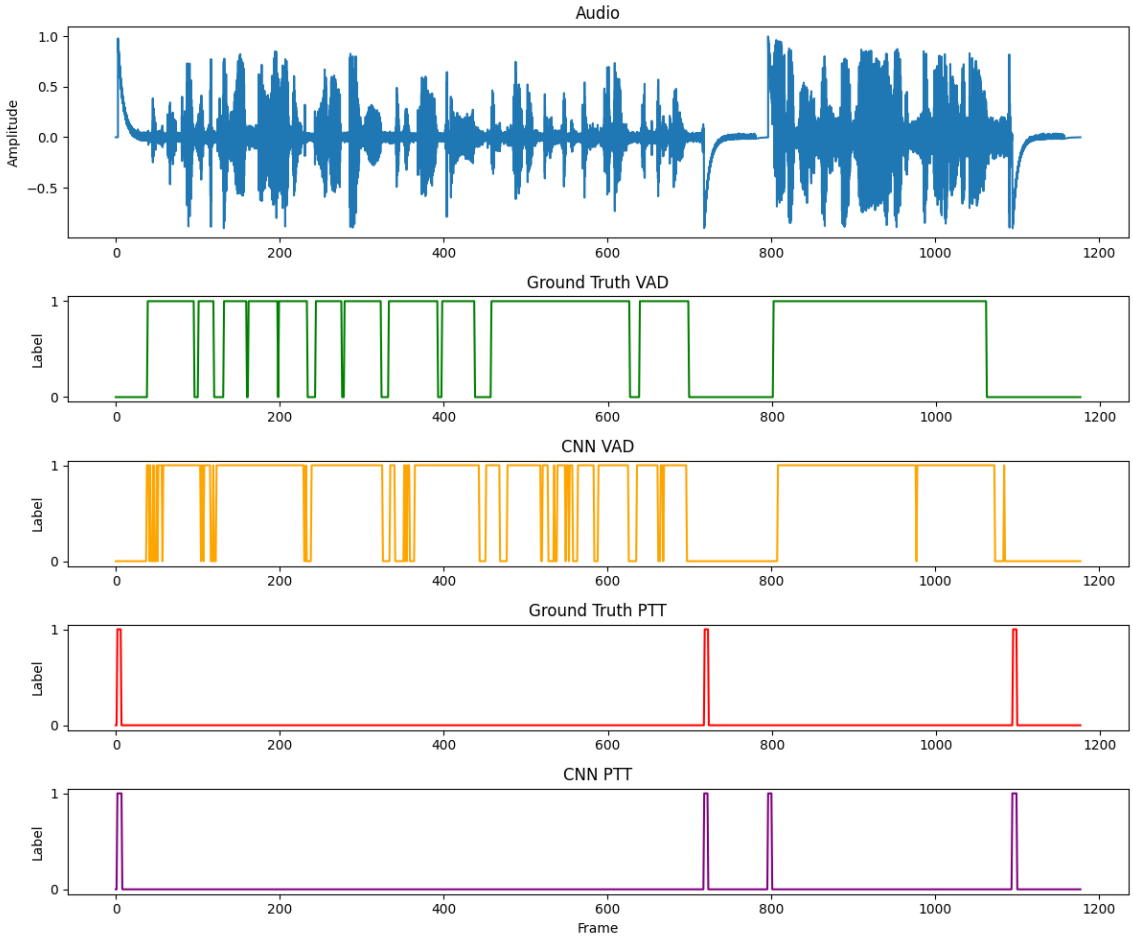


**Figure 6.6:** Validation of unknown audio on voice activity and push-to-talk detection.

The AUC-ROC evaluation resulted in 0.89 for voice activity detection and 0.978 for push-to-talk (figure 6.7). The evaluation was computed by a function from `sklearn` library[2].

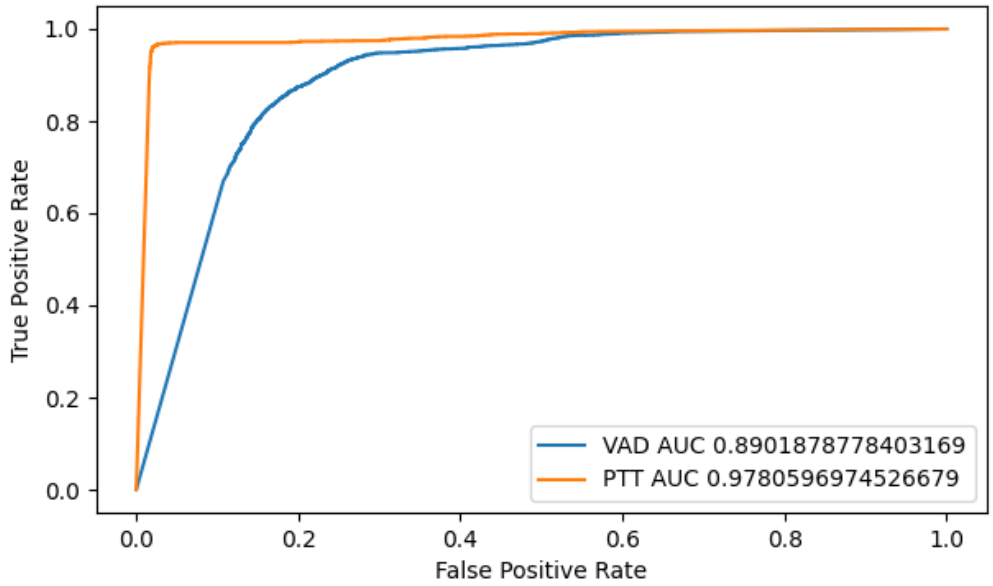


**Figure 6.7:** AUC-ROC evaluation curve computed on both push-to-talk and voice activity detection.

a confusion matrix was employed to evaluate the correctness of the model (figure 6.8). The matrix's values represent average values over every batch in the test dataset.

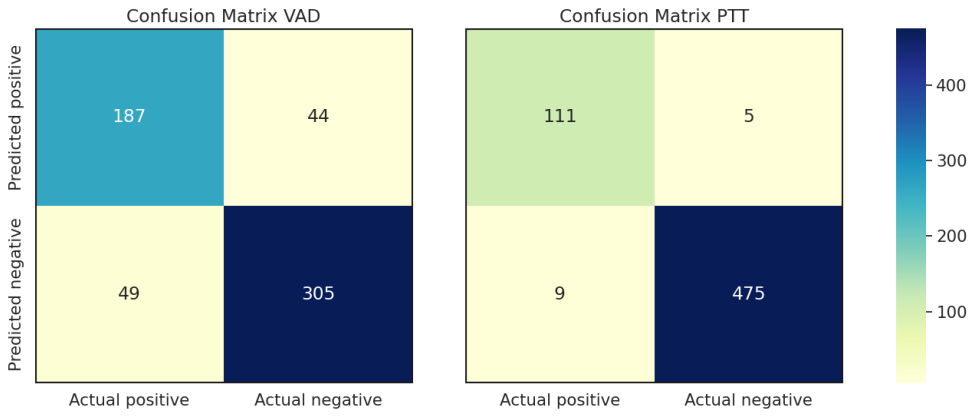


**Figure 6.8:** Confusion matrix of the model for each task.

The model was trained for approximately half an hour with a memory usage of 900 MiB on GPU.

[2]see scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html

After training the proposed experimental models, a comparison between each result is needed. Firstly, the three experiments on the FNN model are evaluated.

| | | Acc. (%) | Rec. (%) | Prec. (%) | F1 (%) | AUC |
|---|---|---|---|---|---|---|
| **A** | **VAD** | 77.21 | 79.00 | 85.70 | 82.20 | 0.718 |
| **B** | **PTT** | 96.60 | 91.30 | 92.50 | 91.30 | 0.977 |
| **C** | **VAD** | 64.88 | 39.40 | 72.50 | 50.90 | 0.780 |
| | **PTT** | 94.10 | 91.86 | 86.00 | 87.80 | 0.975 |

**Table 6.6:** Evaluation metrics of FNN model on different experiments.

Table 6.6 represents evaluation metrics for each experiment mentioned earlier in this section. Each percentage is an average value over test batches results. They could not be compared due to being learned for different tasks, but the overall most successful experiment with this architecture was the experiment with push-to-talk detection.

FNN and CNN were trained to provide multitask predictions under the same conditions and with the same dataset. It is possible to compare their performance (table 6.7). All of the values are an average of each metric over every test batch.

| | | Acc. (%) | Rec. (%) | Prec. (%) | F1 (%) | AUC |
|---|---|---|---|---|---|---|
| **FNN** | **PTT** | 94.10 | 91.86 | 86.00 | 87.80 | 0.975 |
| | **VAD** | 64.88 | 39.40 | 72.50 | 50.90 | 0.780 |
| **CNN** | **PTT** | 96.88 | 94.24 | 90.33 | 92.06 | 0.978 |
| | **VAD** | 81.99 | 81.12 | 77.98 | 79.47 | 0.890 |

**Table 6.7:** Evaluation metrics of FNN and CNN models. The comparison on task is provided individually.

It is possible to state, that architecture with convolutional layers helped the model to predict more correct outputs. The most noticeable difference between the FNN approach and the CNN approach is validation on audio, which does not contain any PTT event. The CNN model can accurately mark each frame as not push-to-talk, but FNN causes one or two false alarms.

### 6.2.2 Training and Validation of GPVAD Model

The GPVAD implementation provides an effective pre-trained model for voice activity detection. This model was trained on a large amount of data and can correctly distinguish speech segments. The GPVAD CRNN architecture is complex and contains various layers such as convolutional layers and a recurrent unit.

The main difference between GPVAD's training and training on experimental models is that the input to the models does not consist of a group of local frames, but rather a global group of frames in the whole audio (see chapter 4). This provides a context for the neural network on a global level of the whole audio.

An architecture of the CRNN model was used in order to train the neural network on push-to-talk detection. A dataset for this training consisted of various audio recordings. These recordings are divided into two classes, one that contains no push-to-talk event and a second that contains at least one. The ratio between frames that contain push-to-talk events and ones that do not contain the event is 10:90. This seemed reasonable because the final evaluation or usage of this model is for audio recordings that do not contain a big amount of push-to-talk events. When training on balanced distribution, the results were poor. Experimenting with various distributions resulted in choosing approximately a 10:90 ratio.

The overall number of recordings for this training was 1000. Each recording was approximately 2 seconds long. The whole dataset is something little over 30 minutes long.

| | **Push-To-Talk** | | **Non-PTT** | |
|---|---|---|---|---|
| | **Frames (%)** | **Recording (%)** | **Frames (%)** | **Recordings (%)** |
| **Dataset** | 11.55 | 58.60 | 88.45 | 41.40 |

**Table 6.8:** Dataset distribution.

The push-to-talk event lasts for 100 ms. During label generation, it is also checked if the peak in the frame is not too close to the end of the frame. This could possibly trigger a false alarm because a majority of the frame is the non-push-to-talk segment. If the peak[3] is in the last 15% of the frame, the frame is considered non-push-to-talk.

The hyperparameters for this training are shown in the table 6.9. The training was done in 10 epochs.

| | **Learning Rate** | **Loss Function** | **Optimizer** | **Batch-Size** |
|---|---|---|---|---|
| **GPVAD PTT** | 0.001 | Cross Entropy | Adam | 64 |

**Table 6.9:** Hyperparameters for GPVAD training.

[3]Peak is a specific sample that was considered as local maximum by the PEEK algorithm provided in the chapter 4.

Weights were initialised by loading one of the provided models in the GPVAD implementations. The training process of this model resulted in 83% accuracy and 80% F1 score. The model was validated on audio and resulted in 94.7% accuracy 6.9.

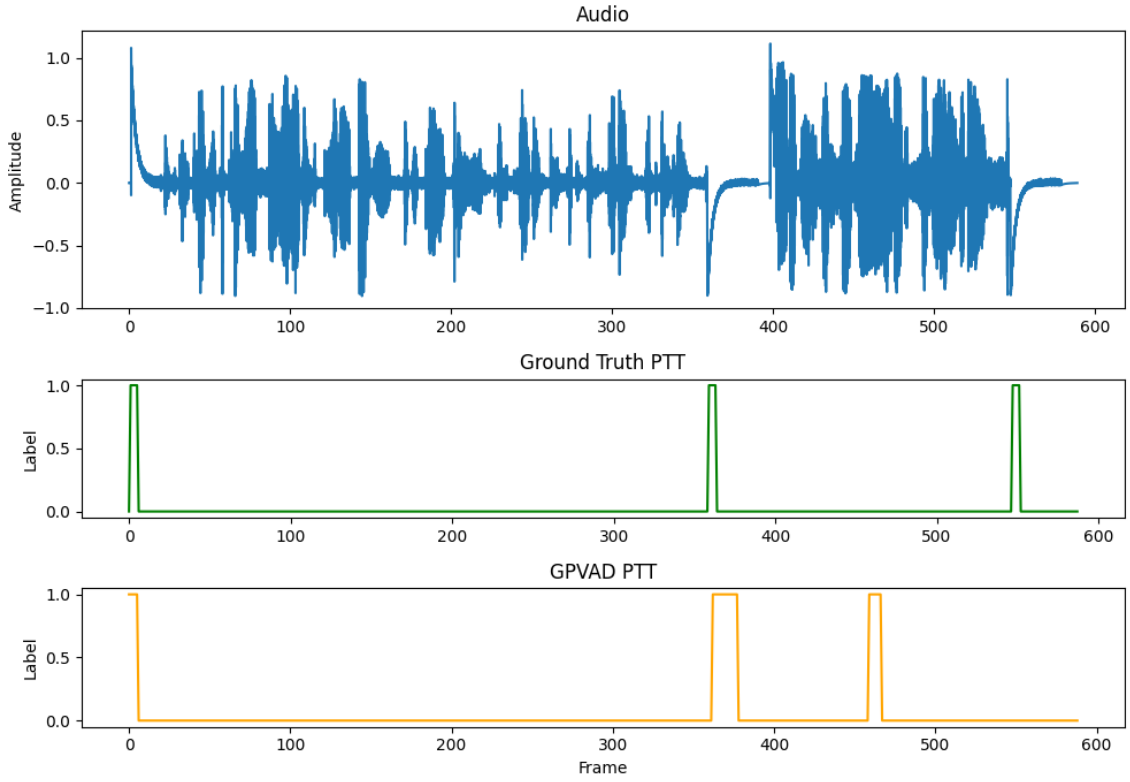


**Figure 6.9:** Validation of unknown audio on push-to-talk detection.

Finally, a GPVAD with adaptation to predict clip level and frame level predictions for push-to-talk is trained. The hyperparameters remain the same as for push-to-talk training (table 6.9) except for the learning rate. The learning rate changed to 0.01. The dataset distribution needed to be changed to be more balanced. The previously mentioned ratio was not suitable for this training and the training resulted in many false negative predictions due to lack of PTT events. The dataset was augmented, so it doubled in the number of recordings. The dataset contained 6000 audio files which represent nearly three and a half hours. The distribution on the first dataset was not sufficient for this problem.

The weights were initialised by loading the `sre` model provided in the GPVAD implementation. This model can accurately predict speech segments, detect speech in noisy environments and can localise speech that is very quiet. The model is a good starting point in order to train a multitask model.

The training process lasted 50 epochs and resulted in a maximum of 66% accuracy in detecting the push-to-talk signal on a dataset with a 35:65[4] ratio. The fine-tuning model only on the last output layer was challenging. The neural network was inclined in predicting more non-push-to-talk events than the ground truth. When providing a dataset with a larger representation of push-to-talk events, the model was still inclined to predict non-push-to-talk events.

[4]push-to-talk frames : non-push-to-talk frames

Even though the weights of the new output layer look “healthy” with majority begin low weights (figure 6.10) [26]. The distribution of weights mimics the Gaussian distribution.

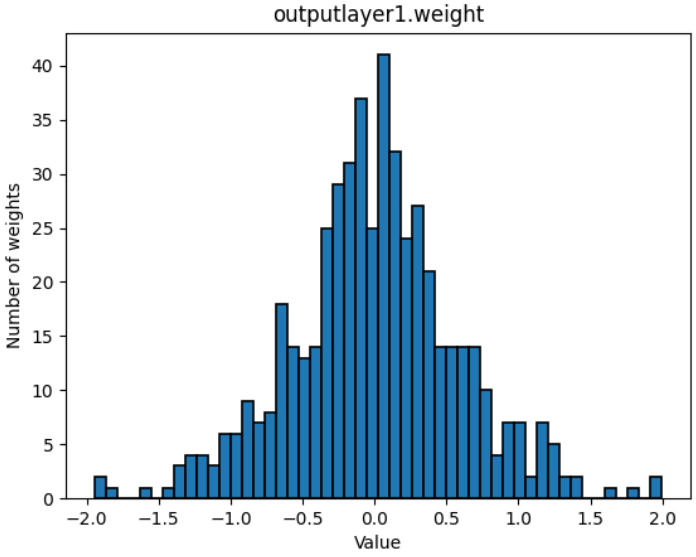


**Figure 6.10:** Histogram of weights on the new output layer for push-to-talk detection.

For post-processing, the only threshold that was used was if the PTT prediction is higher than the non-PTT prediction. This resulted in 93.9% accuracy but caused a false alarm during a speech segment and missed one push-to-talk, so it is discussable to take it into consideration (figure 6.11).

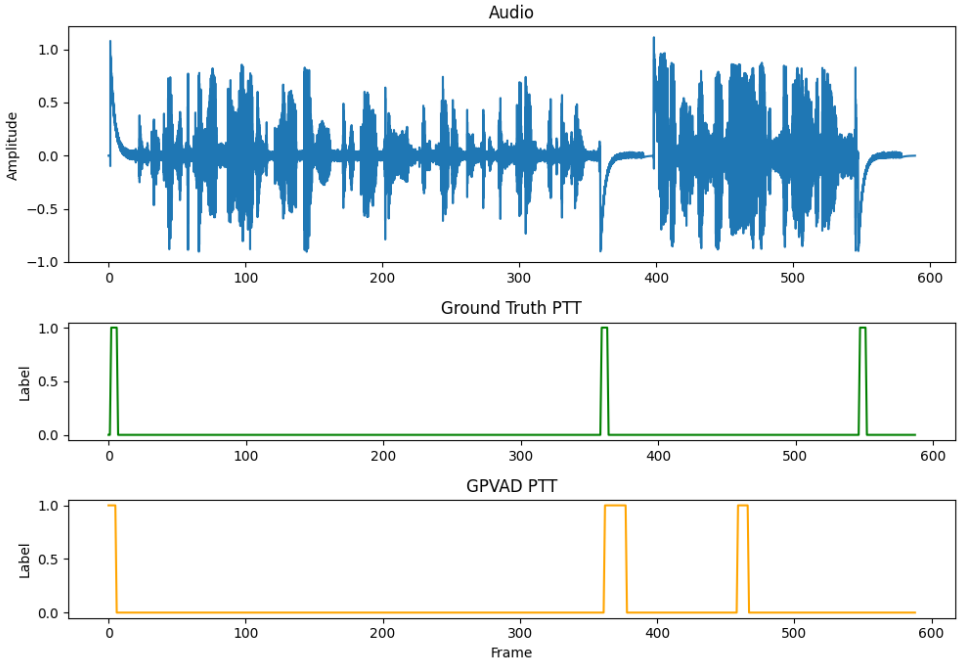


**Figure 6.11:** Validation of unknown audio on voice activity and push-to-talk detection.

## 6.3 Comparison Between Different Approaches

In this section, the used models are compared. The comparison can't be done the exact way as the FNN Experiment C and CNN because GPVAD was pre-trained with much more data. The baseline evaluation metrics for the used model `sre` are provided in the following table:

| | **Prec. (%)** | **Rec. (%)** | **F1 (%)** | **AUC** |
|---|---|---|---|---|
| **aurora_clean** | 96.844 | 95.102 | 95.93 | 98.66 |
| **aurora_noisy** | 90.435 | 92.871 | 91.544 | 97.63 |
| **dcase18** | 89.202 | 88.362 | 88.717 | 95.2 |

**Table 6.10:** The evaluation metrics for baseline model on different datasets, which is further explained in the original GPVAD paper.

CNN, FNN and GPVAD required different datasets in order to learn given tasks. The CNN and FNN were trained from scratch so they needed a reasonable distribution between all four classes. On the other hand, the GPVAD was fine-tuned on push-to-talk detection, so the dataset needed to be more robust on push-to-talk events.

Despite that at least some comparison can be provided. The FNN model achieved 94.10% accuracy for PTT detection and 64.88% for voice activity detection. The CNN model resulted in **96.88%** accuracy for push-to-talk detection and **81.99%** for voice activity detection. The GPVAD achieved 66% accuracy for push-to-talk detection.

This makes the CNN model the best in terms of performance and size. The model consists of 224,032 learnable parameters. GPVAD contains 677,569 learnable parameters.

The best model for sole push-to-talk detection is GPVAD with **83%** accuracy.

The honourable mention does to the proposed PEEK algorithm, which was crucial to achieving these results.

## 6.4 Possible Improvements and Future Work

GPVAD adaptation provided sufficient results. The next step in fine-tuning the model can involve experimenting with freezing only some weights. Another possible improvement is to experiment further with proper data distribution to obtain the most optimal and correct results. Due to inclining to the non-push-to-talk event, a weighted training could be efficient.

To enhance CNN correctness, a bigger and more robust dataset could be provided to achieve even better results.

After adjusting these two models, the proposed solution could help with detecting voice activity and push-to-talk segments in air traffic communication.

Another possible approach to detect push-to-talk events is to train a neural network with an unsupervised method and provide data which contain silence or push-to-talk event.

# Chapter 7

# Conclusion

The main goal of this thesis was to implement and evaluate models that are suitable for voice activity and push-to-talk detection. To properly train the neural network, it is necessary to provide an accurate and robust dataset. The ATCO2 dataset contained four hours of air traffic communication and was suitable for this task. The recordings consisted of 65% speech segments and 35% non-speech segments.

Converting audio files into Mel spectrograms is a common approach for training neural networks. Supervised training requires proper and correct labels. All the labels were frame-level. For experimental models, voice activity detection labels were extracted from files in the ATCO2 dataset archive. The labels for GPVAD training were obtained by prediction from their *sre* model. Push-to-talk events were not labelled, so an algorithm needed to be invented. The best approach lies in localising the local extrema and then selecting only those from which the signal has at least 100 ms of negative amplitude and is the only peak in the closest frame.

Processed data was used to train the neural networks. The two experimental architectures were a basic feed-forward neural network and a simple convolutional neural network. To get familiar with neural networks, the feed-forward model was the stepping stone, and various experiments were tried. This model was trained on push-to-talk and voice activity detection separately as well as in combination. The accuracies of this model were 77.2% for voice activity detection, 96.6% for push-to-talk detection and 83% for both push-to-talk and voice activity detection. The CNN model was trained only for multitask predictions and resulted in 90% accuracy.

The final solution was created through the adaptation of the GPVAD implementation. The GPVAD approach has accurate results for voice activity detection and was a suitable choice for this task.

The architecture of GPVAD was used for training on push-to-talk detection separately. With loading the *sre* model as starting point, the accuracy of this model is 83% and can provide sufficient predictions for push-to-talk events. The final multitasking neural network was trained and the model *sre* was used as starting point as well. An additional output layer was added for clip and frame level prediction. This resulted in 66% accuracy for push-to-talk detection.

In summary, the proposed and experimental models are capable of making the process of segmentation and data preparation more effective. The models can accurately detect push-to-talk events.

# Bibliography


[1] Abdel Hamid, O., Mohamed, A.-r. and Jiang, H. e. a. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 2014, vol. 22, no. 10, p. 1533–1545. DOI: 10.1109/TASLP.2014.2339736.

[2] Abdul, Z. K. and Al Talabani, A. K. Mel Frequency Cepstral Coefficient and its Applications: A Review. *IEEE Access*. 2022, vol. 10, p. 122136–122158. DOI: 10.1109/ACCESS.2022.3223444.

[3] Aggarwal, C. C. *Neural Networks and Deep Learning: A Textbook*. Springer, 2018. ISBN 978-3-319-94462-3. Available at: https://www.bibsonomy.org/bibtex/26681a2e990d3d35887d519e2ec3bd7a9/sdo.

[4] Benzeghiba, M., De Mori, R. and al., O. D. et. Automatic speech recognition and speech variability: A review. *Speech Communication*. 2007, vol. 49, no. 10, p. 763–786. DOI: https://doi.org/10.1016/j.specom.2007.02.006. ISSN 0167-6393. Intrinsic Speech Variations. Available at: https://www.sciencedirect.com/science/article/pii/S0167639307000404.

[5] Billauer, E. *Peakdet: Peak detection using MATLAB (non-derivative local extremum, maximum, minimum)* [http://billauer.co.il/blog/2009/01/peakdet-matlab-octave/]. 2009.

[6] Cho, K., Merrienboer, B. van and al., Ç. G. et. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: Moschitti, A., Pang, B. and Daelemans, W., ed. ACL, 2014, p. 1724–1734. DOI: 10.3115/v1/d14-1179. Available at: https://doi.org/10.3115/v1/d14-1179.

[7] Civil Aviation Authority. *Guide to Aviation* [https://www.caa.co.uk/consumers/guide-to-aviation/]. N.d.

[8] Crawshaw, M. Multi-Task Learning with Deep Neural Networks: A Survey. *CoRR*. 2020, abs/2009.09796. Available at: https://arxiv.org/abs/2009.09796.

[9] Dinkel, H., Chen, Y., Wu, M. and Yu, K. Voice Activity Detection in the Wild via Weakly Supervised Sound Event Detection. In: *Proc. Interspeech 2020*. 2020, p. 3665–3669. DOI: 10.21437/Interspeech.2020-0995. Available at: http://dx.doi.org/10.21437/Interspeech.2020-0995.

[10] Dinkel, H., Wang, S., Xu, X. and al. et. Voice Activity Detection in the Wild: A Data-Driven Approach Using Teacher-Student Training. *IEEE/ACM Transactions*

*on Audio, Speech, and Language Processing*. 2021, vol. 29, p. 1542–1555. DOI: 10.1109/TASLP.2021.3073596. ISSN 2329-9290. Available at: https://ieeexplore.ieee.org/document/9405474/.

[11] EUROCONTROL. *Air Traffic Management*. Eurocontrol, 2017. Available at: https://skybrary.aero/bookshelf/books/2497.pdf.

[12] FAWCETT, T. An introduction to ROC analysis. *Pattern Recognition Letters*. 2006, vol. 27, no. 8, p. 861–874. DOI: https://doi.org/10.1016/j.patrec.2005.10.010. ISSN 0167-8655. ROC Analysis in Pattern Recognition. Available at: https://www.sciencedirect.com/science/article/pii/S016786550500303X.

[13] FEDERAL AVIATION ADMINISTRATION. *Order JO 7110.65Y: Air Traffic Control*. Change 2th ed. Washington, D.C.: U.S. Department of Transportation, February 2018. Available at: https://www.faa.gov/documentLibrary/media/Order/ATC.pdf.

[14] GOODFELLOW, I., BENGIO, Y. and COURVILLE, A. Multi-Layer Perceptrons. In: *Deep Learning*. MIT Press, 2016, chap. 6. Http://www.deeplearningbook.org.

[15] HAJAJ, Y. *A Quick Introduction to Machine Learning for Beginners* [https://www.baeldung.com/cs/machine-learning-intro]. 2021. [Online; accessed 15-April-2023].

[16] HAYKIN, S. S. Feedforward Neural Networks: An Introduction. In: *Proceedings of the International Joint Conference on Neural Networks*. 2004. Available at: https://catalogimages.wiley.com/images/db/pdf/0471349119.01.pdf.

[17] HECKBERT, P. *Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm*. Notes 3, Computer Graphics 2, 15-463. Pittsburgh, PA: School of Computer Science, Carnegie Mellon University, February 1995. Revised 27 Jan. 1998.

[18] LEITNER, B. Z. J. and THORNTON, S. Audio Recognition using Mel Spectrograms and Convolution Neural Networks. In:. 2019.

[19] OLAH, C. *Understanding LSTM Networks* [https://colah.github.io/posts/2015-08-Understanding-LSTMs/]. 2015. Accessed: April 14, 2023.

[20] O'SHAUGHNESSY, D. *Speech Communication: Human and Machine*. Addison-Wesley Publishing Company, 1987. Addison-Wesley series in electrical engineering. ISBN 9780201165203. Available at: https://books.google.cz/books?id=mHFQAAAAMAAJ.

[21] RAMIREZ, J., GORRIZ, J. M. and SEGURA, J. C. Voice Activity Detection. Fundamentals and Speech Recognition System Robustness. In: GRIMM, M. and KROSCHEL, K., ed. *Robust Speech*. Rijeka: IntechOpen, 2007, chap. 1. DOI: 10.5772/4740. Available at: https://doi.org/10.5772/4740.

[22] SALAMON, J. and BELLO, J. P. Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *CoRR*. 2016, abs/1608.04363. Available at: http://arxiv.org/abs/1608.04363.

[23] Seo, S., Kim, C. and Kim, J.-H. Convolutional Neural Networks Using Log Mel-Spectrogram Separation for Audio Event Classification with Unknown Devices. *Journal of Web Engineering*. Jan. 2022, vol. 21, no. 02, p. 497–522. DOI: 10.13052/jwe1540-9589.21216. Available at: https://journals.riverpublishers.com/index.php/JWE/article/view/6889.

[24] Shi, X., Chen, Z. and Wang, H. e. a. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: *Advances in neural information processing systems*. 2015, p. 802–810. Available at: https://proceedings.neurips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

[25] Swets, J. A. Measuring the Accuracy of Diagnostic Systems. *Science*. American Association for the Advancement of Science. 1988, vol. 240, no. 4857, p. 1285–1293. Available at: http://www.jstor.org/stable/1701052.

[26] Swingler, K. *Applying Neural Networks: A Practical Guide*. Morgan Kaufmann, 1996. ISBN 978-0126791709.

[27] Tallman, J. W. What Am I Push-to-Talk Switch? *Flight Training Magazine*. January 2021. Available at: https://www.aopa.org/news-and-media/all-news/2021/january/flight-training-magazine/what-am-i-push-to-talk-switch.

[28] Tanyer, S. G. and Ozer, H. Voice activity detection in nonstationary noise. *IEEE Transactions on Speech and Audio Processing*. IEEE. 2000, vol. 8, no. 4. DOI: 10.1109/89.848229.

[29] University, A. *Voice Activity Detection* [https://speechprocessingbook.aalto.fi/Modelling/Neural_networks.html]. Accessed: 12 April 2023.

[30] University, A. *Voice Activity Detection* [https://speechprocessingbook.aalto.fi/Recognition/Voice_activity_detection.html]. Accessed: 12 April 2023.

[31] V, A. V. Voice Activity Detection Techniques – A Review. *I-Manager's Journal on Digital Signal Processing*. Jul 2021, vol. 9, no. 2. Available at: https://www.proquest.com/scholarly-journals/voice-activity-detection-techniques-review/docview/2644805165/se-2.

[32] Xu, X., Chen, Z. and al., D. X. et. *Mixing Signals: Data Augmentation Approach for Deep Learning Based Modulation Recognition*. 2022.

[33] Zeytinoglu, M., Erdogan, H. and Goksel, C. Voice activity detection using energy-based features. *IEEE Signal Processing Letters*. IEEE. 2008, vol. 15, no. 4, p. 213–216.

# Appendix A

# Contents of the included storage media

- `src/` Folder includes all the source filled as well as GPVAD implementation modules.
- `saved_models/` Folder with trained models of each experiment.
- `dataset/` Folder with push-to-talk dataset and some audiofiles from ATCO2 dataset.
- `README.md` Manual for the source code.
- `requirements.txt` List of Python libraries dependencies
- `poster.png` Poster image.
- `thesis.pdf` Thesis report file.
- `latex/` Folder with LaTeXsource files.