Student Nguyen Truong Phong

QuanTEEM 2023-2025

Signal processing project

Spectrograms usage in machine-learning speech recognition algorithms

I. Project overview

A spectrogram is a 2D representation of a 3D surface, where time is plotted on the x-axis, frequency on the y-axis, and the intensity of the signal (amplitude or power) is represented using color or grayscale. This layout is analogous to an image where pixels represent different values. We can say spectrograms are images of sounds, therefore, by the usage of spectrogram, the voice recognition problem can be solved using image-processing techniques (CNN - Convolutional Neural Network).

This project studies how spectrogram can be used in a specific speech recognition problem – voice commands classification. We will build a CNN model to classify audio files into 8 classes and investigate how different methods of spectrogram preprocessing affect the result of the model.

II. Data exploration

Mini-speech commands dataset will be used. The dataset contains 8,000 files with length of approximately 1 second, separated equally into 8 classes ‘up’, ‘down’, ‘left’, ‘right’, ‘yes’, ‘no’, ‘stop’, ‘go’.

Our data will be separated into 3 datasets: Train set, validation set and test set. There are no duplications inside and between these datasets for the accuracy of our evaluation. The accuracy of the model while training will be evaluated by the validation set. And the accuracy of the model predicting in the test set will be our final evaluation of the model.

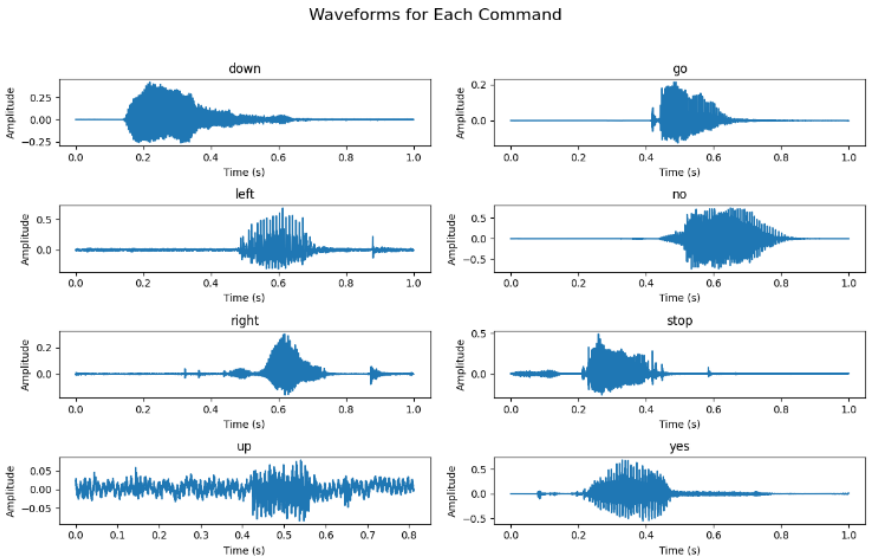

Fig 1. Amplitude graph of each label

III. Spectrogram Analysis

For compatibility while preprocessing data with the model, we use the method **'tf.signal.stft'** (spectrogram base function). This function in Tensorflow is particularly useful in tasks such as speech processing, audio analysis, and creating spectrograms for machine learning applications. It is used for computing the Short-Time Fourier Transform (STFT) of a signal. The STFT is a way to analyze how the frequency content of a signal changes over time. It breaks down a signal into its frequency components as they evolve in short, overlapping time windows. This function returns a Tensor of complex values representing the STFT. The shape of the output is [..., frames, fft_unique_bins] where frames is the number of frames in the STFT, and fft_unique_bins is calculated as fft_length // 2 + 1 (the unique components of the FFT).Each element in the output tensor corresponds to the complex value of a frequency bin at a specific time frame. The real part typically represents the magnitude (amplitude) of the frequency component, and the imaginary part represents the phase.

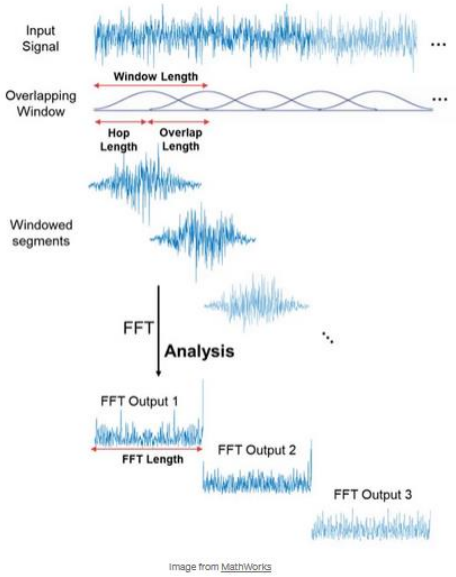

Fig 1. Short time Fourier Transform representation.

Below are the representations of spectrogram with different spectrogram preprocessing methods.

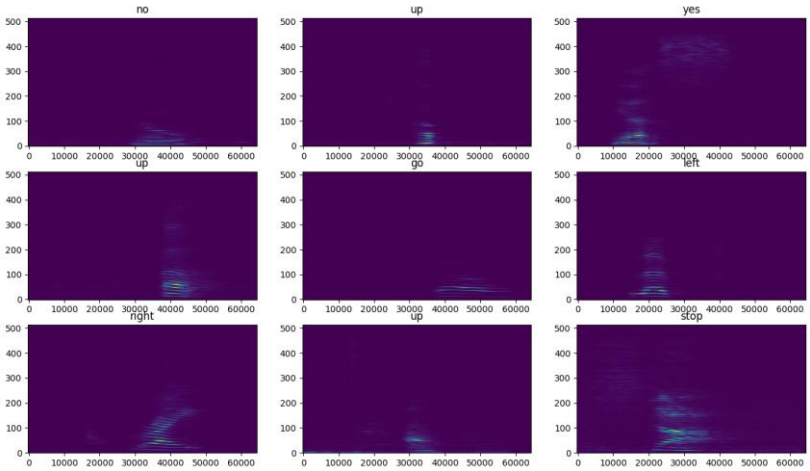

Fig 2. Spectrogram using the spectrogram base function only.

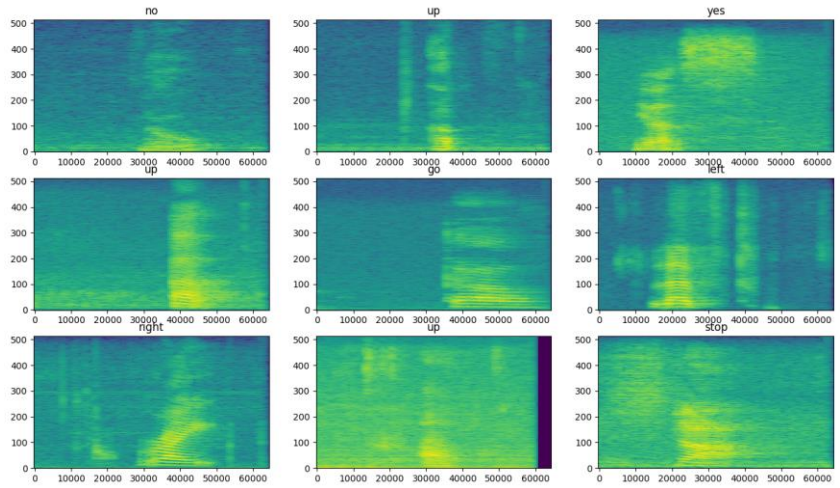

Fig 3. Spectrogram using the log scale spectrogram.

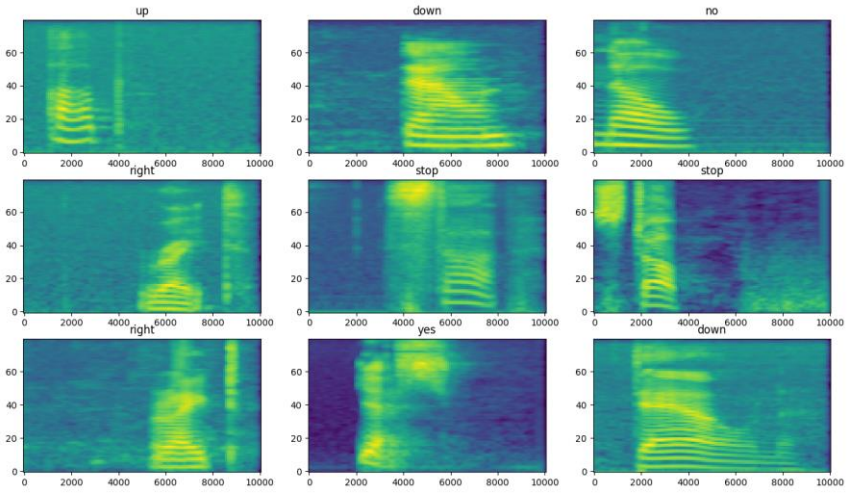

Fig 4. Normalized_mel_spectrogram

Spectrogram Analysis method	"Resolution"	Plot
Base spectrogram function	Low	Barely see the signal
Log scale spectrogram	Medium	Better image quality
Normalized mel_spectrogram	High	Detailed

Table 1. Comparison between the Spectrogram Analysis Methods

The log scale preprocessing procedure is motivated by the human perception of loudness which has logarithmic relationship with the physical energy of sound [1]. In 1937, Stevens, Volkmann, and Newmann proposed a unit of pitch such that equal distances in pitch sounded equally distant to the listener. This is called the mel scale [2].

The reason for normalizing the mel spectrogram is to enhance the model's performance during training. Normalization helps in achieving a consistent scale across different input samples, making the model less sensitive to variations in amplitude and improving convergence during training. This is particularly important for neural network models that might benefit from standardized input features.

We clearly saw that the normalized mel spectrogram preprocessing made the quality of the “images” better, and the information displayed are more detailed. This may indicate that the quality of the model will be better with normalized mel spectrogram.

III. CNN Model building

The structure of the CNN model will be displayed in Fig 5. For easy visualization, Dropout layer, Flatten layer and Attention mechanism layer (LSTM) will not be displayed.

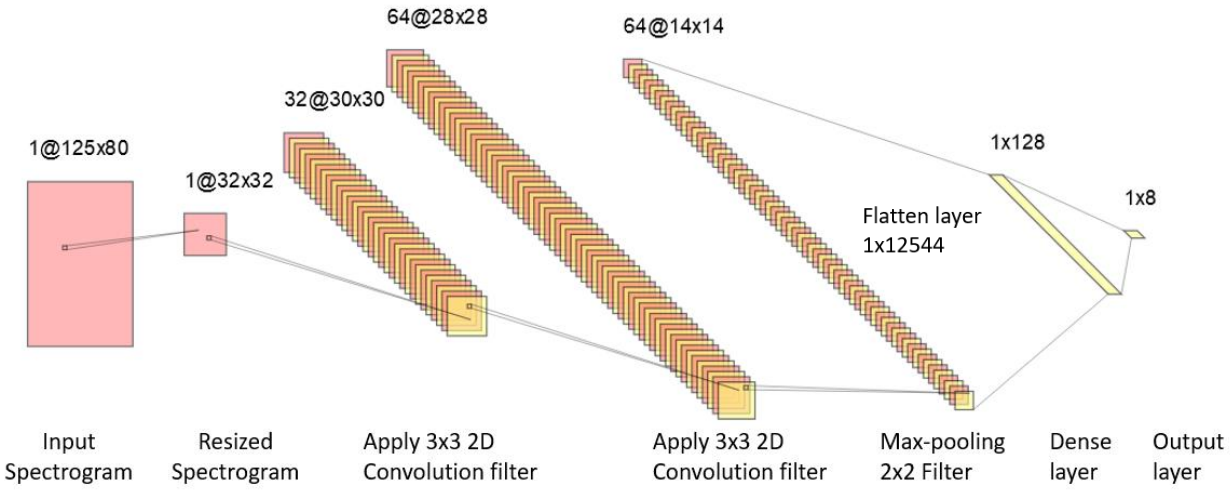

Fig 5. Basic structure of the CNN-model

Basic CNN Structure

The idea is to apply 3x3 filters to the previous layer. The filter will act as a feature-extractor for detecting different features of the image (in this case spectrogram). Type of typical filters: edge detection filters, gaussian blur filters, sharpening filters,…

- Input Layer:
  - The input layer receives the spectrogram data. Spectrogram can be understood as an image with size 125x80 in this case.
- Resizing Layer:
  - Resize the input to (32, 32)
- Normalization Layer:
  - Normalize the resized input.
- Convolutional Layers (Feature extraction):
  - Conv1: 32 filters, kernel size 3x3, ReLU activation. Apply 32 filter sized 3x3 to the (32,32) image, then convolute. As a result we have 32 filter with size (30,30).

    - Conv2: 64 filters, kernel size 3x3, ReLU activation. Apply filters to the (30,30) images received before. As a result, we have 64 filters with size (28,28).
    - MaxPooling: 2x2 max pooling. Max pooling will take a max value of each 2x2 square. As a result, we have 64 filters with half the size of the previous filter (14,14).
    - Dropout: 25% dropout for regularization. This layer randomly sets a fraction of the input units to zero during training, which can be interpreted as training different subnetworks on each iteration. This randomness during training helps prevent the model from becoming too specialized to the training data and enhances its ability to generalize to new, unseen data.
- Flatten Layer:
    - Flatten the output from the convolutional layers. The Flatten layer is introduced to convert the 3D tensor into a 1D tensor. This operation is necessary when transitioning from convolutional layers to dense layers, as dense layers expect 1D input vectors.
- Dense Layers:
    - Dense1: 128 units, ReLU activation.
    - Dropout2: 50% dropout for regularization.

Attention mechanism

The attention mechanism allows the model to dynamically focus on different parts of the input sequence during different windows. This can be particularly useful when certain parts of the sequence are more informative for the task at hand. The attention mechanism helps the model assign different levels of importance to different elements of the input sequence, enhancing its ability to capture long-range dependencies and improve performance, especially in tasks where specific parts of the input sequence are more relevant at different times.

- Reshape Layer:
    - Reshape the output for LSTM input: (-1, 128).
- LSTM Layer:
    - LSTM with 64 units, return sequences.
- Attention Mechanism:
    - Permute1: Permute dimensions.
    - Dense_Att: Dense layer with softmax activation for attention.
    - Permute2: Permute dimensions.
    - Multiply: Element-wise multiplication with LSTM output.
- Flatten Layer (Attention):
- Flatten the output from the attention mechanism.

Downstream layers (layers that are near output)

- Additional Dense Layers:
- Dense2: 128 units, ReLU activation.
- Dropout3: 50% dropout for regularization.
- Output Layer:
- Dense output layer with softmax activation for classification. The 1x8 vector of prediction will be here.

IV. Result

Accuracy and speed of training

One can conclude that using the normalized mel_spectrogram provided the superior accuracy (**88.7%**) to other type of spectrogram processing. This result agree with The Impact of Audio Input Representations on Neural Network based Music Transcription[3] where Mel spectrogram is the best choice for audio preprocessing. For each epoch is much slower in comparison to the base spectrogram function and log scale preprocessing. However, the convergence of the model takes less epochs. We chose the normalized mel spectrogram model for further analysis.

It is also observable that the training time here (about 5 min each) is for CPU only. Which is to say, building a fast model for voice recognition, even used for real-time voice recognition using spectrogram is promising.

Spectrogram Analysis method	Accuracy (%)	No. of epochs	Time (s)	Time per epoch (s)
Base spectrogram function	78.8	26	342	13.2
Using log scale spectrogram	82.5	18	232	12.9
Normalized mel_spectrogram	**88.7**	16	331	20.7

Table.2 Result of different spectrogram processing approaches to audio files

The training and validation curves for the model using normalized mel spectrograms showed no abnormalities. During training, the accuracy of the training dataset consistently increased with each epoch. However, the rate at which the accuracy of the validation dataset increased gradually slowed after each epoch.

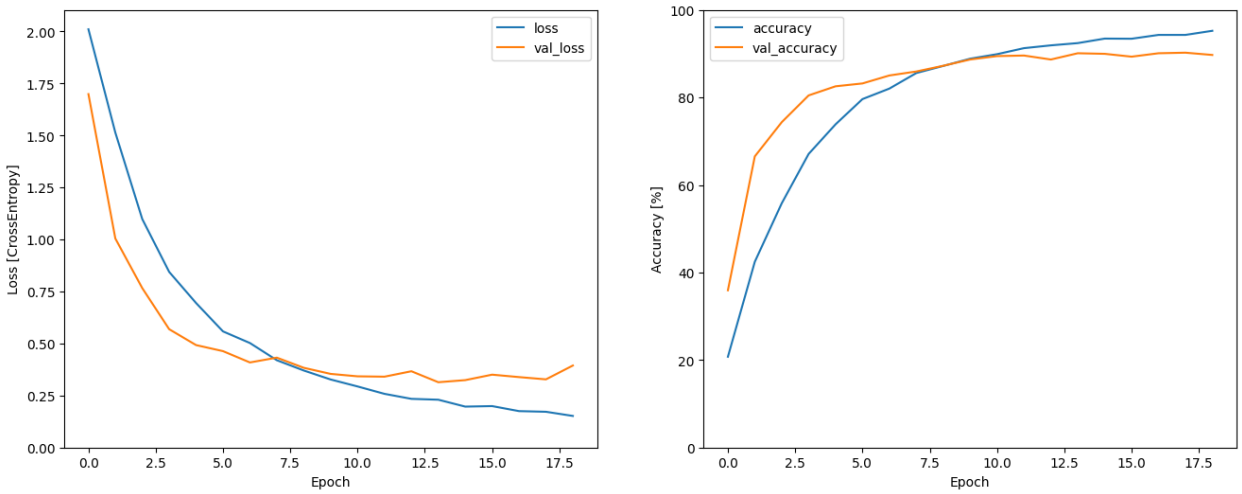

Fig 6. Training and validation loss and accuracy curves of the model using normalized mel spectrogram.

Confusion matrix

This normalized confusion matrix will evaluate the precision of the targeted label and the prediction of the model. If the model perfectly predicts all the test audio, the diagonal of the confusion matrix will be 1 and other cells will be 0.

In this case we see that the model could not classify well between the classes “down”, “go”, “no”, and “stop”. Especially in the case when the label is “go”, the model predicted to “no” 14%. This phenomenon can be explained because of the same vowel “o” of these two words. Also, the confusion between “stop” and “up” is quite high (8%). In fact, "up" and "stop" might have similar acoustic characteristics. The common ending sound “p” and the pitch and duration of these words may lead to the confusion of the model. Here we can say that the spectrogram reflex the “acoustic characteristics” of the sound.

Fig 7. Normalized confusion matrix of the model

Prediction with new recordings

The test showed that the model predicted well with the classes with high accuracy in training, but there are still confusions, for example with the class with “o” vowel in the sound. Below are two bar plots that showed the probability of the predictions of the model with different classes. On the left, we see that the model prediction was right, however, the chance that the model predicts “go” and “stop” are still high. Analyzing 2 files, we saw that the first sound might have made a wrong decision due to the missing of the ending sound “n”. The second “down”, is clearly pronounced, therefore, the spectrogram predicted correctly with a high confidence.

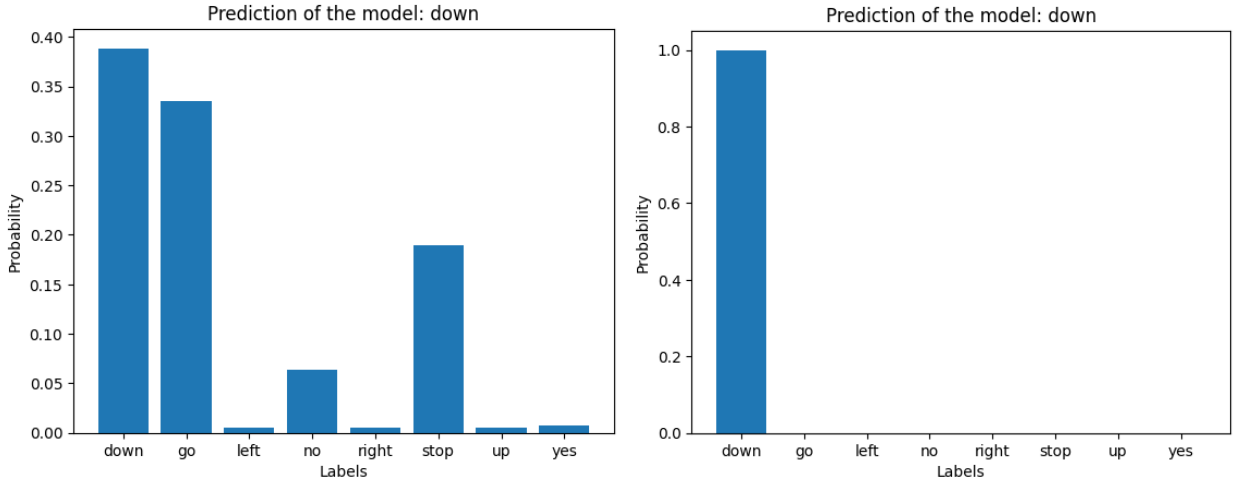

Fig 8. Predictions for the “down” label with a risk of confusion (left) and with high confidence(right)

Prediction with silence – an unassigned class. Since the model did not train with this class, the predictions seem uncertain.

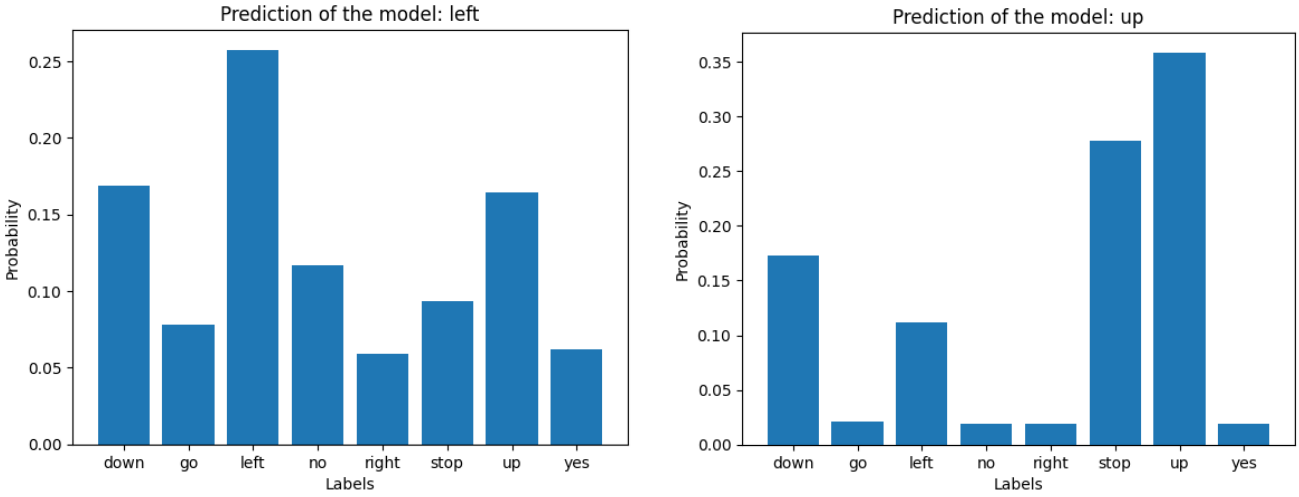

Fig 8. Uncertain predictions with silence

V. Conclusion

Spectrogram is a promising solution for the problem of voice recognition problem due to the reflection of the acoustic characteristics of the sound. A simple CNN model with Attention mechanism was created to run in a CPU with a fast-training time can reach the accuracy of 88.7%. The preprocessing of spectrogram before training will affect the result of the voice recognition model. Normalized mel spectrogram preprocessing is the best processing method in this case. The method showed a higher time of each training epoch, but reduce the number of epochs, then did not slowed down the speed of training. The deep learning model confused between labels with the same acoustic characteristics such as vowels, ending sounds, pitch, and durations. This suggests that spectrograms can be understood as the images of the sounds.

VI. References

[1] K. Choi, G. Fazekas, "A comparison of audio signal preprocessing methods for deep neural networks on music tagging", 2018 26th European Signal Processing Conference.

[2] Leland Roberts, Medium, 2020,, Understanding the Mel Spectrogram.

[3] K. W. Cheuk, K. Agres and D. Herremans, "The Impact of Audio Input Representations on Neural Network based Music Transcription," 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020, pp. 1-6, doi: 10.1109/IJCNN48605.2020.9207605.