Project statement

Each group member wrote their own code and CNN (parts were shared), and contributed to different parts of the development process. We have worked together throughout the process, discussed every step, and consider the combined workload to be equally divided.

Most divisible contributions:

Anne

- Data and ontology inspection as well as a preprocessing pipeline for the multilabel model.
- Reworked code to run in Google Colab and running final models
- Wrote 4 layer CNN and ran final testing

Liva

- Preprocessing pipeline as well as construction, training and evaluating the single label (6 families) CNN model.
- Training/running/evaluating the pretrained model PANNS CNN6 for single label classification (6 families)
- The sound guesser code that uses the best model made by each group member to record and predicts labels from said recording.

Mikas

- Understanding metrics for evaluation, how they are calculated and why each of them are (not) worth using.
- Researching and testing what enhancement were worth looking into, in order to better our model (spectral augmentation, label weighting, focal loss, oversampling)
- Wrote 3 layer CNN and ran initial testing

CNN Classifier for Audio Recognition trained on the FSD50k Data Set

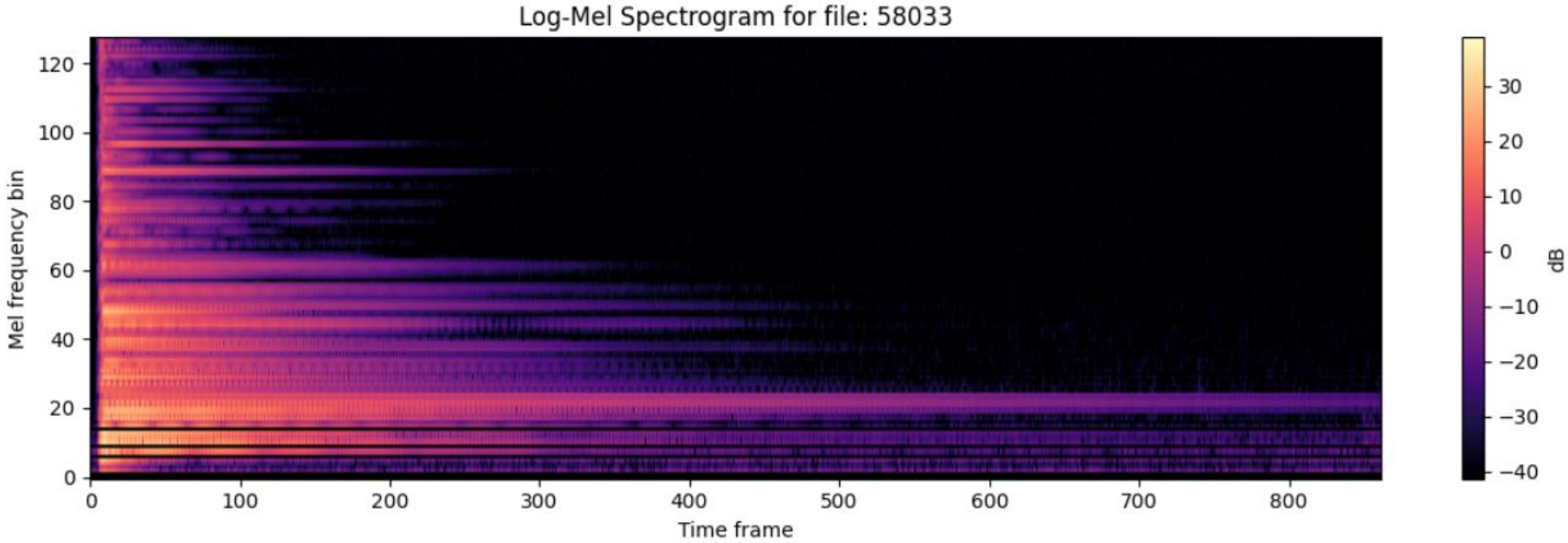
Anne Geday Overgaard (VPN272)
Liva Julie Smidt (RSH344)
Mikas Raith Møller (ZVN840)

Agenda

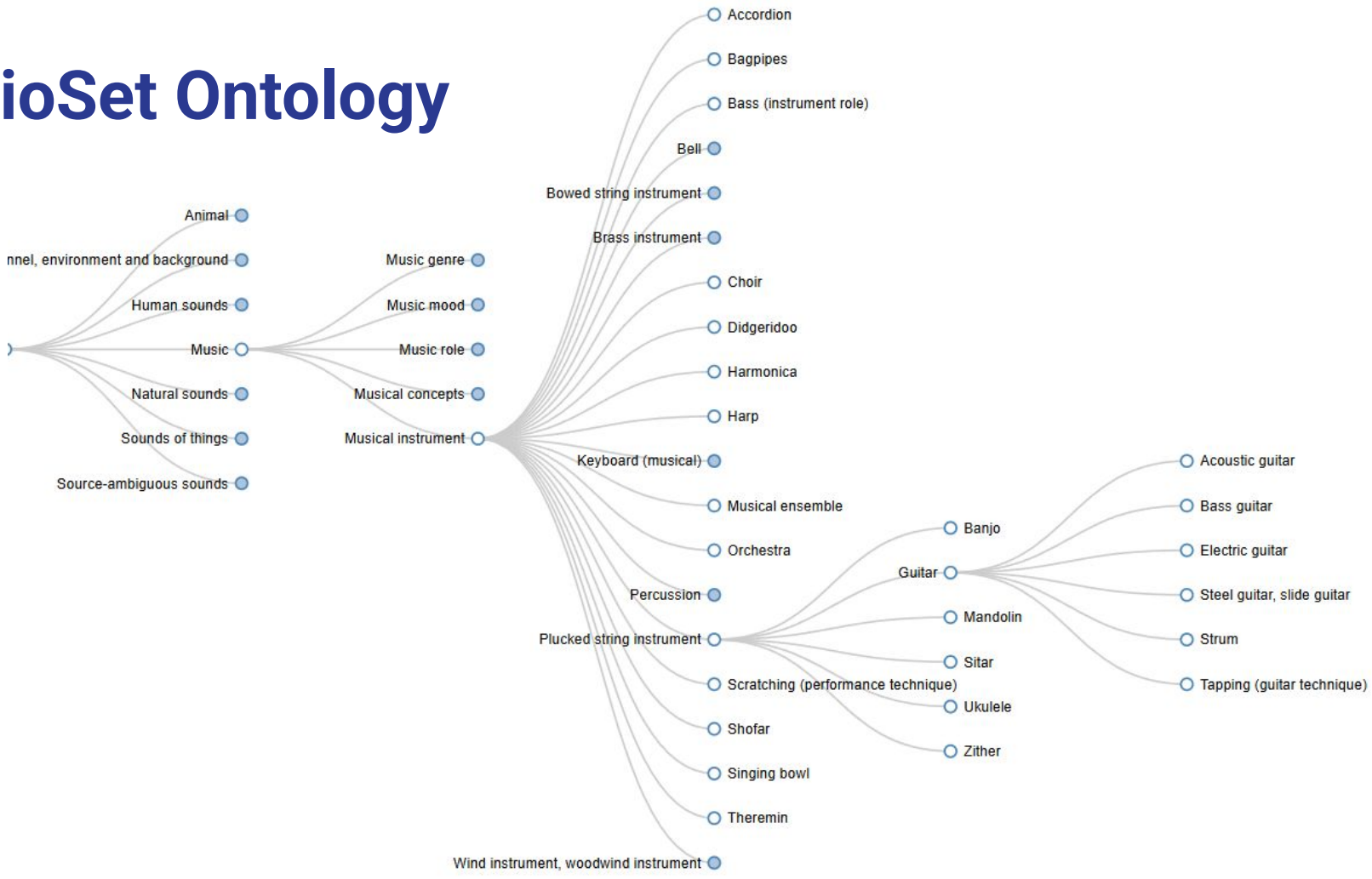
1. Looking at the data
2. Preprocessing
3. Model architectures
4. Single-label model
5. Multi-label model
6. Further work
7. Guess a noise!

Data example

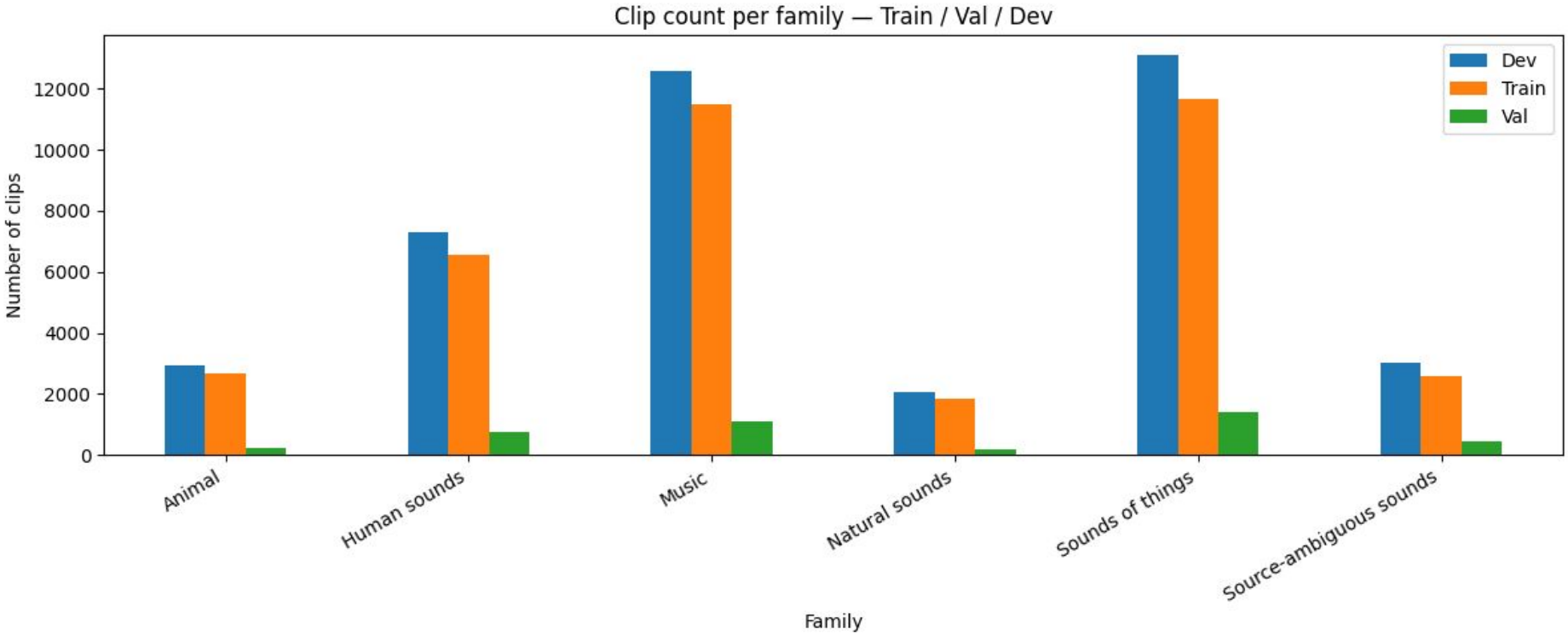
```
Filename: 58033
Active label names: ['Guitar', 'Music', 'Musical_instrument', 'Plucked_string_instrument', 'Strum']
```



AudioSet Ontology



Distribution of clips across families



Uneven sampling across labels

10 most occurring labels

label	# of occurences
Music	11610
Musical_instrument	11609
Domestic_sounds_and_home_sounds	4055
Human_voice	3665
Animal	2976
Percussion	2731
Wind_instrument_and_woodwind_instrument	2458
Vehicle	2325
Bowed_string_instrument	1841
Plucked_string_instrument	1687

10 least occurring labels

label	# of occurences
Speech_synthesizer	52
Conversation	50
Ratchet_and_pawl	49
Gasp	48
Chuckle_and_chortle	47
Glockenspiel	47
Accordion	46
Tabla	44
Typewriter	44
Tick	42

Preprocessing audio files and labels

For multi-label model

.wav Audio file

↓

Raw waveform

What size and resolution should the image be?

↓

Log-mel spectrogram

↓

Save locally as PyTorch tensors

What bit of sound is most relevant?

↓

When training: Load tensor, pad/tile/truncate, and normalize

Audio file name

↓

Look up in CSV file

↓

String of labels

↓

200 input tensor with 0 or 1 for each label

This is now our ‘Target’

Preprocessing audio files and labels

For single-label model

.wav Audio file

↓

Raw waveform

What size and resolution should the image be?

↓

Log-mel spectrogram

↓

Save ~~locally~~ as PyTorch tensors

What bit of sound is most relevant?

↓

When training: ~~Load tensor,~~ pad/tile/truncate, and normalize

Audio file name

↓

Look up in CSV file

↓

String of labels

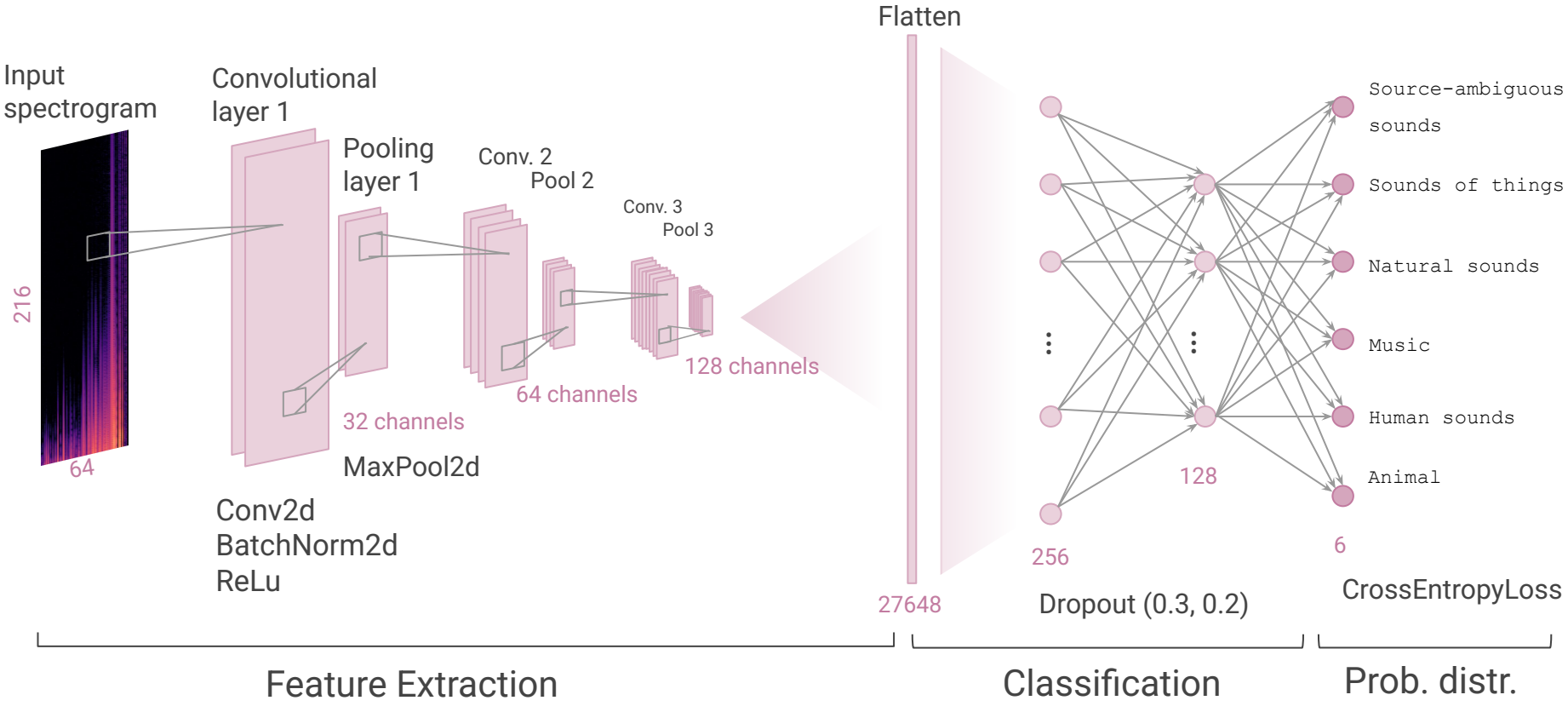
↓

`get_top_level_family(mid):`

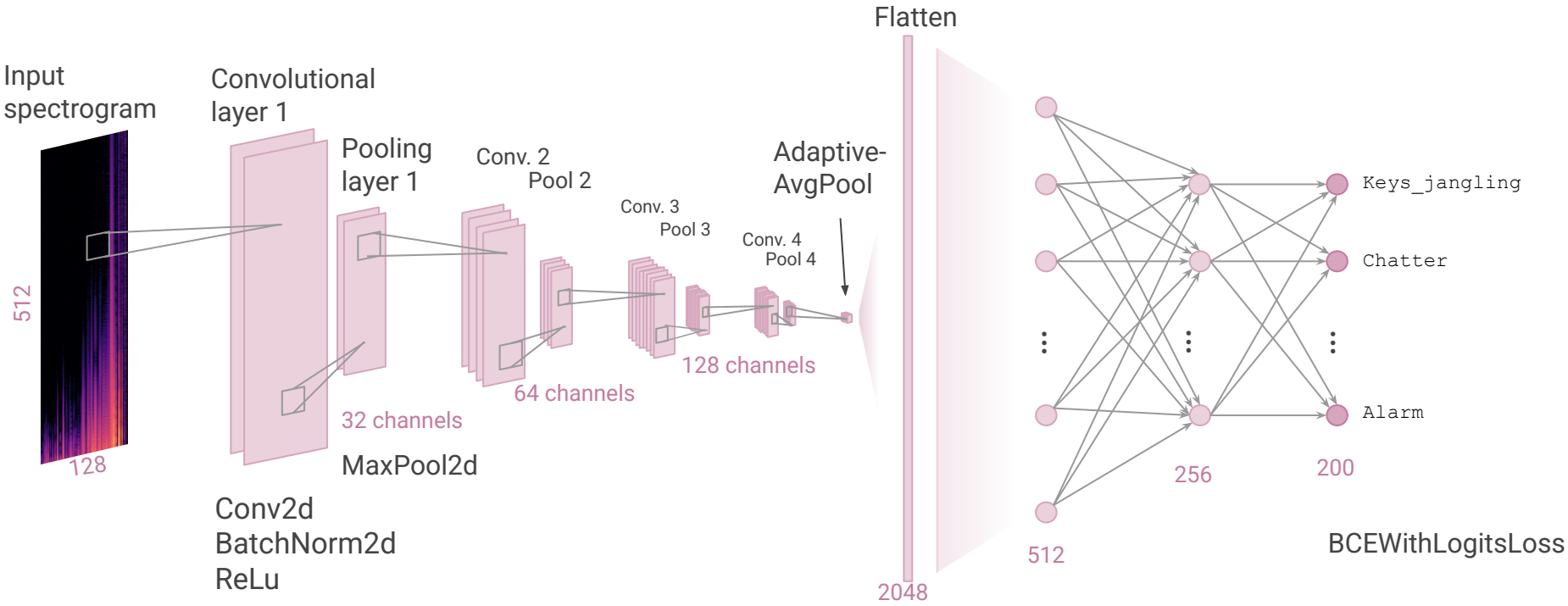
1 family for each label

This is now our 'Target'

Single-label CNN Architecture



Multi-label CNN Architecture



Model Enhancements

List of things we tested for improvements:

1. No help at all
2. Hyper parameter optimization
3. Different types of data augmentation:
    a. Spectral Augmentation
    b. Random cropping of clips
    c. Mixup augmentation
4. Different ways to account for uneven sampling:
    a. Weighted labels (Focal loss + initial bias or simple inverse frequency weights)
    b. Oversampling
    c. Combination of weighted labels and oversampling

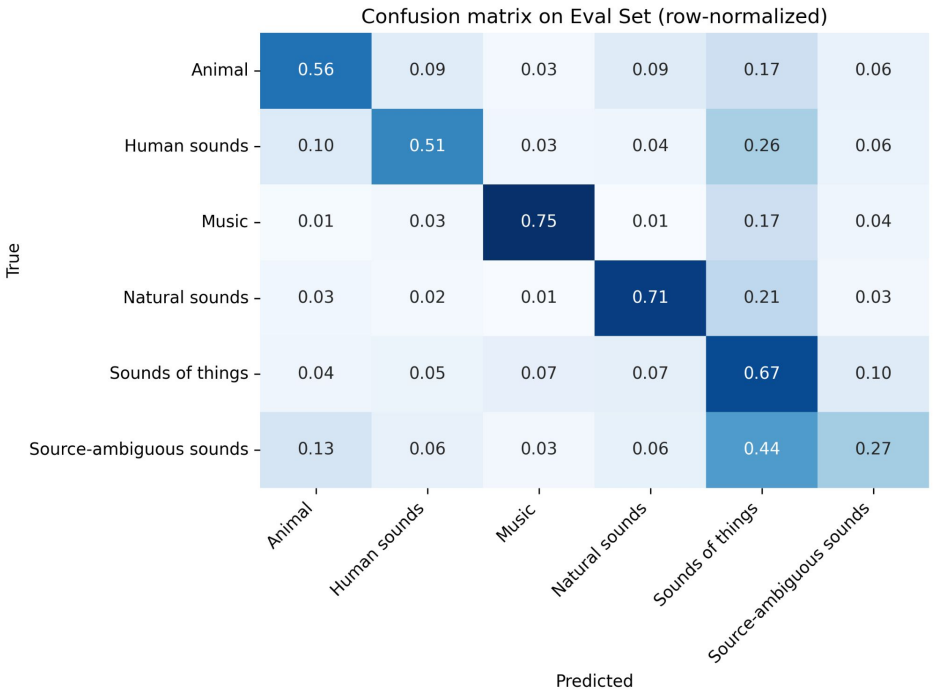
Metrics for evaluation

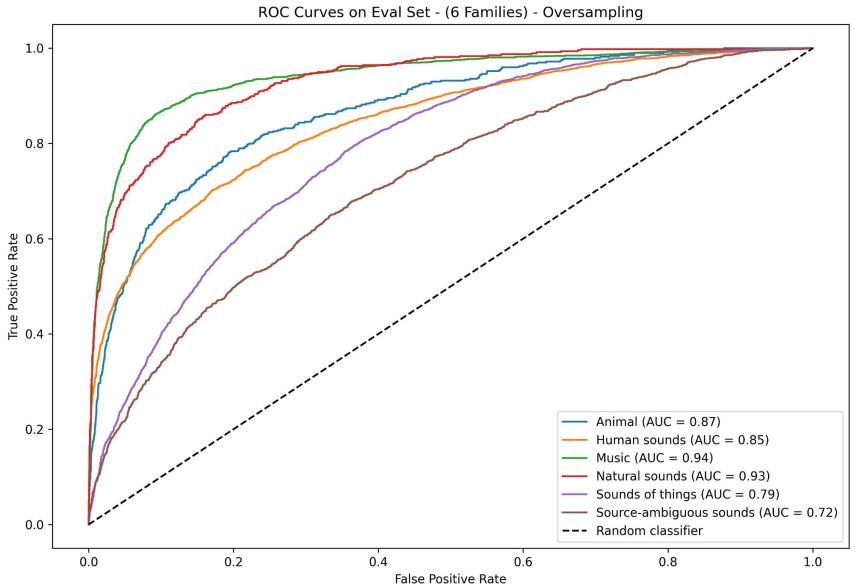
- **SKlearn accuracy score** [0,1]
- F1 score (micro/macro) and threshold tuning [0,1]
- **Mean average precision (mAP)** [0,1]
- d’ (d prime) [0,inf]
- Label-weighted label ranking average precision (lωlrap) [0,1]

How did our models fare at classifying?

CNN single-label

`Eval mAP: 0.577`

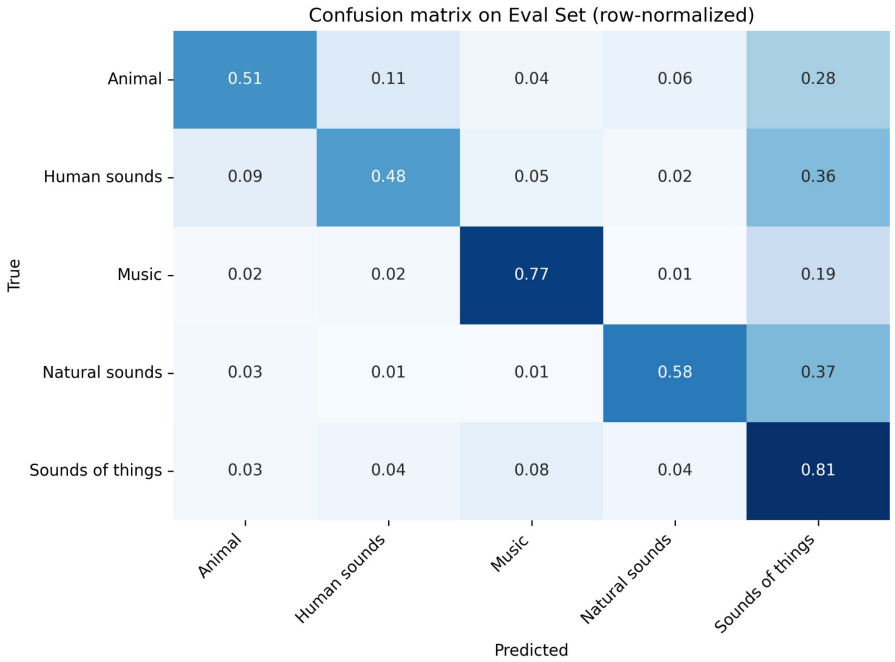


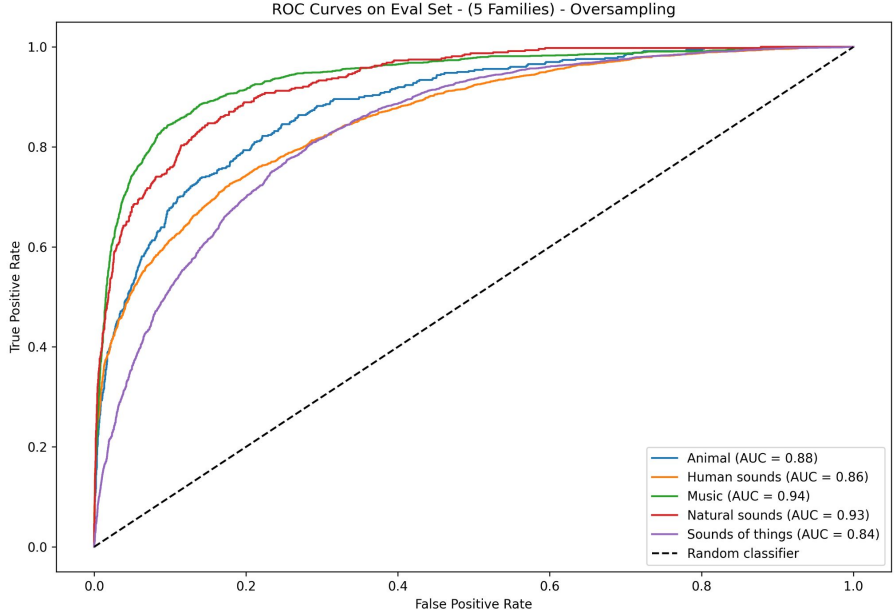


How did our m[illegible] fare at classifying?

NOW ONLY 5 FAMILIES!!!

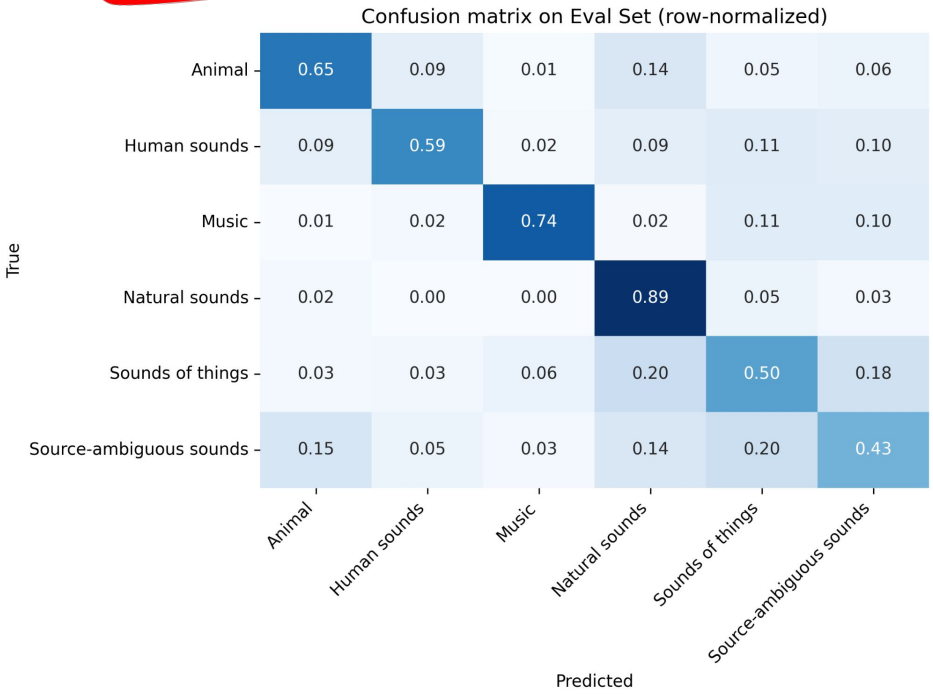
Eval mAP: 0.676

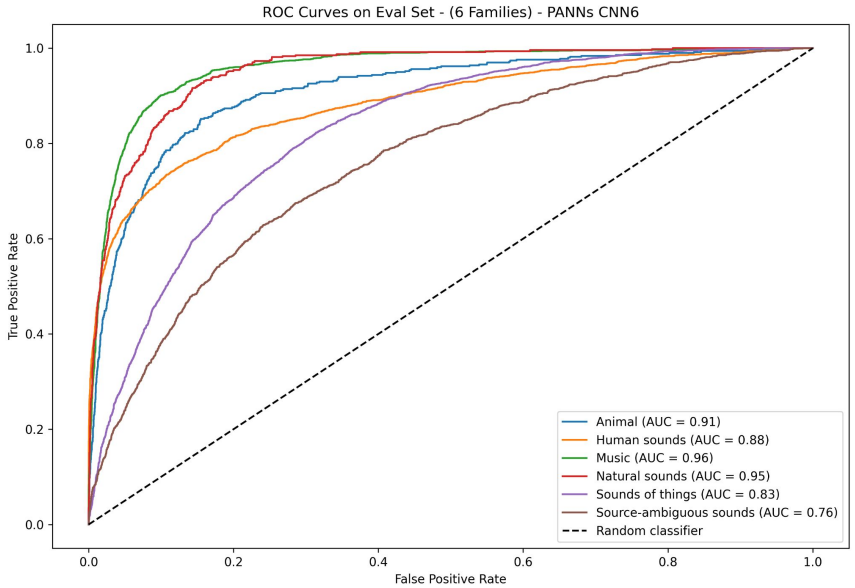




How did our models fare at classifying?

PANNs CNN6 single-label `Eval mAP: 0.624`





How did our models fare at classifying?

Best model!

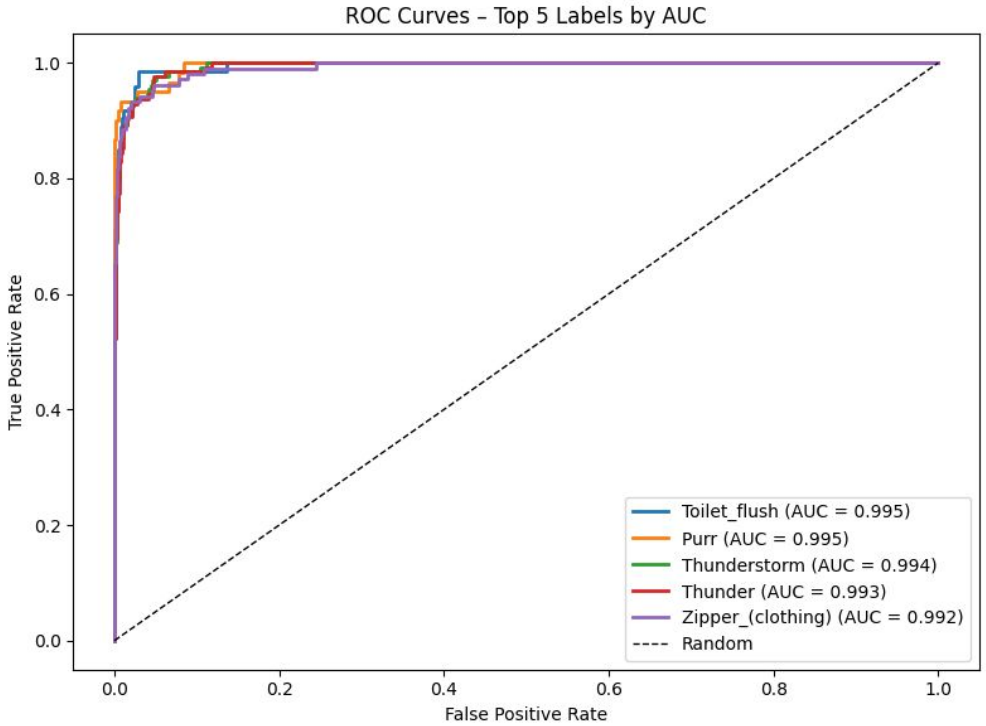
CNN multi-label

	Unhelped	Weighted + focal loss	Oversampling	Oversampling + spectral augmentation + mixup augmentation
mAP **(6 families mAP)**	0.443 (0.747)	0.440 (0.745)	0.439 (0.739)	0.470 (0.754)
d'	2.273	2.292	2.247	2.309
lwlrap	0.633	0.629	0.625	0.643

(All variations use random cropping)

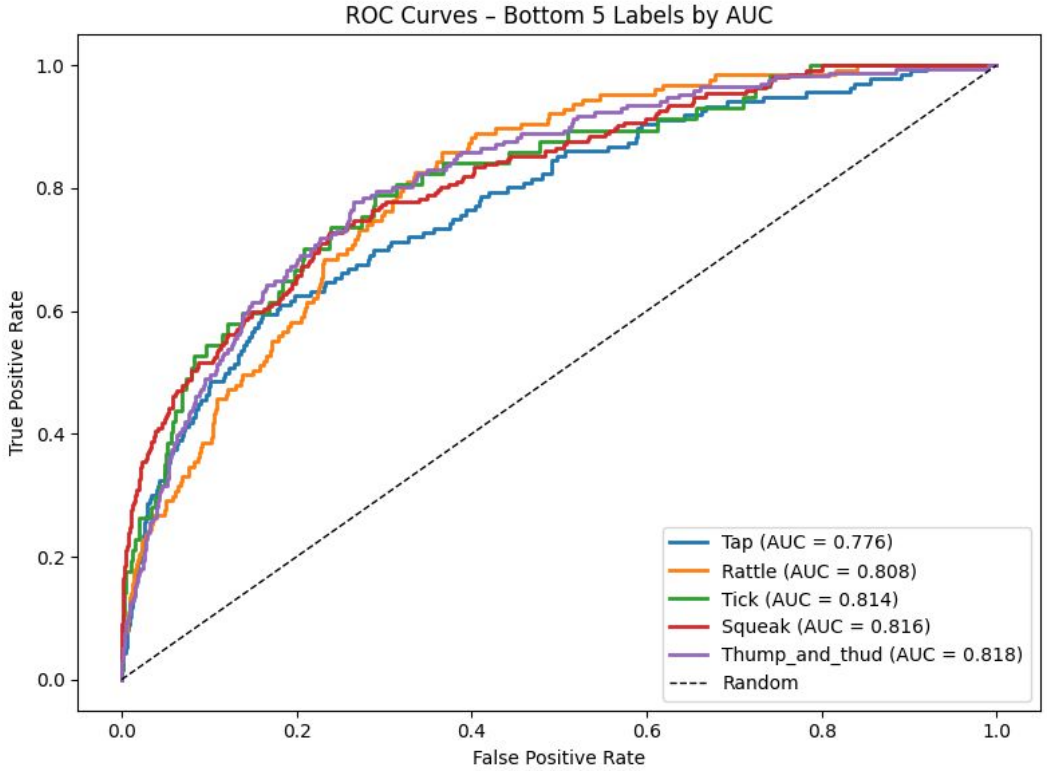
How did our models fare at classifying?

CNN multi-label (best version)



How did our models fare at classifying?

CNN multi-label (best version)

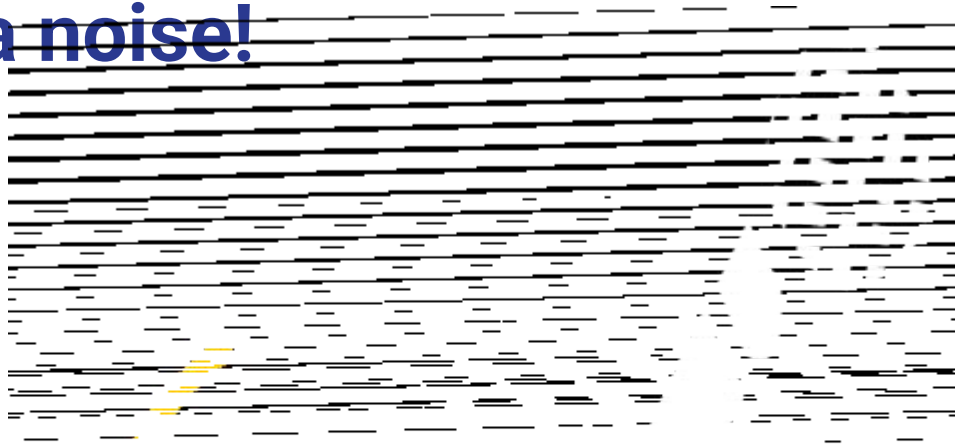


What else could we have done?

Other improvements:

- Test varying batch sizes in dataloader
- Test a different clip length
- Try a pretrained models for multi-label classification
- Pretrained on 5 families instead of 6
- Have better PC’s (more processing power or use Colab earlier than a few days before hand in).
- Try different types of algorithms (not CNNs)

Guess a ~~noise~~!



Appendix

Pytorch Classifier: model structure

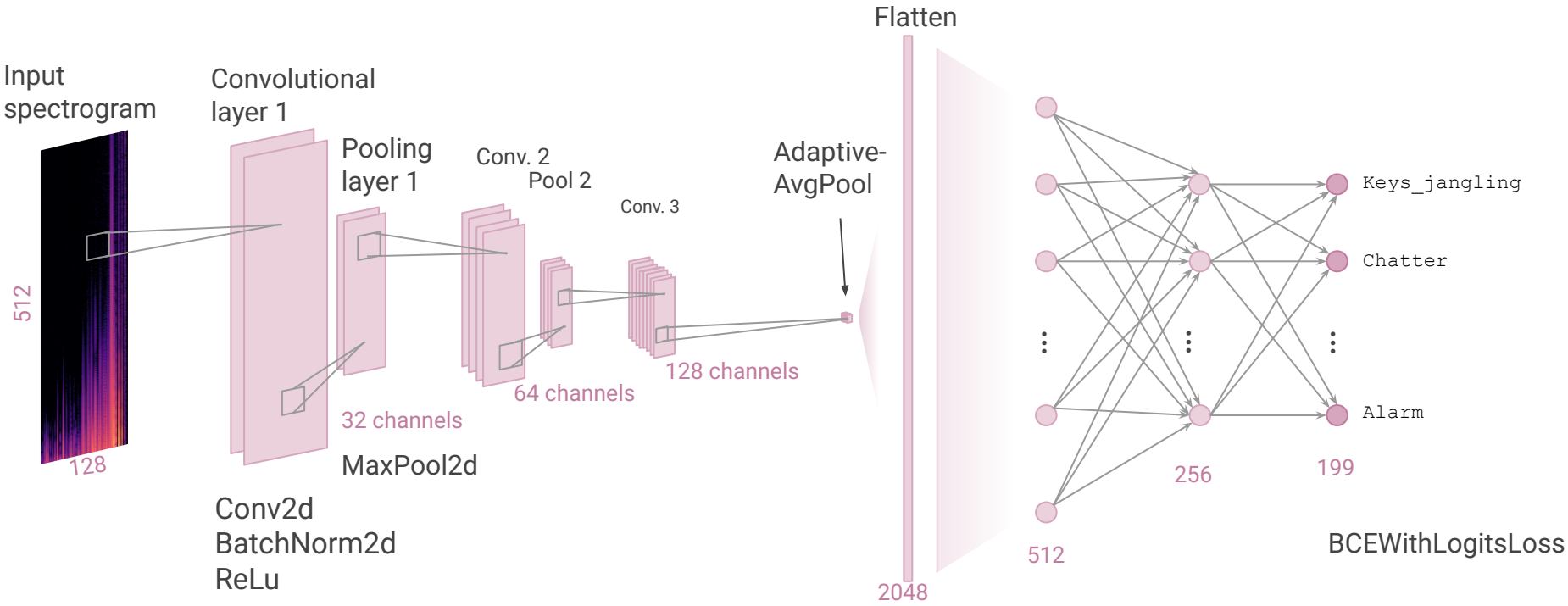
CNN - single-label

- One of six families (AudioSet Ontology)
- HP-optimization (Bayesian optimization with TPEsampler, MedianPruner)
- CrossEntropyLoss
- Our CNN model
  - 3 convolutional layers with:
    - Conv2d → Batchnorm → ReLu → MaxPool
  - Classifier head:
    - 2048 → 512 → 256 → 200
    - Dropout (0.2)
  - BCEWithLogitsLoss
  - Batch 32, 64 mel bins, 22050Hz sample rate, 5 second clips
    - 7 million + trainable params
    - Input shape: [32, 1, 128, 516]
    - Output shape: [32 , 6]
- A pre-trained model PANNs CNN6

CNN - multi-label

- 4 convolutional layers with:
  - Conv2d → Batchnorm → ReLu → MaxPool
- Classifier head:
  - 2048 → 512 → 256 → 200
  - Dropout (0.3, 0.2)
- Scheduler (Initial learning rate = 1e-3)
- BCEWithLogitsLoss
- Batch 32, 128 mel bins, 22050Hz sample rate, 6 second clips
  - 1,472,776 trainable params
  - Input shape: [32, 1, 128, 516]
  - Output shape: [32 , 200]

Abandoned Multi-label CNN Architecture



Abandoned CNN model (had worse results)

Spectrogram input for model:
(32, 1, 128, 512)
Hyperparameters:

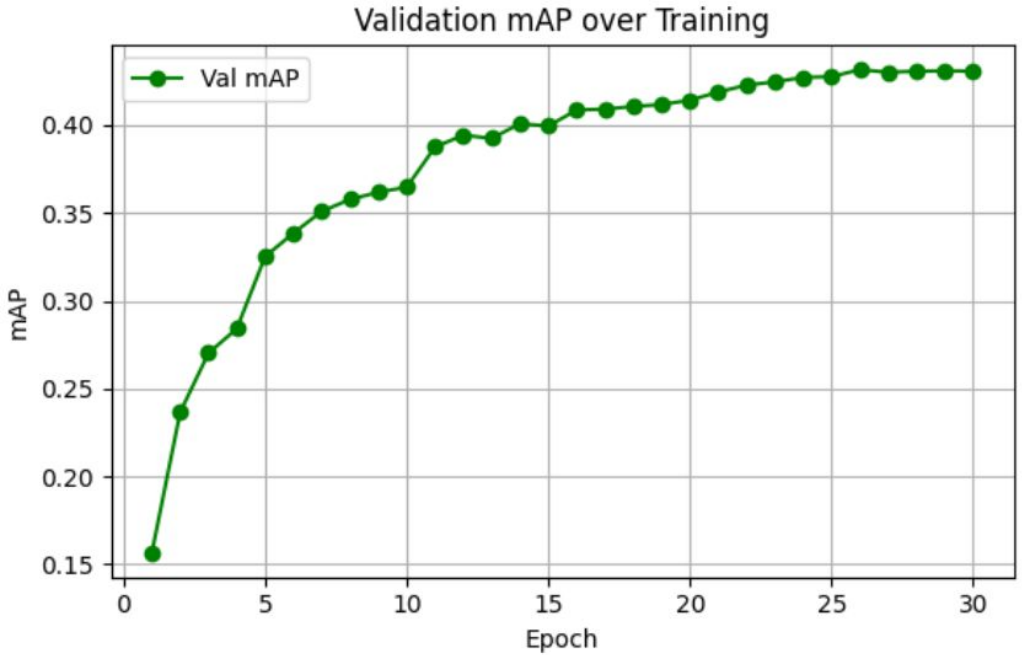
- 3 convolutional layers
- 1.4 million parameters
- learning rate: 1e-3 (using a CosineAnnealingLR scheduler from pytorch
- Does not use Random Cropping

	Unhelped	Weighted + focal loss	Oversampling	Oversampling + spectral augmentation
mAP (6 families mAP)	0.327 (0.6455)	0.368 (0.6684)	0.414 (0.7068)	0.405 (0.6997)
d'	1.934	2.058	2.097	2.096
lwlrap	0.555	0.581	0.610	0.605

Abandoned Pytorch CNN model (had worse results)

Oversampled model: mAP progress on validation data for 30 epoch run ⟶

(Model only ran for 30 epochs and did not stop early. This was the limit for Mikas' PC as it took 20 Hours to compute)



Reference model metrics

TABLE VI
EVALUATION PERFORMANCE FOR THE ARCHITECTURES CONSIDERED

Model	lr	weights	mAP	d'	$l\omega lrap$
CRNN	5e-4	0.96M	0.417 ± 0.003	2.068 ± 0.015	**0.519** ± 0.002
VGG-like	3e-4	0.27M	**0.434** ± 0.002	**2.167** ± 0.011	0.514 ± 0.003
ResNet-18	1e-5	11.3M	0.373 ± 0.001	1.883 ± 0.020	0.465 ± 0.001
DenseNet-121	5e-5	12.5M	0.425 ± 0.002	2.112 ± 0.032	0.505 ± 0.004

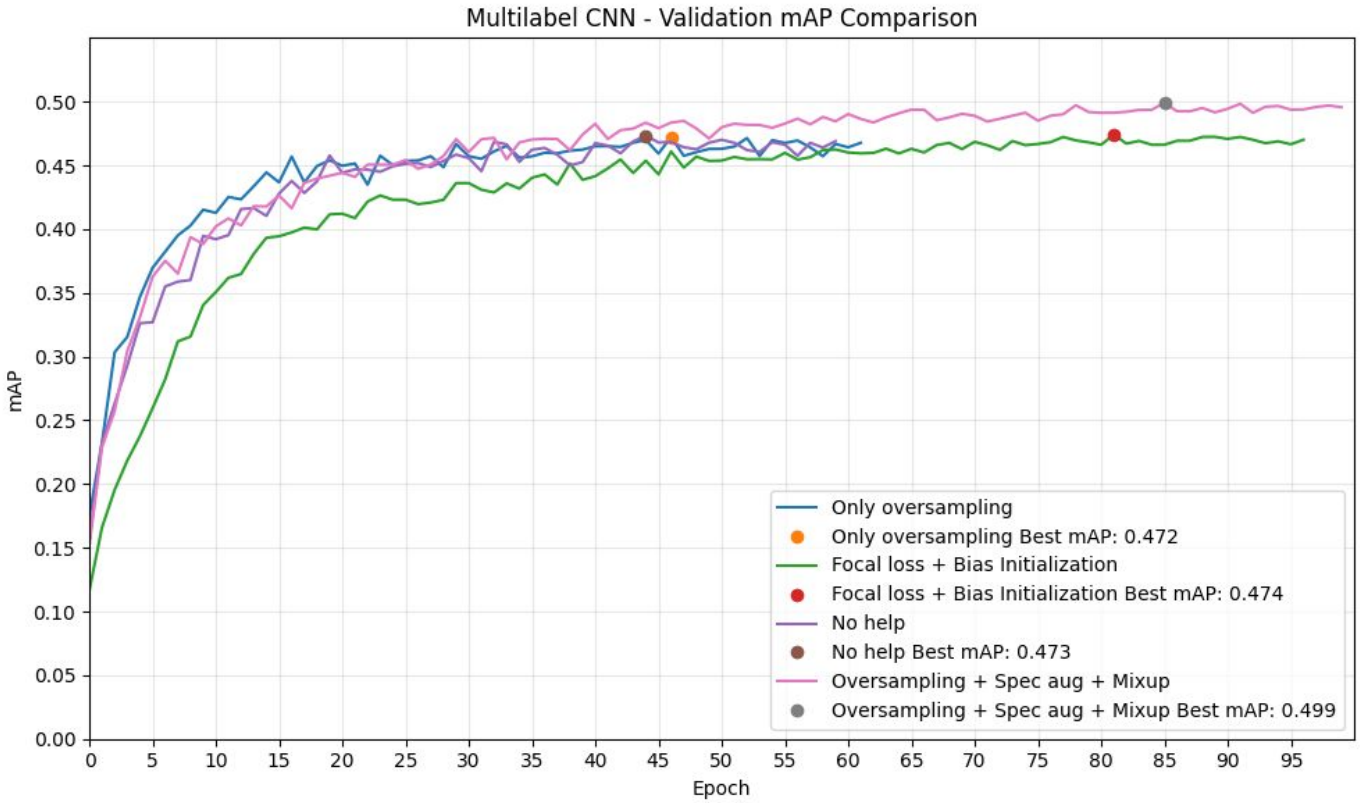
From original FSD50K paper:

Fonseca, E., Favory, X., Pons, J., Font, F., & Serra, X. (2021). Fsd50k: an open dataset of human-labeled sound events. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 30, 829-852.

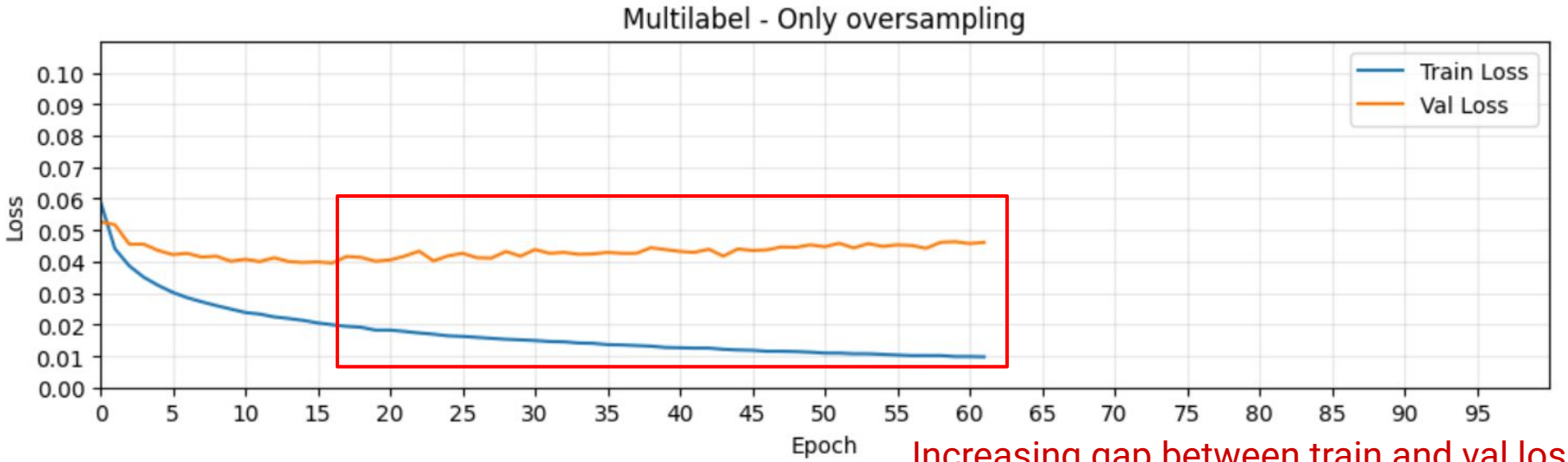
Trained models: Validation set metrics

model	best_epoch	val_mAP	family_mAP	d_prime	lwlrap
Oversampling + Spec aug + Mixup aug	86	0.4990	0.7750	2.5779	0.7078
With focal loss + bias init	91	0.4899	0.7714	2.5694	0.7093
Baseline	45	0.4734	0.7613	2.5361	0.7031
Only oversampling	47	0.4723	0.7518	2.4736	0.6915

Best multilabel model eval mAP comparisons

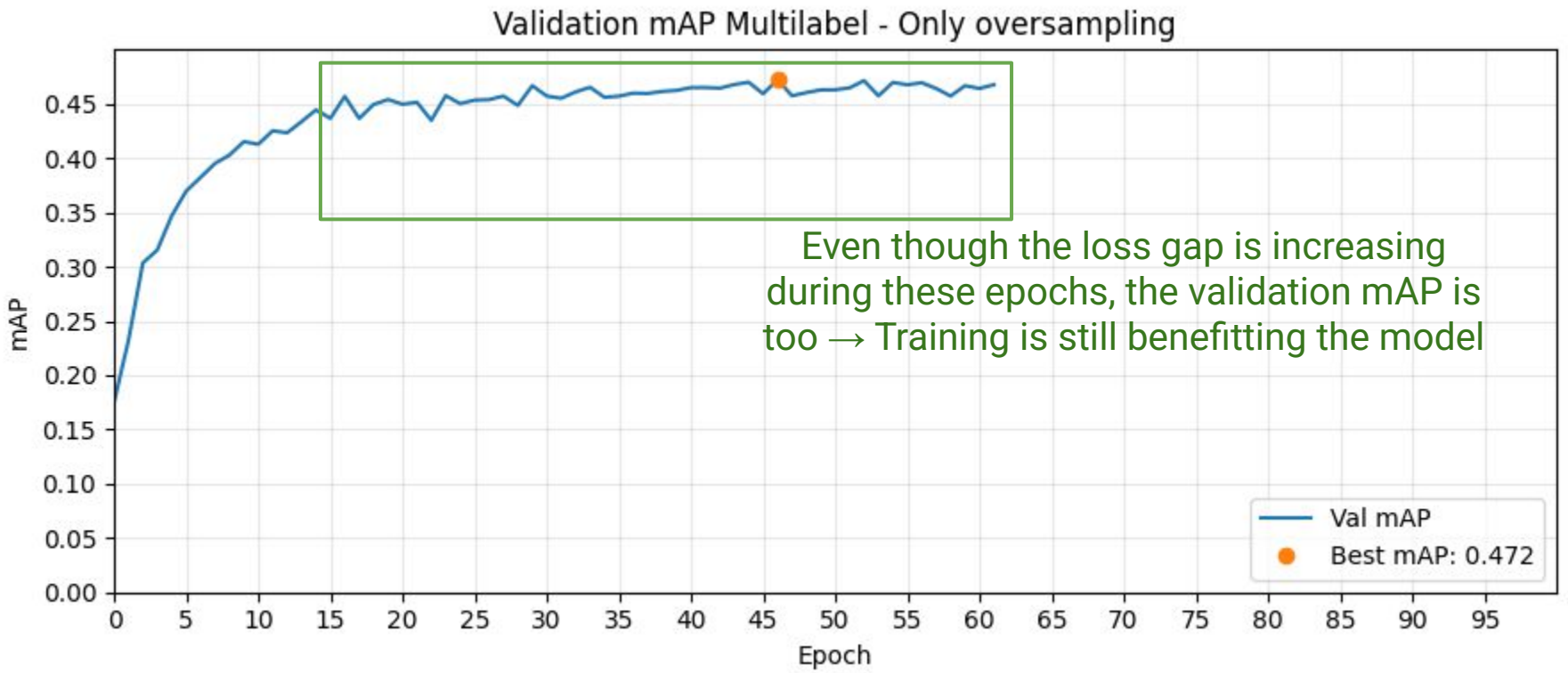


Are we overtraining?



Increasing gap between train and val loss could indicate overtraining (we saw this consistently across model variations)

Are we overtraining? No!



The Processing Problem

Issues:

- 50k audio files with 1.5 million trainable parameters is computationally heavy, and reducing parameters sacrificed model quality.
- Limited GPU access: two group members had Windows CPUs and one had an M2 Mac, meaning most local training had to run overnight
- Storing and transferring 50k preprocessed spectrogram files (about 8GB) individually to a Colab session took several hours

The Processing Problem

Solutions:

- ➢ Testing code changes and new features on a data subset before committing to a full training run
- ➢ Compressing all preprocessed files into a single zip before uploading to Drive, and unzipping to the runtime session, such that Colab setup only took a few minutes
- ➢ Getting access to Colab Pro and run full training with a L4 GPU

Why not use F1 score for evaluating our model?

- **F1 Macro:** takes the average F1 score for all labels

- **F1 Micro**: Pools all True positives (TP) and False Negatives (FN) across labels and calculates a single F1 score

- **F1 problem:** F1 requires a threshold to determine whenever something is either a TP or FN. Determining/setting a threshold can be arbitrary and it is thus preferred to use metrics for evaluation that are not threshold dependent
- Threshold problem can be minimized by e.g. making separate thresholds for each label by varying the threshold for maximizing F1.

$$\mathrm{F1} = \frac{2 * \mathrm{TP}}{2 * \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Binary Cross Entropy vs. focal loss

- For BCE easy negatives contribute more to total loss than than “hard” predictions, which the model should learn from.

- In general Focal loss helps the model learn from positives (α) as they are more rare in FSD50k than negatives and modulates/down weighs easy, or confident, predictions (γ)

Parameters:
- p = model's predicted probability
- y = ground truth label (0 or 1)
- α = class balance weight
- γ = focusing parameter (= 2)

Binary cross entropy

$$-[y \cdot \log(p) + (1-y) \cdot \log(1-p)]$$

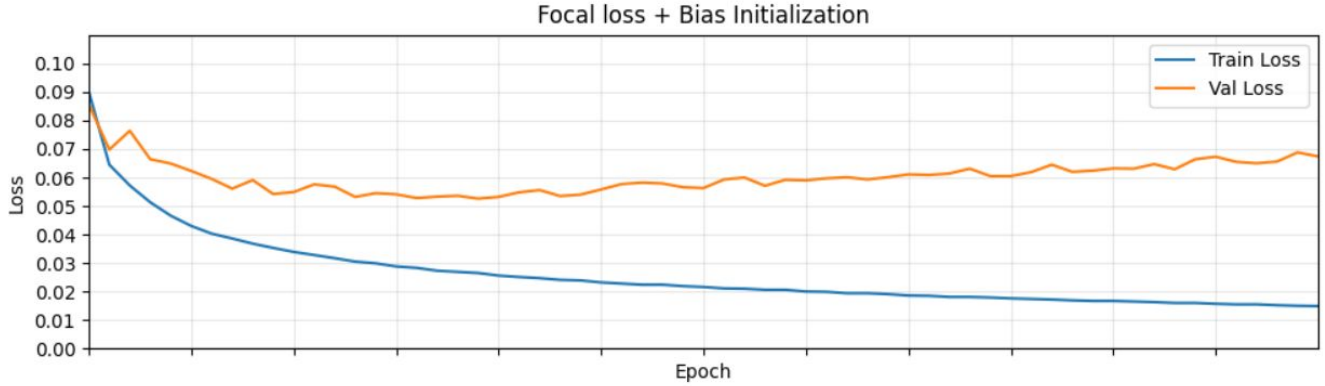
no modulation — all examples weighted equally

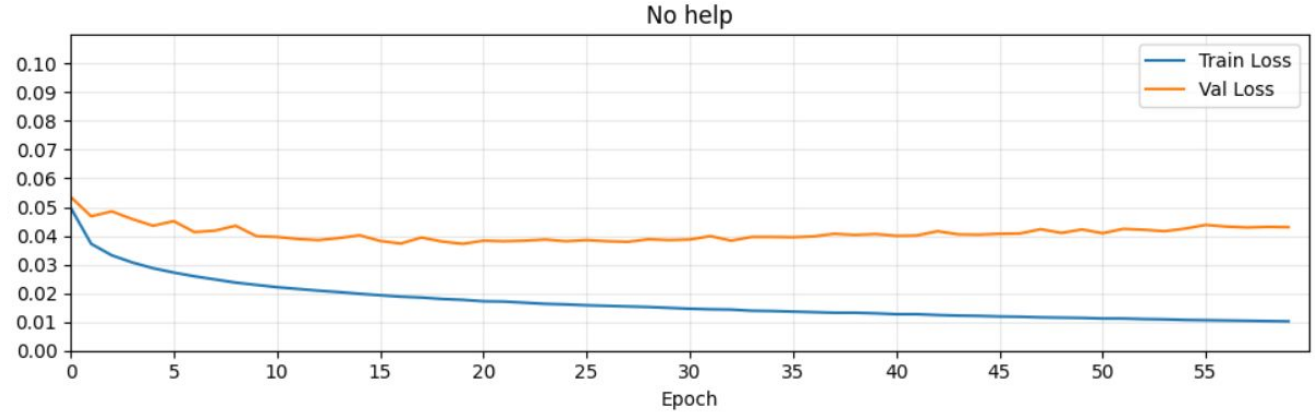
Focal loss

$$-[y \cdot \alpha \cdot (1-p)^{\gamma} \cdot \log(p) + (1-y) \cdot (1-\alpha) \cdot p^{\gamma} \cdot \log(1-p)]$$

modulating factor shown in blue

Binary Cross Entropy vs. focal loss





Focal loss + bias initialization hurts the model!

Model Enhancements - Types of augmentation

Spectral Augmentation:

- Masking bits, of various lengths, at random so that the model does not remember/recognize clips from earlier epochs. Preferably to avoid overfitting

Random cropping of clips:

- Selecting random parts of each spectrogram for training so it does not train on the same parts of the clip each time.

Mixup augmentation:

- At random takes 2 different clips and puts them together so that the model has to learn several noises at once.

Model Enhancements - Fixing uneven sampling

Weighted labels (Focal loss + initial bias or simple inverse frequency weights)

- Weights labels dependent on amount on clips with eac label. Uses Focal Loss so that the model is not rewarded as much for easy predictions (lots of zeroes). The initial bias uses weights from an earlier epoch to help the model when training further

Oversampling

- Synthetically produce more of the clips with rarer labels to even out the sample/label distribution

Combination of weighted labels and oversampling

Single-model specific enhancements

A non exhaustive list of things trying and the **best** of them based on mAP

weighed, unweighted, **oversampling**

No spectral augmentation, 1 layer spec. aug., 2 layers of spec. aug., 50% chance of spec aug

Focaloss, **crosentropyloss**