# Experiments

3-5x cross-validated, 8x oversampled, repeat for each recording type

# DONE

## Baseline:

* Baseline Resnet, no pretrained weights, no augmentations etc.
* Varied parameters: learning rate
* 45-50min for one 5x crossval run, total ~4h

## Add pretrained weights:

* Changed the resnet18 weights to be pretrained.
* Varied parameters: learning rate (same values as before)
* 45-50min for one 5x crossval run, total ~4h

Results: pretrained weights way better in every situation. Best lr is mostly similar except for vowels.

## Z Normalization (frequency wise VS general VS not at all)

Test 1:

* Normalize the input – each frequency bin is standardized with a unique mean and std (specific to each recording type)
* Varied parameters: learning rate (same values as before)

Results: normalizing the input way worse for speech, breath and a bit worse for vowels, cough

Test 1.1:

* Since normalization reduces the magnitude of the input data, the weights might need to get bigger. A large weight decay like I had it set in this.

Results: does not really matter – still worse than without normalization

Test 2:

* Test a general standardization (one mean + std for the entire spectrogram but still specific to each recording type

Results:   
breath seems better off without normalization,   
speech on the other hand improves,   
vowels do not change much at all (slightly better with norm?),   
cough seems slightly worse with norm

But in general, there is no completely clear tendency for either. There are always some settings for lr and wd where normalization goes from good to bad and vice versa.

## Cyclic Shift

* Various learning rates while turning cyclic shift on and off for each of them.

Results: for all but breathing extremely clear improvement of 2-3% for each lr.

For breathing: still mostly improving when turning the shift on but not for every lr.

## Dropout

Adding layers of dropout 2D and one regular dropout into the ResNet. Number of 2D layers and dropout rate are varied.

Test 1:

* Test various combinations of dropout and learning rates.
* Results:  
  breath and cough benefit from using dropout (although there is a single best run without dropout for coughs)  
  speech and vowels degrade in performance when using dropout. It seems that higher learning rates perform better for higher dropout rates.   
  Dropout clearly focuses more on positive examples (going from 50% tpr and 90% tnr to 70% and 80% which decreases the accuracy due to the class imbalance – consistent behavior over ALL modalities)

Test 2:

* Remove some of the dropout 2D layers (maybe 3 instead of 5?)
* Results: Not really any better results than with 4 dropout2D layers

Test 3:

* Check the best runs of some experiment and repeat this one setting with normalization on/off to decide.
* Results:  
  BREATH improves a bit (contrary to what we saw before) when adding nromalization  
  coughs get worse

FINAL CONCLUSION STILL OPEN

## Self-assessment penalization (test on old server, not all modalities necessary)

Penalizing participants that have not reported doing any tests (PCR or antigen) by reducing their loss. Thus, the network learns more from samples that have a PCR test to verify their claim of being healthy/infected.

Test 1:

* Test range of best lr = [6e-4, 3e-4, 1e-4] and vary self\_assessment\_penalty = [0.4, 0.6, 0.8, 1] with 1 being no change at all to previous tests = no penalty for people who did not provide a test
* Results:  
  Speech improves rather clearly for a penalty of 0.4 and 0.6  
  Breath slight improvement for 0.8 (and 0.6)  
  Cough slightly improved for 0.6  
  vowels get rather worse

Overall, adding a penalty term to the non-test diagnosed individuals mostly imrpves the performance slightly. No crazy performance boost but somewhat consistent.

## Exclude “Exposed” Participants

Apparently, the DiCOVA challenge did completely exclude all participants that were exposed to the virus through primary contacts even though some of them did test negative with a PCR test.

Test 1:

* What difference does it make if I exclude those samples from my development set.   
  I will be using the exclude\_confidently\_misclassified flag to test this for the first time. If it improves performance, I need to create a new hyperparameter.
* Results:  
  mostly a slight improvement when excluding the exposed data except for breathing data (slight decrease) and maybe speech (very inconsistent).

Conclusion:

* From now on, the exposed category should be excluded. There may be a slight increase in performance but possibly also not. But mostly it will be consistent with the DiCOVA dataset. This does not matter for the test set because I used the same as in the challenge, but the development set is more similar to the challenge.

## Augmentation Parameters:

watch out for whether the features are standardized or not. You cannot expect the same behavior for random noise/… if the input features have an std of 1 or 15.

Results:

Test 1:

* Transfer function simulations [0, 1, 3, 6] (should be in dB?), also vary lr and check again for normalization and whether it improves the performance.
* Results:  
  It is again not 100% consistent, but the best results are achieved by having some value > 0 for the tf\_simulation.   
  Breath and Cough improve a lot, but it cannot be 100% attributed to the tf\_sim: cough seems to improve most from adding the normalization back in.  
  Breath improved for ALL runs although some runs should have very similar settings as the runs before. Normalization is not quite clear, but it moves the best epoch away from the very early maximum in this modality (epoch 20 or so) and does not drop as much after its max.  
  Speech: normalization improves slightly, tf\_sim does not affect much.

Test 2:

* Random Gain (vary lr and random gain)
* Results:  
  Mostly no effect at all. Maybe a medium gain makes the runs more robust (less extrema in both directions?)  
  Vowels seems to have a rather consistent slight improvement for random gain = 3

Test 3:

* Random Noise
* Results:  
  Cough seems to improve a bit, especially in the AUC PR, similar with breaths.  
  vowels don’t seem affected (positively)  
  speech shows quite clearly its best performance using some noise (at higher lr 3e-4)

In conclusion:

Most of these augmentations do not ALWAYS improve the performance for ALL settings or all modalities (especially gain seems rather irrelevant) but the best performances were mostly observed using some variation of these augmentations.

## Mix-Up

Test 1:

* Vary mix-up alpha = [0.05, 0.2, 0.4] and mix-up probability = [0, 1] alongside lr
* Results:  
  Breath: decent improvement for mix-up, especially for lower lr 6e-5  
  Cough: mix-up looks good for AUC PR (AUC ROC maybe also slightly  
  Speech: looks a bit better (for higher lr 3e-4, low lr seems worse in general)  
  Vowels: don’t seems affected much either way.

Conclusion:

Mix-up seems to improve the performance rather consistently, especially AUC PR but again not 100% conclusive because for several settings, it makes the performance worse as well. But the best performances were measured using mix-up.

Very interesting as well: the loss drastically drops with mix-up (the higher the mix-up alpha the lower it gets)

A graph showing different colored lines

Description automatically generated

## Time Domain Augmentations

Adding samples with time domain augmentations (gain, time stretch, pitch shift, noise) that were calculated beforehand into the training dataset.

Test 1:

* Only testing speech dataset. 5 augmented datasets have been created. 4x the positive class has been augmented (each sample once) and 1x the negative class has been augmented (each sample once as well). All combinations of number of positive VS negative augments are tested to get a perfect ratio or to see whether any augmentations improve performance at all. Varied alongside will be the lr = [6e-4, 3e-4, 1e-4].
* Results:  
  The best performance was by 4x positive and 1x negative oversampled augmented datasets, but the second-best performance was with 0x and 0x respectively. So, it is not 100% conclusive and needs to be tested more but it may improve performance.

Test 2:

* Repeat with more learning rates and a 5x cross validation.
* Results:  
  only speech tested. Using an additional augmented dataset with negative labels improves very consistently the performance. The best settings are 0x1x and 4x1x. The results suggest that the augmentations (or at least the negatively labeled one) perform better with a lower lr.

Test 3:

* Repeat the previous process for the other datasets as well.  
  Breath: neg 1x mostly an improvement, 2x or 4x pos also slightly better  
  Cough: neg 1x seems decent, pos mostly makes it worse  
  Vowels: best without any augmentations

Conclusion: for all but vowels, adding augmentations of negative samples improves the performance. Whether adding augments for positively labelled data helps varies a lot and is not clear.

## Learning rate parameters:

Test 1:

* Lr decay:  
  Vary lr and lr decay [0.95, 0.97, 0.99] which after 1360 epochs results in a coefficient [0.001, 0,02, 0.27]  
  I am also tracking the individual ID performance for a new list for ‘confidently misclassified’ samples.
* Results:

Test 2:

* Weight decay & batch size
* Results:  
  breath: significant influence of batch size (higher is better) and weight decay (lower is better)  
  cough: smaller batch has rather clearly a better AUC PR and maybe slightly better AUCROC, other than that inconclusive  
  speech: not as clear but batch=64 seems best, wd=1e-5 seems most consistent (1e-4 has the best run but also more variance)  
  vowels: wd=1e-4 clearly better than 1e-5, batch either 32 or 128?

Test 3:

* Lr in (0.1, 0.3, 1, 3 as coefficients compared to the output layer lr – in between linearly interpolated)

Results  
Breaths: higher lr better again with a lower lr\_in (0.3)  
Coughs, speech, and vowels: best performance for lr\_in = 1 just like before

## Feature Resolution (time/frequency res)

Test 1:

* Check if I get similar results with a twice the time resolution for the breath feature set (half the hop-size and FFT-length)  
  Result: it seems like it is a little bit worse

Test 2:

* Save weights for all 5 cross-validation runs and test MIL to see if the increased total number of time steps makes up for the slightly worse performance.
* Results:  
  NO! even after mil the results are worse than the tests from before (though very little hyperparameter tuning was done with the new dataset compared to the 46ms one)

Test 3:

* Create augmentations for this dataset and test again.
* Results:  
  no improvement when adding time domain augmentations. 0x 0x best performance

Test 4:

* Test whether the fmax 5500 (compared to the original 11000) with the original hop size and FFT settings gets a better performance.
* Results:  
  NO! seems worse (although no extensive hyperparameter tuning was done)

## N\_time\_steps

THESE EXAMINATIONS ARE IGNORED FOR NOW BECAUSE IT WOULD TAKE TOOOO MUCH TIME! 336 timesteps will be used because everything was finetuned for this until now

Test 1:

* Test also in combinations of dropout (and mix-up) because I suspect that dropout performs better, if there are fewer datapoints, so it does not overfit too easily. n\_time\_steps =[112, 224, 336, 448]
* Results:  
  Dropout does not get better results (maybe for very low n\_time\_steps = 112 but there the performance is just bad in general)  
  ALL modalities and ALL learning rates show that the highest number of time steps performs the best! And the processing time is barely longer.

Test 2:

* Save weights for 224- and 448-time steps. 448 should perform better.  
  Make some MIL hyperparameter runs to check if the improvement is larger for 224 and if this makes up for the fact that it performed worse before.
* Results:  
  Both benefit from using MIL but 224 timesteps benefits more. In the end both achieve mostly similar results after MIL. Except for vowels (something went wrong I think, but I don’t know what) where 224 clearly outperformed 448 after MIL. I think it makes sense to keep 336 as middle ground and because I tuned hyperparameters for this setting. But it would be more impressive in terms of improvement before/after MIL if I chose 224…

Still need to write down the differences in detail. Where the decrease in time steps REALLY negatively affect the performance. If 224 is just slightly lower, I could reasonably use that. 🡪 Find out where the largest drop is!

## Exclude Confidently Misclassified

Test 1:

* Use the examples that were found months ago to be confidently misclassified to see if they still improve the performance when excluded.
* Results:  
  Cough: Clear improvement  
  Breath: decent AUC PR improvement, AUCROC maybe a bit  
  Speech: clear improvement in eval set, slight improvement in test  
  Vowels: only result that seems worse off with the exclusion  
  Test set:  
  significantly less clear improvement but mostly still improvement (aside from vowels)

Test 2:

* test as well with new expanded test sets (I excluded some before because of their bad or lack of audio quality annotations)
* Results (only breath and vowels had a significant amount of added samples in the test set):  
  Breath:  
  - old test set: 80.0 AUCROC, 35.2 AUCPR (exclude=True)  
  - new increased test set: 79.6 AUCROC, 35.5AUCPR (exclude=True)  
  Vowels:  
  - old test set: 81.2, AUCROC, 55.6 AUCPR (exclude=False)  
  - new increased test set: 80.5 AUCROC, 53.5 AUCPR (exclude=False but AUCPR would be better if True)
* Conclusion:  
  the samples that I had filtered out due to a lack in audio quality made sense because the performance dropped a bit when they were added (but it makes sense to add them for comparability with the DiCOVA test set)

## Focal Loss

Test 1:

* Vary lr and focal loss = [0, 0.5, 1, 3] – focal loss=0 corresponds to just using the BCE loss (like before)
* Results:  
  Across all modalities slight to clear improvements for values between 0,5-1. Vowels even performed best with a very high parameter (3), but it also showed a higher variance within the run.

## Residual Norm

We have 5 parameters that can be varied:

1. use\_resnorm = [True, False]:   
   using resnorm layers throughout the network (once every basic block after batchnorm)
2. Resnorm\_affine = [True, False]:  
   learnable parameters to scale and shift after the normalization but before adding input from the skip connection
3. Resnorm\_gamma=[0,…., 1]  
   how much does the original input contribute to the final outut. Value between 0 and 1 for the skip connection coefficient. Instance normalized path gets (1-gamma) as coefficient.
4. Input\_resnorm=[True, False]:  
   Add a resnorm layer at the very beginning of the network
5. Track Running stats:  
   keep a rolling average for each frequency to be normalized with

Results:

* Pretty bad… only breath category uses all parameters for improvement and even there its no sure whether use\_resnorm helps (but input resnorm seems to do so)  
  Cough and speech perform best without any inclusion of resnorm  
  Vowels is very unclear. Input resnorm seems to help slightly but without tracking stats

# DOING

# TODO

## Resnet50

# Multiple Instance Learning

First test Double Dense layer mit 1 linear layer am output nach dem MIL algorithmus

512🡪64🡪(through mil attention mechanism)🡪1

Breath worked quite well, vowels were okay  
speech and cough were pretty bad compared to older results (from the n\_timesteps investigation)

# DOING

Repeat hyperparameter training with additional layers after the MIL attention mechanism:

64 (with dropout)🡪 (through mil attention mechanism)🡪16(with dropout + ReLU) 🡪 1

# TODO