Demonstrating Training of an Interpretable Speech Recognition Network using Human-Guided AI

Mansi Goyal
Language Technology Institute
Carnegie Mellon University
Pittsburgh, PA 15213
mansigoy@cs.cmu.edu

Marta Mendez
Language Technology Institute
Carnegie Mellon University
Pittsburgh, PA 15213
martamendez@cmu.edu

Sreenidhi Sundaram
Language Technology Institute
Carnegie Mellon University
Pittsburgh, PA 15213
sreenids@cs.cmu.edu

Sreedhar Radhakrishnan
Information Networking Institute
Carnegie Mellon University
Pittsburgh, PA 15213
sreedhar@andrew.cmu.edu

1 Abstract

State of the art neural based speech recognition models are "black-box" in nature. Through this experimental research we present an interpretable speech recognition framework that can discriminate between small vocabularies of confusable phonemes. We also contribute a speaker dependent pre-processed dataset of isolated phoneme utterances which was used to train the aforementioned framework. The framework is an amalgamation of 49 shallow detectors consisting of 39 phoneme detectors and 10 specialized classifiers that were designed, trained and tested to perform specific human guided tasks. The complete network obtained an accuracy of 83.9 percent on the validation dataset which was of comparable performance to that of our baseline model - a deeper uninterpretable neural network with regularization. In addition, using the framework, the interpretation of the model's result can be deduced at any time by analyzing the contribution of the individual shallow detectors on the output classification using four intepretability graphs. The constraint of interpretability meant that the system had to be simple yet intuitive as opposed to deeper and more complex. The results of this research is a small step towards building interpretable and human guided deep learning models that have comparable performance to state of the art "black-box" neural models.

2 Introduction

Humans are driven by curiosity. This inherent tendency of "exploring" has led us through over 6 million years of evolution to today. Ironically, we also encourage our algorithms to do the same: explore, make mistakes and learn from them! Despite the excellent performance achieved by Deep Learning approaches across a variety of tasks, many humans still distrust these systems and Artificial Intelligence in general. The problem with many state-of-the-art DL networks is that they are essentially "black box" in nature. Even the DL practitioners cannot explain with 100% confidence the predictions made by the networks they built themselves.

To overcome this problem, specifically in the domain of speech recognition, our team decided to build interpretable neural networks to discriminate between small vocabularies of confusable phonemes under the guidance of Professor James Baker. To circumvent the lack of directly comparable results and relevant data in this specific domain, we built an end-to-end deep learning pipeline right from data creation, preprocessing, feature engineering, to model design, development and validation.

First, a dataset for the experiments was generated manually by the team, wherein recordings of over 38 isolated phonemes were made in male and female voices. The raw speech data was further processed using a window length of 10 milliseconds and 40 log amplitude features sampled on a Mel scale. The ground truth labels for each frame were bootstrapped using the open-source Google Voice Activity detector. The core of the pipeline - the Deep Learning model - is an ensemble of 49 shallow neural networks (sub-networks), each specifically designed to perform human guided tasks, with architectures designed to be interpretable. Out of the 49 specialized networks, 39 were individual phoneme discriminators and 10 were specialized discriminators. More details about these networks is given in the contribution section.

Each of the sub-networks was trained to be a high-performance discriminator on specific tasks, such as differentiating between vowels and consonants, high and low vowels, voiced and unvoiced fricatives, and confusable phoneme pairs. The system takes as input a sequence of frames and returns the corresponding frame-level phoneme label - a probability score. This system can be used in isolation as a frame-level phoneme recognizer or can be plugged in as the frontend to a deeper backend system capable of performing specialized tasks like word detection by back-propagating the error all the way to the frontend. The constraint of interpretability meant that the system had to be simple yet intuitive as opposed to deeper and more complex neural networks that are traditionally “black box” in nature. The interpretability of this system is a small step in the direction of making Deep Learning networks more transparent! Debugging of input edge cases, removal of bias and feature engineering would become more efficient as we would be able to see how different features affect the output clearly. This would lead to easier adaptability by the general public.

The proposed framework is one piece of a much larger long-term project, and our system is aimed at being one component of a much larger system. The long-term project is to show that it is possible to grow an interpretable network with a special architecture to a large size and challenging task while maintaining interpretability, with speech recognition as a demonstration task. The long-term task will be large vocabulary, continuous speech recognition. A tower network with interpretable strata is an example of such a "special architecture." There are other examples. For example, a collection or ensemble of interpretable towers is also interpretable. The role of our system is to be one stratum of such a tower. In some designs, variants of our system may be used as strata in multiple levels of the tower. However, the other components of such a system do not yet exist. Therefore, our chosen dataset was only intended to show that our system can produce interpretable output that could be used by one or more higher strata in such a tower. For this purpose, the dataset is adequate. As additional strata are added, increasingly complex tasks, with increasingly complex datasets will be needed.

3 Literature Review

3.1 Related Work

There has been related work in the field of Interpretable Deep Learning and Explainable AI. However, the proposed framework, as per our knowledge, is a first of its kind in terms of its ability to generalize as well as model higher order feature interactions. For example, other approaches like Neural Additive Models (NAM) [1] apply a linear combination of neural networks, wherein each network corresponds to a single input feature. This makes it easy to interpret the impact of each feature on the final output. However, this network would fail to model high-order feature interactions. We believe our system could perform better in this aspect, as we leverage all the features provided together. We do so by nudging our networks to learn specific tasks (human-guided AI). Moreover, our framework is generalizable in nature as new specialized tasks can easily be added. In addition, while NAM focuses on being interpretable with respect to the features, our focus is on making the network interpretable with respect to the output. Therefore, for a given output we can explain what contributed to that output being fired, and also what caused the model to correctly or incorrectly classify an input.

We also compared our framework to local surrogate models such as LIME [2] and SHAP [3] and from our understanding, our proposed framework would be able to give more concrete explanations of the model results. This is because LIME and SHAP use Machine Learning techniques to interpret

Machine Learning results making it similar to a shot in the dark like approach.

We have provided our literature survey in the areas of Explainable AI, Confusable Words, Tower Networks, AI System Deployment and Speech Recognition in the sub-sections below.

3.2 Explainable AI

A Survey on Explainable Artificial Intelligence (XAI): towards Medical XAI [4] - The paper focuses only on Explainable AI in the medical field. However, it gives a good idea about the advances in this field. We were particularly interested in the signal method of interpretability where the activation values of neurons in a layer are used to interpret the results. For example, these activations can be used to generate heat maps. "Explainable AI," The Royal Society, 28 November 2019 [5] - Advantages on Explainable AI: ensure the system is working as expected, meet regulatory standards, allow those affected by a decision to challenge or change that outcome.

"Explainable Artificial Intelligence (XAI)," Dr. Matt Turek, The U.S. Defense Advanced Research Projects Agency (DARPA) [6] - The Explainable AI (XAI) program with aims to develop a toolkit library consisting of machine learning and human-computer interface software modules that could be used to develop future explainable AI systems.

Learning Interpretable Relationships between Entities, Relations, and Concepts via Bayesian Structure Learning on Open Domain Facts [7] - Although this paper does not directly apply to the field of speech recognition it explores an interesting application in the field of interpretable AI. They used a Bayesian Network Structure Learning (BNSL) network to identify relations between entities.

3.3 Confusable Words

Feature Extraction Techniques with Analysis of Confusing Words for Speech Recognition in the Hindi Language [8] - The paper focuses on confusing words for speech recognition in the Hindi Language. Due to the lack of publicly available data, they had to build their own dataset. Something that we will be also required to do for our project. Another interesting learning from this paper was the usage of energy parameters to improve recognition scores. They used features such as power normalized cepstral coefficients (PNCCs).

Detection of confusable words in automatic speech recognition [9] - The paper focuses on the detection of confusable words using a dissimilarity measure based on phonetic and acoustic information. This paper was published in 2005 and there seems to be minimal work in this area since then. In addition, we did not find extensive literature on the recognition of confusable words and that is one reason we believe our work has the potential to be published.

3.4 Tower Networks

Based on our discussions with Professor Baker, we will address the interpretability issues of deep neural networks by working with a specific deep neural network architecture called "tower networks". We were not able to find much literature online, so this summary is solely based on the information provided by him. These networks are composed of subnetworks called "strata" or "stratum", which are basically sets of one or more neural networks with specific objectives, which may even have their own objective and error cost functions. In training, stratums will receive backpropagation both from their own inner objective, as well as from the final objective of the complete tower. These stratum networks can be designed to explicitly represent specialized knowledge about a particular task: for image recognition, this could be "is-a-part-of" relationship (known as mereologies); for speech recognition, it could be representations of successively longer linguistic units, starting from 10ms spectral frames, to allophones, phonemes, syllables and then words (known as "state space network embedding", that explicitly represents human interpretable knowledge about speech, language, and vocabulary).

3.5 AI System Deployment

Accurate and Compact Large Vocabulary Speech Recognition on Mobile Devices [10] - The paper describes the development of an accurate, small footprint, large vocabulary speech recognizer for mobile devices. Google Inc. describes how they have designed a speech recognition system that is directly deployed on mobile devices. On-device model deployment is becoming increasingly common and if we succeed in our work then it can have an engineering impact in the real world, given the ease of model training and data storage on the user's device due to the ability of the model to learn from a small vocabulary of confusable words.

3.6 Autoencoders for Speech Data

Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent, and variational models[11] helped us get an idea about how autoencoders can be used to generate latent representations of speech. Based on this paper, we think Variational autoencoders (VAEs) would be a great candidate for our problem.

Unsupervised speech representation learning using WaveNet autoencoders [12]: Mainly, three variants were covered: three variants: Gaussian Variational Autoencoder (VAE), a simple dimensionality reduction Bottleneck, and a discrete Vector Quantized VAE (VQ-VAE). This paper also used speaker-independence as a factor to judge the quality of the representations. An important input for us since we wish to investigate that as part of our proposed extensions.

3.7 Speech Generation

We are interested in this domain for automated synthesis of speech data for training our front-line model.

WaveNet: A Generative Model for Raw Audio [13]: A popular approach introduced by Google. This may prove to be useful for us in the future as it can be used to generate raw audio waveforms.

HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis[14]: A great option for us since it is very memory efficient. This paper employs a modified Generative Adversarial Network to produce raw waveforms.

4 Contribution

In this subsection we will be stating our contributions, which include our processed dataset, trained phoneme detector models, trained specialized task classifiers and a complete frame-level phoneme classifier. We also provide the relevant implementation details wherever applicable.

4.1 Dataset

Our first contribution is a pre-processed labelled dataset of phoneme utterances. We have manually generated this dataset by recording 38 isolated phonemes. We then processed this raw data to make it suitable to feed into our model using signal processing concepts. The dataset for each of the 49 shallow detectors was generated, prepared, processed and labelled separately. The first step was to record the isolated phoneme utterances. This was recorded using the Audacity software at a sampling rate of 16000 Hz. The utterances were stored in the .wav file format.

The next step of the data processing pipeline was to generate log amplitude features sampled on a Mel scale. This was performed using the Librosa library in Python wherein a window length of 10ms was applied and 40 filters were used to generate 40 log amplitude features.

Finally, in order to generate frame level labels of our phoneme data, we used Google Voice Activity Detection API on Python with an aggression mode set to 3. Graphs for the frame level voice activity detection as well as the energy levels of the melspectograms were plotted to validate the consistency of the voice activity detector with respect to the energy levels shown on the melspectograms. Finally,

the melspectogram features as well as the frame level labels were stored in the .npy format for each phoneme which was the input data to our detection and classification models.

4.2 Phoneme Detectors

Our second contribution is the training of 38 phoneme detector models that are capable of discriminating each of 38 phonemes as well as silence. Below is the list of the 39 (38+SIL) phonemes we have considered:

AE AH AW AY B CH D DH AE AH AW AY B CH D DH J K LL MM NG NN OH OO OW OY P RR SH SS T TH UE UH VV WW YY ZZ SIL

We wanted each of these detectors to be a high performance discriminator for its particular target phoneme, capable of differentiating that phoneme not only from silence, but also from the other phonemes, thereby reducing the number of false positives.

Our main requirement was to implement an interpretable network, thus we needed to keep these models shallow. The objective of each phoneme detector is to discriminate a particular phoneme from any other sound, so we implemented them as a binary classification problem.

We generated custom Dataset classes to manipulate the frames accordingly, differentiating between two labels, 0 and 1:

- **PhonemeDataset**:
  - for a particular target phoneme - e.g "AA", we fetch all the frames of that phoneme's recording and bootstrap the labels of those frames that contained the target utterance to the class "1", and the remaining (which correspond to SIL because of the nature of the recordings), to the class "0".
  - to make the phoneme detector more robust, we also loaded frames from the recordings of other phonemes and bootstrapped those frames to the class "0". However, this time in order to maintain balance in our dataset balance we discarded all the silence frames and took only 10% of the other phoneme frames.
- **SilenceDataset**:
  - we have a separate dataset class for silence wherein we load all the frames of all the phoneme recordings and bootstrap the label for frames that are silent as "1", and the remaining voiced frames as "0".

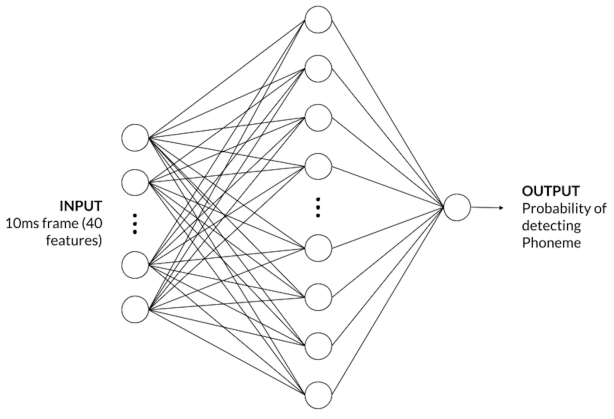

Figure 1: Phoneme Detector Architecture

The final architecture details of the phoneme detectors is as follows:

- PhonemeDetectorModel:
  - Linear Layer (input features:40, output features:128)
  - Batchnorm Layer
  - ReLU activation
  - Output Layer (input features:128, output features:1)
  - Sigmoid activation
- Criterion: Binary Cross Entropy Loss
- Optimizer: SGD with learning rate 0.01 and momentum 0.9
- Scheduler: Reduce LR On Plateau

We trained each of the networks for 100 epochs and saved the models thereby generating 39 extremely powerful high performance phoneme detectors.

4.3 Specialized Task Classifiers

To add more domain knowledge into the network, in addition to the 39 phoneme detectors, we decided to train 10 specialized classifiers wherein each classifier was designed to perform a particular human-guided task.

The goal of each specialized task classifier is to discriminate between two target classes (each of which contains a list of phonemes as described in the table). The specialized classifiers should be able to handle all types of frames passed to it during inference time. Thus, we designed the specialized task classifiers as multiclass classifiers. More specifically, each specialized task classifier differentiates between three classes: by default class 0 will be any other phoneme (not belonging to any of the target categories), class 1 will be one of the target phoneme lists, and class 2 will be the other one as shown in the table.

Below is a table containing the 10 specialized tasks and the phonemes contained in the corresponding categories. We have assigned Class "0" to phonemes that fall under neither category.

Specialized Task Classifiers		
Task	**Class 1**	**Class 2**
Vowels (Class 1) vs Consonants (Class 2)	'EE', 'IH', 'EH', 'AE', 'UH', 'ER', 'AH', 'AW', 'OO', 'UE'	'FF', 'HH', 'MM', 'NN', 'NG', 'RR', 'SS', 'SH', 'VV', 'WW', 'YY', 'ZZ'
High Vowels (Class 1) vs Low Vowels (Class 2)	'EE', 'IH', 'UE', 'OO'	'AE', 'AH', 'AW'
Voiced (Class 1) vs Unvoiced (Class 2) Fricatives	'DH', 'VV', 'ZZ'	'FF', 'SS', 'SH', 'TH'
'SS' (Class 1) vs 'ZZ' (Class 2)	'SS'	'ZZ'
'B' (Class 1) vs 'P' (Class 2)	'B'	'P'
'DH' (Class 1) vs 'TH' (Class 2)	'DH'	'TH'
'WW' (Class 1) vs 'YY' (Class 2)	'WW'	'YY'
'EE' (Class 1) vs 'AW' (Class 2)	'EE'	'AW'
'AH' (Class 1) vs 'AW' (Class 2)	'AH'	'AW'
'MM' (Class 1) vs 'NN' (Class 2)	'MM'	'NN'

The system design of the framework is highly modular and generalizable. Training of the specialized classifiers are automated and addition of new subtasks/classifiers are seamless. The user has to define their task in the utilities file and the respective task is automatically amalgamated into the model training pipeline.

We generated a custom Dataset class to manipulate our data accordingly, in essence, going through all the phoneme recordings, loading the frames and labels and updating the values as per the class 0, class 1 or class 2 definition of the specialized task classifier.

The final architecture of the SpecializedTaskDetector is given below:

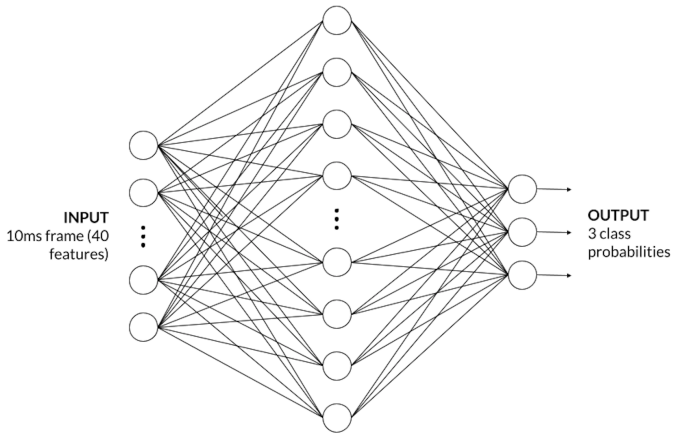

Figure 2: Specialized Classifier Architecture

- SpecializedTaskDetector:
  - Linear Layer (input features = 40, output features = 512)
  - Batchnorm Layer
  - ReLU activation
  - Output Layer (input features = 512, output features = 3) (not including the final softmax layer because it is already part of the Cross-Entropy criterion)
- Criterion: Cross Entropy Loss
- Optimizer: SGD with learning rate 0.01 and momentum 0.9
- Scheduler: Reduce On Plateau

We trained each of the 10 specialized classifiers for 50 epochs and saved the models. So along with the 39 phoneme detector models, we have a total of 49 high performing trained sub-networks.

4.4 Complete Network

The complete network is a linear combination of all the previously trained networks (39 phoneme detectors + 10 specialized task classifiers), and makes use of only one extra final linear layer to generate the output probability scores. In this manner, we maintain interpretability, and the network follows a Human-Guided design.

The input to the complete network is a 10ms frame (in its 40 features Mel spectrogram representation), and the output is a list of 39 probabilities (silence + 38 phonemes). The input is processed in parallel by each of the pre-trained subnetworks (49 in total), and their probability score outputs are concatenated together into a 69-dimensional vector (1 output per phoneme detector and 3 outputs per specialized task classifier). This vector is fed onto a final linear layer consisting of 39 neurons, whose output, after passing through a softmax layer, will provide us the final phoneme probabilities.

We are initializing the weights of the Phoneme Detectors and of the Specialized Task Classifiers with the weights of the pre-trained networks. Even though these subnetworks have already been trained, the weights that connect their outputs to the final linear layer of the complete network have still not been learned. Thus, we need to train the complete network using our dataset, to learn the weights of this final linear layer.

One important aspect we wish to maintain is the interpretability of the subnetworks; because of that, we are locking the weights of the subnetworks to effectively prevent them from being updated during the backward pass of the complete network.

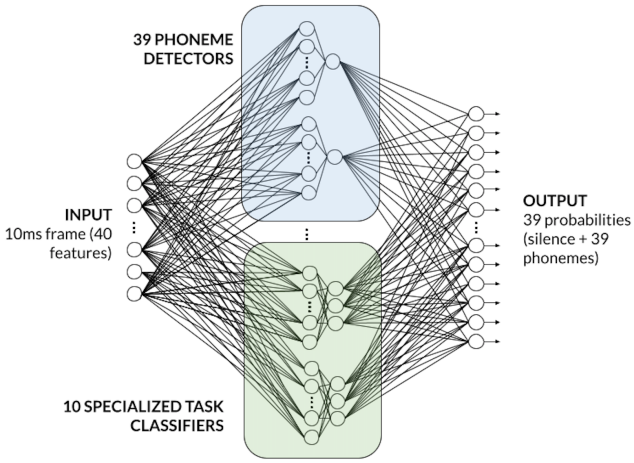

Figure 3: Complete Network Architecture

The final architecture details are as follows:

- FrameToPhonemeClassifier
  - Shallow subnetworks (pre-trained):
    - 39 Phoneme Detectors
    - 10 Specialized Task Classifiers
  - Linear Layer (input features = 69, output features = 39)
  - (not including the final softmax layer because it is already part of the Cross-Entropy criterion)
- Criterion: Cross-Entropy Loss
- Optimizer: SGD with learning rate 0.01 and momentum 0.9
- Scheduler: Reduce LR On Plateau

4.5 Interpretability Graphs

Because our main goal was interpretability, we wanted to have means to understand how our network was processing the different frames. Thus, in order to interpret the results, we decided it would be useful to generate the following four plots:

1. **Final Model Probabilities**: represents in the y axis, the output of the complete network (which corresponds to 39 phoneme probabilities), and in the x axis the phoneme tag.
2. **Shallow Networks Output Probabilities**: represents in the y axis, the output probabilities of each of the subnetworks (39 phoneme detectors + 10 specialized task classifiers). In the case of the specialized task classifiers, we have 3 outputs, which correspond to class 0, class 1 and class 2. The x axis represents the subnetwork name, for phoneme detectors, it is their target phoneme tag, and for the specialized task classifiers, it is the task ID followed by the class ID (0,1,2).
3. **Shallow Networks Contribution to fired neuron**: represents in the y axis, the contribution of each of the subnetworks (39 phoneme detectors + 10 specialized task classifiers) to the final neuron that was fired (argmax of the output probabilities). The contribution is defined as the product of the probability and the weight that connects that output to the fired neuron in the final layer.
4. **Shallow Networks Contribution to ground truth neuron**: this graph is only plotted in case of misclassification. It follows the same idea as graph 3, however it uses the weights of the ground truth neuron (the phoneme that should have been fired) instead.

In the Results section we will provide the sample plots for both a correct and incorrect classification, as well as their interpretation.

5 Experimental Evaluation

5.1 Experiments

The task as we mentioned earlier was to build an interpretable frame by frame phoneme discriminator. In the process of building our system, we had to iterate over multiple design choices and model architectures whilst keeping in mind that the end system had to remain interpretable.

One of the first roadblocks we faced was dealing with manually generated data. The datasets we have dealt with so far in all our homeworks and projects, were already cleaned, pre-processed, labelled and prepared in a way that they could be fed into deep neural networks automatically. Since we generated our own dataset, we had to do a significant amount of speech processing to convert the data we generated into a format that could be used by our model. For speech processing, we built our implementation on top of an existing speech processing code provided to us by TAs. We processed the speech data using a 10ms window length and 40 log amplitude features sampled on the mel scale.

The next roadblock was generating the ground truth labels for each frame in the recordings. Each sound file contained recordings of a particular phoneme. However, the file contained a combination of that phoneme and silence, since the recordings were made in isolation. Hence, we had to split each speech recording of a phoneme into silent and non-silent intervals and label the frames accordingly. To do so we tried Librosa and Google Voice Activity Detector which are both open-source libraries. We decided to go with Google Voice Activity Detector since it gave us more stable results, and had the capability to both preprocess and split a raw signal into intervals. We bootstrapped the labels and created the final dataset.

We started to experiment with building the shallow detectors. Each shallow detector, as mentioned above, is a particular phoneme recognizer. We initially decided to build the Shallow detector using autoencoders to generate optimized embeddings of each phoneme. These optimized embeddings would then be fed into the Tower Network for classification. Our model architecture for each shallow detector looked as shown in Figure 1.

Figure 4: Autoencoder architecture

We implemented this architecture in Pytorch and wrote our own code to generate these embeddings, however, we were advised by Dr. Baker that this design would not be as interpretable as was desired. We then decided to go with the approach as described in the above section.

Additionally, We wanted to explore the possibility of generating 2 different subsequent versions of the network: one using vanilla RNNs, and another using an Encoder-Decoder architecture with attention. We decided to start simple, by adding context to the MLP. However, the performance actually decreased. This has to do with the fact that there is no sequential dependency between our frames, a downside caused by how our dataset is currently structured. We only have recordings of isolated phonemes. Therefore, when we want to train a specific task classifier (distinguishing between vowels and consonants), we scan our different recordings and load the frames that are relevant to our task. In this case, we are stitching together frames that did not occur naturally in any of the recordings. Thus, the model was getting confused. Because of this reason, and the lack of a better dataset in that sense, we decided to not move further forward into the direction of RNNs and Encoder-Decoder architectures. This is however something we would like to explore, so we will look further into it.

5.2 Dataset

One of the most challenging parts of the project was the opportunity to create, preprocess and feature engineer a custom dataset from scratch. We decided to create our own dataset for a number of reasons:

- There was no open source isolated phoneme utterance dataset that was available online as per our knowledge.
- We wanted to build a data processing pipeline that is dialect and gender agnostic thereby enabling the research group to diversify the speech recognition system in the future.
- We wanted experience implementing an end-to-end deep learning pipeline which included data creation, data preprocessing and feature engineering in addition to model design, development and evaluation.

The dataset development pipeline overview is shown below. We have used the audacity platform to record the 38 phonemes at a sampling rate of 16000 Hz. The recorded raw speech data were further processed using a window length of 10 milliseconds and 40 log amplitude features sampled on a Mel scale. The ground truth labels for each frame were bootstrapped using the open-source Google Voice Activity detector. Each recording produced approximately 2000 frames. Since we had 38 isolated phoneme recordings and our models used frame-level data points, the total number of data points in our dataset was approximately 2000*38 = 76000 data points in total. We further split our data into 70 percent training, 15 percent validation, and 15 percent testing. Thus the total number of training data points in our dataset was 0.7*76000 = 52000 data points.

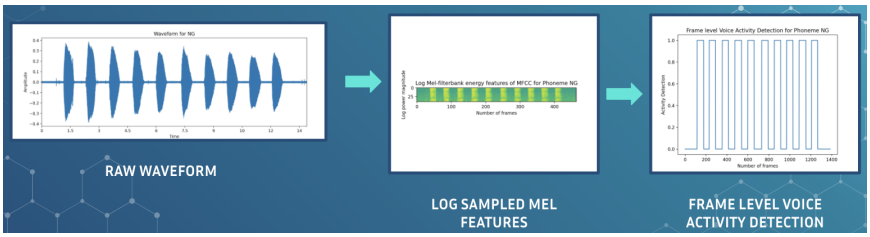

Figure 5: Voice Activity Detection using Google GMM

It must also be noted that there were a number of challenges that we faced that required us to engineer the aforementioned pipeline. For example, lack of ground truth labels was a challenge that we overcame using the voice activity detection module of our pipeline. We had to experiment with different aggressive modes (filtering out non-speech) of the Google Voice Activity Detector to ensure that the frames were correctly labeled. This was done by manually evaluating the voice activity detection graphs of each phoneme and ensuring that the frames were correctly classified by comparing them with their melspectogram energy graphs. In addition, we had to engineer 11 custom data loaders - 1 for the phoneme detectors and 10 for the specialized classifiers to ensure seamless training of our models.

5.3 Evaluation Metrics

The evaluation metrics and their description used have been provided below: Accuracy - A percentage value of the total number of correct classifications across the dataset.

- Precision - A decimal value between 0 and 1 representing the fraction of correctly predicted labels of a particular class with respect to all classifications output as that respective label by the model.
  That is, Precision = True Positive/(True Positive + False Positive)
- Recall - A decimal value between 0 and 1 representing the fraction of correctly predicted labels of a particular class with respect to the total number of instances of the respective class present in the dataset.
  That is, Recall = True Positive/(True Positive + False Negative)

- F1 Score - A metric that identifies a combination of Precision and Recall to account for the Precision-Recall Tradeoff.
  That is, F1 Score = 2*Precision*Recall/(Precision + Recall)

The metrics used for each detector or classifier as well as the reasoning for the same is present in the table below.

Model Metric Use and Reasoning		
Model Type	Metrics Used	Reasoning
Phoneme Detector	Precision, Recall and F1 Score	We chose the metrics because of dataset imbalance due to the presence of a large number of silence frames. We modeled the result of each phoneme in a confusion matrix and evaluated its precision, recall and f1 score.
Specialized Task Classifiers	Precision, Recall and F1 Score	Similar to phoneme detectors, we chose the metrics because of dataset imbalance due to the presence of a large number of silence frames. We modeled the result of each phoneme in a confusion matrix and evaluated its precision, recall and f1 score.
Complete Network	Accuracy	We wanted a consistent metric that we could use for both the baseline model as well as the complete network. In addition, to evaluate the complete network as well as the baseline, percentage of correct classifications was the most appropriate metric.

5.4 Description of Baseline

The baseline architecture chosen for our problem statement was a Multi-Layer Perceptron (MLP) from Homework 1 Part 2 in the Carnegie Mellon University 11785 Deep Learning course that achieved an A+ score on Kaggle with a score of 102 out of 100. The architecture was implemented using PyTorch and the only assumption was that the model would perform the same way for our custom dataset as it did for the dataset provided in Homework 1 Part 2. The architecture was reused, however, the data loader was re-implemented to ensure the program works with our dataset. Our dataset included 76 .npy files (38 mel spectrogram features for each phoneme and 38 frame level labels of the phonemes). Thus, we had to modify the data loader to concatenate the dataset accordingly to generate our features and labels data for the 38 phonemes.

The only caveat was that our proposed architecture did not use regularization techniques and the baseline architecture included dropouts and Xavier Weight Initialization. However, our goal was to check if our interpretable architecture provided similar performance to that of a more complex non-interpretable model and thus the caveat did not end up becoming a major concern but rather strengthened our baseline choice. The architecture of our baseline model is given in Figure 6.

6 Results

6.1 Comparison with Baseline

Here we present a comparison of the framework's performance as compared to the baseline architecture described in 5.4. The architecture of the baseline is described in Figure 6. Below is a snapshot of the results:

Baseline Comparison Results		
	Baseline	Our Framework
Training Accuracy	98.6907%	90.1817%
Validation Accuracy	83.75%	83.9965%

```
Cuda = Truewith num_workers = 8
MLP(
  (mlp_network): Sequential(
    (0): Linear(in_features=40, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
    (3): Dropout(p=0.25, inplace=False)
    (4): Linear(in_features=1024, out_features=1024, bias=True)
    (5): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): LeakyReLU(negative_slope=0.01)
    (7): Dropout(p=0.25, inplace=False)
    (8): Linear(in_features=1024, out_features=2048, bias=True)
    (9): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): LeakyReLU(negative_slope=0.01)
    (11): Dropout(p=0.25, inplace=False)
    (12): Linear(in_features=2048, out_features=1024, bias=True)
    (13): LeakyReLU(negative_slope=0.01)
    (14): Linear(in_features=1024, out_features=1024, bias=True)
    (15): LeakyReLU(negative_slope=0.01)
    (16): Linear(in_features=1024, out_features=512, bias=True)
    (17): LeakyReLU(negative_slope=0.01)
    (18): Linear(in_features=512, out_features=256, bias=True)
    (19): LeakyReLU(negative_slope=0.01)
    (20): Linear(in_features=256, out_features=39, bias=True)
  )
)
```

Figure 6: Baseline Architecture

As seen by these results, the proposed framework has comparable performance to that of the baseline, with the additional benefit of being interpretable. These positive results indicate that such a framework could be used to replace traditional "black-box" MLPs with no compromise in terms of performance.

6.2 Influence of Specialized Task Classifiers

Initially the framework was built with the amalgamation of the 39 individual phoneme detectors. We decided to add Specialized Task classifiers as described in section 4.3 to improve the performance. Given below is a snapshot of how the addition of the specialized task classifiers improved our performance:

Specialized Tasks Influence Results		
	39 Phoneme Detectors	39 Phoneme Detectors + 10 Specialized Tasks
Training Accuracy	88.8502%	90.1817%
Validation Accuracy	82.87019%	83.9965%

As seen from the above results, we see that the addition of the specialized tasks has improved both our training and validation accuracy of the complete frame level phoneme classification network.

6.3 Interpretability Graphs

Our main goal with this project was to implement a high-performance frame to phoneme discriminator that could be easily interpretable. Using Human-Guided AI practices, we decided to implement our final classifier as an ensemble of specialized neural subnetworks, each of which was trained with a particular objective. We developed two different subnetwork types: phoneme detectors and specialized task classifiers. We also added the functionality to plot interpretability graphs for each prediction. As part of the evaluation and results of our architecture, we are including some sample graphs on a correct and incorrect classification.

6.3.1 Correct Classification

In this case, the ground truth label for the provided frame as the phoneme AW.

1. **Final Model Probabilities**: the highest probability (phoneme selected by the network) is indeed phoneme AW, without any competing phonemes.
2. **Shallow Network Output Probabilities**:

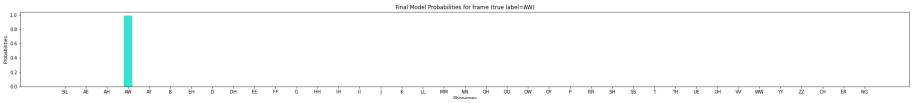

Figure 7: Final Model Probabilities (Correct Classification)

- Phoneme Detectors: for this particular frame, we can see that the highest probabilities were given by the AH and AW detectors.
- Specialized Task Classifiers:
  - Specialized 1_1 (task: vowels vs consonants, where class 1 are vowels) displays a high probability. Indeed, AW is a vowel.
  - Specialized 3_2 (task: high vowels vs low vowels, where class 2 are low vowels) displays a high probability. Indeed, AW is a low vowel.
  - Specialized 5_0 (task: SS vs ZZ, where class 0 is none of them) displays a high probability. Indeed, AW is neither of these phonemes.
  - Specialized 9_2 (task: EE vs AW, where class 2 is AW) displays high probability. Indeed, AW is our target phoneme.
  - Specialized 10_2 (task: AH vs AW, where class 2 is AW) displays high probability. Indeed, AW is our target phoneme. The rest of the specialized classifiers distinguish between pair of phonemes other than AW, so the high probability observed for class 0 is expected

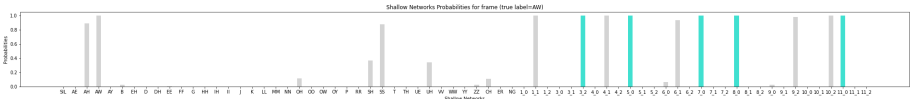

Figure 8: Shallow Network Output (Correct Classification)

3. **Shallow Networks Contribution to fired neuron**: the model is assigning a great contribution to the output of the AW phoneme detector, followed by the specialized task classifiers 1_1, 3_2, 9_2, 10_2 (refer to previous point for an explanation on why these are relevant)

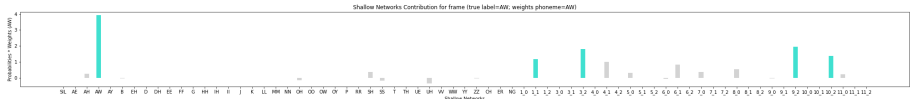

Figure 9: Shallow Network Contribution (Correct Classification)

6.3.2 Incorrect Classification

In this case, the ground truth label for the provided frame as the phoneme AW, however our model misclassified it as OW.

1. **Final Model Probabilities**: the highest probability (phoneme selected by the network) is phoneme OW, followed (although by great difference) by our true label AW.

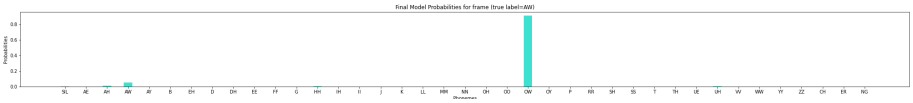

Figure 10: Final Model Probabilities (Incorrect Classification)

2. **Shallow Network Output Probabilities**:
   - Phoneme Detectors: for this particular frame, we can see higher probabilities being output by more phoneme detectors. Specifically, AH, AW and OW show high probabilities

- Specialized Task Classifiers:
  - Specialized 1 (task: vowels vs consonants) this detector seems to struggle, giving some probabilities both for class 0 and class 1 (class 0 being none of the vowel/consonants, class 1 vowels, and class 2 consonants). AW is a vowel, so we should only have had high probability for class 1
  - Specialized 3_2 (task: high vowels vs low vowels, where class 2 is low vowels) displays a high probability. AW is a low vowel, so the system is correct in identifying it as class 2.
  - Specialized 5_0 (task: SS vs ZZ, where class 0 is none of them) displays a high probability. Indeed, AW is neither of these phonemes
  - Specialized 9_2 (task: EE vs AW, where class 2 is AW) displays high probability. Indeed, AW is our target phoneme.
  - Specialized 10_2 (AH vs AW, where class 2 is AW) displays high probability. Indeed, AW is our target phoneme.

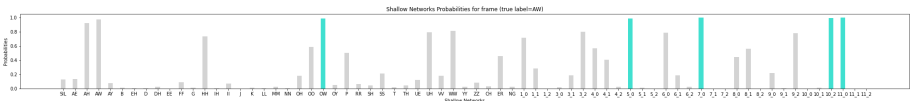

Figure 11: Shallow Network Output (Incorrect Classification)

3. **Shallow Networks Contribution to fired neuron**: the model is assigning a great contribution to the output of the OW phoneme detector, followed by specialized 3_2, 6_0 (B vs P, where class 0 is none of them), 9_2, 10_2

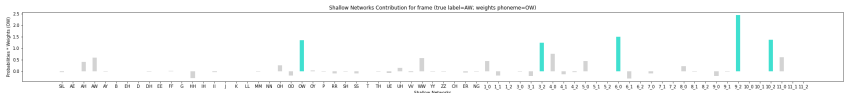

Figure 12: Shallow Network Contribution - Ground Truth Neuron OW (Incorrect Classification)

4. **Shallow Networks Contribution to ground truth neuron**: the neuron that should had fired (AW phoneme) assigns great contribution to the output of the AW phoneme detector, followed by specialized 3_2, 9_2, 10_2.

Figure 13: Shallow Network Contribution - Ground Truth Neuron AW (Incorrect Classification)

7 Conclusion and Future Work

In conclusion, as neural networks are being adopted by businesses universally, interpretability in such models is needed now more than ever, especially when they are being used to perform tasks that have the potential to significantly impact human life and behaviour. In this research we propose a framework that offers interpretability with respect to it's output and has comparable performance to that of a state of the art "black-box" neural network. Our system can be used in isolation as a frame-level phoneme recognizer or can be plugged in as the frontend to a deeper backend system capable of performing specialized tasks like word detection by back-propagating the error all the way to the frontend. The system is an amalgamation of 49 high performing specialized discriminators that were each trained to perform a specific human-guided task. The performance of the system is comparable to that of a zero context Multi-Layer Perceptron (MLP) Classifier designed as part of one of the homework's of the 11785 Deep Learning course at Carnegie Mellon. Our network is human interpretable as at any point in time, we can interpret the reasoning behind the firing of a certain output by generating four different types of Interpretability graphs - namely the Final Model

Probabilities, Shallow Networks Output Probabilities, Shallow Networks Contribution to fired neuron and Shallow Networks Contribution to ground truth neuron. We think that this framework is a great alternative to traditional "black-box" MLPs. In the future, we would perform a thorough analysis of misclassified instances to identify commonly confused phonemes. Based on the results, we would like to incorporate new specific specialized tasks for discriminating such confusable phonemes to improve our system. We also intend to augment our dataset to incorporate sequential data. This would allow us to leverage powerful architectures such as RNNs, Encoder-Decoder with attention to perform advanced tasks like speech recognition.

The interpretability of this system is a small step in the direction of making Deep Learning networks more transparent! Debugging of input edge cases, removing bias, and feature engineering would become more efficient as we would see how different features affect the output clearly. These use cases have important consequences in the real world. For example, Apple Card's algorithm to determine the credit-worthiness of applicants landed the organization in legal trouble when it was discovered that the algorithm was biased against female applicants. These kinds of problems can be avoided if such biases are identified and removed in the development and training phase of the pipeline itself.

Interpretable models like ours would make it easier to solve such issues. Moreover, our system offers a lot of flexibility to engineers and scientists. It can be easily extended by adding new specialized tasks. The system can also be used as an alternative to the lower layers in a deep end-to-end speech recognition system.

Finally, from a real-world standpoint, these networks can also be extended to domains such as assistive technologies, mobility, and healthcare.

8 Division of Work

This work is a result of an equal contribution of all team members.

References

[1] Agarwal, R., Frosst, N., Zhang, X., Caruana, R., Hinton, G.E. (2020). Neural Additive Models: Interpretable Machine Learning with Neural Nets. ArXiv, abs/2004.13912.

[2] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 1135–1144.

[3] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

[4] Tjoa, Erico Guan, Cuntai. (2019). A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI (Preprint).

[5] https://royalsociety.org/topics-policy/projects/explainable-ai/

[6] Kennedy, Mary. (2019). What Do Artificial Intelligence (AI) and Ethics of AI Mean in the Context of Research Libraries?. Research Library Issues. 3-13. 10.29242/rli.299.1.

[7] Zhang, Jingyuan Sun, Mingming Feng, Yue Li, Ping. (2020). Learning Interpretable Relationships between Entities, Relations and Concepts via Bayesian Structure Learning on Open Domain Facts. 8045-8056. 10.18653/v1/2020.acl-main.717.

[8] Bhatt, Shobha Jain, Anurag Dev, Amita. (2021). Feature Extraction Techniques with Analysis of Confusing Words for Speech Recognition in the Hindi Language. Wireless Personal Communications. 10.1007/s11277-021-08181-0.

[9] Anguita, J. & Hernando, Javier & Peillon, Stéphane & Bramoulle, Alexandre. (2005). Detection of confusable words in automatic speech recognition. Signal Processing Letters, IEEE. 12. 585 - 588. 10.1109/LSP.2005.851256.

[10] Xin Lei1 & Andrew Senior & Alexander Gruenstein & Jeffrey Sorensen (2013) Accurate and Compact Large Vocabulary Speech Recognition on Mobile Devices, *INTERSPEECH 2013*.

[11]Roche, F., Hueber, T., Limier, S., and Girin, L., “Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models”, arXiv e-prints, 2018.

[12] Jan Chorowski, Ron J. Weiss, Samy Bengio, and Aaron van den Oord. 2019. Unsupervised Speech Representation Learning Using WaveNet Autoencoders. IEEE/ACM Trans. Audio, Speech and Lang. Proc. 27, 12 (Dec. 2019), 2041–2053.

[13] van den Oord, A., “WaveNet: A Generative Model for Raw Audio”, arXiv e-prints, 2016.

[14] Kong, J., Kim, J. and Bae, J., 2020. HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. arXiv preprint arXiv:2010.05646.

[15] Google VAD Python Module Link:

[16] https://librosa.org/

[17] Article Link: https://towardsdatascience.com/extreme-rare-event-classification-using-autoencoders-in-keras-a565b386f098

APPENDIX

The performance evaluation reports of each of the phoneme detectors and the specialized classifiers have been attached below.

Shallow_Detector_Report_Final_Updated

Phoneme Tag	Class	Precision_Train	Recall_Train	F1_Train	Support_Train	Precision_Dev	Recall_Dev	F1_Dev	Support_Dev
SIL	0	0.9340575455079272	0.8007147890868821	0.8622614917606245	19866	0.9030511811023622	0.7451776649746192	0.8165535654689063	4925
SIL	1	0.8796290665855884	0.9626339256005856	0.9192615658362989	30054	0.8115615615615616	0.9320572512502155	0.867645878481419	5799
AE	0	0.9718706047819972	0.9906810035842294	0.9811856585019524	2790	0.9383070301291249	0.989409984871407	0.9631811487481591	661
AE	1	0.9462809917355371	0.8513011152416357	0.8962818003913895	538	0.9351851851851852	0.7013888888888888	0.8015873015873016	144
AH	0	0.9758403361344538	0.9921680313278747	0.983936451897617	2809	0.9616519174041298	0.965925925925926	0.9637841832963784	675
AH	1	0.9589552238805971	0.8816466552315609	0.9186773905272564	583	0.8357142857142857	0.8181818181818182	0.8268551236749117	143
AW	0	0.9732111385266127	0.9896057347670251	0.9813399680113738	2790	0.9667170953101362	0.9815668202764977	0.9740853658536585	651
AW	1	0.9477477477477477	0.8737541528239202	0.909248055315471	602	0.9210526315789473	0.8641975308641975	0.89171974522293	162
AY	0	0.9569056870363829	0.9723618090452262	0.964571835499377	2786	0.9683734939759037	0.9568452380952381	0.9625748502994012	672
AY	1	0.888243831640058	0.8337874659400545	0.860154602951511	734	0.8333333333333334	0.8734939759036144	0.8529411764705883	166
B	0	0.9484536082474226	0.9831092726645985	0.965470548408937	2901	0.93646408839779	0.9755395683453237	0.9556025369978858	695
B	1	0.8093385214007782	0.5730027548209367	0.6709677419354839	363	0.746268656716418	0.5208333333333334	0.6134969325153375	96
EH	0	0.9808612440191388	0.9925758553905746	0.986683779881277	3098	0.9782608695652174	0.9947984395318595	0.9864603481624759	769
EH	1	0.9601386481802426	0.9022801302931596	0.9303106633081445	614	0.9622641509433962	0.8571428571428571	0.9066666666666666	119
D	0	0.9452449567723343	0.9820359281437125	0.9632892804698973	2672	0.9438040345821326	0.9849624060150376	0.9639440765268579	665
D	1	0.7931034482758621	0.5476190476190477	0.647887323943662	336	0.696969696969697	0.3709677419354839	0.4842105263157895	62
DH	0	0.9398655818889282	0.9862657757980697	0.9625067922477811	2694	0.9225251076040172	0.9669172932330827	0.9441997063142438	665
DH	1	0.9261477045908184	0.7318611987381703	0.8176211453744495	634	0.7924528301886793	0.6086956521739131	0.6885245901639344	138
EE	0	0.9544969783149663	0.9714182344428365	0.9628832705755783	2764	0.916923076923077	0.9551282051282052	0.9356357927786499	624
EE	1	0.8975356679636836	0.8439024390243902	0.8698931489629164	820	0.86	0.7610619469026548	0.8075117370892019	226
FF	0	0.9463286137635825	0.9917184265010351	0.9684919966301601	2898	0.9179415855354659	0.9806835066864784	0.9482758620689655	673
FF	1	0.9503105590062112	0.7379421221864951	0.8307692307692307	622	0.8898305084745762	0.6402439024390244	0.7446808510638299	164
G	0	0.9595536959553695	0.9874416935773233	0.9732979664014146	2787	0.9630723781388478	0.978978978978979	0.9709605361131795	666
G	1	0.8694029850746269	0.667621776504298	0.7552674230145866	349	0.8157894736842105	0.7126436781609196	0.7607361963190185	87
HH	0	0.9374824782730586	0.9864306784660767	0.9613339082938048	3390	0.8951132300357568	0.9652956298200515	0.9288806431663573	778
HH	1	0.885286783042394	0.6141868512110726	0.7252298263534219	578	0.7352941176470589	0.4601226993865031	0.5660377358490566	163
IH	0	0.9638433210579176	0.9829293274155002	0.9732927653820149	2929	0.9829545454545454	0.9746478873239437	0.9787835926449787	710
IH	1	0.8933901918976546	0.7950664136622391	0.8413654618473895	527	0.85	0.8947368421052632	0.8717948717948718	114
II	0	0.9749470712773465	0.9871382636655949	0.9810047931830285	2799	0.963020030816641	0.9689922480620154	0.9659969088098918	645
II	1	0.9421221864951769	0.8919330289193302	0.9163408913213448	657	0.8895027624309392	0.8702702702702703	0.8797814207650274	185
J	0	0.958810888252149	0.9878228782287823	0.9731006906579426	2710	0.9461883408071748	0.9634703196347032	0.9547511312217195	657
J	1	0.8821428571428571	0.6823204419889503	0.7694704049844238	362	0.6883116883116883	0.5955056179775281	0.6385542168674699	89
K	0	0.970304114490161	0.9934065934065934	0.9817194570135747	2730	0.9610194902548725	0.9831288343558282	0.9719484457922669	652
K	1	0.9154929577464789	0.7014388489208633	0.7942973523421588	278	0.828125	0.6708860759493671	0.7412587412587414	79
LL	0	0.952074391988555	0.9921729407379799	0.9717101660887023	2683	0.9198767334360555	0.9770867430441899	0.9476190476190475	611
LL	1	0.9605263157894737	0.7922480620155039	0.8683092608326253	645	0.9020979020979021	0.712707182320442	0.7962962962962963	181
MM	0	0.9402777777777778	0.9912152269399708	0.9650748396293657	2732	0.9225251076040172	0.9892307692307692	0.9547141796585004	650
MM	1	0.9625	0.7817258883248731	0.8627450980392157	788	0.9513888888888888	0.7172774869109948	0.817910447761194	191
NN	0	0.9578090303478904	0.9915708812260536	0.9743975903614457	2610	0.9365079365079365	0.9800664451827242	0.9577922077922079	602
NN	1	0.9648562300319489	0.841225626740947	0.8988095238095237	718	0.9302325581395349	0.8	0.8602150537634408	200
OH	0	0.9526184538653366	0.9841737210158262	0.9681390296886313	2717	0.9619883040935673	0.9850299401197605	0.9733727810650888	668
OH	1	0.9264957264957265	0.802962962962963	0.8603174603174604	675	0.9236641221374046	0.8231292517006803	0.8705035971223022	147
OO	0	0.9601567602873938	0.9882352941176471	0.9739937054828557	2975	0.9467213114754098	0.9774330042313117	0.9618320610687023	709
OO	1	0.9235807860262009	0.7761467889908257	0.843469591226321	545	0.8518518518518519	0.7022900763358778	0.7698744769874476	131
OW	0	0.9837340876944838	0.9932167083184577	0.9884526558891455	2801	0.9968354430379747	0.9559939301972686	0.9759876065065841	659
OW	1	0.9725433526011561	0.9360222531293463	0.9539333805811481	719	0.8578431372549019	0.9887005649717514	0.9186351706036745	177
OY	0	0.9477401129943502	0.9824304538799414	0.9647735442127965	2732	0.9388379204892966	0.9374045801526718	0.9381207028265853	655
OY	1	0.9230769230769231	0.7955801104972375	0.85459940652819	724	0.7657142857142857	0.7701149425287356	0.7679083094555873	174
P	0	0.960586319218241	0.9935983827493261	0.9768135144087446	2968	0.9402173913043478	0.9885714285714285	0.9637883008356546	700
P	1	0.8538461538461538	0.47844827586206895	0.6132596685082872	232	0.7837837837837838	0.3972602739726027	0.5272727272727272	73
RR	0	0.945597709377237	0.992114156965828	0.9682975994135972	2663	0.9364844903988183	0.9649923896499238	0.9505247376311845	657
RR	1	0.9553191489361702	0.7470881863560732	0.838468720821662	601	0.7909090909090909	0.6692307692307692	0.725	130
SH	0	0.9589725294327506	0.984254851702673	0.9714492229851825	2731	0.9279538904899135	0.9757575757575757	0.9512555391432791	660
SH	1	0.9269949066213922	0.8260211800302572	0.8736	661	0.872	0.6855345911949685	0.7676056338028169	159
SS	0	0.9636299435028248	0.9855543517515348	0.974468844849134	2769	0.9894419306184012	0.9647058823529412	0.9769173492181683	680
SS	1	0.9418604651162791	0.8628495339547271	0.9006254343293953	751	0.8628571428571429	0.9556962025316456	0.906906906906907	158
T	0	0.9796954314720813	0.9930172730613744	0.9863113706880818	2721	0.977645305514158	0.9924357034795764	0.984984984984985	661
T	1	0.8978494623655914	0.7488789237668162	0.8166259168704156	223	0.8571428571428571	0.6666666666666666	0.75	45
TH	0	0.9366500829187396	0.9836293974225009	0.9595650696568128	2871	0.8909348441926346	0.9797507788161994	0.9332344213649852	642

TH	1	0.8753315649867374	0.6333973128598849	0.734966592427617	521	0.8796296296296297	0.5523255813953488	0.6785714285714285	172
UE	0	0.9595070422535211	0.992352512745812	0.9756534192624419	2746	0.9523809523809523	0.9865470852017937	0.9691629955947136	669
UE	1	0.9619565217391305	0.8219814241486069	0.8864774624373958	646	0.9256198347107438	0.7724137931034483	0.8421052631578948	145
UH	0	0.96399074991741	0.9911684782608695	0.9773907218221403	2944	0.9578059071729957	0.9605077574047954	0.9591549295774647	709
UH	1	0.9287671232876712	0.7566964285714286	0.833948339483395	448	0.7307692307692307	0.7169811320754716	0.7238095238095238	106
VV	0	0.92019600980049	0.9902071563088513	0.9539187227866472	2655	0.9227467811158798	0.9923076923076923	0.9562638991845811	650
VV	1	0.9514018691588785	0.6906377204884667	0.800314465408805	737	0.9557522123893806	0.6666666666666666	0.7854545454545454	162
WW	0	0.9471658502449265	0.987956204379562	0.9671311182565202	2740	0.9093610698365527	0.972972972972973	0.9400921658986175	629
WW	1	0.9187192118226601	0.7118320610687023	0.8021505376344087	524	0.8468468468468469	0.6064516129032258	0.706766917293233	155
YY	0	0.9403669724770642	0.9907063197026023	0.9648805213613324	2690	0.9008746355685131	0.9840764331210191	0.9406392694063926	628
YY	1	0.9418604651162791	0.705574912891986	0.8067729083665339	574	0.8969072164948454	0.5612903225806452	0.6904761904761905	155
ZZ	0	0.9533527696793003	0.9856819894498869	0.9692478695813264	2654	0.9212481426448736	0.9810126582278481	0.950191570881226	632
ZZ	1	0.934931506849315	0.8100890207715133	0.8680445151033386	674	0.9016393442622951	0.6748466257668712	0.7719298245614035	163

Specialized_Detector_Report_Final

Task Name	Phoneme Tag	Class	Precision_Train	Recall_Train	F1_Train	Support_Train	Precision_Dev	Recall_Dev	F1_Dev	Support_Dev
1_vowel_vs_consonant	['EE', 'IH', 'EH', 'AE', 'UH', 'ER', 'AH', 'AW', 'OO', 'UE']	0	0.91070773744041 07	0.8638260869565217	0.8866476258479115	5750	0.9049881235154394	0.8264642082429501	0.8639455782312925	1383
1_vowel_vs_consonant	['FF', 'HH', 'MM', 'NN', 'NG', 'RR', 'SS', 'SH', 'VV', 'WW', 'YY', 'ZZ']	1	0.9019534184823441	0.9366710013003902	0.9189844348047972	7690	0.8825831702544031	0.9376299376299376	0.909274193548387	1924
3_highvowel_vs_lowvowel	['EE', 'IH', 'UE', 'OO']	0	0.9475106136626785	0.9638790734197095	0.955624756714675	2547	0.8880813953488372	0.9918831168831169	0.9371165644171779	616
3_highvowel_vs_lowvowel	['AE', 'AH', 'AW']	1	0.9457866823806718	0.9218839747271683	0.9336823734729494	1741	0.986737400530504	0.8285077951002228	0.9007263922518159	449
4_voiced_vs_unvoiced_fricatives	['DH', 'VV', 'ZZ']	0	0.9149043303121853	0.8898139079333987	0.9021847070506455	2042	0.8137254901960784	0.896328293736501	0.8530318602261048	463
4_voiced_vs_unvoiced_fricatives	['FF', 'SS', 'SH', 'TH']	1	0.914187643020595	0.9341387373343726	0.9240555127216653	2566	0.9207920792079208	0.8545176110260337	0.886417791898332	653
5_ss_vs_zz	['SS']	0	0.9565807327001357	0.9527027027027027	0.954637779282329	740	0.8478260869565217	0.9873417721518988	0.9122807017543859	158
5_ss_vs_zz	['ZZ']	1	0.9478390461997019	0.9520958083832335	0.9499626587005228	668	0.9854014598540146	0.8282208588957055	0.9	163
6_b_vs_p	['B']	0	0.875	0.8700564971751412	0.8725212464589235	354	0.736	0.9583333333333334	0.832579185520362	96
6_b_vs_p	['P']	1	0.7946428571428571	0.8018018018018018	0.7982062780269059	222	0.9090909090909091	0.547945205479452	0.6837606837606837	73
7_dh_vs_th	['DH']	0	0.901453957996769	0.8815165876777251	0.8913738019169328	633	0.8842975206611571	0.7753623188405797	0.8262548262548263	138
7_dh_vs_th	['TH']	1	0.8592870544090057	0.882466281310212	0.870722433460076	519	0.8359788359788359	0.9186046511627907	0.8753462603878116	172
8_ww_vs_yy	['WW']	0	0.9402390438247012	0.9094412331406551	0.9245837414299707	519	0.9154929577464789	0.8387096774193549	0.8754208754208754	155
8_ww_vs_yy	['YY']	1	0.9197952218430034	0.9472759226713533	0.9333333333333335	569	0.8511904761904762	0.9225806451612903	0.8854489164086687	155
9_ee_vs_aw	['EE']	0	0.9852034525277436	0.9852034525277436	0.9852034525277436	811	0.9294605809128631	0.9911504424778761	0.9593147751605996	226
9_ee_vs_aw	['AW']	1	0.9798994974874372	0.9798994974874372	0.9798994974874372	597	0.9863945578231292	0.8950617283950617	0.9385113268608414	162
10_ah_vs_aw	['AH']	0	0.9446428571428571	0.9329805996472663	0.9387755102040817	567	0.8502994011976048	0.993006993006993	0.9161290322580644	143
10_ah_vs_aw	['AW']	1	0.9358108108108109	0.947008547008547	0.9413763806287172	585	0.9927536231884058	0.845679012345679	0.9133333333333333	162
11_mm_vs_nn	['MM']	0	0.891644908616188	0.8904823989569752	0.8910632746249184	767	0.8720379146919431	0.9633507853403142	0.9154228855721392	191
11_mm_vs_nn	['NN']	1	0.8810198300283286	0.8822695035460993	0.881644239546421	705	0.9611111111111111	0.865	0.9105263157894737	200