Dependencies

In [1]:
```
!pip install ../input/timm-pytorch-image-models/pytorch-image-mod
els-master/
!pip install ../input/torchlibrosa/torchlibrosa-0.0.5-py3-none-an
y.whl
```

```
Processing /kaggle/input/timm-pytorch-image-models/pytorch-image-
models-master
Requirement already satisfied: torch>=1.4 in /opt/conda/lib/pytho
n3.7/site-packages (from timm==0.4.6) (1.7.0)
Requirement already satisfied: torchvision in /opt/conda/lib/pyth
on3.7/site-packages (from timm==0.4.6) (0.8.1)
Requirement already satisfied: future in /opt/conda/lib/python3.7
/site-packages (from torch>=1.4->timm==0.4.6) (0.18.2)
Requirement already satisfied: typing_extensions in /opt/conda/li
b/python3.7/site-packages (from torch>=1.4->timm==0.4.6) (3.7.4.
3)
Requirement already satisfied: dataclasses in /opt/conda/lib/pyth
on3.7/site-packages (from torch>=1.4->timm==0.4.6) (0.6)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/
site-packages (from torch>=1.4->timm==0.4.6) (1.19.5)
Requirement already satisfied: pillow>=4.1.1 in /opt/conda/lib/py
thon3.7/site-packages (from torchvision->timm==0.4.6) (7.2.0)
Building wheels for collected packages: timm
  Building wheel for timm (setup.py) ... - \ | done
  Created wheel for timm: filename=timm-0.4.6-py3-none-any.whl si
ze=292256 sha256=2cf8154825528019f916252583f65c04dd77a168914d1bb0
b807c8ec69a12d50
  Stored in directory: /root/.cache/pip/wheels/b2/4e/24/ca2e6fc7f
ceb1e8f1f4d3e5dd21df64327a03cf318d915c1bb
Successfully built timm
Installing collected packages: timm
Successfully installed timm-0.4.6
Processing /kaggle/input/torchlibrosa/torchlibrosa-0.0.5-py3-none
-any.whl
Installing collected packages: torchlibrosa
Successfully installed torchlibrosa-0.0.5
```

Libraries

In [2]:
```
import cv2
import audioread
import logging
import os
import random
import time
import warnings

import librosa
import numpy as np
import pandas as pd
import soundfile as sf
import timm
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.utils.data as torchdata

from contextlib import contextmanager
from pathlib import Path
from typing import Optional

from albumentations.core.transforms_interface import ImageOnlyTra
nsform
from torchlibrosa.stft import LogmelFilterBank, Spectrogram
from torchlibrosa.augmentation import SpecAugmentation
from tqdm import tqdm
```

Utilities

In [3]:
```
def set_seed(seed: int = 42):
    random.seed(seed)
    np.random.seed(seed)
    os.environ["PYTHONHASHSEED"] = str(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)  # type: ignore
    torch.backends.cudnn.deterministic = True  # type: ignore
    torch.backends.cudnn.benchmark = True  # type: ignore


def get_logger(out_file=None):
    logger = logging.getLogger()
    formatter = logging.Formatter("%(asctime)s - %(levelname)s -
%(message)s")
    logger.handlers = []
    logger.setLevel(logging.INFO)

    handler = logging.StreamHandler()
    handler.setFormatter(formatter)
    handler.setLevel(logging.INFO)
    logger.addHandler(handler)

    if out_file is not None:
        fh = logging.FileHandler(out_file)
        fh.setFormatter(formatter)
        fh.setLevel(logging.INFO)
        logger.addHandler(fh)
    logger.info("logger set up")
    return logger


@contextmanager
def timer(name: str, logger: Optional[logging.Logger] = None):
    t0 = time.time()
    msg = f"[{name}] start"
    if logger is None:
        print(msg)
    else:
        logger.info(msg)
    yield

    msg = f"[{name}] done in {time.time() - t0:.2f} s"
    if logger is None:
        print(msg)
    else:
        logger.info(msg)
```

In [4]:
```
logger = get_logger("main.log")
set_seed(1213)
```

```
2021-04-04 00:58:06,885 - INFO - logger set up
```

Config

In [5]:
```
class CFG:
    ######################
    # Globals #
    ######################
    seed = 1213
    epochs = 35
    train = True
    folds = [0]
    img_size = 224
    main_metric = "epoch_f1_at_05"
    minimize_metric = False

    ######################
    # Data #
    ######################
    train_datadir = Path("../input/birdclef-2021/train_short_audi
o")
    train_csv = "../input/birdclef-2021/train_metadata.csv"
    train_soundscape = "../input/birdclef-2021/train_soundscape_l
abels.csv"

    ######################
    # Dataset #
    ######################
    transforms = {
        "train": [{"name": "Normalize"}],
        "valid": [{"name": "Normalize"}],
        "test": [{"name": "Normalize"}]
    }
    period = 20
    n_mels = 128
    fmin = 20
    fmax = 16000
    n_fft = 2048
    hop_length = 512
    sample_rate = 32000
    melspectrogram_parameters = {
        "n_mels": 224,
        "fmin": 20,
        "fmax": 16000
    }

    target_columns = [
        'acafly', 'acowoo', 'aldfly', 'ameavo', 'amecro',
        'amegfi', 'amekes', 'amepip', 'amered', 'amerob',
        'amewig', 'amtspa', 'andsol1', 'annhum', 'astfly',
        'azaspi1', 'babwar', 'baleag', 'balori', 'banana',
        'banswa', 'banwre1', 'barant1', 'barswa', 'batpig1',
        'bawswa1', 'bawwar', 'baywre1', 'bbwduc', 'bcnher',
        'belkin1', 'belvir', 'bewwre', 'bkbmag1', 'bkbplo',
        'bkbwar', 'bkcchi', 'bkhgro', 'bkmtou1', 'bknsti', 'blbgr
a1',
        'blbthr1', 'blcjay1', 'blctan1', 'blhpar1', 'blkpho',
        'blsspa1', 'blugrb1', 'blujay', 'bncfly', 'bnhcow', 'bobf
ly1',
        'bongul', 'botgra', 'brbmot1', 'brbsol1', 'brcvir1', 'bre
bla',
        'brncre', 'brnjay', 'brnthr', 'brratt1', 'brwhaw', 'brwpa
r1',
        'btbwar', 'btnwar', 'btywar', 'bucmot2', 'buggna', 'bugta
n',
        'buhvir', 'bulori', 'burwar1', 'bushti', 'butsal1', 'buwt
ea',
        'cacgoo1', 'cacwre', 'calqua', 'caltow', 'cangoo', 'canwa
r',
        'carchi', 'carwre', 'casfin', 'caskin', 'caster1', 'casvi
r',
        'categr', 'ccbfin', 'cedwax', 'chbant1', 'chbchi', 'chbwr
e1',
        'chcant2', 'chispa', 'chswar', 'cinfly2', 'clanut', 'clcr
ob',
        'cliswa', 'cobtan1', 'cocwoo1', 'cogdov', 'colcha1', 'col
tro1',
        'comgol', 'comgra', 'comloo', 'commer', 'compau', 'compot
1',
        'comrav', 'comyel', 'coohaw', 'cotfly1', 'cowscj1', 'creg
ua1',
        'creoro1', 'crfpar', 'cubthr', 'daejun', 'dowwoo', 'ducfl
y', 'dusfly',
        'easblu', 'easkin', 'easmea', 'easpho', 'eastow', 'eawpew
', 'eletro',
        'eucdov', 'eursta', 'fepowl', 'fiespa', 'flrtan1', 'foxsp
a', 'gadwal',
        'gamqua', 'gartro1', 'gbbgul', 'gbwwre1', 'gcrwar', 'gilw
oo',
        'gnttow', 'gnwtea', 'gocfly1', 'gockin', 'gocspa', 'gofty
r1',
        'gohque1', 'goowoo1', 'grasal1', 'grbani', 'grbher3', 'gr
cfly',
        'greegr', 'grekis', 'grepew', 'grethr1', 'gretin1', 'grey
el',
        'grhcha1', 'grhowl', 'grnher', 'grnjay', 'grtgra', 'gryca
t',
        'gryhaw2', 'gwfgoo', 'haiwoo', 'heptan', 'hergul', 'herth
r',
        'herwar', 'higmot1', 'hofwoo1', 'houfin', 'houspa', 'houw
re',
        'hutvir', 'incdov', 'indbun', 'kebtou1', 'killde', 'labwo
o', 'larspa',
        'laufal1', 'laugul', 'lazbun', 'leafly', 'leasan', 'lesgo
l', 'lesgre1',
        'lesvio1', 'linspa', 'linwoo1', 'littin1', 'lobdow', 'lob
gna5', 'logshr',
        'lotduc', 'lotman1', 'lucwar', 'macwar', 'magwar', 'malla
r3', 'marwre',
        'mastro1', 'meapar', 'melbla1', 'monoro1', 'mouchi', 'mou
dov', 'mouela1',
        'mouqua', 'mouwar', 'mutswa', 'naswar', 'norcar', 'norfli
', 'normoc', 'norpar',
        'norsho', 'norwat', 'nrwswa', 'nutwoo', 'oaktit', 'obnthr
1', 'ocbfly1',
        'oliwoo1', 'olsfly', 'orbeup1', 'orbspa1', 'orcpar', 'orc
war', 'orfpar',
        'osprey', 'ovenbi1', 'pabspi1', 'paltan1', 'palwar', 'pas
fly', 'pavpig2',
        'phivir', 'pibgre', 'pilwoo', 'pinsis', 'pirfly1', 'plawr
e1', 'plaxen1',
        'plsvir', 'plupig2', 'prowar', 'purfin', 'purgal2', 'putf
ru1', 'pygnut',
        'rawwre1', 'rcatan1', 'rebnut', 'rebsap', 'rebwoo', 'redc
ro', 'reevir1',
        'rehbar1', 'relpar', 'reshaw', 'rethaw', 'rewbla', 'ribgu
l', 'rinkin1',
        'roahaw', 'robgro', 'rocpig', 'rotbec', 'royter1', 'rthhu
m', 'rtlhum',
        'ruboro1', 'rubpep1', 'rubrob', 'rubwre1', 'ruckin', 'ruc
spa1', 'rucwar',
        'rucwar1', 'rudpig', 'rudtur', 'rufhum', 'rugdov', 'rumfl
y1', 'runwre1',
        'rutjac1', 'saffin', 'sancra', 'sander', 'savspa', 'sayph
o', 'scamac1',
        'scatan', 'scbwre1', 'scptyr1', 'scrtan1', 'semplo', 'shi
cow', 'sibtan2',
        'sinwre1', 'sltred', 'smbani', 'snogoo', 'sobtyr1', 'socf
ly1', 'solsan',
        'sonspa', 'soulap1', 'sposan', 'spotow', 'spvear1', 'squc
uc1', 'stbori',
        'stejay', 'sthant1', 'sthwoo1', 'strcuc1', 'strfly1', 'st
rsal1', 'stvhum2',
        'subfly', 'sumtan', 'swaspa', 'swathr', 'tenwar', 'thbeup
1', 'thbkin',
        'thswar1', 'towsol', 'treswa', 'trogna1', 'trokin', 'trom
oc', 'tropar',
        'tropew1', 'tuftit', 'tunswa', 'veery', 'verdin', 'vigswa
', 'warvir',
        'wbwwre1', 'webwoo1', 'wegspa1', 'wesant1', 'wesblu', 'we
skin', 'wesmea',
        'westan', 'wewpew', 'whbman1', 'whbnut', 'whcpar', 'whcse
e1', 'whcspa',
        'whevir', 'whfpar1', 'whimbr', 'whiwre1', 'whtdov', 'whts
pa', 'whwbec1',
        'whwdov', 'wilfly', 'willet1', 'wilsni1', 'wiltur', 'wlsw
ar', 'wooduc',
        'woothr', 'wrenti', 'y00475', 'yebcha', 'yebela1', 'yebfl
y', 'yebori1',
        'yebsap', 'yebsee1', 'yefgra1', 'yegvir', 'yehbla', 'yehc
ar1', 'yelgro',
        'yelwar', 'yeofly1', 'yerwar', 'yeteup1', 'yetvir']

    ######################
    # Loaders #
    ######################
    loader_params = {
        "train": {
            "batch_size": 64,
            "num_workers": 20,
            "shuffle": True
        },
        "valid": {
            "batch_size": 64,
            "num_workers": 20,
            "shuffle": False
        },
        "test": {
            "batch_size": 64,
            "num_workers": 20,
            "shuffle": False
        }
    }
    ######################
    # Split #
    ######################
    split = "StratifiedKFold"
    split_params = {
        "n_splits": 5,
        "shuffle": True,
        "random_state": 1213
    }

    ######################
    # Model #
    ######################
    base_model_name = "tf_efficientnet_b0_ns"
    pooling = "max"
    pretrained = True
    num_classes = 397
    in_channels = 1

    ######################
    # Criterion #
    ######################
    loss_name = "BCEFocal2WayLoss"
    loss_params: dict = {}

    ######################
    # Optimizer #
    ######################
    optimizer_name = "Adam"
    base_optimizer = "Adam"
    optimizer_params = {
        "lr": 0.001
    }
    # For SAM optimizer
    base_optimizer = "Adam"

    ######################
    # Scheduler #
    ######################
    scheduler_name = "CosineAnnealingLR"
    scheduler_params = {
        "T_max": 10
    }
```

Data Loading

In [6]:
```
TARGET_SR = 32000
TEST = (len(list(Path("../input/birdclef-2021/test_soundscapes
/").glob("*.ogg"))) != 0)
if TEST:
    DATADIR = Path("../input/birdclef-2021/test_soundscapes/")
else:
    DATADIR = Path("../input/birdclef-2021/train_soundscapes/")
```

In [7]:
```
all_audios = list(DATADIR.glob("*.ogg"))
all_audio_ids = ["_".join(audio_id.name.split("_")[:2]) for audio
_id in all_audios]
submission_df = pd.DataFrame({
    "row_id": all_audio_ids
})
submission_df
```

Out[7]:

	row_id
0	20152_SSW
1	57610_COR
2	7843_SSW
3	42907_SSW
4	7019_COR
5	54955_SSW
6	10534_SSW
7	2782_SSW
8	11254_COR
9	7954_COR
10	26746_COR
11	18003_COR
12	31928_COR
13	51010_SSW
14	21767_COR
15	14473_SSW
16	44957_COR
17	50878_COR
18	28933_SSW
19	26709_SSW

Define Model

In [8]:
```
def init_layer(layer):
    nn.init.xavier_uniform_(layer.weight)

    if hasattr(layer, "bias"):
        if layer.bias is not None:
            layer.bias.data.fill_(0.)


def init_bn(bn):
    bn.bias.data.fill_(0.)
    bn.weight.data.fill_(1.0)


def init_weights(model):
    classname = model.__class__.__name__
    if classname.find("Conv2d") != -1:
        nn.init.xavier_uniform_(model.weight, gain=np.sqrt(2))
        model.bias.data.fill_(0)
    elif classname.find("BatchNorm") != -1:
        model.weight.data.normal_(1.0, 0.02)
        model.bias.data.fill_(0)
    elif classname.find("GRU") != -1:
        for weight in model.parameters():
            if len(weight.size()) > 1:
                nn.init.orghogonal_(weight.data)
    elif classname.find("Linear") != -1:
        model.weight.data.normal_(0, 0.01)
        model.bias.data.zero_()


def do_mixup(x: torch.Tensor, mixup_lambda: torch.Tensor):
    """Mixup x of even indexes (0, 2, 4, ...) with x of odd index
es
    (1, 3, 5, ...).
    Args:
      x: (batch_size * 2, ...)
      mixup_lambda: (batch_size * 2,)
    Returns:
      out: (batch_size, ...)
    """
    out = (x[0::2].transpose(0, -1) * mixup_lambda[0::2] +
           x[1::2].transpose(0, -1) * mixup_lambda[1::2]).transpo
se(0, -1)
    return out


class Mixup(object):
    def __init__(self, mixup_alpha, random_seed=1234):
        """Mixup coefficient generator.
        """
        self.mixup_alpha = mixup_alpha
        self.random_state = np.random.RandomState(random_seed)

    def get_lambda(self, batch_size):
        """Get mixup random coefficients.
        Args:
          batch_size: int
        Returns:
          mixup_lambdas: (batch_size,)
        """
        mixup_lambdas = []
        for n in range(0, batch_size, 2):
            lam = self.random_state.beta(
                self.mixup_alpha, self.mixup_alpha, 1)[0]
            mixup_lambdas.append(lam)
            mixup_lambdas.append(1. - lam)

        return torch.from_numpy(np.array(mixup_lambdas, dtype=np.
float32))


def interpolate(x: torch.Tensor, ratio: int):
    """Interpolate data in time domain. This is used to compensat
e the
    resolution reduction in downsampling of a CNN.
    Args:
      x: (batch_size, time_steps, classes_num)
      ratio: int, ratio to interpolate
    Returns:
      upsampled: (batch_size, time_steps * ratio, classes_num)
    """
    (batch_size, time_steps, classes_num) = x.shape
    upsampled = x[:, :, None, :].repeat(1, 1, ratio, 1)
    upsampled = upsampled.reshape(batch_size, time_steps * ratio,
classes_num)
    return upsampled


def pad_framewise_output(framewise_output: torch.Tensor, frames_n
um: int):
    """Pad framewise_output to the same length as input frames. T
he pad value
    is the same as the value of the last frame.
    Args:
      framewise_output: (batch_size, frames_num, classes_num)
      frames_num: int, number of frames to pad
    Outputs:
      output: (batch_size, frames_num, classes_num)
    """
    output = F.interpolate(
        framewise_output.unsqueeze(1),
        size=(frames_num, framewise_output.size(2)),
        align_corners=True,
        mode="bilinear").squeeze(1)

    return output


def gem(x: torch.Tensor, p=3, eps=1e-6):
    return F.avg_pool2d(x.clamp(min=eps).pow(p), (x.size(-2), x.s
ize(-1))).pow(1. / p)


class GeM(nn.Module):
    def __init__(self, p=3, eps=1e-6):
        super().__init__()
        self.p = nn.Parameter(torch.ones(1) * p)
        self.eps = eps
    def forward(self, x):
        return gem(x, p=self.p, eps=self.eps)

    def __repr__(self):
        return self.__class__.__name__ + f"(p={self.p.data.tolist
()[0]:.4f}, eps={self.eps})"


class AttBlockV2(nn.Module):
    def __init__(self,
                 in_features: int,
                 out_features: int,
                 activation="linear"):
        super().__init__()

        self.activation = activation
        self.att = nn.Conv1d(
            in_channels=in_features,
            out_channels=out_features,
            kernel_size=1,
            stride=1,
            padding=0,
            bias=True)
        self.cla = nn.Conv1d(
            in_channels=in_features,
            out_channels=out_features,
            kernel_size=1,
            stride=1,
            padding=0,
            bias=True)

        self.init_weights()

    def init_weights(self):
        init_layer(self.att)
        init_layer(self.cla)

    def forward(self, x):
        # x: (n_samples, n_in, n_time)
        norm_att = torch.softmax(torch.tanh(self.att(x)), dim=-1)
        cla = self.nonlinear_transform(self.cla(x))
        x = torch.sum(norm_att * cla, dim=2)
        return x, norm_att, cla

    def nonlinear_transform(self, x):
        if self.activation == 'linear':
            return x
        elif self.activation == 'sigmoid':
            return torch.sigmoid(x)


class TimmSED(nn.Module):
    def __init__(self, base_model_name: str, pretrained=False, nu
m_classes=24, in_channels=1):
        super().__init__()
        # Spectrogram extractor
        self.spectrogram_extractor = Spectrogram(n_fft=CFG.n_fft,
hop_length=CFG.hop_length,
                                                 win_length=CFG.n
_fft, window="hann", center=True, pad_mode="reflect",
                                                 freeze_parameter
s=True)

        # Logmel feature extractor
        self.logmel_extractor = LogmelFilterBank(sr=CFG.sample_ra
te, n_fft=CFG.n_fft,
                                                 n_mels=CFG.n_mel
s, fmin=CFG.fmin, fmax=CFG.fmax, ref=1.0, amin=1e-10, top_db=Non
e,
                                                 freeze_parameter
s=True)

        # Spec augmenter
        self.spec_augmenter = SpecAugmentation(time_drop_width=6
4, time_stripes_num=2,
                                               freq_drop_width=8,
freq_stripes_num=2)

        self.bn0 = nn.BatchNorm2d(CFG.n_mels)

        base_model = timm.create_model(
            base_model_name, pretrained=pretrained, in_chans=in_c
hannels)
        layers = list(base_model.children())[:-2]
        self.encoder = nn.Sequential(*layers)

        if hasattr(base_model, "fc"):
            in_features = base_model.fc.in_features
        else:
            in_features = base_model.classifier.in_features
        self.fc1 = nn.Linear(in_features, in_features, bias=True)
        self.att_block = AttBlockV2(
            in_features, num_classes, activation="sigmoid")

        self.init_weight()

    def init_weight(self):
        init_layer(self.fc1)
        init_bn(self.bn0)

    def forward(self, input):
        # (batch_size, 1, time_steps, freq_bins)
        x = self.spectrogram_extractor(input)
        x = self.logmel_extractor(x)    # (batch_size, 1, time_st
eps, mel_bins)

        frames_num = x.shape[2]

        x = x.transpose(1, 3)
        x = self.bn0(x)
        x = x.transpose(1, 3)

        if self.training:
            x = self.spec_augmenter(x)

        x = x.transpose(2, 3)
        # (batch_size, channels, freq, frames)
        x = self.encoder(x)

        # (batch_size, channels, frames)
        x = torch.mean(x, dim=2)

        # channel smoothing
        x1 = F.max_pool1d(x, kernel_size=3, stride=1, padding=1)
        x2 = F.avg_pool1d(x, kernel_size=3, stride=1, padding=1)
        x = x1 + x2

        x = F.dropout(x, p=0.5, training=self.training)
        x = x.transpose(1, 2)
        x = F.relu_(self.fc1(x))
        x = x.transpose(1, 2)
        x = F.dropout(x, p=0.5, training=self.training)
        (clipwise_output, norm_att, segmentwise_output) = self.at
t_block(x)
        logit = torch.sum(norm_att * self.att_block.cla(x), dim=
2)
        segmentwise_logit = self.att_block.cla(x).transpose(1, 2)
        segmentwise_output = segmentwise_output.transpose(1, 2)

        interpolate_ratio = frames_num // segmentwise_output.size
(1)

        # Get framewise output
        framewise_output = interpolate(segmentwise_output,
                                       interpolate_ratio)
        framewise_output = pad_framewise_output(framewise_output,
frames_num)

        framewise_logit = interpolate(segmentwise_logit, interpol
ate_ratio)
        framewise_logit = pad_framewise_output(framewise_logit, f
rames_num)

        output_dict = {
            "framewise_output": framewise_output,
            "segmentwise_output": segmentwise_output,
            "logit": logit,
            "framewise_logit": framewise_logit,
            "clipwise_output": clipwise_output
        }

        return output_dict
```

Dataset

In [9]:
```
class TestDataset(torchdata.Dataset):
    def __init__(self, df: pd.DataFrame, clip: np.ndarray,
                 waveform_transforms=None):
        self.df = df
        self.clip = clip
        self.waveform_transforms=waveform_transforms

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx: int):
        SR = 32000
        sample = self.df.loc[idx, :]
        row_id = sample.row_id

        end_seconds = int(sample.seconds)
        start_seconds = int(end_seconds - 5)

        start_index = SR * start_seconds
        end_index = SR * end_seconds

        y = self.clip[start_index:end_index].astype(np.float32)

        y = np.nan_to_num(y)

        if self.waveform_transforms:
            y = self.waveform_transforms(y)

        y = np.nan_to_num(y)

        return y, row_id
```

In [10]:
```
def get_transforms(phase: str):
    transforms = CFG.transforms
    if transforms is None:
        return None
    else:
        if transforms[phase] is None:
            return None
        trns_list = []
        for trns_conf in transforms[phase]:
            trns_name = trns_conf["name"]
            trns_params = {} if trns_conf.get("params") is None e
lse \
                trns_conf["params"]
            if globals().get(trns_name) is not None:
                trns_cls = globals()[trns_name]
                trns_list.append(trns_cls(**trns_params))

        if len(trns_list) > 0:
            return Compose(trns_list)
        else:
            return None


def get_waveform_transforms(config: dict, phase: str):
    return get_transforms(config, phase)


def get_spectrogram_transforms(config: dict, phase: str):
    transforms = config.get('spectrogram_transforms')
    if transforms is None:
        return None
    else:
        if transforms[phase] is None:
            return None
        trns_list = []
        for trns_conf in transforms[phase]:
            trns_name = trns_conf["name"]
            trns_params = {} if trns_conf.get("params") is None e
lse \
                trns_conf["params"]
            if hasattr(A, trns_name):
                trns_cls = A.__getattribute__(trns_name)
                trns_list.append(trns_cls(**trns_params))
            else:
                trns_cls = globals().get(trns_name)
                if trns_cls is not None:
                    trns_list.append(trns_cls(**trns_params))

        if len(trns_list) > 0:
            return A.Compose(trns_list, p=1.0)
        else:
            return None


class Normalize:
    def __call__(self, y: np.ndarray):
        max_vol = np.abs(y).max()
        y_vol = y * 1 / max_vol
        return np.asfortranarray(y_vol)
```

```
class NewNormalize:
    def __call__(self, y: np.ndarray):
        y_mm = y - y.mean()
        return y_mm / y_mm.abs().max()


class Compose:
    def __init__(self, transforms: list):
        self.transforms = transforms

    def __call__(self, y: np.ndarray):
        for trns in self.transforms:
            y = trns(y)
        return y


class AudioTransform:
    def __init__(self, always_apply=False, p=0.5):
        self.always_apply = always_apply
        self.p = p

    def __call__(self, y: np.ndarray):
        if self.always_apply:
            return self.apply(y)
        else:
            if np.random.rand() < self.p:
                return self.apply(y)
            else:
                return y

    def apply(self, y: np.ndarray):
        raise NotImplementedError


class NoiseInjection(AudioTransform):
    def __init__(self, always_apply=False, p=0.5, max_noise_level
=0.5, sr=32000):
        super().__init__(always_apply, p)

        self.noise_level = (0.0, max_noise_level)
        self.sr = sr

    def apply(self, y: np.ndarray, **params):
        noise_level = np.random.uniform(*self.noise_level)
        noise = np.random.randn(len(y))
        augmented = (y + noise * noise_level).astype(y.dtype)
        return augmented


class GaussianNoise(AudioTransform):
    def __init__(self, always_apply=False, p=0.5, min_snr=5, max_
snr=20, sr=32000):
        super().__init__(always_apply, p)

        self.min_snr = min_snr
        self.max_snr = max_snr
        self.sr = sr
    def apply(self, y: np.ndarray, **params):
        snr = np.random.uniform(self.min_snr, self.max_snr)
        a_signal = np.sqrt(y ** 2).max()
        a_noise = a_signal / (10 ** (snr / 20))

        white_noise = np.random.randn(len(y))
        a_white = np.sqrt(white_noise ** 2).max()
        augmented = (y + white_noise * 1 / a_white * a_noise).ast
ype(y.dtype)
        return augmented


class PinkNoise(AudioTransform):
    def __init__(self, always_apply=False, p=0.5, min_snr=5, max_
snr=20, sr=32000):
        super().__init__(always_apply, p)

        self.min_snr = min_snr
        self.max_snr = max_snr
        self.sr = sr

    def apply(self, y: np.ndarray, **params):
        snr = np.random.uniform(self.min_snr, self.max_snr)
        a_signal = np.sqrt(y ** 2).max()
        a_noise = a_signal / (10 ** (snr / 20))

        pink_noise = cn.powerlaw_psd_gaussian(1, len(y))
        a_pink = np.sqrt(pink_noise ** 2).max()
        augmented = (y + pink_noise * 1 / a_pink * a_noise).astyp
e(y.dtype)
        return augmented


class PitchShift(AudioTransform):
    def __init__(self, always_apply=False, p=0.5, max_range=5, sr
=32000):
        super().__init__(always_apply, p)
        self.max_range = max_range
        self.sr = sr

    def apply(self, y: np.ndarray, **params):
        n_steps = np.random.randint(-self.max_range, self.max_ran
ge)
        augmented = librosa.effects.pitch_shift(y, self.sr, n_ste
ps)
        return augmented


class TimeStretch(AudioTransform):
    def __init__(self, always_apply=False, p=0.5, max_rate=1, sr=
32000):
        super().__init__(always_apply, p)
        self.max_rate = max_rate
        self.sr = sr

    def apply(self, y: np.ndarray, **params):
        rate = np.random.uniform(0, self.max_rate)
        augmented = librosa.effects.time_stretch(y, rate)
        return augmented
```

```
def _db2float(db: float, amplitude=True):
    if amplitude:
        return 10**(db / 20)
    else:
        return 10 ** (db / 10)


def volume_down(y: np.ndarray, db: float):
    """
    Low level API for decreasing the volume
    Parameters
    ----------
    y: numpy.ndarray
        stereo / monaural input audio
    db: float
        how much decibel to decrease
    Returns
    -------
    applied: numpy.ndarray
        audio with decreased volume
    """
    applied = y * _db2float(-db)
    return applied


def volume_up(y: np.ndarray, db: float):
    """
    Low level API for increasing the volume
    Parameters
    ----------
    y: numpy.ndarray
        stereo / monaural input audio
    db: float
        how much decibel to increase
    Returns
    -------
    applied: numpy.ndarray
        audio with increased volume
    """
    applied = y * _db2float(db)
    return applied


class RandomVolume(AudioTransform):
    def __init__(self, always_apply=False, p=0.5, limit=10):
        super().__init__(always_apply, p)
        self.limit = limit

    def apply(self, y: np.ndarray, **params):
        db = np.random.uniform(-self.limit, self.limit)
        if db >= 0:
            return volume_up(y, db)
        else:
            return volume_down(y, db)


class OneOf:
    def __init__(self, transforms: list):
        self.transforms = transforms
    def __call__(self, y: np.ndarray):
        n_trns = len(self.transforms)
        trns_idx = np.random.choice(n_trns)
        trns = self.transforms[trns_idx]
        y = trns(y)
        return y


class CosineVolume(AudioTransform):
    def __init__(self, always_apply=False, p=0.5, limit=10):
        super().__init__(always_apply, p)
        self.limit = limit

    def apply(self, y: np.ndarray, **params):
        db = np.random.uniform(-self.limit, self.limit)
        cosine = np.cos(np.arange(len(y)) / len(y) * np.pi * 2)
        dbs = _db2float(cosine * db)
        return y * dbs


def drop_stripes(image: np.ndarray, dim: int, drop_width: int, st
ripes_num: int):
    total_width = image.shape[dim]
    lowest_value = image.min()
    for _ in range(stripes_num):
        distance = np.random.randint(low=0, high=drop_width, size
=(1,))[0]
        begin = np.random.randint(
            low=0, high=total_width - distance, size=(1,))[0]

        if dim == 0:
            image[begin:begin + distance] = lowest_value
        elif dim == 1:
            image[:, begin + distance] = lowest_value
        elif dim == 2:
            image[:, :, begin + distance] = lowest_value
    return image


class TimeFreqMasking(ImageOnlyTransform):
    def __init__(self,
                 time_drop_width: int,
                 time_stripes_num: int,
                 freq_drop_width: int,
                 freq_stripes_num: int,
                 always_apply=False,
                 p=0.5):
        super().__init__(always_apply, p)
        self.time_drop_width = time_drop_width
        self.time_stripes_num = time_stripes_num
        self.freq_drop_width = freq_drop_width
        self.freq_stripes_num = freq_stripes_num

    def apply(self, img, **params):
        img_ = img.copy()
        if img.ndim == 2:
            img_ = drop_stripes(
                img_, dim=0, drop_width=self.freq_drop_width, str
ipes_num=self.freq_stripes_num)
```

```
            img_ = drop_stripes(
                img_, dim=1, drop_width=self.time_drop_width, str
ipes_num=self.time_stripes_num)
        return img_
```

Get model

In [11]:
```
def prepare_model_for_inference(model, path: Path):
    if not torch.cuda.is_available():
        ckpt = torch.load(path, map_location="cpu")
    else:
        ckpt = torch.load(path)
    model.load_state_dict(ckpt["model_state_dict"])
    model.eval()
    return model
```

In [12]:
```
def prediction_for_clip(test_df: pd.DataFrame,
                        clip: np.ndarray,
                        model,
                        threshold=0.5):

    dataset = TestDataset(df=test_df,
                          clip=clip,
                          waveform_transforms=get_transforms(phas
e="test"))
    loader = torchdata.DataLoader(dataset, batch_size=1, shuffle=
False)
    device = torch.device("cuda" if torch.cuda.is_available() els
e "cpu")

    model.eval()
    prediction_dict = {}
    for image, row_id in tqdm(loader):
        row_id = row_id[0]
        image = image.to(device)

        with torch.no_grad():
            prediction = model(image)
            proba = prediction["clipwise_output"].detach().cpu().
numpy().reshape(-1)

        events = proba >= threshold
        labels = np.argwhere(events).reshape(-1).tolist()

        if len(labels) == 0:
            prediction_dict[row_id] = "nocall"
        else:
            labels_str_list = list(map(lambda x: CFG.target_colum
ns[x], labels))
            label_string = " ".join(labels_str_list)
            prediction_dict[row_id] = label_string
    return prediction_dict
```

In [13]:
```
def prediction(test_audios,
               weights_path: Path,
               threshold=0.5):
    device = torch.device("cuda" if torch.cuda.is_available() els
e "cpu")
    model = TimmSED(base_model_name=CFG.base_model_name,
                    pretrained=False,
                    num_classes=CFG.num_classes,
                    in_channels=CFG.in_channels)
    model = prepare_model_for_inference(model, weights_path).to(d
evice)

    warnings.filterwarnings("ignore")
    prediction_dfs = []
    for audio_path in test_audios:
        with timer(f"Loading {str(audio_path)}", logger):
            clip, _ = sf.read(audio_path)

        seconds = []
        row_ids = []
        for second in range(5, 605, 5):
            row_id = "_".join(audio_path.name.split("_")[:2]) +
f"_{second}"
            seconds.append(second)
            row_ids.append(row_id)

        test_df = pd.DataFrame({
            "row_id": row_ids,
            "seconds": seconds
        })
        with timer(f"Prediction on {audio_path}", logger):
            prediction_dict = prediction_for_clip(test_df,
                                                  clip=clip,
                                                  model=model,
                                                  threshold=thres
hold)
        row_id = list(prediction_dict.keys())
        birds = list(prediction_dict.values())
        prediction_df = pd.DataFrame({
            "row_id": row_id,
            "birds": birds
        })
        prediction_dfs.append(prediction_df)

    prediction_df = pd.concat(prediction_dfs, axis=0, sort=Fals
e).reset_index(drop=True)
    return prediction_df
```

Prediction

In [14]:
```
weights_path = Path("../input/birdclef2021-effnetb0-starter-weigh
t/best.pth")
submission = prediction(test_audios=all_audios,
                        weights_path=weights_path,
                        threshold=0.5)
submission.to_csv("submission.csv", index=False)
```

```
2021-04-04 00:58:15,279 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/20152_SSW_20170805.ogg] start
2021-04-04 00:58:16,032 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/20152_SSW_20170805.ogg] done in 0.75 s
2021-04-04 00:58:16,034 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/20152_SSW_20170805.ogg] start
100%|██████████| 120/120 [00:02<00:00, 46.55it/s]
2021-04-04 00:58:18,619 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/20152_SSW_20170805.ogg] done in 2.59 s
2021-04-04 00:58:18,622 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/57610_COR_20190904.ogg] start
2021-04-04 00:58:19,385 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/57610_COR_20190904.ogg] done in 0.76 s
2021-04-04 00:58:19,387 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/57610_COR_20190904.ogg] start
100%|██████████| 120/120 [00:01<00:00, 77.80it/s]
2021-04-04 00:58:20,935 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/57610_COR_20190904.ogg] done in 1.55 s
2021-04-04 00:58:20,937 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/7843_SSW_20170325.ogg] start
2021-04-04 00:58:22,359 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/7843_SSW_20170325.ogg] done in 1.42 s
2021-04-04 00:58:22,365 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/7843_SSW_20170325.ogg] start
100%|██████████| 120/120 [00:01<00:00, 70.23it/s]
2021-04-04 00:58:24,083 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/7843_SSW_20170325.ogg] done in 1.72 s
2021-04-04 00:58:24,085 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/42907_SSW_20170708.ogg] start
2021-04-04 00:58:24,896 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/42907_SSW_20170708.ogg] done in 0.81 s
2021-04-04 00:58:24,898 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/42907_SSW_20170708.ogg] start
100%|██████████| 120/120 [00:01<00:00, 63.72it/s]
2021-04-04 00:58:26,786 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/42907_SSW_20170708.ogg] done in 1.89 s
2021-04-04 00:58:26,788 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/7019_COR_20190904.ogg] start
2021-04-04 00:58:27,628 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/7019_COR_20190904.ogg] done in 0.84 s
2021-04-04 00:58:27,631 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/7019_COR_20190904.ogg] start
100%|██████████| 120/120 [00:01<00:00, 74.59it/s]
2021-04-04 00:58:29,246 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/7019_COR_20190904.ogg] done in 1.62 s
2021-04-04 00:58:29,248 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/54955_SSW_20170617.ogg] start
2021-04-04 00:58:30,066 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/54955_SSW_20170617.ogg] done in 0.82 s
2021-04-04 00:58:30,068 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/54955_SSW_20170617.ogg] start
100%|██████████| 120/120 [00:01<00:00, 71.07it/s]
2021-04-04 00:58:31,762 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/54955_SSW_20170617.ogg] done in 1.69 s
2021-04-04 00:58:31,764 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/10534_SSW_20170429.ogg] start
2021-04-04 00:58:32,547 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/10534_SSW_20170429.ogg] done in 0.78 s
2021-04-04 00:58:32,550 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/10534_SSW_20170429.ogg] start
100%|██████████| 120/120 [00:01<00:00, 77.36it/s]
2021-04-04 00:58:34,107 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/10534_SSW_20170429.ogg] done in 1.56 s
2021-04-04 00:58:34,108 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/2782_SSW_20170701.ogg] start
2021-04-04 00:58:35,021 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/2782_SSW_20170701.ogg] done in 0.91 s
2021-04-04 00:58:35,023 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/2782_SSW_20170701.ogg] start
100%|██████████| 120/120 [00:01<00:00, 66.81it/s]
2021-04-04 00:58:36,824 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/2782_SSW_20170701.ogg] done in 1.80 s
2021-04-04 00:58:36,826 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/11254_COR_20190904.ogg] start
2021-04-04 00:58:37,637 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/11254_COR_20190904.ogg] done in 0.81 s
2021-04-04 00:58:37,640 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/11254_COR_20190904.ogg] start
100%|██████████| 120/120 [00:01<00:00, 72.42it/s]
2021-04-04 00:58:39,302 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/11254_COR_20190904.ogg] done in 1.66 s
2021-04-04 00:58:39,304 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/7954_COR_20190923.ogg] start
2021-04-04 00:58:40,148 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/7954_COR_20190923.ogg] done in 0.84 s
2021-04-04 00:58:40,151 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/7954_COR_20190923.ogg] start
100%|██████████| 120/120 [00:01<00:00, 71.78it/s]
2021-04-04 00:58:41,827 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/7954_COR_20190923.ogg] done in 1.68 s
2021-04-04 00:58:41,829 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/26746_COR_20191004.ogg] start
2021-04-04 00:58:42,682 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/26746_COR_20191004.ogg] done in 0.85 s
2021-04-04 00:58:42,686 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/26746_COR_20191004.ogg] start
100%|██████████| 120/120 [00:01<00:00, 71.57it/s]
2021-04-04 00:58:44,366 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/26746_COR_20191004.ogg] done in 1.68 s
2021-04-04 00:58:44,368 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/18003_COR_20190904.ogg] start
2021-04-04 00:58:45,156 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/18003_COR_20190904.ogg] done in 0.79 s
2021-04-04 00:58:45,158 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/18003_COR_20190904.ogg] start
100%|██████████| 120/120 [00:01<00:00, 72.46it/s]
2021-04-04 00:58:46,819 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/18003_COR_20190904.ogg] done in 1.66 s
2021-04-04 00:58:46,821 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/31928_COR_20191004.ogg] start
2021-04-04 00:58:47,723 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/31928_COR_20191004.ogg] done in 0.90 s
2021-04-04 00:58:47,729 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/31928_COR_20191004.ogg] start
100%|██████████| 120/120 [00:01<00:00, 71.37it/s]
2021-04-04 00:58:49,419 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/31928_COR_20191004.ogg] done in 1.69 s
2021-04-04 00:58:49,421 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/51010_SSW_20170513.ogg] start
```

```
2021-04-04 00:58:50,191 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/51010_SSW_20170513.ogg] done in 0.77 s
2021-04-04 00:58:50,193 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/51010_SSW_20170513.ogg] start
100%|██████████| 120/120 [00:01<00:00, 74.35it/s]
2021-04-04 00:58:51,812 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/51010_SSW_20170513.ogg] done in 1.62 s
2021-04-04 00:58:51,815 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/21767_COR_20190904.ogg] start
2021-04-04 00:58:52,642 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/21767_COR_20190904.ogg] done in 0.83 s
2021-04-04 00:58:52,644 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/21767_COR_20190904.ogg] start
100%|██████████| 120/120 [00:02<00:00, 56.74it/s]
2021-04-04 00:58:54,764 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/21767_COR_20190904.ogg] done in 2.12 s
2021-04-04 00:58:54,769 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/14473_SSW_20170701.ogg] start
2021-04-04 00:58:55,654 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/14473_SSW_20170701.ogg] done in 0.89 s
2021-04-04 00:58:55,657 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/14473_SSW_20170701.ogg] start
100%|██████████| 120/120 [00:01<00:00, 67.75it/s]
2021-04-04 00:58:57,565 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/14473_SSW_20170701.ogg] done in 1.91 s
2021-04-04 00:58:57,567 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/44957_COR_20190923.ogg] start
2021-04-04 00:58:58,308 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/44957_COR_20190923.ogg] done in 0.74 s
2021-04-04 00:58:58,310 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/44957_COR_20190923.ogg] start
100%|██████████| 120/120 [00:01<00:00, 64.23it/s]
2021-04-04 00:59:00,184 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/44957_COR_20190923.ogg] done in 1.87 s
2021-04-04 00:59:00,186 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/50878_COR_20191004.ogg] start
2021-04-04 00:59:00,979 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/50878_COR_20191004.ogg] done in 0.79 s
2021-04-04 00:59:00,981 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/50878_COR_20191004.ogg] start
100%|██████████| 120/120 [00:01<00:00, 72.69it/s]
2021-04-04 00:59:02,636 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/50878_COR_20191004.ogg] done in 1.66 s
2021-04-04 00:59:02,638 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/28933_SSW_20170408.ogg] start
2021-04-04 00:59:03,571 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/28933_SSW_20170408.ogg] done in 0.93 s
2021-04-04 00:59:03,574 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/28933_SSW_20170408.ogg] start
100%|██████████| 120/120 [00:01<00:00, 73.45it/s]
2021-04-04 00:59:05,214 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/28933_SSW_20170408.ogg] done in 1.64 s
2021-04-04 00:59:05,216 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/26709_SSW_20170701.ogg] start
2021-04-04 00:59:06,028 - INFO - [Loading ../input/birdclef-2021/
train_soundscapes/26709_SSW_20170701.ogg] done in 0.81 s
2021-04-04 00:59:06,030 - INFO - [Prediction on ../input/birdclef
-2021/train_soundscapes/26709_SSW_20170701.ogg] start
100%|██████████| 120/120 [00:01<00:00, 73.27it/s]
2021-04-04 00:59:07,673 - INFO - [Prediction on ../input/birdclef
```

In [15]:
```python
pd.read_csv("submission.csv")
```

Out[15]:
```
             row_id   birds
   0     20152_SSW_5  nocall
   1    20152_SSW_10  nocall
   2    20152_SSW_15  nocall
   3    20152_SSW_20  nocall
   4    20152_SSW_25  nocall
 ...             ...     ...
2395   26709_SSW_580  nocall
2396   26709_SSW_585  nocall
2397   26709_SSW_590  nocall
2398   26709_SSW_595  nocall
2399   26709_SSW_600  nocall

2400 rows × 2 columns
```

In [ ]:
```python

```