## Summary

## Preprocessing

First, the data was split into probes. This was done with two separate algorithms which are compared: a simple algorithm that smooths local regions and checks if the mean voltage is low, and one that looks at the labels to cut out probes from non-NP regions.

We run a train-test split with 49 train files and 13 test files that are randomly selected.

## Model

Then, the UNet with a LR of 5e-4. Each EncoderBlock has 2x (Conv1D, Batchnorm, ReLu). Each layer of the Unet contains one EncoderBlock then a MaxPool1D. The bottleneck is a single EncoderBlock. The DecoderBlock is the same as the EncoderBlock, except it has some additional logic to fix shapes from non-power of 2 inputs by cropping/padding.

Note that the hyperparameters selected are not necessarily optimal and I did not spend much effort on this yet. I suspect with some small optimization changes such as better parameter searches and tools like warmup etc. we can squeeze some extra performance out of these models.

## Results

### Voltage Only

#### Simple Preprocess

We have an accuracy of 0.8198 and a macro-averaged F1 of 0.3906

Class K: Precision = 0.28, Recall = 0.40, F1 Score = 0.33

Class W: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

Class M: Precision = 0.94, Recall = 0.87, F1 Score = 0.90

Class N: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

Class J: Precision = 0.24, Recall = 0.25, F1 Score = 0.24

Class L: Precision = 0.76, Recall = 0.92, F1 Score = 0.83

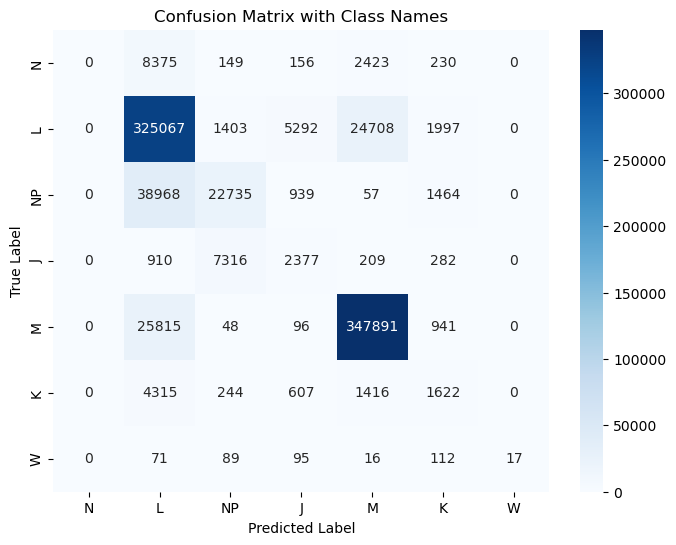
Class NP: Precision = 0.71, Recall = 0.30, F1 Score = 0.43

Overall (Macro) Statistics:

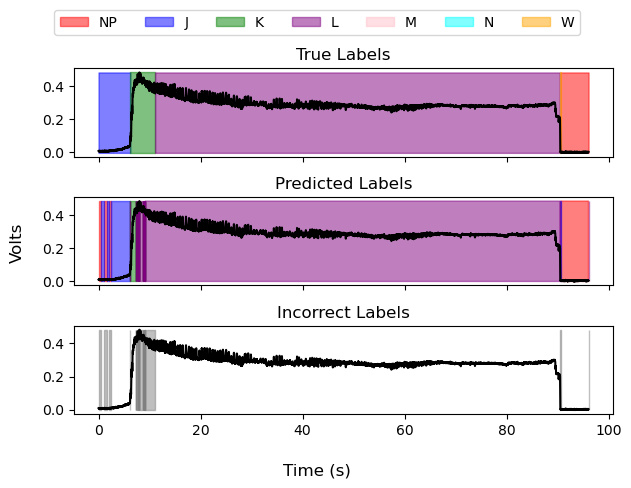
Precision = 0.42, Recall = 0.39, F1 Score = 0.39

We do very well on L and M. We do very poorly on W, and need work on J and K (which have the least data).

A confusion matrix for its performance across labels is below.

****

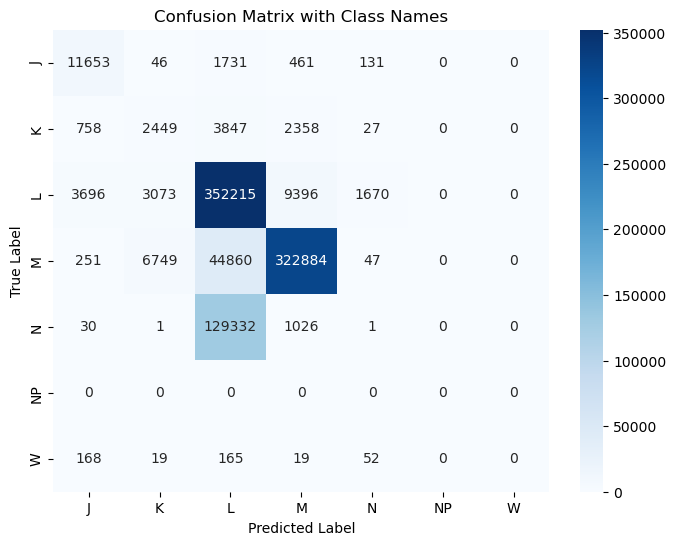
Sometimes it looks good, although even in these examples there is still plenty of barcoding present. This is a good candidate where we do a good job with the macro trends, and it looks like our J and K are about right with some barcoding present.



The probe-detecting algorithm has some issues, especially with weird probes where some things go wrong. The particular file, where there was likely an issue with the physical bug, might require the research to do something like manually select probes.

#### Date Leak Preprocess

Epoch 70/70, Train Loss: 0.3348, Acc: 0.7665, F1: 0.4460



Class K: Precision = 0.20, Recall = 0.26, F1 Score = 0.22

Class W: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

Class M: Precision = 0.96, Recall = 0.86, F1 Score = 0.91

Class N: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

Class J: Precision = 0.70, Recall = 0.83, F1 Score = 0.76

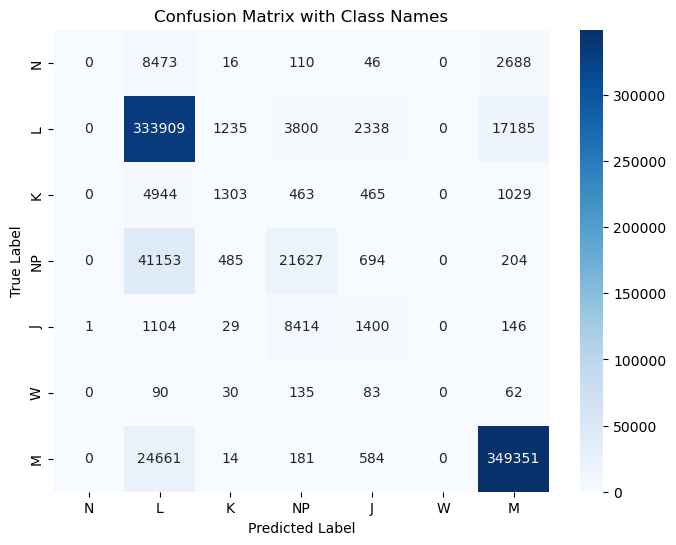
Class L: Precision = 0.66, Recall = 0.95, F1 Score = 0.78

Overall (Macro) Statistics:

Precision = 0.42, Recall = 0.48, F1 Score = 0.45

### Voltage + ClaSPy

#### Simple Preprocess



Class N: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

Class L: Precision = 0.81, Recall = 0.93, F1 Score = 0.86

Class K: Precision = 0.42, Recall = 0.16, F1 Score = 0.23

Class NP: Precision = 0.62, Recall = 0.34, F1 Score = 0.44

Class J: Precision = 0.25, Recall = 0.13, F1 Score = 0.17

Class W: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

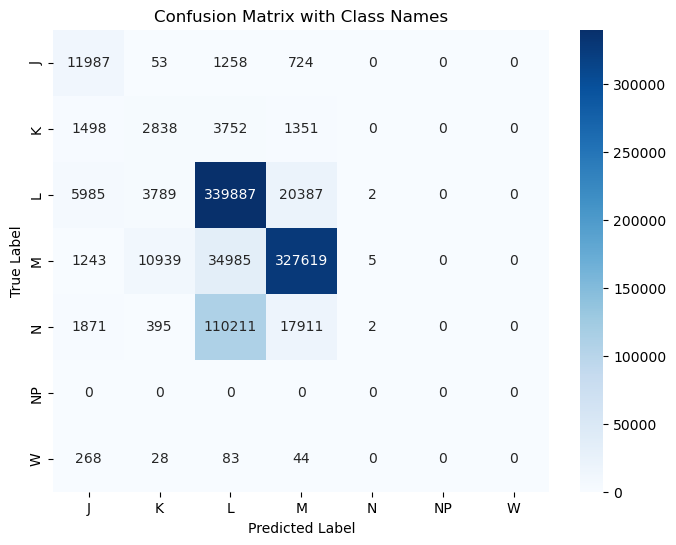
Class M: Precision = 0.94, Recall = 0.93, F1 Score = 0.94

Overall (Macro) Statistics:

Precision = 0.43, Recall = 0.36, F1 Score = 0.38

#### Data Leak Preprocess

Epoch 50/50, Train Loss: 0.2289, Acc: 0.7589, F1: 0.4215



Class K: Precision = 0.16, Recall = 0.30, F1 Score = 0.21

Class W: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

Class M: Precision = 0.89, Recall = 0.87, F1 Score = 0.88

Class N: Precision = 0.22, Recall = 0.00, F1 Score = 0.00

Class J: Precision = 0.52, Recall = 0.85, F1 Score = 0.65

Class L: Precision = 0.69, Recall = 0.92, F1 Score = 0.79

Overall (Macro) Statistics:

Precision = 0.41, Recall = 0.49, F1 Score = 0.42

## Next Steps

* For the probe splitting algorithm, just leak NP data to include no NPs in each probe
* Only train the model on valid probes
* Manual and CRF-based post-processing
* Do hyperparameter search and other optimization

## Conclusions

This model performs well in accuracy space (although note that such a comparison is kind of meaningless since we use different data) and F1 space. The outputs look fairly reasonable, making this a very strong candidate model. It is also very extensible, such as adding input channels or adding a CRF on top of the logits.