Digital Signal Processing Report

Task 2. Audio CNN

Baseline model Architecture

As the classification is binary and classes “yes” and “no” seem not complicated, a simple architecture was used. The model consists of:

- Two convolutional layers with normalization
- Max pooling
- Fully connected layer and Softmax

```
LogMelCNN(
  (conv1): Conv1d(20, 32, kernel_size=(3,), stride=(1,))
  (conv2): Conv1d(32, 16, kernel_size=(3,), stride=(1,))
  (batchnorm): BatchNorm1d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=384, out_features=2, bias=True)
  (fc2): Softmax(dim=1)
```

The task proved to be easy for the model as I was able to receive high quality on both validation and test without regularization methods (like dropout).

Experiment setup

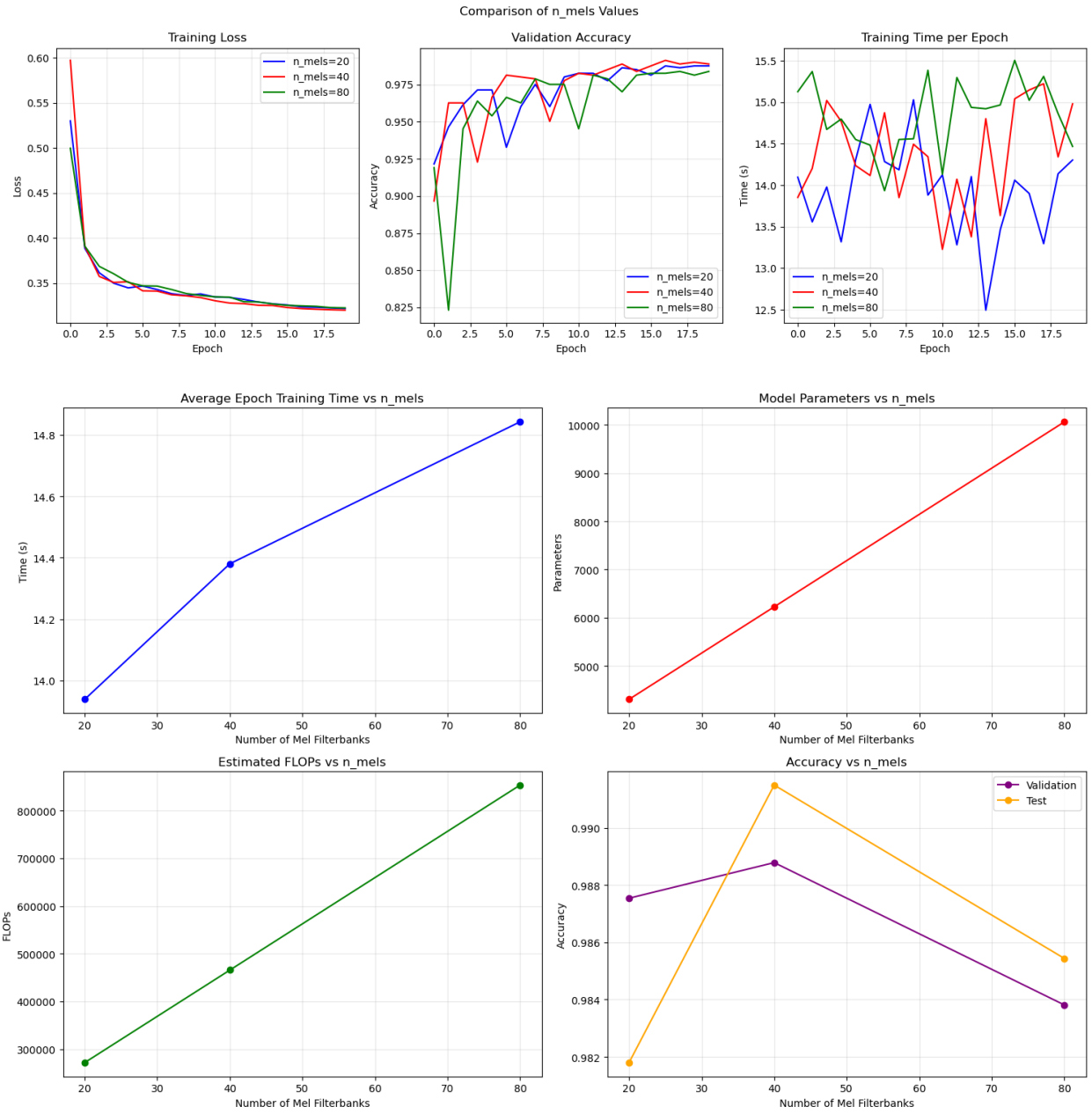
The experiments were conducted through experiment pipeline that:

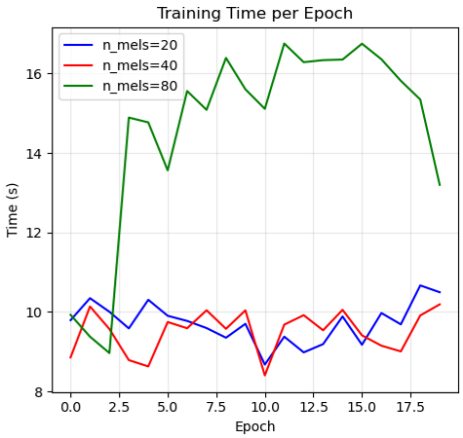
- Trains models with predefined set of *n_mels* parameter / *groups* parameter (20 epochs, OneCycleLR scheduler)
- Calculates validation and test accuracy on separate datasets
- Calculates model parameters and FLOPS
- Plots quality metrics and training logs

Experiment: Mel-Filterbanks

```
Summary of Results:
n_mels     Train Time      Val Acc         Test Acc        Params          FLOPs
--------------------------------------------------------------------------------
20         13.94           0.9875          0.9818          4,306           271,595
40         14.38           0.9888          0.9915          6,226           465,515
80         14.84           0.9838          0.9854          10,066          853,355
```

We can see that on validation *n_mels=40* shows best accuracy both on test and validation. Arguably, the difference in accuracy is very small and even negligeable. We can see that train time, as well as number of parameters and FLOPs increase proportionally to the increase in the number of mel-filterbamks. You can see this dependency on plots below. Thus, as the baseline for further experiment, *n_mels=40* will be used due to its optimal train time and number of parameters.





Interestingly, while running the same experiments the first time, I received far less training time for *n_mels=80*. Before, average train time used to be around 16. I had to rerun experiments and then all experiments more or less coincided in training time.

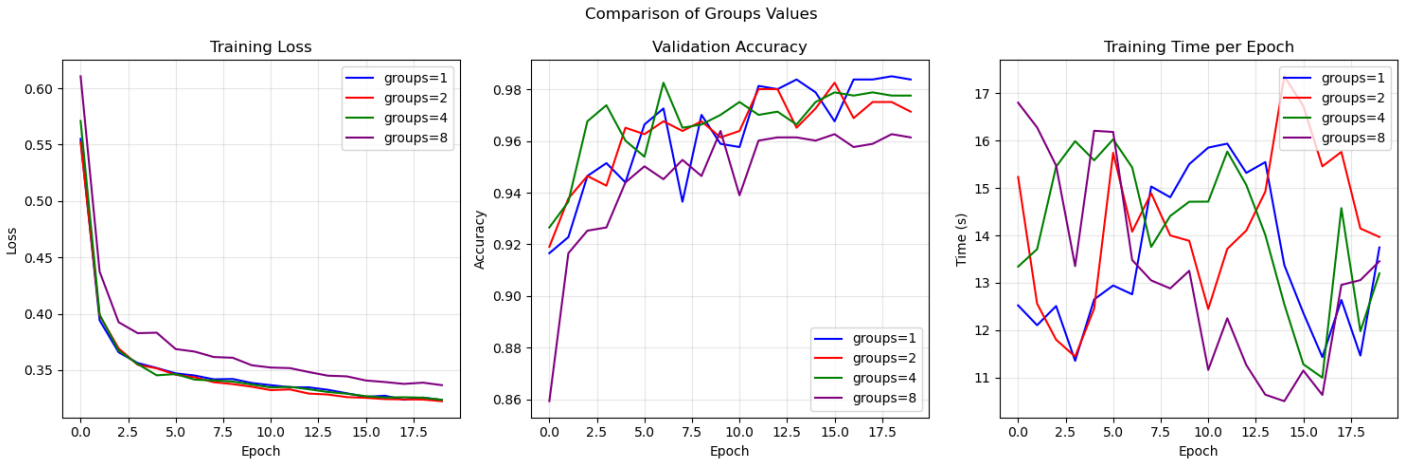
On the plots below, we can see clear dependency between the number of mel-filterbanks, training time, model parameters, and FLOPs

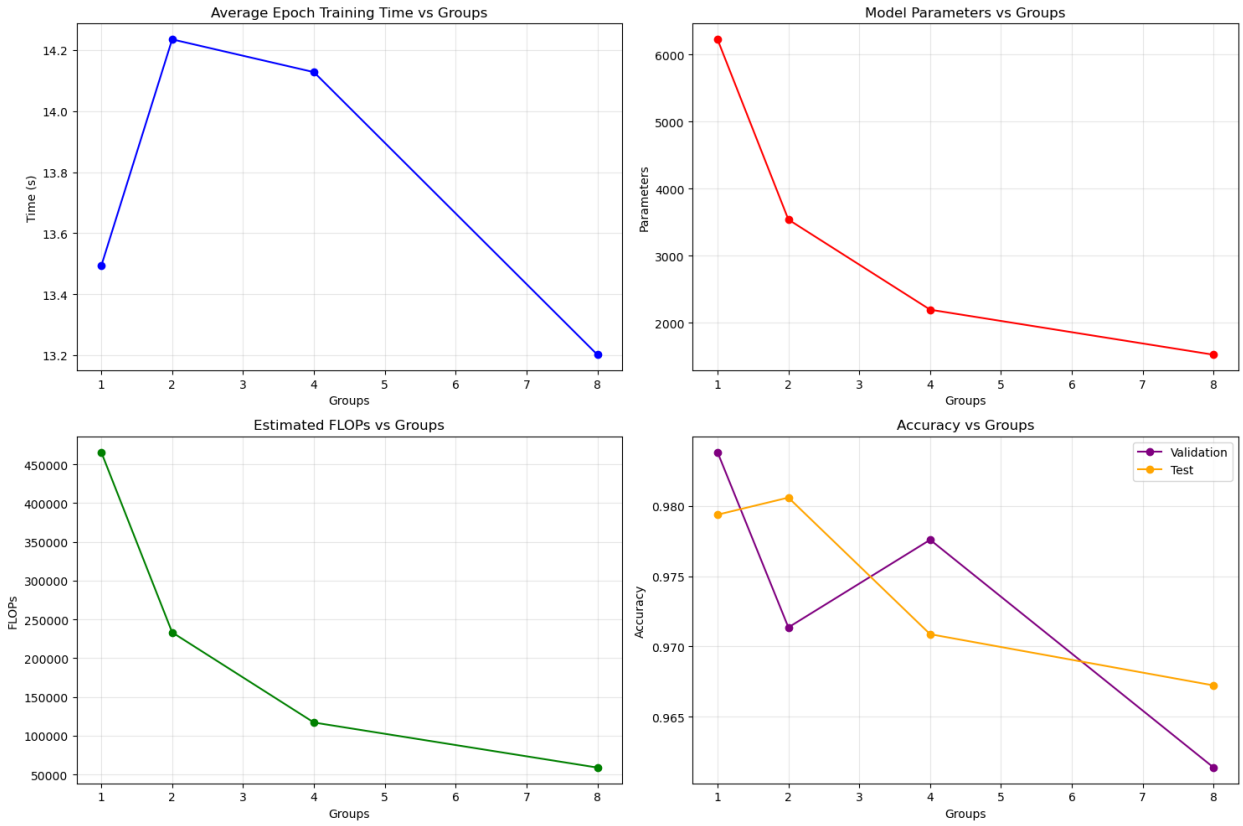
Experiment: groups parameter of Conv1d

Summary of Results:

Groups	Train Time	Val Acc	Test Acc	Params	FLOPs
1	13.49	0.9838	0.9794	6,226	465,515
2	14.24	0.9714	0.9806	3,538	233,195
4	14.13	0.9776	0.9709	2,194	117,035
8	13.20	0.9614	0.9672	1,522	58,955

With groups parameter, the dependency is reverse. We can see a decrease in both number of parameters and FLOPs with the increase of groups. On test *groups=2* seems the most robust and shows higher accuracy than on validation.





Conclusion

The optimal combination of tested hyperparameters ended up being *n_mels=40* and *groups=2.* Larger *groups*, though has less parameters, perhaps needs more epochs to reach the same level of quality. As for n_mels, smaller number of features is enough for model to distinguish classes in not complicated ml tasks. Increasing dimensionality does not only not give sufficient boost in quality, but also increases the computational power expenditure and increases risk of overfitting.