Homework 1 report

Alina Burykina

March 25, 2025

Contents

1. Task 1

In the first part, I implemented a custom class that performs Mel-scale filterbank processing (LogMelFilterBanks) and compared it with the implementation from the torchaudio library.

The implementation uses the Short-time Fourier Transform (STFT) to convert the raw audio signal into the frequency domain, then processes it through the following steps:

1. Compute the frequency spectrum: `X_freq = STFT(x)`
2. Convert to magnitude/power spectrum: `X_freq = |X_freq|^power` (where power=1 gives magnitude, power=2 gives power spectrum)
3. Apply mel-scale filterbanks: `X_mel = X_freq.T @ H_m` (where H_m are the pre-computed mel filters)
4. Take the logarithm: `X_mel = log(X_mel + ε)` (with a small epsilon for numerical stability)

The key components of the implementation are:

- STFT computation with configurable parameters (window size, hop length, etc.)
- Mel filterbank initialization using triangular filters
- Proper handling of edge cases (frequency bounds, normalization)
- Batch processing support

For comparison, I used a randomly generated noise signal. The results showed nearly identical outputs between my implementation and torchaudio's reference implementation, validating the correctness of the approach.

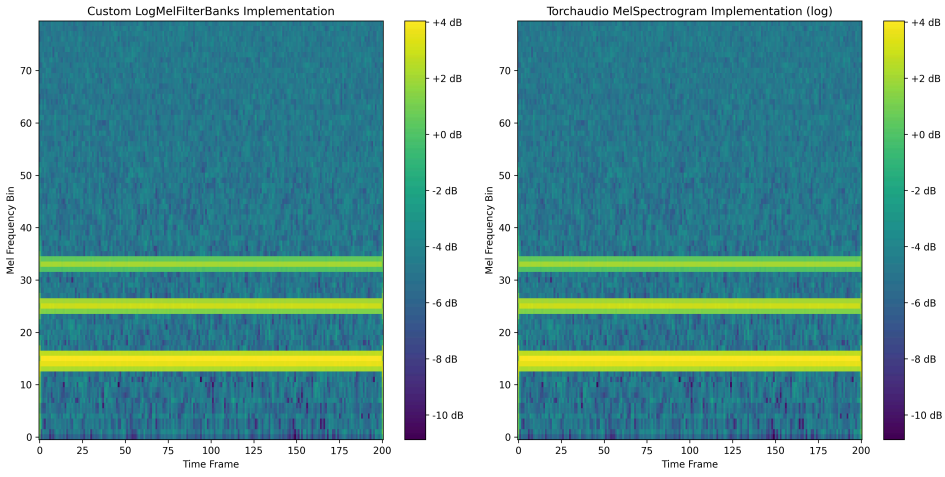


Figure 1: Comparison of my implementation and the torch one

2. Task 2

Binary Classification of Voice Commands

This study focuses on training a deep learning model for voice command classification, specifically distinguishing between "yes" and "no" commands. The experiments compare model performance across different hyperparameters:

Experimental Design

1. Mel-Scale Bands Experiment: Evaluated `n_mels` $\in [20, 40, 80]$
   - Impact on accuracy and computational efficiency
   - Analysis of FLOPs/parameter scaling
2. Convolution Groups Experiment: Evaluated `groups` $\in [1, 2, 4, 8]$ (with fixed `n_mels=40`)
   - Trade-offs between model complexity and performance

Metrics Tracked:
- Training loss
- Validation/test accuracy
- Parameter count
- FLOPs
- Epoch time

Why I Used a 2D CNN Instead of 1D

I chose a 2D CNN for this voice command task because:

1. Spectrograms are 2D – My input is a Mel-spectrogram (time vs. frequency), and 2D convolutions naturally capture patterns in both dimensions. A 1D CNN would treat frequency bins like a flat sequence, losing useful structure.
2. Proven in audio – Most modern speech models (like VGGish) use 2D CNNs on spectrograms because they simply work better than 1D for this type of data.

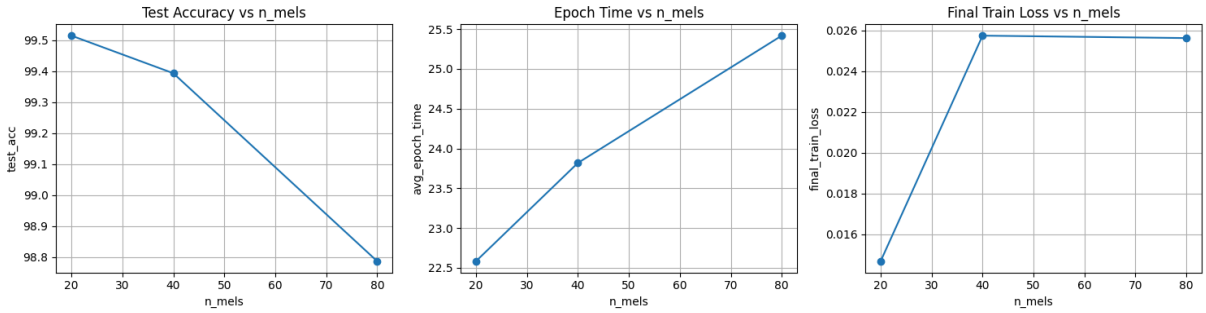
Yes, 1D CNNs can process raw audio directly, but they need bigger filters to match the performance of a 2D CNN on spectrograms. Since I was already converting to Mel-spectrograms anyway, 2D convolutions were the obvious choice.

Such architecture the model hit >99% accuracy — so the approach worked.

Experiment 1: Mel-Scale Bands (n_mels)

Key Observations:
- Performance: Highest test accuracy achieved with `n_mels=80` (marginally better by 0.2% vs. alternatives).
- Interpretation: Lower-dimensional representations may suffice for this binary task, as evidenced by comparable training loss curves across configurations.

Result summary



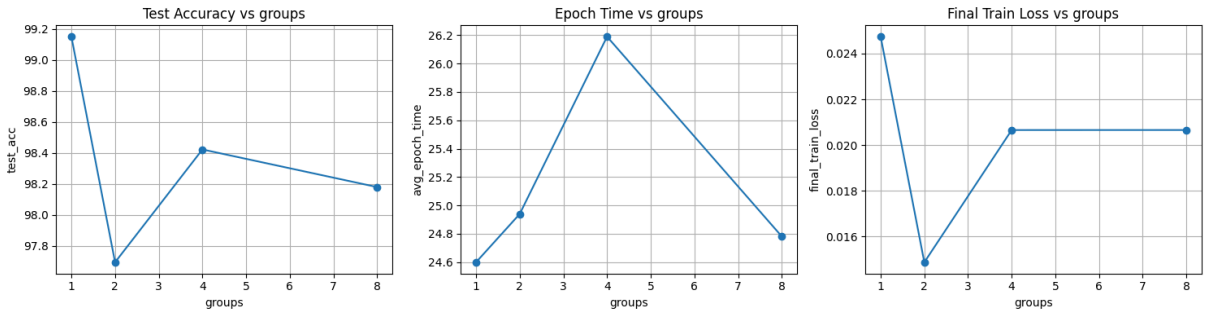
	n_mels	test_acc	avg_epoch_time	best_val_acc	best_train_acc	final_train_loss
0	20	99.514563	22.583363	99.128269	99.528154	0.014689
1	40	99.393204	23.818317	98.505604	99.087763	0.025739
2	80	98.786408	25.417807	98.132005	99.182133	0.025620

Experiment 2: Convolution Groups

Key Observations:

- Efficiency: Model size and FLOPs decrease nonlinearly (exponentially) with higher group counts.
- Performance Peak: Groups=1 and 4 yield optimal test accuracy, while groups=8 causes significant quality degradation.
- Divergence: Despite better test metrics for groups 4 over 8, groups=8 achieves same train loss, possibly due to late-training instability (observed in loss curves).

Results Summary:

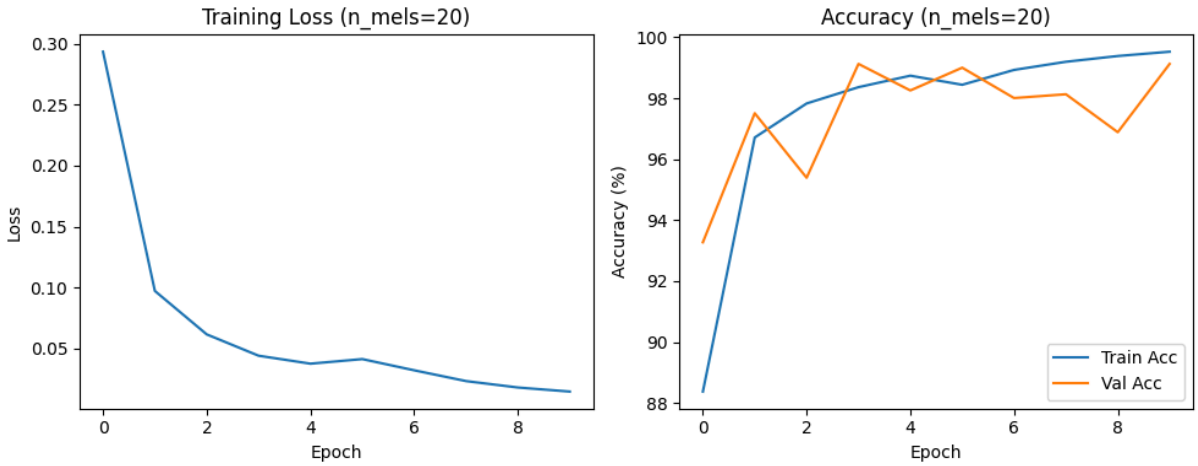


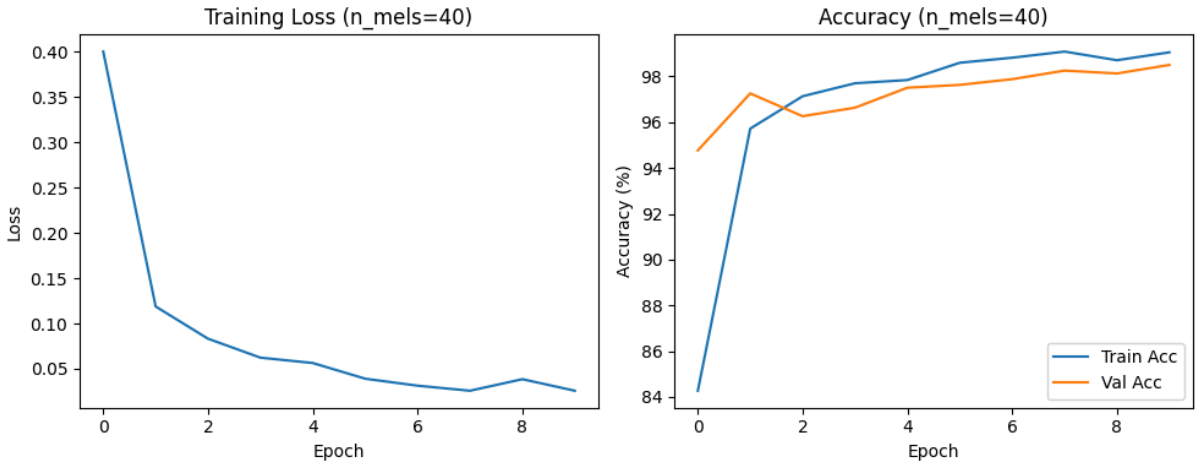
	groups	test_acc	avg_epoch_time	best_val_acc	best_train_acc	final_train_loss	total_params	flops
0	1	99.150485	24.598954	98.505604	99.166405	0.024740	486722	973444
1	2	97.694175	24.939315	98.381071	99.496697	0.014881	440642	881284
2	4	98.422330	26.189857	98.381071	99.323687	0.020659	417602	835204
3	8	98.179612	24.781432	97.633873	99.339415	0.020660	406082	812164

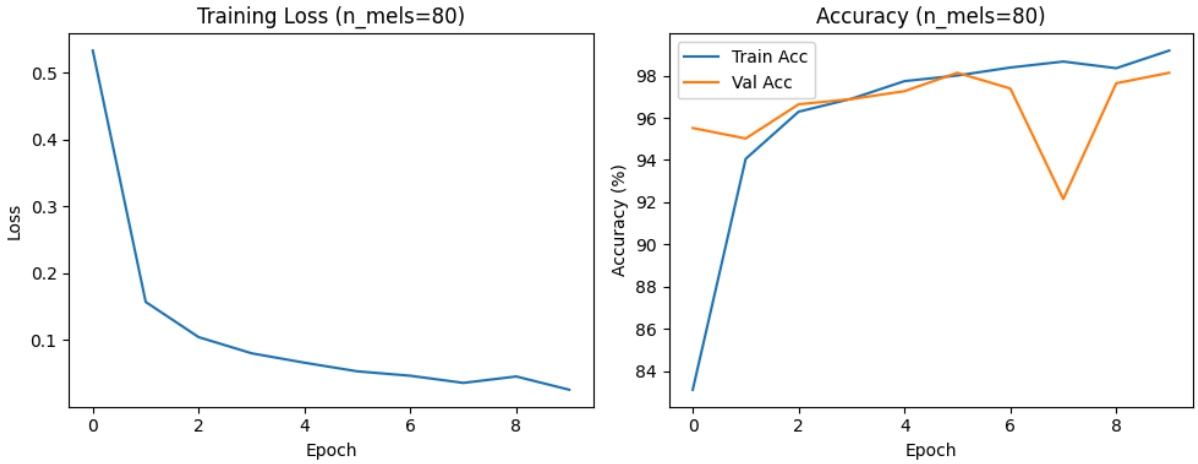
Implications:

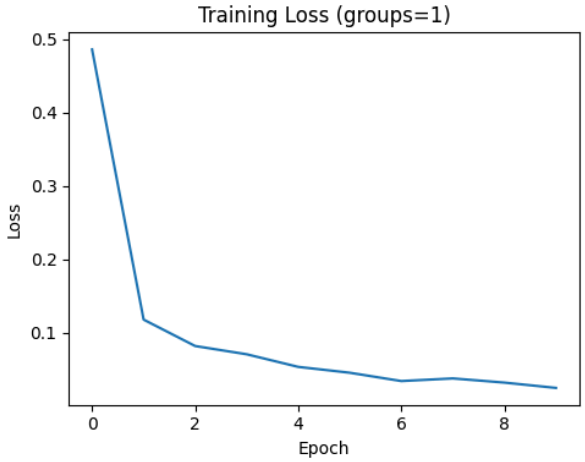
- Groups=1 offer a favorable accuracy/efficiency balance.

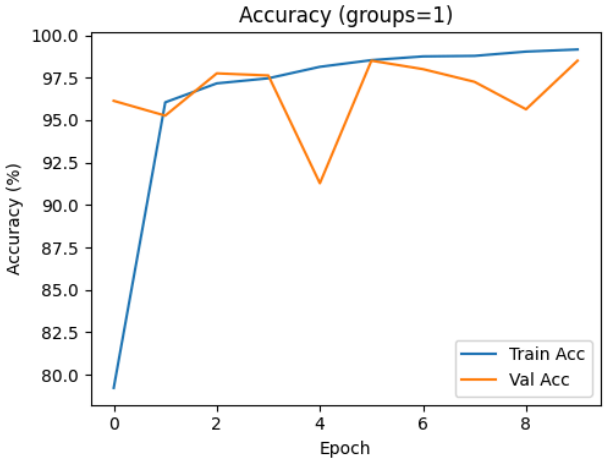
3. Additional materials

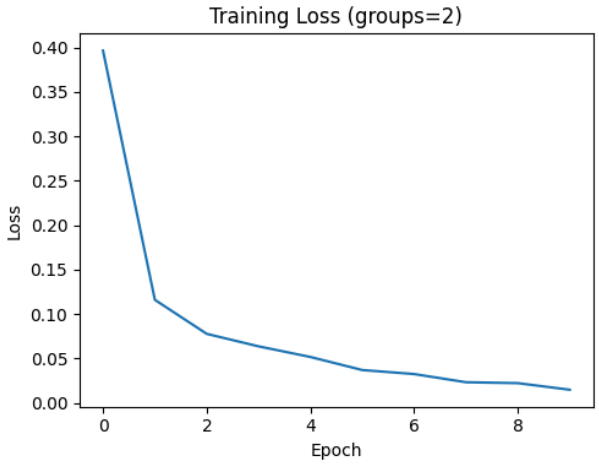


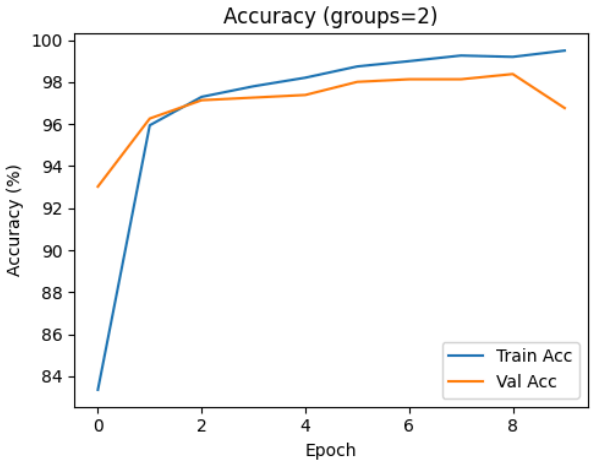


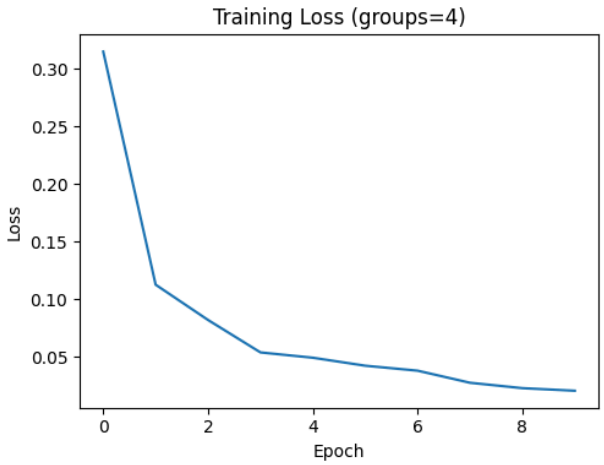


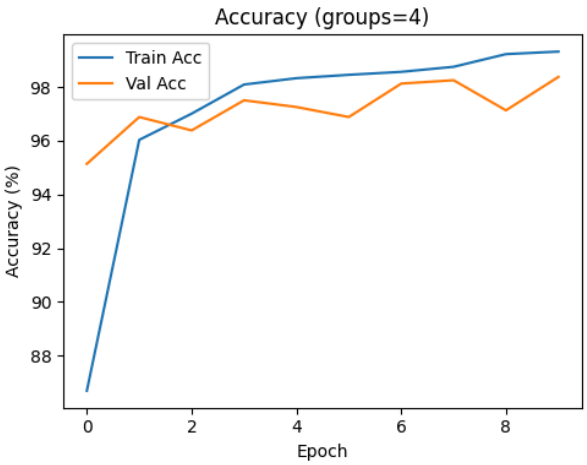

Training Loss (groups=1)
Loss
Epoch

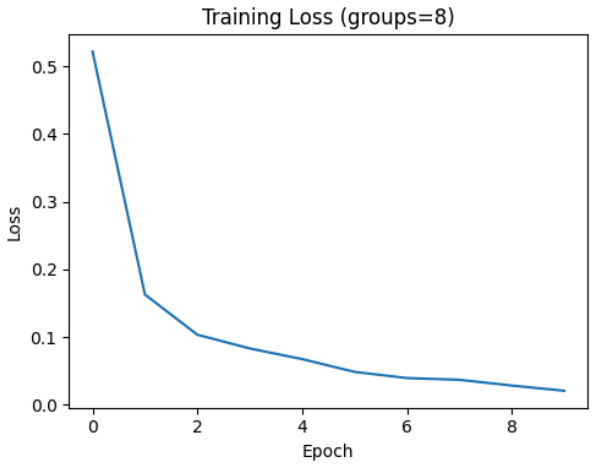

Accuracy (groups=1)
Accuracy (%)
Epoch
Train Acc
Val Acc

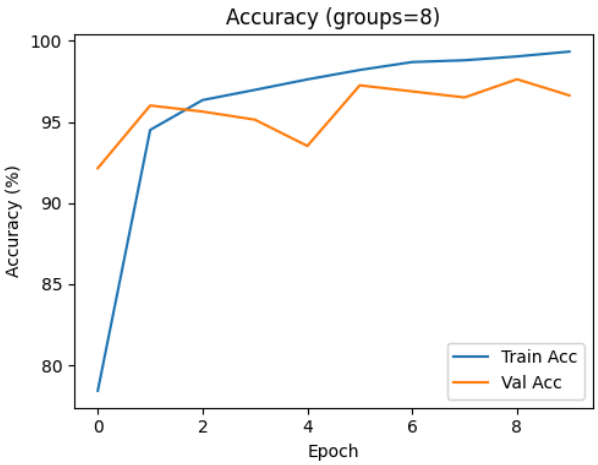

Training Loss (groups=2)
Loss
Epoch


Accuracy (groups=2)
Accuracy (%)
Epoch
Train Acc
Val Acc


Training Loss (groups=4)
Loss
Epoch


Accuracy (groups=4)
Accuracy (%)
Epoch
Train Acc
Val Acc


Training Loss (groups=8)
Loss
Epoch
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8


Accuracy (groups=8)
Accuracy (%)
Epoch
Train Acc
Val Acc
80
85
90
95
100
0
2
4
6
8