Deep Learning for Speech Signals – HW4

Submitters:

Lidor Zino I.D: 313551186

Suf Rubinstein I.D: 205853633

Date: 26/01/2025

Part 2

Q1

- **Training Time**:
    - **CNN**: Requires a training phase where the model learns to generalize features across 30 classes (spoken words). This involves iterating through 30,000 samples (30 words × 1,000 speakers) over several epochs, which is computationally intensive but is efficient for such dataset.
    - **DTW**: Does not require a training phase. However, every test instance must be compared to all training samples, which bypasses training but shifts the burden to inference.
- **Inference Time**:
    - **CNN**: After training, inference is very fast. The model simply performs a forward pass, leveraging pre-learned features to classify the input. Inference time is independent of the train-dataset size.
    - **DTW**: Slower inference because it involves computing the distance between the test sample and all training samples. For the GCommands dataset, this means performing DTW calculations for up to 30,000 comparisons, which is computationally expensive.
- **Memory Requirements**:
    - **CNN**: Memory is primarily used to store the model parameters (weights) and intermediate activations during training and inference. These are constant regardless of the dataset size during inference.
    - **DTW**: Requires storing all 30,000 training samples to compare them with the test sample during inference. This grows linearly with the dataset size, making it less memory-efficient than CNNs for larger datasets.

- **Suitability for GCommands**:
    - **CNN**: Ideal for large, diverse datasets like GCommands. It learns a robust feature representation and generalizes well, making it efficient for classification tasks with many classes and samples per class.
    - **DTW**: Best suited for smaller datasets or scenarios where training time is a bottleneck. For GCommands, DTW's inference cost becomes prohibitive as the dataset size grows.

Q2

- **Dataset A (50 samples)**:
    - Use **DTW**. Since the dataset is small, the lack of a training phase and lower computational overhead make DTW more suitable. The inference time will also be manageable given the small number of samples.
- **Dataset B (1,000,000 samples)**:
    - Use **CNN**. Training a CNN model upfront would be computationally intensive, but once trained, it provides efficient inference. DTW would be infeasible for such a large dataset due to the high memory and computational requirements during inference.

Q3

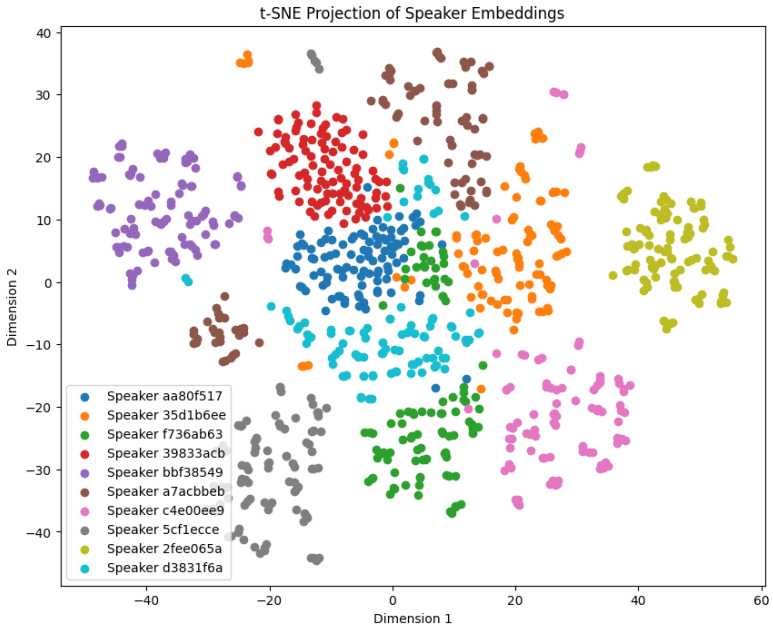
We may use CNN that gets the MFCCs as an input.

- We would design the CNN to process the MFCCs and produce a fixed-size output vector with 7 digits.
- In the output of the CNN we would flatten the feature map into 1D vector.
- Then we would use a fully connected layer to map the features to 7 separate outputs.
- At the end we would use SoftMax activation for each of the 7 outputs.
- The CNN outputs a 7-digit vector, where each position corresponds to a predicted digit

Part 3

3.2

a. this is the output plot:



We have conclusions mainly from to properties:
- Certain Colors Being in Their Own Zone
- Colors Very Close in the Center Yet Not Overlapping

Certain Colors Being in Their Own Zone:

- **Well-Isolated Clusters:** When certain colors (speakers) are in their own zones and show minimal overlap with others, it indicates that the model has successfully captured unique and distinguishable characteristics of those speakers' voices.
- **Distinctive Features:** These speakers likely have distinct vocal patterns or speech characteristics that the embedding model can separate cleanly.

Colors Very Close in the Center Yet Not Overlapping:

- **Speakers with Similar Characteristics:** The close proximity of clusters in the center suggests that these speakers may share some common vocal features, such as pitch, tone, or accent, making them harder to distinguish.
- **Good but Subtle Separation:** Even though the clusters are close, the fact that they "almost don't combine" (i.e., remain somewhat separate) suggests that the embedding model is still able to capture enough subtle differences to differentiate these speakers. This is an indicator of **fine-grained discrimination** by the model.
- **Potential Classification Challenges:** The closeness of the clusters could lead to some misclassifications during the cross-validation task. For example, in the classification results, this might correspond to slightly lower accuracies for these specific speakers.

b.

this is the table of accuracy and std deviation from the notebook:

	Mean Accuracy	Std Deviation
LDA	0.9635	0.009301
SVM	0.9690	0.009434
Random Forest	0.8775	0.008515
Logistic Regression	0.9655	0.005788

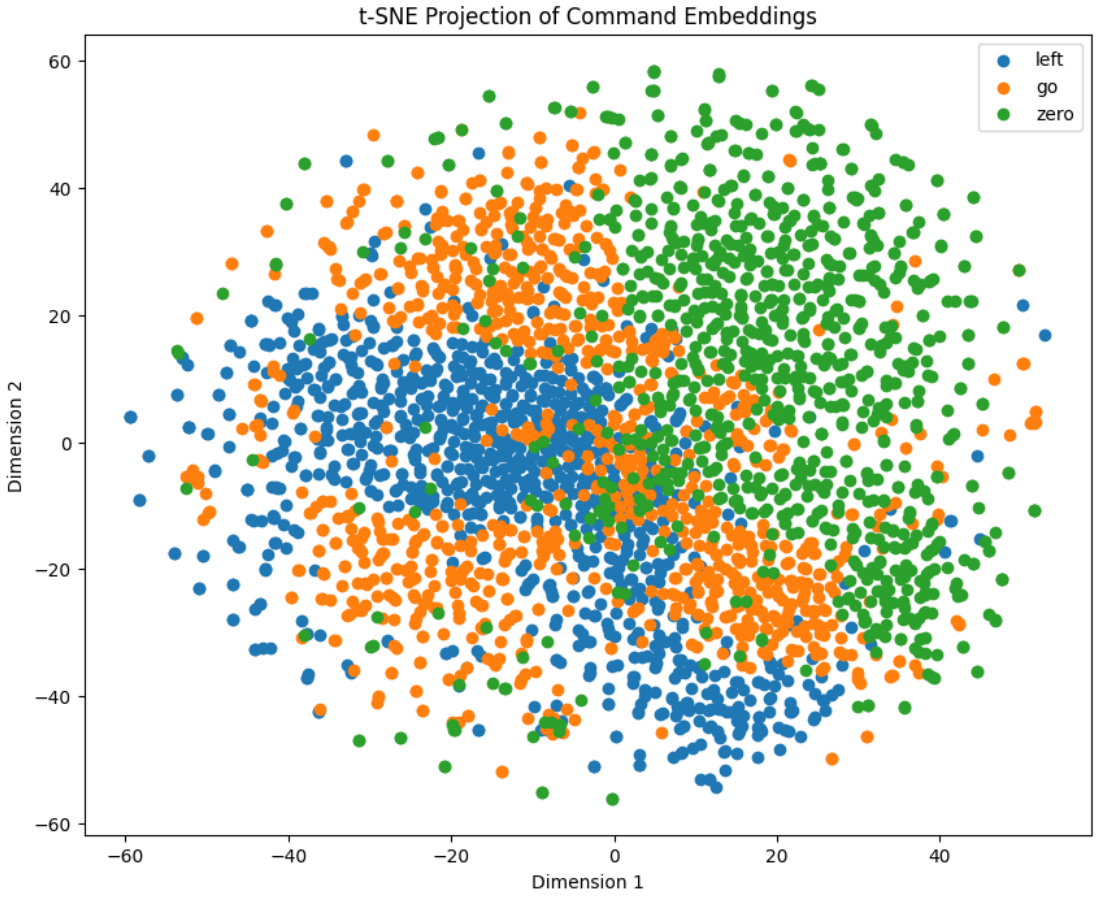
We can see that SVM, LDA and Logistic Regression did the best among these 4, since they have the highest mean accuracy (96.35%, 96.9%, 96.55%). The std deviation may vary by 0.004, but all of them have small std deviations.
random forest is also quite accurate with 87.75% mean accuracy

c.

x – vectors are good representation for speaker classification task; we can see this in the results that weather we change our classifier or not, the results are good – SVM and Logistic regression have very high mean accuracy, and very low std Deviation. Random forest has high mean accuracy, and very low std deviation.

3.3

These are the results for command classification:



As we can see, the commands "left" and "zero" are quite separatable; note that there are some overlaps, and there are a group of green dots in the middle of the blues, but they are quite separatable in general. On the other hand, the command "go" is somewhat separatable from the others. The orange dots are all over the plot, but there are 3 groups in which they are quite isolated – bottom left next to the blues,

bottom right between the green and the blues, and center top where they are not quite isolated but there is a majority of orange dots.

- It means that the model quite succeeded to capture a unique and distinguishable characteristics of "zero" and "left" commands, but not for "go".
- The commands have some common features
- As mentioned before, the closeness could lead to some misclassifications during the cross-validation task.

b.

```
Classifier Performance:
                     Mean Accuracy  Std Deviation
LDA                       0.893667       0.014735
Logistic Regression       0.858333       0.006912
Random Forest             0.823000       0.012311
```

Since SVM would take too much time, we chose to examine the others.

We can see that no classifier has good results as in classifying as in the previous task. mean accuracy got lower, and std deviation got higher.

In conclusion, x-vectors are not good in representing command classification in comparison to speakers' classification as we've seen before. It is better to find another representation.

3.4

Why X-Vectors Excel at Speaker Classification

1. **Speaker-Discriminative Nature**:
    - X-vectors are designed for **speaker recognition**, as outlined in the paper. They are optimized during training to represent speaker characteristics by classifying thousands of speakers in the training dataset. This makes them highly effective for tasks like speaker verification or identification, which explains your high accuracies for speaker classification tasks.
2. **Linear Separability**:
    - The paper highlights that x-vectors, especially after length normalization and using PLDA, work well with **linear classifiers** like LDA and Logistic Regression. This is consistent with the results, where these models performed exceptionally well for speaker classification.
3. **DNN-Based Embeddings**:
    - X-vectors leverage temporal pooling to aggregate frame-level features into a fixed-dimensional embedding. This process captures speaker-specific traits while being robust to variations in speech duration and background noise.

Why X-Vectors Struggle with Command Classification

1. **Lack of Command-Specific Features**:

- X-vectors are trained to distinguish speakers, not the content of their speech (e.g., commands like "left" or "go"). The embeddings primarily encode speaker identity and are less likely to capture fine-grained, phonetic information relevant for command classification.

2. **Command Variability**:
    - Unlike speaker traits, commands may have more **contextual and phonetic variability** (e.g., "left" versus "go"), which x-vectors are not explicitly designed to represent. This explains the lower accuracy for command classification.