University of Crete
Department of Computer Science

Speech Emotion Recognition with Convolutional Neural Networks

B.Sc. Thesis
Marianna Velessioti

Advisor: Yannis Stylianou
Supervisor: George Kafentzis

Περίληψη

Η ομιλία αποτελεί μία από τις πιο εκφραστικές μορφές επικοινωνίας. Οι εναλλαγές στη χροιά, τον τόνο, την ταχύτητα με την οποία μιλάμε, όταν πρόκειται για προφορικό λόγο, αυξάνουν καθοριστικά την εκφραστικότητα μας. Αυτή η παραγλωσσική πληροφορία μας δίνει τη δυνατότητα να εκφράσουμε σκέψεις και προθέσεις πέραν του λεκτικού περιεχομένου. Μια πληθώρα χαρακτηριστικών της φωνής απαρτίζουν το συναίσθημα που εμπεριέχεται σε αυτή, με αποτέλεσμα να γίνεται αντιληπτή η χαρά, η λύπη, η πλήξη, ο θυμός και άλλα συναισθήματα του ομιλητή.

Στην παρούσα πτυχιακή εργασία θα εισάγουμε το θέμα της συναισθηματικής ομιλίας στον τομέα της Επεξεργασίας Φωνής και θα ακολουθήσει μια ανάλυση με τη χρήση ενός Τεχνητού Νευρωνικού Δικτύου. Για την εξαγωγή συναισθήματος από το σήμα της φωνής έχουν χρησιμοποιηθεί διάφορες μέθοδοι Μηχανικής Μάθησης, ωστόσο εμείς προτείνουμε ένα Συνελικτικό Νευρωνικό Δίκτυο, που παίρνει ως είσοδο εικόνα και εξάγει κατανομές συναισθημάτων, από τις οποίες συμπεραίνουμε την τελική ταξινόμιση. Κάνουμε χρήση επίσης μιας πληθώρας βάσεων δεδομένων, για την εκπαίδευση και την αξιολόγηση του μοντέλου, καθεμία από τις οποίες ενδέχεται να υποστηρίζει διαφορετικό αριθμό (και είδος) συναισθημάτων. Ωστόσο το πλήθος τους είναι πεπερασμένο και αναμένεται να εξάγουμε ένα τη φορά για κάθε ηχογράφηση ομιλίας.

Η τεχνική που ακολουθούμε περιγράφεται αναλυτικά, με τις λεπτομέρειες και τα trade-offs της, καθώς και μέθοδοι για βελτίωση της απόδοσης και την ανάκτηση αμερόληπτων αποτελεσμάτων. Παρουσιάζονται, επιπροσθέτως, οι διαδικασίες της δειγματοληψίας και της προεργασίας των δεδομένων προτού αυτά εισαχθούν στο Νευρωνικό Δίκτυο. Τελος, παρατίθενται αποτελέσματα των πειραμάτων που διεξάχθηκαν έπειτα από ανάλυση δεδομένων για να αξιολογηθεί το δίκτυο, για κάθε βάση δεδομένων ξεχωριστά.

Abstract

Speech consists one of the most expressive forms of communication. The alternations in the tone, the volume, the speed, when it comes to oral speech, significantly increase one's expressiveness. This paralinguistic information allows the speaker to express thoughts and intentions apart from the verbal content. A variety of voice traits make up the emotion contained in it, resulting in perceived joy, sadness, boredom, anger and more.

In this thesis, we introduce the subject of the expressive speech in the context of Speech Processing and we follow up with an analysis using a form of Artificial Neural Network. For the extraction of emotion from speech, a variety of Machine Learning techniques gives interesting research outcomes. However, we propose a Convolutional Neural Network, that takes image input and outputs emotion distributions, from which we conclude in the final classification. We also make use of a variety of databases, for the training and evaluation of the model, where each of them might support different number (and type) of emotions. However, their number is finite and we expect to extract one at a time -one for each speech recording.

The technique we follow is described in detail -with all the implicated trade-offs- as well as methods for performance improvement and the acquisition of unbiased results. Furthermore, we present the processes of data sampling and preprocessing before they get fed into the Neural Network. Finally, results of the experiments performed for the evaluation of the network are presented for each database separately.

Contents

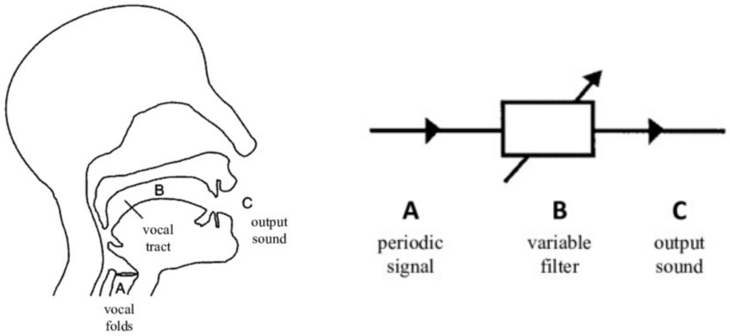
Chapter 1

Introduction

1.1 Human Speech

The human voice has two main characteristics; it can be produced by the human body and it can be perceived as a signal. These mechanisms are related to each other inside the human brain. As long as our brain receives the voice signals, it creates patterns that help us recognize henceforth similar signals.
In engineering, we are interested in the processing of this signal in order to create useful applications. Speaking about language, we refer to something different than plain voice. The language has structure, while the voice does not. Speech is therefore a structured form of the human voice that is produced using a language syntax.

In terms of signal processing, it is widely accepted that voice can be modeled with an input A that is given to a system B and the result is an output signal C, as shown in the figure 1 below. The vibration of the vocal folds produces a varying airflow which may be treated as a periodic signal A, called source, that produces a spectrum of equally-spaced frequency peaks or harmonics, starting with a fundamental frequency F0. The resonance frequencies of the vocal tract (F1, F2, F3) are called *formants* and they can be displayed as spectral peaks in the frequency response of the vocal tract filter. This source signal is input to a system B (the vocal tract). The tract behaves like a variable filter. Its response is different for different frequencies and the frequency response may be further adjusted by changing the position of the tongue, jaw etc.



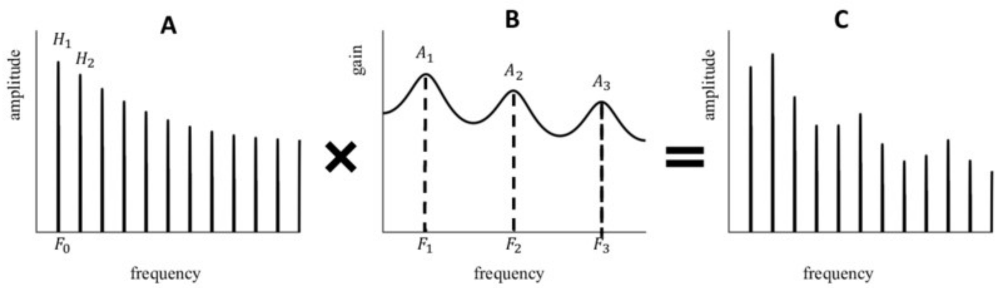


Fig.1 : speech apparatus (top), source-filter model (middle), frequency domain representation (below) [1]

Resonance peaks (A1, A2, A3) add gain to specific frequencies of the harmonic spectrum. The input signal and the vocal tract, together with the radiation properties of the mouth, face and external field, produce the sound output C. These resonances can be determined approximately from the formants (peaks) in the envelope of the sound spectrum. Given a different spectrum A with higher or lower fundamental frequency, the frequency of the output spectrum C will be different as well.

According to the source-filter theory, the vocal-tract filter becomes a linear time-invariant (LTI) system, and an output signal y(t) can be expressed by the convolution of an input signal x(t) and the impulse response of the system h(t). In discrete time:

$$x[n] \to \boxed{\text{h[n]}} \to y[n]$$

That is described with the convolution equation:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$$

The system with impulse response h[n] modifies the signal x[n] in the appropriate way so that the output signal y[n] can be rendered differently each time. The role of the ‘system’ is played by the vocal tract. For instance, we set our mouth and articulators in a way to pronounce the phoneme /a/, but in a different way to pronounce the phoneme /o/.

1.2 Emotion in human speech

This thesis does not focus on features of plain voice but on structured speech, which in fact, includes information about the emotional state of the speaker. This means speech does not only bear a message but also paralinguistic information. This is important because we can understand the emotional condition of the speaker and consider it as a factor in order to make the appropriate decisions. Furthermore, the emotion as a phenomenon colours speech and act as a necessary ingredient for natural two-way communication between humans.

There have already been developed multiple systems that process the speech signal and draw conclusions for the emotional state of the speaker using feature extraction. Systems can use audio only, text only (context-based), image only (face recognition) or a combination of these types of features (multimodal). Focusing on audio only systems, most of them rely on specific features extracted from the speech signal and fed into a classification algorithm.

It would be very useful to build systems that can extract the emotional state of a speaker directly from his/her speech signal, without the necessity of linguistic analysis -just the way that the human brain does!

1.3 Extraction of emotion from speech

One step beyond plain speech recognition lies the recognition of subjective patterns of speech such as the emotion. For extracting emotion from speech signals, algorithms that focus on identifying patterns in human speech have been developed. The speech signal that corresponds to each emotion is found to have different characteristics. Patterns in speech are described by features; features are sets of numbers that collectively and compactly represent characteristics of the speech signal. In the literature, features such as formants, zero-crossing rate, pitch, speech rate, energy, and loudness are often used [6]. However, more sophisticated features such as LPC coefficients, MLPC coefficients, Teager-Kaiser energy, PLPCs, MFCCs, filterbanks, chroma vectors, and other spectral properties (centroid, flux, spread, entropy) have been successfully used [9, 15, 24]. These data are meant to be extracted "automatically" by the respective pattern recognition system.

1.3.1 Emotion Literature and Classic Methods

Before the outburst of CNNs, that are reviewed extensively in the next chapter, a variety of machine learning methods had been developed and used in practice, which offer a great deal of research outputs for speech emotion recognition so far. Classification techniques based on PCA (Principle Component Analysis) [18], LDA (*Linear Discriminant Analysis) [19],* SVMs (*Support Vector Machines*) [17, 20], *kNN* (k-Nearest Neighbors) [13, 14], GMMs (*Gaussian Mixture Modelling*) [9, 16], HMMs (*Hidden Markov Models*) [17], and ANNs (*Artificial Neural Networks*) [10, 11], are briefly reviewed.

Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) are two major techniques used for dimensionality reduction. These methods reduce the dimensionality by projecting the original feature space into a smaller subspace through a transformation. According to the first one, a linear classifier that separates the data into the different classes of a dataset has to be found. This is reduced to a problem of optimization of a cost function and is usually implemented using iterative optimization methods (such as Gradient Descent).

PCA is a method for compressing data into a frame that captures the essence of the original data. In fact, converts the correlations (or the lack of correlation) among the data into a lower dimension; usually in 2-D or 3-D. The samples that are highly correlated cluster together. Some studies in speech emotion recognition adopted PCA to analyze the feature sets [18]. Others use LDA, which actually performs better than PCA in many applications [3, 5]. However, the reduced dimensionality must be less than the number of classes, which consists a limitation for the technique [8].
A feature selection technique named Sequential Forward Selection [6] combines the nearest mean and Bayes classifier where class PDFs are approximated via Parzen windows or modelled as Gaussians. After selecting the best features, the dimensionality is reduced by applying PCA for less computational complexity.

Support Vector Machines (SVMs) is a natural extension of LDCs (Linear Discriminant Classifiers) which provides good generalisation properties even for a large feature vector. The main idea of SVMs is to move the data into a higher dimension and use a Support Vector Classifier (SVC) that separates the higher dimensional data into groups. They use Kernel Functions, such as the Polynomial Kernel, to transform the original input set to a high dimensional feature space and systematically achieve an optimum classification in the new feature space with the SVC. The results of a relevant survey [7] highlight that the technique can achieve high emotional classification scores on both male and female speech.

k-Nearest Neighbors (k-NN) classifiers are also popular since the very first studies [13, 14]. According to this algorithm, a sample is classified by a plurality vote of its neighbors, with the sample being assigned to the class that is most common among the k -already classified- samples whose feature vector has closer values to the new observation. *k* is a positive, usually small integer. Note that training data normalization and weight assignment to the contributions of the neighbors has been proposed to improve the accuracy dramatically. k-NN turned out to be efficient for acted and non-acted emotional speech but, in accordance with LDCs, show problems with “the increasing number of features that leads to regions of the feature space where data is very sparse” [4].

A very powerful and popular soft clustering algorithm is the Gaussian Mixture Modeling [16] which is based on the Gaussian (normal) distribution. In effect, each cluster is modeled according to a different Gaussian. This model is commonly extended to fit a vector of unknown parameters after a several number of iterations upon the data. This flexible and probabilistic approach to model the data inflects that rather than having hard assignments into clusters, like on k-means clustering, each data point can be generated by any of the distributions with a corresponding probability. A research [9] upon the performance of GMMs utilises spectral features of the data and models each emotion as a mixture of Gaussian densities. This method performed better than k-NN for the Berlin emotional database.

Hidden Markov Models have been used widely for speech recognition and emotional speech recognition [2, 3, 17] as well as for a wider variety of applications for sequence modeling. They can be considered as a subclass of the framework of Dynamic Bayesian Networks (DBNs). It is yet another probabilistic model based on the Bayes' theorem that implicates the different states that make up a given dataset. As we feed in evidence about the observed acoustic signal and we run probabilistic inference over this model what we get out is the most likely set of -in our case- phonemes that give rise to the speech signal in question. The overall model can be viewed as graph with finite states -that correspond to phonemes- and transitions, whose weight values represent the probability P(stateX | stateY) i.e. the probability of stateX being the next state given that the stateY is true. While they seem unstructured at the level of random variables, there is a perceived structure that manifests in the sparsity of the conditional probabilities and also in terms of repeated elements within the transition matrix. In the emotional speech recognition case, this means that the same “phoneme” can occur in multiple different acoustic signals and then we can replicate that structure across the different places where the same pattern can be used in the signal.

Finally, Artificial Neural Networks (ANNs) is the most used non-linear discriminative classifier together with decision trees [1]. This model was firstly designed to simulate the way the human brain analyzes and processes information. ANNs have self-learning capabilities that enable them to produce better results as more data become available. Relevant surveys on classic ANN based emotion classification and recognition have provided interesting results with decent classification accuracy for the most common emotional states [10, 11].

1.3.2 Deep Learning Methods

Speaking about ANNs, the modern bibliography tends to suggest models that aim to approach the human brain functionality. In effect, our brain can effortlessly recognize an image of a number, a shape, or an animal because it is trained to recognize these elements from a very young age. Thus, our study, closely related to neural networks, is initially inspired by the human brain, probably the most vital and complex among the human organs, able to recognize objects/patterns with an incredibly intelligent way.

Considering it as a system, the brain network gets its input from the visual cortex and then it processes the snapshot it receives internally. As soon as it receives again an image of a similar object (of the same category) it will most probably “recognize” it.

The more the images of objects of the same category it receives, the easier for the brain to recognize similar objects in the future, even if it perceives them for the first time. Our visual cortex resolves these as representing the same concept. This happens until the process of recognition becomes trivial for our brain.

Such an idea led to the outburst of Deep Learning, a popular, modern day batch of techniques and methods for pattern recognition. Many giant companies (like Google, Facebook, Amazon, YouTube) use Deep Learning at the core of their services in order to upgrade the level of data manipulation and services [25].

The first modern neural nets that were used, known as DNNs (Deep Neural Networks), were composed of simple dense layers. Afterwards, the introduction of convolutional, max pooling and other fully connected layers presented a significant improvement that performed more effectively for special classification purposes. Convolutional Neural Networks (CNNs) for instance, are good for image recognition. Long short-term memory networks (LSTM) on the other hand are found to be good for speech recognition.

Deep Neural Networks

DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back. They consist of an input and an output layer of neurons with inner fully connected layers. A typical DNN is shown in Fig. 2.

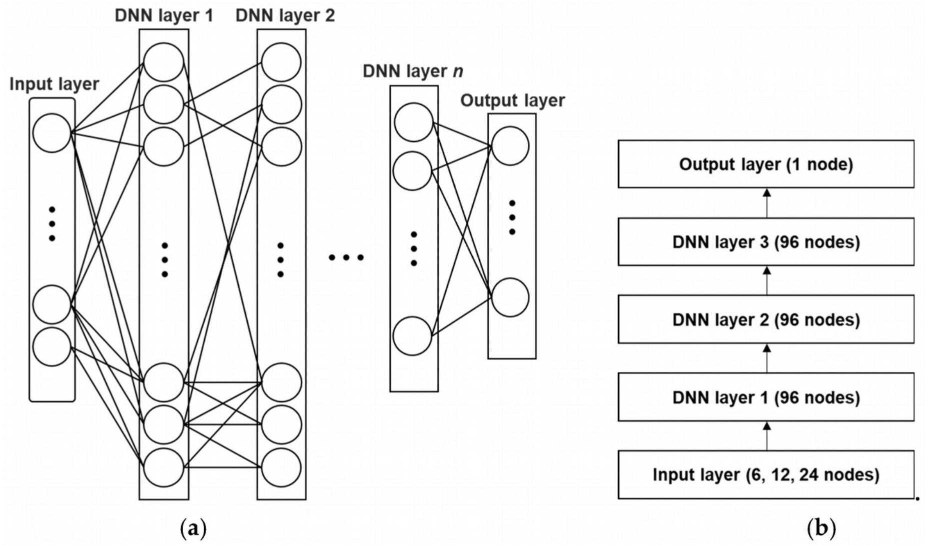


Fig. 2: DNN layers (a), Flow of information between the DNN layers (b) [2]

The layers in between are called “hidden layers”. An artificial neural network is called “deep” when the hidden layers are more than one (usually they are multiple). The above network consists of n hidden layers each one of 96 nodes. Each node is

connected to all the nodes of the next layer. Each one of those connections has its own weight associated with a neuron. Also each node has a bias term. “Learning” is referred to finding the right weights and biases of the network and is achieved during the part of the network’s training that is called **backpropagation**.
The classification data feed the network through the input layer and what we expect from the output is a number that indicates the right class. For an emotional speech recognition model, each neuron of the last layer represents the probabilistic estimation of the system on how much the given image corresponds to the given emotion.

Activations in one layer determine the activations of the next layer. Namely, each node called “neuron” holds a value (or set of values) and each connection holds a relevant weight. Accordingly, each node of a hidden layer is going to receive a weighted sum calculated from the previous layer, described as follows:

$$w_1a_1 + w_2a_2 + \cdots + w_na_n$$

When we compute a weighted sum like this, the result might be any number. For these networks however, we want the activations to be some value between [0 , 1]. The range is selected for two reasons: first to introduce non-linearities and second to obtain a probabilistic output. This is why we use a function that squishes the real number line into this range. A common function that does this is the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

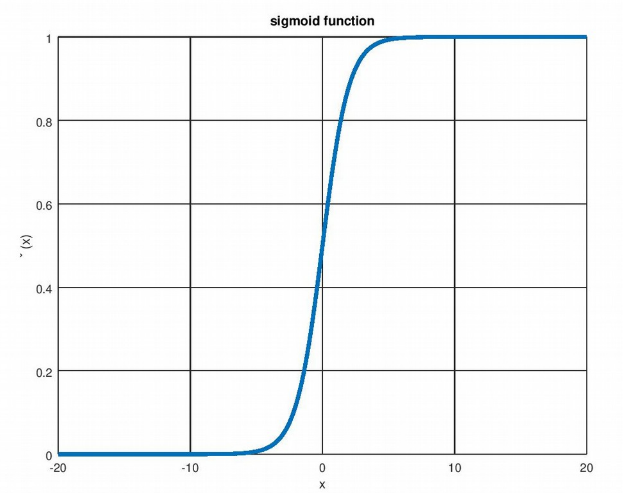


Fig. 3: The sigmoid activation function

Apparently, the very negative values converge to 0 and the very positive ones converge to 1.
Therefore the new computed sum is enclosed by the “activation” function and becomes as follows:

$$\sigma(w_1a_1 + w_2a_2 + \cdots + w_na_n - Bias)$$

The weights indicate what pixel pattern this neuron in the next layer is picking up on, and the “Bias” scalar indicates how high the weighted sum needs to be before the neuron starts getting meaningfully active. That is just the connection of one neuron with all the neurons of the previous layer. In linear algebra:

$$\begin{bmatrix} w_{00} & w_{01} & \dots & w_{0k} \\ w_{10} & w_{11} & \dots & w_{1k} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n0} & w_{n1} & \dots & w_{nk} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} + \begin{bmatrix} bias_0 \\ bias_1 \\ \vdots \\ bias_n \end{bmatrix}$$

By calculating the inner product for each row with the vector and then adding the biases we get a vector where each value corresponds to a neuron. After wrapping up these results by an activation function, for instance a sigmoid (as shown above) or ReLU, we get an activation layer ready to pass through the next layer. Thanks to the activation functions, neural networks are effective in modeling nonlinear mappings.

Training

During training, data may pass through the network multiple times. The number that indicates one single feedforward passage of full trainset is called an **epoch**. In one epoch, data can be split to smaller **batches** that feed the network all at once. The overall dataset can be split into training, validation and test data. In order to evaluate a model we use **performance metrics**, like accuracy. The training accuracy, is the average accuracy the network achieves in one training epoch on the validation (tuning) subset, while the test accuracy is the average accuracy of the network for a given test set (separate from the train set) after one training period. In both cases a forward propagation happens and once the final calculated values reach the output layer, we normally possess a number that represents the possibility for each class to be the right one (and finally pick the one with the highest possibility). A significant distinction here, is that during training we also have backpropagation, which means that the network is updating its weights and biases, while during testing we do not. The only purpose in the last case is to educe the results from a network that has previously learnt to recognize the target patterns. Every single train statistic (mean accuracy, standard deviation) contributes to the overall accuracy of the epoch.

Moreover, data can be **labeled** (hold the name of the class) or **unlabeled**. During training, data -usually- are labeled. When we use a validation set (different than the trainset) to pass data through the network after the training process, these data are also labeled. By making this step, we aim to examine whether our network performs “overfitting” upon the training set. That is, to test whether our network makes a sufficiently accurate classification into data in which it has not yet been trained, or it fits exactly against its training data in a way that leads to misclassifications over the new data. This behavior is explained more extensively in a subsequent section of Chapter 2. Finally, the testing data are fed into the network, which are unlabeled. In fact, it is really common for the test set to perform the activity of the validation and testing at the same time.

1.3.3 Data Bases

The majority of emotional speech data collections encompasses six or seven emotions, although the emotion categories are much more in real life. Although today there are two predominant approaches for the emotion representation, we followed the approach that categorizes the emotions in discrete classes. The other approach categorizes each emotion according to a tuple of speech specifications; *arousal* and *valence*. Respecting the first approach, the most common emotions are: anger, fear, sadness, joy and surprise. Non-basic emotions are called ‘‘higher-level’’ emotions and they are rarely represented in emotional speech data collections. In order to validate our model we used a variety of datasets.

The overall datasets we used for our experimental work are briefly the following:

Berlin

German database consisting of 535 samples. The emotions it provides are 'neutral', 'anger', 'boredom', 'disgust', 'fear', 'happiness' and 'sadness'.

AESI

Greek database consisting of 696 samples. The emotions it provides are 'anger', 'fear', 'joy', 'neutral' and 'sadness'.

SAVEE

English database consisting of 480 samples. The emotions it provides are 'anger', 'fear', 'disgust', 'neutral', 'happiness', 'sadness' and 'surprise'.

TESS

British english database consisting of 2800 samples. The emotions it provides are 'anger', 'fear', 'disgust', 'neutral', 'happiness', 'sadness' and 'surprise'.

CaFE

French database consisting of 936 samples. The emotions it provides are 'anger', 'disgust', 'happiness', 'neutral', 'fear', 'surprise' and 'sadness'.

RAVDESS

English database consisting of 1140 samples. The emotions it provides are 'neutral', 'calm', 'happy', 'sad', 'angry', 'fearful', 'disgust' and 'surprise'. For this dataset specifically, we had to eliminate the long silences at the beginning and at the end of the signals, as they spoiled the performance when using augmentation.

In further detail:

The **Berlin Emotion Speech Database (BES)** is the most often used database in the emotional speech processing community. It is an acted emotional content database created by the audio recordings of ten actors; five male and five female of age 21-35 years.

Athens Emotional States Inventory (AESI) [see web resources, 6] is a dataset in Greek containing audio recordings of five categorical emotions. The items of the AESI consist of 35 sentences each having content indicative of the corresponding emotion. The resulting data include recordings from 20 participants (12 male, 8 female), which resulted in 696 utterances.

Surrey Audio-Visual Expressed Emotion (SAVEE) database [see web resources, 7] has been recorded as a pre-requisite for the development of an automatic emotion recognition system. The database consists of recordings from 4 male actors in 7 different emotions, 480 British English utterances in total.

Toronto Emotional Speech Set (TESS) [see web resources, 8] is a collection of 200 target words spoken in the carrier phrase "Say the word _____' by two actresses (aged 26 and 64 years). Recordings were made of the set portraying each of seven emotions. There are 2800 stimuli in total.

The **Canadian French Emotional (CaFE)** speech dataset [see web resources, 9] contains six different sentences, pronounced by six male and six female actors, in six basic emotions plus one neutral emotion, composing 936 utterances in total. The six basic emotions are acted in two different intensities.

The **Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)** [web resources, 10] contains recordings of 24 actors (12 male, 12 female), vocalizing two lexically-matched statements in a neutral North American accent. The different emotions count to 8 and the vocal channel is either speech or song. There are also two different emotional intensities provided (except neutral emotion). For our experimental work, we used the audio-only modality format, which contains 1440 utterances. Nevertheless, the RAVDESS collection includes 7356 files in total (1440 speech, audio-only + 1012 song, audio-only + 2880 speech, video + 2024 song, video).

All databases were sampled at 16kHz, except CaFE that was initially sampled at 192kHz (we used the downsampled version of 48kHz) and SAVEE at 44.1kHz. However they were both resampled to 16kHz for consistency reasons.

Whether the classification model is speaker dependent, that is, the training corpus includes speech samples from the same speaker, is a matter of choice, as we can choose to remove completely a speaker from the training set and add them only to the test set. However, a real-life emotion recognition system should be speaker-independent [12].

1.4 Objective of the work

Developing and training a neural network has been initially inspired by the human brain functionality -in order to deploy effective systems analogous to the effectiveness of the human brain to recognize patterns. However, the relation between the biological brain and the deep neural networks nowadays is negligible. The objective of our work consists of the computer-based extraction of speech emotion using a convolutional neural network.

1.5 Outline

The rest of this thesis report is organized as follows. Chapter 2 introduces the idea of convolutional neural networks as well as their functionality in depth by analysing more extensively the training process. For this purpose, we shall describe its components and methods that are used for training and optimization. The same chapter details issues of sampling and preprocessing. It also presents the architecture of our model for speech emotion recognition as well as performance issues. Experimental results of the network's overall performance for all the data bases in combination with the related confusion matrices, are presented and discussed in Chapter 3. Finally, Chapter 4 gives concluding remarks.

Chapter 2

Convolutional Neural Networks

2.1 Introduction

Convolutional Neural Networks have been one of the most influential innovations of the last decade in the field of Data Science and Computer Vision. It has been an efficient technique for pattern recognition and works in a remarkably effective way for image inputs. In fact, the term usually refers to a 2-dimensional CNN which is used for image classification. But there are two other types of Convolution Neural Networks used in the real world, which are 1-dimensional and 3-dimensional CNNs.

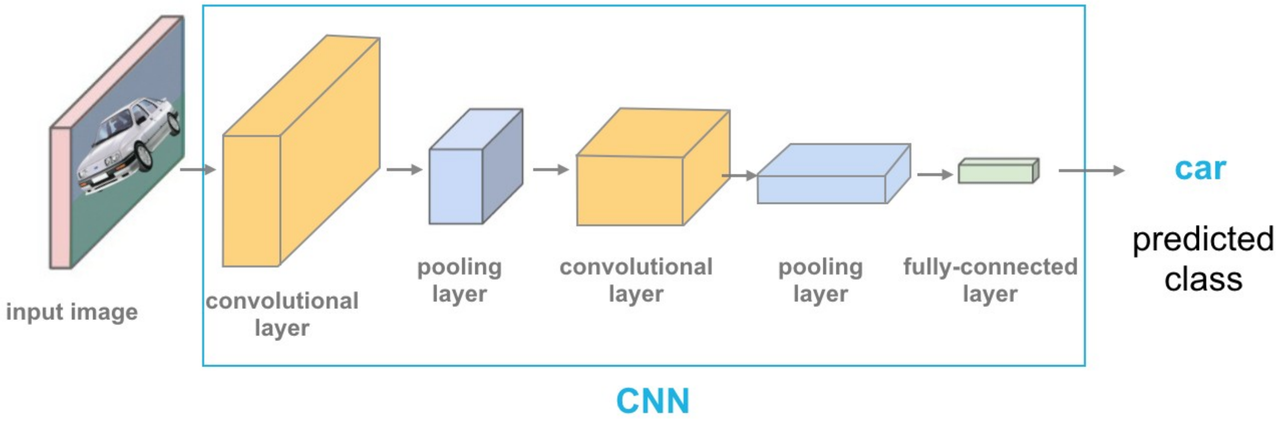


Fig. 4: The core of a Convolutional Neural Network for image input [3]

The difference between a simple DNN, as described in the introduction section, and a CNN lies in the hidden layers part. Convolutional layers are different from standard (dense) layers of canonical ANNs, and they have been invented to receive and process pixel data.

The input layer consists of the image we want to classify and the output layer is a fully connected layer that outputs the predicted class. Intermediate layers perform convolutional and pooling operations consecutively. Note, that such a model requires image data as input, while the data we possess are speech. In this case, one has to make the appropriate transformations in order to conform with this architecture.

2.2 Training and assessment

Respectively with the description of the training process in the introduction, which is similar for all the neural networks, the overall data are divided into a training and a test set. For the network training, 90% of the dataset is usually used, while the rest 10% is used for testing. This happens in a loop of 10 folds each one constituted of a 90-10 analogy of the data set. This method is called cross-validation and is explained in more detail in paragraph 2.2.5.
For the assessment, we usually consider the validation loss or validation accuracy throughout cross-validation, and of course, the test set, for whom the evaluation we shall choose a metric such as the accuracy (for balanced datasets) or f1-score (for imbalanced datasets).

2.2.1 Layers

A convolutional neural network normally consists of three major layers; **convolution** layer, **pooling** layer and a **fully connected** layer. The first two layers can be repeated depending on the depth of the network. The last one is usually placed at the end as a simple dense layer.

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number and the size of the **kernels**. Kernels are basically the filters that are applied on a small region of the image. The kernel size here refers to the width x height of the filter mask. Common choices for kernel size are 3x3 (below case) and 5x5. The values in the filter/kernel are called **weights**. Each weight determines how important the pixel is in forming the output image (feature map).

$$\overset{\text{(a)}}{\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 5 & 3 & 8 & 4 \\ 3 & 3 & 2 & 8 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix}} * \overset{\text{(b)}}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \overset{\text{(c)}}{\begin{bmatrix} 6 & -9 & -8 \\ -3 & -2 & -3 \\ -3 & 0 & -2 \end{bmatrix}}$$

The convolution operation above, between the original input matrix (a) and the filter (b) is applied successively upon 3x3 areas of the input matrix. By default, a filter starts at the upper left corner of the image with the left-hand side of the filter sitting on the far left pixels of the image. The filter is then stepped across the image one column at a time until the right-hand side of the filter is sitting on the far right pixels of the image. The "sliding" of the filter upon the matrix is called **stride**, and is usually equal to 1. When the stride is 2 or more (though this is rare in practice), then the filters move 2 or more pixels at a time. The result is the matrix (c) that consists the output "image" of the convolutional layer and is used as an input for the next layer. Actually no convolution is performed, but a cross-correlation. The following calculations are performed to produce the upper right digit of the output matrix (c):

$$7 \cdot 1 + 2 \cdot 0 + 3 \cdot (-1) + 4 \cdot 1 + 5 \cdot 0 + 3 \cdot (-1) + 3 \cdot 1 + 3 \cdot 0 + 2 \cdot (-1) = 6$$

The size of the output image is based on the formula: **(I – F + 2P)/S + 1**
Here in the example the input size (I) is 5, filter (F) is 3, padding (P) is 0, and stride(S) is 1. So the output size = (5 – 3 + 0) / 1 + 1 = 3 (matrix c).

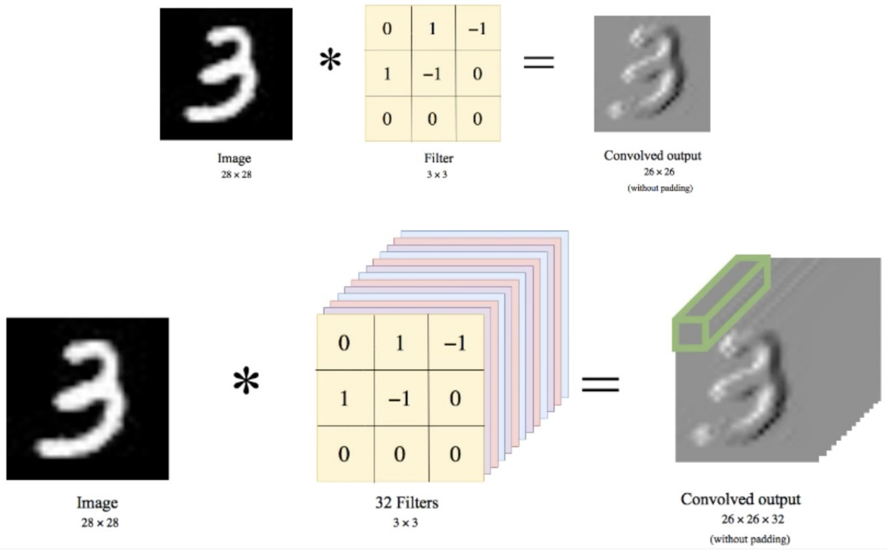
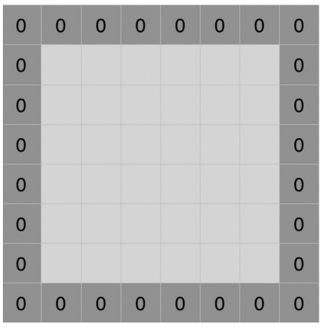


Fig. 5: simple convolution upon MNIST handwritten digit (top),
multiple convolutions (below) [4]

In order to extract the most important features through network training, we perform multiple convolutions, each using a different filter, which will result in the creation of many distinct **feature maps**. The latter ones are the convolved output matrices that contain information about the predominant features. We finally stack all these feature maps together and get the output of the convolution layer.

In realistic examples, however, there is a significant disadvantage. The corner pixels do not contribute as much in feature detection. Furthermore, while the filter is sliding over the input image it performs linear computation which can apparently reduce the dimensionality of the initial matrix in the process of extracting the important features. For this reason, the need arose to add some **padding** around the outline of the image, in order to keep the original dimensions throughout convolutions, and train the network with the most important features subsequently.



0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

Fig. 6: Zero padding added to image

It is usually useful to have a pad such that the size of the convolved output is same as the input size. So the size of the feature map is controlled by 3 parameters that we need to decide before the convolution step: depth (the number of filters we use), stride and zero-padding.

In the normalization step that follows, we apply the activation function; ReLU (Rectified Linear Unit) / Sigmoid. An image may have pixel values ranging from 0 – 255. However, neural networks work best with scaled “strength” values between 0 and 1. In practice the input image to a CNN is a grayscale image ranging in pixel values between 0 (black) and 1 (white). Converting an image from a pixel value range of 0 – 255 to a range of 0 – 1 is called **normalization**. In CNN the normalized input image is filtered and then a convolutional layer is created. Pixel values in the filtered image may fall into different ranges that may contain negative values as well, so to take care of this we apply the activation function. In CNN we often use ReLU (instead of Sigmoid) which simply turns negative pixel values to 0. Convolution is a linear operation -element wise matrix multiplication and addition- so we account for non-linearity by introducing a function like ReLU.

The main goal of the pooling layer, that follows the convolution layer, is dimensionality reduction, meaning reducing the size of an image by keeping a single value from a given window. The most common type of pooling layer is the **maxpooling** layer. This operation breaks an image into smaller patches, and returns the pixel with maximum value from a set of pixels within a patch, subsampling the input matrix. A maxpooling layer is defined by a patch size and stride. For a patch size of 2×2 and a stride of 2, this window will perfectly cover the image. A simple example of 4x4 input matrix is shown in Fig. 7.

$$\begin{bmatrix} 12 & 20 & 30 & 0 \\ 8 & 12 & 2 & 0 \\ 34 & 70 & 37 & 4 \\ 112 & 100 & 25 & 12 \end{bmatrix} \xrightarrow{\text{2x2 Max Pool}} \begin{bmatrix} 20 & 30 \\ 112 & 37 \end{bmatrix}$$

Fig. 7: Max Pooling

The last layer of CNN is the **fully connected** layer. Every output that is produced at the end of the last pooling layer is an input to each node in this fully connected layer. The role of the fully connected layer is to produce a list of class scores and perform classification based on image features that have been extracted by the previous convolutional and pooling layers. So, the last fully connected layer will have as many nodes as there are classes and the sum of output probabilities from the fully connected layer is 1.

2.2.2 Optimization methods

Deep learning is an iterative process. With so many parameters to tune and methods to try, it is important to be able to train models fast, in order to quickly complete the iterative cycle. This is key to increasing the speed and efficiency of a machine learning model. Hence the importance of optimization algorithms such as stochastic gradient descent, min-batch gradient descent, gradient descent with momentum and the Adam optimizer, among others. These methods make it possible for our neural network to learn, however, some of them perform better than others -for specific purposes- in terms of speed.

Essentially what we need to do is to compute the **loss**, which represents how poorly the model performs each time, and try to minimize it, because a lower loss means our model is going to perform better. The process of minimizing (or maximizing) any mathematical expression is called **optimization**.

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses and to provide the most accurate results possible. They solve optimization problems by minimizing the **cost function**. A cost function is a single value, not a vector, because it rates how good the neural network did as a whole with respect to the expected output.

Specifically, the latter one is a function with the following parameters:

$$C(W, B, S^r, E^r)$$

where W is our neural network's weights, B is our neural network's biases, S^r is the input of a single training sample, and E^r is the desired output of that training sample [11].

Initially, it is impossible to know exactly what our model's weights should be right from the start. But there are methods proposed in the literature on how to initialize a DNN [26]. All the optimization algorithms below can be used when training a machine learning model for minimization of the network's loss and configuration of its optimal parameters subsequently.

Gradient descent

Gradient descent is an optimization algorithm for finding a local minimum of a differentiable function. It is used to find the values of a function's parameters (coefficients) that minimize the loss/cost function as far as possible. The gradient descent method starts by defining the initial parameters' values and thereafter it utilizes calculus to iteratively adjust the values so they minimize the given cost function (Least Squares for instance). The weight is initialized using some initialization strategies and is updated with each epoch according to the update equation.

$$\text{Repeat until convergence:}$$

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

The above equation computes the gradient of the cost function J(θ) with respect to the parameters / weights θ_j for the entire training dataset. θ_j essentially represents the step size, α is the learning rate, and the partial derivative of the cost function above represents the slope.

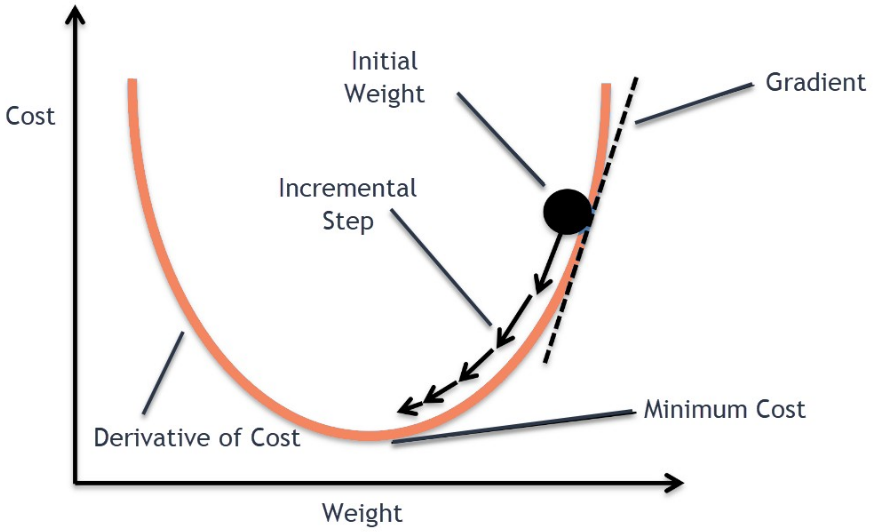


Fig. 8: Gradient descent for minimizing the cost function [12]

Our aim is to get to the bottom of our graph, or to a point where we can no longer move downhill – a local minimum.

The size of the steps gradient descent takes into the direction of the local minimum is determined by the **learning rate**, which figures out how fast or slow we will move towards the optimal weights. It is important to assign an appropriate value to the learning rate -neither too big nor too small- because if the steps it takes are too big, it may not reach the local minimum as it will bounce back and forth between the convex function of gradient descent (see left image below). If we set the learning rate to a very small value, gradient descent will eventually reach the local minimum but that may take a while (see right image below).

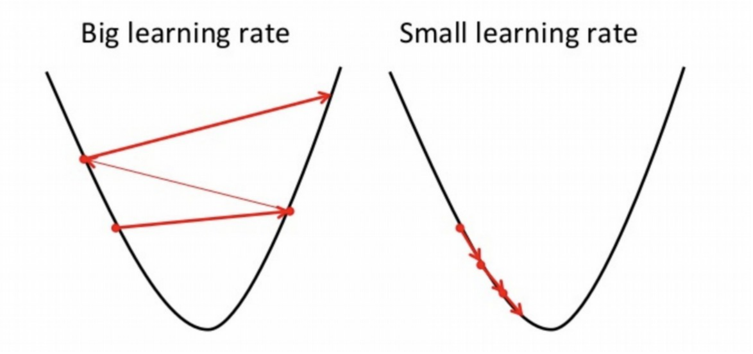


Fig. 9: The results of big learning rate (left) and small learning rate (right) throughout gradient descent [13]

Stochastic Gradient Descent (SGD)

SGD algorithm is an extension of the Gradient Descent and it overcomes some of the disadvantages of the GD algorithm. Gradient Descent has a disadvantage that it requires a lot of memory to load the entire dataset of n-points at a time to compute the derivative of the loss function. In the SGD algorithm derivative is computed taking **one point at a time**.

For each training example $x^{(i)}$ and label $y^{(i)}$, SGD performs a parameter update as follows:

$$\theta \leftarrow \theta - \alpha \frac{\partial}{\partial \theta}(J(\theta; x^{(i)}; y^{(i)}))$$

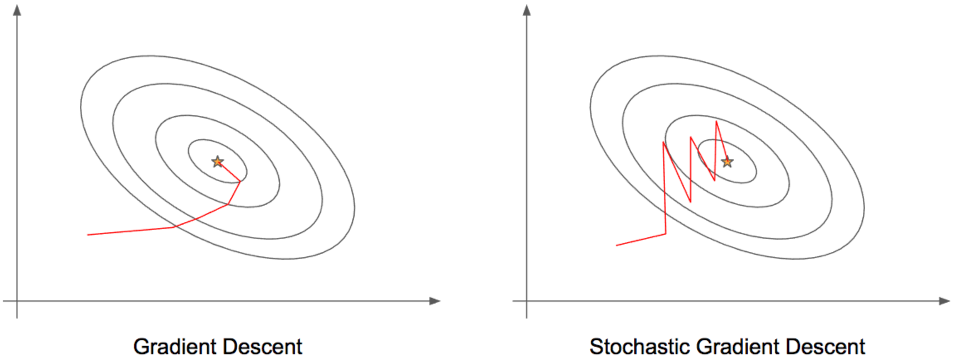


Fig. 10: GD vs SGD [14]
the star denotes a minimum of the cost

For each example above, we take a Gradient Descent step. On the left, we have Gradient Descent (1 step per entire training set) and on the right we have Stochastic Gradient Descent (1 point per step). SGD leads to many oscillations to reach convergence, but each step is a lot faster to compute for SGD than for GD, as it uses only one training sample (in contrast with the whole batch for GD). Typically, to get the best out of both we use Mini-batch gradient descent (MGD) which looks at a smaller number of training set examples at once.

Mini-batch gradient descent (MB-SGD)

MB-SGD algorithm is an extension of the SGD algorithm and it overcomes the problem of large time complexity in the case of the SGD. MB-SGD takes a batch of points or subset of points from the dataset to compute the derivate. For every mini-batch of n training examples it performs a following update:

$$\theta \leftarrow \theta - \alpha \frac{\partial}{\partial \theta}(J(\theta; x^{(i:i+n)}; y^{(i:i+n)}))$$

The update of weight is dependent on the derivate of loss for a batch of points. The updates in the case of MB-SGD are much noisy because the derivative is not always towards minima. MB-SGD divides the dataset into various batches and after every batch, the parameters are updated. This way, it reduces the variance of the parameter updates, which can lead to more stable convergence, and can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient.

Common mini-batch sizes range between 50 and 256, but can vary for different applications. Mini-batch gradient descent is typically the algorithm of choice when training a neural network.

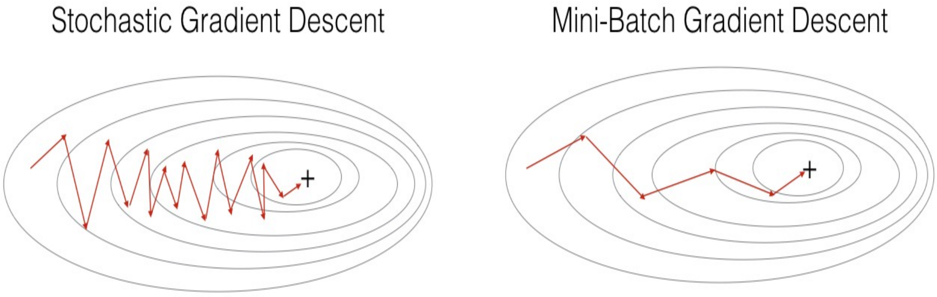


Fig. 11: SGD vs Mini-batch GD [15]

“+” denotes a minimum of the cost

Gradient descent with momentum

Gradient descent with momentum involves applying exponential smoothing to the computed gradient. This will speed up training, because the algorithm will oscillate less towards the minimum and it will take more steps towards the minimum.

The algorithm does this by adding a fraction γ (gamma) of the update vector of the past time step to the current update vector:

$$v_t \leftarrow \gamma v_{t-1} + \alpha \frac{\partial}{\partial \theta} J(\theta)$$

Now, the weights are updated by $\theta = \theta - \upsilon_t$. Note that some implementations exchange the signs in the equations.
The momentum term γ (usually set to 0.9 or a similar value) increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.

Adaptive Moment Estimation (Adam)

Adam [21] is another optimization method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients υ_t, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface [22]. We compute the decaying averages of past and past squared gradients m_t and υ_t respectively as follows:

$$\begin{aligned} m_t &\leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &\leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \end{aligned}$$

m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method.

2.2.3 Overfitting

Having such a large number of parameters inside a neural network has another drawback: **overfitting**. Overfitting is a phenomenon that occurs when a machine learning algorithm attaches too much to the training data provided and loses the ability to generalize. The neural network must learn different interpretations for something that is possibly the same. However, when the model trains for too long on sample data or when the model is too complex, it can start to learn the "noise," or irrelevant information, within the dataset. When the model memorizes the noise and fits too closely to the training set, the model becomes "overfitted," and it is unable to generalize well to new data. If that is the case, then it will not be able to perform the classification or prediction tasks that it was intended for.

Low error rates and a high variance are good indicators of overfitting. In order to prevent this type of behavior, part of the training dataset is typically set aside as the "test set" to check for overfitting. If the training data has a low error rate and the test data has a high error rate, it signals overfitting.

2.2.4 Dropout

It is quite common for a deep neural network to train for a significant time period, or train without enough data. A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to overfitting of the training data.

In machine learning, **regularization** is way to prevent overfitting. Regularization reduces this behavior by adding a penalty to the loss function. By adding this penalty, the model is trained such that it does not learn interdependent set of features weights. Dropout is a method that offers a computationally cheap and effective regularization method to reduce overfitting and improve generalization error in deep neural networks of all kinds, by randomly dropping out neurons during training (Fig. 11 below). Hence, it helps reducing interdependent learning amongst the neurons.

Specifically, during the training phase, for each hidden layer, for each training sample, for each iteration, dropout forces the network to "ignore" (zero out) a random fraction, *p*, of nodes and corresponding activations. During test phase, the network uses all activations, but reduces them by the same factor *p* (to account for the missing activations during training).

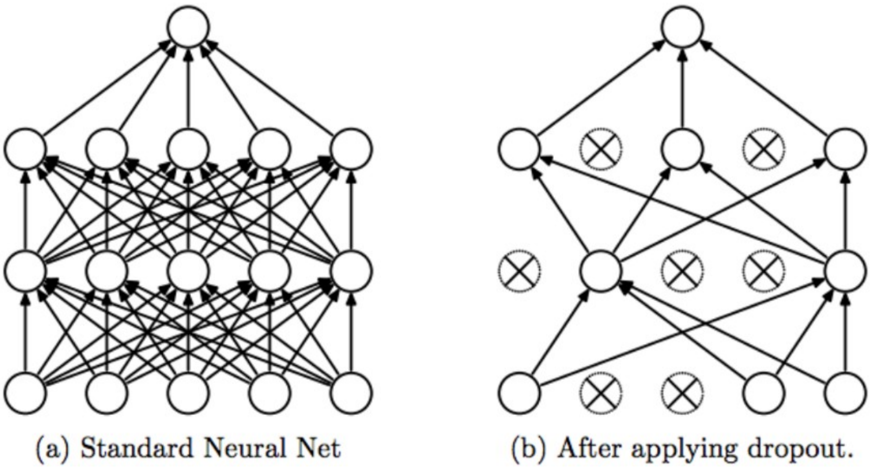


Fig. 12: Neural network before (a) and after
(b) applying dropout [27]

Implementing dropout is easy, if there is a fully connected layer at the end of the convolutional network. The Keras Dropout layer randomly sets input units to 0 with a frequency of *rate* at each step during training time. Parameter ‘rate’ is a float between 0 and 1 that describes the fraction of the input units to drop. ‘noise_shape’ is a 1-D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input and finally the ‘seed’ is a Python integer to use as random seed. The default interpretation of the dropout hyperparameter ‘rate’ is the probability of training a given node in a layer, where 1.0 means no dropout, and 0.0 means no outputs from the layer. A good value in general for dropout in a hidden layer has been found to be between 0.5 and 0.8.

2.2.5 Cross-validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data set. The procedure has a single parameter called k that refers to the number of groups that a given data set is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

It is a very popular technique as it is simple and, at the same time, it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split. In k-fold cross-validation, the dataset passes through the network as many times as k indicates, in a way that each time (among the k-folds) the test set used for the evaluation is different. Hence the network is trained multiple times upon different training and test data, which however belong to the same dataset.

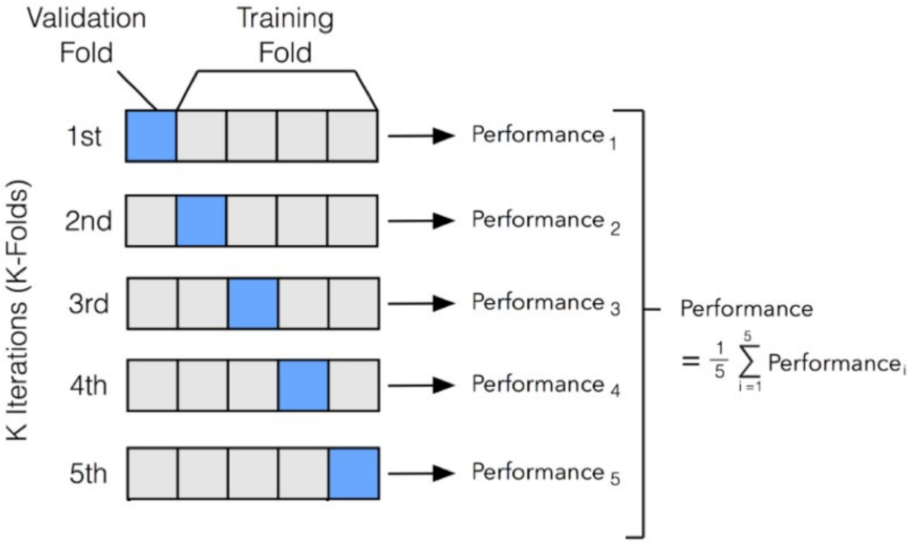


Fig. 13: k-fold cross-validation when k=5 [12]

In the example above, the blue squares constitute the evaluation part which, in this case, is the 1/5 of the dataset, and the rest of them are the training folds. As we can observe, at every row the evaluation set is different and the performance of every fold may vary. The average of the performances (k=5 overall) is calculated as the final estimated performance of the model.

Cross-validation is a useful technique when we don’t possess enough data for our training -which happens very often in practice- or we want to evaluate our model more objectively with low cost. There are many variations of the technique, like the Leave-one-out Cross-Validation (LOO), which as the title dictates leaves one sample out of the whole dataset to use for evaluation and uses the rest of them for training. Obviously, in this case, the number of folds equals the number of instances in the dataset. Nevertheless, the most popular and adequately efficient case is 10-fold cross-validation, that we also use in our speech emotion recognition model.

2.3 Sampling

In this paragraph, we are going to introduce topics of the speech sampling process, which is conducted in order to constitute the datasets we use. The emotion per se, is educed by the speech signal that has been sampled under the appropriate conditions. Usually, persons that express these records are specialized actors in order for the emotion to be as much as real and spontaneous. These records are stored on a database so that we can use them to create subsequently spectrogram images (the actual input of the network).

According to a research [23], acted speech from professionals is the most reliable because professionals are trained to color speech by emotions and such emotions have a great amplitude or strength. Nevertheless, when the emotion is acted, it cannot be absolutely correlated to real conditions. The best practice is to get our samples from real situations, so we can work with datasets that include spontaneous records of speech, which yet has privacy constraints of personal data. In fact, the most important finding so far is the lack of data for research on spontaneous/real-life speech, both in terms of data collections and features.

On the other hand comes the trade-off of emotional subjectivity; assigning one emotion to one situation seems fluid problem because as mentioned, emotions are subjective and secondly, in natural situations emotions can be contrasted [1]. In any case, by using a plurality of data bases we hope to achieve unbiased results as far as possible.

2.4 Preprocessing

Since our research is on speech and the emotion that frames it, the initial signals are converted for the needs of feeding the CNN, so the final samples that are given to the network are in image format. More specifically, our data are imprinted on the visual representation of the spectrum of frequencies of each speech signal as it varies with time; that is a special type of representation that is called **spectrogram**. Spectrograms are turned into images in order to feed the neural network.

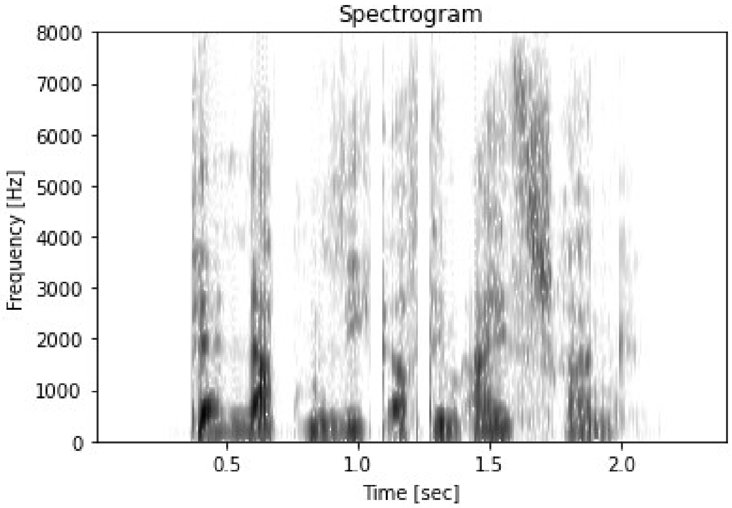


Fig. 14: Wide-band spectrogram from the Berlin dataset in gray scale.
Emotion: anxiety/fear

The spectrum of speech reveals information on the formants, that are one of the quantitative characteristics of the vocal tract. This is the reason we used wideband spectrograms instead of narrowband, as we can educe more information related to the vocal cords. Narrowband spectrograms on the other hand, provide information about the individual harmonics of the voice source.

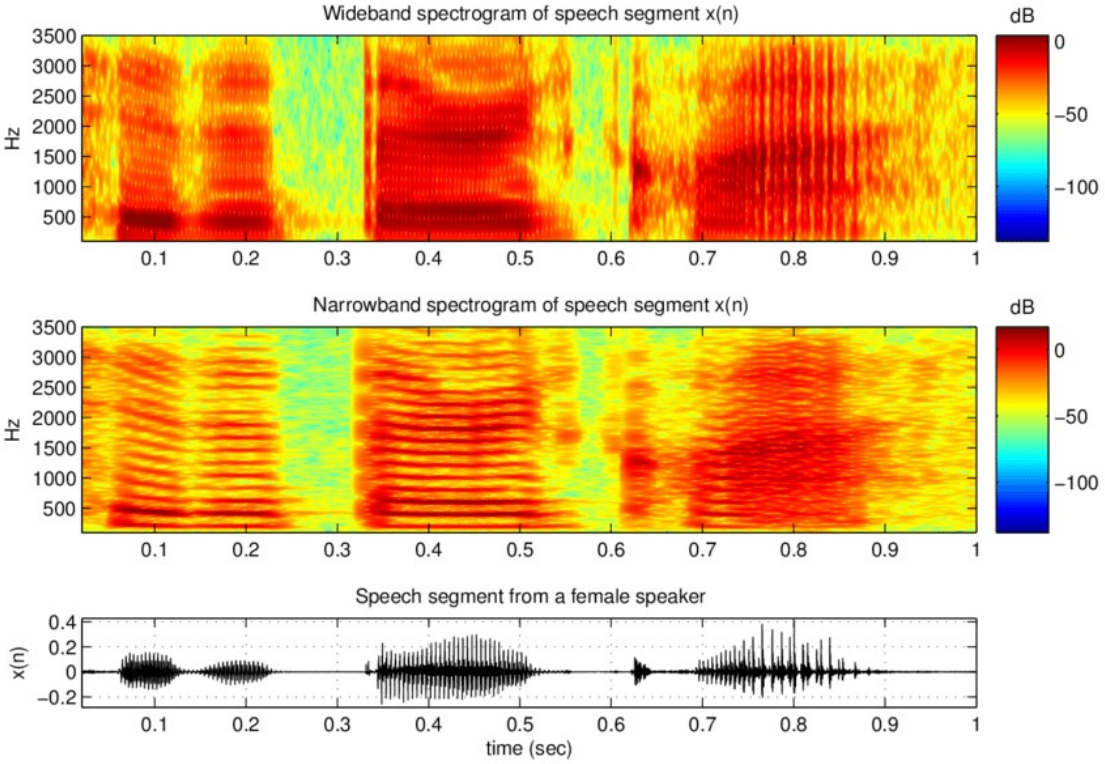


Fig. 15: A wide-band (top) and narrow-band (middle) spectrograms are shown along with the speech signal from a female speaker (bottom), used to derive them. The spoken sentence is: "to the third class" [20]

The spectrogram of Fig. 14 is an example of the images we use for our training, however we implement some transformations before sending them to the model. The final resolution of our images is 129x129 pixels which is a relatively low resolution resulting from data compression. We also transform the spectrogram to the range 0-4 kHz and apply z-normalization at the end. So, for instance, the spectrogram of Fig. 14 after the preprocessing procedure is transformed as shown in Fig. 16.

By the way, after each layer the initial image is reduced and resized, it changes according to the DNN specifications. Namely, convolutional layers sequentially downsample the spatial resolution of images while expanding the depth of their feature maps. This series of convolutional transformations can create much lower-dimensional and more useful representations of images than what could possibly be hand-crafted. So what we get in the first layer with 129x129 pixel neurons undergoes extensive processing throughout the next DL procedure (training).

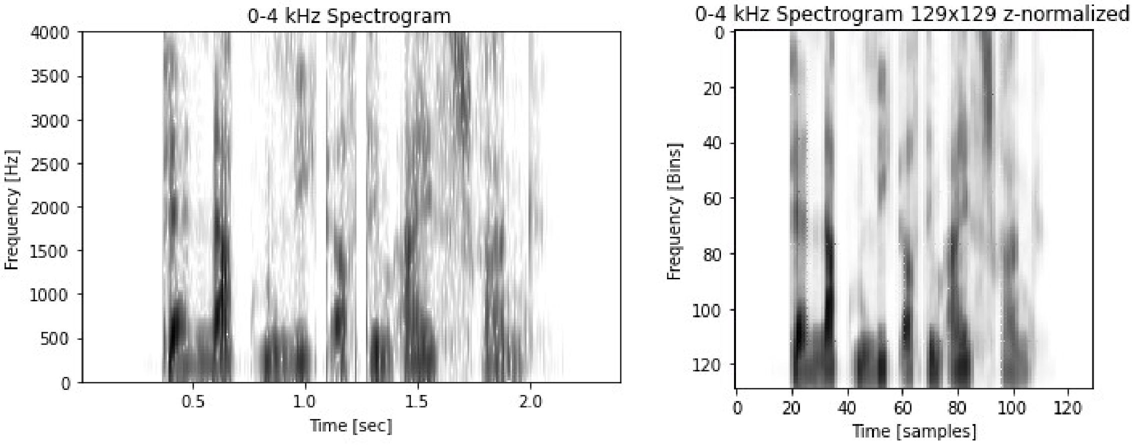


Fig. 16: Spectrogram after reducing length (left),
and after resizing and z-normalization (right)

2.5 Data Augmentation & Performance

The broader idea of Data Augmentation encompasses a suite of techniques that enhance the size and quality of training datasets such that better Deep Learning models can be built using them. It consists a data-space solution to the problem of limited data, which is evident for many data bases we used, and the problem of overfitting that also leads to misclassification all along the model evaluation.

Some of the data bases we used do not contain the number of samples we need to have representative results, so we used data augmentation with adding small noise to the signals, without changing other speech features. Instead of performing image augmentation, we augmented the original speech signals by including white Gaussian noise of 15dB SNR (at a given Signal-to-Noise Ratio). This way, the initial set of samples augmented to 2x and then to 10x for more extensive and unbiased training.

To avoid misclassifications caused by silence injection at the beginning and at the end of the speech signal, after the augmentation of the dataset by adding noise, we used a Voiced-Unvoiced-Silence detector (VUS) in MATLAB that calculates the energy of each 30 ms frame of the signal and eliminates the silent frames, and the total duration of the audio by extension. This step was especially useful for the RAVDESS data base, whose original samples include signigicantly long silences at the beginning and at the end of the signal. Before silence detection, it seemed that the network was learning the noise we injected through augmentation.

What we need is to train our model with minimum memory and time resources as well as achieve simplicity. Performance may vary from one dataset to another. Also the increase of the number of epochs optimizes the results. However, our focus is to keep the network simple without adding extra layers and also keep the number of epochs stable to 100. Data augmentation manages to “fake” a bigger dataset, although it increases a lot training time.

2.6 Model Architecture

The network we propose is being trained to recognize emotions, providing a classifier algorithm of a CNN with two convolutional layers. After each convolution follows the max pooling layer as well as padding. This network is called 2-dimensional CNN because the kernel slides along 2 dimensions on the data. The activation function we use after the convolution is ReLU. Filter sizes may vary; they are 5x5 matrices but for some data sets we observe that 10x10 filters on the first layer perform as well decently. The base architecture with the corresponding parameter values is shown in the flowchart below. The value of dropout is set to 0.2, however, the bigger the augmentation we had, the bigger the dropout we used so it reached as well the value of 0.5 for the ten-fold datasets.

We implement the model using the keras library with our modifications. All experiments are run on a single GPU. All the models are trained/tested based on the same implementation as shown in Fig. 17. This simple architecture consists the core of our network and achieves the results listed in Chapter 3.

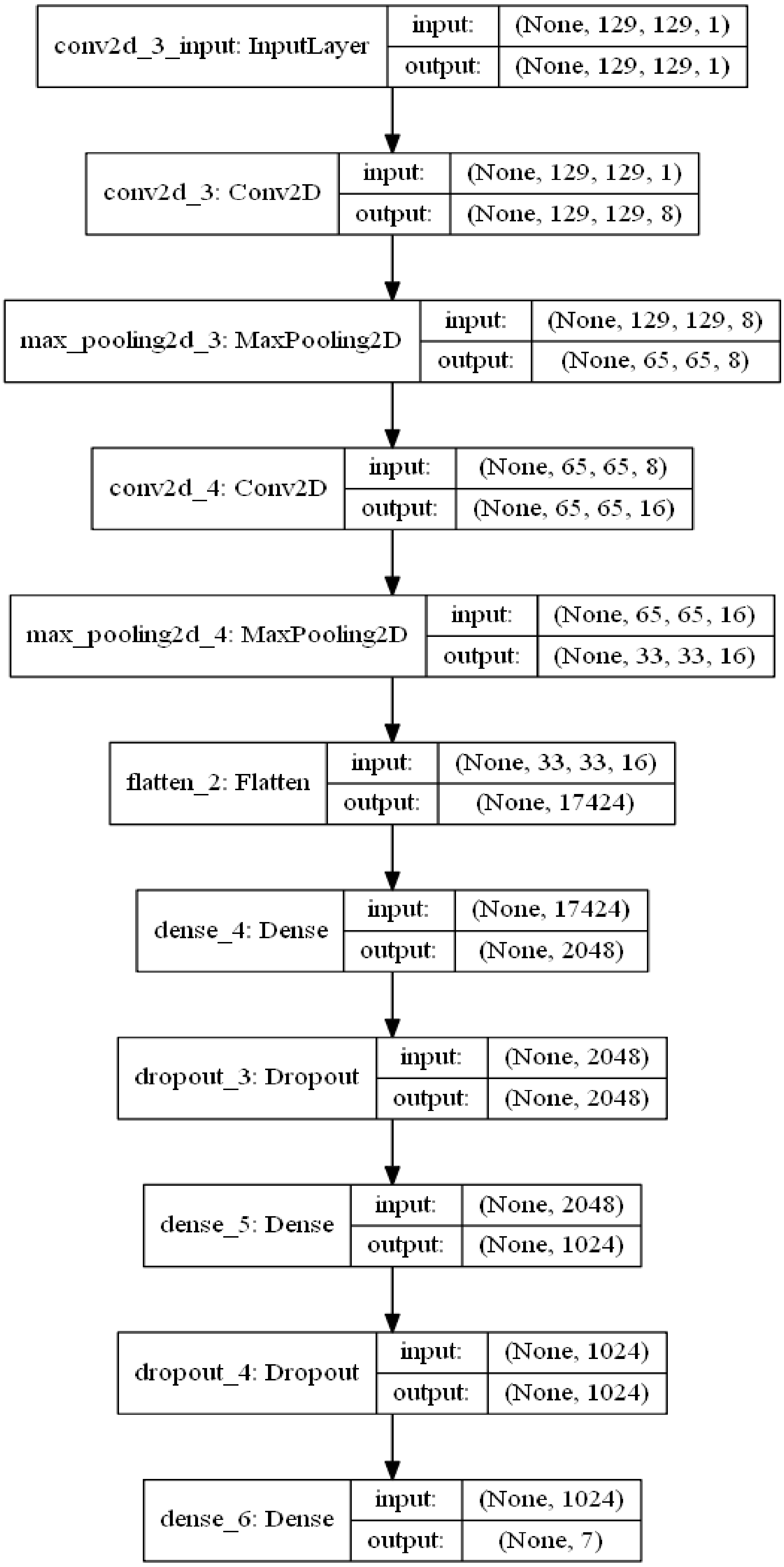


Fig. 17: Convolutional Neural Network architecture flowchart

Chapter 3

Experiments and Results

In this section we will review some measurements of the model’s performance and make relative conclusions regarding the results. Berlin Emotional Speech (BES) is the first dataset that we worked with and thereafter we also turned to other datasets in order to validate the model’s performance more thoroughly.

The results of each dataset for different runs of the model are shown below. Accuracy results and confusion matrices of the latest runs are listed at the end of the chapter for each dataset.

Berlin				
Filter Size 1	Filter Size 2	Dropout	Augmentation	Accuracy (mean - std)
5x5	5x5	0.2	no	57.570 +-4.949
5x5	5x5	0.3	x2	77.888 +-4.884
5x5	5x5	0.5	x10	87.720 +-4.709
10x10	5x5	0.2	no	58.224 +-4.117
10x10	5x5	0.3	x2	77.944 +-6.554
10x10	5x5	0.5	x10	87.458 +-7.852

AESI				
Filter Size 1	Filter Size 2	Dropout	Augmentation	Accuracy (mean - std)
5x5	5x5	0.2	no	80.714 +-1.498
5x5	5x5	0.3	x2	84.511 +-10.649
5x5	5x5	0.5	x10	91.595 +-3.565

10x10	5x5	0.2	no	79.571 +-2.478
10x10	5x5	0.3	x2	88.649 +-2.711
10x10	5x5	0.5	x10	92.529 +-2.983

SAVEE				
Filter Size 1	Filter Size 2	Dropout	Augmentation	Accuracy (mean - std)
5x5	5x5	0.2	no	42.604 +-2.612
5x5	5x5	0.3	x2	49.542 +-6.831
5x5	5x5	0.5	x10	68.771 +-5.653
10x10	5x5	0.2	no	45.521 +-2.836
10x10	5x5	0.3	x2	53.667 +-8.756
10x10	5x5	0.5	x10	70.625 +-4.934

TESS				
Filter Size 1	Filter Size 2	Dropout	Augmentation	Accuracy (mean - std)
5x5	5x5	0.2	no	99.464 +-0.505
5x5	5x5	0.3	x2	88.450 +-3.434
10x10	10x10	0.2	no	97.982 +-2.606
10x10	10x10	0.3	x2	67.425 +-35.368
10x10	5x5	0.2	no	99.018 +-0.918
10x10	5x5	0.3	x2	87.614 +-4.426

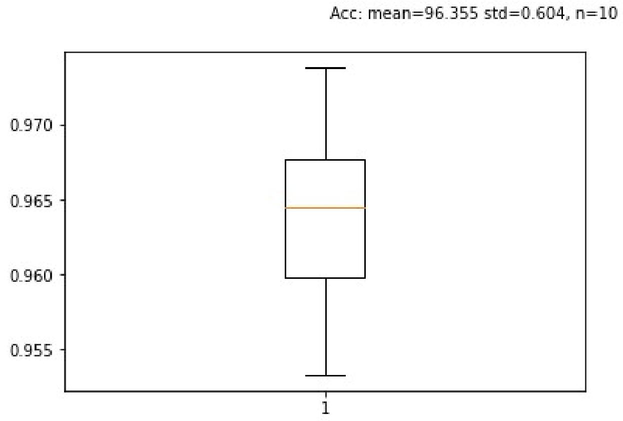
CaFE				
Filter Size 1	Filter Size 2	Dropout	Augmentation	Accuracy (mean - std)
5x5	5x5	0.2	no	25.426 +-3.790
5x5	5x5	0.3	x2	65.256 +-9.469
5x5	5x5	0.5	x10	71.100 +-9.157
10x10	5x5	0.2	no	25.000 +-2.540
10x10	5x5	0.3	x2	58.739 +-7.121
10x10	5x5	0.5	x10	73.098 +-6.482

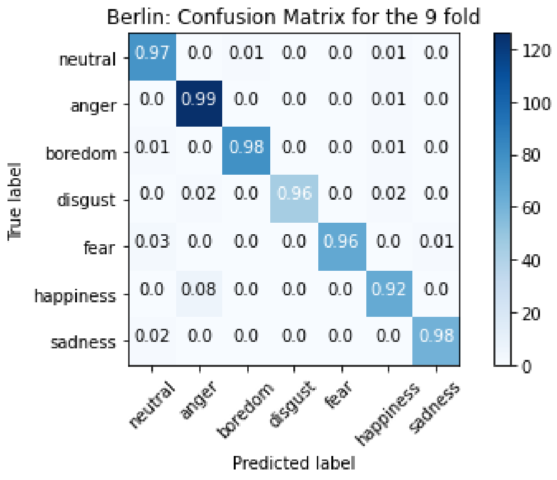
RAVDESS				
Filter Size 1	Filter Size 2	Dropout	Augmentation	Accuracy (mean - std)
5x5	5x5	0.2	no	63.646 +-2.914
5x5	5x5	0.3	x2	59.160 +-4.868
5x5	5x5	0.5	x10	61.264 +-6.213
10x10	5x5	0.2	no	66.285 +-2.367
10x10	5x5	0.3	x2	55.014 +-6.366
10x10	5x5	0.5	x10	59.611 +-5.470

As we can observe our learning method has a different effect on each dataset, but still works better for datasets with higher amount of samples. This verifies that neural networks achieve higher classification accuracy as the given data increases. TESS that achieves the best results, is the biggest dataset between all of them, and additionally, its speech signals are similar except a few words -usually- at the end of the phrase. What actually changes between phrases of the same linguistic content is the prosody of the voice, which is the essential substance of the emotion.

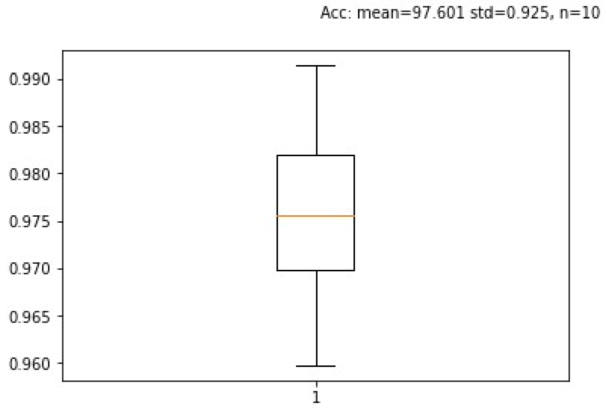
Another factor that must not be neglected is that the number of emotions in the data collections is not the same for all of them. It varies from 5 to 8, and -as known- the more the choices, the less the possibility of choosing the right one incidentally. AESI contains the least emotional states. On the other hand, RAVDESS contains the most.

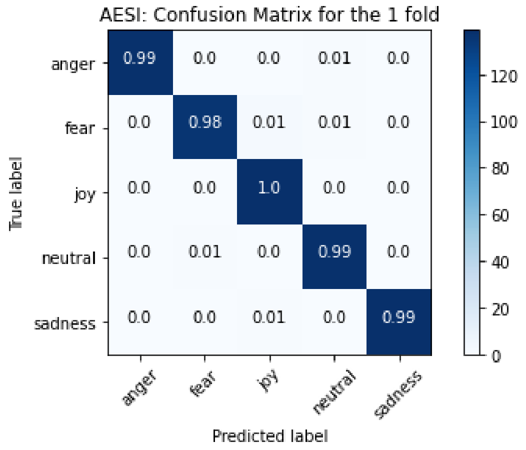
Accuracy results and Confusion Matrices:



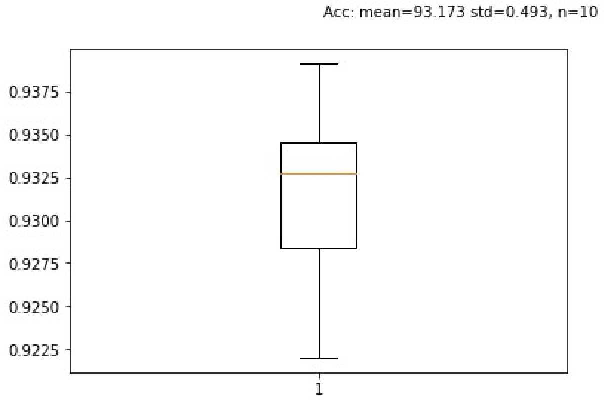


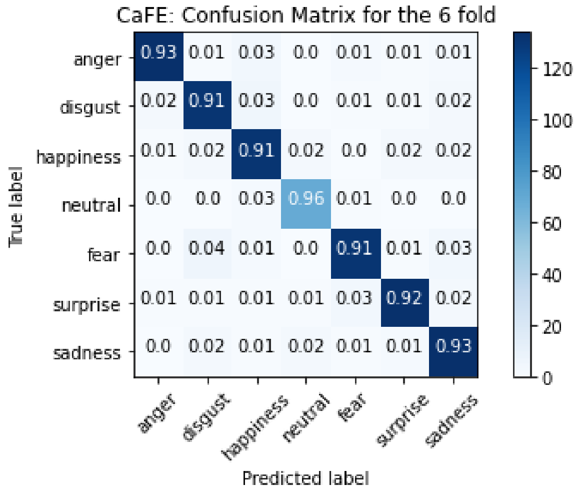
Berlin: mean accuracy and standard deviation



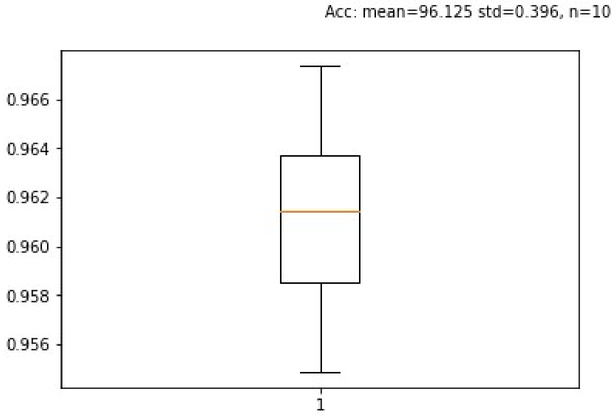


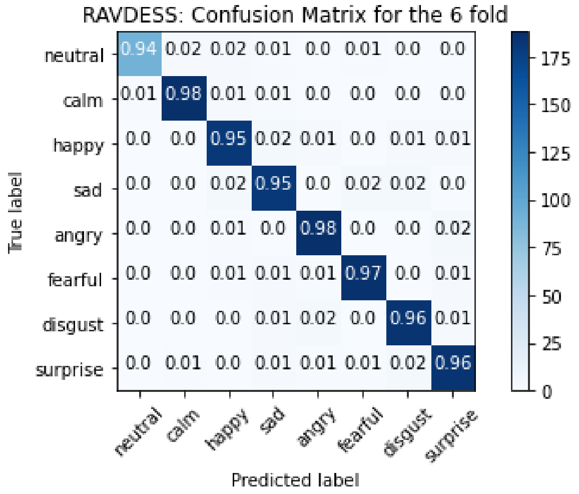
AESI: mean accuracy and standard deviation



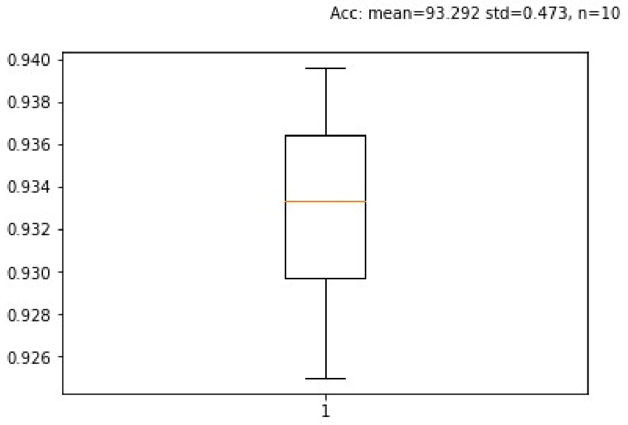


CaFE: mean accuracy and standard deviation

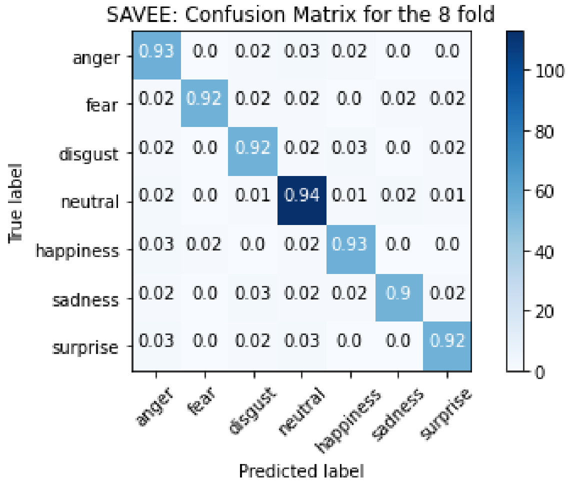




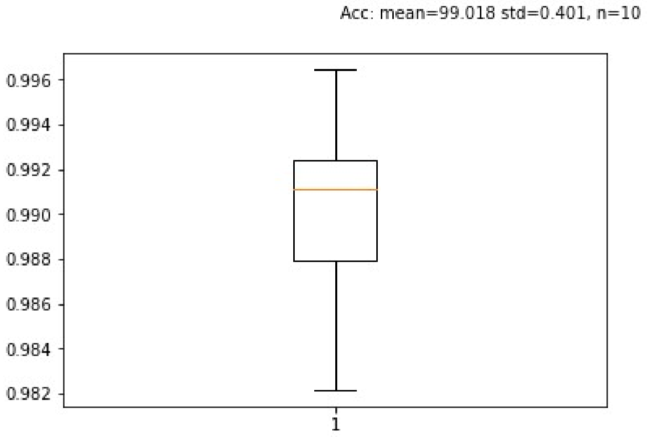
RAVDESS: mean accuracy and standard deviation



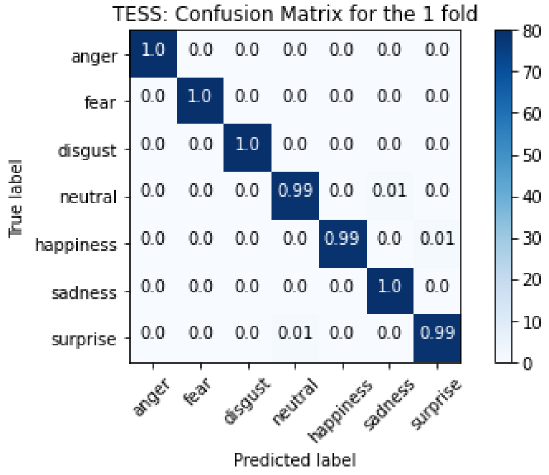
SAVEE: mean accuracy and standard deviation



	anger	fear	disgust	neutral	happiness	sadness	surprise
anger	0.93	0.0	0.02	0.03	0.02	0.0	0.0
fear	0.02	0.92	0.02	0.02	0.0	0.02	0.02
disgust	0.02	0.0	0.92	0.02	0.03	0.0	0.02
neutral	0.02	0.0	0.01	0.94	0.01	0.02	0.01
happiness	0.03	0.02	0.0	0.02	0.93	0.0	0.0
sadness	0.02	0.0	0.03	0.02	0.02	0.9	0.02
surprise	0.03	0.0	0.02	0.03	0.0	0.0	0.92



TESS: mean accuracy and standard deviation



	anger	fear	disgust	neutral	happiness	sadness	surprise
anger	1.0	0.0	0.0	0.0	0.0	0.0	0.0
fear	0.0	1.0	0.0	0.0	0.0	0.0	0.0
disgust	0.0	0.0	1.0	0.0	0.0	0.0	0.0
neutral	0.0	0.0	0.0	0.99	0.0	0.01	0.0
happiness	0.0	0.0	0.0	0.0	0.99	0.0	0.01
sadness	0.0	0.0	0.0	0.0	0.0	1.0	0.0
surprise	0.0	0.0	0.0	0.01	0.0	0.0	0.99

Chapter 4

Conclusion and Future Work

The computational paralinguistic task of recognizing the emotion states from human speech is indeed a challenging problem. Namely, it is not about a classic programming or mathematical problem that is solved with structural logic, instead it is about something that requires a learning process. A non-negligible fact is that the human emotions hide a factor of subjectivity and also increased complexity. In this case, the field of Artificial Intelligence and Deep Learning shall be implicated, as the ever evolving machines may be capable to spot patterns that even a human individual could not recognize.

In the present thesis, we started by introducing the human speech and the speech production system. Our research does not focus on characteristics of plain voice but on structured speech, which in fact contains emotional payload. Hence, we proposed a methodology for the recognition of speech emotion using Convolutional Neural Networks. Specifically, we experimented with simple end-to-end architectures within the Keras framework along with data augmentation. We started by explaining the whole concept of Deep Neural Networks, as well as the most commonly used, already existing Machine Learning techniques proposed in bibliography, and ended up by our unique model implementation, functionality, trade-offs and optimization elements. By explaining the training and evaluation procedures, the reader of this report shall be capable to understand and evaluate the final results of the experimental process. The network achieves indeed a state-of-the-art accuracy in speech emotion recognition using different datasets for testing and a cross-validation practice that creates a reliable and unbiased surface.

This work can be utilized in the broad area of Human-Computer Interaction, in Virtual Reality (VR) environments, Telecommunication systems and many more multimedia applications. As a thesis subject, it shall also trigger future academic study upon a multi-language dataset. So far, we evaluated our model using one single dataset -with its own characteristics- each time. The next step would be the combination of these emotional datasets (or parts of them) and, moreover, to try some different ways in augmenting the data. As an ultimate goal we set the pushing of those architectures to their limits, in order to observe their potential behavior and achieve higher performances.

References

[1] Sierro Hervé, "Extracting emotions from speech signal: State of the art" - Seminar Paper, University of Friburg, 2012

[2] Dimitrios Ververidis and Constantine Kotropoulos, "Emotional speech recognition: Resources, features, and methods", published by Elsevier B.V., Sept. 2006

[3] Moataz El Ayadi, Mohamed S. Kamel, Fakhri Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases", Pattern Recognition, Volume 44, Issue 3, 2011.

[4] Schuller B., Batliner A., Steidl S., Seppi D. "Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge.", Speech Communication, 53(9-10):1062–1087, Feb. 2011

[5] Mitngyu You, Chun Chenr, Jiajun Bul, Jia Liul, Jianhua Tao, "A Hierarchical Framework for Speech Emotion Recognition", IEEE international symposium on industrial electronics, Montreal CA, 9-13 July, 2006.

[6] Dimitrios Ververidis, Constantine Kotropoulos and Ioannis Pitas, "Automatic Emotional Speech Classification", IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal CA, 17-21, May 2004.

[7] Yashpalsing Chavhan, M. L. Dhore and Pallavi Yesaware, "Speech Emotion Recognition Using Support Vector Machines", International Journal of Computer Applications, Feb. 2010

[8] R. Duda, P. Hart, D. Stork, "Pattern Classification", John Wiley and Sons, Jan. 2001

[9] Rahul B. Lanjewar, Swarup Mathurkar and Nilesh Patel, "Implementation and Comparison of Speech Emotion Recognition System Using Gaussian Mixture Model (GMM) and K-Nearest Neighbor (K-NN) Techniques", published by Elsevier B.V., 2015

[10] Mudasser Iqbal, Syed Ali Raza, Muhammad Abid, Furqan Majeed and Ans Ali Hussain, "Artificial Neural Network based Emotion Classification and Recognition from Speech", International Journal of Advanced Computer Science and Applications. 11, 2020.

[11] S.A. Firoz, S.A. Raji and A.P. Babu, "Automatic Emotion Recognition from Speech Using Artificial Neural Networks with Gender-Dependent Databases", International Conference on Advances in Computing, Control, and Telecommunication Technologies, Dec. 2009

[12] Jan Rybka and Artur Janicki, "Comparison of speaker dependent and speaker independent emotion recognition", International Journal of Applied Mathematics and Computer Science, Dec. 2013

[13] F. Dellaert, T. Polzin, and A. Waibel, "Recognizing emotion in speech" in Proc. ICSLP, Philadelphia, PA, 1996, pp. 1970 – 1973

[14] V. Petrushin, "Emotion in speech: Recognition and application to call centers", Artificial Neural Net. Engr. (ANNIE), 1999

[15] Md. Kamruzzaman Sarker, Kazi Md. Rokibul Alam and Md. Arifuzzaman, "Emotion Recognition from Human Speech: Emphasizing on Relevant Feature Selection and Majority Voting Technique", Khulna University of Engineering and Technology, 2018

[16] Ivan J. Tashev, Zhong-Qiu Wang and Keith Godin, "Speech emotion recognition based on Gaussian Mixture Models and Deep Neural Networks", IEEE, 2017 Information Theory and Applications Workshop (ITA), Feb. 2017

[17] Yi-Lin Lin and Gang Wei, "Speech emotion recognition based on HMM and SVM", IEEE, 2005 International Conference on Machine Learning and Cybernetics, pp. 4898-4901, Vol. 8, Aug. 2005

[18] Ajit P. Gosavi and S.R. Khot, "Emotion recognition using Principal Component Analysis with Singular Value Decomposition", IEEE, 2014 International Conference on Electronics and Communication Systems (ICECS), Feb. 2014

[19] J. Mao, Y. He and Z. Liu, "Speech Emotion Recognition Based on Linear Discriminant Analysis and Support Vector Machine Decision Tree," *2018 37th Chinese Control Conference (CCC)*, pp. 5529-5533, July 2018

[20] Chavhan, Yashpalsing & Dhore, Manikrao & Pallavi, Yesaware, "Speech Emotion Recognition Using Support Vector Machines", International Journal of Computer Applications, Feb. 2010

[21] Kingma, D. P., & Ba, J. L. (2015), "Adam: a Method for Stochastic Optimization", International Conference on Learning Representations, 1–13

[22] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017), "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." in Advances in Neural Information Processing Systems 30 (NIPS 2017)

[23] Lech Margaret, Stolar Melissa, Best Christopher, Bolia Robert, "Real-Time Speech Emotion Recognition Using a Pre-trained Image Classification Network: Effects of Bandwidth Reduction and Companding", Frontiers in Computer Science, 2020

[24] Namrata Dave, "Feature extraction methods LPC, PLP and MFCC in speech recognition", International Journal For Advance Research in Engineering And Technology, Jul 2013

[25] Paul Covington, Jay Adams, Emre Sargin, "Deep Neural Networks for YouTube Recommendations", ACM Conference on Recommender Systems, Sep 2016

[26] Wadii Boulila, Maha Driss, Mohamed Al-Sarem, Faisal Saeed, Moez Krichen, "Weight Initialization Techniques for Deep Learning Algorithms in Remote Sensing: Recent Trends and Future Perspectives", Feb 2021

[27] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Web sources & mores

[1] Reproduced by Joe Wolfe, BSc Qld, BA UNSW, PhD ANU, School of Physics, The University of New South Wales, Sydney

[2] https://www.mdpi.com/2073-4433/10/11/718/htm

[3] https://cezannec.github.io/Convolutional_Neural_Networks/

[4]https://becominghuman.ai/from-human-vision-to-computer-vision-convolutional-neural-network-part3-4-24b55ffa7045

[5] https://www.topcoder.com/blog/convolutional-neural-networks-in-pytorch/

[6] https://robotics.ntua.gr/aesi-dataset/

[7] http://kahlan.eps.surrey.ac.uk/savee/

[8] https://tspace.library.utoronto.ca/handle/1807/24487

[9] https://zenodo.org/record/1478765#.YTln8PwzY5k

[10] https://smartlaboratory.org/ravdess/

[11] http://neuralnetworksanddeeplearning.com/

[12]https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1

[13] https://www.educative.io/edpresso/learning-rate-in-machine-learning

[14] https://pythonmachinelearning.pro/complete-guide-to-deep-neural-networks-part-2/

[15] https://datascience-enthusiast.com/DL/Optimization_methods.html

[16] https://keras.io/api/layers/regularization_layers/dropout/

[17]https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

[18] https://medium.com/analytics-vidhya/cross-validation-techniques-bacb582097bc

[19]https://www.researchgate.net/figure/A-wide-band-top-and-narrow-band-middle-spectrograms-are-shown-along-with-the-speech_fig22_242685014