Tampere University

Laura Sipiä

MOBILE PHONE SPEAKER AUDIO QUALITY CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Prof. Tuomas Virtanen, Assoc. Prof. Davide Taibi
May 2021

ABSTRACT

Laura Sipiä: Mobile Phone Speaker Audio Quality Classification with Convolutional Neural Networks
Master of Science Thesis
Tampere University
Master's Programme in Information Technology
May 2021

There are many processes that have been automated which previously would have been done by a human. The process of quality evaluation is one of those processes. Machine learning is one of the most trending techniques of current time. Solutions using machine learning are used in various application areas e.g. object detection and image classification. Research and development has been done regarding quality evaluation using machine learning based solutions. Despite the huge interest in machine learning, there are very few existing solutions for audio quality classification with machine learning so far.

This thesis examines the possibility of using machine learning based solution for audio quality classification of mobile phone speakers. The thesis is executed as a comparison study between three models that have different design principles. The overall performance of each model is evaluated using 5-fold cross-validation. The generalization ability of each model is evaluated with leave-one-group-out cross-validation. The models are compared to each other based on the training metrics (loss and accuracy), time spent on training and the averaged performance in the two forementioned cross-validation methods.

All three classifiers produced good results, exceeding expectations. Two of the models in the comparison study attained over 0.98 AUC value and the third one attained over 0.95 AUC value. The generalization ability was measured by comparing the performance metrics, F2-score and AUC value of the 5-fold cross-validation and the leave-one-group-out cross-validation. With the best performing model, the difference in the results were less than 0.005 for the F2-score and less than 0.001 for the AUC value. This means that regardless of the phone model, the classification model could classify audio samples with nearly equal performance. The results gained from this research show that machine learning based model can be used for mobile phone speaker audio quality classification.

Keywords: audio quality, audio quality evaluation, audio classification, machine learning, classification

TIIVISTELMÄ

Laura Sipiä: Matkapuhelimen kaiuttimen äänenlaadun luokittelu konvoluutioneuroverkoilla
Diplomityö
Tampereen yliopisto
Tietotekniikan DI-ohjelma
Toukokuu 2021

Automatisaatio on käytössä monissa prosesseissa, joiden toteuttamiseen vaadittiin ennen ihmisiä. Laadun evaluointi on yksi tämänkaltainen prosessi. Koneoppiminen on yksi nykyhetken kukoistavimmista tekniikoista. Sitä hyödyntäviä ratkaisuja on käytössä monella alalla, kuten hahmon tunnistuksessa ja kuvan luokittelussa. Tutkimus ja kehitystyötä on tehty koneoppimisen hyödyntämisestä myös laadun evaluoinnissa. Huolimatta koneoppimisen suuresta suosiosta, koneoppimisen hyödyntämisestä äänenlaadun luokittelussa on kuitenkin olemassa hyvin vähän ratkaisuja.

Tämä diplomityö tutkii mahdollisuutta hyödyntää koneoppimispohjaista ratkaisua matkapuhelimen kaiuttimen äänenlaadun luokittelussa. Diplomityö on toteutettu kolmen eri suunnitteluperiaatteen omaavan mallin vertailututkimuksena. Jokaisen mallin kokonaisvaltainen suorituskyky evaluoitiin 5-kertaisella ristiinvalidoinnilla ja kyky yleistyä validoitiin leave-one-group-out-ristiinvalidoinnilla. Malleja vertailtiin toisiinsa opetusvaiheen metriikoiden, opetukseen kuluneen ajan sekä kahden esitetyn ristiinvalidoinnin suorituskyvyn perusteella.

Kaikki kolme luokittelijaa tuottivat hyviä tuloksia, ylittäen ennakko-odotukset. Jokainen malli saavutti yli 0.95 AUC-arvon ja kaksi vertailututkimukseen osallistunutta mallia saavuttivat jopa yli 0.98 AUC-arvon. Kyky yleistyä mitattiin vertailemalla 5-kertaisen ristiinvalidoitia ja leave-one-group-out -ristiinvalidointia käyttäen F2-tulosta ja AUC-arvoa. Parhaiten suoriutuvalla mallilla tuloksien erot olivat vähemmän kuin 0.005 F2-tuloksessa ja väemmän kuin 0.001 AUC-arvossa. Tämä tarkoittaa, että puhelimen mallista riippumatta luokittelija pystyy luokittelemaan ääninäytteen lähes yhtä hyvällä suorituskyvyllä. Saadut tulokset osoittavat, että koneoppimispohjaista mallia pystytään käyttämään matkapuhelimen kaiuttimen äänenlaadun luokittelussa.

Avainsanat: äänenlaatu, äänenlaadun evaluointi, äänen luokittelu, koneoppiminen, luokittelu

PREFACE

Thanks to my supervisors Tuomas Virtanen and Davide Taibi.

I would like to thank OptoFidelity for giving me the possibility of doing the thesis of this interesting topic. I want to thank Seppo Kuusisto for guidance and my colleges for all the support they gave me during the thesis process.

I want to say special thanks to my partner for cheering and believing in me throughout the process. Finally, thanks to my dog Java who made sure I wasn't in the world of neural networks for too long periods of time.

Tampereella, 3rd May 2021

Laura Sipiä

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

1D	1 Dimensional
2D	2 Dimensional
ANN	Artificial Neural Network
ASC	Acoustic Scene Classification
AUC	Area Under the (ROC) Curve
CNN	Convolutional Neural Network
DUT	Device Under Test
fn	false negative
fp	false positive
HLF	High Level Features
LDA	Linear Discriminant Analysis
LLF	Low Level Features
MFCC	Mel Frequency Cepstral Coefficients
MLP	MultiLayer Perceptron
MSE	Mean-Square Error
NN	Neural Network
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Oversampling TEchnique
STFT	Short-Time Fourier Transform
SVM	Support Vector Machines
tn	true negative
tp	true positive

1. INTRODUCTION

1.1 Background

We are living the time where information technology reaches to every area of life. For many decades mechanisms have been implemented to automate processes that have previously required human input. As computers have become faster and are able to process numerous times more data than a few decades ago, the amount of machine vision and learning algorithms has exploded and become available for everyone. Due to this change, new areas and processes can now be automated.

There are many industries that have automated quality inspection processes that have previously been done by a human. Using machine vision for visual quality grading and analysis has been a popular research area in many areas e.g. food industry [1][2], manufacturing [3][4], and touch panels such as mobile phone displays [5].

Audio quality evaluation algorithms have existed for a long time [6]. The popularity of machine learning has also reached the domain of audio quality evaluation. Using machine learning for audio quality evaluation is still fairly little researched domain, but there are studies that have successfully shown that machine learning models can be used for audio quality evaluation [7][8]. Because everyone has a recorder (mobile phone) at hand at all times, recording sounds is very easy. Therefore, the audio quality of recordings have aroused interest for many researchers [7][9].

1.2 Objective of the Thesis

The objective of the thesis is to implement a model that can classify audio samples into good quality and bad quality samples regardless of the phone model used. This thesis is done as a research for audio test of an automated mobile phone grading machine. There is an existing implementation for audio quality testing in another product meant for mobile phone functional testing that uses audio processing methods in the evaluation. The result from the audio quality test is based on detecting known defects from the signal. The ability of machine learning to pick up patterns from the signal that are not obvious or known is a tempting possibility for quality evaluation. The research done in the thesis intends to find out whether machine learning based model can detect the already known

defects from the data. The benchmarking results are gathered with the mobile phone functionality testing system. The results received from the testing system are used as the ground truth for the data used in the study.

In this thesis, we create three deep learning based models that are taught to classify audio samples to either pass (good quality sample) or fail (bad quality). The type of the defect is not considered but all the samples that have one or multiple defects are considered to belong into the same class. The three models implemented have different design principles. The objective is to find a model that reaches competitive results with the testing system.

1.3 Structure of the Thesis

The thesis is structured as follows. Section 2 takes a closer look at the popularly used audio quality evaluation techniques. Section 3 presents the history and trends of audio signal classification and the current state-of-the-art methods for audio classification with raw audio waveform. The section also gives a short introduction to general classification techniques, evaluation metrics and methods used in the field. The problem and popular solutions of imbalanced classification is presented as well. In section 4, the experiments implemented in the thesis are explained in detail. The section contains overall introduction to the comparison study, description of the dataset and the architectures of the three models implemented and the description of the comparison study process. Section 5 shows the results gained from the comparison of the three classifiers. Finally, in section 6 the research is concluded and future work is elaborated.

2. RELATED WORK

Audio quality evaluation solutions are implemented based on the phase of interest in the pipeline of recording the sound, passing it to the speaker and playing the sound from the speaker. For example, there are solutions that evaluate the quality of the sound event itself that the audio contains [10][11], the quality of recording process [7][8][11] and the quality of the speaker that plays the audio signal [12].

Nowadays recording sound events is easy and fast because the recording device is in the pocket at all times in the form of a mobile phone. Because the process of recording a sound is simplified into pressing a single button without any preparations required, the audio quality of the recordings can be really bad. This has sparked a lot of research to find solutions to detect the bad audio quality already in the recording phase and thus diminishing the amount of bad quality recordings [7][8]. Typically, the audio quality evaluation solutions in the context of recordings categorize the sound based on the sound context (e.g. traffic, birds singing) [7] or the expected origin of the bad quality (e.g. noise caused by wind) [8].

Typically in audio quality evaluation solutions the expected defects in the audio signal are the key factor in the evaluation process [8][11][12]. Examples of the typical defects in music audio quality evaluation are gaps in the sound, extensive silence, noise bursts and humming tones [11]. The defects can be identified with digital signal processing algorithms [11][12] or using the combination of audio signal feature extraction and machine learning based solutions [8].

There are also solutions that classify the audio signal to a quality level instead of detecting defects from the signal [7]. The solutions that utilize machine learning in evaluation of audio quality, the ground truth (the level of quality) for the audio signal is typically collected from the subjective feedback from a group of observers of the sound [7]. Having extensive annotated dataset is very important for comparative research to exist in machine learning based solutions. Currently, there are not publicly available datasets for audio quality classification. Even though there is some research and implementation done in the field of audio quality classification with machine learning, it is still fairly little researched area.

3. AUDIO CLASSIFICATION

Audio classification has been an active research topic for many decades. This section first presents two widely applied classification methods, neural networks and its subclass convolutional neural networks, that are bases for many classification solutions today. After presenting the foundation for neural network based classification methods, a look is taken at the history and trends of audio classification solutions. Finally, three different classification methods using raw audio wavefrom as input are described in detail.

3.1 General Classification Techniques

3.1.1 Classification

Classification is a procedure where a set of data is categorized into groups based on the patterns inside the data samples. The groups contain data that have patterns similar to each other and dissimilar to patterns in data samples belonging to other groups. The model that has the duty to group the set of data is called a classifier. In machine learning, classifiers can be taught with a set of data that has predefined labels that define the class/group/category the data sample belongs to. The classifier learns the patterns inside the data samples that define the classes. After the training process, classifier can then classify unseen data, that has no label yet, into appropriate class based on the learned patterns. This process is called supervised learning. There is also the possibility that the classifier has no a priori knowledge of the patterns that define the classes, but has to separate the data samples from each other on the go. This on the other hand is called unsupervised learning. [13] This thesis focuses on supervised learning classification techniques.

There are many classification algorithms developed for machine learning. Supervised classification algorithms can be coarsely divided into linear classifiers [14] and non-linear classifiers [15]. The basic idea for all linear classifiers is the same; they all try to fit a straight line in between the groups so that they are optimally separated. Visualization of linear classification is presented in figure 3.1. The method for optimally fitting the line between the groups is different for each linear classifier. Some well-known linear classifiers are linear discriminant analysis (LDA) [16], support vector machines (SVM)

[17] and linear regression [18].

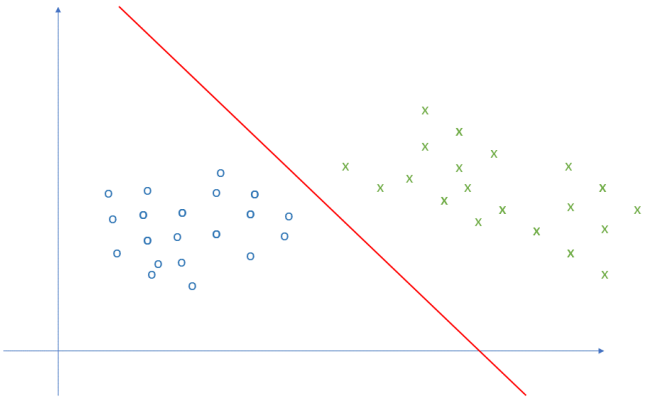


Figure 3.1. *Linearly separable classes*

The advantage of the non-linear classifiers is that the classes, that are not linearly separable, can be classified. Visualization of non-linearly separable classes is presented in the figure 3.2. Due to this advantage, the non-linear classifiers are very popular. Some well-known non-linear classifiers are decision trees [19] and neural networks (NN).

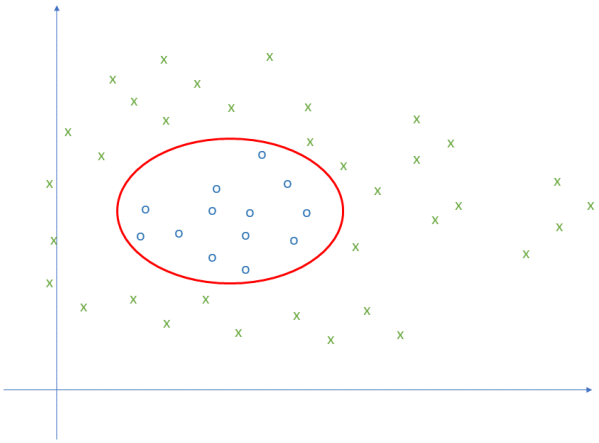


Figure 3.2. *Non-linearly separable classes*

The popularity of Neural Networks based classifiers has exploded in the 21st century. The following two subsections take a closer look at Neural Networks and it's branch Convolutional Neural Networks.

3.1.2 Neural Networks

Artificial neural networks (ANN), or popularly referred as neural networks (NN), are computational processing systems that are inspired by the human nervous system. Roughly

put, the biological nervous system is a network of small processing units called neurons that are linked to each other by synapses. The network of human nervous system can receive information of the current surroundings, process it and decide an action based on previous experience. This process of determining an action (output) based on received information (input) applying the previous experience (learned model) is the base idea behind the artificial neural networks. [20]

Artificial neural network consists of small processing units called perceptrons (or neurons) that take in signal, process it and produce an output. These individual perceptrons are interconnected to each other and together they can learn from the input in order to optimize the final output. [21]

The theory behind ANN is in linear combination. Perceptron takes in a data sample that is a vector of input values $x_1, x_2, ..., x_n$. Each input value x_i is multiplied with a weight value w_i. All the input values multiplied with the respective weight values are then summed together in the perceptron forming a linear combination formula for the output $u = \sum_{i=1}^{n} w_i x_i$.

This linear combination produces an output that is a representation of the given input value affected by the weight values. A constant bias value b is added to the linear combination formula to add flexibility to the model. Without bias, the line formed by the linear combination would always go through origin. However, with bias, fitting the model to the given data is more flexible and can reflect the real data better.

Representing a dataset with a linear combination is not optimal. Non-linearity is introduced to the linear model to improve the performance of the model. Before forming a final output from the perceptron, the summation of the input values multiplied with respective weights and the added bias is passed through a non-linear activation function φ to obtain output y of the non-linear function as $y = \varphi(u + b)$.

Perceptron model, as it is usually visualized, is presented in the figure 3.3. In the figure, $x_1, x_2, ..., x_n$ are the input values, $w_1, w_2, ..., w_n$ are the weight values for each input value, b is the bias and y is the output value.

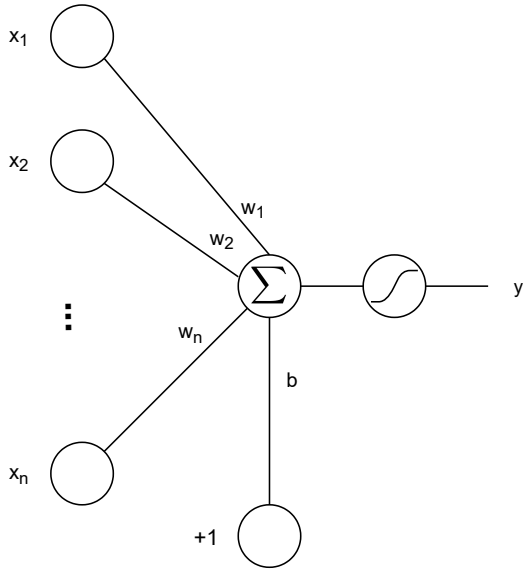


Figure 3.3. *Perceptron model*

The individual perceptrons form layers that are connected to neurons on the subsequent layer forming a network of perceptrons. A structure that has multiple layers of neurons is called multilayer perceptron (MLP). The structure of a multilayer perceptron is visualized in the figure 3.4.

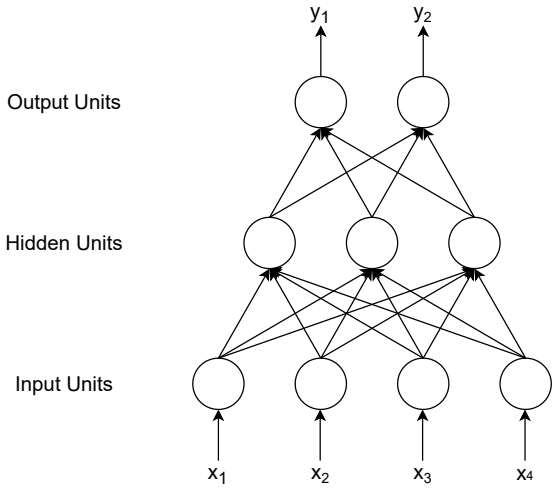


Figure 3.4. *Feed forward network*

The idea in MLP structures is that the data flows from bottom layers to top layers. The first layer is called the input layer. The input layer consists of as many neurons as there are elements in the input array $X = x_1, x_2, x_3, ..., x_n$. Each neuron is connected to another neuron in the following layer. The outputs of the lower layers are the inputs of the upper layers tuning the linear combination formula into $y_k = \varphi(\sum_{i \varepsilon L} w_{ik} y_i + b_k)$, where the output of a neuron k is calculated with the outputs y_i from the previous layer L by multiplying each output y_i with the weight value of the connection w_{ik} between the two

neurons k and i and adding the bias b_k of the neuron. All the layers that are between the input layer and the last layer of the network are called hidden layers. The last layer of the network produces the output $Y = y_1, ..., y_n$ from the network and is therefore called an output layer. Networks that follow this kind of structure, that pass output values from input layer to output layer without cyclic connections, are called feed forward networks.

The structure and connections in the network do not produce an optimal representation of the input as is. The network has to be taught to represent the given data first. This phase is called training the network. The training happens by tuning the weight vectors of each layer by gradient based back propagation. Networks that have more than one hidden layer are considered as deep neural networks. Deep learning models have made many present classification tasks possible that have not been able be solved before [22].

The idea behind training a neural network is to find the weight values that minimize the difference between the desired outcome t and the outcome produced by the network y. The error between the two outcomes is calculated using a loss function. Loss function (E) is a measure that implies the difference between the "correct" outcome and the network output. [23] There are many loss functions available but one of the simplest and used is mean-square error (MSE) that is defined as $MSE = \frac{1}{N}\sum_{i=0}^{N}(t_i - y_i)^2$, where N is the number of predicted values, t_i is the truth value and y_i is the predicted value.

During the training process weight values are updated based on the calculated loss. Gradient descent based training has been used since the 1980s [24]. In gradient descent, weight values W are updated based on the gradient of the loss function E. A simple gradient descent algorithm is $W_k = W_{k-1} - \xi\frac{\partial E(W)}{\partial W}$. [23] Gradient of loss function gives a value of the direction where the loss grows the sharpest. As seen from the equation of the simple gradient descent algorithm, the weights are updated with the negative of the gradient $\frac{\partial E(W)}{\partial W}$. This adjusts the weights to the direction where the loss function decreases the sharpest (the opposite direction of the gradient). Adjusting weights recursively based on the gradient descent algorithm, it allows the weights to slowly reach the optimal weight values that minimize the loss function. This is the basis of training neural networks and it has been that since the 1980s. [24]

The gradient descent algorithm has been used in various applications for decades and during that time, many variations of it have been applied to neural networks. The algorithms that utilize the loss function to update the weights are today called optimizers.

Backpropagation algorithm enables computation of gradients of stacked layers from the output layer of the network, all the way to the input layer. The idea of backpropagation algorithm was proposed by Rumelhart et al. [24] and has since been by far the most used learning algorithm in neural networks. The idea of the backpropagation algorithm is that the gradients for all the network layers can be calculated from the derivative of the cost function $\frac{\partial E}{\partial y}$ with respect to the output. Taking a derivative from the output with respect to

the input values using the equation 3.1 the error derivative of the cost function is received with respect to the input $\frac{\partial E}{\partial u}$.

$$\frac{\partial E}{\partial u_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{u_k} \tag{3.1}$$

As mentioned before, the top layers are connected to the previous layer. This leads to that the error derivative for any hidden layer k with respect to the output of the layer can be calculated as $\frac{\partial E}{\partial y_k} = \sum_{l \varepsilon U} w_{kl} \frac{\partial E}{\partial u_l}$, where U are the units in the upper layer.

From there applying equation 3.1, the error derivative is gained with respect to the inputs of the layer. Chaining these two equations, we can derive the gradients for the weights for each layer. [25]

3.1.3 Convolutional Neural Networks

Convolutional neural networks (CNN) have been used for pattern recognition since the 1980s. CNN models have been used for image classification and object recognition since the technique was invented. [23][26][27] During the last decade, CNN popularity has increased significantly and the application areas of CNN have diversified to e.g. music and speech analysis [28]. Convolutional neural network classifier takes in a raw input signal or hand-crafted features calculated from the signal, passes the data through several stacked layers extracting features from the input and finally passes the extracted features to fully connected layers that produce the final classification (see figure 3.5). The stacked layers consist of convolution layers, pooling layers and non-linearity activation layers. The activation layers are not usually drawn to CNN architecture images.

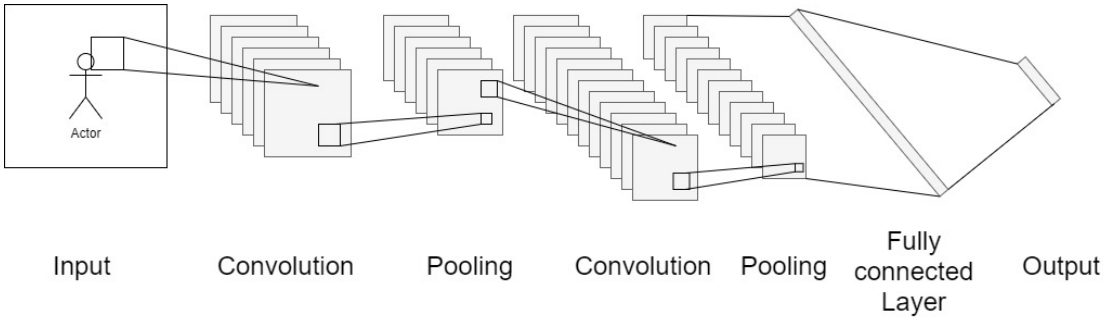


Figure 3.5. *Typical Convolutional Neural Network (CNN) structure*

Convolutional neural network has the ability to extract local, spatial and temporal features. This is accomplished by stacking convolution and pooling layers one after another. Convolution layers produce feature maps/filters of the input data by sliding a convolution window (called kernel) over the input data with defined steps. In each step, convolution operation is applied to that part of the input data that the kernel is presently on. Convolution kernels contain weights that are used for applying the convolution. For each step,

a single value is produced to a feature map and by sliding the kernel over the input data feature maps are produced. The convolution operation is visualized in the figure 3.6. Each feature map represents some feature of the input data and for each feature map a different convolution kernel is used. Typically, convolution layers produce 10-500 feature maps depending on the input data and application area.

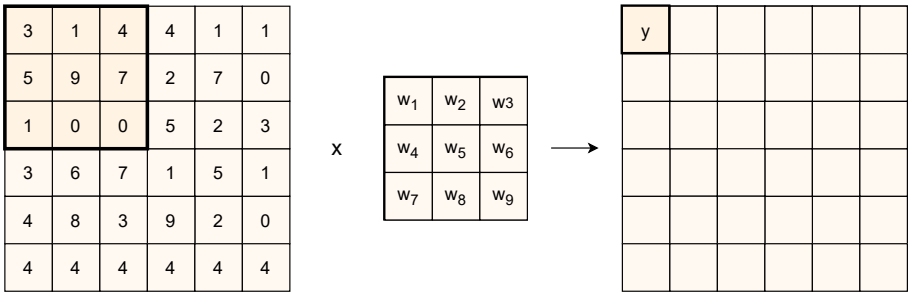


Figure 3.6. *Convolution operation visualized with 3x3 kernel*

Rectified linear unit (ReLU) is a very popular non-linearity function used in CNN models regardless the classification task [29][30][31]. One of the reasons why ReLU activation is so popular is because deep CNNs with ReLUs train faster than its competitors [30]. ReLU activation function is calculated as $ReLU = max(0, u)$, where u is the output before activation.

Pooling layers sub-sample the feature maps spatially and/or temporally. The sub-sampling is done by averaging or taking the maximum value of a set of samples in a feature map, depending on the pooling technique used, and using the gained value as a representation of that set of values in the pooled feature map. Pooling is visualized in figure 3.7.

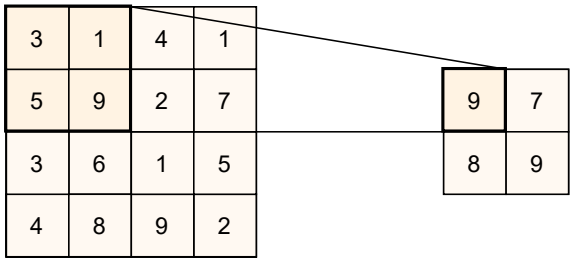


Figure 3.7. *Max pooling visualized with pool size of 2x2*

Due to the pooling, multiple layers on features can be extracted from local connections to more abstract and high level features. The first feature maps are gained by applying the convolution to local areas, generating feature maps of local connections in the image. After applying pooling, the features extracted from the data are more abstract and contain larger features in the original data. This ability to represent the given data in multiple levels has enabled the successes in image classification, object detection and many more application areas that were not feasible before [25].

Overfitting is a common issue while training a neural network. It means that the network "memorizes" the training data instead of learning the desired patterns in it. If a model overfits to the training data, it will not be able to perform well with unseen data. To tackle this problem, a layer called dropout is used. As the name indicates, the layer drops out units in a neural network. This means that units, including their incoming and outgoing connections, are temporarily removed. The units that are removed are chosen randomly and the number of removed units is given as a parameter to the layer. The parameter is given as a probability of retaining a unit in the network. A common parameter for dropout is 0.5 which indicates that each unit in the network is retained with the probability of 0.5. [32]

Methods applied to prevent overfitting are called generalization methods. Using dropout is one of the methods used. Another method is called batch normalization. This technique is based on a normalization step applied to layer inputs. Batch normalization is also used because it enables networks to converge faster. [33]

Convolutional neural networks have gained huge popularity with classification tasks using 2-dimensional data, such as images, but has since been adopted to 1-dimensional data classification tasks as well. [22][34]

3.2 History and Trends of Audio Classification

Audio classification tasks can be coarsely divided into three sub-categories: music signal classification, speech signal classification and environmental sounds classification [35][36]. Each sub-domain can be further divided to various application areas.

In the field of speech signal classification, speech recognition and speech enhancement are examples of heavily researched application areas. Speech signal classification can be for example used for detecting the emotion of the speaker [37]. Speech signal classification has also been applied to health industry. There are many techniques that use speech enhancement in the hearing devices for people with hearing disability [36][38].

Many people have at some point heard a song that they do not know but would like to hear again and used the application Shazam [39] to find out what the song is. Shazam is an application that uses music retrieval algorithm to identify the song currently playing. There are many more existing application areas that use music signals for classification e.g. instrument classification [40] and music genre classification [41].

In the field of environmental sound classification, acoustic scene classification is one of the most popular research areas at the moment [42]. Acoustic scene classification (ASC) has many application areas e.g. robotics, hearing aids and context awareness in smart devices. As a research area ASC is not new, but the popularity of machine learning has boosted its popularity in the past decade. [42] It is no wonder that it has been proven that

machine learning models can perform with high accuracy in acoustic scene classification tasks [43].

The pipeline of audio signal classification generally consists of two parts: feature extraction and classification using the extracted features. In the feature extraction step, a selection of feature vectors is extracted from the audio signal. Features that are extracted from the signal vary based on the audio signal content. [44]

Feature extraction is an important step in the audio signal classification pipeline because the outcome of the classification depends of the features extracted from the signal [44]. Typically, audio signals are transformed to time-frequency representations to better capture patterns in intricate sound sources. Widely used time-frequency representations are e.g. spectrograms and mel-filterbanks. [34] One of the most popular feature vectors in audio signal classification is mel-frequency cepstral coefficients (MFCC) [34][44][45][46]. MFCCs are power spectrum representations of audio samples. MFCC use mel-scale frequency bands that mimic the human hearing scale. This makes MFCC a key feature in many audio signal classification tasks. [36] It is important to understand which features represent the audio signal content optimally, and not all features are well suited for every audio signal. Even though MFCC is one of the most used feature in feature extraction, it may not represent all signals optimally. As MFCC is scaled to mimic human auditory system, it may not be the best feature to represent an audio signal that is not meant for human ear.

There are many techniques used in the field for audio signal classification. One popular technique is feeding CNN with images of time-frequency representations of the audio signal e.g. short-time fourier transform (STFT) spectrograms or MFCC [47][48]. These models can utilize the state-of-the-art model for image classification. Also, solutions using 1D representations of the extracted features have been developed. Hussain Md. et al. introduced a method called SwishNet that uses 1D CNN and MFCC features for classifying music, noise and speech [45]. During the last decade there has been research and development in the area of feeding networks with raw audio signals instead of representations of them. The next section presents the state-of-the-art techniques in that field.

3.3 Audio Classification Methods Using Raw Audio Waveform

Sparked by the success in image classification tasks, CNN has been adopted also to the field of audio classification [49]. As described in the previous section, CNN has already been used in audio classification using hand-crafted features that have then been fed to 2D CNN. During the last years, CNN has also been used in audio classification using the raw audio waveform as the input. There are many application areas that have already adopted this new approach for example acoustic scene classification [22][29][50], speech recognition [51], speaker spoofing detection [52] and speech acoustic modeling [53].

In image classification tasks, CNN can be fed with raw image data instead of hand crafted features because CNN can extract features from the raw data that are invariant to spatial and temporal changes. The feature extraction is data-driven and therefore the features are optimized for that specific task [49]. Classifiers trained with raw audio waveform have been proven to reach equally good results in same tasks than classifiers using hand-crafted features [22][54]. There are also results that prove raw waveform based classifiers to outperform spectral feature based classifiers [51].

CNN architectures using raw audio waveform as input can be roughly divided into two main categories based on the CNN structure:

- (i) classifiers that are end-to-end 1D CNN [29][34] and
- (ii) classifiers that are a combination of 1D CNN and 2D CNN [49][52].

Frame-level end-to-end 1D CNN proposed by Abdoli S. et al. [29] is an example of a classifier of the first (i) category. The classifier was developed for environmental sound classification task. In the proposed architecture, raw waveform is fed into a 1D CNN and processed with stacked convolution and pooling layers visualized in the figure 3.8.

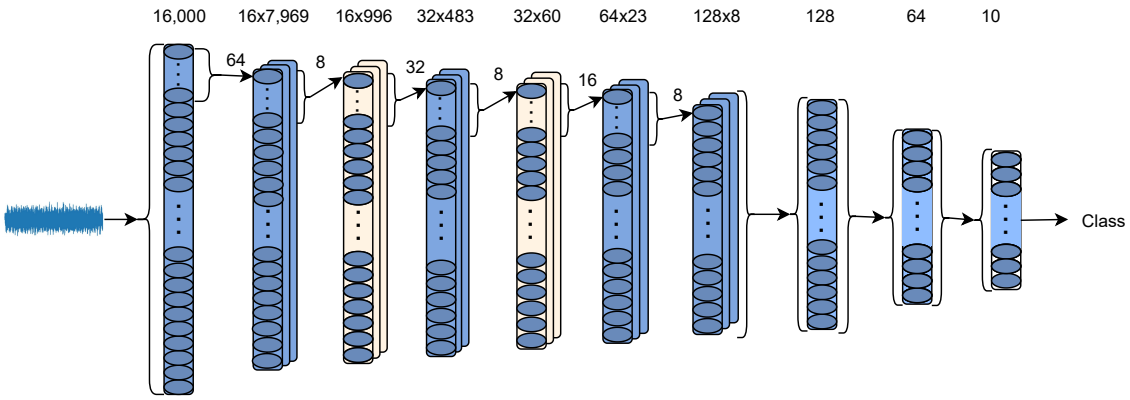


Figure 3.8. *Frame-level end-to-end 1D CNN reproduced based on [29]*

This architecture uses large filter sizes in the convolutional layers throughout the network, starting with filter size of 64 in the first convolutional layer and reducing the filter size by half in each subsequent layer. By using large filter sizes, the convolution operation is applied using "frames" of the signal instead of multiple individual samples. Due to this design principle, the architecture is thought as frame-level 1D CNN.

Another example of the first (i) category is SampleCNN by Lee J. et al. [31] which is a sample-level end-to-end 1D CNN architecture. SampleCNN architecture uses very small (2-5 samples) filter sizes in the convolutional layers which would correspond to pixel-level in image data. Lee J. et al. noted that using frame-level filter sizes in convolutional layers prevent from learning all the phase variations in the audio signal and therefore

sample-level filter sizes would be superior [31]. The idea difference between frame-level architecture and sample-level architecture is visualized in figure 3.9.

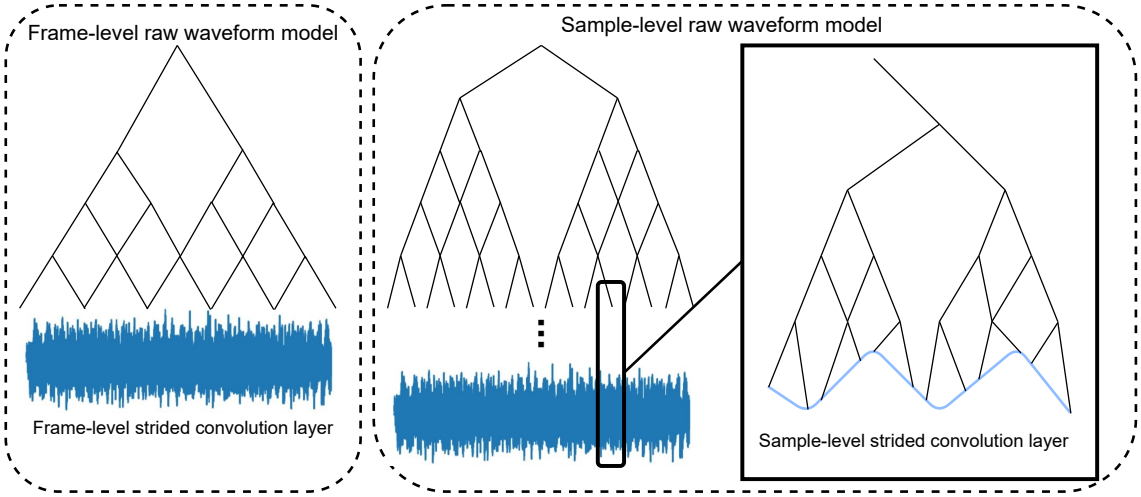


Figure 3.9. *Frame-level architecture vs. sample-level architecture. Reproduced based on [31]*

SampleCNN developed an architecture design called $m^n - DCNN$ model. This model enables using different sizes of input data using same architecture principle. In the proposed model, m refers to the filter size and pooling length and n refers to the depth of the network. m and n are directly related to the input size so that $m * m^n$ equals the input size. The network architecture is constructed so that the first layer is a convolutional layer with filter size of m, number of strides equal to m and number of feature maps is 128. All the following layers are constructed of blocks with convolutional layer (filter size of m and stride of 1) followed by pooling layer (pooling length of m and stride of m). The blocks differ only with the number of feature maps generated in the convolutional layers. This parameter is configurable by the network user. [31]

AclNet by Huang J. et al. [49] is an example of the second (ii) category of end-to-end CNN that use raw waveform as the input. This architecture consists of two blocks: low-level features (LLF) and high-level features (HLF). The network is constructed in a way where the input signal is first fed to the LLF block extracting the low-level features of the signal and then the extracted features are passed to the HLF block that then extracts the high-level features and does the classification. The LLF block consists of two subsequent convolutional layers followed by one max pooling layer. The outcome from the LLF block can be viewed as an imitation of the spectral feature, and FIR decimation filterbank due to the two stages of 1D strided convolutions. HLF takes in 2D image-like tensor. Before passing the output from LLF block, the result is transposed to image-like representation. HLF block then follows similar structure than CNNs for image classification. In AclNet, number of architectures were tried, but an architecture similar to VGG network [55] was found the best for the task at hand.

3.4 Imbalanced Classification

When the distribution of classes inside a dataset is not even, the classification task is categorized as imbalanced classification. Most common classification algorithms expect the class distribution to be balanced and therefore these algorithms perform poorly with imbalanced classification tasks. As the real-world domain dataset are proven to be often naturally imbalanced, this has become important and popularly researched issue. [56]

As mentioned, the imbalanced classification problem has been researched a lot during the last two decades and many solutions have been invented to overcome the problem. One popular solution for the problem is to even out the class distribution with re-sampling the data. Re-sampling can be accomplished with over-sampling, under-sampling and combination of the two previous sampling techniques. In case of over-sampling, the minority class samples are generated one way or another depending on the technique used, to even out the distribution. On the contrary with under-sampling, the majority class samples are rarefied to even out the distribution. Often these two techniques do not perform optimally by themselves but as a combination. There are many techniques available for re-sampling e.g. random oversampling and undersampling, synthetic sampling and cluster-based sampling. [56]

Synthetic minority oversampling technique (SMOTE) was proposed in 2002 by Chawla N. et al. [57] as an improvement for random oversampling and has since become one of the most popular oversampling techniques in the field [56][58]. The SMOTE algorithm generates synthetic samples from random minority class samples with the help of their k-nearest minority class neighbors and linear interpolation. The SMOTE algorithm has three steps:

1. A random sample is chosen from the minority class.
2. K-nearest minority class samples are searched and randomly one of them is chosen.
3. New sample is generated by taking the difference of the two samples and multiplying it with a random weight in between 0 and 1.

The generation process of one synthetic sample is visualized in the figure 3.10 with 4 nearest neighbors.

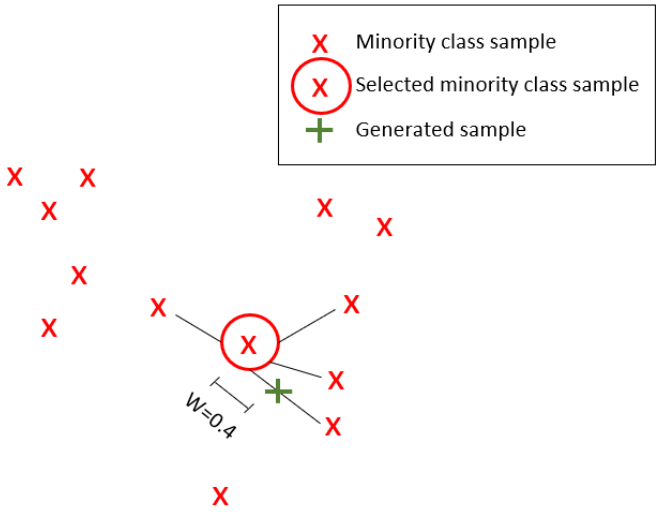


Figure 3.10. *SMOTE sample generation visualized. The visualization is reproduced based on [58]*

Even though the SMOTE algorithm is an improvement to random oversampling, it has some drawbacks as well. One of the drawbacks is the algorithm's way of emphasizing the imbalance within the minority class. This is caused by the way SMOTE randomly chooses, with uniform probability, a minority class sample to oversample. Regions that are sparse are therefore likely to left sparse and regions that are dense become denser. Another weakness of the SMOTE is the way it easily creates noise to dataset that already contains noisy samples. If a minority class sample is located among majority class samples and is selected for oversampling, the new generated sample will most likely also reside among the majority class samples. The more samples are generated among the majority class samples, the more likely they are selected as the sample used in oversampling. This process easily causes noise in the data. [58]

Despite the clear weaknesses that the SMOTE algorithm has, it is widely used by researchers and practitioners assumably due to its simplicity. To overcome the disadvantages of the SMOTE algorithm, numerous variations have been proposed e.g. Borderline-SMOTE [59], SMOTEBoost [60] and KMeansSMOTE [58].

3.5 Classification Evaluation

Evaluation metrics are used to evaluate the performance of classifiers. Evaluation metrics can be thought as measurement tools of classifiers performance. Each of the evaluation metrics describe the performance of the classifier in a different point of view by evaluating different characteristics of the classifier. All of the evaluation metrics give a single scalar value for the classifier performance. Comparing different classifiers is easy because the metrics are single values. [61]

This section presents the most popular and used evaluation metrics at the moment. The metrics presented are well suitable for binary classification problems. These metrics are used in section 4 to evaluate and compare the models in the comparison study.

3.5.1 Confusion Matrix

Confusion matrix is a fundamental performance metric that lays the ground for many popular performance metrics [62]. The idea of the confusion matrix is to represent the distribution of the predicted classes compared to the actual classes in a concise manner. Confusion matrix in generally presented with two-by-two matrix that represents the results of binary classification [61]. However, confusion matrix can be extended to evaluate the performance of multi-class classifiers as well.

Two-by-two confusion matrix is formed by having the actual classes as rows and the predicted classes as columns (or vice-versa) (see table 3.1).

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	True Positive (tp)	False Negative (fn)
Actual Negative Class	False Positive (fp)	True Negative (tn)

Table 3.1. *Confusion matrix*

The cells inside the matrix give an information about the number of true positive (tp), false positive (fp), false negative (fn) and true negative (tn) samples. A sample is considered as true positive when the predicted class is the same as the actual class. Negative samples that are predicted as positive are counted as false positive. Similarly, a negative sample belongs to true negatives if it is predicted as negative and positive sample that is predicted as negative belongs to the false negatives. [61] With this logic the number of correctly classified samples are located in the diagonal of the matrix.

3.5.2 Accuracy

Accuracy is the most used evaluation metric in any classification task. Accuracy measures and summarizes the quality of the classifier with one value. This makes accuracy a very popular metric used often in classifier comparison. [61] Accuracy is calculated using the true positive, true negative, false positive and false negative values of confusion matrix as $accuracy = \frac{tp+tn}{tp+fp+tn+fn}$, where tp is the number of true positives, tn number of true negatives, fp number of false positives and fn number of false negatives.

Even though accuracy summarizes the classifier quality truthfully in many cases, there are cases where accuracy does describe the classifier quality accurately. For example, in a binary classification problem where the test set has 90 positive samples and 10

negative samples and the classifier predicts all of the samples as positive. This makes the accuracy to be 90% which sounds like a good result, even though actually none of the negative class samples were predicted correctly. Therefore, accuracy is not the optimal performance metric to use when the test set is imbalanced.

3.5.3 Precision and recall

Precision and recall are popular performance metrics that are also bases for other performance metrics. Unlike accuracy, precision and recall produce reliable results also with imbalanced test data as they can be calculated per class. Precision measures the classification result correctness. Precision is calculated as $precision = \frac{tp}{tp+fp}$, where tp is the number of true positives and fp the number of false positives. [63] Intuitively, precision is a measure of correctly classified positive samples of all the samples predicted as positive.

Recall (or Sensitivity) on the other hand, measures the completeness of the classification results. Completeness (or accuracy) in this context means the fraction of actual positive samples that were predicted as positive out of all actual positive samples. Recall is calculated as $recall = \frac{tp}{tp+fn}$, where tp means the number of true positives and fn means the number of false negatives. Intuitively, recall means the portion of actual positives that were correctly classified.

A perfect classifier would not classify any negative samples as positive and none of the positive samples as negative. This means that the number of false positives would be 0 and therefore precision would be 1. Also, the number of false negatives would be 0 and therefore recall would also be 1. Precision and recall can be calculated per class highlighting the difference in performance between classes. [63]

3.5.4 F-score

F1-score (or F1-measure) is a harmonic mean of precision and recall. F1-score is calculated as $F1 - score = \frac{2\,x\,Recall\,xPrecision}{Recall\,x\,Precision}$. [63] This equation is a specific case of more general F-measure called β varied F-score that is calculated as $F_\beta - score = \frac{(1+\beta^2)\,x\,Recall\,x\,Precision}{\beta^2\,x\,Recall\,x\,Precision}$, where the value of β is the relative importance of precision compared to recall. It can be calculated as $\beta = \frac{C(FN)}{C(FP)}$, where C(FN) means cost of false negative (also thought as importance of recall) and C(FP) means cost of false positive (also thought as importance of precision) [63].

The β is 1 when the precision and recall are equally important. This is the case in F1-score. If β is more than one, cost of false negative predictions is greater than false positive, meaning that recall is more important than precision. Using β value more than one is generally used in cases where the distribution of dataset is not balanced and minority class samples predicted as majority class samples is costly. Respectively, β

less than one emphasizes the importance of precision. [63] F-score can be calculated separately for each class. This is especially useful when the dataset is imbalanced.

3.5.5 AUC-ROC

A receiver operating characteristics (ROC) graph is a method for visualizing and ranking classifiers based on their performance. ROC graphs are especially used in presenting the trade-off between success rate and false positive rate. Because of this quality, ROC graphs are especially useful with datasets with unbalanced class distribution and unequal classification false alarm costs. [62]

ROC graph is a two-dimensional graph that has true positive (tp) rate, calculated as $tp\ rate \approx \frac{Positives\ correctly\ classified}{Total\ positives}$, in the Y axis and false positive (fp) rate, calculated as $fp\ rate \approx \frac{Negatives\ incorrectly\ classified}{Total\ negatives}$, in the X axis. In case of a discrete classifier the classification result would only yield one point to the ROC space instead of a curve. However, in case of probabilistic classifiers such as neural networks, a curve presenting multiple tp and fp rates can be calculated. This is possible due to the fact that neural network classifier produces probabilities as output from the model and by thresholding the floating point value a classification decision is made. By using a set of classification results and changing the classification threshold value, multiple points of true positive and false positive rates can be calculated, forming a curve to the ROC space. [62]

ROC graphs depict curves that's shape describes the performance of the classifier. A ROC graph is visualized in the figure 3.11. If the curve is linear and goes diagonally across the graph as the dotted line in the presented figure, the classifier performance is as good as a random classifiers. Classifier is considered the better the more the ROC curve goes above the diagonal line. A perfect classifier would produce a ROC curve that goes from the origin straight up to point (0,1) and would curve straight from there to point (1,1) (dark green line in the presented graph).

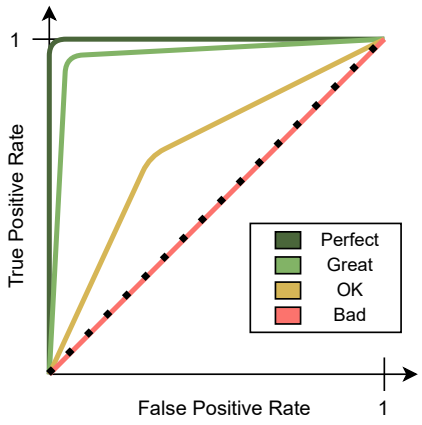


Figure 3.11. *ROC graph with color coding representing the performance of a classifier*

ROC graph is a visualization that cannot itself be used during classifier training, therefore performance metric called area under the (ROC) curve, or AUC, is used to represent the ROC curve instead. As the name of the performance metric describes, the metric measures the area under the ROC curve. The closer AUC value is to 1 the better the classifier performs. In the case of a random classifier, the AUC score is 0.5 which can be also visualized from the presented ROC figure. [62]

3.5.6 Cross-Validation

The previously presented classifier evaluation metrics have been techniques that are used for evaluating the classifier performance. These performance scores can then be used for comparing different classifiers given the same classification task. Before comparing classifiers with each other, the performance scores must be validated for each classifier.

The performance score that a classifier outputs with training data and testing data is rarely the same. Evaluating a classifier with training data gives too optimistic results as the classifier learns the training data too well. The wanted outcome of training a classifier is that it performs well with unseen data. Generally, data sets are split into training data and testing data. Training data is solely used in training the classifier and testing data is only used for evaluating the model. The testing data is not used during the training time and therefore is unseen data to the classifier. This approach leads to better understanding on the classifier's performance. The approach gives performance score for one unseen data set that may represent certain subset of data but may not give accurate estimate on the classifier's performance on the whole data set. To validate classifier performance cross-validation techniques such as k-fold cross-validation, monte carlo cross-validation and leave-one-group-out cross-validation are widely used.

K-Fold Cross-Validation

K-fold cross-validation is one of the most used classifier validation techniques. In the validation procedure, a data set is split into k disjoint approximately same sized folds which each in turn is used as the test set and all other folds are used for training the classifier. The final performance score of the classification algorithm is formed by averaging the k performance scores resulting from the k-fold cross-validation. [64]

Leave-One-Group-Out Cross-Validation

Leave-one-group-out cross-validation uses information about the data in forming the folds for cross-validation. Instead of randomly selecting k-folds from the data set, leave-one-group-out cross-validation technique uses pre-defined groups as folds. The validation procedure follows the same principle as with k-fold cross-validation. One group is sepa-

rated as the testing group and the model is trained with the rest of the groups. Validation is done for each group as the test group in turn. Dataset can be split into groups based on e.g. experiment rounds, devices that are used in recording the data or timestamp. [65]

4. METHODS

The research done in this thesis aims to find out whether machine learning based model can distinguish good audio quality from bad audio quality. The model should be able to classify the audio quality for phone models it is trained with, but also for phone models that were not used in the training data. The idea is that a new phone model could be evaluated with the model without the need for re-training the model.

The audio signals evaluated in the research are samples recorded from mobile phones' receiver speakers. The baseline results for the research are gained from an existing audio quality testing system. The existing system expects to find certain defects from the audio signal and uses signal processing methods to detect them. The expected defects are for example low volume, rattling and noise. The performance of the existing system is limited to the known defects which means that, in case audio signal contains some unknown defects, the existing system would not be able to catch it. The ability of machine learning to learn patterns from a dataset that are not obvious or even comprehensible by a human, makes it a tempting method for quality evaluation.

In this study three different CNN based models are compared. All of the models use raw audio waveform as the input and are end-to-end convolutional neural networks that output a binary result (pass or fail). The models differ from each other the way the audio features are extracted from the raw waveform in the convolutional neural networks. Three models were chosen as inspiration for the models implemented for this study from the presented models in section 3.3. The three chosen models are: a frame level 1D CNN by Abdoli S. et al. [29], a sample level 1D CNN by Lee J. et al. [34] and AclNet by Huang J. et al. [49]. Each model inspired by the three architectures mentioned are presented in detail in section 4.2.

The three models are evaluated with 5-fold cross-validation and using performance metrics accuracy, precision, recall, F2-score and AUC presented in detail in section 3.5. In addition to evaluating the models with 5-fold cross-validation, leave-one-group-out cross-validation is used to evaluate the generalization capability of models. The results received

from the leave-one-group-out cross-validation are evaluated using F2-score and AUC. Also, the training performance of the classifiers is evaluated. The comparison of the three models is based on the training performance, the performance in 5-fold cross-validation and in leave-one-group-out cross-validation.

4.1 Dataset

4.1.1 Data Gathering

The dataset used in the thesis experiment consists of audio samples from mobile phones speakers. The audio samples are recorded by the testing system. The testing system performs multiple tests on a mobile phone to access the quality of the phone. One of the test cases done on a device under test (DUT) is a speaker test. This test case analyses the sound quality of the mobile phone speakers. Mobile phones have two speakers: receiver speaker and loudspeaker. The test case has two tests for accessing the sound quality of both speakers separately. Therefore, each speaker test produces two audio samples, one recorded from the DUT's receiver speaker and the other from the DUT's loudspeaker. This thesis focuses only in receiver speaker.

During the speaker test, DUT plays a specific signal from the speaker under test. The signal is a 1 kHz sinusoidal signal that is played for two seconds. Therefore, each audio sample contains 96 000 floating point values with sampling frequency of 48 kHz. The dataset used for the experiment performed in this thesis contains 33 398 receiver speaker samples. The dataset contains audio samples recorded from 65 different mobile phone models. The distribution of audio samples per mobile phone model is presented in the figure 4.1.

The distribution of the audio samples among the 65 phone models is not even. There is one clear dominating phone model (phone model 0) that forms about 9% of the whole data. There are also 8 phone models (phone models 57-64) that only have 1 audio sample per phone model. Most of the phones have 1% - 5% share of the data. If the dataset would only contain a few phone models, it would affect the generality of the model. The model could learn patterns unique to those phone models instead of the patterns that define the audio quality.

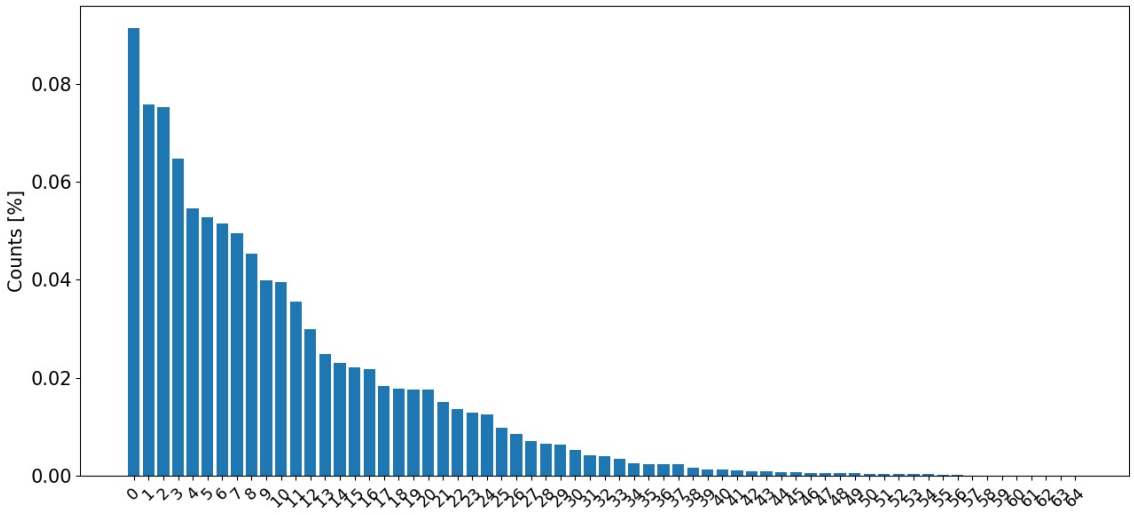


Figure 4.1. *Receiver speaker audio sample counts per phone model*

Each audio sample has already been analysed by the testing system and a pass/fail result has been given to the sample. The testing system evaluates the samples with signal processing metrics that each give a value for the signal. The verdict of pass or fail is derived by using threshold values. If the value received from a signal processing metric is over (or under depending on the metric) the threshold for any of the metrics measured, the sample is classified as failed. If the sample passes all the signal processing metrics used, the sample is classified as passed. The distribution of failed and passed results is presented in table 4.1.

Table 4.1. *Audio sample distribution to failed and passed samples*

Test outcome	Pass	Fail
Count	29 458	3 963

As the table presents, the number of passed audio samples is a lot greater than the number of failed samples. The imbalance ratio is calculated in the equation 4.1.

$$IR = \frac{29458}{3963} \approx 7.43 \tag{4.1}$$

The ratio is over 7, therefore it shows that the dataset is highly imbalanced for the favor of passed samples.

The results from the testing system were used as the ground truth for the samples used in the comparison study. The results were chosen to be the ground truth because human ear cannot hear the subtle changes in sound quality that the testing system can. Also, it was not necessary to use another set of labels because the objective of the study was to find out whether machine learning model can define a separation between subtle

changes in audio signal. The samples were pre-processed with methods described in the next section.

4.1.2 Data Preparations

Even though the samples were already classified by the testing system, there was a possibility that some samples were misclassified for some reason. One reason for false negative results was background noise during the recording. Due to this fact, the data was cleaned from clear misclassifications. This was done by listening all the samples. Clear misclassifications were assigned a new label.

The dataset used in the experiments was highly imbalanced in sense of failed and passed samples, as described in the previous section. Before training, the data imbalance had to be fixed. The data set was re-sampled using the SMOTE oversampling technique presented in section 3.4. For every train-test split minority class samples were oversampled using SMOTE generating failed samples so that failed class had 20% the amount of samples that passed class had.

Each sample had 96 000 floating point values resulting in 188 KB in file size. Taking all the dataset samples the data size was over 6 GB. The amount of data passing through a neural network is directly related to the training time. To reduce the training time, sampling rate of 11 025 Hz was used, reducing the data set size to 1.5 GB. This means that the input size for the classifiers was 22 050 floating point values.

4.2 Used Classification Methods

This section presents the models used in the comparison study. The models use different design principles but for each model the same activation function, ReLU, was used in the convolutional and dense layers except for the output layer for which sigmoid activation function was used. Also, all of the models are optimized with adam optimizer and the loss function used was binary-crossentropy.

In each sub section, a visualization of the architecture of the models is presented and each visualization obeys the following rules. Convolutional layers are visualized as Conv A/B, C in the figure where A is the kernel size, B is the number of strides and C is the number of feature maps. Max pooling layers are presented as MaxPool D/E where D is the pooling size and E is the number of strides used. The number in dense layer is the number of neurons and the number in dropout layers is the percentage of weights dropped.

4.2.1 Frame level 1D CNN

One of the classifiers implemented for the comparison study was a model presented by Abdoli S. et al. [29]. This model was chosen into the comparison study because the classifier was built for environmental sound classification task. The data used in the comparison study is not meant for human ears and therefore models that have been made for music or speech data were found least suitable for the study. Environmental sound classifiers need to work with machinery sounds such as sirens or drilling sounds which are closest sounds to the signal used in the dataset.

In the article Abdoli S. et al. present multiple architectures for the model related to the input size given to the network. The model used in this thesis is a copy of the architecture presented for input size 32 000. The model will be called frame-level model from now on. The network architecture of the frame-level model used in the comparison study is visualized in figure 4.2 along the output shape after each layer.

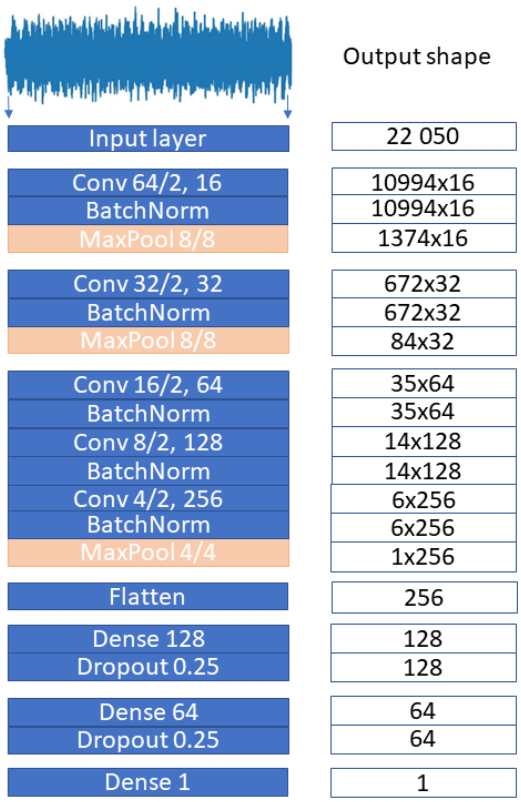


Figure 4.2. *Frame-level architecture visualization*

The model has 290 481 parameters, where 289 489 are trainable and 992 are non-trainable parameters. Because the model contains five strained convolutional layers and 3 max pooling layers, the signal is sub sampled all the way to 1 item per feature map in the convolutional layers. The model inputs only 256 items to the fully connected layers and therefore limits the number of trainable parameters in the fully connected layers.

Due to the low number of parameters in the fully connected layers and the low number of feature maps in the bottom layers, the total number of parameters in the network stays reasonably low compared to other deep networks that can have millions of trainable parameters [22][30].

4.2.2 Sample level 1D CNN

SampleCNN by Lee J. et al. [34] was chosen to be the inspiration for another model used in the comparison study. The model is referred as the sample-level model from now on. SampleCNN was chosen to the comparison study to experiment the effect of the reduced filter size to the features extracted and therefore to the overall performance.

The architecture of the sample-level model follows the architecture presented in the original paper by Lee J. et al. [31] that proposed sample-level architecture. SampleCNN model is designed with the principle of $m * m^n = input\ size$ as described in section 3.3. The closest architecture structure to input size 22 050 presented in the paper that follows the SampleCNN design principle is achieved with $3 * 3^8 = 19\ 683$. This architecture is presented in the figure 4.3.

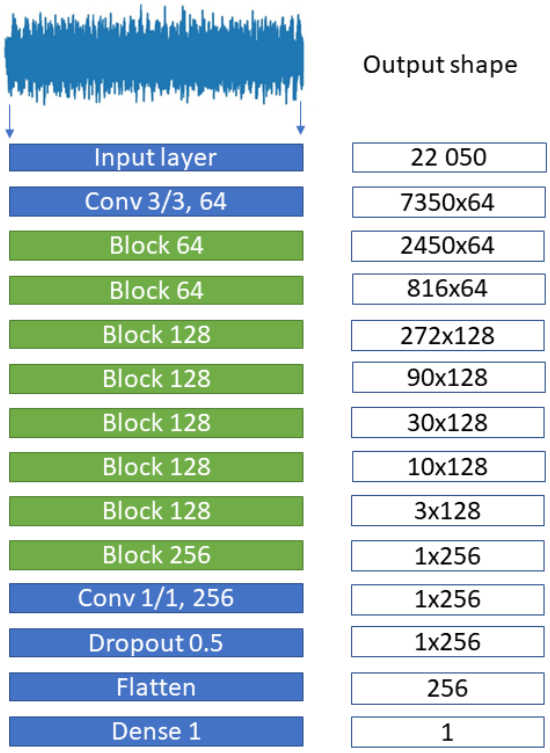


Figure 4.3. *Sample-level architecture visualization*

The middle part of the network consists of similar blocks presented in figure 4.4, where only the number of filters generated in the convolutional layer differ. The number of filters used in the block is shown in the block layers of the network architecture figure.

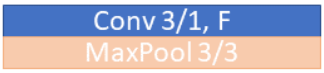


Figure 4.4. *Sample-level architecture block*

The network has 1 789 569 trainable parameters and 11 trainable layers. As the network does not have any fully connected layers except the output layer, the learning happens completely in the convolutional layers. Lee J. et al. discovered that having 128 filters in the first layer, increasing the number of filters to 256 in the block layers and eventually to 512 to the last layers to yield the best performance [31]. The model was trained with number of filters half of what was used in the original SampleCNN to reduce the number of parameter. However, it was found that better results were gained with exact same number of filters suggested in the SampleCNN architecture.

4.2.3 AclNet inspired model

The third model built for the comparison study was inspired by AclNet by Huang J. et al [49]. The model is referred as the two-block model from now on. AclNet model has a totally different architecture design compared to the frame-level and sample-level models. Because of that, AclNet was chosen to be one of the models in the comparison study.

The two-block models network architecture follows the same structure as AclNet. It has a 1D CNN block in the bottom called LLF (Low level features), reshaping layer in the between and a 2D CNN block on top called HLF (High level features). The 1D CNN block is visualized in the figure 4.5.

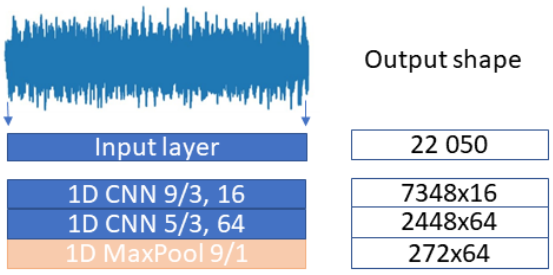


Figure 4.5. *Two-block architecture 1D CNN visualization*

The architecture of the LLF block follows the architecture of the LLF block in the AclNet. The number of strides was left for the user to decide. In the two-block model, the number of strides was chosen to be 3 in LLF block.

The architecture of the HLF follows the same architecture as the HLF block of the AclNet, except for the input size in the first layer due to the fact that the original audio signal was

of different size than the signal that AclNet was built for. The 2D CNN block is visualized in the figure 4.6.

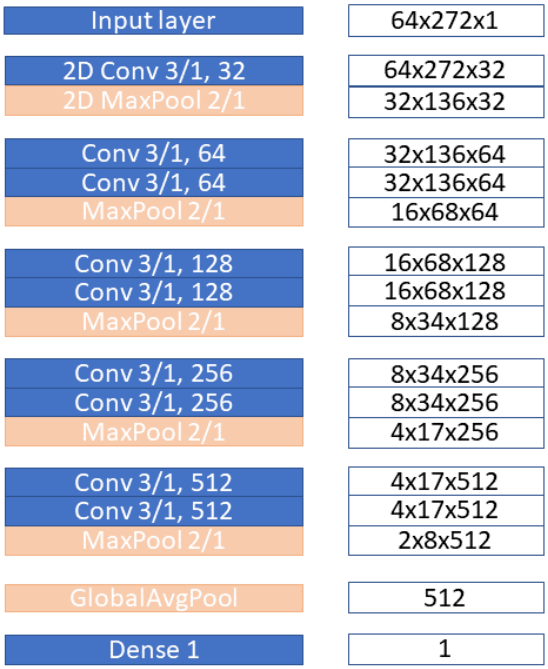


Figure 4.6. *Two-block architecture 2D CNN visualization*

The number of parameters in the whole network, including the 1D CNN block and the 2D CNN block, is 4 970 913. The number of parameters is high even though there are no fully connected layers. The comparably large number of parameters comes from the 12 trainable layers with up to 512 feature maps in the last convolutional layers.

4.3 Comparison study execution

Each classifier presented in section 4.2 was passed through a 5-fold cross-validation pipeline visualized in the figure 4.7. The pipeline was constructed from 5 folds, where each of the folds was used as the test set in turn and the rest of the folds were used for training. In every fold, the training data was resampled using the SMOTE oversampling technique. All of the classifiers were trained with the same data and evaluated against the same test split. Each model was trained using 25 epochs and batch size of 32 samples.

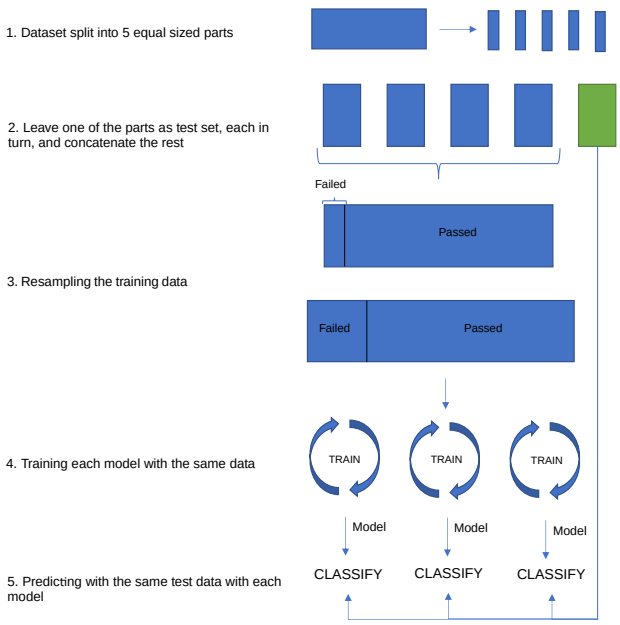


Figure 4.7. *5-fold cross-validation pipeline*

In the training phase, an additional validation split was used so that 20% of the training data was used as test data after each epoch. By having an additional validation data for each epoch, the performance of the classifier with unseen data can be tracked during the training phase. From each epoch loss and accuracy values were saved separately for training data and the validation data. The final model was chosen during the training phase by saving the model that had the lowest validation loss value.

In addition to the 5-fold cross-validation pipeline, leave-one-group-out cross-validation was used to see how the classifiers behave with phone models that they have not seen before. This evaluation was important since the audio quality classifier should be able to classify phones that it has seen before and also phones that have not been included in the training data.

Table 4.2. *Distribution of passed and failed samples among the chosen leave-one-group-out phone models*

Group	Passed samples	Failed samples
1	788	401
2	507	322
3	2 444	414
4	1 150	362

The evaluation was executed so that four phone models were chosen from the models used in the dataset. Samples belonging to each phone model acted as the test set in turn and samples belonging to all other phone models acted as the training set. The

phone models were chosen so that the test set would contain both passed samples and failed samples. The distribution of passed and failed samples per chosen phone model is presented in the table 4.2. The pipeline for resampling the training data and the training process were the same as with the 5-fold cross-validation.

The comparison study was executed using Tensorflow 2.3 and NVIDIA Quadro RTX 4000 GPU.

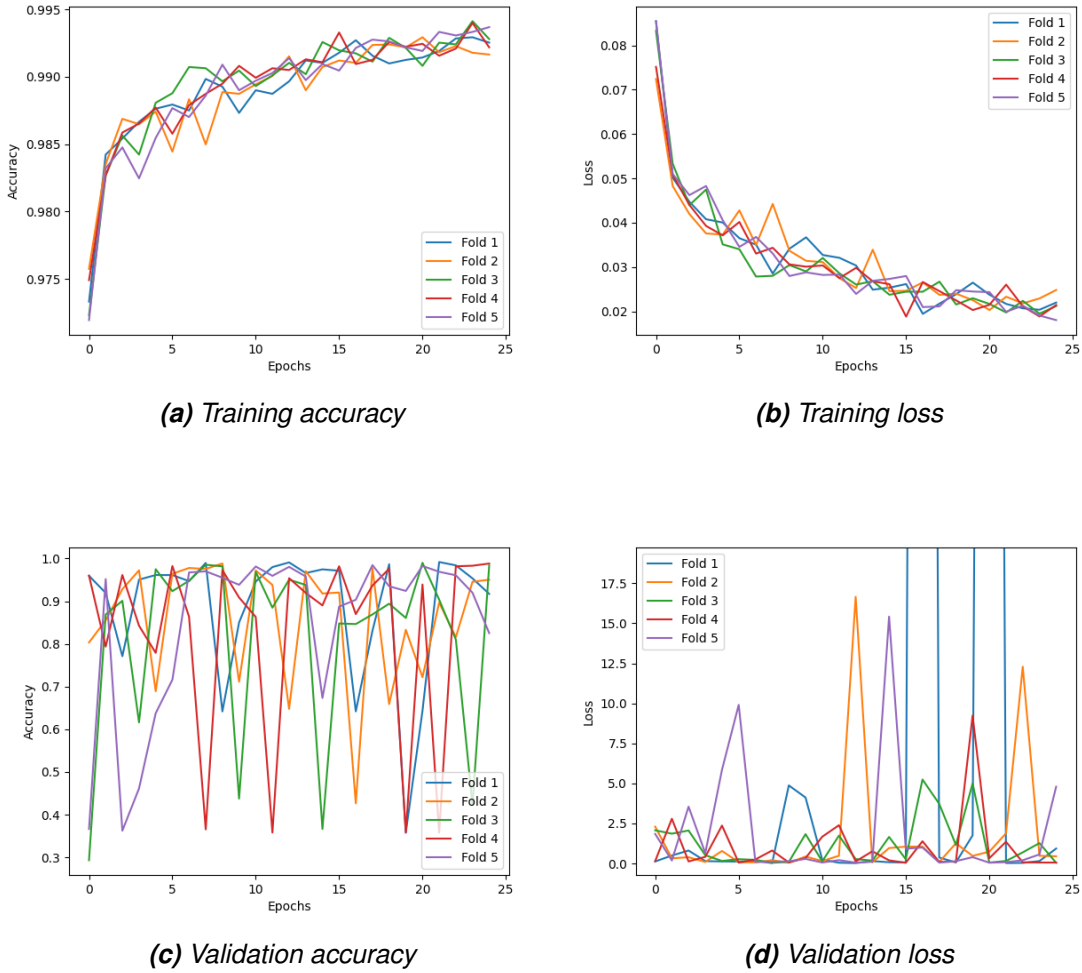
5. RESULTS

The results were evaluated in two parts: training time results and classification results. The training time results were evaluated based on the metrics gathered during the training phase, and the actual time it took to train the classifiers. The classification results were evaluated based on two cross-validation techniques: k-fold cross-validation which was used for evaluating the overall performance of each classifier, and leave-one-group-out cross-validation which was used for evaluating the generalization ability of each classifier. The classifiers were compared to each other using the performance metrics presented in the section 3.5. Only the k-fold cross-validation training phase results were evaluated.

5.1 K-fold cross-validation training evaluation

The training and validation loss and accuracy values over all epochs are plotted separately in figures 5.1 - 5.3 for each model used in the comparison study. For each figure the plots a and b are the training time loss and accuracy values whereas the plots c and d are visualizations of loss and accuracy values gained with the validation data.

For frame-level model, the training and validation plots (see figure 5.1) are very different in terms of the smoothness of the curves. The training time loss and accuracy curves are very smooth, there are no major differences in the values between the epochs. However, with the validation data, the values, both loss and accuracy, change dramatically between the epochs. Because of the dramatic changes that the loss values can have between the epochs, it is important to save the models weights after epochs that have reached a new minimum of loss values. If the weights used in the predictions would be the ones gained after the last epoch, the performance of the classifier would change dramatically also between each evaluation fold.



(a) *Training accuracy*

(b) *Training loss*

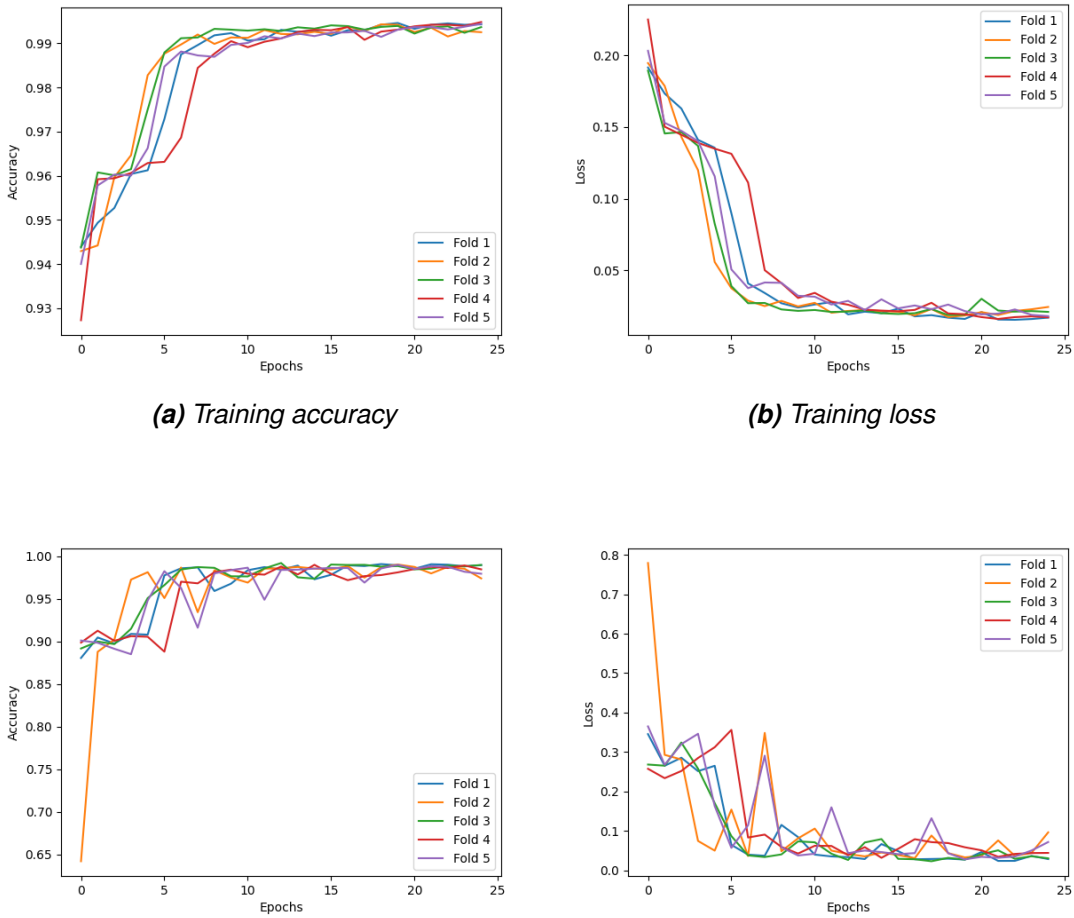
(c) *Validation accuracy*

(d) *Validation loss*

Figure 5.1. *Training and validation loss and accuracy over all epochs with frame-level model*

From the training time curves it can also be seen that the model starts converging to the maximum accuracy and minimum loss in just a few epochs. The training accuracy starts from value 0.97 and reaches 0.99 already at epoch 10. Similarly with the loss values, the starting value is around 0.5 but reaches 0.03 already at epoch 10. Although the validation curves are jumping up and down, for each fold the maximum value of validation accuracy is above 0.98 and the minimum loss value is at 0.05 or below.

The plots generated from the loss and accuracy values with sample-level model are shown in the figure 5.2. A clear trend can be seen from both (training and validation) accuracy and loss value plots. The values in each fold seem to follow a certain pattern which implies that the model's behaviour is predictable regardless the data set given to it.



(a) *Training accuracy*

(b) *Training loss*

(c) *Validation accuracy*

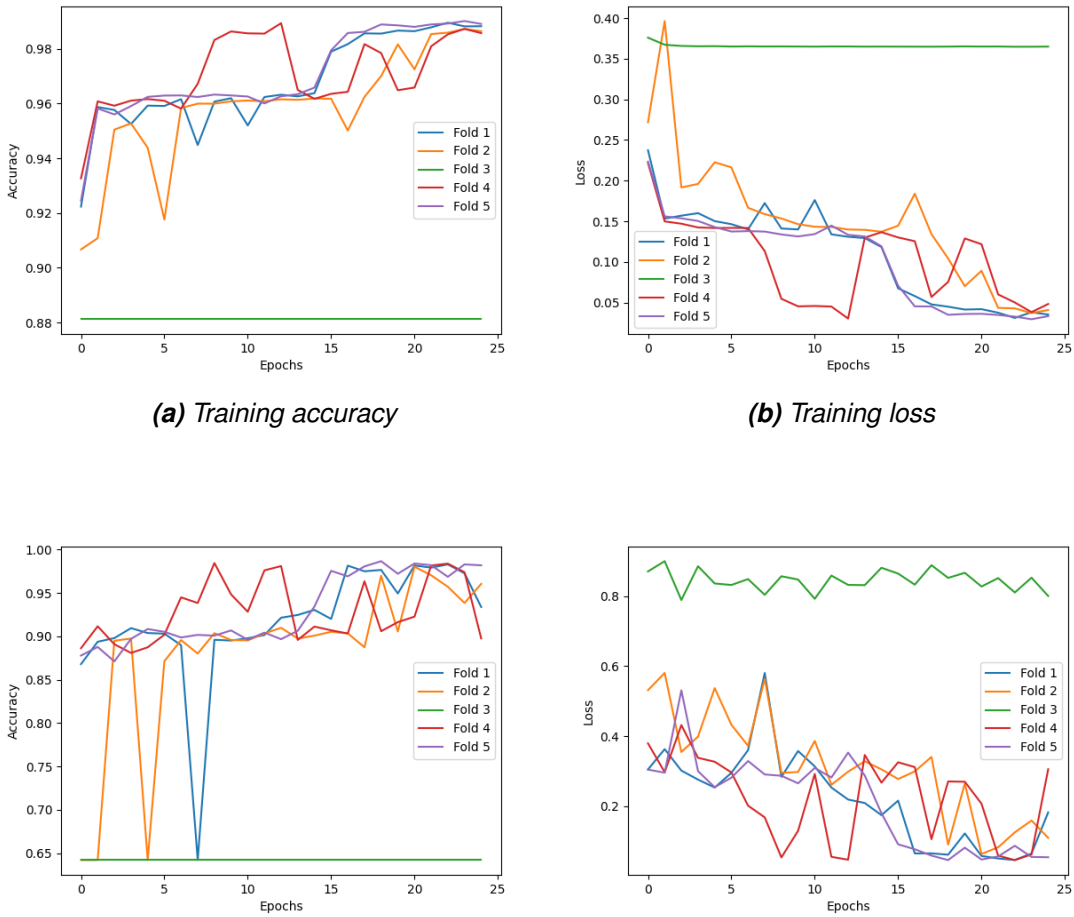
(d) *Validation loss*

Figure 5.2. *Training and validation loss and accuracy over all epochs with sample-level model*

The loss and accuracy values of the training phase seem to converge in two steps. The values reach the first step after a couple of epochs. The values stay in this level for another couple of epochs and then around epoch 10 converge to the final level of loss and accuracy. The model reaches above 0.99 training accuracy and below 0.025 training loss in each fold. Similarly to the training data plots, there is a clear pattern in the plots generated with validation data. The values do not converge as smoothly as the training ones, but there still is clear convergence happening towards the final epochs. Validation accuracy reaches 0.99 or above in each fold. The loss value converges around 0.04 or below in each fold.

Similar plots of loss and accuracy values for the two-block model are presented in the figure 5.3. The first thing that stands out from the training data plots is that loss and accuracy values of one fold have stuck to a single value. This means that the model has not learned, probably due to getting stuck in a local minima value, where the gradient of the loss is not changing anymore with the given learning step. For the other folds, there

is a clear convergence happening in the training data plots towards the last epochs. The model reaches 0.985 or above training accuracy and below 0.05 training loss in each fold except for the one that got stuck. For the validation data values, there is some spread on the curve shape. The curve of the same fold, that got stuck in the local minima, has a shape differentiating from the other loss and accuracy curves. However, it is clear that the trend in the validation accuracy is upwards and downwards for the loss curves. The model reaches 0.98 or above validation accuracy and 0.07 or below validation loss in each fold.



(a) *Training accuracy*

(b) *Training loss*

(c) *Validation accuracy*

(d) *Validation loss*

Figure 5.3. *Training and validation loss and accuracy over all epochs with two-block model*

Comparing the plots of the three models, there are clear differences. The frame-level model has the smoothest training loss and accuracy curves but has the most spread in the validation loss and accuracy values. The sample-level training metrics always seem to follow the same patterns. Thus, the sample-level model is the most predictable in terms of training behaviour. As for the two-block model, it is the only model that has loss/accuracy curves that have stuck on the same value, meaning the model has not learned after the first epochs. Based on the metrics from the training phase, sample-level model seems to

reach the best accuracy and loss values, 0.99 and 0.04 respectively, starts to converge already at epoch 10 and is predictable. Thus, based on the showed metrics, sample-level model seems to perform the best.

Table 5.1. *The training times from each fold and the averaged value for each classifier*

Classifier	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Frame-level	**6m 0s**	**5m 58s**	**5m 57s**	**5m 59s**	**5m 59s**	**5m 59s**
Sample-level	22m 39s	20m 37s	20m 37s	20m 39s	20m 38s	21m 2s
Two-block	11m 18s	11m 19s	11m 11s	11m 16s	11m 16s	11m 16s

In addition to the loss and accuracy values, also the time spent on the model training was documented. The training times were gathered from the training metric plots created automatically by Tensorboard. The training times for each model are shown in the table 5.1. There is a clear difference in the training times. Training the frame-level model takes only 1/4 of the time that is spent on training the sample-level model or half that is spent on training the two-block model. This is due to the low number of parameters (290 481) compared to the number of parameters in the sample-level (1 789 569) and two-block model (4 970 913). In this respect, the frame-level model performs clearly the best.

5.2 K-fold cross-validation results

The three models compared in the study were evaluated with 5-fold cross-validation pipeline described in section 4.3. The pipeline consisted of 5 folds. From each fold, classifier performance was evaluated with metrics: precision, recall, F2-score, accuracy and AUC. Saving the performance metrics from each fold, the stability of the classifier could be evaluated. The scores saved from all folds for each classifier are gathered to the table 5.2. In this table, the best values in each metric are bolded per fold. The model having the most bolded values in a fold has the fold number bolded as well indicating that the model has performed the best compared to the other two models.

Table 5.2. *Performance metrics for sample-level, frame-level and two-block classifiers from each fold*

Classifier	Fold	Accuracy	Precision	Recall	F2-score	AUC
Frame-level	**1**	**0.994**	**0.998**	**0.996**	**0.996**	**0.990**
	2	0.993	**0.999**	0.993	0.994	**0.993**
	3	0.994	0.994	0.999	**0.998**	0.977
	4	**0.993**	0.992	**0.999**	**0.998**	0.971
	5	0.991	0.997	**0.993**	**0.994**	0.985
Sample-level	1	0.993	0.996	**0.996**	**0.996**	0.985
	2	**0.995**	0.998	**0.996**	**0.996**	0.990
	3	**0.995**	**0.997**	0.997	0.997	**0.989**
	4	0.992	**0.997**	0.994	0.995	**0.986**
	5	**0.992**	**0.998**	0.992	**0.994**	**0.990**
Two-block	1	0.990	0.995	0.993	0.994	0.978
	2	0.987	0.995	0.990	0.991	0.978
	3	0.882	0.881	**1.0**	0.974	0.5
	4	0.992	**0.997**	0.994	0.995	**0.986**
	5	0.989	0.995	0.993	**0.993**	0.977

It must be noted that the accuracy values of the frame-level model are consistently above 0.99 in the table 5.2, whereas the validation accuracies presented in the figure 5.1c are ranging between [0.28, 0.99] between the epochs. The consistent behaviour of the model's predictions is possible because the model used in the predictions was chosen based on the minimum validation loss value among the epochs.

As can be seen from the table, the sample-level model has three folds with the best performance and frame-level model had the best performance in two folds. The two-block model has performed the worst by not having any best performing folds. Overall, the respective values of sample-level model and frame-level model are very close to each other. This means that their performance, measured with the metrics used, is very similar. All of the values for frame-level and sample-level models are above 0.99 except for AUC where the values range from 0.971 to 0.993 with frame-level model and from 0.985 to 0.990 with sample-level model. The range of AUC values for the two-block model is [0.5, 0.986] which is clearly worse than the values of the two other models. The range difference between AUC and the other metrics can be explained with the fact that AUC value indicates stronger the performance of the classifier with respect to the false positives

than the other metrics. In an imbalanced dataset, even a low number of false positives is going to affect the AUC performance by a large factor.

To summarize the performance of each classifier, the averaged values of the performance metrics over all folds are gathered to the table 5.3. The averaged values of sample-level model and frame-level model are very similar and in the case of accuracy and F2-score they are the same. More distinct difference between the sample-level model and frame-level model can be seen in the average value of AUC. The sample-level model clearly out performs two-block model but also has clear difference to frame-level model.

Table 5.3. *Performance metrics for sample-level, frame-level and two-block classifiers*

Classifier	Accuracy	Precision	Recall	F2-score	AUC
Frame-level	**0.993**	0.996	**0.996**	**0.996**	0.983
Sample-level	**0.993**	**0.997**	0.995	**0.996**	**0.988**
Two-block	0.967	0.973	0.993	0.988	0.883

In quality evaluation, classifying a bad quality sample as good quality sample (false positive classification) can be costly. Therefore, it is important to evaluate the classifiers performance in regard to the number of false positives. For this reason, the false positive rates (FPR) have been collected in the table 5.4 for each classifier. The values are calculated from the averaged number of false positives (fp) and number of true negatives (tn) over all folds.

Table 5.4. *False positive rate for each classifier calculated from averaged values fp and tn of each fold*

Classifier	False positive rate [FPR]
Frame-level	0.03
Sample-level	**0.019**
Two-block	0.227

Looking at the false positive rates presented in the table, it is clear that sample-level model has better ability to classify the bad quality samples correctly than the frame-level model or the two-block model, based on the lowest FPR value.

5.3 Leave-one-group-out cross-validation results

The ability to generalize was evaluated with leave-one-group-out cross-validation technique (see section 3.5) so that one phone model represented one group. The performance was evaluated with AUC and F2-score. The results are presented in the table 5.5.

Table 5.5. *Leave-one-group-out performance cross-validation performance metrics*

Classifier	Group	F2-score	AUC
Frame-level	1	**0.991**	0.958
	2	**0.994**	0.980
	3	**1.0**	**0.995**
	4	0.981	0.988
Sample-level	1	0.987	**0.966**
	2	0.992	**0.995**
	3	0.996	**0.995**
	4	**0.990**	**0.993**
Two-block	1	0.982	0.933
	2	0.960	0.835
	3	0.962	0.5
	4	**0.990**	**0.993**

The table shows that the performance of the three models is not as good as it was when samples of the phone models were included in the training data. This behaviour was expected because each phone model brings its own distinct patterns also into the audio signal. However, the results in F2-score are above 0.98 and above 0.95 for AUC values with frame-level and sample-level models. The performance of the two-block model is inferior to the two other models as expected based on the 5-fold cross-validation results and does not reach as good scores as the two others in either performance metric. The generalization ability is measured with the difference in the averaged values of F2-score and AUC between the 5-fold cross-validation and the leave-one-group-out cross-validation. The difference values are shown in the table 5.6.

Table 5.6. *Difference in averaged F2-score and AUC between 5-fold and leave-one-group-out cross-validation results for each model*

Classifier	Difference in F2-score	Difference in AUC
Frame-level	0.00450	0.00275
Sample-level	0.00475	0.00075
Two-block	0.01450	0.06775

The table shows that the average values in the F2-score and AUC between the two cross-validation techniques used is very small, less than 0.07 for all of the models and less than 0.005 for frame-level and sample-level models. Based on these results the ability to evaluate speaker audio quality of unseen phone models is as good as the ability to evaluate the quality of seen phone models. The table also shows that the ability to generalize is very similar between frame-level and sample-level models based on the F2-score. However in

the light of AUC, sample-level model clearly outperforms the two others in the ability to generalize.

6. CONCLUSIONS

In this thesis three CNN based models were implemented and compared using the general classification performance metrics. The objective of the thesis was to examine the possibility of using machine learning based classifier for mobile phone speaker audio quality evaluation regardless of the phone model.

The CNN based models participating in the comparison study were evaluated based on the training metrics (loss and accuracy), time spent on training, classification performance with 5-fold cross-validation and leave-one-group-out cross-validation. Based on the results presented in the previous section, it can be concluded that machine learning based model is able to find the subtle differences in the audio signal that separate good audio quality from bad audio quality. All of the models used in the comparison study attained over 0.90 accuracy on average and two of them (frame-level and sample-level) reached over 0.99 accuracy values. Also, the AUC value, which is important in imbalanced data evaluation, got over 0.98 score for frame-level and sample-level models on average.

In the light of the results presented in the previous section, models that use 1D CNN throughout the classifier architecture perform better for the task at hand than a model which converts the data from 1D to 2D in the middle of the network. Based on the loss and accuracy metrics and the training time, the two-block model does not perform the best in either. The frame-level model outperforms the two others in the training time, but the sample-level model outperforms the two others in the loss and accuracy metrics. In a situation where the stability and predictability of the model are more important that the training time, the sample-level model seems to be the model to use for the task at hand. On the other hand, the best accuracy and loss values measured with the frame-level model are not far from the same values of the sample-level model. The training of the frame-level model is 4 times faster than the sample-level model, thus in case where the training speed is a crucial factor in choosing the model, the frame-level model would be a good choice.

One of the requirements in this research was that the audio quality evaluation model would be able to classify the audio signal also from phone models that were not included in the training data. Based on the results presented in the Evaluation section, the frame-level and sample-level models fill this requirement. Given all the results, sample-level

model outperformed the competitors and thus it can be concluded that using small filter sizes in the convolutional layers suit the best for the audio signals used in the research.

The research was implemented using ground truth labels received from another testing system. Because of that, the machine learning based solutions could only detect defects that the other testing system could also catch and thus be only as good as the other testing system. As a future research, another dataset could be collected so that the ground truth labels would be annotated by a group of people. The label would be given based on subjective feedback from the observers not liming to the known defects. This way a machine learning based classifier could find patterns of defects that were not known and thus outperform the existing testing system.

REFERENCES

[1] Sousa Silva, M. de, Cruz, L. F., Bugatti, P. H. and Saito, P. T. M. Automatic Visual Quality Assessment of Biscuits Using Machine Learning. *Artificial intelligence and soft computing : 19th International Conference, ICAISC 2020, Zakopane, Poland, October 12-14, 2020, proceedings, part II /*. Cham, Switzerland : Springer, 2020, pp. 59–70. ISBN: 3-030-61533-2.

[2] Cubero, S., Aleixos, N., Moltó, E., Gómez-Sanchis, J. and Blasco, J. Advances in Machine Vision Applications for Automatic Inspection and Quality Evaluation of Fruits and Vegetables. *Food and Bioprocess Technology* 4.4 (2011), pp. 487–504.

[3] Liu, Z., Tang, R., Duan, G. and Tan, J. TruingDet: Towards high-quality visual automatic defect inspection for mental surface. *Optics and Lasers in Engineering* 138 (2021).

[4] Rathod, J. M. and Salehi, H. S. Vision system with deep learning classifiers for automatic quality inspection. *Proceedings of SPIE - The International Society for Optical Engineering*. Vol. 11400. 2020. ISBN: 9781510635777.

[5] Lin, H. .-. and Tsai, H. .-. Automated quality inspection of surface defects on touch panels. *Journal of the Chinese Institute of Industrial Engineers* 29.5 (2012), pp. 291–302.

[6] Thiede, T., Treurniet, W. C., Bitto, R., Schmidmer, C., Sporer, T., Beerends, J. G., Colomes, C., Keyhl, M., Stoll, G., Brandenburg, K. and Feiten, B. PEAQ - the ITU standard for objective measurement of perceived audio quality. *AES: Journal of the Audio Engineering Society* 48.1-2 (2000), pp. 3–29.

[7] Fazenda, B., Kendrick, P., Cox, T., Li, F. and Jackson, I. Perception and automated assessment of audio quality in user generated content. eng. *139th Audio Engineering Society International Convention, AES 2015*. 2015.

[8] Kendrick, P., Cox, T., Li, F., Fazenda, B. and Jackson, I. Wind-induced microphone noise detection - Automatically monitoring the audio quality of field recordings. *2013 IEEE International Conference on Multimedia and Expo (ICME)*. 2013. ISBN: 1-4799-0014-1.

[9] Kendrick, P., Jackson, I. R., Fazenda, B. M., Cox, T. J. and Li, F. F. Microphone handling noise: Measurements of perceptual threshold and effects on audio quality. *PLoS ONE* 10.10 (2015).

[10] Giannakopoulos, T., Siantikos, G., Perantonis, S., Votsi, N.-E. and Pantis, J. Automatic soundscape quality estimation using audio analysis. eng. *Proceedings of the*

8th ACM International Conference on pervasive technologies related to assistive environments. PETRA '15. ACM, 2015, pp. 1–9. ISBN: 1450334520.

[11] Alonso-Jiménez, P., Joglar-Ongay, L., Serra, X. and Bogdanov, D. Automatic detection of audio problems for quality control in digital music distribution. English. *AES 146th International Convention*. 2019.

[12] Temme, S. and Dobos, V. Evaluation of audio test methods and measurements for end-of-line automotive loudspeaker quality control. *142nd Audio Engineering Society International Convention 2017, AES 2017*. 2017.

[13] Theodoridis, S. and Koutroumbas, K. Chapter 1 - Introduction. *Pattern Recognition (Fourth Edition)*. Ed. by S. Theodoridis and K. Koutroumbas. Fourth Edition. Boston: Academic Press, 2009, pp. 1–12. ISBN: 978-1-59749-272-0. DOI: https://doi.org/10.1016/B978-1-59749-272-0.50003-7. URL: https://www.sciencedirect.com/science/article/pii/B9781597492720500037.

[14] Theodoridis, S. and Koutroumbas, K. Chapter 3 - Linear Classifiers. *Pattern Recognition (Fourth Edition)*. Ed. by S. Theodoridis and K. Koutroumbas. Fourth Edition. Boston: Academic Press, 2009, pp. 91–150. ISBN: 978-1-59749-272-0. DOI: https://doi.org/10.1016/B978-1-59749-272-0.50005-0. URL: https://www.sciencedirect.com/science/article/pii/B9781597492720500050.

[15] Theodoridis, S. and Koutroumbas, K. Chapter 4 - Nonlinear Classifiers. *Pattern Recognition (Fourth Edition)*. Ed. by S. Theodoridis and K. Koutroumbas. Fourth Edition. Boston: Academic Press, 2009, pp. 151–260. ISBN: 978-1-59749-272-0. DOI: https://doi.org/10.1016/B978-1-59749-272-0.50006-2. URL: https://www.sciencedirect.com/science/article/pii/B9781597492720500062.

[16] Izenman, A. J. *Linear Discriminant Analysis*. eng. 1st ed. 2008. Springer Texts in Statistics. New York, NY: Springer New York, 2008. ISBN: 1-282-03779-X.

[17] Izenman, A. J. *Support Vector Machines*. eng. 1st ed. 2008. Springer Texts in Statistics. New York, NY: Springer New York, 2008. ISBN: 1-282-03779-X.

[18] Weisberg, S. *Applied linear regression*. eng. 4th ed. Wiley Series in Probability and Statistics. Hoboken, New Jersey: Wiley, 2014. ISBN: 1-118-78955-5.

[19] Quinlan, J. R. Induction of Decision Trees. *Machine Learning* 1.1 (1986), pp. 81–106.

[20] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[21] O'Shea, K. and Nash, R. *An Introduction to Convolutional Neural Networks*. URL: https://arxiv.org/abs/1511.08458 (visited on 02/12/2021).

[22] Dai, W., Dai, C., Qu, S., Li, J. and Das, S. Very deep convolutional neural networks for raw waveforms. (2017), pp. 421–425.

[23] Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[24] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. *Learning Internal Representations by Error Propagation*. eng. 1985.

[25] Lecun, Y., Bengio, Y. and Hinton, G. Deep learning. *Nature* 521.7553 (2015). cited By 24341, pp. 436–444. DOI: 10.1038/nature14539.

[26] LeCun, Y., Huang, F. J. and Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. eng. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2004, pp. II97–II104.

[27] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. and Jackel, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. eng. *Neural computation* 1.4 (1989), pp. 541–551. ISSN: 1530-888X.

[28] Piczak, K. J. Environmental sound classification with convolutional neural networks. (2015), pp. 1–6. ISSN: 1551-2541.

[29] Abdoli, S., Cardinal, P. and Lameiras Koerich, A. End-to-end environmental sound classification using a 1D convolutional neural network. *Expert Systems with Applications* 136 (2019), pp. 252–263. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.06.040. URL: https://www.sciencedirect.com/science/article/pii/S0957417419304403.

[30] Krizhevsky, A., Sutskever, I. and Hinton, G. ImageNet classification with deep convolutional neural networks. eng. *Communications of the ACM* 60.6 (2017), pp. 84–90. ISSN: 0001-0782.

[31] Lee, J., Park, J., Kim, K. L. and Nam, J. Sample-level Deep Convolutional Neural Networks for Music Auto-tagging Using Raw Waveforms. eng. (2017). URL: https://arxiv.org/abs/1703.01789 (visited on 02/12/2021).

[32] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[33] Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. (2015), pp. 448–456.

[34] Lee, J., Park, J., Kim, K. and Nam, J. *SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification*. eng. 2018.

[35] Lee, J., Kim, T., Park, J. and Nam, J. Raw waveform-based audio classification using sample-level CNN architectures. *arXiv preprint arXiv:1712.00866* (Dec. 4, 2017).

[36] Sharma, G., Umapathy, K. and Krishnan, S. Trends in audio signal feature extraction methods. *Applied Acoustics* 158 (2020), p. 107020. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2019.107020. URL: https://www.sciencedirect.com/science/article/pii/S0003682X19308795.

[37] Al Dujaili, M. J., Ebrahimi-Moghadam, A. and Fatlawi, A. Speech emotion recognition based on SVM and KNN classifications fusion. *International Journal of Electrical and Computer Engineering* 11.2 (2021), pp. 1259–1264.

[38] Borisagar, K. R., Thanki, R. M. and Sedani, B. S. *Speech enhancement techniques for digital hearing aids*. Springer, 2019.

[39] Wang, A. The Shazam music recognition service. *Communications of the ACM* 49.8 (2006), pp. 44–48.

[40] Eronen, A. and Klapuri, A. Musical instrument recognition using cepstral coefficients and temporal features. *2000 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. 2. IEEE, 2000, pp. 753–756. ISBN: 0-7803-6293-4.

[41] Li, T., Ogihara, M. and Li, Q. A comparative study on content-based music genre classification. eng. *Proceedings of the 26th annual international ACM SIGIR conference on research and development in informaion retrieval*. SIGIR '03. ACM, 2003, pp. 282–289. ISBN: 9781581136463.

[42] Mesaros, A., Heittola, T. and Virtanen, T. Acoustic Scene Classification: An Overview of Dcase 2017 Challenge Entries. *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 411–415. ISBN: 153868151X.

[43] Alamir, M. A. A novel acoustic scene classification model using the late fusion of convolutional neural networks and different ensemble classifiers. *Applied Acoustics* 175 (2021), p. 107829. DOI: https://doi.org/10.1016/j.apacoust.2020.107829. URL: https://www.sciencedirect.com/science/article/pii/S0003682X20309348.

[44] Zahid, S., Hussain, F., Rashid, M., Yousaf, M. H. and Habib, H. A. Optimized Audio Classification and Segmentation Algorithm by Using Ensemble Methods. *Mathematical Problems in Engineering* 2015 (May 2015). DOI: 10.1155/2015/209814.

[45] Hussain, M. S. and Haque, M. SwishNet: A Fast Convolutional Neural Network for Speech, Music and Noise Classification and Segmentation. (Nov. 2018). URL: https://arxiv.org/abs/1812.00149 (visited on 04/16/2021).

[46] Purohit, T., Agrawal, A. and Ramasubramanian, V. Acoustic Scene Classification Using Deep CNN on Raw-waveform Technical Report. Detection, Classification of Acoustic Scenes and Events 2018: Challenge, 2018.

[47] Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J. and Wilson, K. CNN architectures for large-scale audio classification. eng. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135. ISBN: 1509041176.

[48] Salamon, J. and Bello, J. P. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters* 24.3 (2017), pp. 279–283.

[49] Huang, J. J. and Leanos, J. J. A. AclNet: efficient end-to-end audio classification CNN. eng. (2018). URL: https://arxiv.org/abs/1811.06669 (visited on 02/12/2021).

[50] Tokozume, Y. and Harada, T. Learning environmental sounds with end-to-end convolutional neural network. eng. (2017), pp. 2721–2725. ISSN: 2379-190X.

[51] Palaz, D., Doss, M. M. and Collobert, R. Convolutional neural networks-based continuous speech recognition using raw speech signal. (2015), pp. 4295–4299.

[52] Dinkel, H., Qian, Y. and Yu, K. Investigating Raw Wave Deep Neural Networks for End-to-End Speaker Spoofing Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.11 (2018), pp. 2002–2014. DOI: 10.1109/TASLP.2018.2851155.

[53] Hoshen, Y., Weiss, R. J. and Wilson, K. W. Speech acoustic modeling from raw multichannel waveforms. (2015), pp. 4624–4628.

[54] Sainath, T., Weiss, R. J., Senior, A., Wilson, K. and Vinyals, O. Learning the speech front-end with raw waveform CLDNNs. (2015).

[55] Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. eng. *arXiv preprint arXiv:1409.1556* (2014).

[56] He, H. and Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284.

[57] Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. SMOTE: Synthetic Minority Over-sampling Technique. eng. *The Journal of artificial intelligence research* 16 (2002), pp. 321–357. ISSN: 1076-9757.

[58] Douzas, G., Bacao, F. and Last, F. Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences* 465 (2018), pp. 1–20.

[59] Han, H., Wang, W.-Y. and Mao, B.-H. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. eng. *Advances in Intelligent Computing*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 878–887. ISBN: 3540282262.

[60] Chawla, N. V., Lazarevic, A., Hall, L. O. and Bowyer, K. W. SMOTEBoost: Improving prediction of the Minority class in boosting. eng. *Lecture notes in computer science*. Berlin: Springer, 2003, pp. 107–119. ISBN: 9783540200857.

[61] M, H. and M.N, S. A Review on Evaluation Metrics for Data Classification Evaluations. eng. *International journal of data mining & knowledge management process* 5.2 (2015), pp. 1–11. ISSN: 2231-007X.

[62] Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters* 27.8 (2006). ROC Analysis in Pattern Recognition, pp. 861–874. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.10.010. URL: https://www.sciencedirect.com/science/article/pii/S016786550500303X.

[63] Bekkar, M., Djemaa, H. K. and Alitouche, T. A. Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl* 3.10 (2013).

[64] Wong, T.-T. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition* 48.9 (2015), pp. 2839–2846. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2015.03.009. URL: https://www.sciencedirect.com/science/article/pii/S0031320315000989.

[65] *3.1. Cross-validation: evaluating estimator performance. 3.1.2.3.2. Leave One Group Out*. URL: https://scikit-learn.org/stable/modules/cross_validation.html#leave-one-group-out (visited on 03/03/2021).