"Speech recognition using CNN, autoencoder followed by SVM and Shallow Inception"

Konstantinos Panagiotakis†

***Abstract*—Speech recognition is one of the cornerstones of human-machine interaction. In this letter, architectures such as CNN, autoencoders followed by SVM and Inception were explored. The models were trained on a reference dataset for small-footprint keyword spotting (KWS), more in particular on a subset of 9 words out of the original 30. The best performing architecture in terms of accuracy was the shallow Inception model with an accuracy of 89%, on the other hand, the best performing architecture in terms of memory allocation and training times was a simple CNN architecture with dropout and L2 regularization with a memory allocation 412.03 MB and a training time of 178 s. This work is aimed to aid and further develop the field of speech recognition by exploring various architectures in the field of ML.**

***Index Terms*—Speech Recognition, KWS, CNN, Autoencoder, SVM, Inception.**

I. Introduction

Speech-related technologies are gaining popularity with the rise of mobile devices. For instance, Google Now, Apple's Siri, Microsoft's Cortana, and Amazon's Alexa all utilize speech recognition for interaction. [1] Google's "Ok Google" [2] feature enables hands-free speech recognition on mobile devices with a small memory footprint and low computational power, utilizing a Deep Neural Network (DNN) that outperforms traditional Hidden Markov Models for keyword spotting and offers flexibility in adjusting model size. [3] [4] Previous work on this field was tailored more towards parameter tuning of CNN architectures, [5] and the exploration of regular Deep Neural Network (DNN) and the more sofisticated Connectionist Temporal Classifier (CTC) architecture explored in [6].

The problem at hand delt in this letter is the one of speech recognition for single words, also known as KWS. In this letter we further explore novel architectures built on the already explored CNN and DNN architectures with exploration of autoencoders followed by an SVM, Inception V4 and a simplified version of the Inception architecture. Autoencoders were used for data denoising and compression. When applied to MFCC image datasets, they consist of an encoder and a decoder. The encoder compresses input data into a lower-dimensional representation, while the decoder reconstructs the original input from this compressed representation. By minimizing reconstruction error during training, autoencoders learn to filter out noise and capture essential features. The decoder can generate denoised images, and the compressed representations where used for the classification using an SVM. This approach enabled to design a model trained on compressed data without the loss of performance accuracy while improving the memory allocation and training times.

Another explored approach was the Inception v4. Inception v4 is a deep convolutional neural network renowned for image classification tasks. It employs the "Inception module" to efficiently capture features at multiple scales within the same layer. This architecture excels in learning hierarchical features from images, enabling robust and accurate classification. It's a powerful model for image classification, adept at leveraging hierarchical features from input images. During the exploration of this architecture it was noticeable that the number of parameters in the model was extremely high and the padding shape to fit the data in this model also resulted on enlarged arrays filled with regions of no information. I therefore explored a simplified approach where I use a version of the Inception structure with only one stem, inception, reduction, average pooling and dense layer, instead of the regular Inception V4 architecture. This simplified architecture was chosen so that less padding would be applied to the original data and so that less parameters would be fed into the model, reducing the parameters from 41 million to about 1.4 million. However, this came at the cost of limiting the capturing of features at different scales that different kernel sizes within each of the inception blocks allow to do.

In this paper the main goal is to find new ways of treating the issue of speech recognition with more advanced architectures aimed at improving accuracies, memory allocation, training times while reducting the model parameters.

Further work can be aimed at the combination of these tools in obtaining a more robust architecture to do all of the above. An idea could be to use autoencoders for feature extraction, denoising and compression of the data before feeding it into a simplified Inception v4 network. This would allow for data feature extrection and compression before feeding the batched datasets into a deep model like the Inception v4 that aims at efficiently capturing features at multiple scales within the same layer.

This letter is structured as follows. In Section II, Related Work, we describe the state of the art models used in previous iterations of speech recognition research, the system and data models are respectively presented in Sections III, Processing Pipeline and IV, Signals and Features. The proposed models architectures is detailed in Section V, Learning Framework and its performance evaluation is carried out in Section VI, Results. Concluding remarks are provided in Section VII.

†Department of Physics of Data, University of Padova, email: {konstantinos.panagiotakis}@studenti.unipd.it

II. Related Work

In this section I outline a comprehensive review of the literature on keyword spotting (KWS) systems, by delving into seminal papers that have significantly shaped the field, examining their contributions, key findings, and areas for further improvement.

Initially, the groundbreaking work by [5] underscored the remarkable effectiveness of Deep Neural Networks (DNNs) in KWS tasks, surpassing the performance of traditional Hidden Markov Model (HMM) systems. This paper not only demonstrated the superior capabilities of DNNs in capturing complex acoustic features for precise keyword recognition but also sheds light on their potential limitations in capturing local correlations within input data and addressing translational variance in speech signals. Building upon these insights, subsequent studies by [6] introduced Convolutional Neural Networks (CNNs) as a promising alternative to DNNs for acoustic modeling in KWS. These studies underscored the advantages of CNNs in capturing local correlations in input data while significantly reducing model size compared to DNNs, thereby addressing some of the limitations observed in DNN-based systems. However, despite these advancements, challenges persisted in existing CNN architectures, particularly in terms of computational efficiency and parameter optimization for specific KWS tasks. Moreover, the reliance on keyword-specific speech data presented significant hurdles in acquiring sufficient training samples and managing the associated high costs of data acquisition. In response to these challenges, their work proposes a novel KWS system that integrates both DNNs and Connectionist Temporal Classifier (CTC) on power-constrained small-footprint mobile devices, aiming to maximize the utilization of a vast general corpus from continuous speech recognition. Another important approach to solving the speech recognition task was done by [7]. The study by Chia-Hsuan Li et al. from the College of Electrical Engineering and Computer Science at National Taiwan University introduces a novel listening comprehension task named Spoken SQuAD, aiming to address the challenge of understanding spoken content in machine comprehension. While significant progress has been made in machine comprehension (MC) on text, accessing large collections of spoken content poses more difficulties for humans, making it imperative to develop machines capable of automatically understanding spoken content. The authors propose Spoken SQuAD as an extraction-based Spoken Question Answering (SQA) task, highlighting the catastrophic impact of speech recognition errors on machine comprehension. They explore various approaches to mitigate this impact, including utilizing different subword units like phoneme and syllable sequence embeddings to enhance word embeddings and subsequently improve comprehension accuracy. The study provides insights into the challenges of MC on spoken content and offers innovative solutions to enhance machine comprehension in real-world scenarios with ASR errors. As shown in this comprehensive review of the literature on keyword spotting (KWS) systems highlights the evolution from DNN-based approaches to the integration of novel techniques like CNNs and CTC, while also acknowledging the emerging challenges in computational efficiency and data acquisition costs, alongside the innovative efforts in addressing speech recognition errors through advanced methodologies such as Spoken SQuAD and subword unit embeddings.

III. Processing Pipeline

A. CNN

The research commenced with a primary focus on developing a Convolutional Neural Network (CNN) model tailored specifically for the classification of audio signals. This endeavor involved exploring various types of signals to be classified, as detailed in section IV on SIGNALS AND FEATURES. The architecture of the CNN model encompasses convolutional and pooling layers, strategically designed to learn hierarchical features directly from the input spectrogram representations of the audio signals. These layers operate in tandem to extract intricate patterns and structures embedded within the audio data, leveraging the CNN's innate capability for hierarchical feature learning. Subsequently, fully connected layers are employed to interpret the learned features and facilitate accurate predictions. During the training phase, the model optimizes its parameters using the Adam optimizer and minimizes the categorical cross-entropy loss function, both widely recognized as standard choices for classification tasks.

B. CNN with Dropout and L2 Regularization

The CNN model with dropout and L2 regularization is constructed to classify Mel Frequency Cepstral Coefficients (MFCC) representation of audio signals effectively. The architecture consists of convolutional layers followed by max-pooling layers to extract hierarchical features from spectrogram representations of the input audio signals similar to the vanilla CNN approach described in the previous section. Dropout layers are integrated to prevent overfitting by randomly deactivating a fraction of neurons during training. L2 regularization is applied to penalize large weights and encourage smoother model outputs, enhancing generalization. The model is then flattened and connected to fully connected layers for classification. The Adam optimizer is utilized with a specified learning rate, and categorical cross-entropy loss is employed as the optimization criterion.

C. Autoencoder and SVM Classifier

The second part of the research focused on employing an autoencoder followed by a Support Vector Machine (SVM) classifier for the classification of audio signals. The autoencoder is designed to learn a compressed representation of the input MFCC representations of audio signals through an encoder-decoder architecture. The encoder consists of convolutional and max-pooling layers, followed by a flattening layer and a dense layer to produce a low-dimensional latent representation (code). Conversely, the decoder comprises dense and

transpose convolutional layers to reconstruct the input from the code. The autoencoder is trained using Mean Squared Error (MSE) loss and Adamax optimizer. Once trained, the encoder's weights are saved and restored for subsequent use. The latent representation (code) learned by the autoencoder is then fed into a vanilla Support Vector Machine (SVM) classifier for classification. The SVM classifier is trained on the flattened decoded data and predicts the class labels for the test data. This combined approach offers a framework for leveraging the representation learning capabilities of autoencoders followed by the discriminative power of SVM classifiers for audio signal classification tasks.

D. Simplified Inception v4 Architecture

The final part of the research introduces a simplified version of the Inception v4 architecture customized for audio signal classification. The vanilla Inception v4 structure exels both in feature extraction and hierarchical representation learning. Its depth, numerous branches and various kernel sizes allow the Inception v4 to explore features at different scales. In our case, excessive padding in image inputs was applied hence diminishing the effective receptive field, diluting relevant information, increasing computational costs, and leading to overemphasis on background features, adversely affecting the model's training and generalization performance. The vanilla Inception v4 model had poor performances in terms of classification metrics and training times, therefore a new architecture was applied. The surface area dimensionality of our dataset did not allow for repeated reshaping and reduction through pooling and convolutions throughout multiple layers and various kernel sizes, therefore a decision was made to remove the Inception B and C blocks and the reduction B block, and keep the kernel size small. This came at the cost of hierarchical feature learning as deep inception and reduction blocks equipped with various kernel sizes tend to explore. This simplified Inception architecture focuses on computational efficiency and streamlined design for audio signal classification with fewer branches, smaller, uniformly sized kernels, and simpler convolutional structures in both the stem, inception and reduction blocks with the trade-off of diminishing the hierarchical feature learning process from spectrogram representations. In this simplified Inception architecture, the number of parameters decreased significantly from 41 million to 1.4 million, leading to a more efficient and streamlined model for audio signal processing. The stem block, similar to the original Inception-v4 design, employs convolutional layers with batch normalization and max-pooling to extract fundamental features from input spectrograms. Inception blocks incorporate parallel convolutional pathways with varying receptive field sizes, enabling the capturing of diverse features. The model's training pipeline utilizes the Adam optimizer with categorical cross-entropy loss, facilitating effective learning of discriminative features from the spectrogram data. This tailored simplified Inception architecture offers a robust framework for audio signal classification, capitalizing on its ability to learn complex features and patterns inherent in spectrogram representations at the cost of hiererarchical feature learning while ensuring computational efficiency.

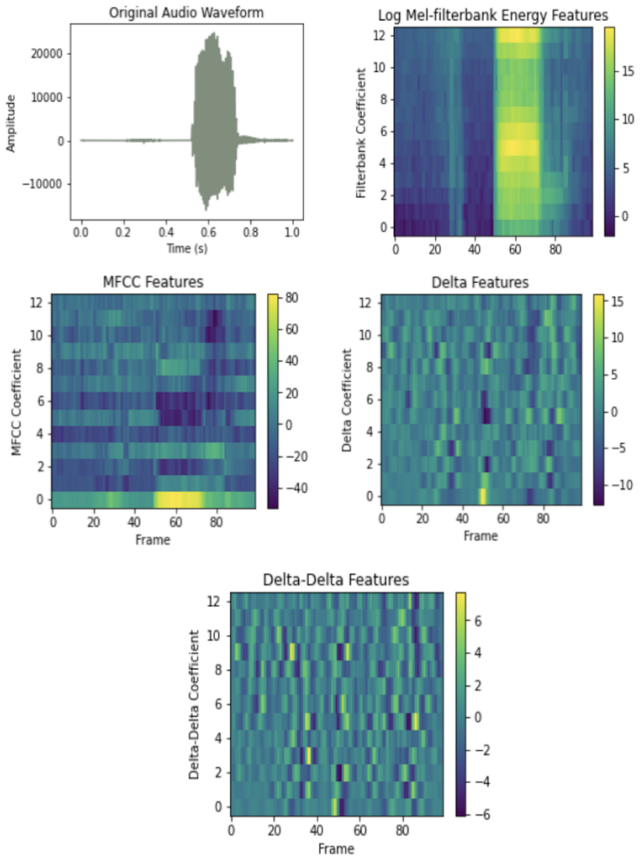

Fig. 1: The figure displays various acoustic features extracted from the original audio waveform. Subplots depict the original audio waveform, or signal, Log Mel-filterbank energy, Mel Frequency Cepstral Coefficients (MFCC), Delta, and Delta-Delta features, providing insights into the energy distribution, spectral characteristics, and temporal dynamics of the audio signal.

IV. Signals and Features

The dataset used serves as a pivotal reference in the realm of small-footprint keyword spotting (KWS), offering researchers and practitioners a rich resource for developing and evaluating KWS systems. Released in August 2017, this dataset comprises 65,000 succinct one-second-long utterances encompassing a diverse set of 30 distinct words. What makes this dataset particularly valuable is its varied source, as it originates from thousands of different individuals. Each word in the dataset is saved as a .wav file, where the signal is present. The signal was transformed into its respective Mel Frequency Cepstral Coefficients (MFCC) representation. Each word in the dataset was saved as an array of shape N, 99, 13, 1 where N is the number of samples, or instances for a specific word, 99 is the dimension in time space and 13 is the dimension in the MFCC space. Padding was applied to the

dataset and the input was reshaped from N, 99, 13, 1 to N, 100, 20, 1. This allowed for dimension matching and model compatibility of the data especially with the autoencoder and inception models where multiple convolutional and pooling steps are involved. In the vanilla CNN and in the CNN with dropout and L2 regularization padding was not applied.

The dataset was batched, cached and also shuffled. Batching, caching, and shuffling are essential techniques used in dataset processing pipelines to enhance efficiency and improve model training. Batching involves grouping multiple data samples into batches, which helps in parallelizing computations and optimizing memory usage during training. Caching is a powerful feature that allows intermediate dataset results to be stored, reducing redundant computations and speeding up subsequent data loading. Shuffling randomly reorders the dataset samples, which prevents the model from learning sequential patterns and ensures better generalization during training. These operations collectively optimize the data pipeline, enabling smoother and more efficient model training processes, especially when dealing with large datasets.

Each feature vector was calculated from the conversion of raw audio signals into MFCCs. Single audio signals were transformed into their respective MFCCs through a series of well-defined steps. First, the signal was segmented into short frames to simplify analysis, with a standard frame length of 25 milliseconds used. Next, the power spectrum of each frame was computed to understand the frequency content of the signal over time. Following this, a Mel filterbank was applied to the power spectrum, with different filterbanks capturing energy within specific frequency ranges. Subsequently, the Discrete Cosine Transform (DCT) is applied to decorrelate the filterbank energies and compact their representation, with only a subset of DCT coefficients retained to discard redundant information and improve performance. A full feature vector was then calculated by assembling the MFCCs, Delta coefficients, Delta-Delta coefficients, and energy coefficients (signal, Delta, and Delta-Delta), each extracted from the corresponding feature matrices and concatenated into a single vector for each frame, plus the signal, Delta and Delta-Delta energy coefficients.

One hot encoding was applied to each label to convert categorical data into a numerical format that was suitable for training machine learning models, ensuring that the model can effectively interpret and learn from the categorical information represented by the labels. One hot encoding was used instead of vanilla label encoding because it ensured that the model would not interpret ordinal relationships amongst categories.

The models were trained on a subset of 9 words out of the 30 available words from the dataset so that the training arrays would not be so computationally expensive. The dataset underwent a systematic division into training and testing subsets, ensuring a balanced representation across classes. Subsequently, the dataset was reshaped into features and corresponding labels, with an additional array representing signal data. A randomized selection of indices ensured equitable distribution of samples across classes, followed by shuffling to introduce

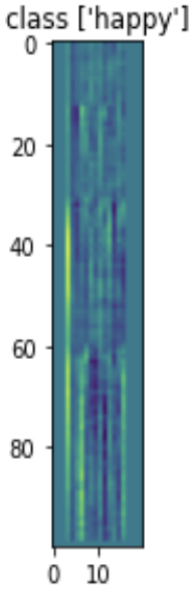

Fig. 2: The figure displays a padded sample of an mfcc signal, this is the input in each of the models, a tensor of N, 99, 13, 1 dimensions, padded to N, 100, 20, 1. This tensor represents a batch of N samples, each containing a sequence of 99 time samples. Within each time sample, there are 13 features representing the Mel Frequency Cepstral Coefficients (MFCC), and each MFCC coefficient is represented in a single channel.

randomness into the dataset. The partitioning into training and testing subsets was achieved using the train˙test˙split function from the scikit-learn library, with a designated test size of 20% to evaluate model performance. The encoded labels were decoded back to their original categorical form for both training and testing sets, thereby preserving the interpretability of the results.

V. Learning Framework

A. CNN Architecture

The convolutional neural network (CNN) model discussed in this study consists of two Conv2D layers, each followed by a MaxPooling2D layer, and two Dense layers. The first Conv2D layer applies 32 filters of size (3, 3), resulting in an output shape of (None, 97, 11, 32). Subsequently, the MaxPooling2D layer reduces the spatial dimensions, resulting in an output shape of (None, 48, 5, 32). The second Conv2D layer applies 64 filters of size (3, 3), producing an output shape of (None, 46, 3, 64). Another MaxPooling2D layer further reduces the spatial dimensions, resulting in an output shape of (None, 23, 1, 64). The Flatten layer reshapes the output into a one-dimensional vector. The first Dense layer consists of 64 neurons, followed by the second Dense layer with 9 neurons, corresponding to the number of output classes.

B. CNN with dropout and L2 regularization Architecture

A CNN with dropout and L2 regularization is then studied in this letter. This model incorporates dropout and L2

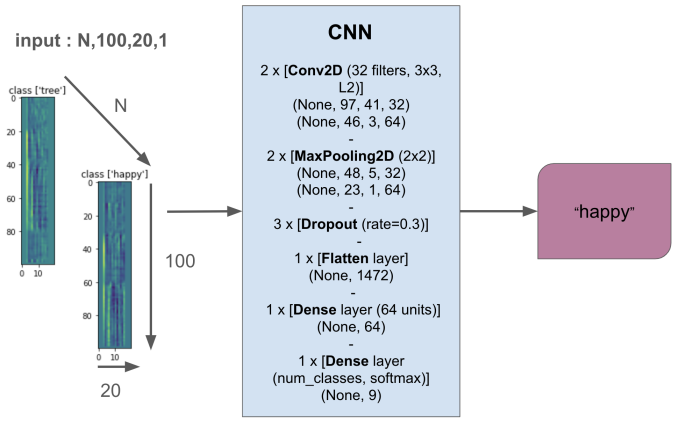

Fig. 3: The figure displays the CNN model, augmented with dropout and regularization techniques, offers a robust framework for audio classification tasks. With convolutional and pooling layers followed by dropout layers to prevent overfitting, the model ensures effective feature extraction and generalization.

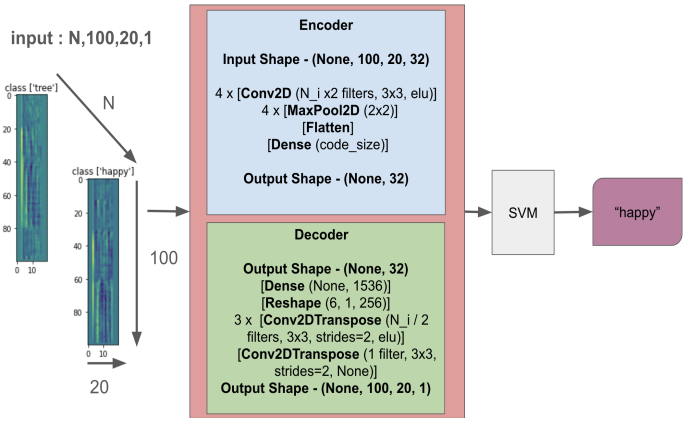

Fig. 4: The figure displays the used autoencoder that compresses input spectrogram representations into a low-dimensional latent space, which is then fed into an SVM classifier for accurate classification. This integrated method effectively leverages the autoencoder's representation learning and SVM's discriminative power for audio signal classification.

regularization techniques to mitigate overfitting and enhance generalization performance. The model architecture is very similar to the vanilla CNN described above. The main difference lies in a dropout layer applied after each of the Max Pooling operations, which randomly sets a fraction of input units to zero during training, effectively introducing noise and preventing over-reliance on specific features. The way the model flattens the layers using a flatten layer is also the same as the vanilla CNN. After the Dense layer, another dropout layer is applied to further prevent over-reliance on specific features. The final Dense layer with 9 neurons is the applied, corresponding to the number of output classes.

Both dropout and L2 regularization are crucial in combating overfitting in neural networks. Dropout randomly drops units from the network during training, preventing co-adaptation of feature detectors and forcing the network to learn more robust features. L2 regularization, also known as weight decay, adds a penalty term to the loss function, discouraging large weights and encouraging a simpler model that generalizes better to unseen data. By incorporating dropout and L2 regularization into the CNN model, the study aims to improve its generalization performance and mitigate overfitting, ultimately enhancing its ability to accurately classify audio signals.

The most relevant building blocks of this CNN with dropout and regularization are described in the following equations below.

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij}) \tag{1}$$

Loss: Cross-entropy loss function, where N is the number of samples, C is the number of classes, y_{ij} is the true label, and p_{ij} is the predicted probability.

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla L(\theta_t) \tag{2}$$

Gradient Descent: Update rule for the model parameters θ, where η is the learning rate and $\nabla L(\theta_t)$ is the gradient of the loss function with respect to the parameters.

$$\text{Dropout: Output} = \frac{\text{Input}}{1 - \text{Dropout Rate}} \tag{3}$$

Dropout: Equation for dropout regularization, where the output is scaled by the inverse of the dropout rate to maintain the expected value of the activations.

$$\text{L2 Regularization: Loss} = \text{Loss} + \frac{\lambda}{2}\sum_{i}\sum_{j} W_{ij}^2 \tag{4}$$

L2 Regularization: Additional term added to the loss function to penalize large weights, where λ is the regularization parameter and W_{ij} are the weights of the model.

C. Autoencoder followed by SVM Architecture

The autoencoder model researched in this letter is a vanilla autoencoder that comprises of an encoder and a decoder, aimed at learning efficient representations of input data and subsequently reconstructing the original input.

The encoder, begins with a Conv2D layer with 32 filters of size (3, 3), resulting in an output shape of (None, 100, 20, 32). This is followed by max-pooling, reducing the spatial dimensions by half. Subsequent Conv2D layers with increasing filter sizes (64, 128, 256) are applied, each followed by max-pooling layers, resulting in a progressively reduced spatial dimensionality while increasing the number of channels. The output is then flattened into a one-dimensional vector, which serves as the encoded representation of the input, known as the code.

$$\text{Encoder: } c_k = f(W_1 x_k + b_1) \tag{5}$$

Where c_k is the output of the encoder (latent representation), f is the activation function, W_1 is the weight matrix of the encoder, x_k is the input data vector, and b_1 is the bias vector of the encoder.

The decoder, takes the encoded representation and aims to reconstruct the original input shape. It begins with a dense layer that reshapes the encoded representation into a suitable input shape for the convolutional transpose layers. Conv2D transpose layers with decreasing filter sizes (128, 64, 32, 1) are applied successively, each aiming to upsample the encoded features back to the original input shape. The final Conv2D transpose layer outputs the reconstructed image with an output shape of (None, 100, 20, 1), hence reconstructing a denoised, smoothed version of the original image.

$$\text{Decoder: } \hat{x}_k = g(W_2 c_k + b_2) \tag{6}$$

where $\hat{x}_k$ is the output of the decoder, g is the activation function, W_2 is the weight matrix of the decoder, c_k is the output of the encoder (latent representation), and b_2 is the bias vector of the decoder.

The autoencoder is trained using Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

Where n is the number of samples, y_i is the true value, and $\hat{y}_i$ is the predicted value.

The subsequent application of an SVM classifier to the decoded images further enhances the model's capabilities for classification tasks, showcasing the versatility and effectiveness of autoencoder-based approaches in image analysis and classification.

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1,\ldots,n \\ & \xi_i \geq 0, \quad i = 1,\ldots,n \end{aligned} \tag{8}$$

where $\mathbf{w}$ is the weight vector, b is the bias term, ξ_i are the slack variables, C is the regularization parameter, y_i is the label of the i-th training sample, and $\mathbf{x}_i$ is the feature vector of the i-th training sample.

D. Simplified Inception Architecture

The simplified shallow Inception model is a variant of the Inception v4 architecture designed for multiclass classification tasks. It comprises of a series of convolutional layers with varying filter sizes and depths, interspersed with batch normalization and activation layers, aimed at efficiently capturing complex patterns in the input data.

The stem block, composed of eight layers, is integral for extracting features from the input data. Beginning with an InputLayer defining the input shape of (None, 100, 20, 1), it progresses through Conv2D layers, generating feature maps with respective output shapes of (None, 49, 9, 32) and (None, 47, 7, 64). BatchNormalization and Activation layers stabilize learning and introduce non-linearity, preserving the output shapes. The stem block culminates with a MaxPooling2D layer, downsampling feature maps to (None, 23, 3, 64).

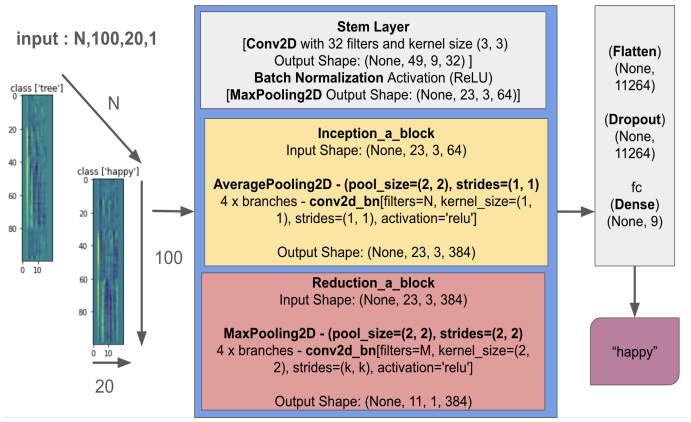

Fig. 5: The figure displays the simplified Inception architecture, tailored for audio signal classification which focuses on computational efficiency and streamlined design, featuring fewer branches, smaller kernels, and simpler convolutional structures than the v4. Despite the trade-off in hierarchical feature learning, this model offers a robust framework for capturing complex features and patterns in spectrogram representations, with significantly reduced parameter count for improved computational efficiency.

Next, the inception A block, comprises of sixteen layers, plays a pivotal role in feature extraction within the neural network. Initiated by Conv2D layers, this block extracts features from the input data, producing feature maps with output shapes of (None, 23, 3, 64) and (None, 23, 3, 96). Subsequently, BatchNormalization layers stabilize the learning process, followed by Activation layers introducing non-linearity. The block concludes with a concatenation operation, merging feature maps from different branches into a unified representation with an output shape of (None, 23, 3, 384). This sequence of operations within the Inception A block facilitates comprehensive feature extraction, vital for subsequent layers to build upon.

The reduction A block, consisting of sixteen layers, serves as a critical component for dimensionality reduction and feature aggregation within the neural network architecture. It begins with Conv2D layers that extract features from the input data, yielding feature maps with output shapes of (None, 23, 3, 192) and (None, 23, 3, 224). BatchNormalization layers are then applied to stabilize the learning process, followed by Activation. Subsequently, MaxPooling2D layers further condense the feature representation by down-sampling, resulting in an output shape of (None, 11, 1, 384). The reduction A block concludes with concatenation, merging feature maps from different branches into a unified representation with an output shape of (None, 11, 1, 1024). This amalgamation of operations

	CNN on spectro	CNN on MFCC	CNN on Full Feat Vec
Epochs	30	30	30
Params	1.1e5	1.1e5	8.6 e5
Train_time	**176 s**	187 s	754 s
Memory Allocation	**1437 MB**	1877 MB	3272 MB
Accuracy	0.74	**0.80**	0.78
Recall	0.74	**0.79**	0.77
F1-Score	0.74	**0.79**	0.77

TABLE 1: Comparison of performance metrics including number of epochs, parameters, training time, memory allocation, and accuracy for the CNN applied on a regular spectrogram, MFCC, and on a full feature vector.

within the reduction A block enables effective dimensionality reduction and feature aggregation, crucial for enhancing the network's representational capacity and overall performance.

The last three layers comprise of the Flatten layer that reshapes the input into a one-dimensional array, resulting in an output shape of (None, 11264). The Dropout layer randomly sets a fraction of input units to zero during training to prevent overfitting. And finally, the Dense layer produces the final output predictions, with a shape of (None, 9), representing the number of output classes. These layers collectively handle data transformation and address overfitting in the model.

The absence of additional inception blocks reduces the model's capacity to capture features at multiple scales, as seen in traditional Inception architectures where parallel branches with different filter sizes are utilized within each inception block. Additionally, the omission of reduction blocks may lead to feature maps with larger spatial dimensions, potentially affecting the efficiency of feature extraction and computational overhead.

VI. Results

A. CNN on different feature vectors

The results presented in this table showcase the performances of the simple vanilla CNN without dropout and regularization on three different training sets: regular spectrogram, MFCC and full feature vector. A spectrogram represents the time-frequency content of an audio signal, the MFCC (Mel-frequency cepstral coefficients) captures spectral features of the signal, and a full feature vector encompasses a comprehensive set of features derived from the audio data. The superior training performance of a vanilla CNN on MFCC compared to a regular spectrogram may originate from the MFCC's ability to compactly represent relevant spectral features, enhancing the network's capacity to learn discriminative patterns in audio data, and its inherent characteristics emphasizing perceptually relevant frequency bands and reducing dimensionality. The full feature vector contains concatenated information, including MFCC coefficients, Delta coefficients, Delta-Delta coefficients, signal energy coefficients, Delta energy coefficients, and Delta-Delta energy coefficients, extracted from audio frames and organized into a 43-dimensional array, a much

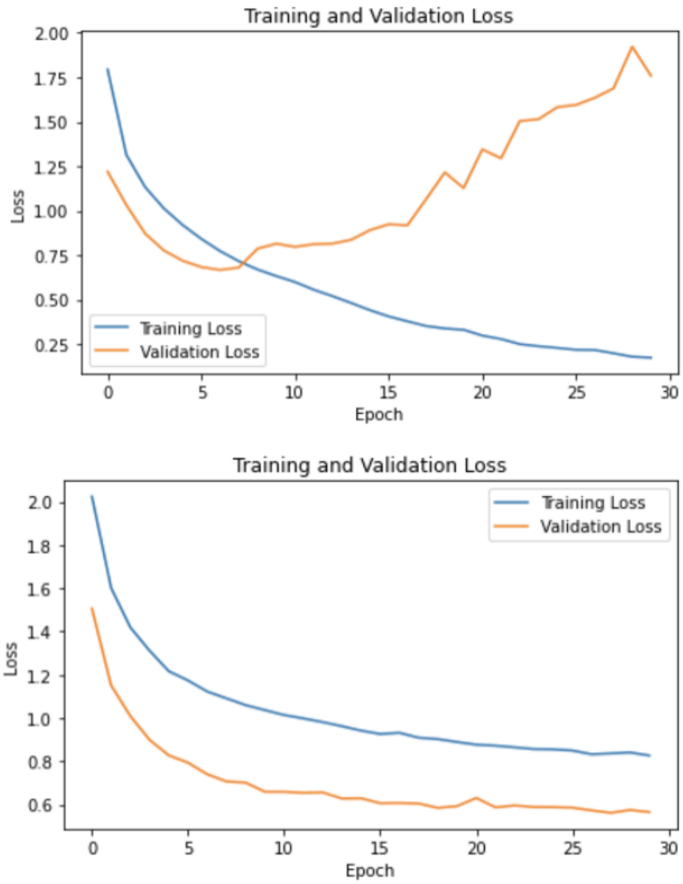

Fig. 6: Loss curves for vanilla CNN and CNN with dropout and L2 regularization on the MFCC dataset

computationally expansive feature vector than the MFCC. The slightly worse performances from the CNN trained on the full feature vector from the CNN trained on the MFCC and the 5x increase in training times justify the use of the MFCC vector as as the main training feature for all of the subsequent models. Looking at table 1, all models were trained for 30 epochs, with the number of parameters being consistent at 1.1e5 for the first two and 8.6e5 for the last. The training times vary, with spectrograms taking 176 seconds, MFCC 187 seconds, and full feature vectors 754 seconds. Memory allocation follows a similar trend, with spectrograms requiring 1437 MB, MFCC 1877 MB, and full feature vectors 3272 MB. In terms of performance metrics, CNN on MFCC yields the highest accuracy at 0.80, followed closely by CNN on full feature vectors with 0.78, and CNN on spectrograms with 0.74. The recall, F1-score, and accuracy metrics exhibit similar patterns across the three feature representations.

B. CNN with dropout and L2 regularization

Given the high degree of overfitting from the vanilla CNN model on all of the feature vectors, dropout and L2 regularization was applied. This allowed the reduce of overfitting and increase in accuracy from 0.8 to 0.84. Therefore this was the best vanilla CNN model used for benchmark among

	CNN + drop & L2	SVM with encoded features	Shallow Inception
Epochs	30	N (SVM)	30
Params	**1.1e5**	5e5 (enc) 4e5 (dec)	1.5e6
Train_time	**178 s**	304 s (SVM)	852 s
Memory Allocation	**412.03 MB**	5716.99 MB	1052.67 MB
Accuracy	0.85	0.71	**0.89**
Recall	0.85	0.68	0.88
F1-Score	0.85	0.69	0.88

TABLE 2: Comparison of performance metrics including the number of epochs, parameters, training time, memory allocation, and accuracy for the CNN with Dropout and L2 regularization, Autoencoder + SVM, and Simplified Inception models.

other more advanced models. A simple grid search over the following hyperparameter space was conducted:'dropout˙rate': [0.2, 0.3, 0.4], 'l2˙regularization': [0.0001, 0.001]. A deeper grid search in terms of more hyperparameters such as the number of convolutional layers, the number and size of filters/kernels, strides, pooling operations, activation functions, learning rate and more could be conducted in the future for a more optimized CNN architecture. The grid search selected a dropout rate of 0.3 and an L2 regularization of 0.0001 achieving an accuracy of 0.85, a slight improvement in all metrics from the previous CNN with dropout and regularization.

C. Model Comparison

After the performances of a simple CNN with dropout and L2 regularization where analyzed, more complex models where also researched. The results presented in the table offer valuable insights into the performance and characteristics of three distinct model architectures. Notably, the Inception model stands out with the highest accuracy of 0.89, indicating its effectiveness in capturing complex features and patterns within the dataset. This superior performance aligns with the model's significantly larger parameter count. Conversely, the Autoencoder + SVM model exhibits the lowest accuracy of 0.71, despite its substantial memory allocation and longer training time. The CNN model, while achieving a respectable accuracy of 0.85, strikes a balance between model complexity, training efficiency (as evidenced by its relatively short training time), and memory allocation. Overall, these results highlight the trade-offs between model complexity, computational resources, and performance, underscoring the importance of carefully selecting and optimizing model architectures to suit the specific requirements and constraints of the task at hand.

VII. Concluding Remarks

In this letter different feature vectors, from regular spectrograms, MFCC and full feature vectors where used as the training sets of a vanilla CNN model, and the performances of the model were analyzed on the different sets of feature vectors, the MFCC resulted as the best feature vector to use both in a

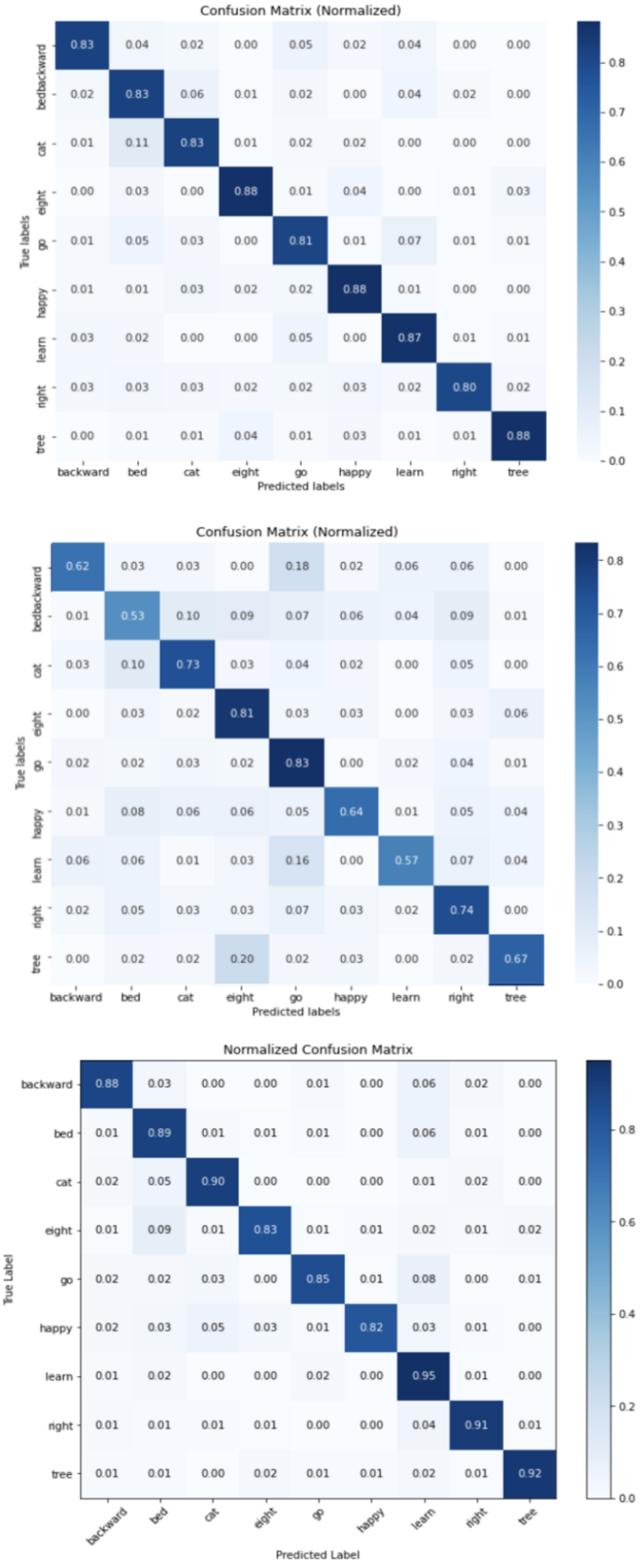

Fig. 7: Confusion Metrices for CNN, Autoencoder + SVM, Shallow Inception respectively.

accuracy and computational cost point of view. Overfitting was a main issue regarding the training of the simple vanilla CNN, therefor dropout and L2 regularizations where applied to increase the generalization of the model. A grid search over a small space of dropout and L2 regularization hyperparameters was conducted. After the optimization of the CNN model with dropout and L2 regularization was conducted, more complex architectures where researched such as autoencoders followed with a simple SVM and a shallow Inception model, with the latter achieving the best performances in terms of classification performance and the vanilla CNN achieving the best computational efficiency and memory allocation performances during training. I found that given the dimensionality of the MFCC feature vectors, the standard Inception v4 architecture fails to perform both in accuracy and computational cost due to large amounts of padding in each image hence diminishing the effective receptive field, diluting relevant information, increasing computational costs, and leading to overemphasis on background features, adversely affecting the model training and generalization performance. A simpler Inception v4 architecture was found to be the best solution, lowering the number of inception and reduction blocks as well as the depth of the stem cell block, while preserving the integrity of the MFCC feature dimensionality with a smaller amount of padding in each image. A more detailed grid search over the hyperparameter space for both the shallow Inception and CNN models is something that could improve performances even further. A combination of techniques such as autoencoder followed by the shallow Inception model is also something that can be explored as it would allow to firstly extract robust and compact representations from raw data with an autoencoder and then leveraging the diverse receptive fields of the Inception model to capture hierarchical features, enhancing the overall model's performance with the shallow Inception model. During this research I have learned the importance of data pre processing, normalization, augmentation, and the effects that a poorly managed dataset can have on the training of a model. I have also learnt that the dataset dimensionality is an extremely important factor when choosing a model, and it's something that needs to be taken into consideration from the get go, while learning to combine different techniques and models to optimize different process and improving performances.

REFERENCES

[1] D. B. J. Schalkwyk, "Your word is my command," *Advances in Speech Recognition*, pp. 61–90, 2010.

[2] C. P. G. Chen and G. Heigold, "Small-footprintKeyword Spotting using Deep Neural Networks," *Proc.ICASSP*, 2014.

[3] Y. LeCun and Y. Bengio, "Convolutional Networks for Images, Speech, and Time-series," *MIT Press*, 1995.

[4] H. J. O. Abdel-Hamid, A. Mohamed and G. Penn, "Applying Convolutional Neural Network Concepts to Hybrid NN-HMM Model for Speech Recognition," *Proc. ICASSP*, 2012.

[5] C. P. Tara N. Sainath, "Convolutional Neural Networks for Small-footprint Keyword Spotting," Sept. 2015.

[6] J. Z. Zhiming Wang, Xiaolong Li, "SMALL-FOOTPRINT KEYWORD SPOTTING USING DEEP NEURAL NETWORK AND CONNECTIONIST TEMPORAL CLASSIFIER," Sept. 2017.

[7] C.-L. L. H.-y. L. Chia-Hsuan Li, Szu-Lin Wu, "Spoken SQuAD: A Study of Mitigating the Impact of Speech Recognition Errors on Listening Comprehension," 2018.