GDSC Soution D

Two-sided approach

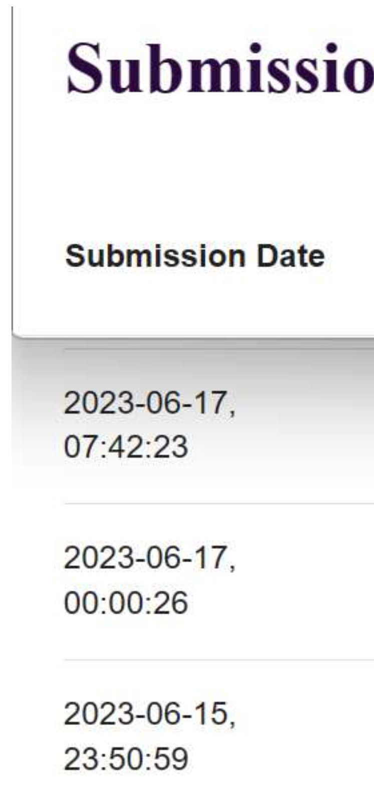
1. Walk through the tutorials and play with paramete

2. Create a code base that can also be used to
    - classify other animal sounds, e.g. bird songs a
    - can be used e.g. in Kaggle competitions (e.g.
    - can run on a local machine
    - implement ideas and best practices from thes

Approach 1: Bas

Parameters to play with:

- Learning Rate
- Sampling Rate
- Number of Epochs
- Patience Number
- Loss Function


Submissio

Submission Date
2023-06-17, 07:42:23
2023-06-17, 00:00:26
2023-06-15, 23:50:59

Approach 1

- Learning rate less than 1e-6 in combination with r
- Jobs with more than 50 epochs crashed due to A
every epoch
- Sampling rate 44.1 kHz better than default 16 kH
- High Patience numbers tend to overfit
- Cross Entropy Loss yields better results than F1-

Approach 2

- Reusability:
  - Able to run on local machine with minimum ef
  - Able to be used in Kaggle competitions
  - Able to classify a variety of animal sounds (e.g
- Implement Best Practices from Kaggle competitio
  - Experiment with different pretrained models
  - Experiment with different optimizers
  - Hyperparameter tuning
  - Use a learning rate scheduler
- Experiment with audio-specific settings

Approach 2: Experiment

- Play with different time windows (0.6 to 60 secon
- Experiment with various data augmentations
- Different minimum and maximum frequencies
- Consolidate audio chunks
- How to mix white noise and background noise to

Approach 2

- Steps
  1. load all audio data
  2. split audios into equally sized chunks
  3. upsample chunks to handle underrepresentec
  4. create augmentation plan for chunks
  5. train, validate and predict for chunks
  6. consolidate predictions for chunks into predict
- Modes
  1. pretrain: for hyperparameter tuning
  2. train and validate
  3. predict

Approach 2: Hyperp

- Python package “Optuna”
  - Easy to implement
  - State-of-the-art optimization to find parameter
  - Not part of the provided GDSC package
- To accelerate the process a subset of 10 classes
  maximum of 8 epochs

Approach 2: Param

- Pretrained model ['efficientnet_b0', 'resnet18', 'ale
- Optimizer ['Adam', 'RMSprop', 'SGD']
- Learning rate and minimum learning rate (for LR
- Target sampling rate
- Number of n-mels (for mel spectrogram; skipped
- Time windows for audio chunks
- Augmentation parameters:
  - application probability for each type of augme
  - maximal strength of noise vs. signal (backgrou
  - impulse response
  - jitter noise
  - time stretch and pitch shift
  - time masking and frequency masking

Approach 2: Hyperparar

- Tuning all parameters at once is by far too ineffici
  - Tuning single parameters and small groups of other
  - Made a decision which parameter to tune and assumptions

  “If there is only time to optimize one hyper-para descent, then this is the hyper-parameter that is [1]

Approach 2: Tuned & Ski

- Found optima:
  - Learning rate: 1.5e-3
  - Minimum learning rate (for LR scheduler): 1/2
  - Time windows: long window: 40 secs, focus w
- Skipped
  - Pretrained model: All bird song recognition alg
  - Optimizer: Adam is the fastest converging alg
  - Audio Sample Rate: used 44.1 kHz (so that th
  - n_mels for Mel-Spectrogram: EfficientNet_B0
  - augmentation related parameters
  - maximum and minumum frequencies (maximu
  played with various minimum frequencies betw

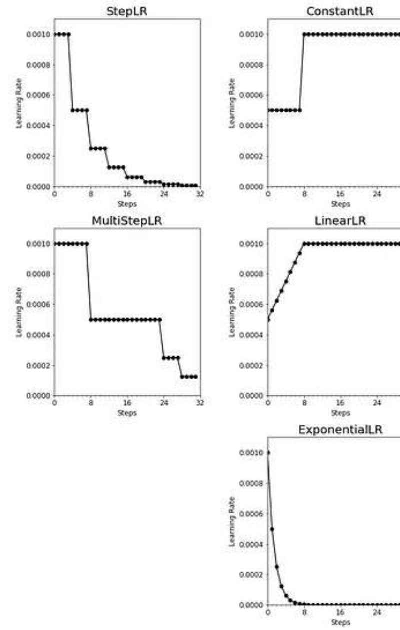
Approach 2: Learnin

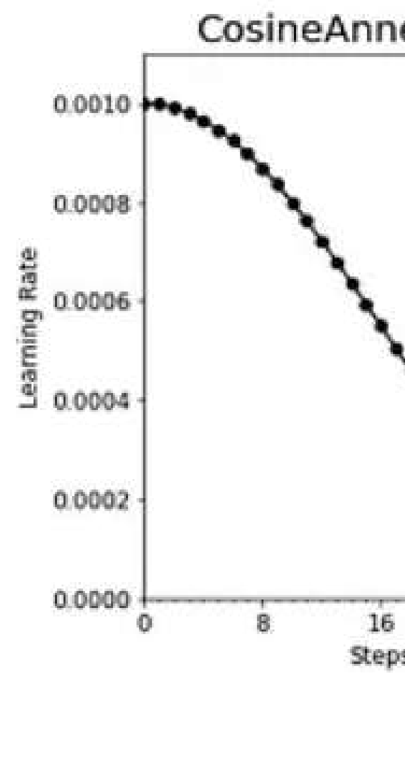
State-of-the-art learning rate scheduler that is used for bird song recognition:

CosineAnnealingLR

Required parameters:

- Initial learning rate
- Minimum learning rate

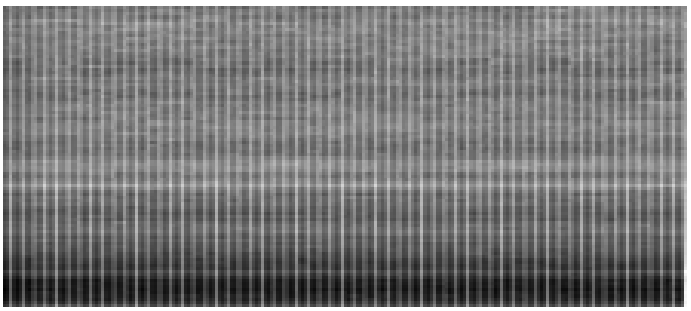
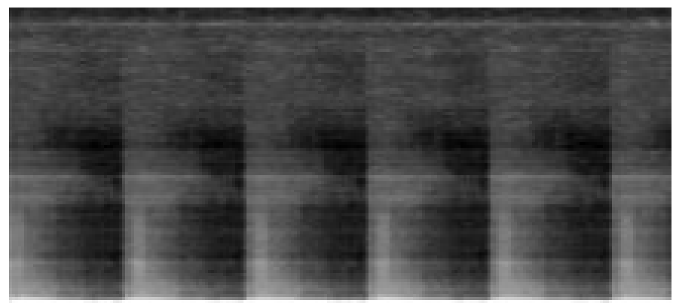
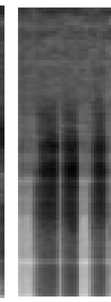




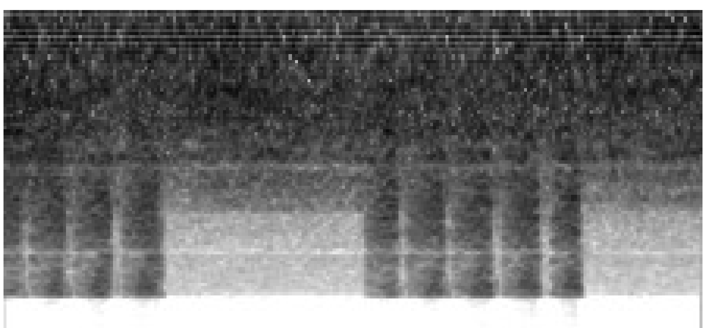
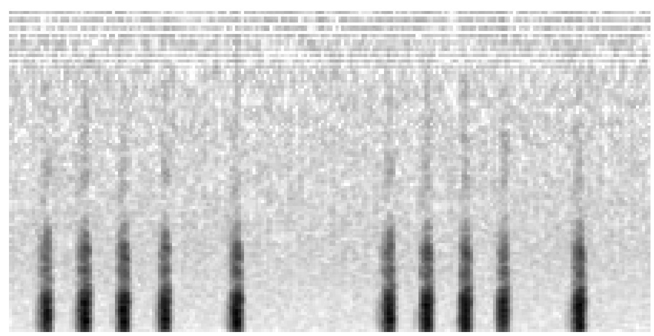
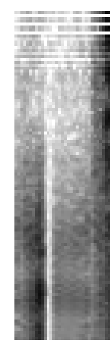
Approach 2: Ti

Special patterns appear in the spectrogram for grass
(chirp vs. no chirp) and on very short time windows (
chirp.

Examples from 60s time window:

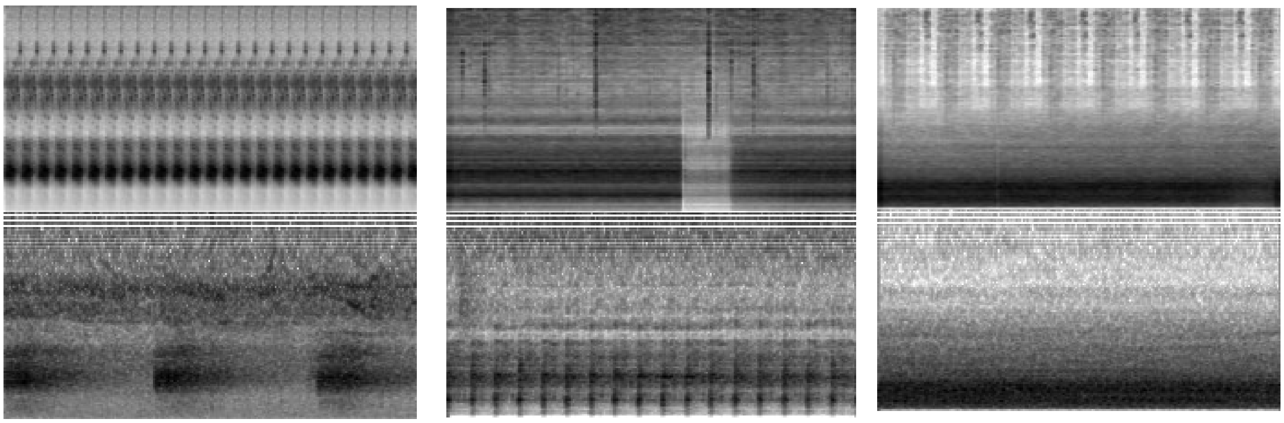
Examples from 0.6s time window:

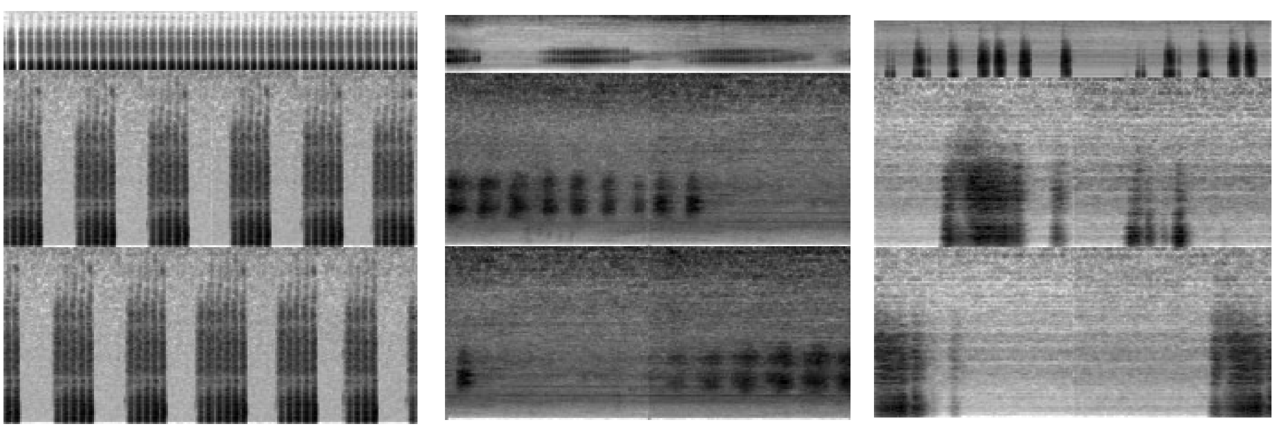
In order to make use of all the information, spectrogr
were combined:

Approach 2: Combi

Examples of combined spectrograms. In some cases
provide distinct patterns



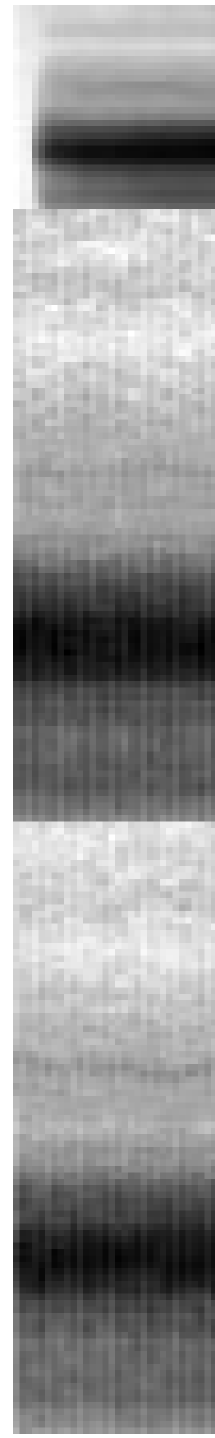
long: 60s, focus 0.6s



long 16s (top) with four foci à 1s

Approach 2: Selectio

Depending on how many foci I want to use I would split the long time window into that number of chunks, then take the absolute values of the audio data, calculate the rolling average of the focus window, and select the one with the highest value (to make sure to not just have noise there)



Detailed view of last example from previous slide (40s top + 4x2s)

Approach 2: Au

Applied augmentations:

1. Background noise: Taken from publicly availab
   wind and rain). Mixed to signal at maximal stren
2. White noise: Mixed to signal at maximal stren
3. Jitter: Sampling jitter occurs when audios are
   have been downsampled to 44.1 kHz, about 25
   finally back to 44.1 kHz
4. Time stretch: Maximal stretch +/-2.5%, probab
5. Pitch shift: Maximally 0.5 semitones, probabili

Skipped:

1. Time masking and frequency masking: results
2. Cancelled due to bugs: extracting signals from
   overall median / time median / frequency media

Approach 2: M

Upsampling: In order to deal with class imbalances, have been upsampled to a certain amount (number between 40 and 80)

Augmentation plan: After splitting the audio data into will be determined which augmentation to apply

Consolidation of predictions of chunks to predictions classes (labels): Easy ways such as building the ave value yielded inferior results, hence transformations probabilities are about equal to one 90% probability

Local vs. Cloud: In order to make the locally develop code requires a few changes:

- parsing some arguments, such as input source a
- Code related to the optuna package have to be s locally, predictions only occur in the cloud and do

Approach 2: F

Various time windows yielded mixed results, between 71 and 83.

The submission titled “final submission” uses the best result (4 times focus 2 secs) and the result from approach 1 by applying the ensemble method of voting.

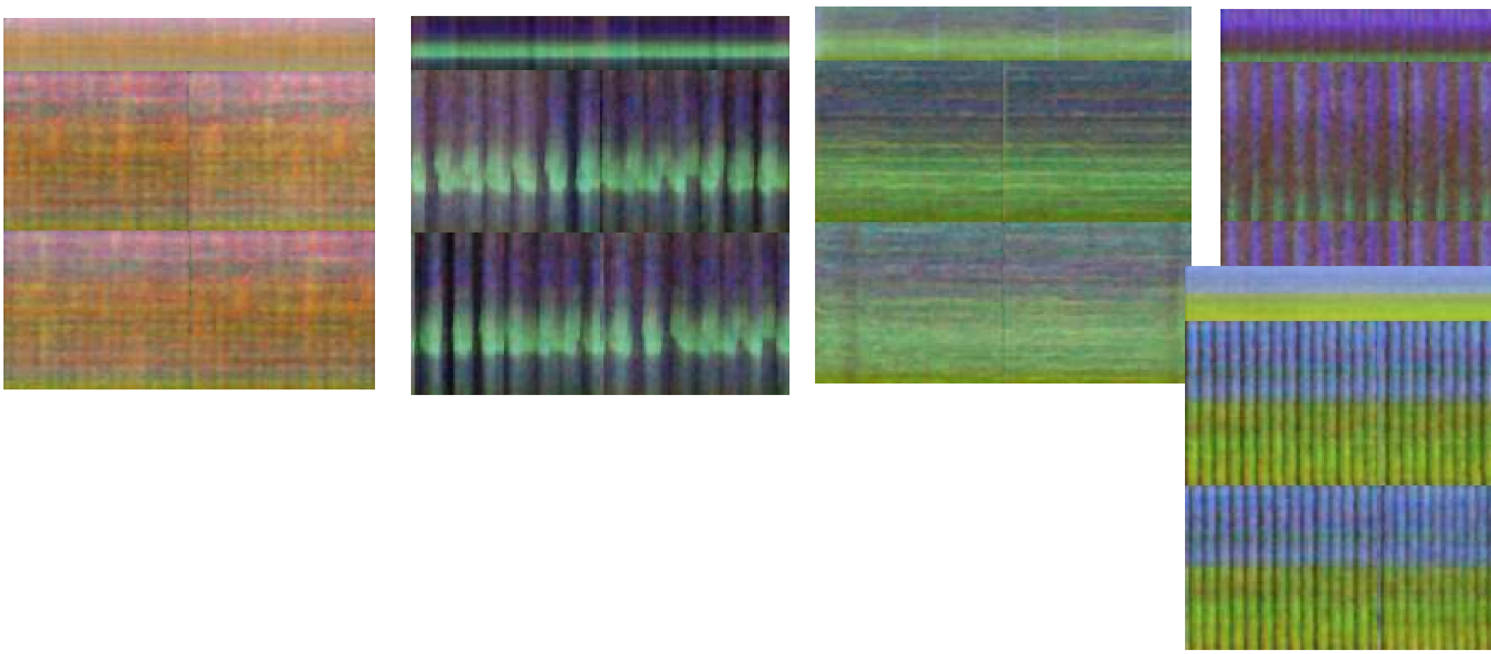
At this time I was supposed to be finished (rank 24 at that time)

Submission Date	De
2023-07-09, 06:13:13	fina
2023-07-09, 05:41:17	ens
2023-07-09, 03:37:12	4 ti
2023-07-07, 23:05:23	tim
2023-07-04, 07:01:53	tim

Approach 2: Next

The idea of using colors for different frequency band
give it a shot. Also, I wanted to arrange several focus

Examples (Four octaves projected to RGB channels,



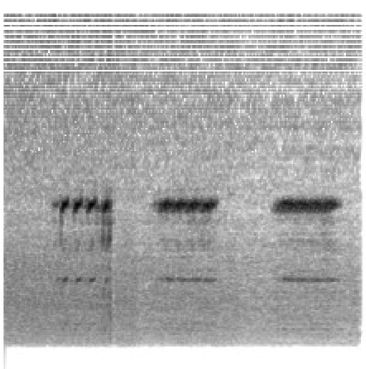
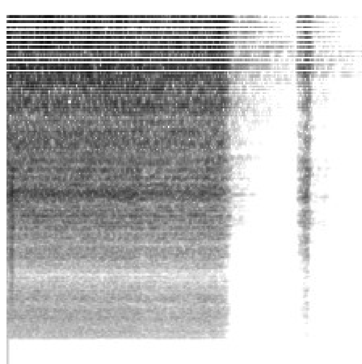
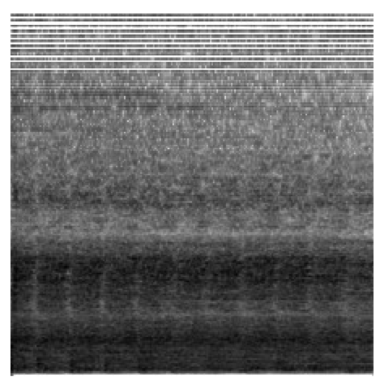
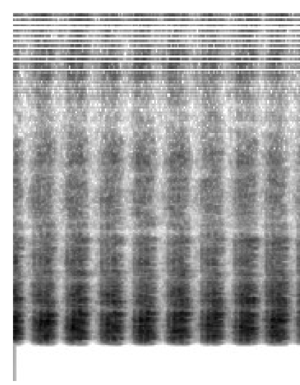
Approach 2: Se

Results using colors were disappointing, however using more focus windows seemed to yield better results. My reasoning was that pushing that to the extreme might achieve even better results.

Submission Date
2023-07-12, 22:46:25
2023-07-11, 00:01:49

One last attempt before the deadline with only 1 second focus windows, Result also disappointing.

Submission Date
2023-07-15, 16:08:47

Approach 2

To close things up I decided to do a weighted voting
results, the results being the weights.

Submission Date	Description
2023-07-15, 23:50:34	voted

Huge surprise:
Creating a strong prediction from several inferior pre

Things

Given more time, there would have been more things

- Make attempts “normalizing” the spectrograms us and frequency or overall median
- Make attempts to extract signal from noise (would identify)
- Create an additional class for “noise only” that wil predictions of audio chunks into predictions for th

Thank

Leaderboard Validation

Rank	Team Name	Validation Scor
1	It is a bug	92.0271
2	Gandalf the Data-Wise and Aragorn the Code-Master	90.9662
3	HiveMind Data Insectors	89.7086
4	BuzzAstral	89.5362
5	The Buzz Group	89.3105
6	gracehopper	89.2014
7	ANTificial neural network	88.7356

Code Re

Code to make program cloud (AWS) compatible:

```
parser = argparse.ArgumentParser()
parser.add_argument("--data_channel", type=str, default=os.env
parser.add_argument("--test_dir", type=str, default="test")
parser.add_argument("--output_dir", type=str, default=os.envir
args, _ = parser.parse_known_args()
output_dir = f"{args.output_dir}/"
input_root = f"{args.data_channel}/"
```

Relevant programs and jupyter notebooks:

- baseline_ast_train: based on tutorial; result 81.96%
- train_2: long time window: 40 sec, focus window: 1 sec; result 79.
- train_2: long time window: 60 sec, focus window: 0.6 sec; result 7
- train_2: long time window: 16 sec, four focus windows: 2 sec; resu
- train_3: long time window: 40 sec, four focus windows: 2 sec, usin
- train_4: long time window: 16 sec, four focus windows: 1 sec; resu
- train_6: long/short time window: 1.25 sec; result 73.70%
- train_x: cloud version, copied and adapted from train_2/3/4/5/6
- voting: combine previous results; result

other:

- train_5: long time window: 2.4 sec, focus windows: 2 sec (cancell

Sourc

[1] Hyperparameter Tuning: https://machinelearningr
learning-neural-network-practitioners/

[2] Learning rate schedulers: https://towardsdatascie
schedulers-in-pytorch-24bbb262c863

[3] Image classification tutorial for beginners: https://t
classification-tutorial-for-beginners-94ea13f56f2

[4] Multiclass Image Classification with PyTorch: http
image-classification-with-pytorch-af7578e10ee6

[5] Hyperparameter Tuning with Optuna: https://optur
use-optuna-without-remote-rdb-servers

[6] Comparison of Pretrained Models: https://www.m