Spoken Language Recognition

Gal Koaz, Gabriel Dunaevsky, Idan Kaminetsky, Or Haim Anidjar

School of Computer Science, Ariel University, Israel

`koazgal@gmail.com gabidunaevsky10@gmail.com idankam98@gmail.com`

April 2023

1. Abstract

In this paper, we took an existing language recognition model and tried to improve it with several methods. This model applies x-vectors to the task of spoken language recognition. Long-term language characteristics are captured in the network by a temporal pooling layer that aggregates information across time.

our work utilized data sets sourced from Common-Voice, a valuable resource for multilingual speech data. We specifically focused on ten languages, carefully selected from three distinct language families: Indo-European, Semitic, and East Asian.

The main method we used to improve the existing model is changing the architecture of the TDNN layers. The first significant change is the addition of 1X1 TDNN intermediate layers between the existing layers. Another change is in the structure of the TDNN to a funnel structure. In addition, a grid search was performed to find the optimal dilation and context size values in the TDNN layers.

These optimal values help to find the complex patterns that represent the different languages in the audio segments. After significant training on the original data, additional training was performed for the purpose of fine-tuning the model on data with augmentations.

On our final model we achieved an impressive 0.969 accuracy on the validation data set. After the final model was decided, Tested on the test data set and received a score of 0.7 accuracy. This score is significantly higher than the baseline model (0.54), but nevertheless shows a clear OVERFITTING of the final model, a problem that must be solved in follow-up studies.

2. Introduction

Idan:

The history of spoken language recognition (SLR) may be traced back to early rule-based systems, which were followed by statistical and machine learning-based approaches in the 1980s and 1990s, respectively. In recent years, advanced statistical models like GMMs paired with acoustic characteristics, as well as deep learning-based approaches such as CNNs and LSTM networks (To read more on CNN and LSTM [1, 2]), have been successfully applied to SLR [3]. Speech-to-text transcription, language identification in contact centers, and security are all applications of SLR. SLR systems are likely to grow more accurate and robust in the future as technology advances, allowing for increased cross-cultural communication.

Gal:

Spoken language recognition is an exciting and vital topic that is continually evolving due to the advancement of modern technologies. Identifying the language spoken has become a vital tool for communication between diverse individuals in an era where communication crosses the borders of countries and cultures as we can learn from here [4]. Spoken language recognition research focuses on the development of technology and systems that allow for the automatic and more accurate identification, analysis, and understanding of spoken language. The research's major purpose is to assist people and systems in communicating with a speaking entity in a language they do not understand, as well as to increase communication and efficiency between people and systems. The creation of advanced algorithms and models, such as the recognition of the manner of speaking and auxiliary effects in speech, as well as the production of databases and their usage to run more accurate language recognition systems, may be included in the research techniques.

Idan:

Although spoken language recognition has made great progress over the years, there are still several hurdles [5] that must be overcome in order to produce accurate and dependable findings. Background noise is one of the most serious issues, as it can considerably reduce the accuracy of spoken language recognition systems. Noise can come from a multitude of sources, including traffic, wind, and other external variables, as well as other speakers in the same room or nearby. Variations in dialects and accents are another problem. People frequently talk with varied accents and dialects, even within the same language, making it challenging for voice recognition algorithms to reliably identify the spoken language. This is especially difficult in countries with a significant degree of linguistic diversity, such as India, where different regions have their own distinct dialects and accents.

Idan:

Another issue is a lack of training data. To learn and increase their accuracy over time, speech recognition algorithms rely substantially on training data. Obtaining huge volumes of high-quality training data, on the other hand, might be problematic, especially for languages with fewer speakers or dialects that are not commonly spoken. As a result, models may be under-trained, resulting in poor accuracy and reliability. Furthermore, spoken language recognition systems may have difficulty identifying spoken language in real-time, especially when dealing with spontaneous speech, such as in genuine conversation. The necessity for fast and efficient algorithms that can handle massive amounts of data in real-time makes real-time processing difficult. Overall, these difficulties in spoken language recognition underscore the importance of ongoing research and development in this sector in order to enhance the accuracy and reliability of these systems. To address these problems and increase the performance of spoken language recognition systems, further technological breakthroughs, such as improved algorithms and data collection methods, will be required.

Gal:

Speaker embedding [6] is a critical problem in Spoken Language Recognition (SLR) that includes extracting compact and discriminative representations from voice data. The most preva-

lent speaker embedding techniques are x-vector, i-vector [7], and d-vector [8]. The x-vector approach uses a deep neural network (DNN) to extract a variable-length speaker embedding from a speech signal. The DNN is trained on a large amount of speaker-labeled data and is typically composed of convolutional and recurrent layers to capture both short-term and long-term information from the speech signal. The resulting x-vector is a low-dimensional representation that can be used for speaker verification, speaker identification, and language identification tasks as we read here [9].

Gal:

The i-vector approach uses a factor analysis model to extract a fixed-length speaker embedding from a speech signal. The i-vector is typically extracted from a low-dimensional representation of the speech signal using a probabilistic model called a Gaussian mixture model-universal background model (GMM-UBM), for more information [10]. The i-vector captures the speaker-specific information from the speech signal and has been widely used in speaker verification tasks. However, it is less commonly used in speaker identification and language identification tasks due to its fixed-length representation. The d-vector approach is similar to the x-vector approach but is trained with a larger amount of data and a longer sequence of speech signals. The d-vector is typically extracted from a deep neural network that is trained on large-scale speaker-labeled data, and it aims to capture the speaker-specific information from the entire speech segment. The resulting d-vector is a low-dimensional representation that can be used for various SLR tasks, including speaker verification, speaker identification, and language identification [11]. Compared to the i-vector, the d-vector is more robust to varying speech conditions and has shown promising results in various SLR applications.

Gal:

The majority of modern language recognition systems rely on i-vectors [12]. The conventional method consists of a universal background model (UBM) and a huge projection matrix T that are trained to maximize the likelihood of the training data [13]. The projection converts the (UBM) high-dimensional statistics into a low-dimensional form known as an i-vector. I-vectors are typically categorized after extraction using Gaussian models, logistic regression, or cosine scoring. Deep neural networks (DNN) trained as acoustic models for automated speech recognition (ASR) are used in cutting-edge i-vector systems [14]. This is frequently achieved in one of two ways: either the ASR DNN posteriors replace those from a Gaussian mixture model (GMM), or the i-vector system is trained using bottleneck characteristics extracted from the DNN [15, 16].

Idan:

The various techniques for spoken language recognition (SLR) are categorized into two types: feature-based and model-based [17]. Model-based strategies develop models of language-specific characteristics, whereas feature-based methods extract acoustic features. Some of the most often used SLR techniques are the Gaussian Mixture Model (GMM), the Hidden Markov Model (HMM) (For more information [18]), the Support Vector Machine (SVM) [19], Neural Networks (NN) [20], Convolutional Neural Network (CNN) [21], and Long Short-Term Memory (LSTM) [22]. Each technique has advantages and disadvantages, and the technique used is determined by the individual application and available training data. In addition, neural network-based algorithms, such as CNNs and LSTMs, are excellent for dealing with complex feature spaces and temporal fluctuations in voice signals, but they require a significant amount of training.

Gal:

In this article, we will look at the technologies used to recognize spoken language and the technical and practical issues that arise as a result of this process. We use x-vectors to perform spoken language recognition. To evaluate and represent data in vector space, the Vector-X technique is utilized. In the case of spoken language identification, Vector-X may be beneficial as a tool for studying and modeling speech [23]. When using Vector-X for spoken language identification, the main job is to represent the voice as a vector in vector space. This vector represents the entire speech, including the words used, the grammatical structure of the speech (e.g., tenses, degrees, and so on), and any specific speech processing (e.g., voice pressure, tone, and so on). This data can be utilized as consistent information for machine learning models for recognizing and classifying spoken language by comparing the vector encoding of the voice to applicable language models. Furthermore, Vector-X may be utilized to distinguish between speaker goals, identify the melody used in the speech, and grasp the speaker's tone. To read more about X-vectors [24].

Table 1: *Comparison of i-vector, x-vector, and d-vector between Technique by the Representation, Modeling, Training, Architecture*

Technique	i-vector	x-vector	d-vector
Representation	Low-dim.	High-dim.	High-dim.
Modeling	Generative	Discriminative	Hybrid
Training	Supervised	Supervised	Sup./Unsup.
Architecture	GMM-UBM	CNN	LSTM

Gal:

Table 1 provides a comparison of three techniques commonly used for speaker recognition: i-vector, x-vector, and d-vector[25]. It highlights differences in their representation, modeling approach, training, and architecture. The i-vector is a low-dimensional technique with a generative modeling approach and requires supervised training using a GMM-UBM architecture. The x-vector is a high-dimensional technique with a discriminative modeling approach, requiring supervised training using a CNN architecture. The d-vector is a hybrid approach with high-dimensional representation, requiring both supervised and unsupervised training using an LSTM architecture.

Idan:

Spoken language recognition has numerous applications in security, language acquisition, speech treatment, customer service, and virtual assistants. It can be used in security systems to detect unauthorized access and identify suspects in forensic investigations. It can provide feedback on pronunciation and intonation to language learners and construct tailored language learning programs. It can diagnose speech abnormalities and give tailored therapy in speech therapy. Spoken language recognition can also be used in customer service and virtual assistants to improve communication and more precisely understand voice commands especially in children [26].

Idan:

The use of natural language processing and machine learning methods to improve the accuracy of language recognition systems is part of the future of spoken language recognition. There is also a priority on developing systems that can understand emotions and intent in spoken language, which could be valuable in customer service and healthcare applications. Deep

learning techniques, for example, are likely to play a key role in enhancing the accuracy of language recognition systems [27]. The advancement of spoken language recognition technology has enormous implications for cross-cultural communication, and it has the potential to play a critical role in breaking down language barriers and creating greater understanding and collaboration among individuals from various backgrounds.

2.0.1. Our contribution

This paper offers different methods to improve the x-vector and other embedding models using different architectures and various of data augmentations. In addition, this paper an accuracy improves TDNN and x-vector algorithms by using less data than the baseline model by this paper [28]. The algorithm was trained and tested on three different families of languages (Semitic, Indo-European, and East Asian) and 10 sublanguages(Arabic, Hausa, Farsi, Spanish, French, Italian, Russian, Chinese, Japanese, and Thai). Those languages were not chosen randomly, these languages are covering a big part of the non-English speakers around the world that a solution can help the language barrier around the world.

2.0.2. Paper Structure

Gal:

The remainder of this paper is structured as follows: Section 3 related work that mainly Section 3 referred to previous works that mainly used x-vector and TDNN and different methods. Section 4 talks about the dataset used in the article, how it is built and what it consists of and how it was chosen, and the pre-processing. Section 5 presents the approach employed in the article and the modification made to the new TDNN model, Section 6 shows the process of the experiment and the results of the new model including the modification and exploitation of the architectural structure that is different from the basic model, Section 7 provides a discussion, conclusions, and suggestions for future follow-up work, Table 2 provides a list of abbreviations that are commonly used in this paper.

Gabi:

Table 2: *List of abbreviations*

Abbreviation	# Meaning
TDNN	Time delay neural network
SLR	Spoken language recognition
GMM	Gaussian mixture models
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory Networks
DNN	Deep Neural Network
UBM	Universal Background Model
ASR	Automatic Speech Recognition
HMM	Hidden Markov Model
SVM	Support Vector Machine
DNN	Depp Neural Network
NN	Neural Network
RNN	Recurrent Neural Network
GPU	Graphics Processing Unit
NIST	National Institute of Standards Technology
MMI	Maximum Mutual Information
LID	language identification
WAV	Waveform Audio File Format
PC	Personal Computer

3. Related Work

Gabi:

There is a lot of information and studies about this subject. In [29], a different approach has been taken. They presented an LSTM RNN system running on limited computational resources (a single GPU) that outperforms a reference i-vector system on a subset of the NIST Language Recognition Evaluation (8 target languages, 3s task). In addition they extended previous experiments modeling unseen languages, They also analyze the effect of even more limited test data and they proved that with as little as 0.5s an accuracy of over 50% can be achieved they also claiming that LSTM RNN based approaches using open source implementations outperform the standard i-vector based approach when dealing with short test durations with an improvement of ~26% in terms of Equal Error Rate with respect to the reference i-vector based system.

Gabi:

another work by [30], trained i-vectors with Multiclass Discriminative.They showed an approach where a Gaussian i-vector classifier is trained using Maximum Mutual Information (MMI) to directly optimize the multiclass calibration criterion so that no separate back-end is needed. Results on the NIST LRE11 standard evaluation task confirm that high performance and calibration are maintained with this new single-stage approach.In addition, the system is extended to the open set task using the additive Gaussian noise model, and this is also discriminatively trained to improve performance.

Gabi:

A different work by [31], also took a different approach. They present a single sequence-to-sequence automatic speech recognition (ASR) model trained on 9 different Indian languages, which have very little overlap in their scripts. They took a union of language-specific grapheme sets and train a grapheme-based sequence-to-sequence model jointly on data from all languages. They found that this model, which is not explicitly given any information about language identity, improves recognition performance by 21% relative compared to analogous sequence-to-sequence models trained on each language individually. They also modified the model to accept a language identifier as an additional input feature and improved performance by an additional 7% relative and eliminate confusion between different languages.

Gabi:

In [32], they tried a different way. They introduce a stacked architecture that uses a time delay neural network (TDNN) to model long-term patterns for spoken language identification.The first component of the architecture is a feed-forward neural network with a bottleneck layer that is trained to classify context-dependent phone states (senones). The second component is a TDNN that takes the output of the bottleneck, concatenated over a long time span, and produces a posterior probability over the set of languages. The proposed system outperforms a state-of-the-art shifted delta cepstra i-vector system and provides complementary information to fuse with the new generation of bottleneck-based i-vector systems that model short-term dependencies. The proposed system provides a relative gain of 27% in identification accuracy.

Gabi:

In [33], they were using a Deep neural networks (DNN) and bottleneck joint i-vector systems.they did it by using discriminative Gaussian classifier combined with naive fusion used all of the development data for system design rather than holding some out for separate back-end training. They replaced i-

vectors with more powerful x-vectors to further improve language recognition accuracy. They were able to achieve the JHU HLTCOE LRE17 language recognition system which gave them very good results.

Gabi:

In [28]. They apply x-vectors to the task of spoken language recognition and deep neural network (DNN). They found that long-term language characteristics are captured in the network by a temporal pooling layer that aggregates information across time. Once extracted, x-vectors utilize the same classification technology developed for i-vectors. Finally they found that the best-performing system uses multilingual bottleneck features, data augmentation, and a discriminative Gaussian classifier.

Gabi:

In [34]. They propose two novel frontends for robust language identification (LID) using a convolutional neural network (CNN) trained for automatic speech recognition (ASR). The results that they got are complementary and give significant gains of up to 20% relative to the best single system when combined.

Gabi:

In [35], they showed a way using iVectors Space. here they recognize language in the iVector space, they e experiment with three different linear classifiers which one based on a generative model (SVM), where classes are modeled by Gaussian distributions with shared covariance matrix, and two discriminative classifiers, namely linear Support Vector Machine and Logistic Regression. Their dataset contain 23 target languages, and files of 3, 10 and 30 s. The results are shown in terms of Cavg ×100 defined in the NIST LRE 2009 Evaluation Plan [36].

Gabi:

In [37], they investigated the use of automatically collected web audio data. They generate semirandom search phrases from language-specific Wikipedia data that are then used to retrieve videos from YouTube for 107 languages. Speech activity detection and speaker diarization are used to extract segments from the videos that contain speech. The size of the resulting training set (VoxLingua107) is 6628 hours (62 hours per language on the average) and it is accompanied by an evaluation set of 1609 verified utterances. Another thing that they are doing is a post-filtering to remove segments from the database that are likely not in the given language which is increasing the proportion of correctly labeled segments to 98%, based on crowdsourced verification.

Gabi:

In [38], They was working with audio clips by unknown speaker, regardless of gender, manner of speaking, and distinct age speaker. The model uses audio files and converts those files into spectrogram images. The model applies the convolutional neural network (CNN) to bring out main attributes or features to detect output easily. The main objective is to detect languages out of 25 languages such as English, French, Spanish, and German. To this expirament they was using a data set from Kaggle. These audio files are comprised of utterances, each of them spanning over a fixed duration of 10 seconds. The whole dataset is split into training and test sets. Preparatory results give an overall accuracy of 98%. Extensive and accurate testing show an overall accuracy of 88%.

4. Datasets

Gabi:

In the field of speech recognition, the ability to accurately transcribe and understand diverse languages is of utmost importance. To address this challenge, our work utilized datasets sourced from Common Voice, a valuable resource for multilingual speech data [39]. We specifically focused on ten languages, carefully selected from three distinct language families: Indo-European, Semitic, and East Asian. This article aims to shed light on the significance of recognizing and supporting speech recognition in these languages, considering the substantial number of non-English speakers worldwide [40]. We believe that with these languages Our work aimed to contribute to the development of multilingual speech recognition systems. Such systems have the potential to bridge language barriers, empower non-English speakers, and foster inclusivity on a global scale.

Gabi:

1. Indo-European Languages:

Among the Indo-European language family, we chose to include Spanish, French, Italian, and Russian in our study. These languages were selected for their remarkable diversity and widespread usage across multiple continents. By incorporating languages from different regions, we aimed to capture the richness and variation present within this language family [41]. Now we will present some information about each language that we were using in the Indo-European Languages before the preprocess:

1. Spanish - in the Spanish language there is 24 Recorded hours and 11 Validated hours. The number of different speakers in the language are 207 speakers.

2. French - in the French language there is 12 Recorded hours and 15 Validated hours. The number of different speakers in the language are 298 speakers.

3. Italian - in the Italian language there is 4 Recorded hours and 4 Validated hours. The number of different speakers in the language are 49 speakers.

4. Russian - in the French language there is 8 Recorded hours and 5 Validated hours. The number of different speakers in the language are 103 speakers.

Gabi:

2. Semitic Languages:

In our research, we recognized the importance of Semitic languages, which have historical and cultural significance in various parts of the world [42]. To represent this language family, we focused on Arabic, Hausa, and Farsi. These languages are spoken by millions of people globally and play a vital role in their respective communities. By including Semitic languages in our study, we aimed to address the specific challenges and intricacies associated with their speech recognition. Here is some statistics about each language that we were using in the Semitic Languages before the pre-process:

1. Arabic - in the Arabic language there is 2 Recorded hours and 1 Validated hours. The number of different speakers in the language are 77 speakers.

2. Hausa - in the Hausa language there is 13 Recorded hours and 4 Validated hours. The number of different speakers in the language are 39 speakers.

3. Farsi - in the Farsi language there is 2 Recorded hours and 4 Validated hours. The number of different speakers in the language are 37 speakers.

Gabi:

3. East Asian Languages:

The third group we explored consisted of East Asian languages. Chinese, Japanese, and Thai were selected as representatives of this diverse and vibrant language family. East Asian

languages possess unique phonetic structures and tonal characteristics, making them intriguing subjects for speech recognition research. Given the sizable populations that speak these languages, their accurate recognition holds great significance for enabling effective communication and access to technology for non-English speakers [43]. We will present some knowledge about each language that we were using in the East Asian Languages before the pre-process:

1. Chinese - in the Chinese language there is 5 Recorded hours and 95 Validated hours. The number of different speakers in the language are 189 speakers.

2. Japanese - in the Japanese language there is 4 Recorded hours and 3 Validated hours. The number of different speakers in the language are 36 speakers.

3. Thai - in the Thai language there is 7 Recorded hours and 5 Validated hours. The number of different speakers in the language are 77 speakers.

Gabi:

Table 3: *Languages and sub-languages*

Languages	sub-languages
Indo-European	Spanish,French,Italian,Russian.
Semitic	Arabic,Hausa,Farsi.
East Asian	Chinese,Japanese,Thai.

* A quick note about the data is that every data contains male and female as speakers, the ages of the speakers are between teenagers to old and for each language there are several dialects inside the segments. Unfortunately we were not able to find some information about the distribution for each language on this statistics.

Another thing that is important to mention is the differences between Recorded hours and Validated hours in Common Voice dataset , the advantages of validated data [44].

- Recorded hours refer to the total amount of audio data that has been contributed to the Common Voice project. It represents the cumulative duration of all the voice recordings that users have submitted.

- Validated hours, on the other hand, represent the subset of recorded hours that have been manually reviewed and verified by the Common Voice community. The validation process involves volunteers listening to and evaluating the contributed recordings to ensure they meet certain quality criteria, such as clarity, naturalness, and adherence to the specified prompts. Validated hours reflect the portion of recorded data that has been approved and deemed suitable for use in training machine learning models.

Understanding languages, not speakers it's crucial for the model to learn the linguistic content rather than the specific characteristics of individual speakers. This is due to several reasons:

1. Generalization: The goal of models in speech recognition or language identification is to understand the spoken content, regardless of who the speaker is. If the model learns to recognize specific speakers instead of the language, it may fail to generalize to new speakers. This could severely limit the model's utility in real-world applications, as it would only work well for speakers it has been trained on [45].

2. Variation in speech characteristics: Each individual speaker has unique speech characteristics, such as pitch, speed, accent, etc. If a model is trained on data from a limited number of speakers, it might become biased toward those specific speakers' characteristics. To ensure the model works well with a wide range of speakers, it is important to include a diverse set of speakers in the training data.

3. Privacy considerations: In many applications, the identity of the speaker is sensitive information. Models that focus on understanding languages rather than identifying speakers are more privacy-friendly, as they don't need to store or learn from personal voice characteristics.

4. Efficiency and scalability: Learning languages rather than speakers is generally a more scalable approach. There are potentially billions of different speakers, but only a few thousand languages. It would be much more efficient to build models that understand languages rather than trying to handle each individual speaker separately.

5. Addressing data imbalance: In most datasets, the number of utterances per speaker is often imbalanced, with some speakers being over-represented and others under-represented. If a model learns to recognize speakers, it may become biased towards speakers with more data. On the other hand, focusing on languages allows the model to learn more generalized, language-specific features[46].

Gabi:

As part of the research we decided to check the necessity of the pre-process on the dataset that we took from common voice. First of all Data pre-processing is an important step in any machine learning or data analytics project because raw data collected from various sources frequently contains noise, outliers, missing values, or is in an unusable format. The goal of pre-processing is to transform this raw data into a clean, understandable, and efficient format suitable for machine learning algorithms. It improves the quality and reliability of the data, reduces complexity, and can improve the accuracy and efficiency of the subsequent analysis or prediction. Without pre-processing, machine learning models may be built on inaccurate or irrelevant data, resulting in misleading or incorrect results[47]. In our article the data set from common voice is considered as a clean dataset, but several steps related to this problem can be performed.

4.1. Balanced data

Gabi:

The first thing that was done is to take the same amount of data from each language. We chose the minimum number of voice segments that we had among these 10 languages, which was 994 segments so totally we are working with 9940 voice segments. The impotence of working with balanced data is very big and this is several reasons that explain this:

1. Avoiding Bias: Imbalanced data can introduce bias in predictive models. When the dataset is skewed towards one class, the model can become overly sensitive to the majority class and may struggle to accurately predict the minority class. This can lead to poor performance and skewed results.

2. Accurate Representation: Balanced data ensures that each class is represented equally, allowing the model to learn from a diverse range of examples. It helps in capturing the true distribution of the data and prevents the model from making assumptions based on an imbalanced representation.

3. Reliable Evaluation: Working with balanced data allows for a more reliable evaluation of model performance. When evaluating the model on imbalanced data, metrics like accuracy can be misleading. For instance, if 95% of the data belongs to the majority class, a model that predicts all instances as the majority class will have high accuracy, even though it fails to classify the minority class correctly. Balanced data enables more accurate assessment of metrics like precision, recall, and F1-score.

4. Improved Generalization: Models trained on imbalanced data may struggle to generalize well to new, unseen data. Balanced datasets help models to learn more robust and representative patterns, which can improve their ability to make accurate predictions on unseen examples.

5. Better Decision Making: In domains where the cost of misclassification is significant, such as healthcare or fraud detection, working with balanced data becomes crucial. A model trained on imbalanced data might make incorrect predictions that have severe consequences. By balancing the data, the model can make more informed decisions across all classes.

To read more about the importance of balanced data [48].

After The Creation of 994 segments to each language we decided to divide the data to 3 parts which are:
-Train
-Test
-Validation
Gal:
We decided to split the data as follows (Train - 70%, Validation - 20%, Test - 10%). Splitting the data into training, testing, and validation sets is a crucial step in the development and evaluation of machine learning models [49]. Here are the main reasons why data splitting is important:

* Model Training: The training set is used to train the machine learning model. It contains labeled data that the model uses to learn patterns and relationships between input features and their corresponding outputs. By exposing the model to a diverse range of examples during training, it can better generalize and make accurate predictions on unseen data.

* Model Evaluation: The testing set is used to evaluate the performance of the trained model. It contains labeled data that the model has not seen during training. By evaluating the model on this independent dataset, we can assess its ability to generalize and make accurate predictions on new, unseen data. This helps us understand how well the model is likely to perform in real-world scenarios.

* Hyperparameter Tuning: Hyperparameters are configuration settings that are not learned from the data but are set prior to training. These include parameters like learning rate, batch size, regularization strength, etc. The validation set is used to tune and optimize these hyperparameters. Multiple models with different hyperparameter settings can be trained and evaluated on the validation set to select the best-performing model. This helps prevent overfitting and improves the generalization ability of the model.

* Preventing Data Leakage: Splitting the data into separate sets helps prevent data leakage, which can lead to biased and overly optimistic performance estimates. Data leakage occurs when information from the test or validation sets unintentionally leaks into the training process. By keeping these sets isolated, we can obtain unbiased estimates of the model's performance on new, unseen data.

* Generalization Assessment: The testing set provides an unbiased estimate of the model's performance on new, unseen data. It helps determine whether the model has learned meaningful patterns and can generalize well beyond the training data. By evaluating the model's performance on the testing set, we can make informed decisions about deploying the model in real-world applications.

Overall, data splitting into training, testing, and validation sets is essential for building robust and reliable machine-learning models. It helps in training the model, evaluating its performance, tuning hyperparameters, preventing data leakage, and assessing generalization ability.

Here is some resource that is talking about that importance [50].
Gal:
The second thing that was done is to convert the mp3 files that we were receiving from Common Voice to wav files, There are reasons to convert audio files from MP3 to WAV format. The following are the primary reasons Lossless vs. Lossy Compression, Uncompressed Data, 3. Sampling Rate Consistency, Compatibility with Libraries.

4.2. Cleaning data

Gal:
The third thing that was done in the pre-process that we did is to "clean" all the "dead" parts for each segment that we are working with, in any data analysis or machine learning project, the quality and cleanliness of the data used play a vital role in determining the accuracy and reliability of the results. When saying "dead" parts we mean to the portions of data within a particular segment that has no voice or the voice is very low proportional to the thresh hold that we determined which means these segments are irrelevant. here are some advantages to "clean the data:

1. Enhanced Accuracy: Removing "dead" parts eliminates outdated or irrelevant data, ensuring that the analysis is focused on the most current and meaningful information. This helps in reducing noise and unnecessary complexity in the dataset.

2. Improved Data Integrity: By cleaning out redundant or invalid data, the overall integrity and consistency of the dataset are improved. This increases the reliability of subsequent analyses and models built upon the cleaned data.

3. Unbiased Analysis: By removing "dead" parts, we reduce the risk of bias that can arise from including irrelevant or outdated data. A clean dataset ensures that the analysis is based on accurate and representative information, leading to more reliable insights and decisions.

To read more about this importance [51].

4.3. Data augmentation

Gal:
The fourth thing that was done with the dataset is to generate another three different types to get three different augmentations as we saw [52]. The extra three augmentations that we choose are:

1. Speed: Modifying the speed of audio clips can simulate the variation in speech rate among different speakers. By accelerating or decelerating speech signals, we can generate new training instances and help the model become invariant

to the rate of speech. It may improve the model's ability to understand fast or slow speakers [53].

2. Pitch: Pitch perturbation is a way to mimic the natural variation in the pitch of human voices. Shifting the pitch of the audio signal can also help make the model robust to different speaker characteristics. A model trained with pitch-augmented data might better handle audio from speakers with unusually high or low pitches[54].

3. Noise: Adding different types of noise to your audio data is a common way to simulate real-world conditions, where clean, noise-free audio is rarely available. This can help improve the model's performance in noisy conditions. Different types of noise (white noise, pink noise, environmental noise) could be used based on the application[52].

There are several reasons why data augmentation is important to us and as a result [55], we have implemented these methods in our process

1. Improving model performance: Data augmentation techniques create variations of the existing data, helping the model learn and generalize better. This can lead to improved performance on unseen data and enhance the robustness of the model.

2. Mitigating overfitting: Overfitting is a common problem in machine learning where a model learns the training data too well and performs poorly on unseen data. By increasing the diversity and size of the training data through augmentation, the model can be prevented from memorizing the training examples and instead, learn more general patterns.

3. Addressing class imbalance: In many real-world scenarios, the distribution of classes in the data is imbalanced, which can bias the model towards the majority class. Augmentation can be used to create more instances of the minority class(es), balancing the class distribution and helping the model to learn more representative features of each class.

4. Simulating real-world conditions: Especially in fields like image and audio processing, data augmentation can simulate different real-world conditions that were not present in the original dataset, such as different lighting conditions in images or background noise in the audio.

5. Framework

5.1. Baseline model and new model

5.1.1. Introduction to Baseline Model

Gal:
The x-vector DNN architecture is the most common. Before the nonlinearity, x-vectors are extracted at layer segment 6. F-dimensional characteristics can be accepted by the input layer. The softmax layer's L denotes the number of training languages.
Gal:
The DNN configuration is outlined in Table 1. The input to the DNN is a sequence of T speech frames. The first five layers process the data on a frame-by-frame basis, with a limited temporal context centered on the current frame t. The input

Layer	Layer context	Tot. context	In x out
frame1	$[t-2, t+2]$	5	$5F$x512
frame2	$\{t-2, t, t+2\}$	9	1536x512
frame3	$\{t-3, t, t+3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	$1500T$x3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xL

Figure 1: *Architecture Source Model (Baseline) based on the Spoken Language Recognition using X-vectors [28] from github.com/KrishnaDN/x-vector-pytorch*

layer, frame1, for example, splices together the F-dimensional features at frames t 2, t 1, t, t + 1, and t + 2, giving it a total temporal context of 5 frames. The following layer's input, frame2, is the spliced output of frame1 at t 2, t, and t + 2. This layer extends the temporal context produced by the previous layer, giving frame2 a total context of 9 frames. This technique is repeated in the subsequent layers, resulting in frame 5 viewing a total context of 15 frames.
Idan:
The statistics pooling layer collects data across time dimensions, allowing the following layers to function on the full segment [56]. A sequence of T 1500-dimensional vectors from the previous layer, the frame5, is fed into the pooling layer. The result is the mean and standard deviation of the input (each vector has 1500 dimensions). These statistics are concatenated (to form a 3000-dimensional vector) and sent via the segment-level levels before reaching the softmax output layer. The nonlinearities are represented using rectified linear units (ReLUs [57]). Section 6 networks have between 4.2 and 4.6 million parameters.
Idan:

5.1.2. The New Model

Table 4: *Architecture our Model with the changes from the baseline model, Adding new layers and changing the In X out Values*

Layer	Layer context	Tot. context	Tot. width of context	In x out
frame 1	$\{t-2, t+2\}$	3	5	3Fx1280
frame 2	$\{t\}$	3	5	1280x1280
frame 3	$\{t-4, t-2, t, t+2, t+4\}$	7	9	8960x1024
frame 4	$\{t\}$	7	9	1024x1024
frame 5	$\{t-1, t+1\}$	9	11	9216x768
frame 6	$\{t\}$	9	11	768x512
frame 7	$\{t\}$	9	11	512x256
stats pooling	$[0, T)$	T	T	256Tx512
segment 8	$\{0\}$	T	T	512x512
segment 9	$\{0\}$	T	T	512x512
softmax	$\{0\}$	T	T	512xL

Table 4 presented above shows the new structure of the integrated Model in this research. A 2 new layers were added (layer 2 and layer 4) these 2 layers were added as an intermediate layers and not part of the funnel structure. The rest layers changed to be in the structure of a funnel.

Idan:
In our method, we chose to work on the BASELINE

MODEL and change its architecture and hyperparameters in several ways to improve the model's generalizability and its results. We ran the changes we made separately on the baseline model to see the significance of each of the changes. In the article, we will present 3 key changes that significantly improved the results of the model:

1. Grid Search on the 'context size' and 'dilation' hyperparameters in the TDNN layers[58].

2. Changing the number of neurons in the TDNN layers from a fixed number (512) to a funnel shape [59].

3. Adding intermediate layers for further processing of the frames and context [60].

In the following sections, we will present each of these changes in detail. We will include a general explanation of each topic, the theoretical meaning of these changes, important details about the actual implementation and finally, we will present the results. In the end, we created an integrated model which contains these three improvements. The results of this model will be presented later.
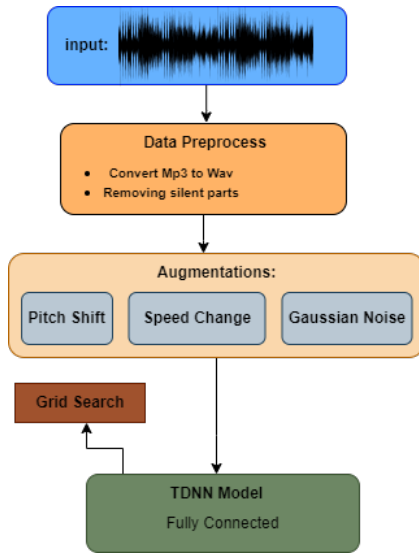

Figure 2: *Structure of the network: In the first stage we insert mp3 as input, Covert the audio in the data Preprocess to wav, made some augmentations such as pitch shift, speed, and adding Gaussian noise, feed the TDNN model and loop over Grid Search*

5.1.3. Loss Function

Gabi:

In the new model and the baseline model as well Cross Entropy Function Loss is used. A presentation of a simple cross-entropy function:

$$-(y\log(p) + (1-y)\log(1-p))$$

Where:
- p is the predicted probability, and
- y is the indicator (0 or 1 in the case of binary classification)

Basically, Cross entropy loss measures the difference between the discovered probability distribution of a machine learning classification model and the predicted distribution [61].

in more detail the cross-entropy between two probability distributions p and q over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set if a coding scheme used for the set is optimized for an estimated probability distribution q, rather than the true distribution p.

To read more on the Cross-Entropy Function Loss, [62].

5.2. Cleaning silent sections

Gabi:

As part of the goal of the research, to improve the baseline model we chose for language recognition, we also performed an initial data processing process. This process involved cleaning the audio clips from silent sections. we conducted a study to investigate the potential benefits of cleaning silent sections in audio data for improving the success of our model. Our motivation behind exploring this approach stemmed from previous research literature that suggested the effectiveness of cleaning silent sections in enhancing model performance [63]. We hypothesized that by removing silent segments, we could reduce noise and irrelevant information, thereby allowing the model to focus on the meaningful audio content.

To test this hypothesis, we cleaned the audio files we are working on and retrained the null model. Although we expected a significant improvement in the results of the model, we did not see such a result. The problem is that we utilized audio clips from the Common Voice dataset, which is known for its high-quality and clean speech data. we realized that the audio clips in this dataset were already remarkably clean to the point where any additional cleaning, including the removal of silent sections, rendered negligible improvements. The lack of significant improvements can be attributed to the fact that the audio clips were devoid of substantial background noise, distortions, or other artifacts that typically warrant cleaning interventions. Therefore, we do not present the results of this experiment, but it is important for us to emphasize that in most cases it is necessary to clean the audio segments, unlike our work.

5.3. Architecture improvements

5.3.1. Grid Search for Context Size and Dilation Hyper- parameters

Idan:

In TDNN (Temporal Convolutional Neural Network) layers, the concepts of context size and dilation play crucial roles in capturing temporal dependencies and modeling sequential data [64].

The context size in a TDNN layer refers to the number of adjacent frames or time steps considered during the convolution operation. It determines the range of temporal information that the layer can capture. A larger context size allows the layer to incorporate a broader temporal context, whereas a smaller context size focuses on local temporal relationships. By considering a wider context, TDNN layers can capture longer-range dependencies and learn higher-level temporal patterns in the data [65]. This can be particularly useful in tasks where understanding longer-term context is important, such as speech recognition or natural language processing. On the other hand, smaller context sizes are more suitable for capturing fine-grained details and immediate temporal relationships. The choice of an appropriate context size depends on the specific characteristics of the data and the task at hand. It requires balancing the need to capture relevant temporal dependencies while avoiding excessive

computational complexity or overfitting.

Idan:

Dilation refers to the spacing between the input frames or time steps that the TDNN layer considers during the convolution operation. It controls the skip or gap between the frames and influences the receptive field of the layer. In TDNN layers, a dilation of 1 means that there is no skip or gap between the input frames. Each frame is considered sequentially during the convolution. A dilation greater than 1 introduces skips or gaps between the frames, allowing the layer to capture non-adjacent temporal relationships and capture information at larger time scales. By using dilation, TDNN layers can capture temporal dependencies that span a larger range without significantly increasing the number of parameters or computational complexity. This is particularly useful for modeling long-range dependencies and capturing global temporal patterns in the data. The choice of dilation value depends on the specific requirements of the task and the desired balance between capturing local and global temporal relationships. Smaller dilation values (e.g., 1) are suitable for capturing fine-grained local patterns, while larger dilation values allow for modeling longer-range dependencies.

In summary, the context size and dilation in TDNN layers are essential parameters for modeling sequential data. The context size determines the range of temporal information considered, while dilation controls the skip or gap between frames and influences the receptive field. Choosing appropriate values for context size and dilation is crucial for capturing relevant temporal dependencies, modeling different temporal scales, and effectively learning from sequential data.

Grid search is a good way to find optimal values for dilation and context size [66]. Grid search is a systematic and exhaustive search method that can be used to find optimal values for dilation and context size in TDNN layers. It involves defining a grid of possible values for each hyperparameter and evaluating the model's performance for all possible combinations of these values.

Idan:

Grid search is a technique used for hyperparameter optimization in machine learning. It involves defining a grid of hyperparameter values to be searched exhaustively. The grid search algorithm then trains and evaluates the model for each combination of hyperparameters in the grid and selects the one that performs the best. the steps involved in grid search:

1. Define the hyperparameters to be tuned: Identify the hyperparameters in your model that you want to optimize. In our project, those are context size and dilation in the 3 first TDNN layers.

2. Define the grid: Specify the set of possible values for each hyperparameter. context size values are [2, 3, 5, 7] and dilation values are [1, 2, 3].

3. Generate combinations: Create all possible combinations of hyperparameters from the defined grids.

4. Training and evaluation: For each combination of hyperparameters, train the model using the training dataset and evaluate its performance using a validation set. This involves fitting the model, making predictions, and computing a performance metric.

5. Select the best combination: Compare the performance of models trained with different hyperparameter combinations and select the one with the best performance metric. This is typically done by choosing the hyperparameters that maximize the metric or minimize a loss function.

As mentioned above, grid search exhaustively explores all possible combinations of hyperparameters, which can be computationally expensive for large hyperparameter spaces. However, it ensures a thorough search and can help in finding the optimal set of hyperparameters for your model. Grid search is a commonly used technique for hyperparameter tuning and can be a good option for model improvement for several reasons:

1. Exhaustive search: Grid search performs an exhaustive search over a predefined grid of hyperparameter values. It systematically explores all possible combinations, ensuring that no potential set of hyperparameters is overlooked. This comprehensive search approach increases the likelihood of finding a good set of hyperparameters that can improve model performance.

2. Simplicity and transparency: Grid search is a straightforward and easy-to-understand approach. It allows you to define the hyperparameter grid explicitly, making it transparent and interpretable. You can easily specify different values and ranges for hyperparameters, enabling you to control the exploration space according to your domain knowledge or intuition.

3. Reproducibility: Grid search is deterministic, meaning that it produces the same results each time it is run with the same hyperparameter grid. This makes the experimentation process reproducible, allowing you to compare different models consistently and track the impact of hyperparameter changes over time.

4. Baseline performance: Grid search provides a baseline for model performance. By trying different hyperparameter combinations, you can observe the variations in model performance metrics and identify the best-performing set of hyperparameters. This helps you understand the impact of different hyperparameters on your model and sets a benchmark for future improvements.

5. Compatibility with limited datasets: In cases where you have a limited amount of data, grid search can be a suitable option. It can explore the hyperparameter space more exhaustively compared to other optimization techniques like random search, which might require a larger number of iterations to achieve similar coverage.

Despite its benefits, grid search also has some limitations. It can be computationally expensive, especially when dealing with a large number of hyperparameters or when the grid size is extensive. Additionally, grid search does not consider the interactions between hyperparameters, and it assumes that each hyperparameter is independent of others. As a result, it may miss out on complex interactions that can affect model performance.

To mitigate these limitations, more advanced techniques like random search, Bayesian optimization, or evolutionary algorithms can be used. These techniques provide more efficient ways to search the hyperparameter space and often outperform grid search in terms of finding the best hyperparameter combinations within a limited computational budget. The use of these methods will be left for future work.

Gal:

Gabi:

It is important to be precise about the process we did using grid search in our method. In each of the first 3 layers, there are the parameters context size and dilation, that is, there are 6 hyper-parameters for which we are looking for the optimal combination values in the selected values to check from, which are [2, 3, 5, 7] for context size and [1, 2, 3] for dilation. Each layer has 12 possible combinations. If we were to search for the most threatening values for all 3 layers together, we would reach a huge amount of combinations (12 power 3). Due to computa-

Context size	Dilation	Accuracy	Loss
2	1	0.39	1.70
2	2	0.54	1.37
2	3	0.56	1.28
3	1	0.34	1.90
3	2	0.56	1.29
3	3	0.60	1.21
5	1	0.55	1.35
5	2	0.66	1.07
5	3	0.59	1.20
7	1	0.60	1.21
7	2	0.59	1.20
7	3	0.62	1.08

tional resource limitations, we tested each layer separately - we looked for the optimal values for the first layer while fixing the existing values in the other layers. Also, in each combination, we ran a relatively small amount of iterations to speed up the search process, for the first 2 layers we ran 10 epochs, and for the third layer 20 epochs. After finding the optimal values for the first layer, we determined them as well and searched for the optimal values for the second layer. In the same way, we also searched for the optimal values for the third layer.

The optimal values we found are: Layer 1 - context size=3, dilation =2. accuracy=0.72, loss=0.88. Layer 2 - context size=5, dilation =2, accuracy=0.66, loss=1.07. Layer 3 - context size=2, dilation =1, accuracy=0.85, loss=0.49.

It is important to note that the values chosen are not necessarily the absolute optimal values for several reasons: a. The number of epochs is significantly smaller and we chose the optimal values after these iterations. B. The selectable values for context size and dilation are limited. third. Choosing the values for each layer separately limits the testing of the options and therefore the chosen values are probably not the best. However, they should indeed indicate a direction to improve the model.

5.3.2. 1x1 TDNN intermediate layers

Idan:

Another improvement that was done in our research is adding 1x1 TDNN intermediate layers to the baseline model[28]. In the research there was a TDNN (Time delay neural network, to read more about TDNN [67]) of 5 layers, In this research 2 more layers were added, the first layer was added between the original layers 1 and 2 and the second layer was added between the original layers 2 and 3.

TDNN layers with a context size of 1 and dilation of 1 are commonly referred to as "1x1 TDNN layers." These layers are a specific type of temporal convolutional layer that operates on sequential data, such as speech or text, by considering only the current frame or time step without any temporal context or skip connections.

The context size of 1 means that these layers capture information from the current frame or time step only, without taking into account any neighboring frames. The dilation of 1 indicates that there are no skip connections or gaps between the input and output frames.

1x1 TDNN layers are often used in neural network architectures to introduce non-linearities and transformations to the input data [68]. They can be beneficial for capturing fine-grained local features and patterns in the sequential data. By focusing solely on the current frame, these layers can emphasize immediate temporal relationships and dependencies, making them suitable for tasks where the current time step is particularly relevant or where fine-grained information is important.

In various applications such as speech recognition, natural language processing, and audio processing, 1x1 TDNN layers can be used to enhance the model's ability to capture intricate temporal patterns at a fine-grained level and improve the overall performance and representation power of the network.

When a TDNN layer has a context size of 1 and dilation of 1, it means that it only considers the current frame without looking at any surrounding frames [69]. This can lead to improvement in certain scenarios due to the following reasons:

1. Local pattern detection: By focusing only on the current frame, the TDNN layer becomes more sensitive to local patterns within the input. It can capture specific details and variations in the features that might be important for discrimination. This local pattern detection can help the model to better distinguish between different classes or categories.

2. Reduction of temporal dependencies: In some cases, the temporal dependencies between frames may not be as significant or informative for the task at hand. By using a context size of 1, the TDNN layer effectively reduces the influence of temporal dependencies, allowing the model to focus more on the current frame's characteristics. This can be beneficial when the task requires emphasizing instantaneous or short-term information rather than long-term dependencies.

3. Non-linearity enhancement: The TDNN layer with a context size of 1 and dilation of 1 introduces an additional non-linear transformation to the feature representation. It enables the model to capture complex relationships and nonlinear interactions within the input features. This increased non-linearity can enhance the discriminative power of the model by enabling it to learn more intricate decision boundaries.

4. Parameter efficiency: Using a context size of 1 reduces the number of parameters required in the TDNN layer compared to larger context sizes. This can be advantageous in terms of computational efficiency and model complexity, especially when dealing with large-scale datasets or limited computational resources.

It's important to note that the effectiveness of using a context size of 1 and dilation of 1 depends on the specific characteristics of the dataset and the requirements of the task. In some cases, incorporating information from neighboring frames or using larger context sizes might be more beneficial. It's recommended to experiment and tune these hyperparameters based on the specific problem at hand.

Also, we can use 1x1 TDNN layers as intermediate layers. Additional layers inserted between the TDNN layers with a context size bigger than 1 and dilation bigger than 1 can lead to improvement for several reasons:

1. Feature transformation: These intermediate layers serve as additional stages for feature transformation and extraction. The TDNN layers with larger context sizes and dilations are designed to capture long-range dependencies and context information, which can be helpful for modeling temporal relationships. However, introducing intermediate layers allows for more flexibility in transforming the features before passing them to the

next TDNN layer. These transformations can enhance the discriminative power of the model by extracting more informative representations from the input.

2. Non-linear mapping: The intermediate layers can introduce non-linear mappings between the TDNN layers. This non-linearity enables the model to capture complex relationships and interactions within the feature space. By adding these non-linear mapping layers, the model becomes more expressive and can learn more intricate decision boundaries, which can lead to improved performance.

3. Hierarchical feature learning: The intermediate layers provide an opportunity for hierarchical feature learning. Each TDNN layer, including the added intermediate layers, can learn different levels of abstraction from the input features. The initial TDNN layers with larger context sizes and dilations can capture high-level temporal patterns, while the intermediate layers can focus on refining and extracting more specific information relevant to the task. This hierarchical feature learning can help the model to better discriminate between different classes or categories.

4. Parameter efficiency and regularization: The intermediate layers can also act as a form of regularization and parameter efficiency. By introducing these additional layers, the model's capacity is increased, allowing it to capture more complex relationships within the data. This can help prevent overfitting and improve generalization performance. Additionally, by inserting smaller layers between the larger TDNN layers, the total number of parameters in the model remains manageable, which is particularly useful when dealing with limited computational resources or large-scale datasets.

5. Capturing different temporal scales: The intermediate layers can capture information at different temporal scales. While the TDNN layers with larger context sizes and dilations capture long-range dependencies, the intermediate layers with smaller context sizes and dilations can focus on capturing short-term dependencies or fine-grained details within the input. This multi-scale analysis enables the model to consider different temporal dynamics and enhances its ability to discriminate between classes.

Overall, adding intermediate layers between TDNN layers with larger context sizes and dilations allows for more flexible feature transformation, introduces non-linear mappings, facilitates hierarchical feature learning, provides regularization, and enables the model to capture information at different temporal scales. These factors collectively contribute to improving the model's performance and its ability to learn discriminative representations from the input data.

An example to a research that his topic was about adding layers to a deep convolutional network [70].

For a language detection model, adding intermediate layers between TDNN layers with larger context size and dilation can be meaningful [71], here are several reasons:

1. Capturing language-specific patterns: Languages have distinct linguistic characteristics, such as phonetic, syntactic, and semantic patterns. By introducing intermediate layers, the model can learn language-specific features at different levels of abstraction. The initial TDNN layers with larger context size and dilation can capture general temporal patterns across languages, while the intermediate layers can focus on capturing language-specific nuances and intricacies. This helps the model to better discriminate between languages based on their unique linguistic patterns.

2. Modeling temporal dependencies: Languages exhibit diverse temporal dependencies, including short-term dependencies within words and longer-range dependencies within sentences or discourse. The intermediate layers can capture these temporal dependencies by incorporating smaller context size and dilation. This allows the model to learn the specific temporal dynamics of different languages, enabling better language discrimination.

3. Handling language variations: Languages can exhibit significant variations in terms of accents, dialects, writing styles, and linguistic variations across regions or social groups. The intermediate layers provide a mechanism for the model to learn and encode these variations. By inserting layers that refine and extract more specific information, the model can adapt to different language variations and make more accurate predictions.

4. Robustness to noise and input variations: Language detection models often need to handle noisy or distorted input, such as low-quality audio recordings or text with misspellings or abbreviations. The intermediate layers can act as filters and transform the input features in a way that enhances their robustness to noise and input variations. These layers can help the model to focus on the most relevant and discriminative information for language detection, improving its performance in challenging conditions.

5. Generalization across languages: By incorporating intermediate layers, the model becomes more flexible and capable of capturing diverse language patterns. This improved generalization enables the model to detect languages accurately, even for languages that were not explicitly seen during training. The intermediate layers help the model to extract high-level features that are informative and discriminative across different languages, leading to better language detection performance on unseen data.

Overall, adding intermediate layers between TDNN layers with larger context sizes and dilations in a language detection model allows for capturing language-specific patterns, modeling temporal dependencies, handling language variations, improving robustness to noise, and enhancing generalization across languages. These benefits collectively contribute to building a more accurate and robust language detection system.

In current work we add 2 TDNN layers, specifically 'tdnn2' and 'tdnn4', which act like 1x1 TDNN intermediate layers.

'tdnn2': This layer takes the output from 'tdnn1' as input. 'tdnn4': This layer takes the output from 'tdnn3' as input. Both use a context size of 1 and dilation of 1, meaning it only considers the current context. The purpose of this layer is to further transform the features and increase the non-linearity of the model. as mentioned before, It helps capture more complex patterns in the input. This layers aim to increase the model's capacity to capture intricate patterns and enhance its discriminative power. These additional intermediate TDNN layers ('tdnn2' and 'tdnn4') provide more non-linear transformations and intermediate representations of the input features. They allow the model to learn more complex relationships and patterns in the data, potentially leading to improved performance.

It is important to note that the additional TDNN layers (tdnn2 and tdnn4) are not actually 1x1 TDNN layers. It is true that they have a context size of 1 and their dilation is still 1. Therefore, they are similar to the previous TDNN layers in terms of their receptive field so they do not operate on single frame. However, these intermediate TDNN layers still serve a purpose in the model. They introduce additional non-linear transformations and increase the capacity of the model to capture complex patterns and relationships in the input data.

By inserting these intermediate layers between the TDNN

layers with larger context sizes and dilations, the model becomes more expressive and can learn more intricate decision boundaries. These layers provide more flexibility in transforming the features before passing them to the next TDNN layer, allowing the model to refine and extract more specific information relevant to the task.

While the TDNN layers with larger context sizes and dilations focus on capturing long-range dependencies and global temporal patterns, the intermediate layers with a context size of 1 help to enhance the representation power of the model by introducing additional non-linear mappings and capturing fine-grained local features.

For a language detection model, these intermediate layers can be meaningful because they allow the model to capture both global and local patterns in the input data. Languages have specific linguistic characteristics and variations that can be captured at different levels of abstraction. By incorporating intermediate layers, the model can learn to extract language-specific features and capture both short-term and long-term dependencies in language data.

Overall, these intermediate TDNN layers contribute to improving the model's ability to capture intricate patterns, enhance non-linear transformations, and refine the representations of the input features. This can be beneficial for tasks such as language detection, where capturing both global and local language patterns is important for accurate classification.

The information in the section above was processed from the following articles [65, 67, 46]

5.3.3. TDNN funnel structure

Gabi:

In the Original structure, there are 5 TDNN layers of 512 neurons each. In this research, the structure of the network has changed to a funnel shape. a funnel shape of a structure in the literature known more by the name "bottleneck" Here is some more information about this principle [72]. The funnel structure has some advantages:

1. Dimensionality reduction: The funnel structure gradually reduces the dimensionality of the data as it progresses through the network. This dimensionality reduction can lead to more efficient and compact representations, making the subsequent layers more manageable.

2. Feature extraction and abstraction: The funnel structure enables the network to perform hierarchical feature extraction and abstraction. As the data passes through successive layers, the network can capture increasingly higher-level and abstract representations of the input. This hierarchical processing allows the network to learn complex patterns and dependencies in the data by progressively building upon the extracted features.

3. Information bottleneck and regularization: The narrowing structure of the network acts as an information bottleneck, forcing the network to prioritize and compress the most relevant information. This can act as a form of regularization, preventing overfitting and encouraging the network to learn more discriminative and generalizable representations. The bottleneck forces the network to focus on the most informative features, which can improve the model's performance and generalization capabilities.

4. Efficient computation and memory usage: The funnel structure can lead to more efficient computation and memory usage. As the dimensionality decreases in the bottleneck layers, the number of parameters and computations required decreases as well. This can result in faster training and inference

times, especially when dealing with large-scale datasets or limited computational resources.

5. Multi-resolution processing: A funnel structure can enable multi-resolution processing of the input data. By downsampling the data in the initial layers and then upsampling it in the subsequent layers, the network can simultaneously capture both global and local information.

6. Parameter efficiency: The funnel structure promotes parameter efficiency. With the decreasing number of channels or neurons in the bottleneck layers, the network requires fewer parameters to represent the compressed information. This parameter efficiency can lead to models with a smaller memory footprint and reduced computational complexity, making them more suitable for deployment on resource-constrained devices or in scenarios with limited computational resources.

7. Robustness to noise and variations: The funnel-like structure can help make the network more robust to noise and variations in the input data. The initial layers with larger receptive fields or pooling operations can provide a global view of the input, capturing important patterns and reducing the impact of local noise or variations. As the network progresses, the subsequent layers can focus on more localized and detailed information, refining the representations while filtering out noisy or irrelevant signals.

* The last section is very interesting to the topic of the research and can very help. Additionally, in some sources, it was mentioned that the first layer needs to start from a size of the Fourth root of the data size, since the fact in this research the data size is 9940 segments a Fourth root is too small so the sizes were chosen differently.

5.4. Final model

5.4.1. Integrated Model

After seeing the 3 key changes and their improvements over the baseline model (as shown in the sections above), our next step was to combine these 3 improvements into one model. We built the integrated model to contain the values found in the grid search, with 1X1 TDNN intermediate layers and a funnel structure, table 4. The results of this model appear in the results section and indicate that the combination of these methods does contribute to strengthening the model.

5.4.2. Fine Tunning using Augmentations

After running the integrated model we reached a result of 0.95 accuracy as can be seen in the results section. The last step we performed to improve our model is additional training on augmented data that includes changes in audio speed, adding noise and changing the pitch of the sound. Training on such data has great theoretical significance. Training on augmented data that includes changes in audio speed, adding noise, and changing the pitch of the sound can be beneficial for several reasons:

1. Increased Robustness: By exposing the model to a diverse range of audio variations, it becomes more robust to real-world conditions and different types of audio inputs. Augmentation helps the model generalize better and perform well even when encountering variations in speed, noise, and pitch.

2. Improved Generalization: Augmentation introduces variations that simulate real-world scenarios and increases the diversity of the training data. This helps prevent overfitting and improves the model's ability to generalize to unseen data. The

model learns to focus on the essential features and becomes less sensitive to minor variations in the input.

3. Data Expansion: Augmentation techniques effectively increase the size of the training dataset. With more data, the model can learn a more accurate representation of the underlying patterns and relationships in the audio. This can help mitigate the risk of overfitting when training on a limited amount of original data.

4. Noise Robustness: Adding noise to the audio can help the model learn to distinguish the relevant signal from background noise. This is particularly useful in speech-related tasks where the presence of noise is common. Training on augmented data with noise improves the model's ability to handle noisy environments and enhances its overall performance in real-world scenarios.

5. Addressing Data Imbalance: Augmentation can also be used to address data imbalance issues. By applying augmentation techniques to the minority classes or underrepresented samples, the model receives more exposure to these classes and learns to classify them effectively. This helps to overcome the challenges posed by unbalanced datasets and ensures that the model performs well across all classes.

Overall, training on augmented data helps create a more robust and generalizable model by exposing it to a wider range of variations and realistic conditions. This approach enhances the model's ability to handle diverse audio inputs and improves its performance in real-world applications. As can be assumed, this step significantly improved the model: from 0.95 to 0.97 accuracy on the validation set.

6. Experimental Evaluation And Results

Gal:

As were explained in the Framework there are a few changes that were tested separately, here is a comparison table showing the difference between all of them (The number of epochs is different because of the hardware restrictions):

Table 6: *Results Comparison on Validation Datasset for the Model with each of our improvements. The loss to each stage is also shown in ths table.*

Model	Accuracy	Loss
Baseline	0.54	1.44
Grid Search	0.85	0.49
Intermediate layers	0.79	0.60
Funnel	0.92	0.27
Integrated Model	0.95	0.2
Final Model	0.969	0.2

Baseline Model - In the first approach to deal with the problem we are facing the base model that was taken from [28] and based on the x-vector method approach that implemented in the following GitHub: "https://github.com/KrishnaDN/x-vector-pytorch". In this baseline model dataset is not provided, therefore there is a free hand to choose what dataset to use which makes it inaccuracy and does not conform to appropriate expectations. result: after a correct selection of a data set from the database of common Voice, the model was trained on parameters as defined by the authors of the basic model without any further changes, the model was trained in the Google Colab environment, which provides the high serious ability and fast

running with a combination of GPU powers, the model reached results of 0.54 Accuracy and 1.44 Loss.

Grid Search: The first step taken on the model itself in order to improve its accuracy is the search for the significant values for the model as stated in the grid-search subsection of the framework which explains precisely the process done on the model. Result: The model reached an accuracy of 0.85 with a loss of 0.49 so the choice of using grid search is very relevant to the model itself in order to move forward and find the ideal parameters for the model.

1x1 TDNN intermediate layers: By adding 2 intermediate layers the model has shown an improvement of 0.25 in accuracy, the loss has shown an improvement as well, the loss by using the intimidating layers is 0.60. The intermediate layers probably add some power to their study, to read more in the "1x1 TDNN intermediate layers" section.

Funnel structure in TDNN layers: The first layer in the model with the funnel structure contains 1280 neurons, from the first layer we are cutting 256 neurons between each layer and finishing in the last layer with 256 neurons, This model has shown the best results which are 0.92 accuracy with 0.27 loss.

Integrated Model: The integrated model is a model that integrates all 3 previous methods (1x1 TDNN intermediate layers, Grid Search, Funnel structure in TDNN layers). The integrated model has shown very promising results with an accuracy of 0.95 a loss of 0.2.

Final model: after all the changes that the model went through, which include all the steps we mentioned above, the final model received all the combinations of the improvements in order to reach the maximum accuracy that is available to us, in addition to this, in order to give additional auditory reinforcement to the final model, the model was trained on Data Augmentation which contains files Audio with changes such as adding background noise, different audio speed and a high pitched voice which gives the model another important learning aspect for success and accuracy on the test.
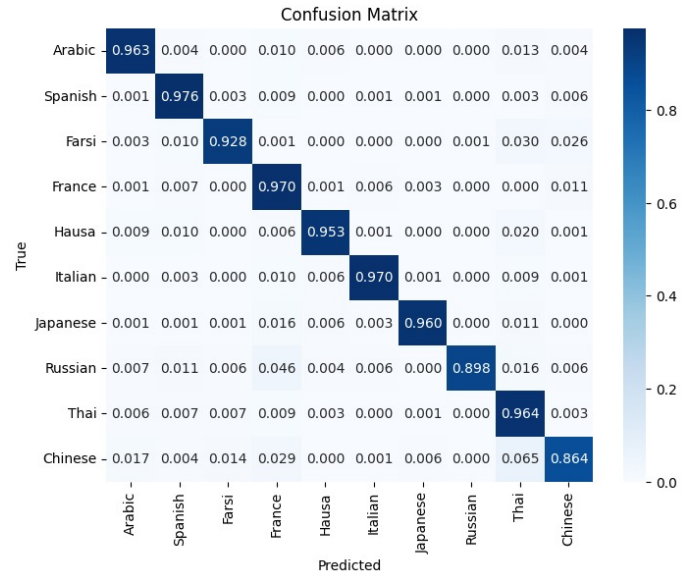

Figure 3: *Presenting a Confusion Matrix on Validation Dataset on the Final Model The Confusion Matrix is showing a very good results almost without any error.*
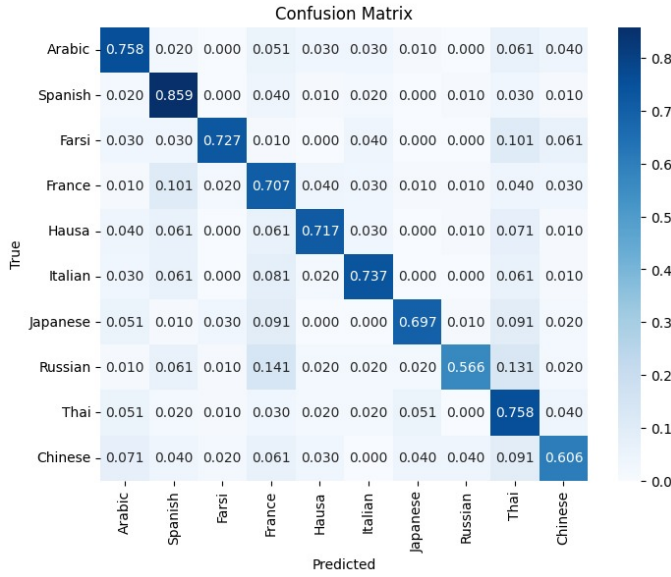
Gabi:

Figure 4: *Presenting a Confusion Matrix on test Dataset on the Final Model The Confusion Matrix is showing an improvement from the baseline model but probably also an over-fitting comparing to the results in the validation.*

Validation Set is used during model training to assess performance and tune hyperparameters. In addition it Helps in detecting overfitting during training [73]. In validation data is typically used multiple times, and models can be adjusted based on the results. The Test Set held back until the final stage and used to evaluate the model's generalization on unseen data. The Test provides an unbiased assessment of the model's performance in real-world scenarios. This final model scored an impressive 0.969 accuracy on the validation dataset. After the final model was decided, it was tested on the test dataset and received a score of 0.7 accuracy. This score is significantly higher than the baseline model (0.54), but nevertheless shows a clear OVERFITTING of the model, a problem that must be solved in follow-up studies.

We have decide to show also the following metrics: F1-score, Recall, Precision. For each meric respectivly [74, 75, 76].

7. Discussion

Gabi:

There were some limitations to the research that will be elaborated on:

1. Hardware restrictions: This research was conducted using PCs (Personal Computers), therefore there were hardware restrictions. The models in the research are powerful models and it is complicated to run them on regular PCs. To solve this problem Google Colab was used to run this model, but there are some restrictions on Google Colab as well.

2. Dataset: The Data to this research was taken from Common Voice, there wasn't exactly the data required to this research, information about the data is also very restricted and hardware playing a role here as well.

3. economic restrictions: This section is very related to the previous sections. Without an economic restrictions the previous sections could be solved and much more resources and effort could be use on this project.

Table 7: *Results Comparison Between Validation and Test for each of the 10 languages that were tested. as we can see Spanish got the best results on test and validation and the Russian language got the The least good results.*

Language	Validation	Test
Arabic	0.96	0.75
Spanish	0.97	0.86
Farsi	0.93	0.72
France	0.97	0.70
Hausa	0.95	0.71
Italian	0.97	0.73
Japanese	0.96	0.69
Russian	0.90	0.56
Thai	0.96	0.75
Chinese	0.86	0.60

Table 8: *Baseline and final model metrics comparison by Accuracy, Loss, F1 - Score, Recall, Precision*

Metrics	Baseline	Final model on validation	Final model on test
Accuracy	0.54	0.97	0.71
Loss	1.44	0.2	0.90
F1-score	-	0.94	0.72
Recall	-	0.94	0.71
Precision	-	0.95	0.74

4. Time constraints: Conducting research with complex models and large datasets requires significant computational resources and time. Due to time constraints, it was not feasible to explore alternative models or expand the dataset further. This may have limited the generalizability of the findings.

Gabi:

While language recognition systems have made significant advancements in recent years, there are still some challenges and limitations associated with their solutions. Here are a few problems that can arise:

1. Ambiguity: Some languages share similar characteristics or have significant dialectal variations, making it challenging to differentiate between them accurately. For example, distinguishing between Norwegian, Swedish, and Danish can be difficult due to their linguistic similarities.

2. Code-Switching: Code-switching refers to the practice of alternating between two or more languages within a single conversation or text. It is prevalent in multilingual communities. Language recognition systems may struggle to accurately identify the language in code-switched texts, leading to errors or misclassifications.

3. Limited training data: Building an accurate language recognition system requires a large amount of diverse and representative training data. However, obtaining such data for all languages can be a daunting task. The availability of training data for less widely spoken or low-resource languages is often limited, leading to reduced performance for those languages.

4. Uncommon or emerging languages: Language recognition systems typically perform better on widely spoken and well-established languages. However, for less commonly spo-

ken or emerging languages, the lack of data and resources can hinder accurate language identification.

5. Noise and poor audio quality: In speech-based language recognition, background noise, poor audio quality, and accents can introduce challenges. These factors can affect the performance of automatic speech recognition systems, leading to inaccuracies in language identification.

6. Limited context: Language recognition systems typically rely on short segments of text or speech for identification, which can limit the available contextual information. The absence of sufficient context may result in misclassifications, especially when distinguishing between closely related languages.

Gal:

While language recognition technology offers many benefits, it's crucial to consider the potential ethical implications:

Privacy Concerns: As language recognition often involves analyzing personal communications, there could be significant privacy concerns. It's crucial to handle such data responsibly, respecting user privacy and complying with relevant data protection regulations.

Bias: Like any AI system, language recognition systems could manifest bias, especially if the training data is skewed towards certain languages. This could result in lower accuracy for underrepresented languages, which could have various negative implications.

Misuse: The technology could potentially be misused for surveillance or other unethical purposes. It's important to develop safeguards against such misuse.

Results In the current work, we saw at the end of the process that the results from the data of the audio segments, the Spanish received the highest scores in both the test and the validation, and the Russian received the lowest score. We think that this may be due to the length of the segments of the recordings in the languages and the amount of validation in the Spanish language is higher than in the Russian language, and in the language In Russian, the length of the recording is very long

8. Conclusions & Future Work

Idan:

1. As can be seen in the results, adding 1X1 intermediate layers in the TDNN layers was very significant. This phenomenon can be due to positive reasons such as those mentioned in the section dealing with this (further processing of the data, production of more complex features and patterns, etc.). However, adding layers gives more power to the model and therefore it can "memorize" the data or find other patterns such as speaker recognition.

Idan:

2. The most significant change is the change in the number of neurons in the TDNN layers to a funnel shape. Our hypothesis is that the theoretical meaning of the funnel shape is what caused the model to improve results, but similar to conclusion 2, here too there is the possibility that due to this structure, there is a greater amount of neurons in the model and therefore it can memorize data or recognize speakers more easily.

Idan:

3. Although we made as many efforts as possible to neutralize the possibility that the model identifies speakers and not languages, for example, choosing a dataset that contains as many speakers as possible in each language (taking into account resource limitations), it is not necessary that the model does indeed identify languages and not speakers. In order to address the problems arising from sections 2 and 3, we recommend that

in follow-up studies a similar experiment be performed on a more significant amount of data and with an even greater variety of speakers and thus enable the negation of situations such as speaker identification and confirm the hypothesis that these layers help to identify languages. Furthermore, it is useful to perform tests on this model with other audio databases of the languages we used.

Idan:

4. The last significant change is adding a 1X1 TDNN intermediate layers. According to the results obtained, it is clear that there is a lot of power by adding new layers. In future work we encourage to find the optimal intermediate layers number and position.

Gabi:

We also suggest some future works that can follow our research in various topics.

The dataset for this research was taken from Common Voice, a lot of Common Voice dataset is a Validate hours, meaning data that people were verified by some criteria. For future work, we can use a "dirty" dataset Which means data that was taken without any human verification. There is a big difference between working with "clean" and "dirty" data, "clean" data is accurate, complete, consistent, and free from errors or inconsistencies, while "dirty" data contains inaccuracies, missing values, duplicates, outliers, or other issues. There are some key differences when working with "clean" and "dirty" data for example Data Accuracy, Data Completeness, Data Consistency, Data Trustworthiness, Data Preparation Efforts, Analysis Reliability, Resource Utilization, Data Interpretation. There are a lot of researches that was made on this topic here is one fore example [77].

Gabi:

By looking at the sections above "clean" data is more reliable and much more comfortable to work with but dose not represent the "real life", therefore working with "dirty" data is suggested for future works' and here are some suggestions to future works:

1. Cleaning dead parts from the audio files when using "dirty" audio files , this kind of operation can make the data shorter and leaving only the parts that are important.

2. Separate voices, This is a well-known problem that is called Speaker Recognition. Sometimes an overlap between several speakers can be Not understandable to separate the voices it can be very helpful, to read more about Speaker Recognition [78].

3. Separate languages, it can be data that can contain various of languages on each segment, understanding the way to work with this kind of data can be very interesting.

* Collecting data for a project can be very challenging due to various factors, so it must be taken into account for future works.

Idan:

Regarding to the changes of the model or the structure of the model many experiments can be done. in the following section some ideas to improve the model would be given that can be done by a future research, some of them would be related to the structure and some of them would be related to some parameters.

1. Checking different activation functions: In our research the activation function was the ReLU (rectified linear unit), but there are different activation functions that maybe some of them will work better. Activation functions can play a crucial role in neural networks by introducing non-linearity to the model's

computations. They are applied to the outputs of individual neurons or layers and determine whether the neuron should be activated or not based on the input it receives.

Activation functions primarily introduce non-linear transformations to the network, allowing it to learn complex and non-linear mappings. Without non-linear activation functions, neural networks would be limited to representing only linear relationships between input and output. The ability to model complex decision boundaries is a critical aspect of activation functions, enabling neural networks to tackle tasks that involve non-linear relationships and intricate patterns.

Another crucial role of activation functions is in gradient propagation during backpropagation [79], which is essential for training the network. Well-behaved activation functions ensure that gradients can be effectively propagated through the network layers, enabling the model to learn from the training data. The choice of activation functions impacts the stability and convergence of the network during training, with proper functions preventing issues like vanishing or exploding gradients.

Activation functions also contribute to the normalization and regularization of neural networks. Functions like Batch Normalization or Layer Normalization help reduce internal covariate shift [80], improving network stability and convergence. They can also mitigate overfitting and improve the model's generalization capabilities.

Different activation functions possess varying properties, such as saturation regions, slopes, or thresholds, which affect the network's expressive power. The choice of activation functions allows for the customization of the network's ability to represent and approximate different types of functions, aligning with the requirements of the task at hand.

Lastly, activation functions provide control over the activation levels of neurons, allowing the network to emphasize or suppress specific input ranges. This control can be valuable in scenarios where certain input ranges require special attention or handling.

Overall, activation functions are vital components of neural networks, introducing non-linearity, enabling the modeling of complex decision boundaries, facilitating gradient propagation, contributing to normalization and regularization, providing expressive power, and allowing for activation control. Careful consideration and selection of appropriate activation functions are crucial to ensuring the network's effectiveness in capturing and representing the desired patterns and relationships in the data .

* Many researches were some about the importance of the activation functions and this is an example to one of them [81].

2. Early stopping: Early stopping is a technique used during the training of a neural network to prevent overfitting and determine the optimal number of training iterations or epochs. It involves monitoring the performance of the model on a validation set and stopping the training process when the performance on the validation set starts to degrade or reach a plateau.

The basic idea behind early stopping is that as the model continues to train, it gradually learns to fit the training data more closely. However, if training continues for too long, the model may start to overfit the training data and lose its ability to generalize to new, unseen data. Early stopping helps address this issue by stopping the training process at an earlier stage, before overfitting occurs. To read more about early stopping [82]

3. Ensemble methods: Ensemble methods are techniques that involve combining the predictions of multiple individual models to form a stronger, more robust prediction. The idea behind ensemble methods is that by leveraging the diversity and complementary strengths of multiple models, it is possible to improve the overall predictive accuracy and generalization of the ensemble [83]. there are several types of Ensemble methods techniques that can be tested in a future researches we suggest the Averaging methods:

- Averaging methods: Averaging methods involve training multiple models independently and averaging their predictions. This can be done by taking the average of their outputs (for regression problems) or by taking a majority vote or weighted vote (for classification problems). Common averaging methods include simple averaging, weighted averaging, and stacking. To read more about the different techniques [84].

The success of ensemble methods relies on the diversity of the individual models within the ensemble. The diverse models should be able to capture different aspects or perspectives of the underlying data, increasing the ensemble's ability to generalize well.

Ensemble methods often improve generalization by reducing the impact of overfitting. By combining multiple models, ensemble methods help reduce model bias and variance, leading to more stable and reliable predictions. The ensemble is less likely to be affected by the idiosyncrasies of a single model and is more likely to make accurate predictions on unseen data.

in conclusion ensemble methods offer a powerful approach to improving predictive accuracy, reducing overfitting, and increasing the robustness of models by combining the strengths of multiple models and therefore we suggest to test this. There are many articles that were written on the idea of ensemble learning, here is one of them [85].
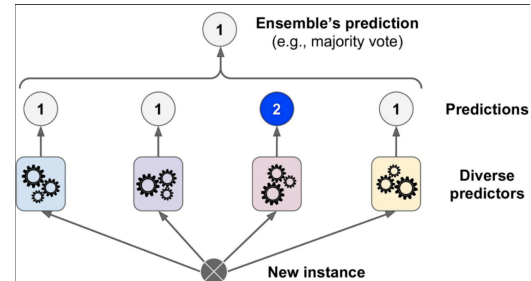

Figure 5: *An example to ensemble model that getting several results from different models and aggregate it to 1 answer*

Gabi:

We also suggest that some future works would be regarding to a different methods There are various ways to try to solve the language recognition problem , here are some different methods that could work:

1. N-gram Language Models: N-gram language models are statistical models that estimate the probability of encountering a sequence of N consecutive words in a given language. These models capture the statistical properties of languages and can be used to compare the likelihood of a text belonging to different languages based on the observed N-grams. The language with the highest probability is considered the recognized language [86].

2. Machine Learning with Text Features: Machine learning algorithms can be trained on labeled datasets to learn patterns and characteristics of different languages. Features such

as word frequencies, character n-grams, syntactic features, or statistical measures can be extracted from the text samples and used as input for classification algorithms like decision trees, random forests, support vector machines (SVM), or deep learning models [87].

3. Neural Networks with another Word Embeddings: Neural network architectures, such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs), can be employed for language recognition [88]. Word embeddings, such as Word2Vec [89] or GloVe [90], can be used to represent words as dense vector representations, which capture semantic and syntactic information. These embeddings can be fed into the neural network models to learn language patterns and make predictions [91].

9. References

[1] J. Wu, "Introduction to convolutional neural networks," *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, no. 23, p. 495, 2017.

[2] R. Zhao, J. Wang, R. Yan, and K. Mao, "Machine health monitoring with lstm networks," in *2016 10th international conference on sensing technology (ICST)*. IEEE, 2016, pp. 1–6.

[3] N. Basnin, L. Nahar, and M. S. Hossain, "An integrated cnn-lstm model for bangla lexical sign language recognition," in *Proceedings of International Conference on Trends in Computational and Cognitive Engineering: Proceedings of TCCE 2020*. Springer, 2020, pp. 695–707.

[4] S. M. Siniscalchi, J. Reed, T. Svendsen, and C.-H. Lee, "Universal attribute characterization of spoken languages for automatic spoken language recognition," *Computer Speech & Language*, vol. 27, no. 1, pp. 209–227, 2013.

[5] P. Sahu, M. Dua, and A. Kumar, "Challenges and issues in adopting speech recognition," *Speech and Language Processing for Human-Machine Communications: Proceedings of CSI 2015*, pp. 209–215, 2018.

[6] T. Stafylakis, J. Rohdin, O. Plchot, P. Mizera, and L. Burget, "Self-supervised speaker embeddings," *arXiv preprint arXiv:1904.03486*, 2019.

[7] A. Sizov, K. A. Lee, and T. Kinnunen, "Direct optimization of the detection cost for i-vector-based spoken language recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 3, pp. 588–597, 2017.

[8] R. Featherstone, "A beginner's guide to 6-d vectors (part 1)," *IEEE robotics & automation magazine*, vol. 17, no. 3, pp. 83–94, 2010.

[9] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[10] G. S. Morrison, "A comparison of procedures for the calculation of forensic likelihood ratios from acoustic–phonetic data: Multivariate kernel density (mvkd) versus gaussian mixture model–universal background model (gmm–ubm)," *Speech Communication*, vol. 53, no. 2, pp. 242–256, 2011.

[11] M. Radfar, A. Mouchtaris, and S. Kunzmann, "End-to-end neural transformer based spoken language understanding," *arXiv preprint arXiv:2008.10984*, 2020.

[12] R. D. P. D. Najim Dehak, Patrick Kenny and P. Ouellet, "Front-end factor analysis for speaker verification," 2011.

[13] P. Matějka, O. Glembek, F. Castaldo, M. J. Alam, O. Plchot, P. Kenny, L. Burget, and J. Černocky, "Full-covariance ubm and heavy-tailed plda in i-vector speaker verification," in *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2011, pp. 4828–4831.

[14] M. Rouvier and B. Favre, "Speaker adaptation of dnn-based asr with i-vectors: Does it actually adapt models to speakers?" in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[15] T. N. H. M. O. G. J. M. P. Matejk, L. Zhang and B. Zhang, "Neural network bottleneck features for language identification," 2014.

[16] D. R. F. Richardson and N. Dehak, "Deepneural network approaches to speaker and language recognition," 2015.

[17] N. Adaloglou, T. Chatzis, I. Papastratis, A. Stergioulas, G. T. Papadopoulos, V. Zacharopoulou, G. J. Xydopoulos, K. Atzakas, D. Papazachariou, and P. Daras, "A comprehensive study on sign language recognition methods," *arXiv preprint arXiv:2007.12530*, vol. 2, no. 2, 2020.

[18] S. R. Eddy, "Accelerated profile hmm searches," *PLoS computational biology*, vol. 7, no. 10, p. e1002195, 2011.

[19] P. Heracleous, K. Takai, K. Yasuda, Y. Mohammad, and A. Yoneyama, "Comparative study on spoken language identification based on deep learning," in *2018 26th European signal processing conference (EUSIPCO)*. IEEE, 2018, pp. 2265–2269.

[20] B. Luo, Y. Feng, Z. Wang, S. Huang, R. Yan, and D. Zhao, "Marrying up regular expressions with neural networks: A case study for spoken language understanding," *arXiv preprint arXiv:1805.05588*, 2018.

[21] S. Mukherjee, N. Shivam, A. Gangwal, L. Khaitan, and A. J. Das, "Spoken language recognition using cnn," in *2019 International Conference on Information Technology (ICIT)*. IEEE, 2019, pp. 37–41.

[22] L. Zhang, D. Ma, X. Zhang, X. Yan, and H. Wang, "Graph lstm with context-gated mechanism for spoken language understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 9539–9546.

[23] B. M. L. Srivastava, M. Maouche, M. Sahidullah, E. Vincent, A. Bellet, M. Tommasi, N. Tomashenko, X. Wang, and J. Yamagishi, "Privacy and utility of x-vector based speaker anonymization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2383–2395, 2022.

[24] D. Raj, D. Snyder, D. Povey, and S. Khudanpur, "Probing the information encoded in x-vectors," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 726–733.

[25] S. Wang, Z. Huang, Y. Qian, and K. Yu, "Discriminative neural embedding learning for short-duration text-independent speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1686–1696, 2019.

[26] L. Singh and M. Chee, "Rise and fall: effects of tone and intonation on spoken word recognition in early childhood," *Journal of Phonetics*, vol. 55, pp. 109–118, 2016.

[27] R. Shah, B. M. Joshi, J. Shah, M. Patel, A. Rana, and R. Roy, "Summary of spoken indian languages classification using ml and dl," in *2022 6th International Conference on Electronics, Communication and Aerospace Technology*. IEEE, 2022, pp. 1021–1025.

[28] A. M. G. S. D. P. S. K. David Snyder, Daniel Garcia-Romero, "Spoken language recognition using x-vectors," 2018.

[29] J. G.-D. D. T. T. J. G.-R. Ruben Zazo*, Alicia Lozano-Diez, "Language identification in short utterances using long short-term memory (lstm) recurrent neural networks," 2016.

[30] A. McCree, "Multiclass discriminative training of i-vector language recognition," 2016.

[31] R. J. W. B. L. P. M. E. W. K. R. Shubham Toshniwal, Tara N. Sainath, "Multilingual speech recognition with a single end-to-end model," 2018.

[32] A. M. Daniel Garcia-Romero, "Stacked long-term tdnn for spoken language recognition," 2016.

[33] G. S. Alan McCree, David Snyder and D. Garcia-Romero, "Language recognition for telephone and video speech: The jhu hltcoe submission for nist lre17," 2018.

[34] A. L. M. M. N. S. Yun Lei, Luciana Ferrer, "Application of convolutional neural networks to language identification in noisy conditions," 2014.

[35] L. B. O. G. David Martínez, Oldrich Plchot and P. M., "Language recognition in ivectors space," 2011.

[36] NIST, "Nist lre 2009 evaluation plan," 2009.

[37] T. A. a. Jorgen Valk, "Voxlingua107: A dataset for spoken language recognition," 2020.

[38] V. K. M. K. M. Gundeep Singh, Sahil Sharma and M. Masud, "Voxlingua107: A dataset for spoken language recognition," 2021.

[39] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[40] S. M. Hunt and R. Bhopal, "Self report in clinical and epidemiological studies with non-english speakers: the challenge of language and culture," *Journal of Epidemiology & Community Health*, vol. 58, no. 7, pp. 618–622, 2004.

[41] B. W. Fortson IV, *Indo-European language and culture: An introduction.* John Wiley & Sons, 2011.

[42] R. Hetzron and A. S. Kaye, "Semitic languages," in *The world's major languages.* Routledge, 2009, pp. 564–572.

[43] E. J. Henderson, "The topography of certain phonetic and morphological characteristics of south east asian languages," *Lingua*, vol. 15, pp. 400–434, 1965.

[44] A. Bakowski, V. Dekỳš, L. Radziszewski, and Z. Skrobacki, "Validation of traffic noise models," in *AIP Conference Proceedings*, vol. 2077, no. 1. AIP Publishing LLC, 2019, p. 020005.

[45] C.-H. Lee, "Principles of spoken language recognition," *Springer Handbook of Speech Processing*, pp. 785–796, 2008.

[46] D. Garcia-Romero and A. McCree, "Stacked long-term tdnn for spoken language recognition." in *Interspeech*, 2016, pp. 3226–3230.

[47] M. Kang and J. Tian, "Machine learning: Data pre-processing," *Prognostics and Health Management of Electronics: Fundamentals, Machine Learning, and the Internet of Things*, pp. 111–130, 2018.

[48] F. Mirus, T. C. Stewart, and J. Conradt, "The importance of balanced data sets: Analyzing a vehicle trajectory prediction model based on neural networks and distributed representations," in *2020 International Joint Conference on Neural Networks (IJCNN).* IEEE, 2020, pp. 1–8.

[49] G. J. Bowden, H. R. Maier, and G. C. Dandy, "Optimal division of data for neural network models in water resources applications," *Water resources research*, vol. 38, no. 2, pp. 2–1, 2002.

[50] N. Saunshi, A. Gupta, and W. Hu, "A representation learning perspective on the importance of train-validation splitting in meta-learning," in *International Conference on Machine Learning.* PMLR, 2021, pp. 9333–9343.

[51] E. D. Maughan and L. J. Schneider, "Comparing apples with apples: The importance of clean data," *NASN School Nurse*, vol. 34, no. 2, pp. 107–108, 2019.

[52] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[53] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Sixteenth annual conference of the international speech communication association*, 2015.

[54] T. Hasija, V. Kadyan, and K. Guleria, "Out domain data augmentation on punjabi children speech recognition using tacotron," in *Journal of Physics: Conference Series*, vol. 1950, no. 1. IOP Publishing, 2021, p. 012044.

[55] N. Dvornik, J. Mairal, and C. Schmid, "On the importance of visual context for data augmentation in scene understanding," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 6, pp. 2014–2028, 2019.

[56] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *arXiv preprint arXiv:1803.10963*, 2018.

[57] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[58] P. Probst, B. Bischl, and A.-L. Boulesteix, "Tunability: Importance of hyperparameters of machine learning algorithms," *arXiv preprint arXiv:1802.09596*, 2018.

[59] H. Yu, P. M. Kim, E. Sprecher, V. Trifonov, and M. Gerstein, "The importance of bottlenecks in protein networks: correlation with gene essentiality and expression dynamics," *PLoS computational biology*, vol. 3, no. 4, p. e59, 2007.

[60] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3159–3166.

[61] C. H. Li and C. Lee, "Minimum cross entropy thresholding," *Pattern recognition*, vol. 26, no. 4, pp. 617–625, 1993.

[62] X. Chen, S. Kar, and D. A. Ralescu, "Cross-entropy measure of uncertain variables," *Information Sciences*, vol. 201, pp. 53–60, 2012.

[63] R. L. Rodriguez and P. I. Joel, "Filipino emotion classification in speech signals based on audio features and transcribed text," *College of Computer Studies De La Salle University, Manila Philippines.*

[64] E. Demirel, S. Ahlbäck, and S. Dixon, "Automatic lyrics transcription using dilated convolutional neural networks with self-attention," in *2020 International Joint Conference on Neural Networks (IJCNN).* IEEE, 2020, pp. 1–8.

[65] Y.-Q. Yu and W.-J. Li, "Densely connected time delay neural network for speaker verification." in *INTERSPEECH*, 2020, pp. 921–925.

[66] R. Madaan, D. Maturana, and S. Scherer, "Wire detection using synthetic data and dilated convolutional networks for unmanned aerial vehicles," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2017, pp. 3487–3494.

[67] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth annual conference of the international speech communication association*, 2015.

[68] D. T. Toledano, M. P. Fernández-Gallego, and A. Lozano-Diez, "Multi-resolution speech analysis for automatic speech recognition using deep neural networks: Experiments on timit," *PLoS one*, vol. 13, no. 10, p. e0205355, 2018.

[69] S. Li, X. Lu, R. Takashima, P. Shen, T. Kawahara, and H. Kawai, "Improving ctc-based acoustic model with very deep residual time-delay neural networks." in *Interspeech*, 2018, pp. 3708–3712.

[70] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning.* PMLR, 2014, pp. 647–655.

[71] X. Miao, I. McLoughlin, W. Wang, and P. Zhang, "D-mona: A dilated mixed-order non-local attention network for speaker and language recognition," *Neural Networks*, vol. 139, pp. 201–211, 2021.

[72] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 ieee information theory workshop (itw).* IEEE, 2015, pp. 1–5.

[73] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[74] Z. C. Lipton, C. Elkan, and B. Narayanaswamy, "Thresholding classifiers to maximize f1 score," *arXiv preprint arXiv:1402.1892*, 2014.

[75] J. G. Raaijmakers and R. M. Shiffrin, "Models for recall and recognition," *Annual review of psychology*, vol. 43, no. 1, pp. 205–234, 1992.

[76] M. Kane, "The precision of measurements," *Applied measurement in education*, vol. 9, no. 4, pp. 355–379, 1996.

[77] S. Porwal and D. Vora, "A comparative analysis of data cleaning approaches to dirty data," *International Journal of Computer Applications*, vol. 62, no. 17, pp. 30–34, 2013.

[78] J. P. Campbell, "Speaker recognition: A tutorial," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1437–1462, 1997.

[79] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[80] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[81] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[82] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 2002, pp. 55–69.

[83] Z.-H. Zhou, "Ensemble methods," *Combining pattern classifiers. Wiley, Hoboken*, pp. 186–229, 2014.

[84] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[85] T. G. Dietterich *et al.*, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, no. 1, pp. 110–125, 2002.

[86] B. Roark, M. Saraclar, and M. Collins, "Discriminative n-gram language modeling," *Computer Speech & Language*, vol. 21, no. 2, pp. 373–392, 2007.

[87] Z. Halim, M. Waqar, and M. Tahir, "A machine learning-based investigation utilizing the in-text features for the identification of dominant emotion in an email," *Knowledge-based systems*, vol. 208, p. 106443, 2020.

[88] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *arXiv preprint arXiv:1702.01923*, 2017.

[89] K. W. Church, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[90] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[91] S. Settle and K. Livescu, "Discriminative acoustic word embeddings: Tecurrent neural network-based approaches," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 503–510.