

# Freesound Audio Taggin 2019 - Solution Part 1

This is based upon

- <https://www.kaggle.com/daisukelab/cnn-2d-basic-solution-powered-by-fast-ai> (<https://www.kaggle.com/daisukelab/cnn-2d-basic-solution-powered-by-fast-ai>)
- Based on the 2nd version. <https://www.kaggle.com/daisukelab/clf-to-multi-cnn-2d-basic-2-preprocessed-dataset> (<https://www.kaggle.com/daisukelab/clf-to-multi-cnn-2d-basic-2-preprocessed-dataset>)
- Borrowing model from <https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch> (<https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch>)
- Using preprocessed dataset. [https://www.kaggle.com/daisukelab/fat2019\\_prep\\_mels1](https://www.kaggle.com/daisukelab/fat2019_prep_mels1) ([https://www.kaggle.com/daisukelab/fat2019\\_prep\\_mels1](https://www.kaggle.com/daisukelab/fat2019_prep_mels1))

Please read README.md for more information like SpecMix...

```
In [1]: import os
from pathlib import Path
import pickle
import random
import warnings

import numpy as np
import pandas as pd
from scipy.stats import rankdata
from sklearn.model_selection import KFold
import librosa
import librosa.display
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
import IPython
import IPython.display
import PIL
import seaborn as sns

import torch
import torch.nn as nn
import torch.nn.functional as F

from fastai import *
from fastai.vision import *
from fastai.vision.data import *
from fastai.callbacks import *
```

```
In [2]: def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True

SEED = 2019
seed_everything(SEED)
```

```
In [3]: TOY_MODE = False
TTA_SHIFT = 32 # TTA: predict every TTA_SHIFT
bs=128
n_splits = 10

In [4]: DATA = Path('../input')
PREPROCESSED = Path('/media/eric/Data/freesound-audio-tagging-2019/preprocessed')
WORK = Path('work')
Path(WORK).mkdir(exist_ok=True, parents=True)
Path(PREPROCESSED).mkdir(exist_ok=True, parents=True)

CSV_TRN_CURATED = DATA/'train_curated.csv'
CSV_TRN_NOISY = DATA/'train_noisy.csv'
CSV_SUBMISSION = DATA/'sample_submission.csv'

MELS_TRN_CURATED = PREPROCESSED/'mels_train_curated.pkl'
MELS_TRN_NOISY = PREPROCESSED/'mels_train_noisy.pkl'
MELS_TEST = PREPROCESSED/'mels_test.pkl'

DATA_CURATED = DATA/'train_curated'
DATA_NOISY = DATA/'train_noisy'
DATA_TEST = DATA/'test'

trn_curated_df = pd.read_csv(CSV_TRN_CURATED)
trn_noisy_df = pd.read_csv(CSV_TRN_NOISY)
test_df = pd.read_csv(CSV_SUBMISSION)

In [83]: # # Drop samples with incorrect label
# # Discussion from organizers: https://www.kaggle.com/c/freesound-audio-tagging-2019/discussion/93480
useless = ['f76181c4.wav', '77b925c2.wav', '6a1f682a.wav', 'c7db12aa.wav', '7752cc8a.wav', '1d44b0bd.wav']
# trn_curated_df = trn_curated_df[~trn_curated_df.fname.isin(useless)]
```

## Audio conversion to 2D

Almost copied from my repository: <https://github.com/daisukelab/ml-sound-classifier> (<https://github.com/daisukelab/ml-sound-classifier>)

Handle sampling rate 44.1kHz as is, no information loss.  
Size of each file will be 128 x L, L is audio seconds x 128; [128, 256] if sound is 2 s long.  
Convert to Mel-spectrogram, not MFCC. We are handling general sound rather than human voice. <https://en.wikipedia.org/wiki/Spectrogram>

```
In [6]: def read_audio(conf, pathname, trim_long_data):
    y, sr = librosa.load(pathname, sr=conf.sampling_rate)
    # trim silence
    if 0 < len(y): # workaround: 0 length causes error
        y, _ = librosa.effects.trim(y) # trim, top_db=default(60)
    # make it unified length to conf.samples
    if len(y) > conf.samples: # long enough
        if trim_long_data:
            y = y[0:0+conf.samples]
    else: # pad blank
        padding = conf.samples - len(y)      # add padding at both ends
        offset = padding // 2
        y = np.pad(y, (offset, conf.samples - len(y) - offset), 'constant')
    return y

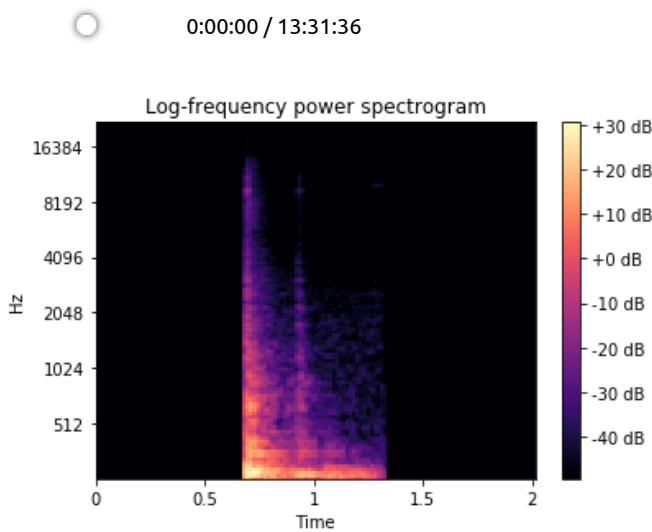
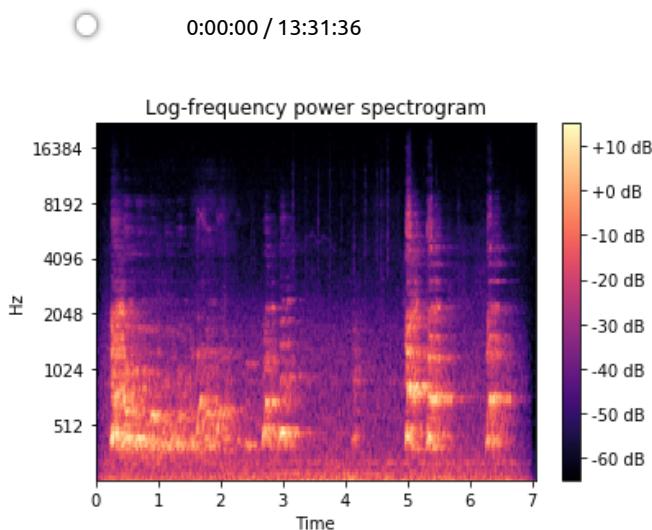
def audio_to_melspectrogram(conf, audio):
    spectrogram = librosa.feature.melspectrogram(audio,
                                                   sr=conf.sampling_rate,
                                                   n_mels=conf.n_mels,
                                                   hop_length=conf.hop_length,
                                                   n_fft=conf.n_fft,
                                                   fmin=conf.fmin,
                                                   fmax=conf.fmax)
    spectrogram = librosa.power_to_db(spectrogram)
    spectrogram = spectrogram.astype(np.float32)
    return spectrogram

def show_melspectrogram(conf, mels, title='Log-frequency power spectrogram'):
    librosa.display.specshow(mels, x_axis='time', y_axis='mel',
                             sr=conf.sampling_rate, hop_length=conf.hop_length,
                             fmin=conf.fmin, fmax=conf.fmax)
    plt.colorbar(format='%+2.0f dB')
    plt.title(title)
    plt.show()

def read_as_melspectrogram(conf, pathname, trim_long_data, debug_display=False):
    x = read_audio(conf, pathname, trim_long_data)
    mels = audio_to_melspectrogram(conf, x)
    if debug_display:
        IPython.display.display(IPython.display.Audio(x, rate=conf.sampling_rate))
    show_melspectrogram(conf, mels)
    return mels

class conf:
    # Preprocessing settings
    sampling_rate = 44100
    duration = 2
    hop_length = 347*duration # to make time steps 128
    fmin = 20
    fmax = sampling_rate // 2
    n_mels = 128
    n_fft = n_mels * 20
    samples = sampling_rate * duration

    # example
    x = read_as_melspectrogram(conf, DATA_CURATED/'0006ae4e.wav', trim_long_data=False,
                               debug_display=True)
    x = read_as_melspectrogram(conf, DATA_CURATED/'05e4ad19.wav', trim_long_data=False,
                               debug_display=True)
```



### Making 2D mel-spectrogram data as 2D 3ch images

So that normal CNN image classifier can handle.

```
In [7]: def mono_to_color(X, mean=None, std=None, norm_max=None, norm_min=None, eps=1e-6):
    # Stack X as [X,X,X]
    X = np.stack([X, X, X], axis=-1)

    # Standardize
    mean = mean or X.mean()
    std = std or X.std()
    Xstd = (X - mean) / (std + eps)
    _min, _max = Xstd.min(), Xstd.max()
    norm_max = norm_max or _max
    norm_min = norm_min or _min
    if (_max - _min) > eps:
        # Scale to [0, 255]
        V = Xstd
        V[V < norm_min] = norm_min
        V[V > norm_max] = norm_max
        V = 255 * (V - norm_min) / (norm_max - norm_min)
        V = V.astype(np.uint8)
    else:
        # Just zero
        V = np.zeros_like(Xstd, dtype=np.uint8)
    return V

def convert_wav_to_image(df, source, img_dest):
    print(f'Converting {source} -> {img_dest}')
    X = []
    for i, row in tqdm_notebook(df.iterrows(), total=df.shape[0]):
        x = read_as_melspectrogram(conf, source=str(row.fname), trim_long_data=False)
        x_color = mono_to_color(x)
        X.append(x_color)
    pickle.dump(X, open(img_dest, 'wb'))
    return X
```

```
In [8]: if not MELS_TRN_CURATED.is_file():
    convert_wav_to_image(trn_curated_df, source=DATA_CURATED, img_dest=MELS_TRN_CURATED)

if not MELS_TRN_NOISY.is_file():
    convert_wav_to_image(trn_noisy_df, source=DATA_NOISY, img_dest=MELS_TRN NOI SY)

if not MELS_TEST.is_file():
    convert_wav_to_image(test_df, source=DATA_TEST, img_dest=MELS_TEST)
```

## Official Metric

```
In [9]: # from official code https://colab.research.google.com/drive/1AgPdhSp7ttY1803fEOH0QKlt_3HJDLi8#scrollTo=cRCaC1b9ogU
def _one_sample_positive_class_precisions(scores, truth):
    """Calculate precisions for each true class for a single sample.

    Args:
        scores: np.array of (num_classes,) giving the individual classifier scores.
        truth: np.array of (num_classes,) bools indicating which classes are true.

    Returns:
        pos_class_indices: np.array of indices of the true classes for this sample.
        pos_class_precisions: np.array of precisions corresponding to each of those classes.
    """
    num_classes = scores.shape[0]
    pos_class_indices = np.flatnonzero(truth > 0)
    # Only calculate precisions if there are some true classes.
    if not len(pos_class_indices):
        return pos_class_indices, np.zeros(0)
    # Retrieval list of classes for this sample.
    retrieved_classes = np.argsort(scores)[::-1]
    # class_rankings[top_scoring_class_index] == 0 etc.
    class_rankings = np.zeros(num_classes, dtype=np.int)
    class_rankings[pos_class_indices] = range(num_classes)
    # Which of these is a true label?
    retrieved_class_true = np.zeros(num_classes, dtype=np.bool)
    retrieved_class_true[class_rankings[pos_class_indices]] = True
    # Num hits for every truncated retrieval list.
    retrieved_cumulative_hits = np.cumsum(retrieved_class_true)
    # Precision of retrieval list truncated at each hit, in order of pos_label
    precision_at_hits = (
        retrieved_cumulative_hits[class_rankings[pos_class_indices]] /
        (1 + class_rankings[pos_class_indices].astype(np.float)))
    return pos_class_indices, precision_at_hits

def calculate_per_class_lwlrap(truth, scores):
    """Calculate label-weighted label-ranking average precision.

    Arguments:
        truth: np.array of (num_samples, num_classes) giving boolean ground-truth
               of presence of that class in that sample.
        scores: np.array of (num_samples, num_classes) giving the classifier-uncertainty's
               real-valued score for each class for each sample.

    Returns:
        per_class_lwlrap: np.array of (num_classes,) giving the lwlrap for each class.
        weight_per_class: np.array of (num_classes,) giving the prior of each class within the truth labels. Then the overall unbalanced lwlrap is simply np.sum(per_class_lwlrap * weight_per_class)
    """
    assert truth.shape == scores.shape
    num_samples, num_classes = scores.shape
    # Space to store a distinct precision value for each class on each sample.
    # Only the classes that are true for each sample will be filled in.
    precisions_for_samples_by_classes = np.zeros((num_samples, num_classes))
```

```
In [10]: class Lwlrap(Callback):
    """ class to compute lwlrap during fastai training"""

    def on_epoch_begin(self, **kwargs):
        self.accumulator = lwlrap_accumulator()

    def on_batch_end(self, last_output, last_target, **kwargs):
        self.accumulator.accumulate_samples(last_target.cpu().numpy(), torch.sigmoid(last_output).cpu().numpy())

    def on_epoch_end(self, last_metrics, **kwargs):
        return add_metrics(last_metrics, self.accumulator.overall_lwlrap())
```

```
In [11]: def _sliding_df(df):
    df_multi = pd.DataFrame(columns=df.columns)

    for index, row in tqdm_notebook(df.iterrows(), total=df.shape[0]):
        idx = CUR_X_FILES.index(row.fname)
        x = PIL.Image.fromarray(CUR_X[idx])
        time_dim, base_dim = x.size
        s = math.ceil((time_dim-conf.n_mels) / TTA_SHIFT) + 1

        fname = row.fname
        tmp = pd.DataFrame(columns=df.columns)
        for crop_x in [int(np.around((time_dim-conf.n_mels)*x/(s-1))) if s != 1
        else 0 for x in range(s)]:
            row.fname = fname + '!' + str(crop_x)
            tmp = tmp.append(row)
        df_multi = df_multi.append(tmp)

    return df_multi.reset_index(drop=True)

def _kfold_prediction(kf, df, file_pattern):
    """ Compute prediction, thruth, index on validation set on models trained using K-Fold """
    overall_preds = None
    overall_thruth = None
    overall_index = None

    for fold, (train_index, valid_index) in enumerate(kf.split(df)):
        print(f'Fold {fold+1}/{n_splits}')

        filename = file_pattern.format(fold=fold)
        learn.load(filename)

        df_multi = _sliding_df(df.iloc[valid_index])

        src = (ImageList.from_df(df_multi, WORK)
               .split_by_idxs(df_multi.index, df_multi.index)
               .label_from_df(label_delim=','))
    )
        data = (src.transform(tfms, size=128)
                .databunch(bs=bs))
    )

    learn.data = data

    # Ensure we have enough data
    assert learn.data.classes == labels, set(labels) - set(learn.data.classes)

    preds, thruth = learn.get_preds(ds_type=DatasetType.Valid)

    fname = df_multi.fname.apply(lambda x: x.split('!')[0])

    preds = pd.DataFrame(preds.cpu().numpy())
    preds['fname'] = fname
    preds = preds.groupby('fname').mean().reset_index(drop=True)

    thruth = pd.DataFrame(thruth.cpu().numpy())
    thruth['fname'] = fname
    thruth = thruth.groupby('fname').mean().reset_index(drop=True)
```

## Borrowed model

Thanks to <https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch> (<https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch>), borrowing simple model here.

```
In [12]: class ConvBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()

        self.conv1 = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, 3, 1, 1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(out_channels, out_channels, 3, 1, 1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )

        self._init_weights()

    def _init_weights(self):
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.kaiming_normal_(m.weight)
                if m.bias is not None:
                    nn.init.zeros_(m.bias)
            elif isinstance(m, nn.BatchNorm2d):
                nn.init.constant_(m.weight, 1)
                nn.init.zeros_(m.bias)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = F.avg_pool2d(x, 2)
        return x

class Classifier(nn.Module):
    def __init__(self, num_classes=1000): # <===== modificaition to comply fast.ai
        super().__init__()

        self.conv = nn.Sequential(
            ConvBlock(in_channels=3, out_channels=64),
            ConvBlock(in_channels=64, out_channels=128),
            ConvBlock(in_channels=128, out_channels=256),
            ConvBlock(in_channels=256, out_channels=512),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1)) # <===== modificaition to comply fast.ai
        self.fc = nn.Sequential(
            nn.Dropout(0.2),
            nn.Linear(512, 128),
            nn.PReLU(),
            nn.BatchNorm1d(128),
            nn.Dropout(0.1),
            nn.Linear(128, num_classes),
        )

    def forward(self, x):
        x = self.conv(x)
        #x = torch.mean(x, dim=3) # <===== modificaition to comply fast.ai
        #x, _ = torch.max(x, dim=2) # <===== modificaition to comply fast.ai
        x = self.avgpool(x) # <===== modificaition to comply fast.ai
        x = self.fc(x)
        return x
```

## File/folder definitions

- df will handle training data.
- test\_df will handle test data.

```
In [51]: df = pd.concat([trn_curated_df, trn_noisy_df], ignore_index=True, sort=True)

X_train = pickle.load(open(MELS_TRN_CURATED, 'rb')) + pickle.load(open(MELS_TRN
_NOISY, 'rb'))

CUR_X_FILES, CUR_X = list(df.fname.values), X_train

del df; gc.collect()
```

```
Traceback (most recent call last):
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/queues.py", line 24
2, in _feed
    send_bytes(obj)
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
200, in send_bytes
    self._send_bytes(m[offset:offset + size])
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
404, in _send_bytes
    self._send(header + buf)
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
368, in _send
    n = write(self._handle, buf)
BrokenPipeError: [Errno 32] Broken pipe
```

Out[51]: 6295

```
Traceback (most recent call last):
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/queues.py", line 24
2, in _feed
    send_bytes(obj)
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
200, in send_bytes
    self._send_bytes(m[offset:offset + size])
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
404, in _send_bytes
    self._send(header + buf)
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
368, in _send
    n = write(self._handle, buf)
OSErrno: [Errno 9] Bad file descriptor
Traceback (most recent call last):
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/queues.py", line 24
2, in _feed
    send_bytes(obj)
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
200, in send_bytes
    self._send_bytes(m[offset:offset + size])
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
404, in _send_bytes
    self._send(header + buf)
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
368, in _send
    n = write(self._handle, buf)
BrokenPipeError: [Errno 32] Broken pipe
Exception in thread QueueFeederThread:
Traceback (most recent call last):
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/queues.py", line 23
2, in _feed
    close()
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
177, in close
    self._close()
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
361, in _close
    _close(self._handle)
OSErrno: [Errno 9] Bad file descriptor

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/eric/anaconda3/lib/python3.7/threading.py", line 917, in _bootstr
ap_inner
    self.run()
  File "/home/eric/anaconda3/lib/python3.7/threading.py", line 865, in run
    self._target(*self._args, **self._kwargs)
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/queues.py", line 26
3, in _feed
    queue_sem.release()
ValueError: semaphore or lock released too many times
Exception in thread QueueFeederThread:
Traceback (most recent call last):
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/queues.py", line 23
2, in _feed
    close()
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
177, in close
    self._close()
  File "/home/eric/anaconda3/lib/python3.7/multiprocessing/connection.py", line
361, in _close
```

## Custom open\_image for fast.ai library to load data from memory

- Important note: Random cropping 1 sec, this is working like augmentation.

```
In [14]: # !!! use globals CUR_X_FILES, CUR_X
def open_fat2019_image(fn, convert_mode, after_open)->Image:
    # open
    fname = fn.split('/')[-1]
    if '!' in fname:
        fname, crop_x = fname.split('!')
        crop_x = int(crop_x)
    else:
        crop_x = -1
    idx = CUR_X_FILES.index(fname)
    x = PIL.Image.fromarray(CUR_X[idx])
    # crop
    time_dim, base_dim = x.size
    if crop_x == -1:
        crop_x = random.randint(0, time_dim - base_dim)
    x = x.crop([crop_x, 0, crop_x+base_dim, base_dim])
    # standardize
    return Image(pil2tensor(x, np.float32).div_(255))

vision.data.open_image = open_fat2019_image
```

```
In [15]: # Define image augmentation
tfms = get_transforms(
    do_flip=False, max_rotate=0, max_lighting=0.1, max_zoom=1.05, max_warp=0.,
)
```

## Follow multi-label classification

- Almost following fast.ai course: <https://nbviewer.jupyter.org/github/fastai/course-v3/blob/master/nbs/dl1/lesson3-planet.ipynb> (<https://nbviewer.jupyter.org/github/fastai/course-v3/blob/master/nbs/dl1/lesson3-planet.ipynb>)
- But pretrained=False

```
In [16]: def borrowed_model(pretrained=False, **kwargs):
    return Classifier(**kwargs)
```

```
In [17]: # Run "fit_one_cycle" and display some graphics
def learning(learn, cycle, lr):
    assert learn.data.classes == labels, set(labels) - set(learn.data.classes)

    learn.lr_find()
    learn.recorder.plot()
    plt.show()

    learn.fit_one_cycle(3 if TOY_MODE else cycle, lr)

    learn.recorder.plot()
    plt.show()

    learn.recorder.plot_lr()
    plt.show()

    learn.recorder.plot_losses()
    plt.show()
```

```
In [18]: class MyMixUpCallback(LearnerCallback):
    def __init__(self, learn:Learner):
        super().__init__(learn)

        # spec_augment hyperparameter
        self.masking_max_percentage=0.25

        # mixup hyperparameter
        self.alpha = .4

    # Inspiration from: https://arxiv.org/pdf/1904.08779.pdf
    # Do: Frequency masking, Frequency masking
    # Dont: Time warping
    # Bonus: Replace masking with a piece of another sample
    def _spec_augment(self, last_input, last_target):
        # Spec Augmentation
        shuffle = torch.randperm(last_target.size(0)).to(last_input.device)
        x1, y1 = last_input[shuffle], last_target[shuffle]

        batch_size, channels, height, width = last_input.size()
        h_percentage = np.random.uniform(low=0., high=self.masking_max_percentage, size=batch_size)
        w_percentage = np.random.uniform(low=0., high=self.masking_max_percentage, size=batch_size)
        alpha = (h_percentage + w_percentage) - (h_percentage * w_percentage)
        alpha = last_input.new(alpha)
        alpha = alpha.unsqueeze(1)

        new_input = last_input.clone()

        for i in range(batch_size):
            h_mask = int(h_percentage[i] * height)
            h = int(np.random.uniform(0.0, height - h_mask))
            new_input[i, :, h:h + h_mask, :] = x1[i, :, h:h + h_mask, :]

            w_mask = int(w_percentage[i] * width)
            w = int(np.random.uniform(0.0, width - w_mask))
            new_input[i, :, :, w:w + w_mask] = x1[i, :, :, w:w + w_mask]

        new_target = (1-alpha) * last_target + alpha*y1
        return new_input, new_target

    # Inspired from fastai implementation of https://arxiv.org/abs/1710.09412
    def _mixup(self, last_input, last_target):
        lambd = np.random.beta(self.alpha, self.alpha, last_target.size(0))
        lambd = np.concatenate([lambd[:,None], 1-lambd[:,None]], 1).max(1)
        lambd = last_input.new(lambd)
        shuffle = torch.randperm(last_target.size(0)).to(last_input.device)
        x1, y1 = last_input[shuffle], last_target[shuffle]
        new_input = (last_input * lambd.view(lambd.size(0),1,1,1)) + x1 * (1-lambd).view(lambd.size(0),1,1,1)
        if len(last_target.shape) == 2:
            lambd = lambd.unsqueeze(1).float()
        new_target = last_target.float() * lambd + y1.float() * (1-lambd)
        return new_input, new_target

    def on_batch_begin(self, last_input, last_target, train, **kwargs):
        if not train: return
        new_input, new_target = self._mixup(last_input, last_target)
        new_input, new_target = self._spec_augment(new_input, new_target)
        return {'last_input': new_input, 'last_target': new_target}
```

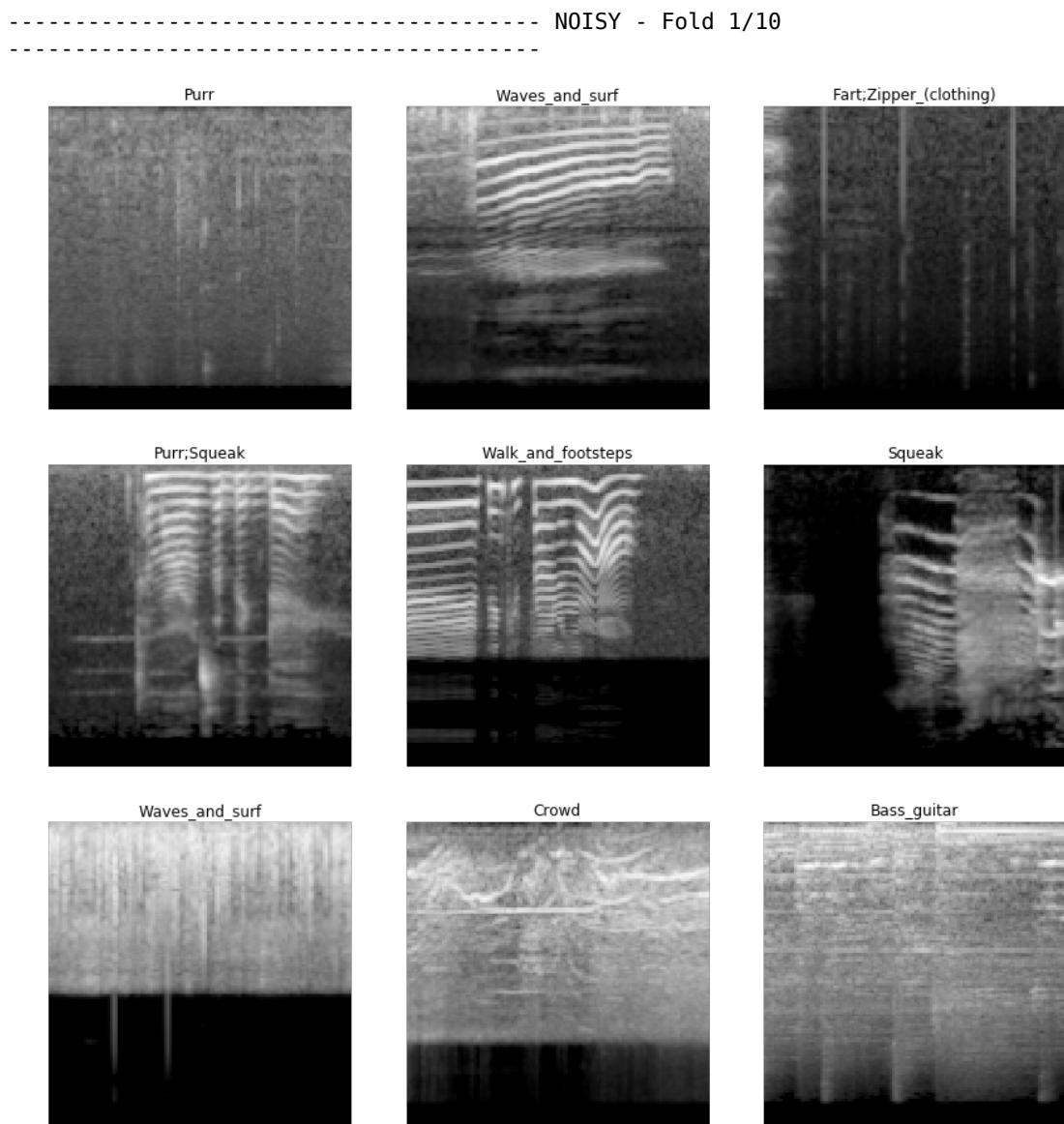
```
In [19]: labels = [
    'Accelerating_and_revving_and_vroom',
    'Accordion',
    'Acoustic_guitar',
    'Applause',
    'Bark',
    'Bass_drum',
    'Bass_guitar',
    'Bathtub_(filling_or_washing)',
    'Bicycle_bell',
    'Burping_and_eructation',
    'Bus',
    'Buzz',
    'Car_passing_by',
    'Cheering',
    'Chewing_and_mastication',
    'Child_speech_and_kidSpeaking',
    'Chink_and_clink',
    'Chirp_and_tweet',
    'Church_bell',
    'Clapping',
    'Computer_keyboard',
    'Crackle',
    'Cricket',
    'Crowd',
    'Cupboard_open_or_close',
    'Cutlery_and_silverware',
    'Dishes_and_pots_and_pans',
    'Drawer_open_or_close',
    'Drip',
    'Electric_guitar',
    'Fart',
    'Female_singing',
    'Female_speech_and_womanSpeaking',
    'Fill_(with_liquid)',
    'Finger_snapping',
    'Frying_(food)',
    'Gasp',
    'Glockenspiel',
    'Gong',
    'Gurgling',
    'Harmonica',
    'Hi-hat',
    'Hiss',
    'Keys_jangling',
    'Knock',
    'Male_singing',
    'Male_speech_and_manSpeaking',
    'Marimba_and_xylophone',
    'Mechanical_fan',
    'Meow',
    'Microwave_oven',
    'Motorcycle',
    'Printer',
    'Purr',
    'Race_car_and_auto_racing',
    'Raindrop',
    'Run',
    'Scissors',
    'Screaming',
    'Shatter',
    'Sigh',
    'Sink_(filling_or_washing)'
```

```
In [20]: kf_noisy = KFold(n_splits=n_splits, shuffle=True, random_state=67890)
```

```
In [21]: kf_curated = KFold(n_splits=n_splits, shuffle=True, random_state=123)
```

## Train on noisy data

```
In [22]: for fold, (train_index, valid_index) in enumerate(kf_noisy.split(trn_noisy_d  
f)):  
    print('-' * 40, f'NOISY - Fold {fold+1}/{n_splits}', '-' * 40)  
  
    src = (ImageList.from_df(trn_noisy_df, WORK)  
           .split_by_idxs(train_index, valid_index)  
           .label_from_df(label_delim=',')  
    )  
    data = (src.transform(tfms, size=128)  
            .databunch(bs=bs)  
    )  
    data.show_batch(3)  
    plt.show()  
  
    learn = cnn_learner(data, borrowed_model, pretrained=False, metrics=[Lwlrap  
()])  
    learn.callback_fns.append(MyMixUpCallback)  
    learn.unfreeze()  
  
    assert learn.data.classes == labels  
#     break  
    learning(learn, 10, slice(1e-5, 3e-2))  
    learning(learn, 100, 1e-3)  
  
    learn.save(f'stage-1_fold-{fold}')  
  
    if TOY_MODE:  
        break
```



```
In [23]: learn.model
```

```
Out[23]: Sequential(  
    (0): Sequential(  
        (0): Sequential(  
            (0): ConvBlock(  
                (conv1): Sequential(  
                    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
                    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
                    (2): ReLU()  
                )  
                (conv2): Sequential(  
                    (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
                    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
                    (2): ReLU()  
                )  
            )  
            (1): ConvBlock(  
                (conv1): Sequential(  
                    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
                    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
                    (2): ReLU()  
                )  
                (conv2): Sequential(  
                    (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
                    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
                    (2): ReLU()  
                )  
            )  
            (2): ConvBlock(  
                (conv1): Sequential(  
                    (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
                    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
                    (2): ReLU()  
                )  
                (conv2): Sequential(  
                    (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
                    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
                    (2): ReLU()  
                )  
            )  
            (3): ConvBlock(  
                (conv1): Sequential(  
                    (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
                    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
                    (2): ReLU()  
                )  
                (conv2): Sequential(  
                    (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
                    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
                )  
            )  
        )  
    )  
)
```

```
In [24]: kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-1_fold-{fold}')
```

Overall lwlrmp on CURATED dataset: 0.41096481268327745

	lwlrap	weight
Tap	0.027562	0.013039
Raindrop	0.030165	0.013039
Yell	0.045917	0.013039
Female_speech_and_womanSpeaking	0.051134	0.013039
Male_speech_and_manSpeaking	0.078842	0.013039
Run	0.091376	0.013039
Cupboard_open_or_close	0.093436	0.013039
Whispering	0.132053	0.013039
Fill_(with_liquid)	0.138031	0.008693
Bass_drum	0.144150	0.013039
Burping_and_eructation	0.152787	0.013039
Mechanical_fan	0.154831	0.008519
Strum	0.188433	0.013039
Keys_jangling	0.190058	0.013039
Buzz	0.204003	0.009736
Slam	0.209160	0.013039
Trickle_and_dribble	0.209426	0.009214
Clapping	0.230242	0.013039
Fart	0.230567	0.013039
Drip	0.246597	0.013039
Meow	0.254810	0.013039
Electric_guitar	0.255392	0.013039
Walk_and_footsteps	0.262422	0.013039
Male_singing	0.265152	0.013039
Sneeze	0.265667	0.010953
Finger_snapping	0.266323	0.013039
Gurgling	0.268217	0.013039
Tick-tock	0.283312	0.012517
Drawer_open_or_close	0.284116	0.013039
Waves_and_surf	0.286819	0.013039
...	...	...
Sink_(filling_or_washing)	0.483997	0.013039
Chink_and_clink	0.486003	0.013039
Car_passing_by	0.497806	0.013039
Gasp	0.498232	0.008345

```
In [25]: kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-1_fold-{fold}')
```

Overall lwlrapp on NOISY dataset: 0.6505740867763616

	<b>lwlrap</b>	<b>weight</b>
<b>Zipper_(clothing)</b>	0.248449	0.0125
<b>Burping_and_eructation</b>	0.248790	0.0125
<b>Fart</b>	0.310956	0.0125
<b>Buzz</b>	0.381784	0.0125
<b>Gasp</b>	0.394716	0.0125
<b>Writing</b>	0.396788	0.0125
<b>Sneeze</b>	0.396947	0.0125
<b>Finger_snapping</b>	0.425786	0.0125
<b>Computer_keyboard</b>	0.446361	0.0125
<b>Tick-tock</b>	0.450959	0.0125
<b>Cupboard_open_or_close</b>	0.458769	0.0125
<b>Whispering</b>	0.461395	0.0125
<b>Scissors</b>	0.474020	0.0125
<b>Shatter</b>	0.490433	0.0125
<b>Sigh</b>	0.495949	0.0125
<b>Toilet_flush</b>	0.497040	0.0125
<b>Meow</b>	0.505523	0.0125
<b>Knock</b>	0.523774	0.0125
<b>Bicycle_bell</b>	0.532929	0.0125
<b>Chewing_and_mastication</b>	0.535547	0.0125
<b>Strum</b>	0.546121	0.0125
<b>Male_singing</b>	0.552236	0.0125
<b>Chink_and_clink</b>	0.552872	0.0125
<b>Squeak</b>	0.564156	0.0125
<b>Purr</b>	0.573379	0.0125
<b>Skateboard</b>	0.575323	0.0125
<b>Electric_guitar</b>	0.591425	0.0125
<b>Crackle</b>	0.619393	0.0125
<b>Keys_jangling</b>	0.620099	0.0125
<b>Bass_guitar</b>	0.622011	0.0125
...	...	...
<b>Printer</b>	0.724457	0.0125
<b>Glockenspiel</b>	0.725519	0.0125
<b>Bus</b>	0.727974	0.0125
<b>Sink_(filling_or_washing)</b>	0.734114	0.0125

**Train on curated data**

```
In [26]: for fold, (train_index, valid_index) in enumerate(kf_curated.split(trn_curated_df)):
    print('-' * 40, f'CURATED - Fold {fold+1}/{n_splits}', '-' * 40)

    learn.load(f'stage-1_fold-{fold}')

    src = (ImageList.from_df(trn_curated_df, WORK)
           .split_by_idxs(train_index, valid_index)
           .label_from_df(label_delim=','))
    )
    data = (src.transform(tfms, size=128)
            .databunch(bs=bs))
    )
    data.show_batch(3)
    plt.show()

    learn.data = data

    learning(learn, 100, 3e-3 / 5)

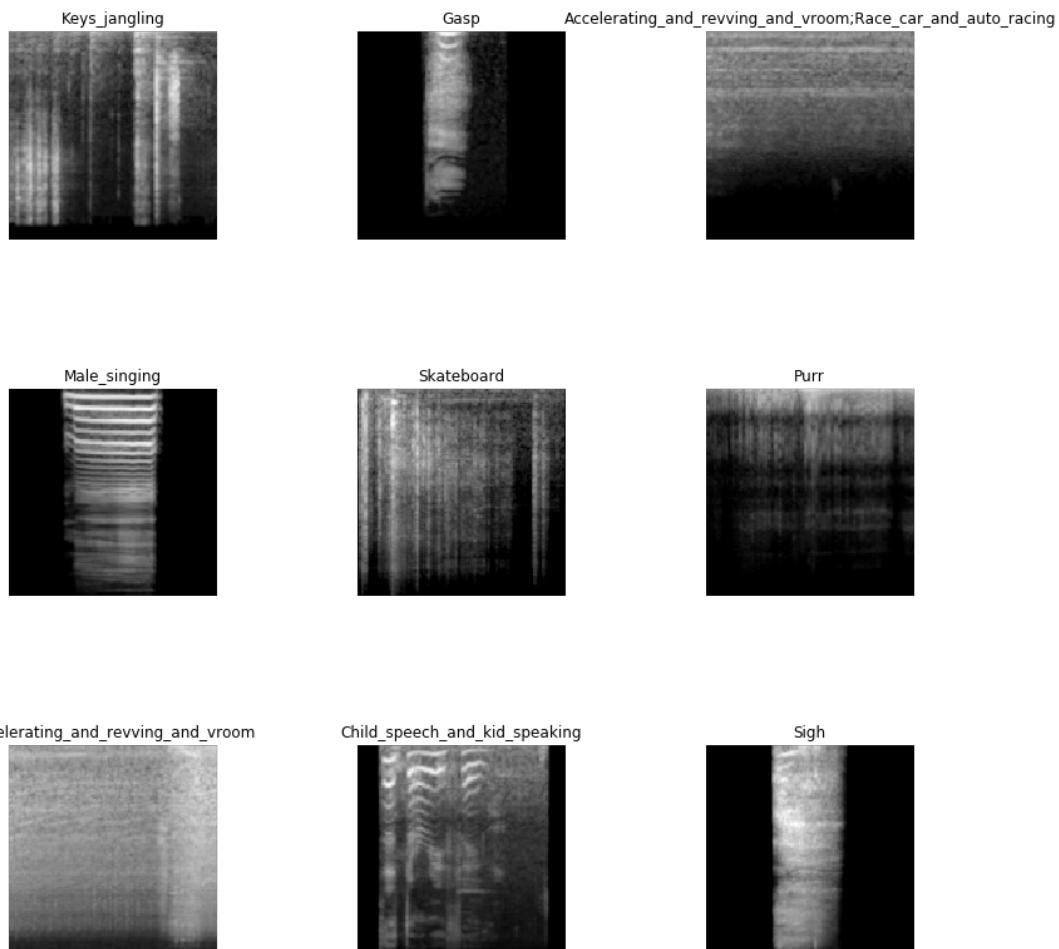
    learn.save(f'stage-2_fold-{fold}')
    learn.export(f'stage-2_fold-{fold}.pkl')

if TOY_MODE:
    break
```

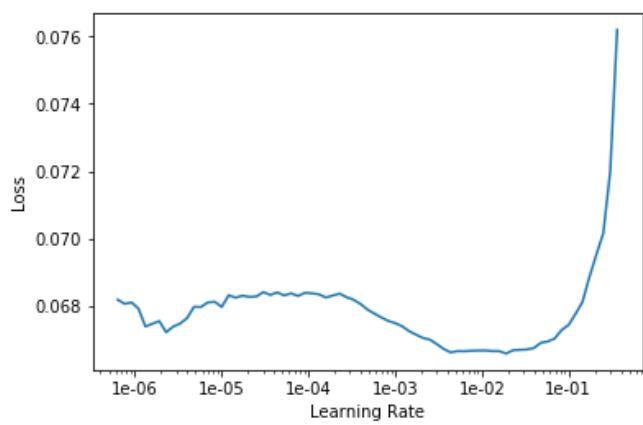
----- CURATED - Fold 1/10 -----

/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

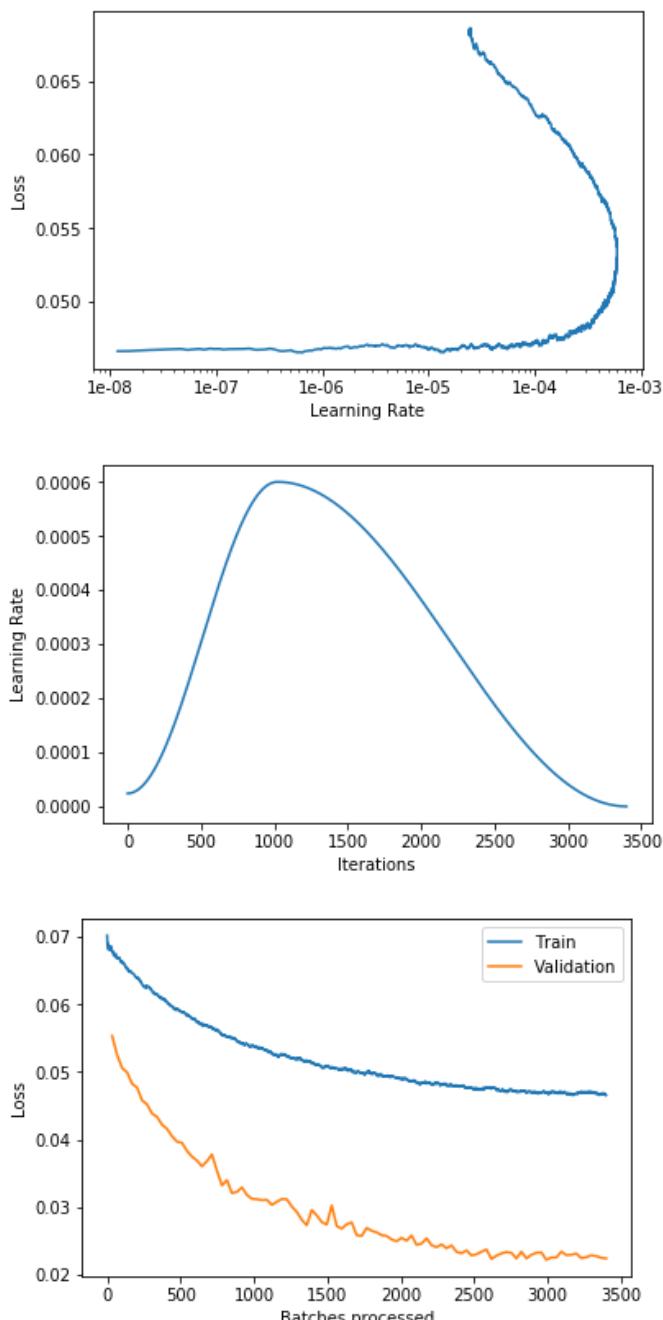
"type " + obj.\_\_name\_\_ + ". It won't be checked "



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.067866	0.055346	0.408862	00:11
1	0.066923	0.052537	0.447787	00:10
2	0.066184	0.050595	0.472755	00:10
3	0.065301	0.049942	0.486378	00:10
4	0.064648	0.048257	0.509498	00:10
5	0.063915	0.047690	0.513258	00:10
6	0.063022	0.045798	0.555040	00:10
7	0.062773	0.045198	0.561189	00:10
8	0.062113	0.043846	0.588285	00:10
9	0.061522	0.043352	0.589102	00:10
10	0.060875	0.042207	0.604505	00:10
11	0.060357	0.041577	0.602563	00:10
12	0.059724	0.040487	0.619578	00:10
13	0.059366	0.039677	0.639559	00:10
14	0.058886	0.039505	0.636833	00:10
15	0.058502	0.038326	0.650279	00:10
16	0.057888	0.037463	0.661164	00:10
17	0.057466	0.036891	0.685946	00:10
18	0.057010	0.036026	0.689493	00:10
19	0.056835	0.036803	0.675800	00:10
20	0.056590	0.037790	0.656678	00:10
21	0.056179	0.035389	0.690107	00:10
22	0.055574	0.033191	0.731869	00:10
23	0.055270	0.033993	0.723821	00:10
24	0.055049	0.032075	0.733597	00:10
25	0.054614	0.032251	0.749017	00:10
26	0.054165	0.032907	0.734246	00:10
27	0.054204	0.031804	0.736992	00:10
28	0.053949	0.031226	0.741722	00:10
29	0.053662	0.031166	0.744537	00:10
30	0.053437	0.031027	0.745704	00:10
31	0.053226	0.031088	0.759122	00:10
32	0.052768	0.030327	0.752099	00:10
33	0.052563	0.030794	0.743760	00:10
34	0.052678	0.031195	0.745335	00:10
35	0.052402	0.031162	0.754028	00:10



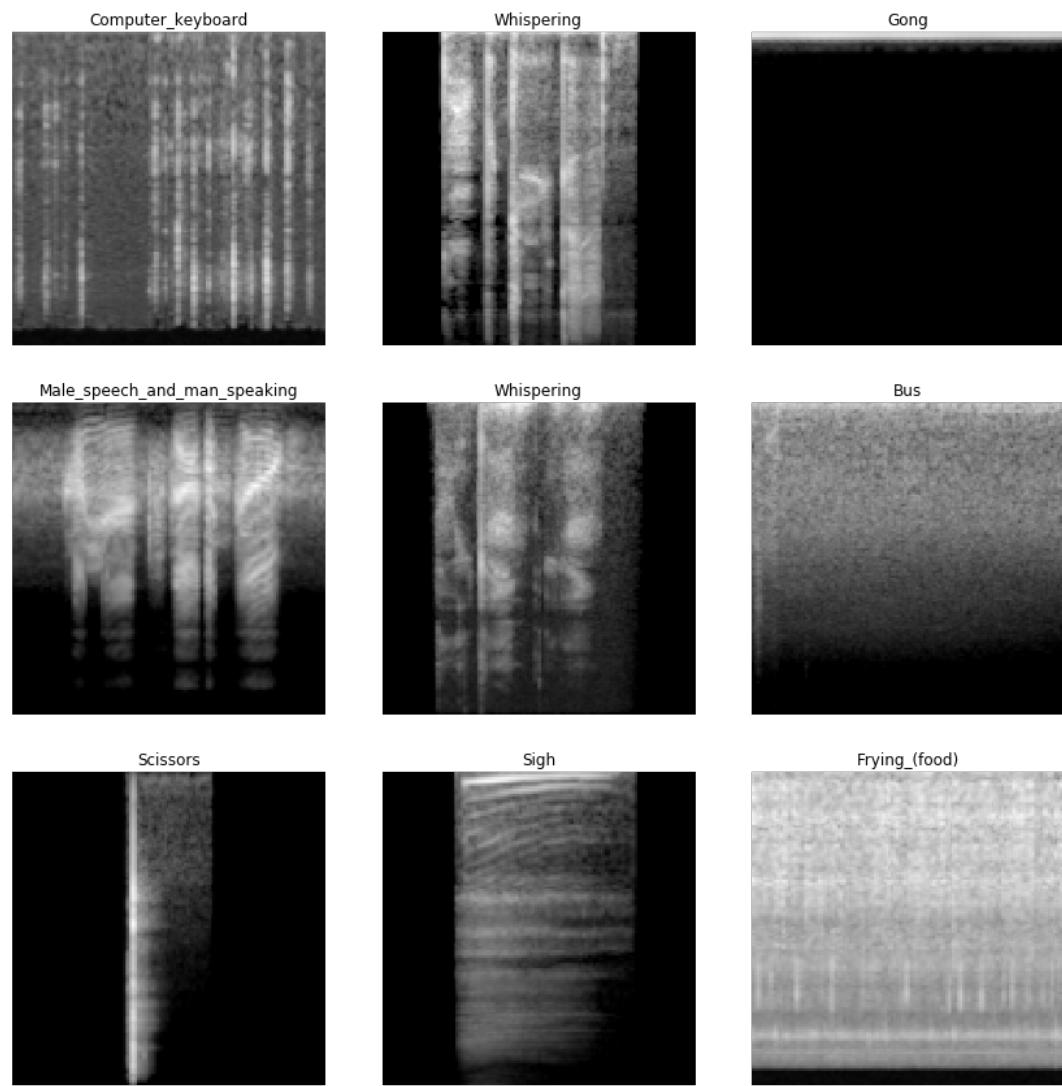
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us  
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo  
n't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

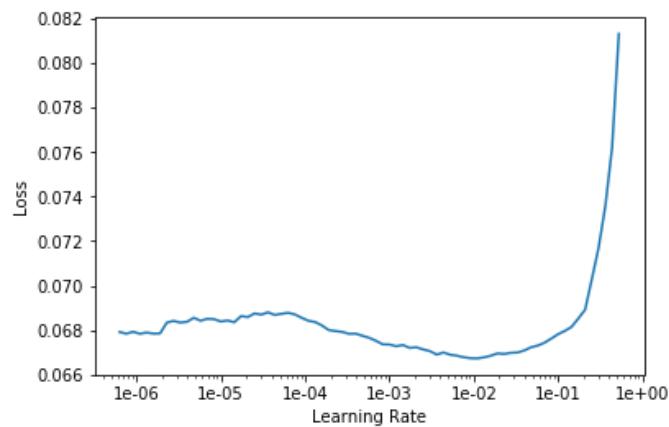
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us  
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo  
n't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

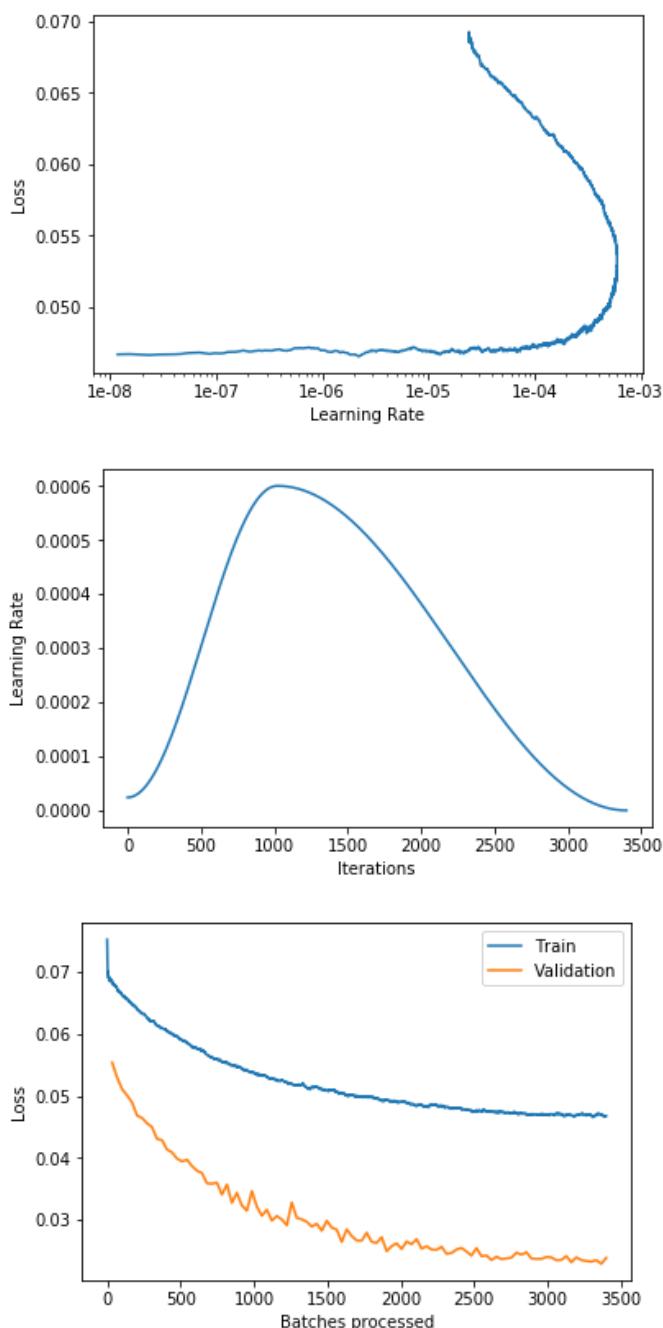
```
----- CURATED - Fold 2/10  
-----
```



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.068434	0.055376	0.418316	00:10
1	0.067394	0.052947	0.458699	00:10
2	0.066347	0.051080	0.474810	00:10
3	0.065625	0.050052	0.495010	00:10
4	0.064813	0.048925	0.519421	00:10
5	0.064003	0.046802	0.534355	00:10
6	0.063277	0.046437	0.550619	00:10
7	0.062637	0.045552	0.557155	00:10
8	0.062064	0.044950	0.589955	00:10
9	0.061358	0.043031	0.623684	00:10
10	0.060946	0.042837	0.616257	00:10
11	0.060516	0.041314	0.633922	00:10
12	0.060051	0.040855	0.650002	00:10
13	0.059623	0.039857	0.652941	00:10
14	0.058995	0.039393	0.665328	00:10
15	0.058544	0.039690	0.671004	00:10
16	0.058011	0.038641	0.673749	00:10
17	0.057663	0.037914	0.684056	00:10
18	0.057306	0.037524	0.668953	00:10
19	0.056575	0.035862	0.709568	00:10
20	0.056257	0.035798	0.706700	00:10
21	0.055963	0.035984	0.705666	00:10
22	0.055517	0.034028	0.729705	00:10
23	0.055330	0.035678	0.716885	00:10
24	0.054974	0.032718	0.743899	00:10
25	0.054641	0.034315	0.721101	00:10
26	0.054361	0.032284	0.753653	00:10
27	0.054215	0.031441	0.762334	00:10
28	0.053749	0.034613	0.729958	00:10
29	0.053457	0.032078	0.735587	00:10
30	0.053259	0.030618	0.754037	00:10
31	0.053096	0.031609	0.744968	00:10
32	0.052861	0.029841	0.768394	00:10
33	0.052530	0.030587	0.767477	00:10
34	0.052462	0.030016	0.763173	00:10
35	0.052215	0.029086	0.780617	00:10



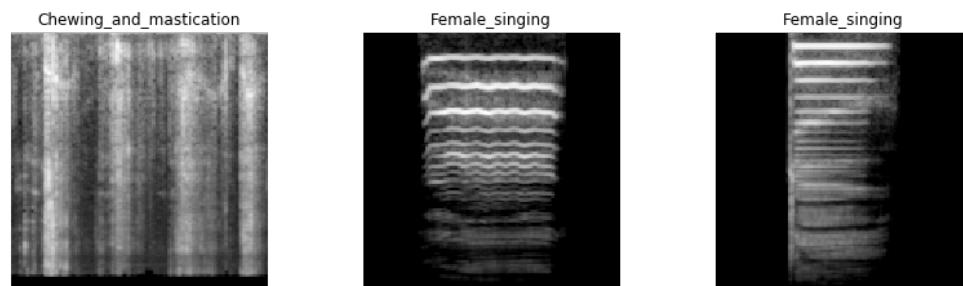
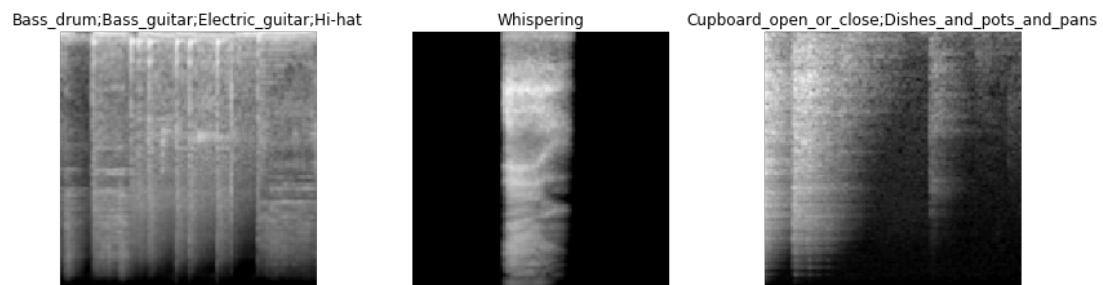
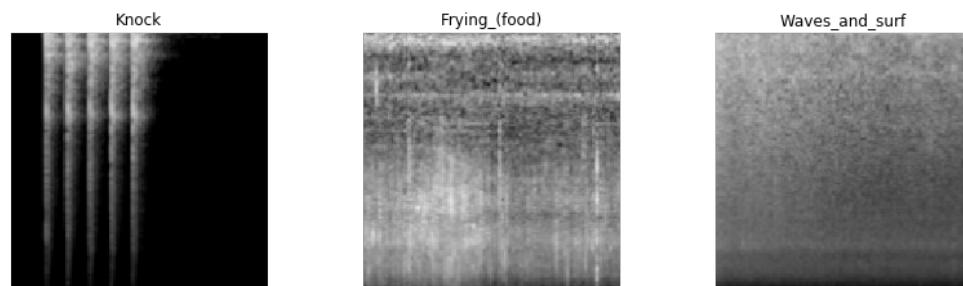
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

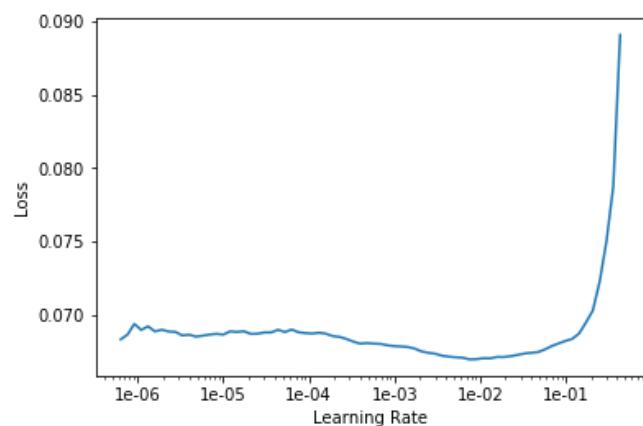
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

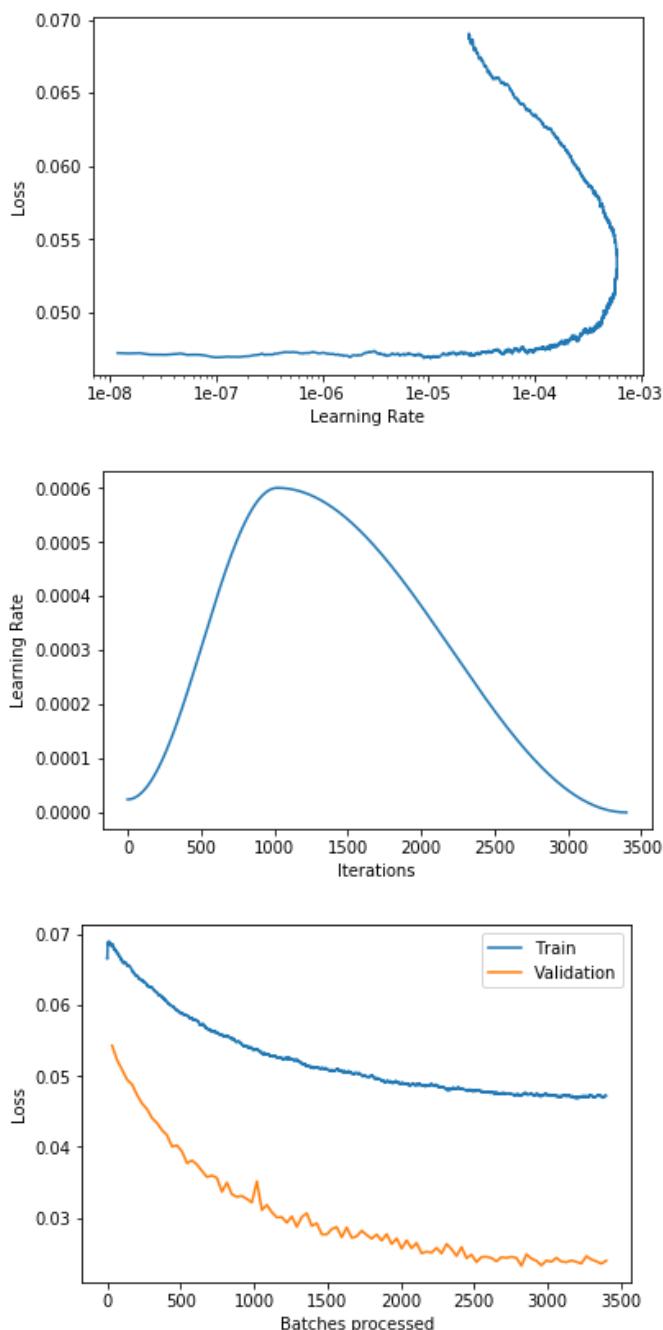
----- CURATED - Fold 3/10  
-----



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.068403	0.054323	0.402808	00:10
1	0.067302	0.052254	0.439193	00:10
2	0.066378	0.050900	0.458043	00:10
3	0.065665	0.049487	0.477995	00:10
4	0.064977	0.048812	0.490253	00:10
5	0.064224	0.047315	0.518597	00:10
6	0.063522	0.046138	0.542084	00:10
7	0.062909	0.045326	0.539836	00:10
8	0.062482	0.044069	0.566573	00:10
9	0.061820	0.043354	0.591176	00:10
10	0.061159	0.042357	0.611309	00:10
11	0.060458	0.041651	0.609330	00:10
12	0.059880	0.040044	0.646331	00:10
13	0.059313	0.040199	0.634363	00:10
14	0.058892	0.039339	0.646399	00:10
15	0.058590	0.037696	0.667022	00:10
16	0.058280	0.038077	0.683737	00:10
17	0.057879	0.037494	0.687979	00:10
18	0.057262	0.036585	0.694533	00:10
19	0.056726	0.035735	0.697914	00:10
20	0.056463	0.035943	0.686055	00:10
21	0.056207	0.035612	0.704370	00:10
22	0.055858	0.033634	0.717684	00:10
23	0.055718	0.034951	0.711538	00:10
24	0.055181	0.033286	0.711698	00:10
25	0.054806	0.032926	0.712805	00:10
26	0.054536	0.033055	0.717100	00:10
27	0.054195	0.032629	0.723548	00:10
28	0.054023	0.032137	0.727348	00:10
29	0.053769	0.035105	0.700390	00:10
30	0.053280	0.031061	0.755390	00:10
31	0.053133	0.031780	0.743098	00:10
32	0.052858	0.030754	0.751730	00:10
33	0.052628	0.030022	0.768639	00:10
34	0.052610	0.030066	0.747631	00:10
35	0.052536	0.029277	0.750011	00:10



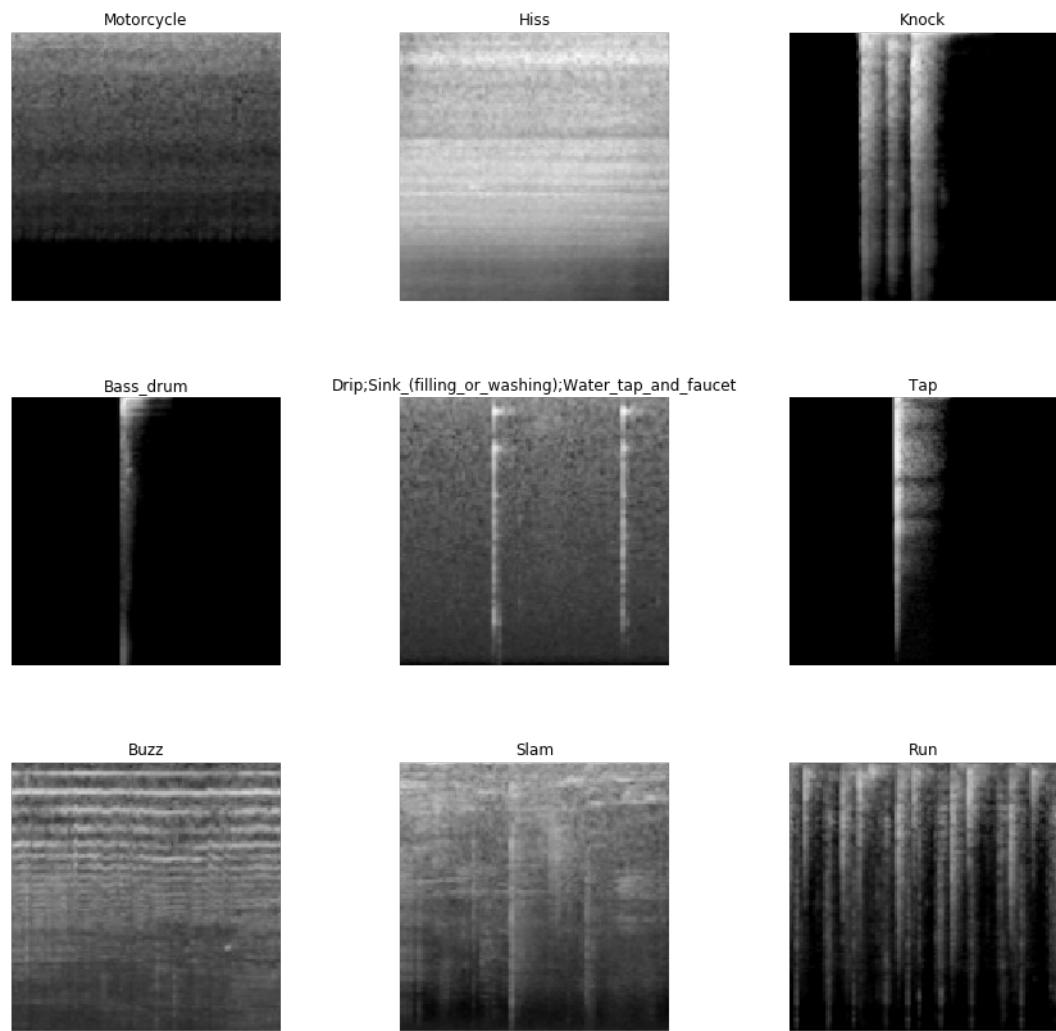
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us  
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo  
n't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

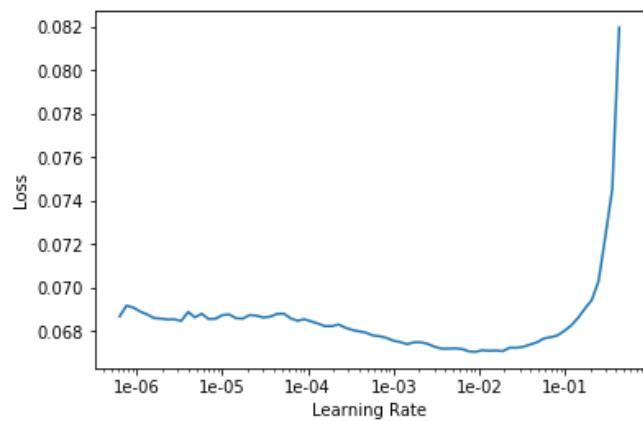
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us  
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo  
n't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

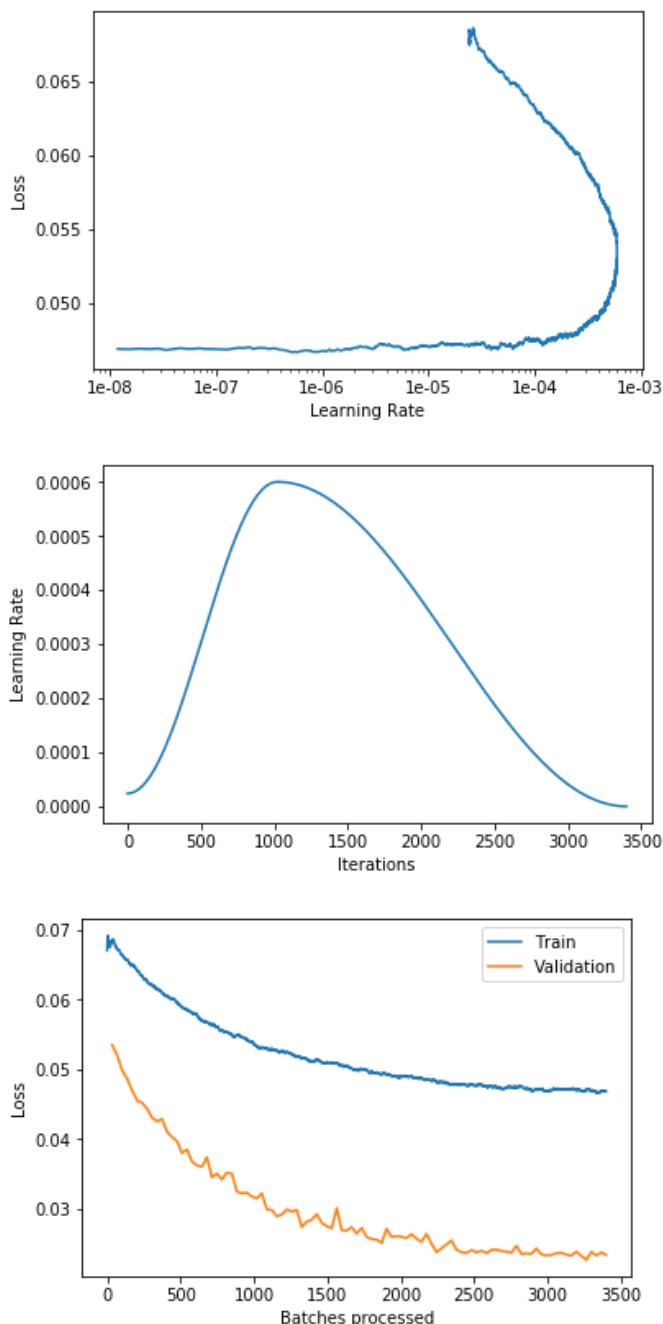
```
----- CURATED - Fold 4/10  
-----
```



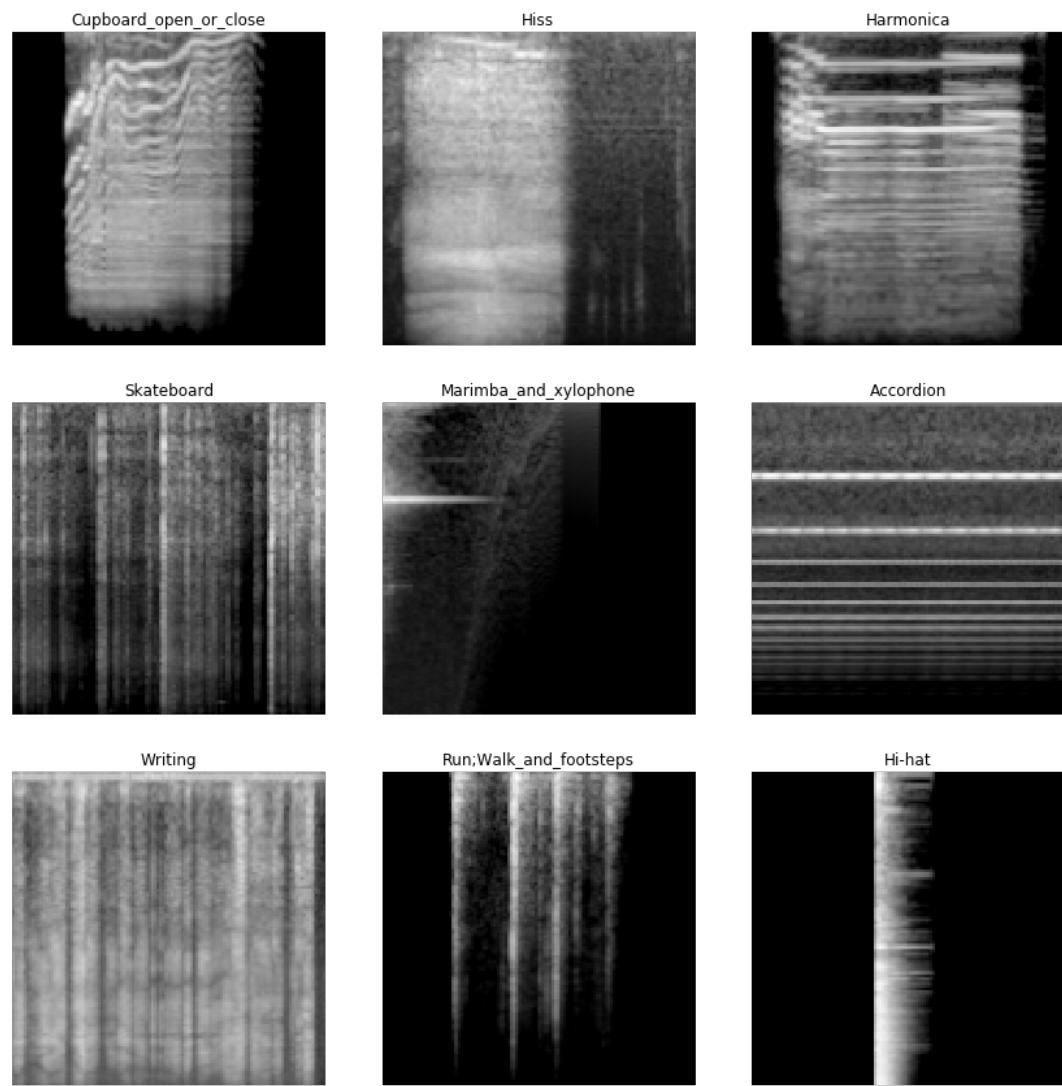
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



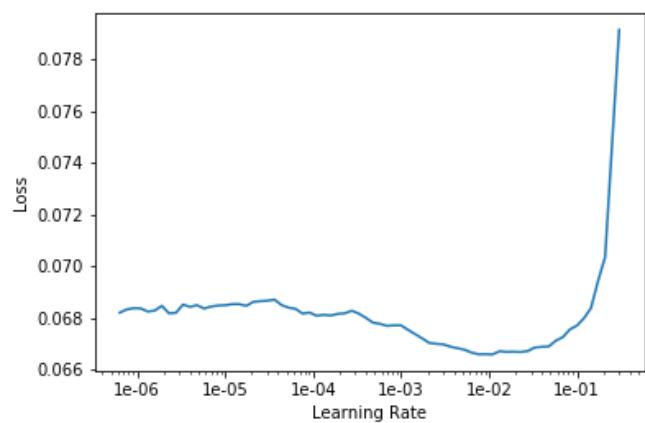
epoch	train_loss	valid_loss	lwlrap	time
0	0.068305	0.053503	0.452716	00:10
1	0.067168	0.052031	0.462611	00:10
2	0.066370	0.049883	0.501358	00:10
3	0.065642	0.048670	0.511634	00:10
4	0.064967	0.046999	0.540920	00:10
5	0.064191	0.045504	0.564924	00:10
6	0.063388	0.045211	0.585965	00:10
7	0.062558	0.044349	0.592418	00:10
8	0.062062	0.043088	0.602161	00:10
9	0.061538	0.042530	0.623057	00:10
10	0.060998	0.042877	0.617307	00:10
11	0.060521	0.041044	0.649367	00:10
12	0.060145	0.040302	0.651826	00:10
13	0.059536	0.039689	0.660296	00:10
14	0.058904	0.038017	0.674430	00:10
15	0.058547	0.038515	0.662752	00:10
16	0.058114	0.036778	0.697237	00:10
17	0.057745	0.036237	0.693035	00:10
18	0.056986	0.036045	0.692009	00:10
19	0.056752	0.037404	0.678047	00:10
20	0.056335	0.034530	0.706133	00:10
21	0.056050	0.035081	0.710374	00:10
22	0.055553	0.034204	0.714019	00:10
23	0.055284	0.035199	0.696108	00:10
24	0.054908	0.035050	0.709759	00:10
25	0.054888	0.032489	0.738598	00:10
26	0.054645	0.032246	0.730025	00:10
27	0.054334	0.032343	0.739364	00:10
28	0.053880	0.031770	0.738984	00:10
29	0.053311	0.031524	0.732276	00:10
30	0.053175	0.032217	0.726224	00:10
31	0.053094	0.029945	0.760948	00:10
32	0.052967	0.029870	0.756901	00:10
33	0.052700	0.028954	0.769794	00:10
34	0.052648	0.029192	0.768495	00:10
35	0.052173	0.029931	0.756818	00:10



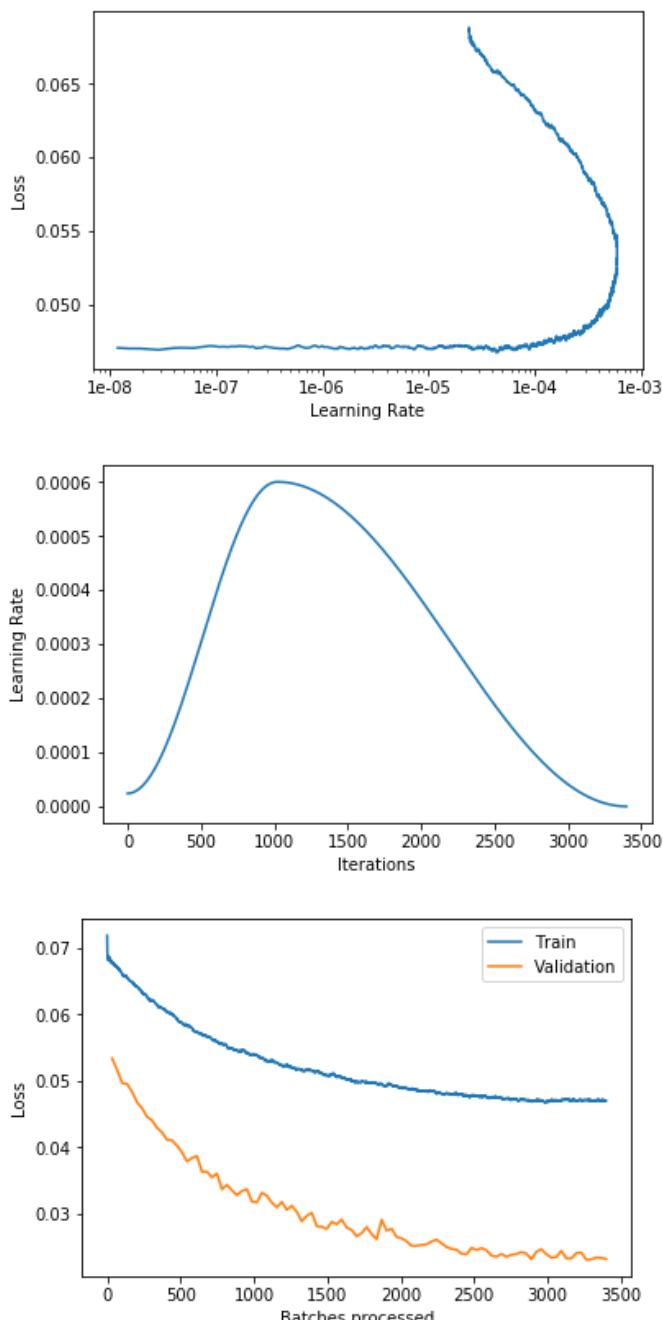
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
    "type " + obj.__name__ + ". It won't be checked "
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
    "type " + obj.__name__ + ". It won't be checked "
----- CURATED - Fold 5/10
-----
```



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.067845	0.053399	0.464275	00:10
1	0.067256	0.051696	0.477656	00:10
2	0.066229	0.049707	0.504184	00:10
3	0.065616	0.049530	0.509044	00:10
4	0.064889	0.048369	0.526988	00:10
5	0.064240	0.046776	0.546110	00:10
6	0.063413	0.045902	0.559164	00:10
7	0.062754	0.044589	0.590870	00:10
8	0.062036	0.044062	0.597335	00:10
9	0.061476	0.042942	0.625444	00:10
10	0.061000	0.042270	0.625714	00:10
11	0.060556	0.041129	0.641077	00:10
12	0.060022	0.041059	0.648890	00:10
13	0.059413	0.040240	0.653879	00:10
14	0.058758	0.039245	0.662507	00:10
15	0.058398	0.037880	0.687971	00:10
16	0.057947	0.038369	0.685797	00:10
17	0.057438	0.038693	0.690973	00:10
18	0.056996	0.036310	0.709058	00:10
19	0.056716	0.036327	0.718502	00:10
20	0.056378	0.035460	0.718757	00:10
21	0.056014	0.036052	0.702675	00:10
22	0.055637	0.033654	0.728865	00:10
23	0.055264	0.034336	0.720956	00:10
24	0.054922	0.033509	0.737364	00:10
25	0.054604	0.032788	0.737075	00:10
26	0.054542	0.033390	0.740911	00:10
27	0.054297	0.033682	0.731206	00:10
28	0.054017	0.031826	0.748651	00:10
29	0.053876	0.031691	0.756682	00:10
30	0.053455	0.033144	0.741164	00:10
31	0.053000	0.032695	0.738758	00:10
32	0.053055	0.031677	0.758191	00:10
33	0.052814	0.030948	0.746690	00:10
34	0.052421	0.031745	0.757508	00:10
35	0.052361	0.030578	0.758723	00:10



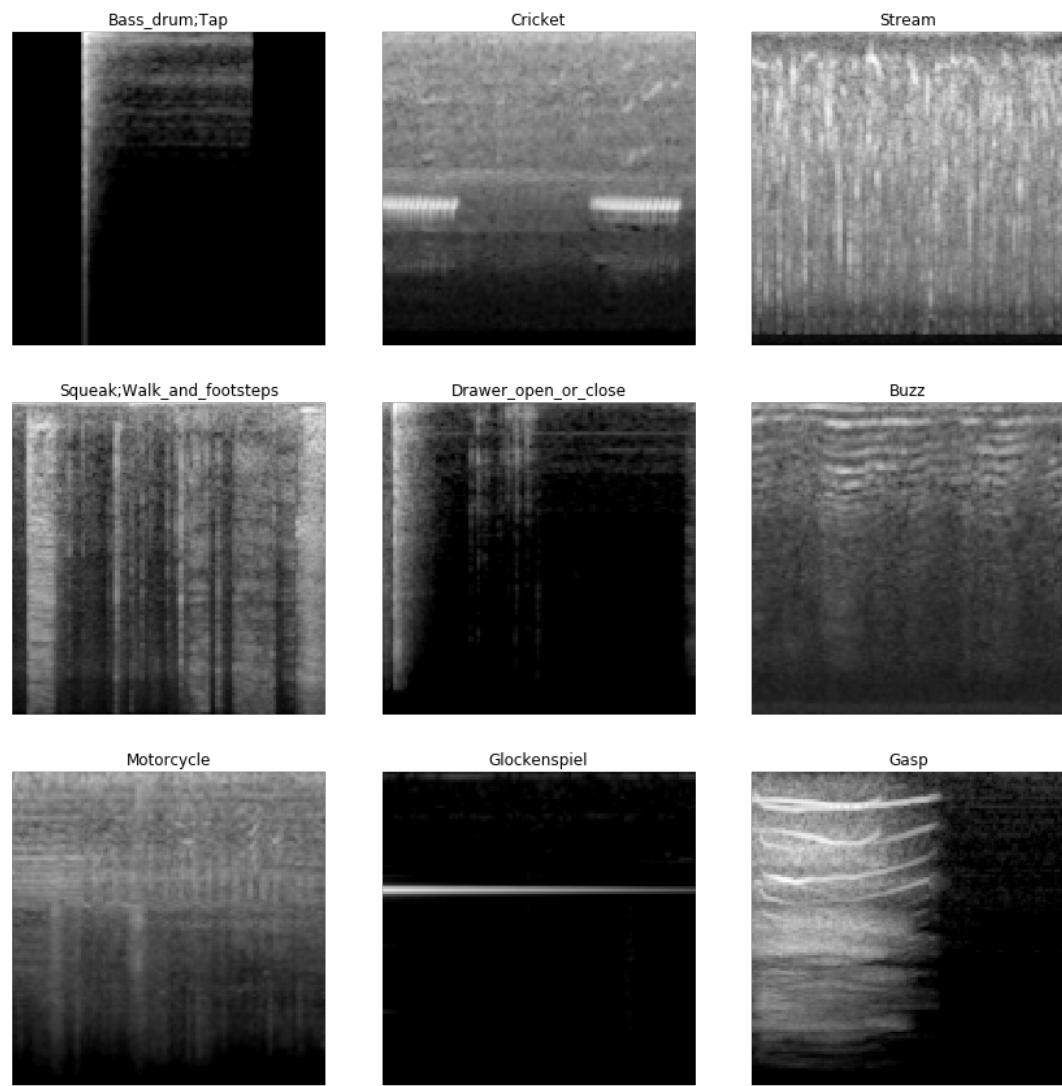
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

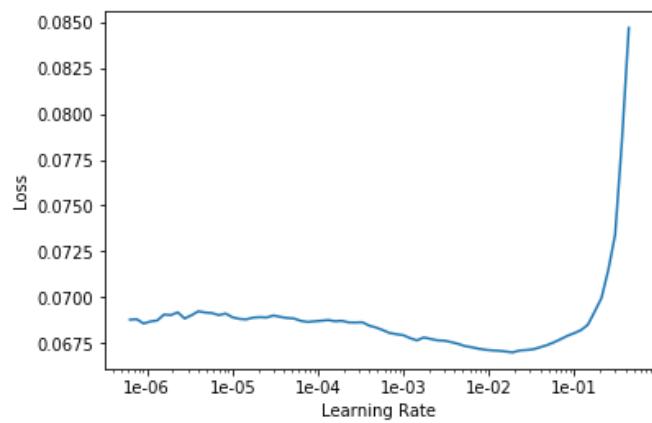
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

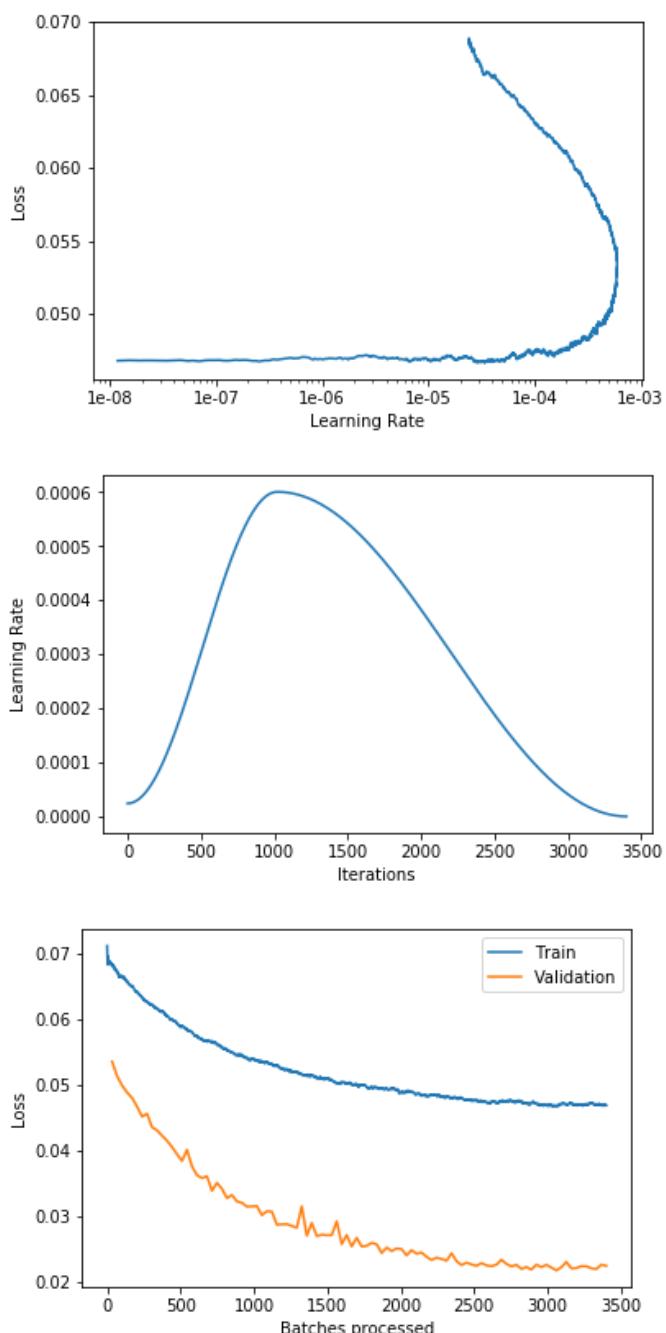
----- CURATED - Fold 6/10  
-----



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.068218	0.053511	0.447022	00:10
1	0.067224	0.051269	0.481466	00:10
2	0.066481	0.049857	0.485509	00:10
3	0.065785	0.048785	0.503204	00:10
4	0.064936	0.047963	0.516308	00:10
5	0.064136	0.046633	0.544908	00:10
6	0.063287	0.045124	0.558814	00:10
7	0.062708	0.045557	0.556973	00:10
8	0.062166	0.043512	0.584187	00:10
9	0.061633	0.043025	0.597525	00:10
10	0.061061	0.042218	0.617828	00:10
11	0.060522	0.041441	0.622112	00:10
12	0.059860	0.040502	0.632348	00:10
13	0.059335	0.039394	0.653031	00:10
14	0.058838	0.038376	0.659707	00:10
15	0.058422	0.040060	0.636937	00:10
16	0.057909	0.037524	0.683942	00:10
17	0.057438	0.036213	0.705636	00:10
18	0.056955	0.035751	0.702655	00:10
19	0.056622	0.036036	0.708744	00:10
20	0.056583	0.033819	0.722496	00:10
21	0.056207	0.035033	0.715650	00:10
22	0.055744	0.034145	0.721754	00:10
23	0.055278	0.032707	0.733996	00:10
24	0.054891	0.033185	0.722423	00:10
25	0.054555	0.032205	0.744596	00:10
26	0.054371	0.031999	0.744729	00:10
27	0.054162	0.031406	0.768814	00:10
28	0.053891	0.031458	0.766255	00:10
29	0.053617	0.031503	0.750130	00:10
30	0.053524	0.030116	0.773604	00:10
31	0.053384	0.030707	0.761559	00:10
32	0.053155	0.030661	0.749990	00:10
33	0.052790	0.028666	0.789411	00:10
34	0.052520	0.028708	0.784852	00:10
35	0.052356	0.028752	0.767261	00:10



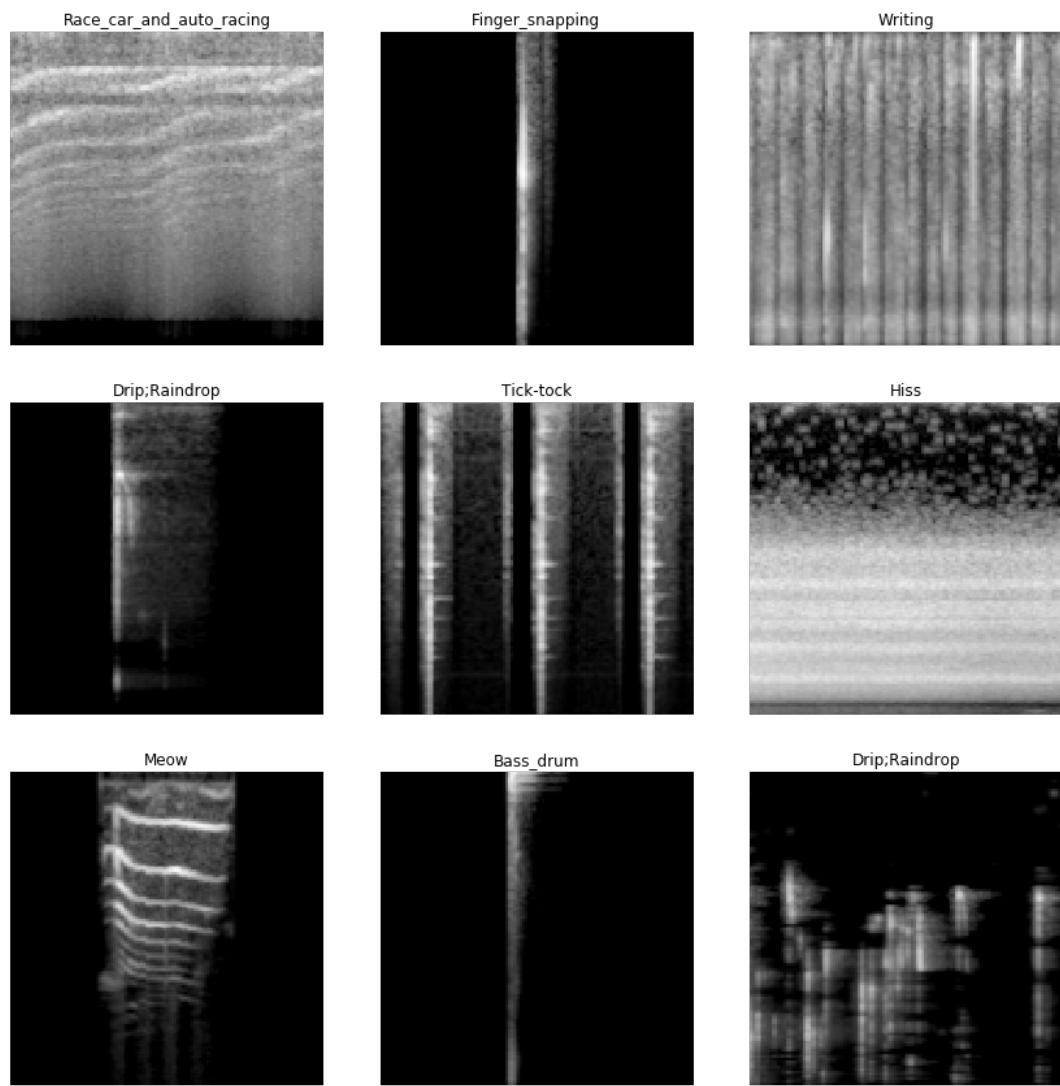
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

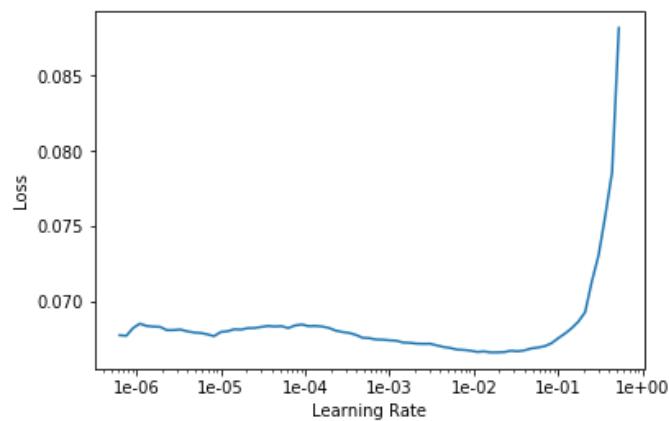
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

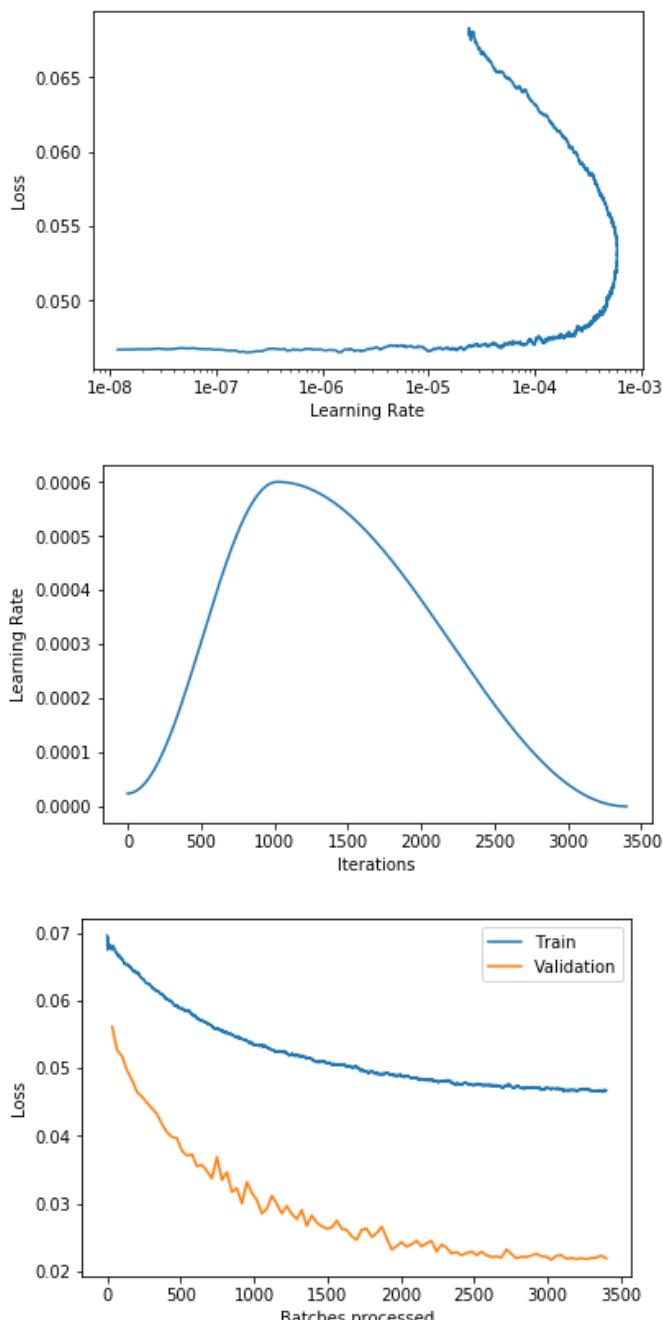
----- CURATED - Fold 7/10  
-----



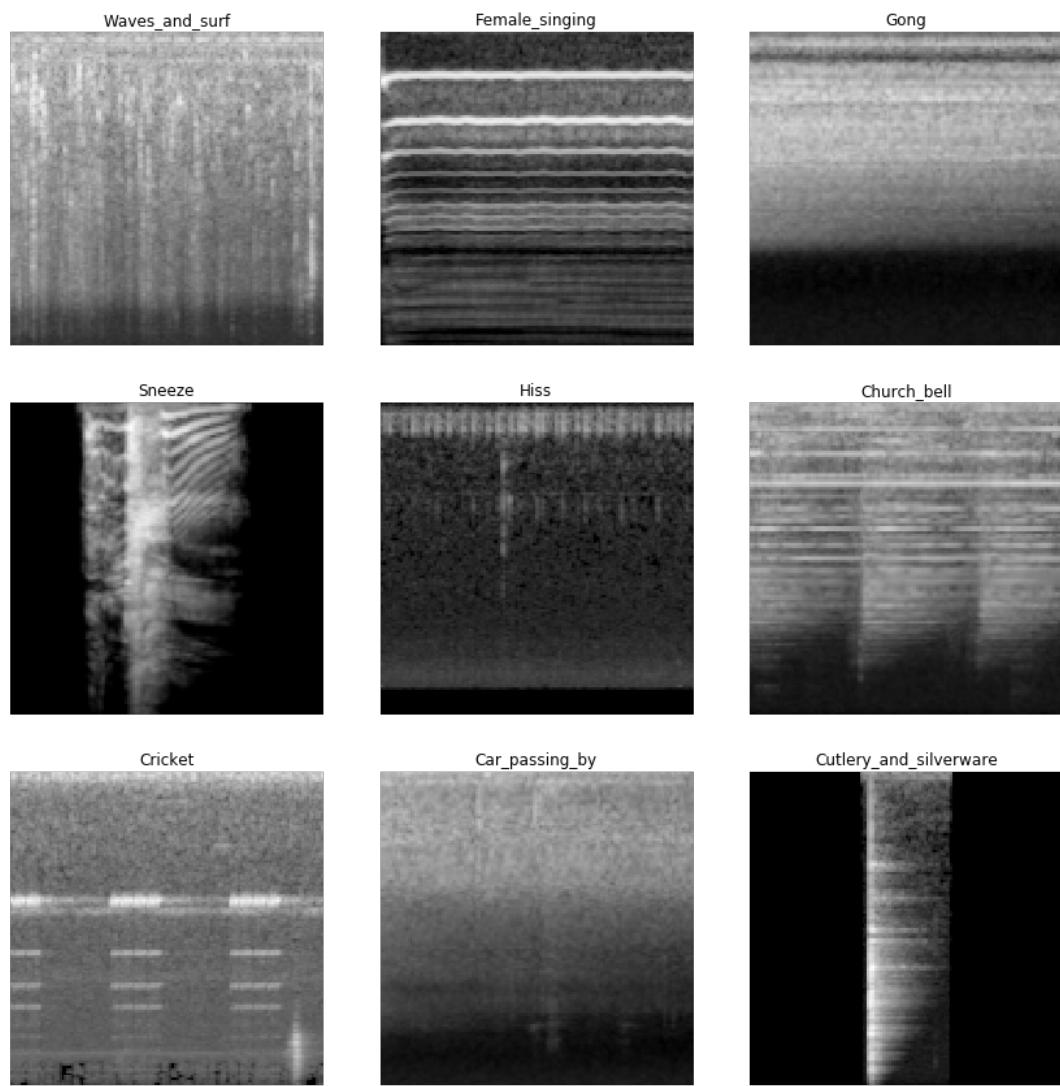
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



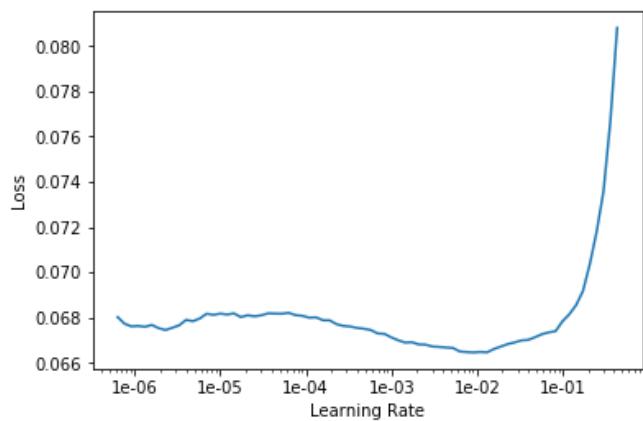
epoch	train_loss	valid_loss	lwlrap	time
0	0.067820	0.056111	0.441370	00:10
1	0.066980	0.052661	0.482764	00:10
2	0.066143	0.051761	0.499426	00:10
3	0.065345	0.049621	0.512956	00:10
4	0.064668	0.048258	0.545881	00:10
5	0.064032	0.046445	0.563697	00:10
6	0.063285	0.045792	0.572174	00:10
7	0.062543	0.044883	0.593601	00:10
8	0.062016	0.044050	0.598059	00:10
9	0.061397	0.043225	0.620751	00:10
10	0.060940	0.041742	0.648530	00:10
11	0.060237	0.040519	0.647493	00:10
12	0.059798	0.039864	0.672226	00:10
13	0.059258	0.039658	0.659723	00:10
14	0.058859	0.037799	0.700772	00:10
15	0.058376	0.037088	0.704970	00:10
16	0.058023	0.037297	0.702202	00:10
17	0.057429	0.035525	0.711630	00:10
18	0.057007	0.035719	0.718788	00:10
19	0.056602	0.034821	0.725638	00:10
20	0.056180	0.033726	0.725962	00:10
21	0.055802	0.036910	0.684044	00:10
22	0.055505	0.033495	0.733264	00:10
23	0.055249	0.034655	0.725939	00:10
24	0.054955	0.031714	0.757990	00:10
25	0.054600	0.032323	0.749895	00:10
26	0.054320	0.030028	0.768307	00:10
27	0.054070	0.033177	0.721378	00:10
28	0.053801	0.031602	0.752606	00:10
29	0.053419	0.030529	0.752261	00:10
30	0.053414	0.028554	0.796318	00:10
31	0.053112	0.029259	0.782296	00:10
32	0.052717	0.031196	0.762485	00:10
33	0.052471	0.029927	0.766059	00:10
34	0.052477	0.028577	0.778785	00:10
35	0.052188	0.029669	0.777965	00:10



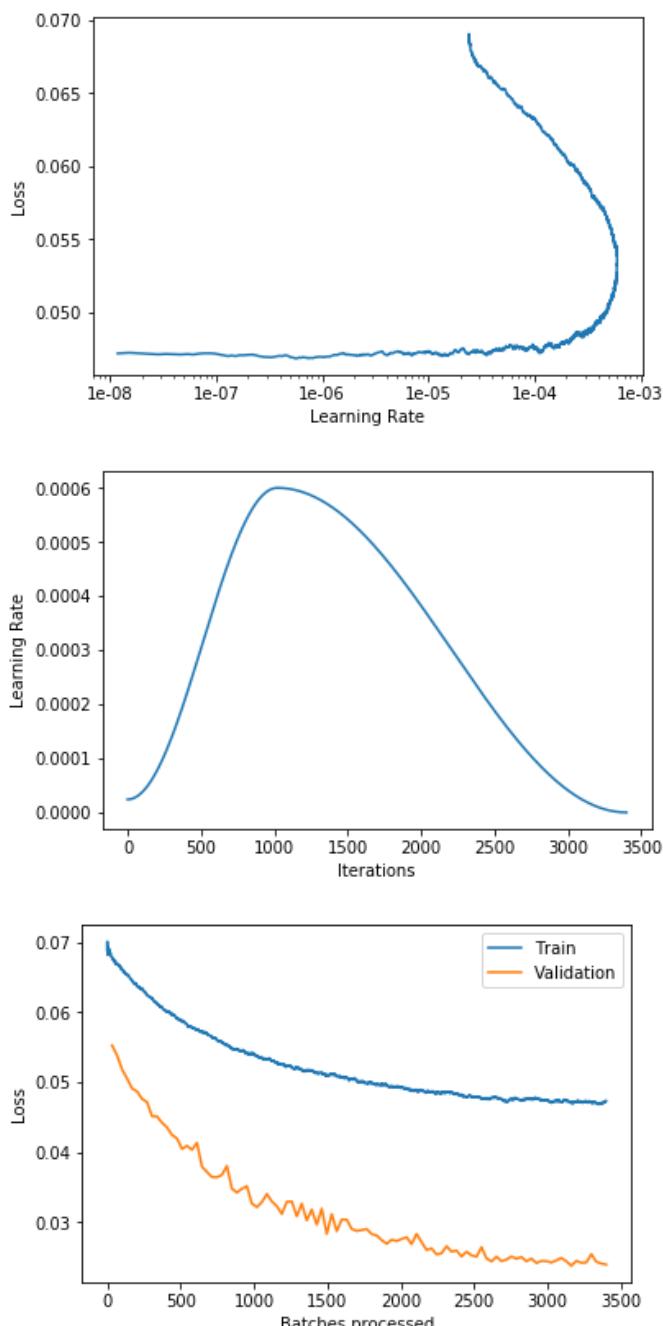
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
    "type " + obj.__name__ + ". It won't be checked "
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
    "type " + obj.__name__ + ". It won't be checked "
----- CURATED - Fold 8/10
-----
```



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.067870	0.055235	0.417623	00:10
1	0.066837	0.053790	0.450238	00:10
2	0.066050	0.051794	0.475368	00:10
3	0.065268	0.050471	0.491169	00:10
4	0.064602	0.049076	0.517110	00:10
5	0.063771	0.048655	0.527178	00:10
6	0.063270	0.047581	0.541648	00:10
7	0.062435	0.047126	0.561008	00:10
8	0.061849	0.045113	0.589007	00:10
9	0.061235	0.045074	0.600626	00:10
10	0.060678	0.044180	0.590181	00:10
11	0.060116	0.043508	0.603708	00:10
12	0.059789	0.042433	0.618656	00:10
13	0.059060	0.041932	0.631550	00:10
14	0.058759	0.040434	0.642256	00:10
15	0.058278	0.040866	0.647316	00:10
16	0.057771	0.040325	0.664877	00:10
17	0.057505	0.041308	0.639862	00:10
18	0.057129	0.037899	0.673354	00:10
19	0.056873	0.037146	0.710806	00:10
20	0.056515	0.036416	0.703063	00:10
21	0.056107	0.036400	0.698666	00:10
22	0.055721	0.036719	0.700843	00:10
23	0.055307	0.038014	0.675145	00:10
24	0.054908	0.034768	0.707235	00:10
25	0.054698	0.034196	0.718813	00:10
26	0.054484	0.034735	0.716302	00:10
27	0.054157	0.035125	0.707977	00:10
28	0.054003	0.032640	0.743422	00:10
29	0.053768	0.032119	0.744239	00:10
30	0.053474	0.032827	0.743814	00:10
31	0.053296	0.034023	0.718796	00:10
32	0.053083	0.032932	0.733243	00:10
33	0.052768	0.032247	0.751117	00:10
34	0.052504	0.031148	0.755955	00:10
35	0.052281	0.032890	0.717711	00:10



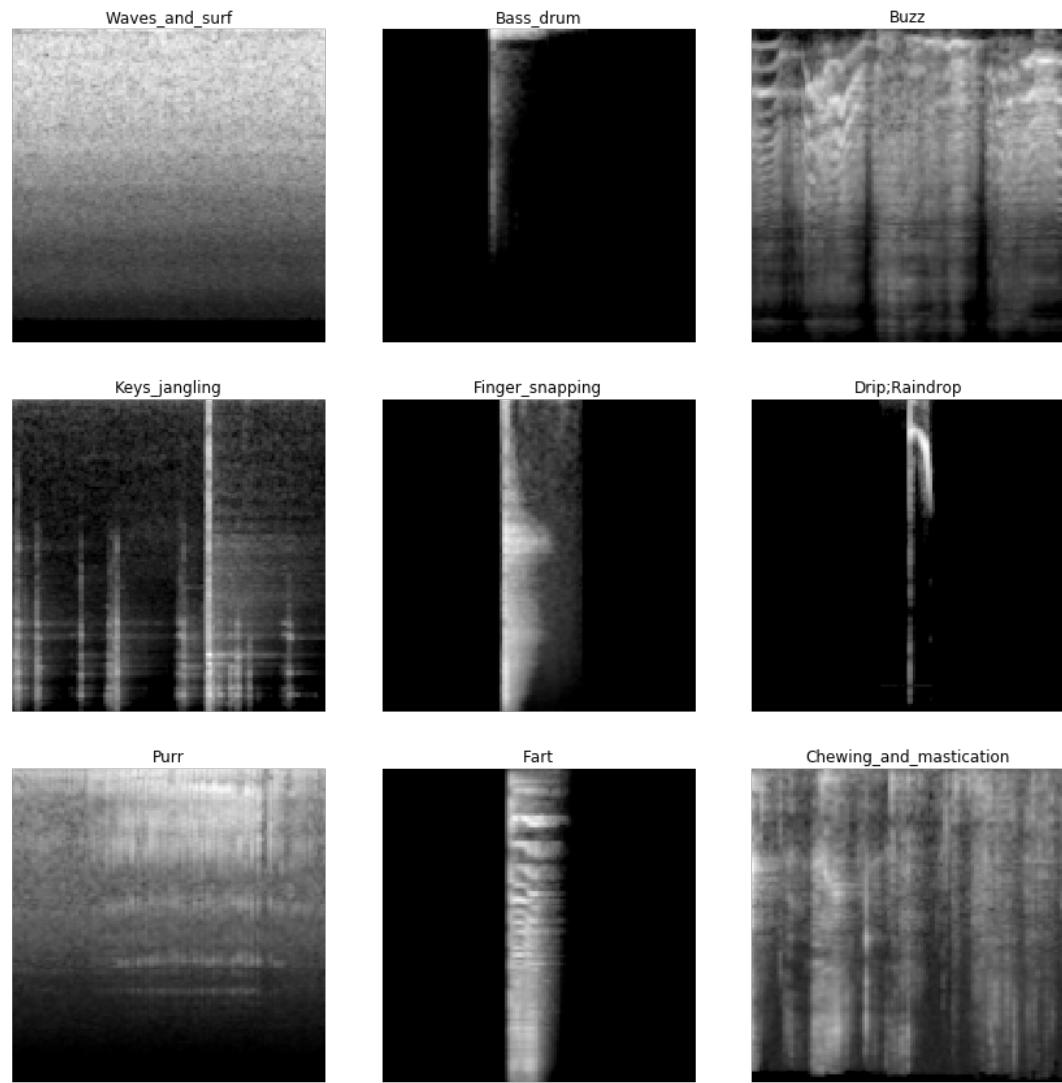
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

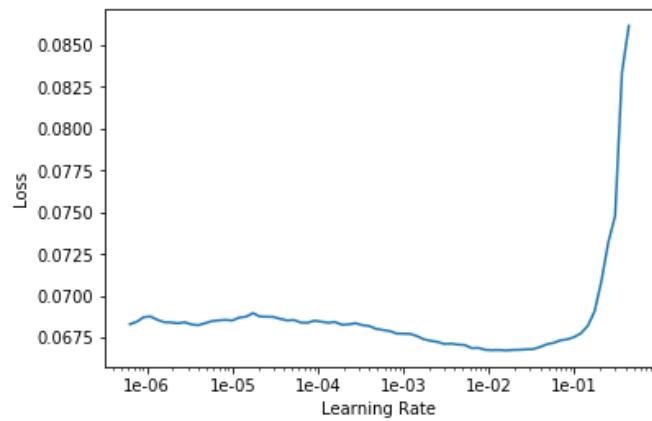
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

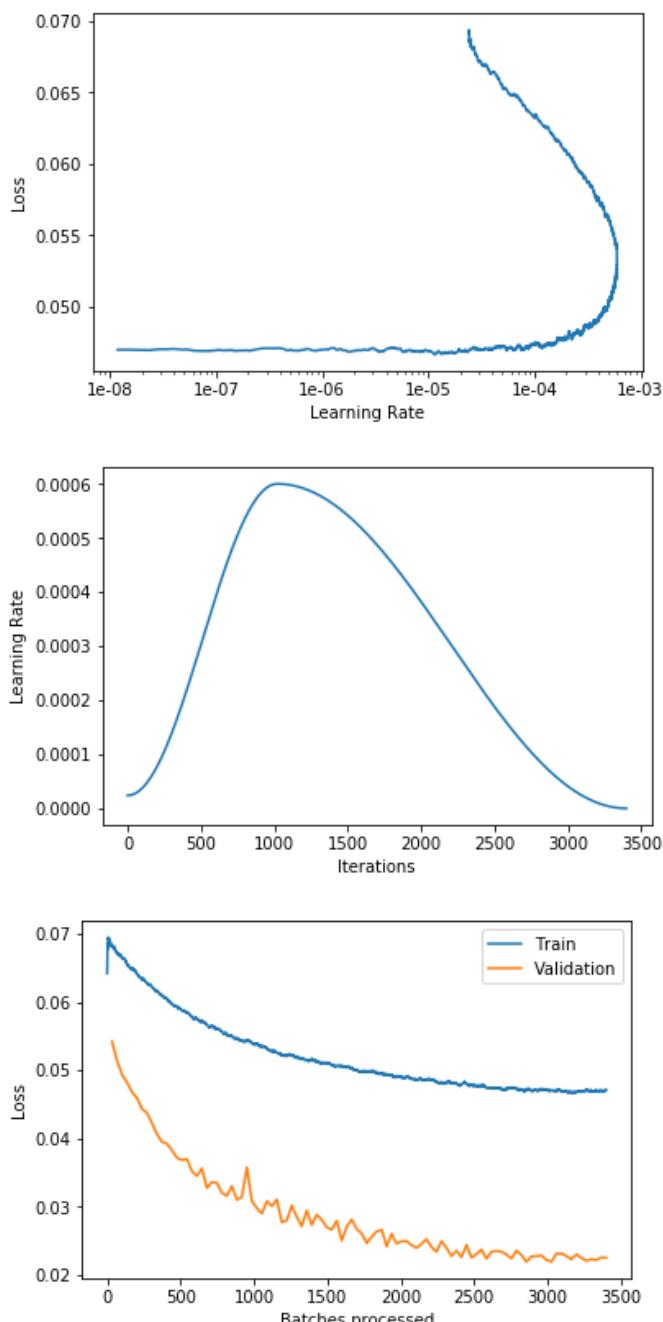
----- CURATED - Fold 9/10  
-----



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.068185	0.054233	0.443308	00:10
1	0.067239	0.051328	0.471018	00:10
2	0.066451	0.049355	0.501174	00:10
3	0.065602	0.048182	0.508987	00:10
4	0.064850	0.046783	0.535360	00:10
5	0.064069	0.045888	0.558231	00:10
6	0.063407	0.044306	0.583845	00:10
7	0.062775	0.043720	0.590483	00:10
8	0.062149	0.042208	0.619397	00:10
9	0.061578	0.040626	0.643714	00:10
10	0.060944	0.039490	0.663027	00:10
11	0.060405	0.039232	0.655802	00:10
12	0.059741	0.038245	0.669091	00:10
13	0.059328	0.037147	0.688545	00:10
14	0.058815	0.036823	0.706141	00:10
15	0.058446	0.036968	0.702541	00:10
16	0.057857	0.035166	0.722456	00:10
17	0.057579	0.034487	0.723887	00:10
18	0.057235	0.035603	0.709122	00:10
19	0.056647	0.032723	0.736066	00:10
20	0.056501	0.033590	0.724687	00:10
21	0.056123	0.033465	0.735701	00:10
22	0.055621	0.032020	0.752031	00:10
23	0.055315	0.031596	0.769815	00:10
24	0.055088	0.033014	0.725570	00:10
25	0.054709	0.031012	0.755533	00:10
26	0.054324	0.031428	0.747482	00:10
27	0.054275	0.035759	0.695024	00:10
28	0.054009	0.030831	0.746806	00:10
29	0.053853	0.029779	0.775632	00:10
30	0.053467	0.028992	0.783678	00:10
31	0.053183	0.030818	0.750974	00:10
32	0.052732	0.030063	0.763468	00:10
33	0.052621	0.031060	0.742135	00:10
34	0.052269	0.027699	0.794714	00:10
35	0.052146	0.027976	0.783296	00:10



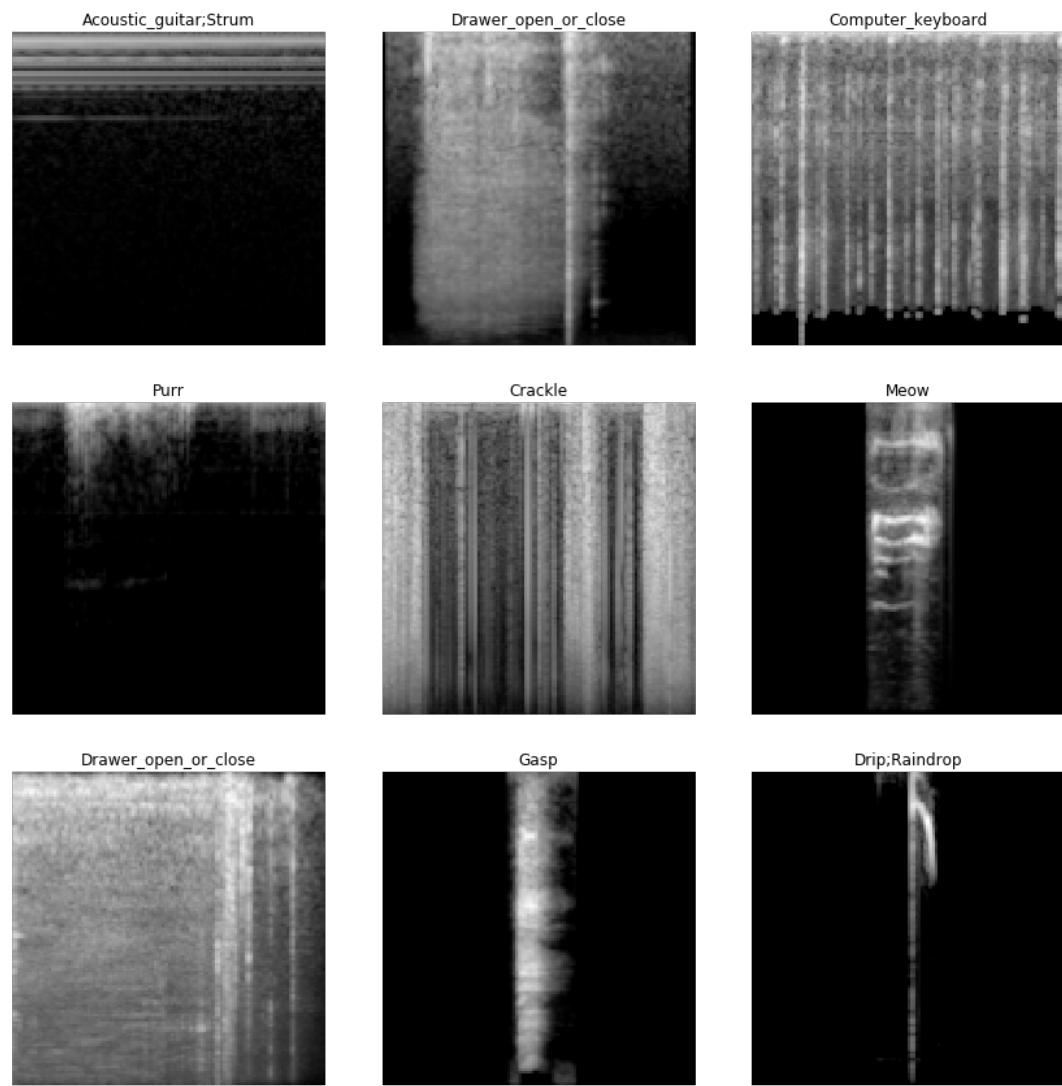
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us  
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo  
n't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

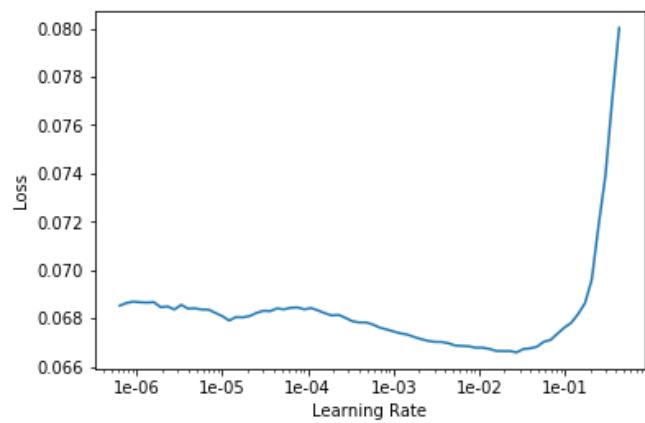
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us  
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo  
n't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

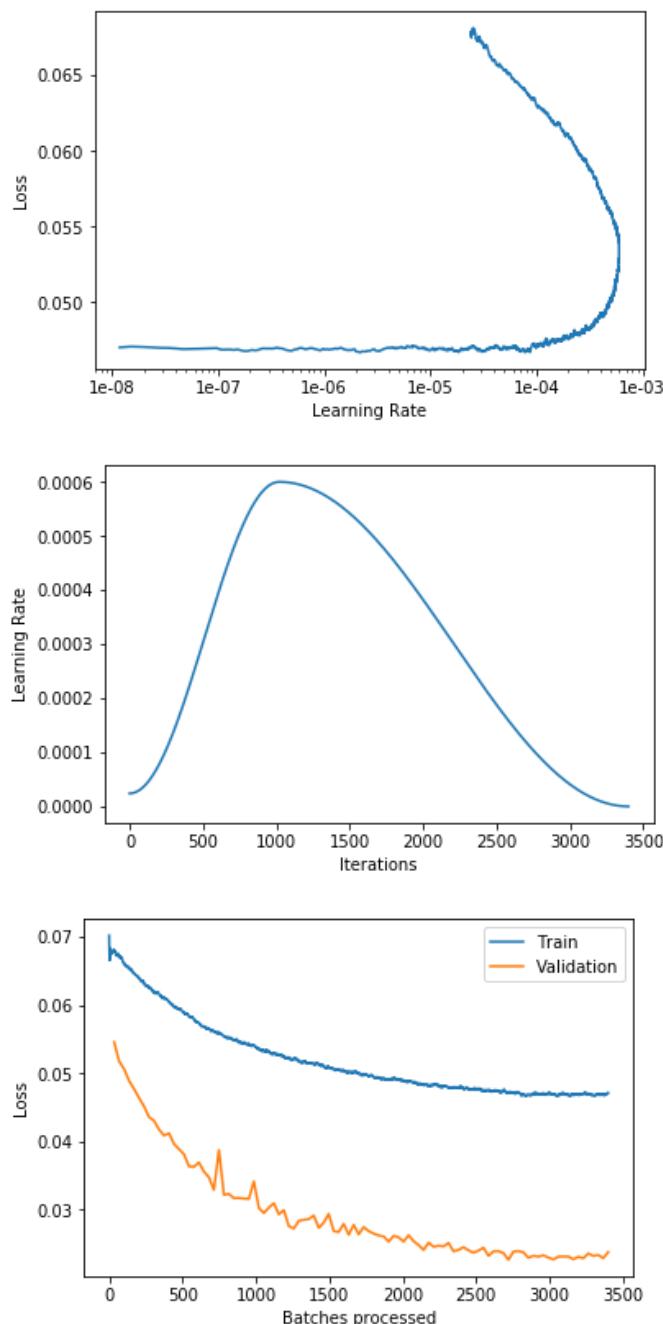
```
----- CURATED - Fold 10/10  
-----
```



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.068041	0.054592	0.431336	00:10
1	0.067200	0.051769	0.469170	00:10
2	0.066111	0.050633	0.498281	00:10
3	0.065422	0.048908	0.515635	00:10
4	0.064658	0.047710	0.545453	00:10
5	0.064010	0.046402	0.564189	00:10
6	0.063280	0.045171	0.570703	00:10
7	0.062639	0.043584	0.592577	00:10
8	0.061968	0.043056	0.610842	00:10
9	0.061616	0.041773	0.621353	00:10
10	0.061175	0.040893	0.636486	00:10
11	0.060449	0.041222	0.635485	00:10
12	0.059882	0.039649	0.656311	00:10
13	0.059422	0.038911	0.674440	00:10
14	0.058906	0.038139	0.677010	00:10
15	0.058438	0.036368	0.696898	00:10
16	0.057929	0.036312	0.708041	00:10
17	0.057390	0.036948	0.684182	00:10
18	0.056864	0.035591	0.705721	00:10
19	0.056587	0.034770	0.707753	00:10
20	0.056157	0.032930	0.743005	00:10
21	0.055942	0.038776	0.658227	00:10
22	0.055451	0.032206	0.755429	00:10
23	0.055286	0.032333	0.742987	00:10
24	0.054863	0.031708	0.749739	00:10
25	0.054647	0.031745	0.764983	00:10
26	0.054494	0.031657	0.751168	00:10
27	0.054353	0.031622	0.744292	00:10
28	0.054121	0.034176	0.712993	00:10
29	0.053511	0.030268	0.761421	00:10
30	0.053475	0.029541	0.776675	00:10
31	0.053310	0.030330	0.761848	00:10
32	0.052898	0.030978	0.746316	00:10
33	0.052711	0.029340	0.774208	00:10
34	0.052460	0.029959	0.772812	00:10
35	0.052201	0.027616	0.701869	00:10



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
"type " + obj.__name__ + ". It won't be checked "
```

```
In [27]: kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-2_fold-{fold}')
```

Overall lwlrmp on CURATED dataset: 0.8622168112364244

	lwlrap	weight
Fill_(with_liquid)	0.529234	0.008693
Squeak	0.611169	0.013039
Hiss	0.691345	0.013039
Chink_and_clink	0.705192	0.013039
Walk_and_footsteps	0.705671	0.013039
Gasp	0.714394	0.008345
Tap	0.739426	0.013039
Mechanical_fan	0.742898	0.008519
Trickle_and_dribble	0.747705	0.009214
Bus	0.756433	0.013039
Cutlery_and_silverware	0.758868	0.013039
Traffic_noise_and_roadway_noise	0.766698	0.013039
Clapping	0.768214	0.013039
Water_tap_and_faucet	0.770419	0.013039
Sink_(filling_or_washing)	0.777535	0.013039
Accelerating_and_revving_and_vroom	0.781042	0.013039
Dishes_and_pots_and_pans	0.792881	0.013039
Buzz	0.797091	0.009736
Yell	0.797680	0.013039
Frying_(food)	0.798048	0.010953
Microwave_oven	0.802672	0.013039
Male_speech_and_manSpeaking	0.804848	0.013039
Slam	0.805107	0.013039
Motorcycle	0.805333	0.013039
Cupboard_open_or_close	0.807495	0.013039
Bathtub_(filling_or_washing)	0.821460	0.013039
Knock	0.825413	0.013039
Scissors	0.827212	0.013039
Chewing_and_mastication	0.838138	0.013039
Waves_and_surf	0.838794	0.013039
...	...	...
Whispering	0.902607	0.013039
Raindrop	0.903333	0.013039
Gong	0.906794	0.013039
Bicycle_bell	0.916986	0.011648

```
In [28]: kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-2_fold-{fold}')
```

Overall lwlrap on NOISY dataset: 0.3814153195846499

	<b>lwlrap</b>	<b>weight</b>
Cupboard_open_or_close	0.050421	0.0125
Raindrop	0.062463	0.0125
Finger_snapping	0.064026	0.0125
Tap	0.078296	0.0125
Burping_and_eructation	0.091234	0.0125
Sigh	0.098085	0.0125
Keys_jangling	0.100882	0.0125
Shatter	0.101463	0.0125
Drip	0.115767	0.0125
Gasp	0.120609	0.0125
Fart	0.133108	0.0125
Slam	0.135386	0.0125
Strum	0.169422	0.0125
Knock	0.170624	0.0125
Glockenspiel	0.171884	0.0125
Trickle_and_dribble	0.184798	0.0125
Chink_and_clink	0.186886	0.0125
Bicycle_bell	0.191690	0.0125
Yell	0.195711	0.0125
Computer_keyboard	0.203171	0.0125
Zipper_(clothing)	0.213536	0.0125
Dishes_and_pots_and_pans	0.219779	0.0125
Scissors	0.229209	0.0125
Writing	0.230643	0.0125
Cutlery_and_silverware	0.231084	0.0125
Buzz	0.232530	0.0125
Run	0.234273	0.0125
Gong	0.234379	0.0125
Bass_guitar	0.236406	0.0125
Drawer_open_or_close	0.237253	0.0125
...	...	...
Accelerating_and_revving_and_vroom	0.467326	0.0125
Screaming	0.470764	0.0125
Frying_(food)	0.476772	0.0125
Child_speech_and_kidSpeaking	0.497841	0.0125

## Look for useful noisy samples

```
In [31]: def lwlrap_per_element(fname):
    overall_preds, overall_thruth, overall_index = _kfold_prediction(kf_noisy,
trn_noisy_df, fname)

    m = np.zeros_like(overall_preds)
    for i, (p, t) in enumerate(zip(overall_preds, overall_thruth)):
        idx, val = _one_sample_positive_class_precisions(p, t)
        m[i, idx] = val
    m = pd.DataFrame(m, columns=learn.data.classes, index=overall_index)

    m['fname'] = trn_noisy_df.fname

    for fold, (train_index, valid_index) in enumerate(kf_noisy.split(trn_noisy_
df)):
        m.at[valid_index, 'fold'] = fold

    return m
```

```
In [32]: m1 = lwlrap_per_element('stage-1_fold-{fold}')
```

```
In [33]: m2 = lwlrap_per_element('stage-2_fold-{fold}')
```

```
In [34]: m = m1.copy()
m[learn.data.classes] = np.sqrt(m1[learn.data.classes] * m2[learn.data.classe
s])
```

```
In [35]: del m1, m2
gc.collect();
```

```
In [36]: # Select samples from noisy dataset that look promising:  
# - suppose correct label from organiser  
# - high lwlrap on stage 1 (model with noisy elements only)  
# - high lwlrap on stage 2 (model warmed up with noisy elements and then fine tuned with curated dataset)  
  
lwlrap_threshold = .5  
count_limit = 5 # per fold and per class  
  
ok_noisy_items = []  
  
for i, x in enumerate(learn.data.classes):  
    print('\n', '*' * 20, x, '*' * 20)  
    print('Found', m[(m[x] >= lwlrap_threshold)].shape[0], 'promising element(s)')  
    for fold in range(n_splits):  
        selected = m[(m[x] >= lwlrap_threshold) & (m.fold == fold)].sort_values(x, ascending=False)[:count_limit].index  
        ok_noisy_items.extend(selected)  
        print('fold ', fold, ':', m[(m[x] >= lwlrap_threshold) & (m.fold == fold)].shape[0], m.iloc[selected].fname.tolist())  
  
print('\nFinal selection includes', len(ok_noisy_items), '"noisy" items')
```

```
***** Accelerating_and_revving_and_vroom *****
Found 148 promising element(s)
fold 0 : 15 ['08d8148a.wav', 'a4435a9d.wav', 'bddcf02f.wav', '34a8b8db.wav', '9076a3e2.wav']
fold 1 : 11 ['c092764a.wav', '7a22fec9.wav', 'f682f18c.wav', '3c2f3122.wav', '87fac7cf.wav']
fold 2 : 10 ['1b5fda28.wav', '8119602a.wav', '3d18e0a3.wav', 'e445db5e.wav', 'ffd1c0e2.wav']
fold 3 : 15 ['a0da98c4.wav', '33ab75d8.wav', '38b3d195.wav', 'b84be9e3.wav', 'a7063b6a.wav']
fold 4 : 17 ['f5a8204a.wav', 'a2ae680a.wav', 'a1e99e2b.wav', 'c5e0e576.wav', 'e79da000.wav']
fold 5 : 16 ['413f17c4.wav', 'ac830ed7.wav', 'e4298185.wav', '73bfc028.wav', 'ab1c9211.wav']
fold 6 : 15 ['0af6c5e5.wav', 'e5165478.wav', '1924aa99.wav', 'bc69bdfe.wav', 'd140606a.wav']
fold 7 : 19 ['049998bc.wav', '4d335487.wav', '49034bb5.wav', 'af229cd1.wav', 'e8d63fcb.wav']
fold 8 : 14 ['f13ae3d3.wav', 'c5e85231.wav', '2872af5d.wav', '750c435b.wav', '77b14052.wav']
fold 9 : 16 ['a6631a37.wav', 'f683db6f.wav', '9dab1ff8.wav', 'ad6c0e45.wav', 'f0014f2c.wav']

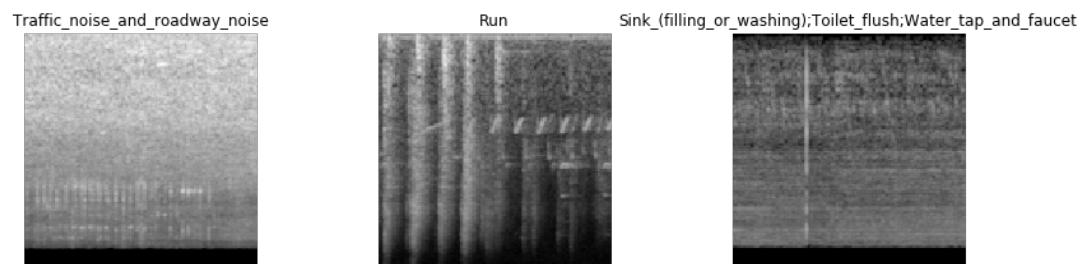
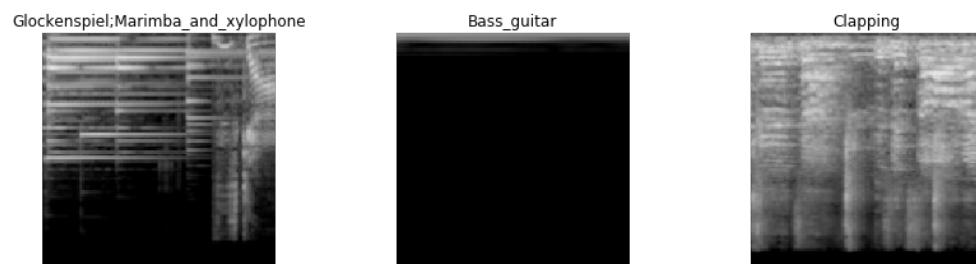
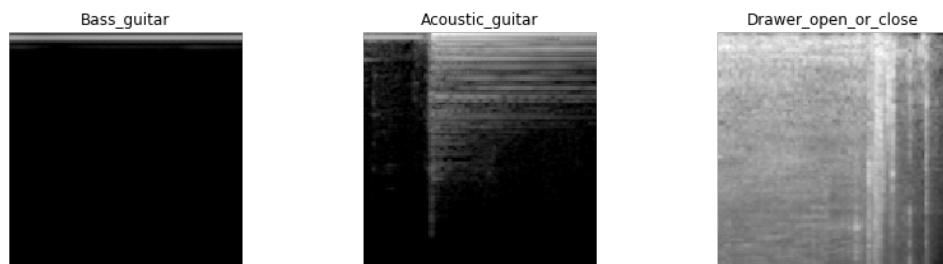
***** Accordion *****
Found 210 promising element(s)
fold 0 : 22 ['212a39e6.wav', 'e247cbd5.wav', '9ebd56b4.wav', 'd4d07f32.wav', 'd7ddf51c.wav']
fold 1 : 18 ['a274cb3a.wav', '3005b0ef.wav', '26ada8a4.wav', '20930ddf.wav', 'ec5ca64e.wav']
fold 2 : 22 ['1344ff56.wav', 'ec1de3c0.wav', 'dca727ef.wav', 'bccbc224.wav', '2d0fc86b.wav']
fold 3 : 21 ['4c84a00d.wav', '5805271e.wav', '83ae4438.wav', '0019adae.wav', '02ca6017.wav']
fold 4 : 27 ['226f3843.wav', '22978236.wav', '32273484.wav', 'da8769b0.wav', '56c3ebbe.wav']
fold 5 : 23 ['07c14838.wav', 'b2de494b.wav', '466f4543.wav', '5040d3b8.wav', 'dbaaef3b.wav']
fold 6 : 21 ['104ecb51.wav', 'd00893e2.wav', 'aea8a786.wav', '0c7bf9a1.wav', '68e95813.wav']
fold 7 : 18 ['49e35910.wav', 'a75a71e3.wav', '34364fff.wav', 'b019a385.wav', '978cfa10.wav']
fold 8 : 25 ['35d6ea77.wav', '48cc6b53.wav', 'd92c1a99.wav', '9c4199ce.wav', '29dc9519.wav']
fold 9 : 13 ['6e5ea69a.wav', 'ac9a944b.wav', '6c2622ca.wav', '3e9aec4c.wav', '8847ea41.wav']

***** Acoustic_guitar *****
Found 233 promising element(s)
fold 0 : 31 ['8da38b58.wav', '2cca4818.wav', '771f6fa2.wav', '92b289f3.wav', '971a8507.wav']
fold 1 : 23 ['49fda59f.wav', '5367af5c.wav', 'c2dfb020.wav', 'fabab3e5.wav', '6af5a226.wav']
fold 2 : 20 ['41d6f81b.wav', '228edfff.wav', '1beab2e8.wav', '839829ea.wav', '024cefba.wav']
fold 3 : 32 ['0a883fd0.wav', 'b3a4c530.wav', '282c241f.wav', 'e9fb76a.wav', 'fad9687c.wav']
fold 4 : 25 ['0f89d754.wav', '9b8fecd3.wav', '6eb972c0.wav', '400d1f0f.wav', '2f2af21e.wav']
fold 5 : 17 ['1562390d.wav', '1bd62d11.wav', '9a5927fc.wav', '19ff3d96.wav', 'dd696002.wav']
fold 6 : 22 ['0a46cdf6.wav', '7f52911f.wav', 'be2d26da.wav', 'e62a6277.wav', 'c029ca5c.wav']
fold 7 : 18 ['c1445256.wav', '3e2fd240.wav', '081fefc1.wav', 'ef668f97.wav', '
```

**Train on Selected Noisy + Curated**

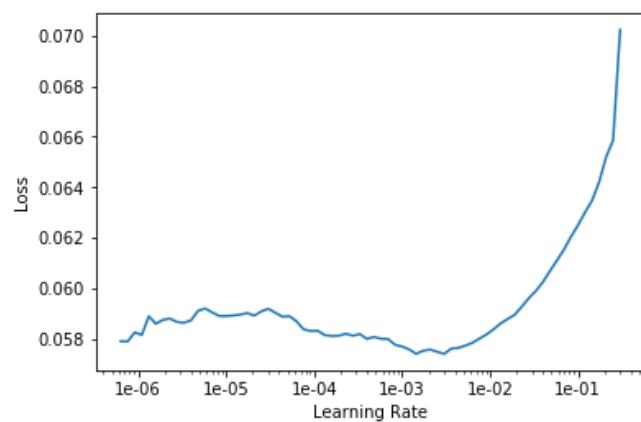
```
In [70]: for fold, ((train_index1, valid_index1), (train_index2, valid_index2)) in enumerate(zip(kf_curated.split(trn_curated_df), kf_noisy.split(trn_noisy_df))):  
    print('-' * 40, f'CURATED+NOISY - Fold {fold+1}/{n_splits}', '-' * 40)  
  
    train_index2 = list(set(train_index2).intersection(set(ok_noisy_items)))  
  
    mix_df = pd.concat([  
        trn_curated_df.iloc[train_index1],  
        trn_noisy_df.iloc[train_index2],  
        trn_curated_df.iloc[valid_index1],  
#         trn_noisy_df.iloc[valid_index2], # compute lwlrap only on curated  
    ], ignore_index=True)  
    train_index = mix_df[:len(train_index1)+len(train_index2)].index  
    valid_index = mix_df[len(train_index1)+len(train_index2):].index  
  
    src = (ImageList.from_df(mix_df, WORK)  
        .split_by_idxs(train_index, valid_index)  
        .label_from_df(label_delim=',')  
    )  
    data = (src.transform(tfms, size=128)  
        .databunch(bs=bs)  
    )  
    data.show_batch(3)  
    plt.show()  
  
    learn.load(f'stage-2_fold-{fold}')  
    learn.data = data  
  
#     learning(learn, 100, 1e-2)  
    learning(learn, 300, 2e-3)  
  
    learn.save(f'stage-10_fold-{fold}')  
    learn.export(f'stage-10_fold-{fold}.pkl')  
  
    if TOY_MODE:  
        break
```

----- CURATED+NOISY - Fold 1/10 -----

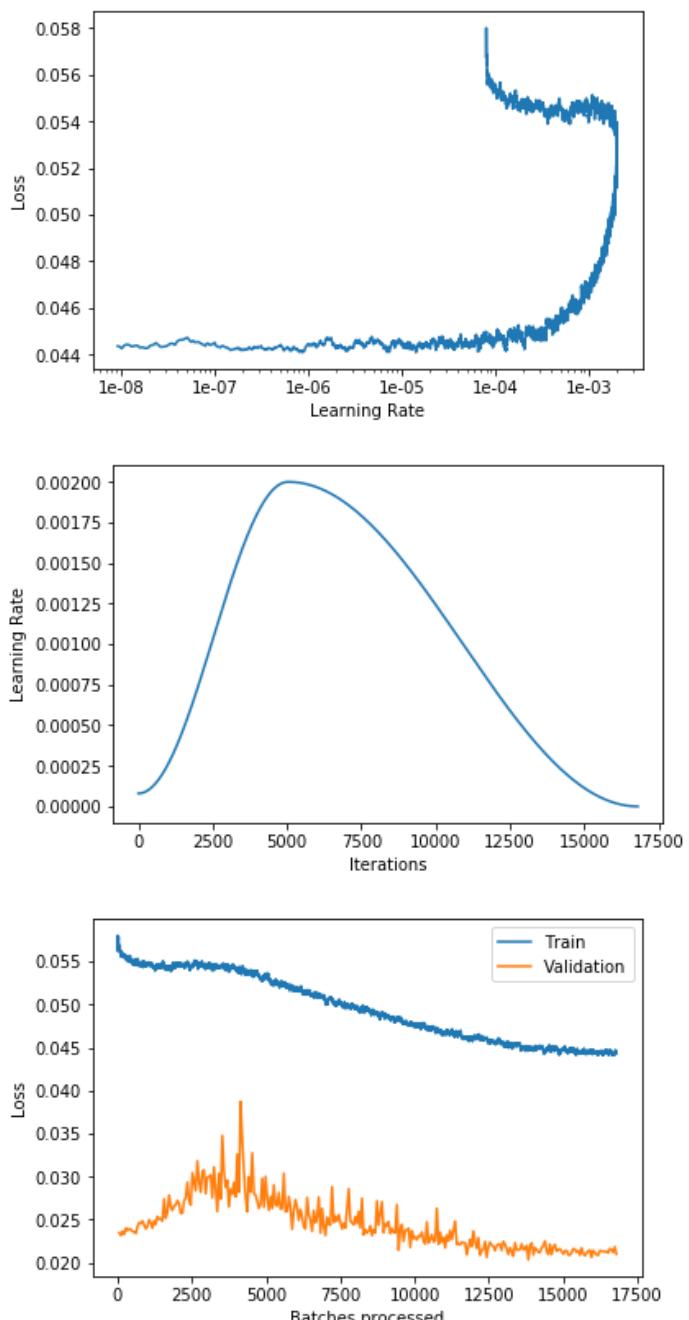


```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

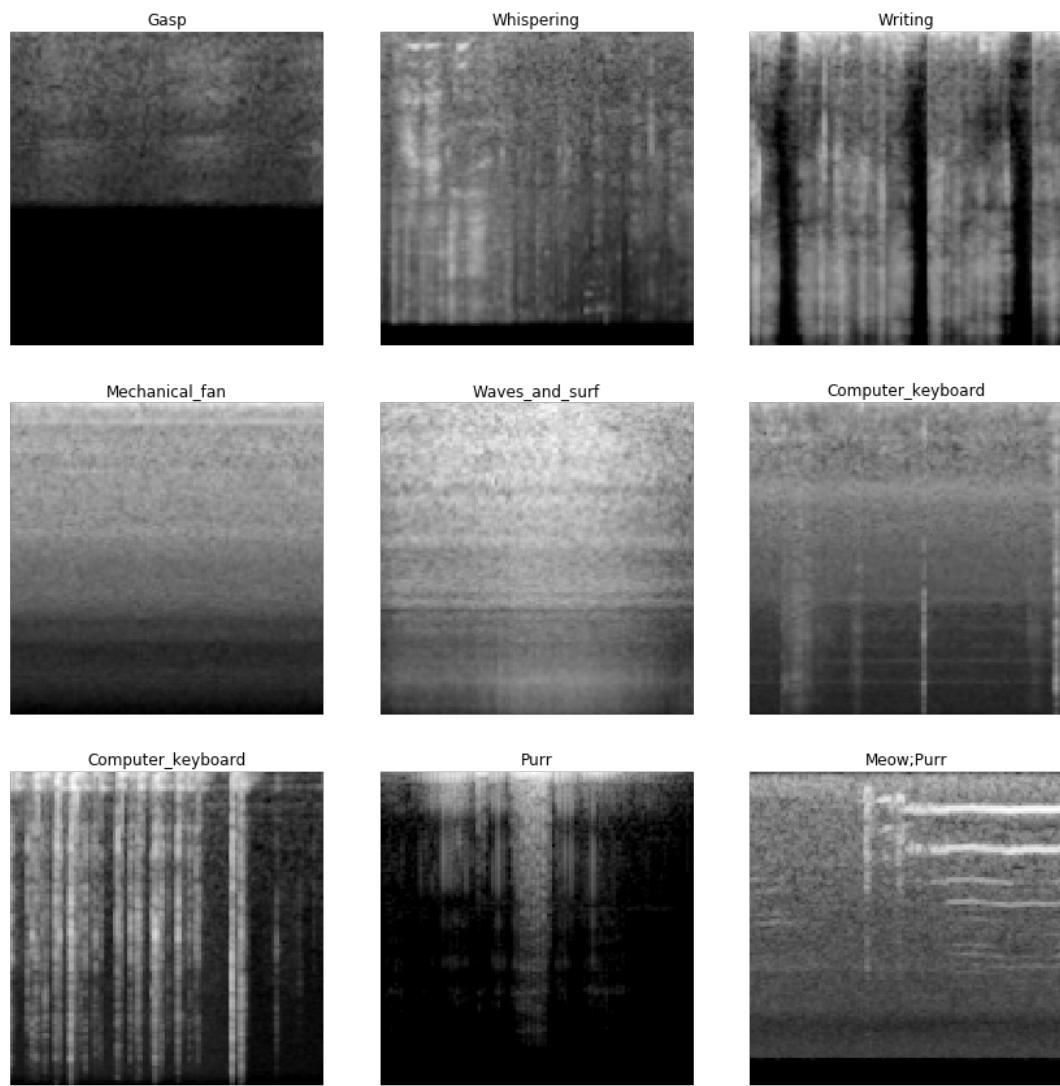


epoch	train_loss	valid_loss	lwlrap	time
0	0.056604	0.023484	0.811862	00:18
1	0.055824	0.023179	0.821491	00:17
2	0.055727	0.023649	0.820743	00:17
3	0.055610	0.023286	0.813921	00:17
4	0.055363	0.024015	0.816151	00:17
5	0.055419	0.023756	0.816400	00:17
6	0.055244	0.023944	0.819504	00:17
7	0.054968	0.023840	0.810626	00:17
8	0.054976	0.023633	0.825156	00:17
9	0.054969	0.023567	0.815957	00:18
10	0.055044	0.023409	0.818621	00:17
11	0.055191	0.024201	0.808950	00:17
12	0.054721	0.024662	0.811694	00:18
13	0.054868	0.024547	0.811912	00:18
14	0.054825	0.024847	0.809213	00:18
15	0.054776	0.024385	0.814918	00:18
16	0.054654	0.023932	0.817674	00:18
17	0.054662	0.024234	0.815602	00:18
18	0.054714	0.024977	0.799796	00:18
19	0.054527	0.024642	0.804987	00:18
20	0.054593	0.024011	0.815292	00:18
21	0.054483	0.024874	0.807748	00:18
22	0.054375	0.024457	0.819575	00:18
23	0.054525	0.025371	0.805249	00:18
24	0.054505	0.025089	0.800842	00:18
25	0.054369	0.025036	0.800132	00:18
26	0.054763	0.024857	0.808564	00:18
27	0.054543	0.027397	0.766538	00:18
28	0.054307	0.025168	0.802380	00:18
29	0.054243	0.025880	0.794424	00:18
30	0.054294	0.027845	0.770591	00:18
31	0.054263	0.026558	0.788059	00:18
32	0.054482	0.026077	0.801397	00:18
33	0.054334	0.026620	0.786540	00:18
34	0.054627	0.026690	0.784165	00:18
35	0.054800	0.027208	0.776085	00:18



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
  "type " + obj.__name__ + ". It won't be checked "
```

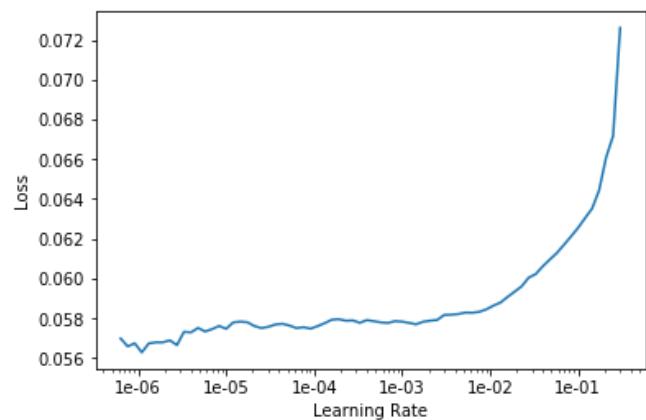
```
----- CURATED+NOISY - Fold 2/10
-----
```



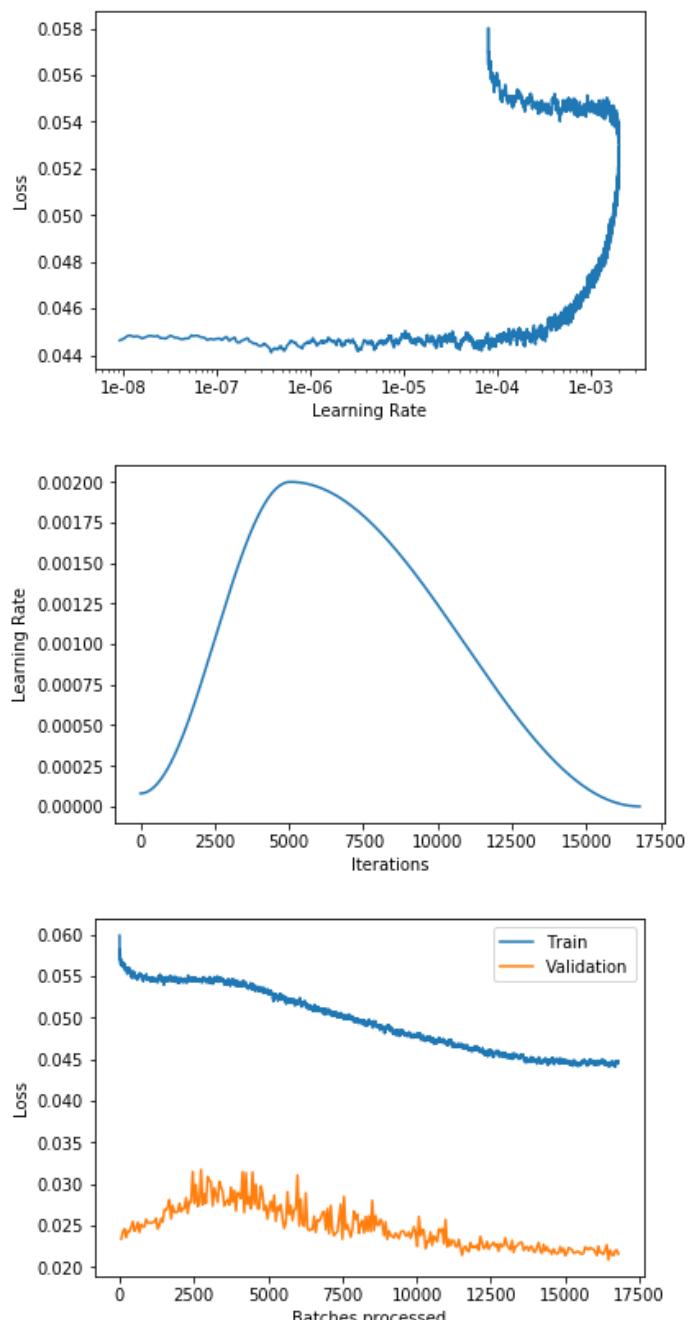
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

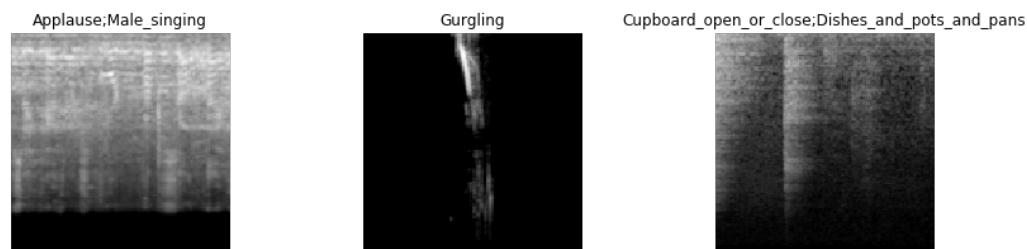
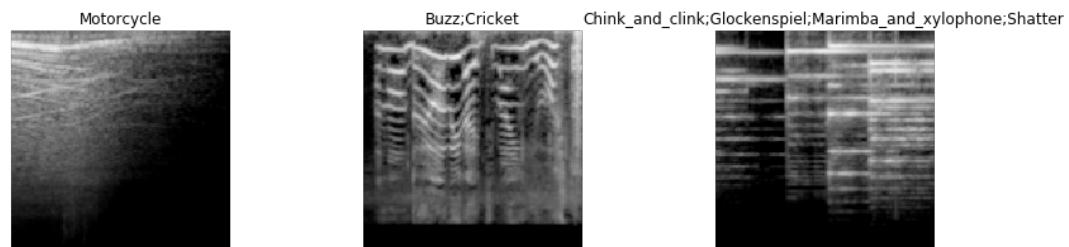
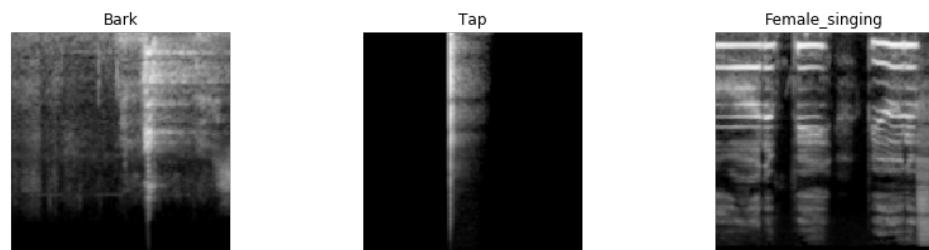


epoch	train_loss	valid_loss	lwlrap	time
0	0.056707	0.023416	0.821857	00:17
1	0.056648	0.024257	0.811793	00:17
2	0.056150	0.024614	0.816326	00:17
3	0.055908	0.023577	0.828864	00:17
4	0.055853	0.024340	0.818745	00:17
5	0.055869	0.024293	0.820242	00:17
6	0.055157	0.024842	0.822522	00:17
7	0.055096	0.025487	0.816613	00:17
8	0.055041	0.024343	0.824565	00:17
9	0.055022	0.024713	0.811780	00:17
10	0.054988	0.025404	0.802417	00:17
11	0.055204	0.024199	0.824800	00:17
12	0.055112	0.025144	0.808049	00:17
13	0.054655	0.024894	0.816461	00:17
14	0.054943	0.024867	0.821521	00:17
15	0.055075	0.025771	0.801954	00:17
16	0.054797	0.025316	0.810659	00:17
17	0.054541	0.025338	0.819460	00:17
18	0.054653	0.025285	0.808083	00:17
19	0.054711	0.025426	0.810744	00:17
20	0.054622	0.025442	0.809361	00:17
21	0.054588	0.025862	0.801844	00:17
22	0.054405	0.024616	0.823401	00:17
23	0.054908	0.025059	0.819691	00:17
24	0.054747	0.025832	0.805237	00:17
25	0.054454	0.025448	0.804195	00:17
26	0.054392	0.026953	0.799237	00:17
27	0.054777	0.026869	0.778941	00:17
28	0.054583	0.027256	0.793989	00:17
29	0.054892	0.028114	0.785666	00:17
30	0.054729	0.025838	0.810791	00:17
31	0.054765	0.027396	0.782952	00:17
32	0.054728	0.026628	0.797135	00:17
33	0.054493	0.026624	0.800358	00:17
34	0.054391	0.026652	0.786857	00:17
35	0.054667	0.027661	0.788293	00:17



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

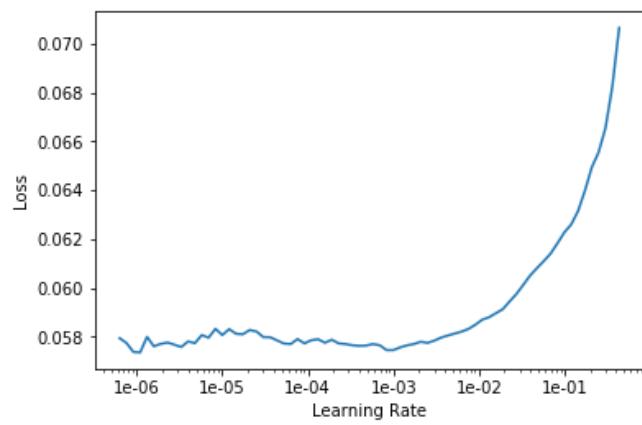
```
----- CURATED+NOISY - Fold 3/10  
-----
```



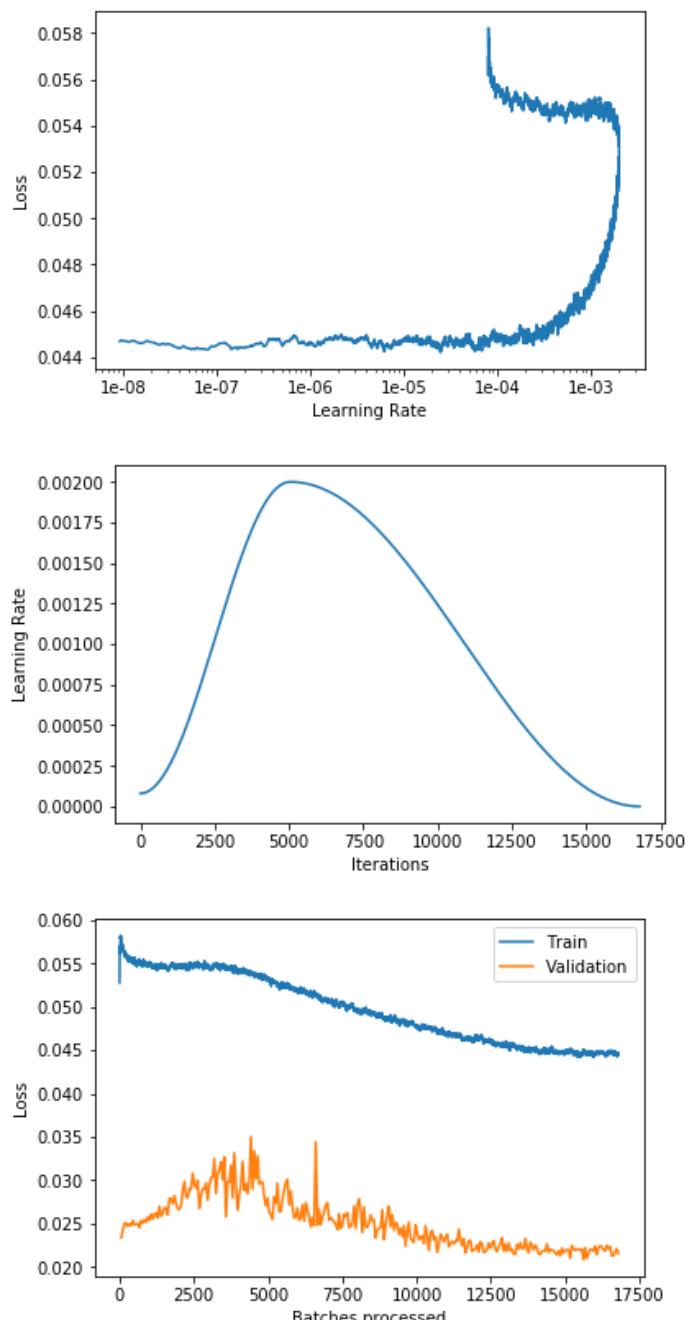
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

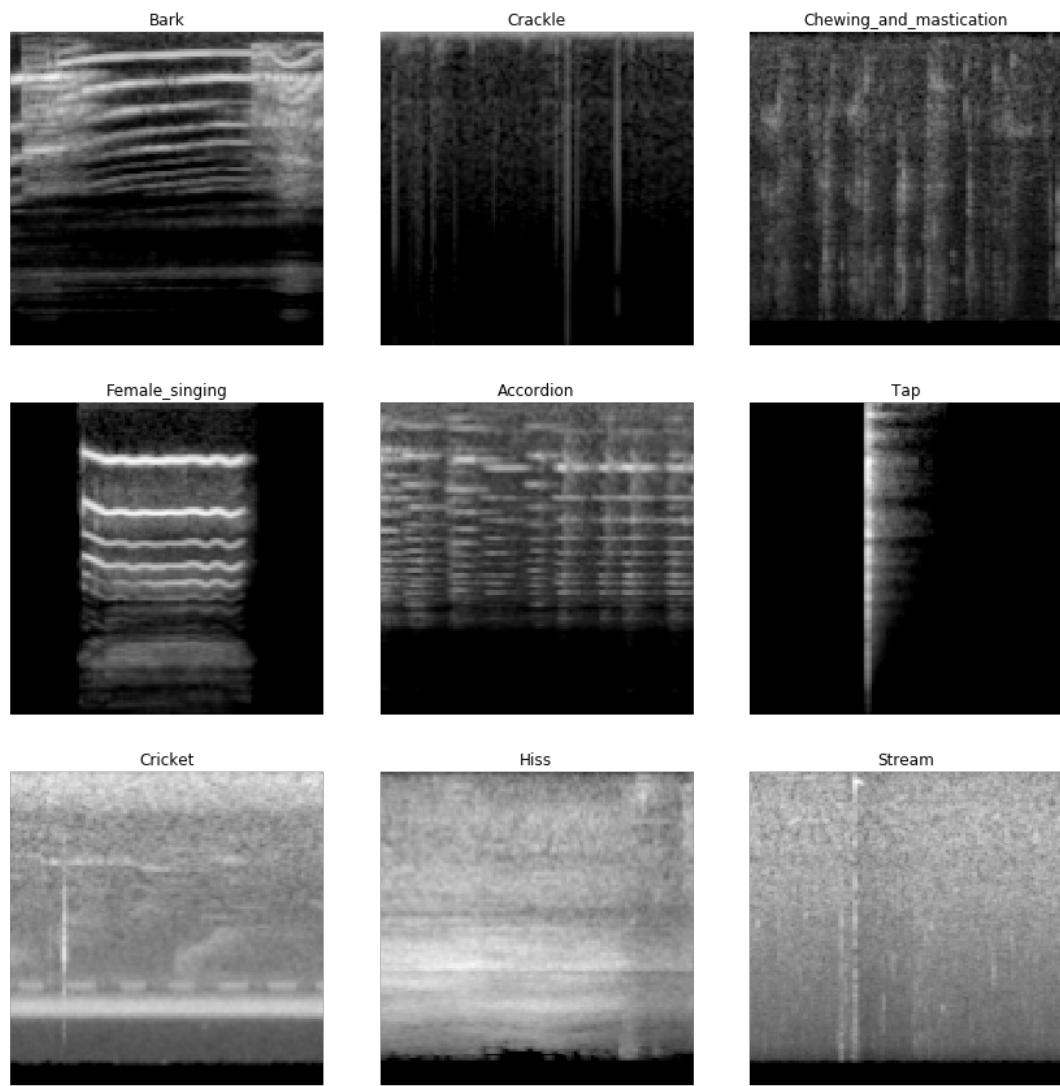


epoch	train_loss	valid_loss	lwlrap	time
0	0.057502	0.023360	0.818330	00:17
1	0.057100	0.024247	0.818254	00:17
2	0.056134	0.025028	0.808396	00:17
3	0.055889	0.025061	0.799858	00:17
4	0.055740	0.024668	0.804790	00:17
5	0.055769	0.024974	0.811054	00:17
6	0.055336	0.024759	0.805660	00:17
7	0.055091	0.025346	0.805887	00:17
8	0.055481	0.024860	0.805014	00:17
9	0.055119	0.024897	0.804919	00:17
10	0.055215	0.025002	0.806035	00:17
11	0.055320	0.024539	0.816623	00:17
12	0.055165	0.025330	0.806490	00:17
13	0.054951	0.025070	0.808153	00:17
14	0.055103	0.025476	0.809867	00:17
15	0.054799	0.025157	0.812653	00:17
16	0.055048	0.025751	0.803237	00:17
17	0.055135	0.025390	0.801261	00:17
18	0.054810	0.026091	0.798326	00:17
19	0.054715	0.025500	0.802273	00:17
20	0.054551	0.026117	0.794076	00:17
21	0.054561	0.026244	0.790477	00:17
22	0.054592	0.025659	0.797907	00:17
23	0.054795	0.026998	0.781375	00:17
24	0.054968	0.025886	0.801843	00:17
25	0.054735	0.026216	0.802008	00:17
26	0.054750	0.026850	0.794228	00:17
27	0.054474	0.026758	0.793077	00:17
28	0.054434	0.026011	0.805623	00:17
29	0.054600	0.027921	0.777744	00:17
30	0.054817	0.027841	0.785104	00:17
31	0.054807	0.027288	0.767724	00:17
32	0.054834	0.028195	0.781067	00:17
33	0.054609	0.027269	0.779361	00:17
34	0.054574	0.026687	0.799070	00:17
35	0.054748	0.026657	0.799379	00:17



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
  "type " + obj.__name__ + ". It won't be checked "
```

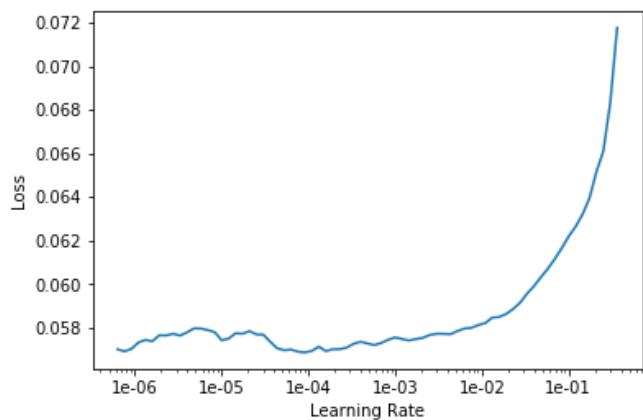
```
----- CURATED+NOISY - Fold 4/10
-----
```



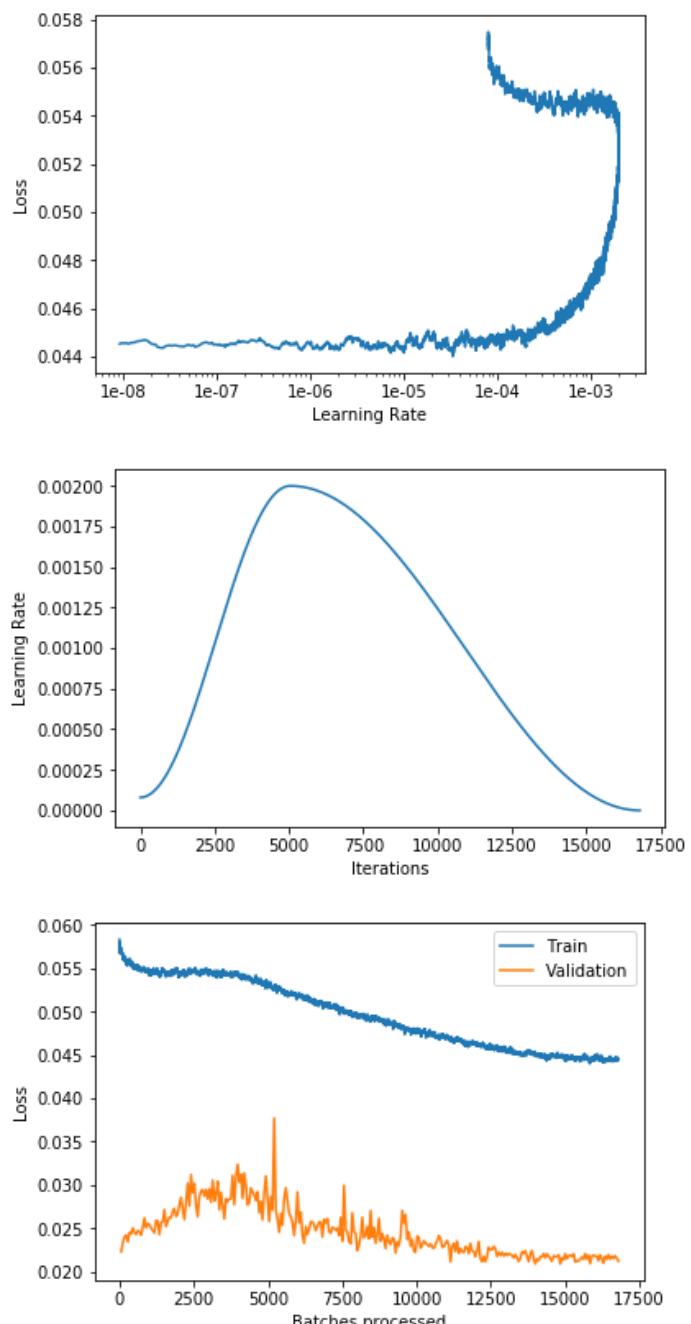
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.057186	0.022306	0.824730	00:17
1	0.056248	0.023357	0.812927	00:17
2	0.056218	0.024014	0.812432	00:17
3	0.055755	0.024223	0.813763	00:17
4	0.055759	0.023448	0.821557	00:17
5	0.055774	0.024510	0.806825	00:17
6	0.055508	0.024389	0.814173	00:17
7	0.055471	0.025028	0.801672	00:17
8	0.055264	0.024518	0.813174	00:17
9	0.055203	0.024290	0.812464	00:17
10	0.054880	0.024739	0.814820	00:17
11	0.054778	0.024579	0.804828	00:17
12	0.054778	0.024234	0.818273	00:17
13	0.054856	0.025136	0.796899	00:17
14	0.055022	0.026203	0.793266	00:17
15	0.054717	0.025104	0.810235	00:17
16	0.054784	0.025531	0.797754	00:17
17	0.054429	0.025429	0.804412	00:17
18	0.054648	0.025033	0.815644	00:17
19	0.054718	0.024749	0.810553	00:17
20	0.054489	0.024488	0.811980	00:17
21	0.054440	0.025901	0.801981	00:17
22	0.054708	0.025843	0.796201	00:17
23	0.054375	0.024296	0.823037	00:17
24	0.054675	0.025675	0.804891	00:17
25	0.054943	0.025777	0.792132	00:17
26	0.054728	0.027168	0.786867	00:17
27	0.054359	0.025235	0.810031	00:17
28	0.054527	0.025831	0.798825	00:17
29	0.054610	0.026276	0.791537	00:17
30	0.054728	0.026245	0.801096	00:17
31	0.054628	0.026944	0.782299	00:17
32	0.054587	0.027277	0.777994	00:17
33	0.054807	0.026389	0.791308	00:17
34	0.054606	0.026680	0.797277	00:17
35	0.054436	0.027746	0.774035	00:17



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
"type " + obj.__name__ + ". It won't be checked "
```

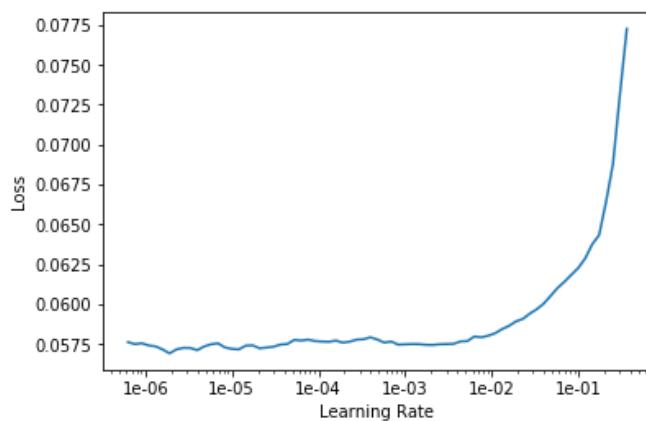
```
----- CURATED+NOISY - Fold 5/10
-----
```



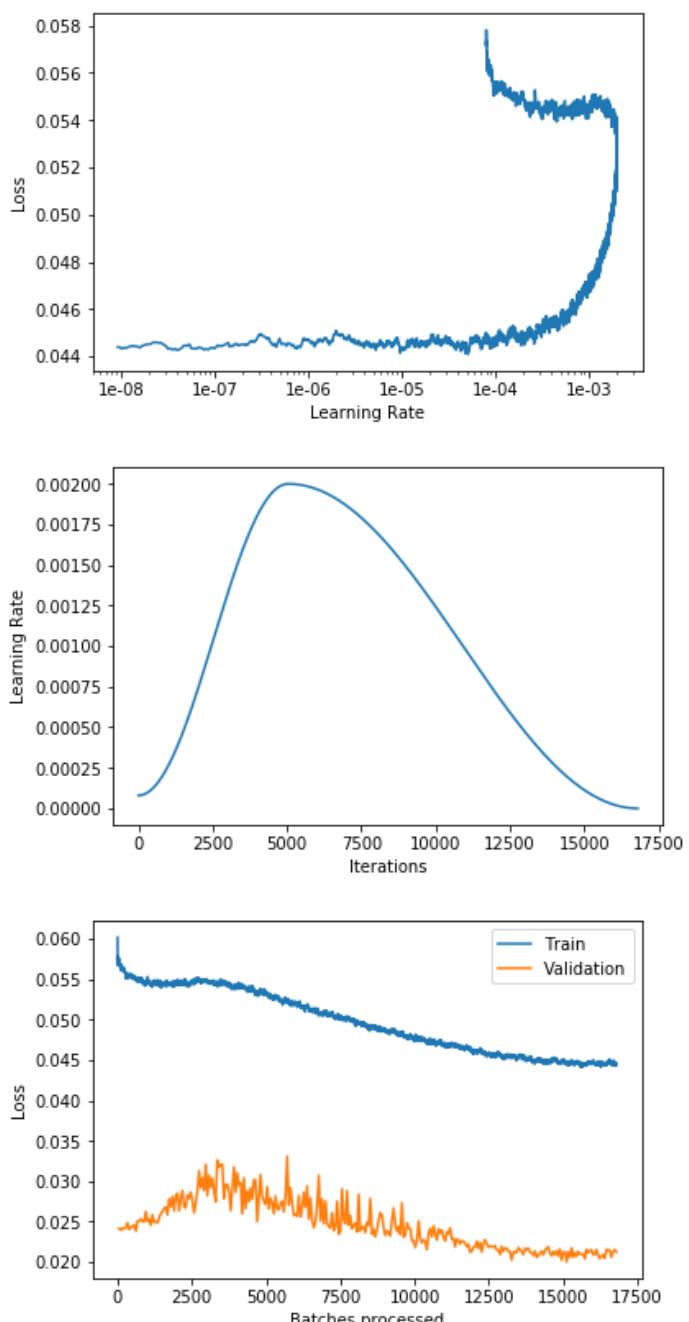
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

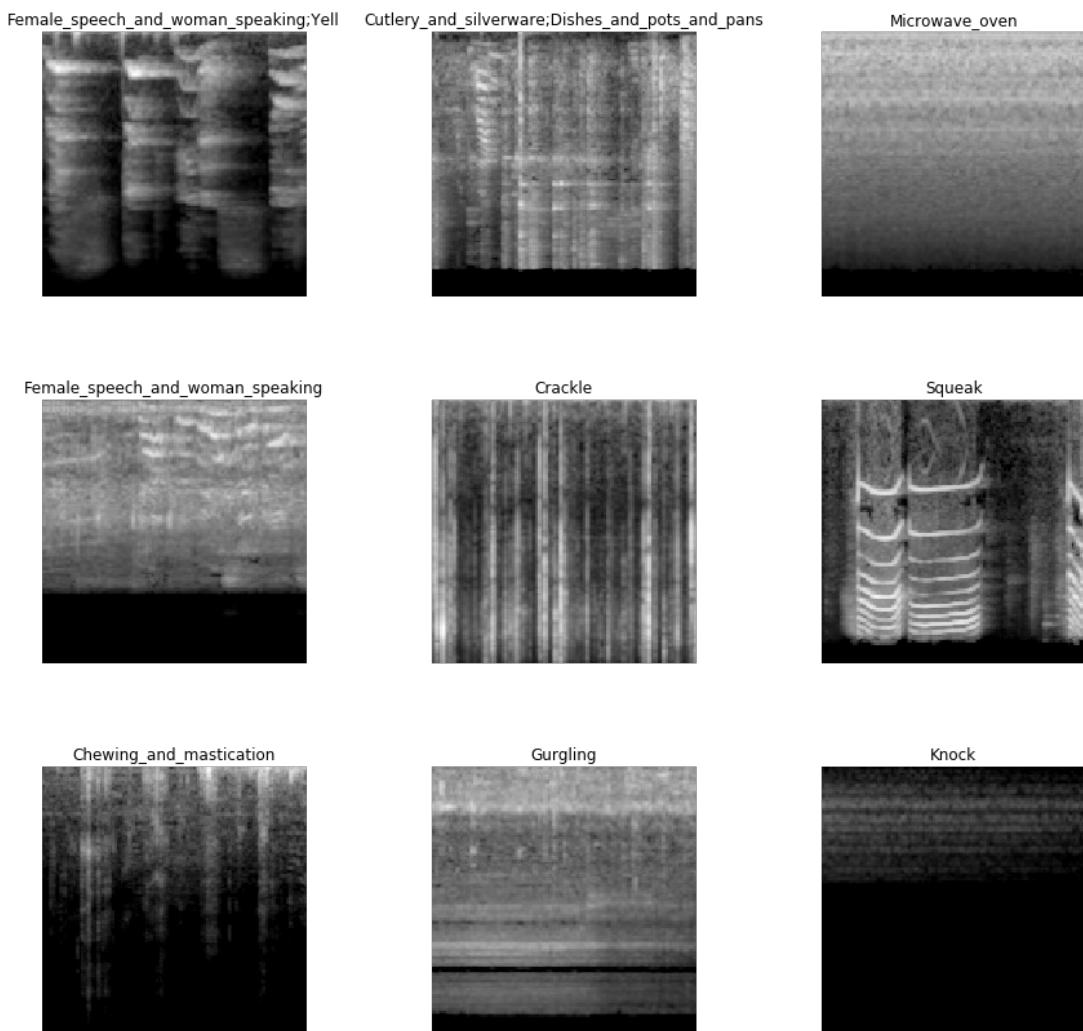


epoch	train_loss	valid_loss	lwlrap	time
0	0.057044	0.024161	0.819230	00:17
1	0.056628	0.023958	0.830739	00:17
2	0.056272	0.024250	0.821069	00:17
3	0.056022	0.024125	0.831095	00:17
4	0.055452	0.024302	0.829438	00:17
5	0.055468	0.024866	0.817254	00:17
6	0.055357	0.024041	0.839131	00:17
7	0.055482	0.024264	0.832108	00:17
8	0.055228	0.024438	0.819041	00:17
9	0.055088	0.024613	0.827360	00:17
10	0.055054	0.023857	0.838216	00:17
11	0.054902	0.025098	0.823029	00:17
12	0.054766	0.025086	0.826395	00:17
13	0.055105	0.025347	0.822700	00:17
14	0.054715	0.025038	0.822133	00:17
15	0.054626	0.025014	0.822617	00:17
16	0.054750	0.026255	0.807017	00:17
17	0.054853	0.025037	0.821507	00:17
18	0.054549	0.025315	0.815385	00:17
19	0.054353	0.024581	0.831012	00:17
20	0.054459	0.025855	0.817802	00:17
21	0.054574	0.024912	0.824342	00:17
22	0.054565	0.025037	0.818775	00:17
23	0.054825	0.024859	0.825060	00:17
24	0.054490	0.025494	0.828124	00:17
25	0.054423	0.025969	0.807472	00:17
26	0.054421	0.026251	0.809753	00:17
27	0.054582	0.026633	0.811536	00:17
28	0.054322	0.026808	0.804764	00:17
29	0.054424	0.025862	0.809785	00:17
30	0.054499	0.027805	0.796188	00:17
31	0.054501	0.026869	0.793412	00:17
32	0.054589	0.028116	0.787738	00:17
33	0.054662	0.026340	0.822783	00:17
34	0.054493	0.028188	0.787466	00:17
35	0.054675	0.028502	0.791758	00:17



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

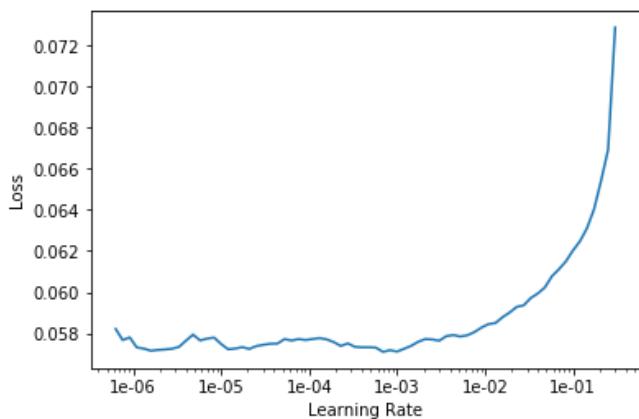
```
----- CURATED+NOISY - Fold 6/10  
-----
```



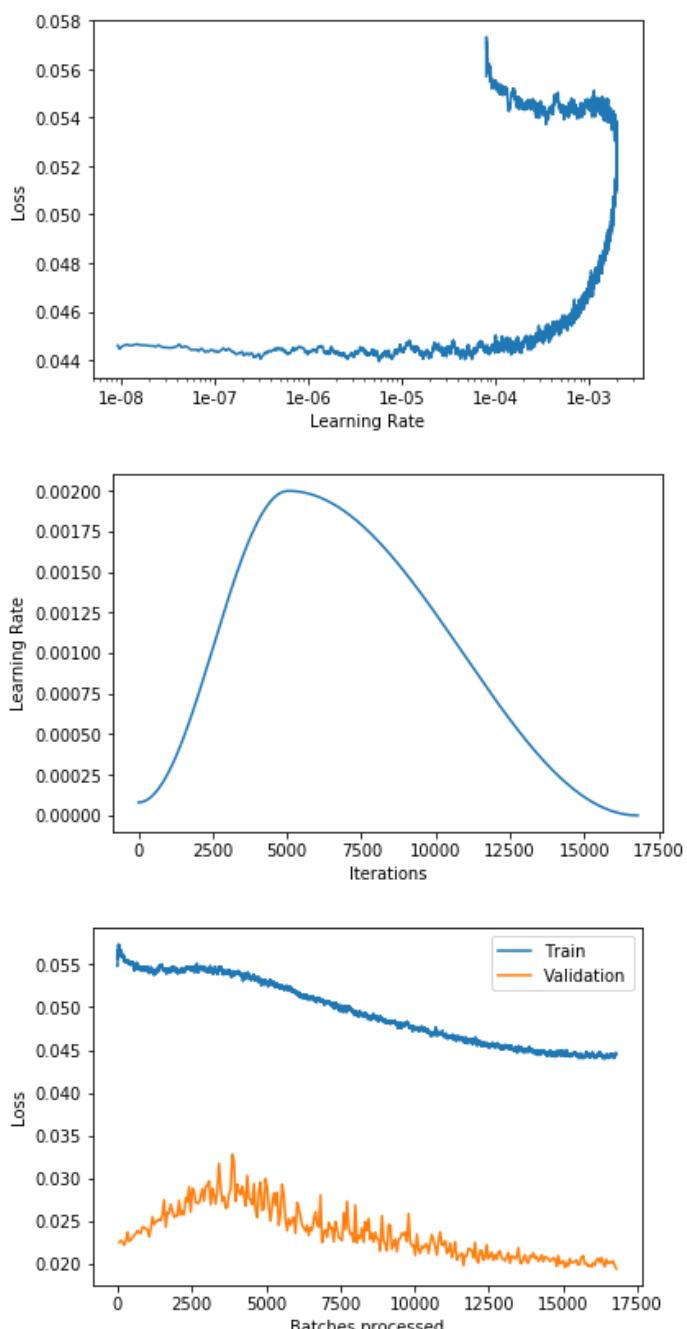
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

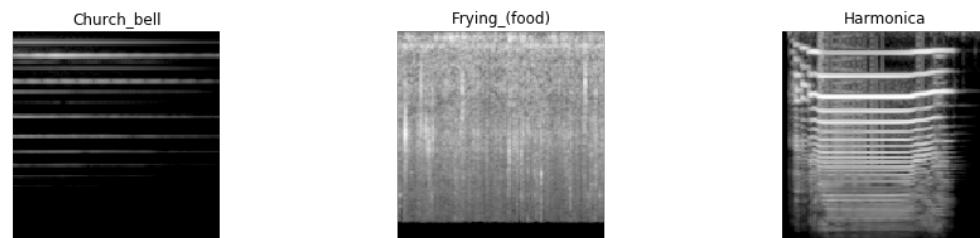
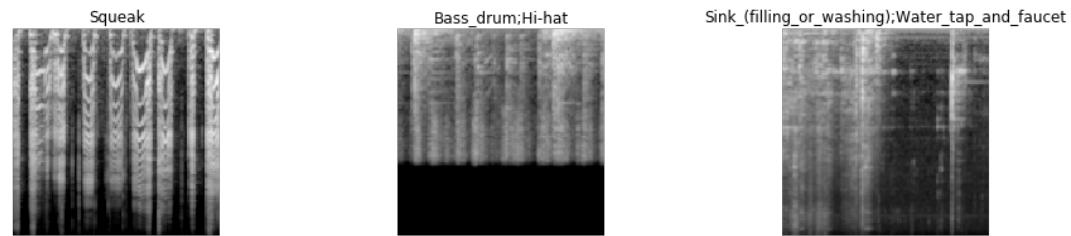
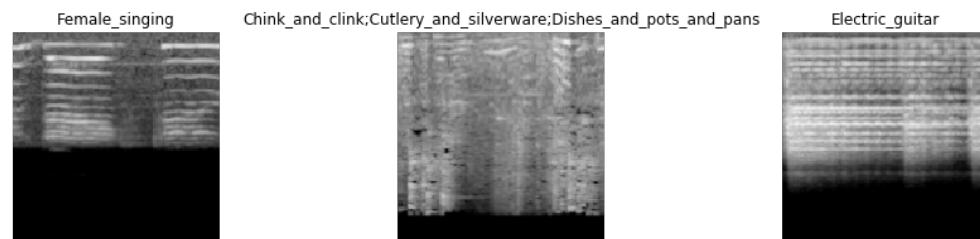


epoch	train_loss	valid_loss	lwlrap	time
0	0.056882	0.022539	0.837351	00:17
1	0.056334	0.022753	0.834765	00:17
2	0.056070	0.022571	0.839513	00:17
3	0.055645	0.022225	0.839605	00:17
4	0.055457	0.022675	0.834350	00:17
5	0.055283	0.023741	0.826357	00:17
6	0.055228	0.022678	0.833135	00:17
7	0.055197	0.022823	0.831195	00:17
8	0.055075	0.023058	0.835461	00:17
9	0.054570	0.023381	0.827844	00:17
10	0.054983	0.023564	0.832060	00:17
11	0.054803	0.023902	0.828830	00:17
12	0.054617	0.023667	0.830768	00:17
13	0.054422	0.023609	0.829311	00:17
14	0.054496	0.024373	0.822691	00:17
15	0.054654	0.024304	0.818046	00:17
16	0.054542	0.024086	0.834180	00:17
17	0.054557	0.024068	0.817891	00:17
18	0.054475	0.023203	0.828162	00:17
19	0.054690	0.024142	0.826804	00:17
20	0.054204	0.025519	0.809055	00:17
21	0.053957	0.024611	0.820192	00:17
22	0.054252	0.024952	0.821863	00:17
23	0.054403	0.025040	0.817996	00:17
24	0.054774	0.025258	0.818975	00:17
25	0.054859	0.025094	0.815174	00:17
26	0.054407	0.025808	0.808205	00:17
27	0.054377	0.027488	0.771583	00:17
28	0.054513	0.024333	0.820771	00:17
29	0.054299	0.025995	0.796920	00:17
30	0.054222	0.025952	0.805655	00:17
31	0.054490	0.026995	0.779905	00:17
32	0.054669	0.026277	0.810330	00:17
33	0.054529	0.025480	0.802843	00:17
34	0.054539	0.025691	0.799920	00:17
35	0.054158	0.025629	0.803171	00:17



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
  "type " + obj.__name__ + ". It won't be checked "
```

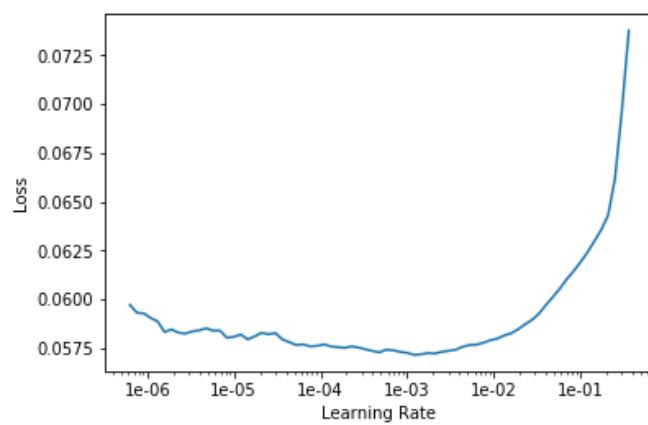
```
----- CURATED+NOISY - Fold 7/10
-----
```



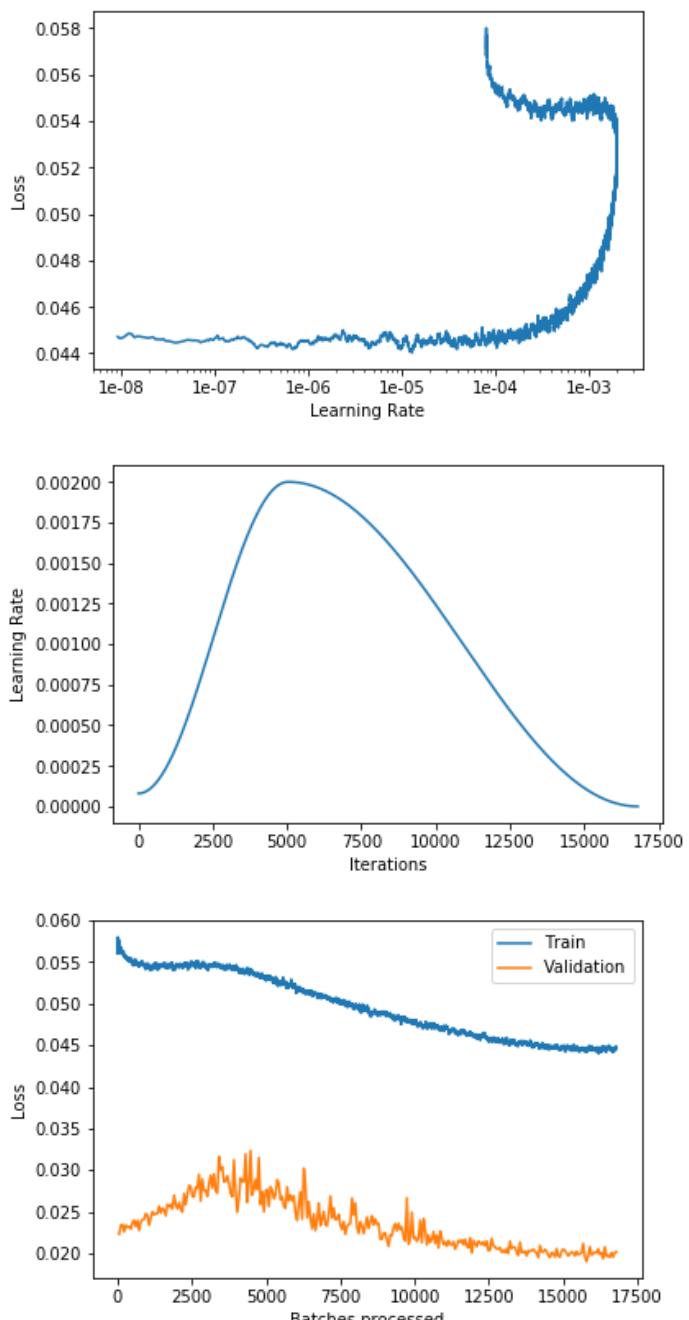
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

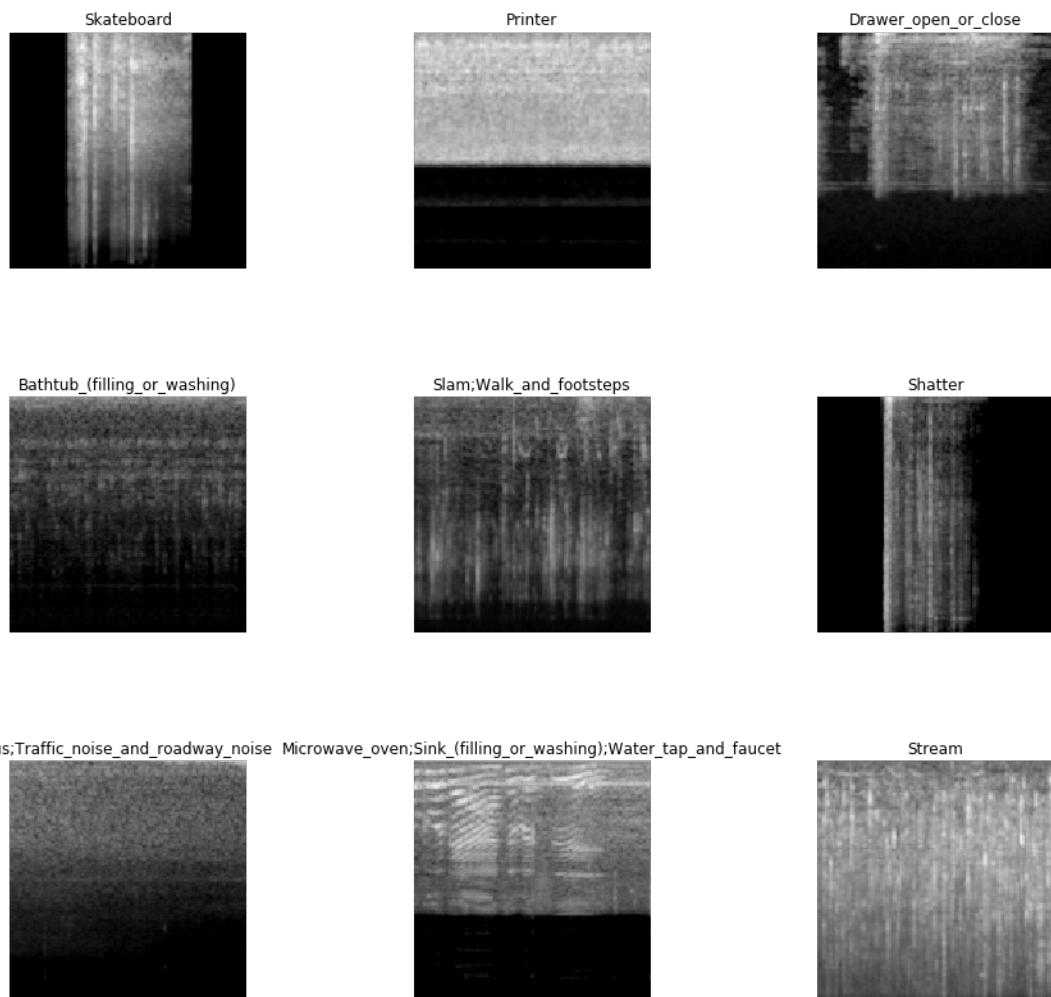


epoch	train_loss	valid_loss	lwlrap	time
0	0.057316	0.022329	0.841667	00:17
1	0.056535	0.023391	0.831719	00:17
2	0.056074	0.023455	0.823810	00:17
3	0.055881	0.022588	0.841351	00:17
4	0.055549	0.023377	0.840897	00:17
5	0.055516	0.023153	0.847745	00:17
6	0.055425	0.023215	0.836914	00:17
7	0.055237	0.022940	0.840258	00:17
8	0.054918	0.023720	0.831887	00:18
9	0.055017	0.023964	0.837969	00:17
10	0.054964	0.023208	0.836657	00:17
11	0.054711	0.023957	0.837246	00:17
12	0.054851	0.022766	0.844262	00:17
13	0.054784	0.023786	0.828069	00:17
14	0.054913	0.024061	0.831984	00:17
15	0.054534	0.024112	0.827900	00:17
16	0.054390	0.024770	0.818718	00:17
17	0.054582	0.024587	0.829297	00:17
18	0.054415	0.023402	0.836558	00:17
19	0.054346	0.024272	0.819859	00:18
20	0.054604	0.025010	0.816251	00:18
21	0.054560	0.024664	0.834763	00:18
22	0.054583	0.024794	0.822857	00:18
23	0.054315	0.024781	0.839641	00:18
24	0.054500	0.025651	0.819550	00:18
25	0.054492	0.024854	0.826998	00:18
26	0.054614	0.025433	0.829694	00:18
27	0.054596	0.025231	0.829036	00:18
28	0.054345	0.024533	0.824570	00:18
29	0.054352	0.025874	0.819821	00:18
30	0.054702	0.025617	0.808194	00:18
31	0.054632	0.025888	0.809149	00:18
32	0.054458	0.025350	0.817960	00:18
33	0.054350	0.026982	0.785578	00:18
34	0.054639	0.025875	0.808531	00:18
35	0.054553	0.025367	0.815063	00:18



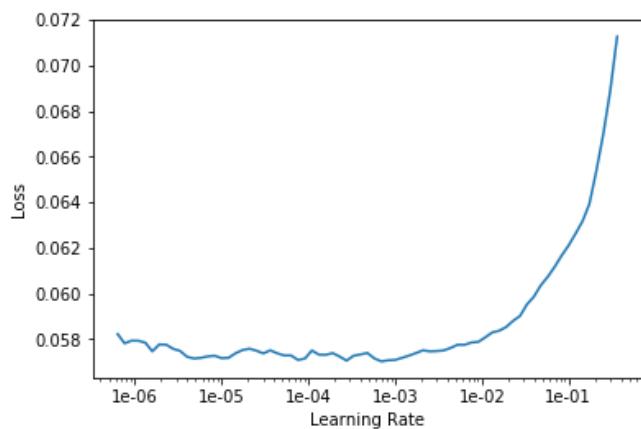
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

```
----- CURATED+NOISY - Fold 8/10  
-----
```

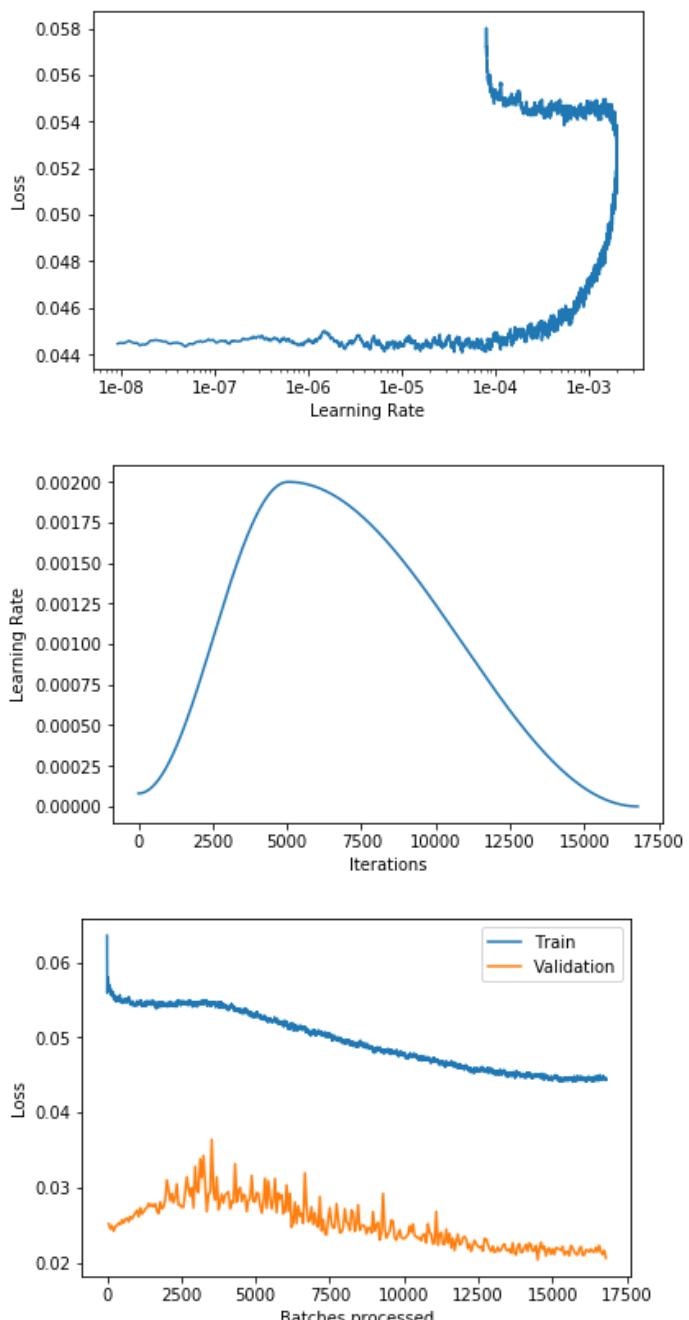


```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

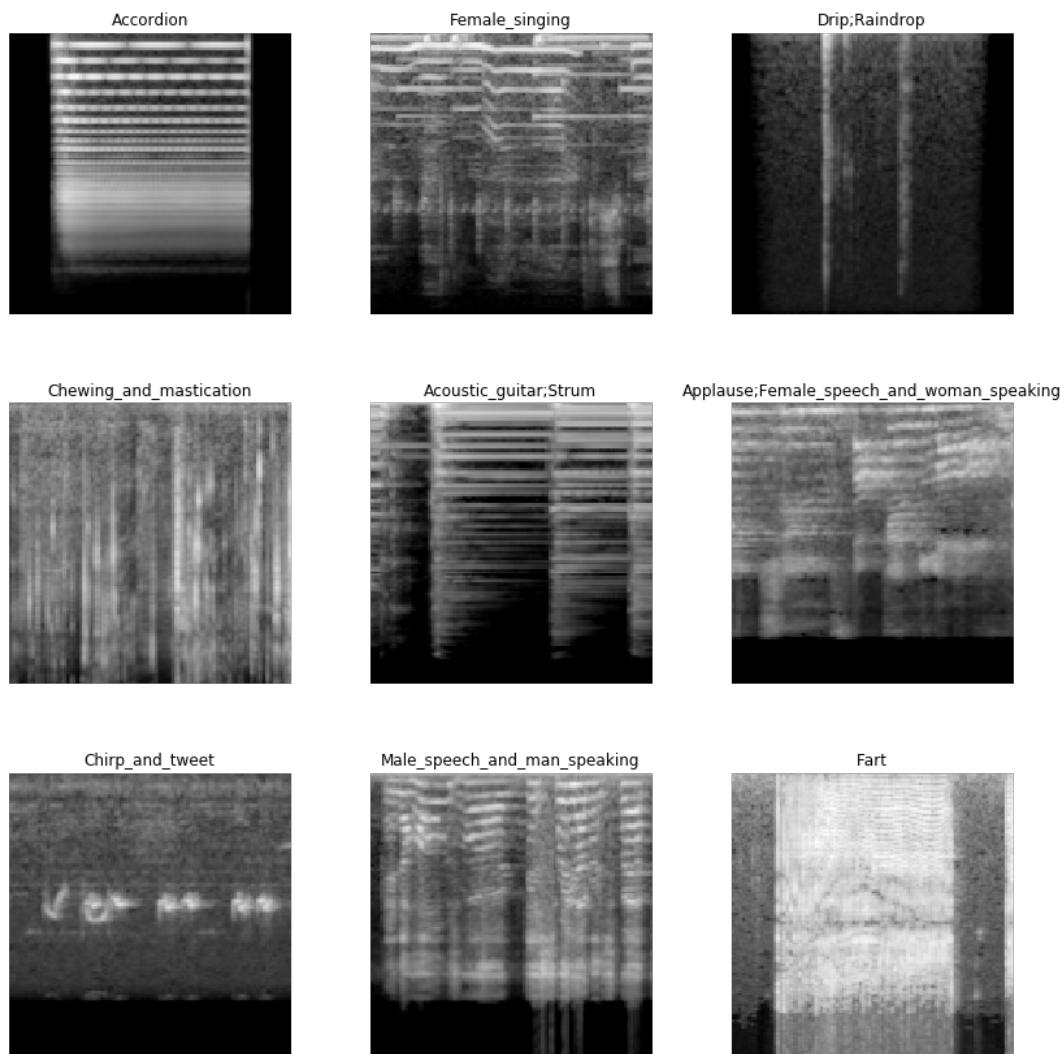


epoch	train_loss	valid_loss	lwlrap	time
0	0.056809	0.025170	0.807519	00:17
1	0.056327	0.024499	0.817057	00:17
2	0.055843	0.024902	0.816686	00:17
3	0.055242	0.024197	0.827524	00:17
4	0.055342	0.024830	0.822752	00:17
5	0.055212	0.024952	0.817645	00:17
6	0.055107	0.025292	0.812336	00:17
7	0.055026	0.025106	0.814816	00:17
8	0.055015	0.025701	0.814898	00:17
9	0.054879	0.025290	0.815145	00:17
10	0.055063	0.025770	0.803378	00:17
11	0.054959	0.025905	0.799450	00:17
12	0.054934	0.025433	0.811939	00:17
13	0.054647	0.026147	0.800416	00:17
14	0.054529	0.025822	0.817137	00:17
15	0.054444	0.026275	0.800868	00:17
16	0.054430	0.026607	0.806144	00:17
17	0.054679	0.026952	0.797005	00:17
18	0.054579	0.026117	0.801299	00:17
19	0.054654	0.027319	0.787045	00:17
20	0.054413	0.026484	0.807626	00:17
21	0.054428	0.026779	0.798557	00:17
22	0.054670	0.027160	0.803713	00:17
23	0.054806	0.027332	0.798136	00:17
24	0.054588	0.027870	0.805102	00:17
25	0.054639	0.027487	0.811315	00:17
26	0.054433	0.027781	0.792565	00:17
27	0.054528	0.027458	0.797414	00:17
28	0.054526	0.027831	0.792149	00:17
29	0.054273	0.026851	0.802051	00:17
30	0.054393	0.026558	0.803709	00:18
31	0.054431	0.028082	0.794148	00:18
32	0.054579	0.027186	0.797958	00:18
33	0.054486	0.027437	0.805832	00:19
34	0.054452	0.028348	0.796113	00:19
35	0.054511	0.029002	0.750153	00:19



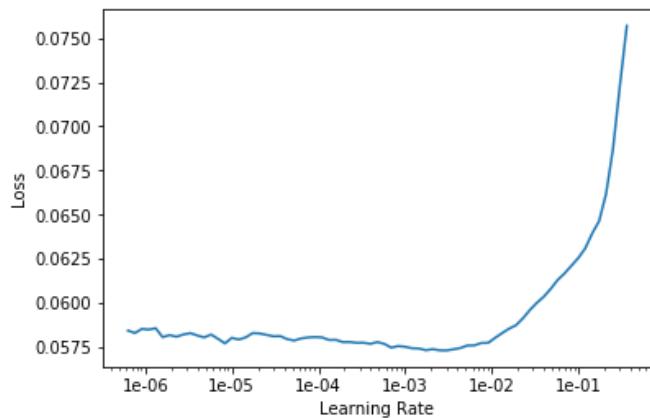
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

```
----- CURATED+NOISY - Fold 9/10  
-----
```

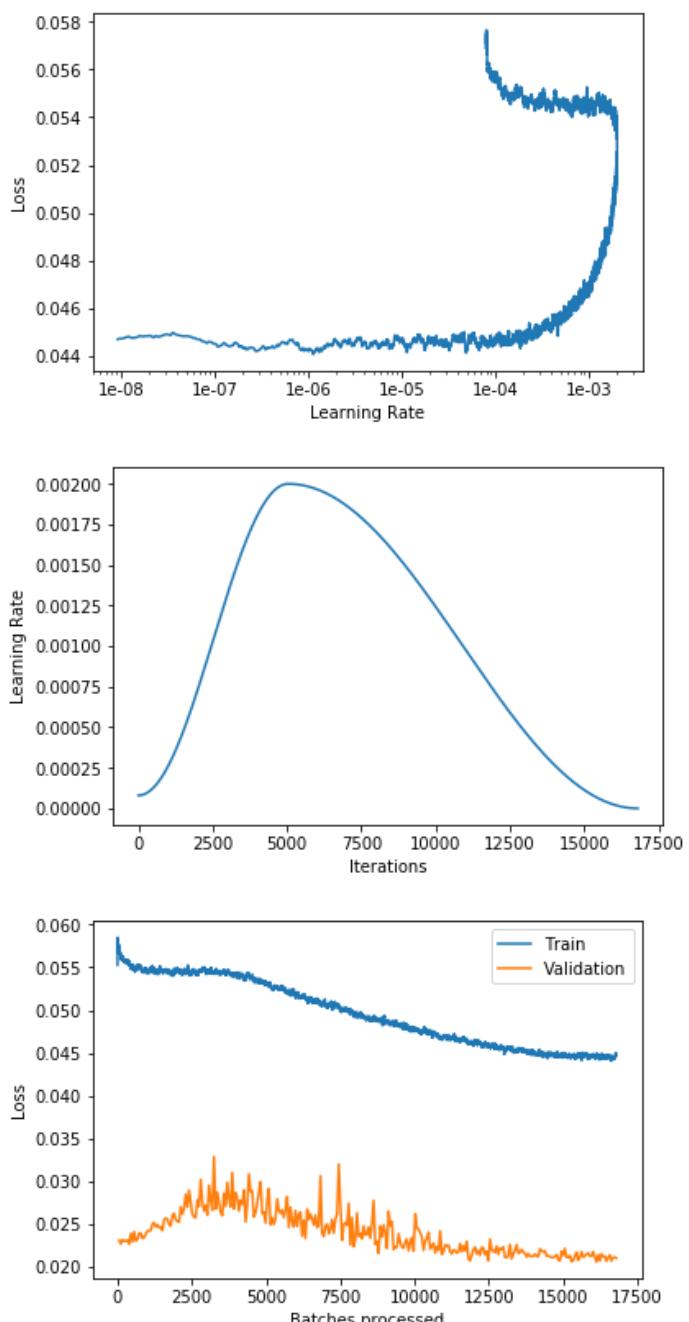


```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

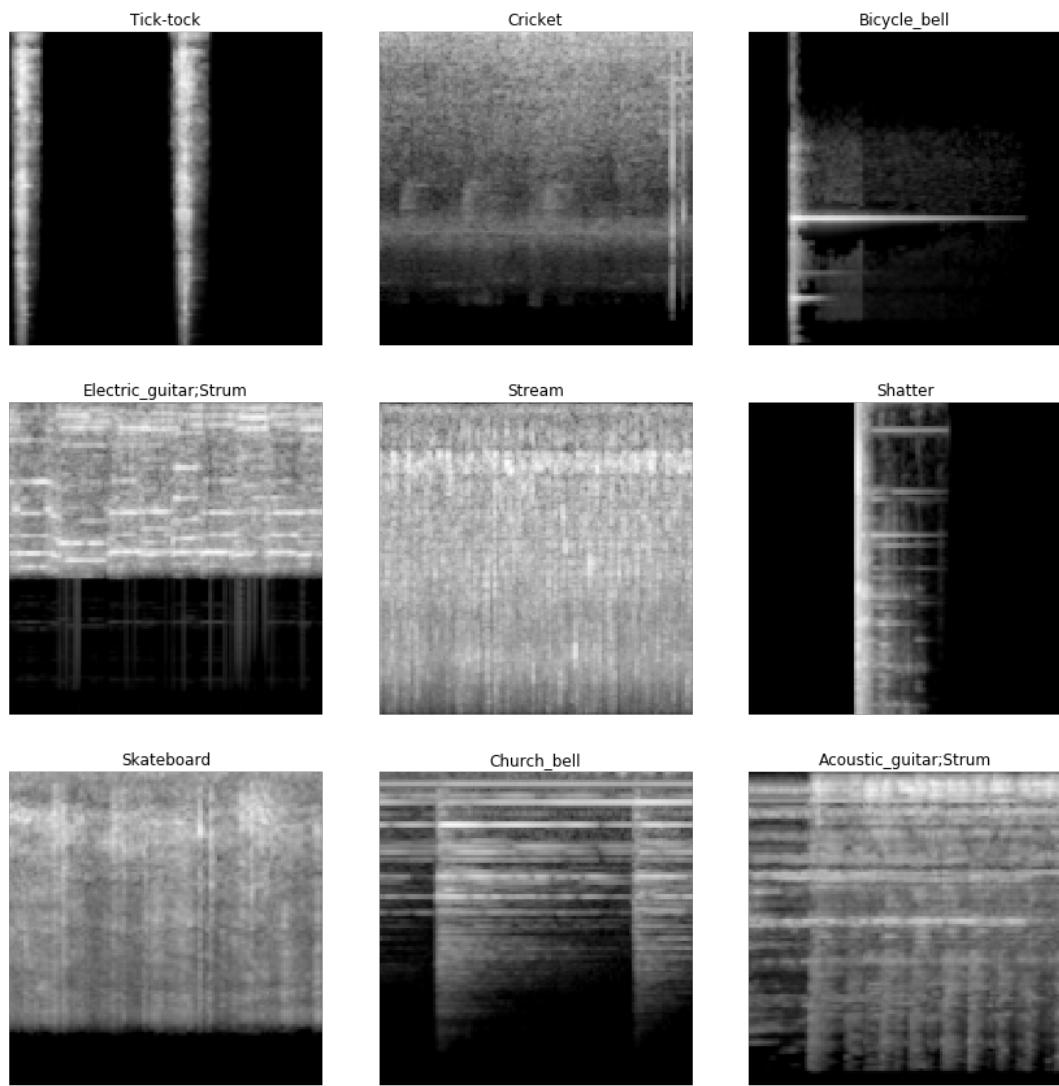


epoch	train_loss	valid_loss	lwlrap	time
0	0.057248	0.023055	0.828958	00:17
1	0.056265	0.022653	0.831030	00:17
2	0.055987	0.023141	0.832521	00:17
3	0.055700	0.022988	0.828120	00:17
4	0.055641	0.022968	0.835475	00:17
5	0.055514	0.023106	0.835262	00:17
6	0.055331	0.022630	0.841995	00:17
7	0.055034	0.023878	0.820424	00:17
8	0.055080	0.022898	0.842762	00:17
9	0.054952	0.024086	0.826223	00:17
10	0.054904	0.023006	0.841145	00:17
11	0.054974	0.023168	0.836914	00:17
12	0.054906	0.023927	0.822487	00:17
13	0.054890	0.023873	0.828567	00:17
14	0.054648	0.023282	0.825262	00:17
15	0.054478	0.023418	0.835527	00:17
16	0.054935	0.023806	0.826907	00:17
17	0.054473	0.024187	0.826526	00:17
18	0.054693	0.024210	0.813561	00:17
19	0.054900	0.024202	0.815314	00:17
20	0.055050	0.025010	0.807600	00:17
21	0.054581	0.025124	0.805762	00:17
22	0.054647	0.024643	0.830598	00:17
23	0.054359	0.024898	0.814449	00:17
24	0.054688	0.024559	0.821255	00:17
25	0.054755	0.024648	0.818967	00:17
26	0.054694	0.024215	0.824243	00:17
27	0.054635	0.025481	0.807619	00:17
28	0.054793	0.025665	0.805245	00:17
29	0.054653	0.025541	0.806076	00:17
30	0.054578	0.024956	0.811235	00:17
31	0.054440	0.024746	0.814830	00:17
32	0.054190	0.025930	0.811075	00:17
33	0.054632	0.026207	0.811898	00:17
34	0.054532	0.025423	0.805663	00:17
35	0.054663	0.025352	0.803573	00:17



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

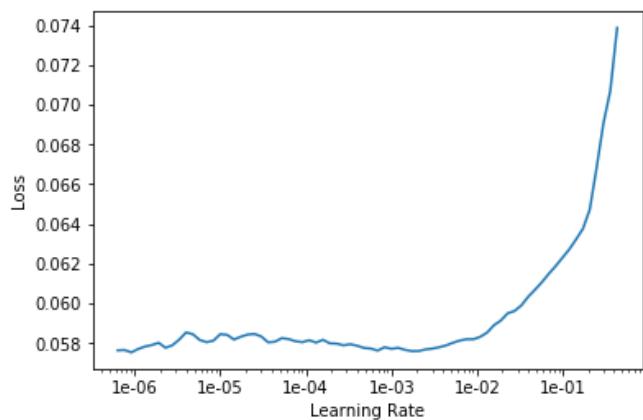
```
----- CURATED+NOISY - Fold 10/10  
-----
```



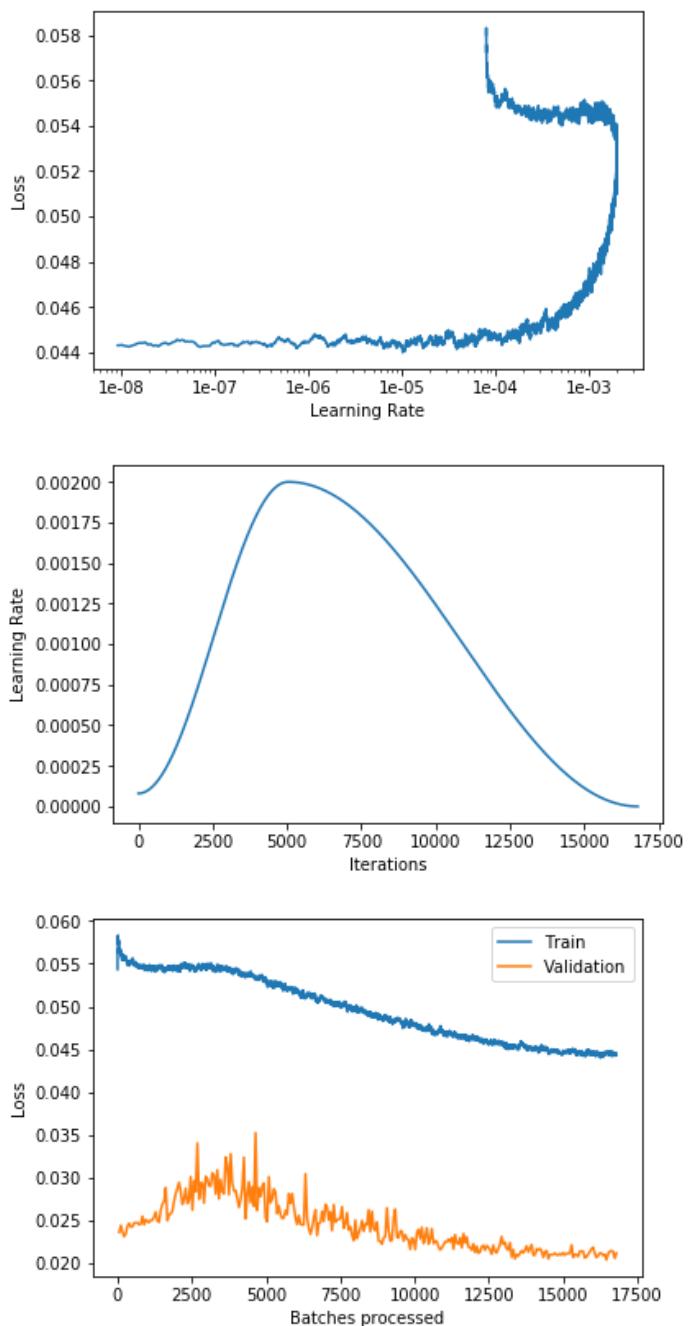
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.056944	0.023557	0.823789	00:17
1	0.056377	0.024361	0.811401	00:17
2	0.055647	0.023597	0.828464	00:17
3	0.055584	0.023055	0.831785	00:17
4	0.055571	0.023412	0.827034	00:17
5	0.055190	0.024311	0.820489	00:17
6	0.055063	0.024603	0.815311	00:17
7	0.055407	0.024187	0.817767	00:17
8	0.055407	0.024219	0.822831	00:17
9	0.055089	0.024656	0.822588	00:17
10	0.054910	0.024653	0.815350	00:17
11	0.054615	0.024687	0.805599	00:17
12	0.054642	0.024450	0.815265	00:17
13	0.054650	0.024519	0.824252	00:17
14	0.054774	0.025427	0.814397	00:17
15	0.054488	0.024454	0.818865	00:17
16	0.054417	0.025036	0.807047	00:17
17	0.054461	0.024804	0.820564	00:17
18	0.054557	0.024714	0.818194	00:17
19	0.054568	0.024997	0.822191	00:17
20	0.054363	0.024970	0.815035	00:17
21	0.054499	0.025380	0.809474	00:17
22	0.054596	0.025919	0.806544	00:17
23	0.054436	0.024866	0.820711	00:18
24	0.054836	0.024747	0.821278	00:17
25	0.054509	0.026154	0.808455	00:17
26	0.054397	0.026859	0.788753	00:17
27	0.054274	0.027144	0.797872	00:17
28	0.054483	0.028781	0.768652	00:17
29	0.054586	0.024930	0.811367	00:17
30	0.054476	0.025622	0.807390	00:17
31	0.054351	0.025716	0.803862	00:17
32	0.054481	0.026053	0.797376	00:17
33	0.054462	0.027181	0.796453	00:17
34	0.054428	0.028095	0.773691	00:17
35	0.054603	0.028659	0.764602	00:17



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
  "type " + obj.__name__ + ". It won't be checked "
```

```
In [71]: kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-10_fold-{fold}')
```

Overall lwlrmp on CURATED dataset: 0.8792998601729842

	lwlrap	weight
Fill_(with_liquid)	0.612078	0.008693
Squeak	0.648857	0.013039
Walk_and_footsteps	0.676538	0.013039
Mechanical_fan	0.718587	0.008519
Tap	0.730889	0.013039
Chink_and_clink	0.743390	0.013039
Hiss	0.747537	0.013039
Cutlery_and_silverware	0.778966	0.013039
Frying_(food)	0.782194	0.010953
Buzz	0.792114	0.009736
Slam	0.793000	0.013039
Traffic_noise_and_roadway_noise	0.799773	0.013039
Yell	0.802118	0.013039
Clapping	0.806247	0.013039
Male_speech_and_manSpeaking	0.811864	0.013039
Water_tap_and_faucet	0.813878	0.013039
Sink_(filling_or_washing)	0.815445	0.013039
Motorcycle	0.817905	0.013039
Microwave_oven	0.820599	0.013039
Bus	0.827321	0.013039
Dishes_and_pots_and_pans	0.828604	0.013039
Run	0.835460	0.013039
Accelerating_and_revving_and_vroom	0.842000	0.013039
Trickle_and_dribble	0.843449	0.009214
Crowd	0.848535	0.013039
Stream	0.848852	0.013039
Bathtub_(filling_or_washing)	0.856889	0.013039
Drip	0.858263	0.013039
Car_passing_by	0.858349	0.013039
Chirp_and_tweet	0.865052	0.013039
...	...	...
Raindrop	0.919841	0.013039
Keys_jangling	0.923063	0.013039
Race_car_and_auto_racing	0.925978	0.009736
Sigh	0.929825	0.009910

```
In [72]: kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-10_fold-{fold}')
```

Overall lwlrap on NOISY dataset: 0.5671609663778308

	<b>lwlrap</b>	<b>weight</b>
Burping_and_eructation	0.144436	0.0125
Cupboard_open_or_close	0.172749	0.0125
Finger_snapping	0.191994	0.0125
Raindrop	0.247609	0.0125
Keys_jangling	0.247799	0.0125
Sigh	0.264568	0.0125
Shatter	0.264755	0.0125
Fart	0.273453	0.0125
Zipper_(clothing)	0.276470	0.0125
Tap	0.308443	0.0125
Gasp	0.338424	0.0125
Buzz	0.352866	0.0125
Writing	0.353835	0.0125
Computer_keyboard	0.373626	0.0125
Tick-tock	0.394039	0.0125
Slam	0.394995	0.0125
Scissors	0.405324	0.0125
Knock	0.420263	0.0125
Sneeze	0.428044	0.0125
Purr	0.449069	0.0125
Bicycle_bell	0.459327	0.0125
Chink_and_clink	0.463298	0.0125
Bass_guitar	0.494233	0.0125
Chewing_and_mastication	0.504562	0.0125
Drip	0.505812	0.0125
Crackle	0.513236	0.0125
Whispering	0.521537	0.0125
Squeak	0.530789	0.0125
Fill_(with_liquid)	0.532975	0.0125
Skateboard	0.534338	0.0125
...	...	...
Electric_guitar	0.658164	0.0125
Clapping	0.659538	0.0125
Sink_(filling_or_washing)	0.661888	0.0125
Female_singing	0.663096	0.0125

```
In [108]: def lwlrap_per_sample(fname, kf, df):
    overall_preds, overall_thruth, overall_index = _kfold_prediction(kf, df, fname)

    m = pd.DataFrame(
        [_one_sample_positive_class_precisions(p, t)[1].mean() for i, (p, t) in
         enumerate(zip(overall_preds, overall_thruth))],
        columns=['lwlrap'],
        index=overall_index
    )

    m[['fname', 'labels']] = df[['fname', 'labels']]

    return m.sort_values('lwlrap', ascending=False)
```

```
In [109]: m4 = lwlrap_per_sample('stage-10_fold-{fold}', kf_noisy, trn_noisy_df)
```

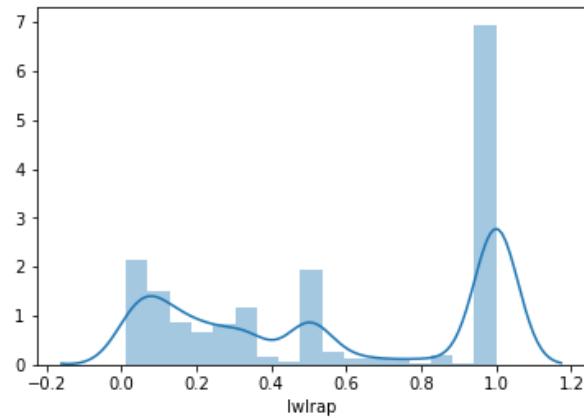
In [110]: m4

Out[110]:

	lwlrap	fname	labels
19798	1.000000	ffd7d1fe.wav	Applause
19561	1.000000	fc631546.wav	Male_speech_and_manSpeaking
3951	1.000000	327f4ab3.wav	Female_speech_and_womanSpeaking,Applause
3976	1.000000	32cfcd1a.wav	Accordion
19609	1.000000	fd0411df.wav	Tap
3992	1.000000	32fa6d6a.wav	Applause
4028	1.000000	336df3ed.wav	Motorcycle,Traffic_noise_and_roadway_noise
4034	1.000000	338f1a0b.wav	Trickle_and_dribble
19558	1.000000	fc455cfe.wav	Acoustic_guitar
4110	1.000000	34b97903.wav	Male_speech_and_manSpeaking
4064	1.000000	340bdc49.wav	Clapping
19532	1.000000	fbf15c98.wav	Hiss
4080	1.000000	34477c68.wav	Waves_and_surf
4091	1.000000	347f5f35.wav	Yell,Hi-hat
19526	1.000000	fb8dc33.wav	Bus
4096	1.000000	348d0c89.wav	Chirp_and_tweet
3934	1.000000	323837b5.wav	Stream
3932	1.000000	322eaa58.wav	Walk_and_footsteps
19632	1.000000	fd5ad3d0.wav	Male_speech_and_manSpeaking
3922	1.000000	321494a4.wav	Male_singing
3914	1.000000	31fecbba.wav	Motorcycle
19693	1.000000	fe3afea8.wav	Male_speech_and_manSpeaking
3900	1.000000	31d1c9c4.wav	Bathtub_(filling_or_washing)
19708	1.000000	fe7a4bf1.wav	Female_singing
3870	1.000000	3164260e.wav	Bark
3866	1.000000	315a9966.wav	Dishes_and_pots_and_pans,Cutlery_and_silverware
19723	1.000000	fea68c59.wav	Electric_guitar
3837	1.000000	31152ca1.wav	Mechanical_fan
19741	1.000000	fef9a3ee.wav	Bark
19784	1.000000	ff960ab8.wav	Applause
...	...	...	...
17182	0.013514	de2e9e0b.wav	Finger_snapping
15956	0.013514	ce3c8fe7.wav	Cupboard_open_or_close
14793	0.013333	be273a3c.wav	Race_car_and_auto_racing
10556	0.013333	8837134f.wav	Fart

```
In [116]: (m4.lwlrp < .5).sum(), (m4.lwlrp >= .5).sum(), (m4.lwlrp == 1).sum()  
Out[116]: (8581, 11234, 7981)
```

```
In [123]: sns.distplot(m4.lwlrp);
```



```
In [124]: ok_noisy_items2 = m4[m4.lwlrp == 1].index  
len(ok_noisy_items2)  
Out[124]: 7981
```

```
In [125]: for fold, ((train_index1, valid_index1), (train_index2, valid_index2)) in enumerate(zip(kf_curated.split(trn_curated_df), kf_noisy.split(trn_noisy_df))):
    print('-' * 40, f'CURATED+NOISY+2 - Fold {fold+1}/{n_splits}', '-' * 40)

    train_index2 = list(set(train_index2).intersection(set(ok_noisy_items2)))

    mix_df = pd.concat([
        trn_curated_df.iloc[train_index1],
        trn_noisy_df.iloc[train_index2],
        trn_curated_df.iloc[valid_index1],
#        trn_noisy_df.iloc[valid_index2], # compute lwlrap only on curated
    ], ignore_index=True)
    train_index = mix_df[:len(train_index1)+len(train_index2)].index
    valid_index = mix_df[len(train_index1)+len(train_index2):].index

    src = (ImageList.from_df(mix_df, WORK)
           .split_by_idxs(train_index, valid_index)
           .label_from_df(label_delim=','))
    data = (src.transform(tfms, size=128)
            .databunch(bs=bs))
    data.show_batch(3)
    plt.show()

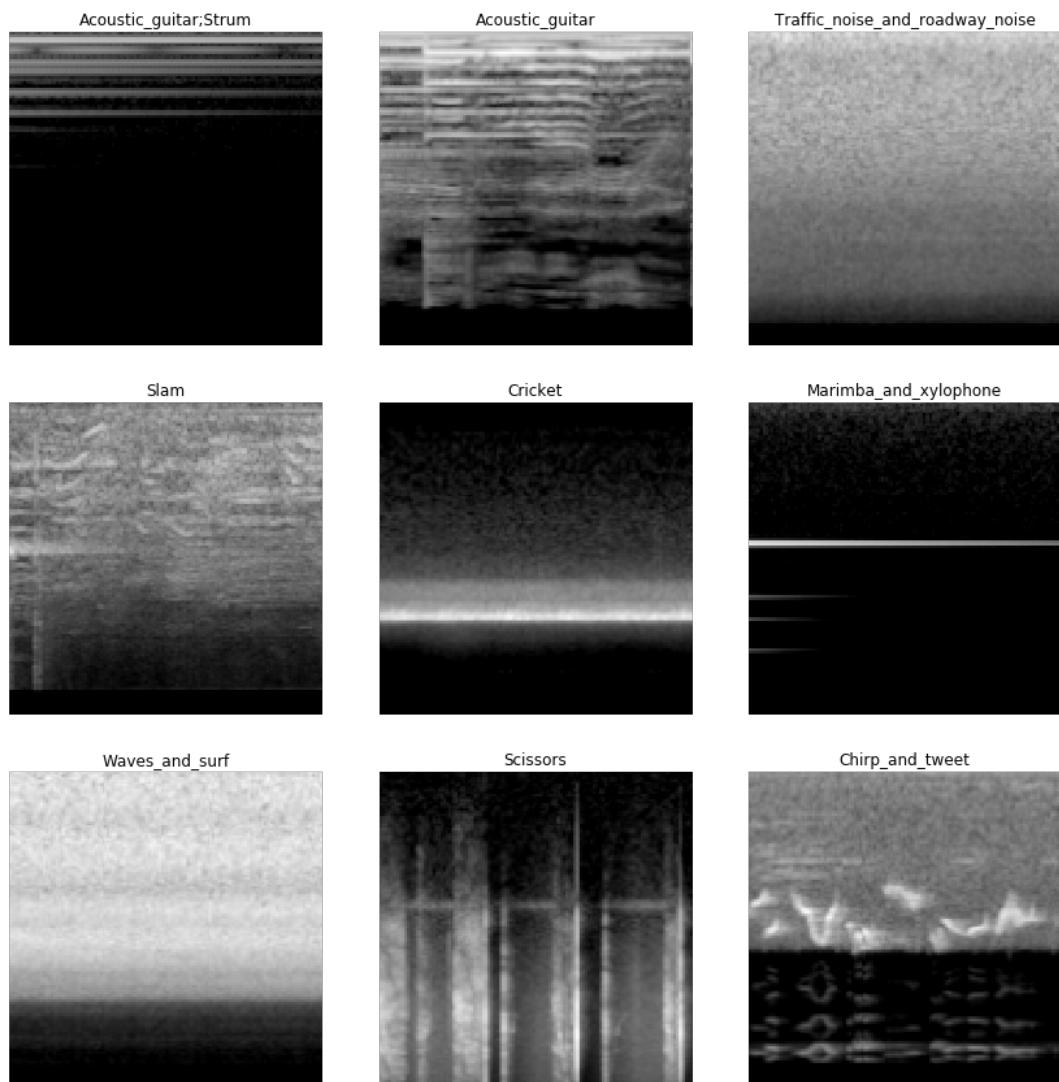
    learn.load(f'stage-10_fold-{fold}')
    learn.data = data

    learning(learn, 100, 1e-2)

    learn.save(f'stage-11_fold-{fold}')
    learn.export(f'stage-11_fold-{fold}.pkl')

    if TOY_MODE:
        break
```

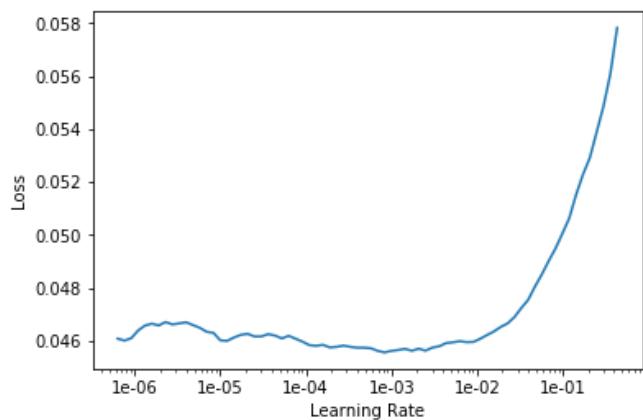
----- CURATED+NOISY+2 - Fold 1/10 -----



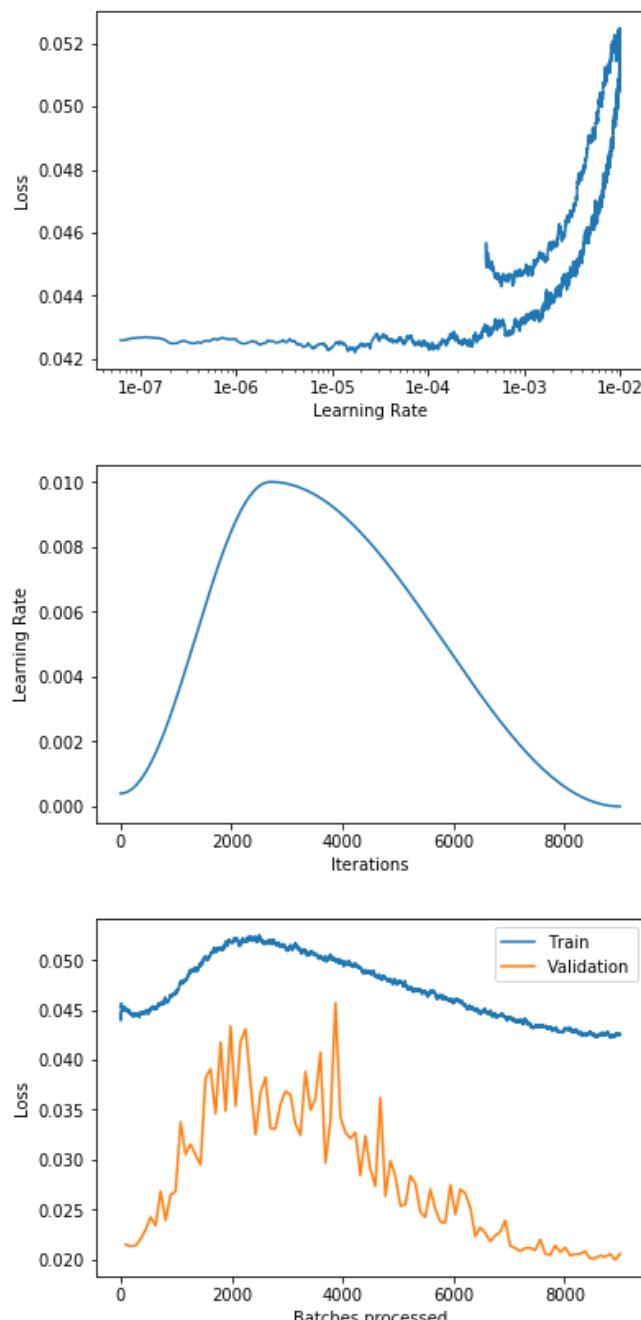
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

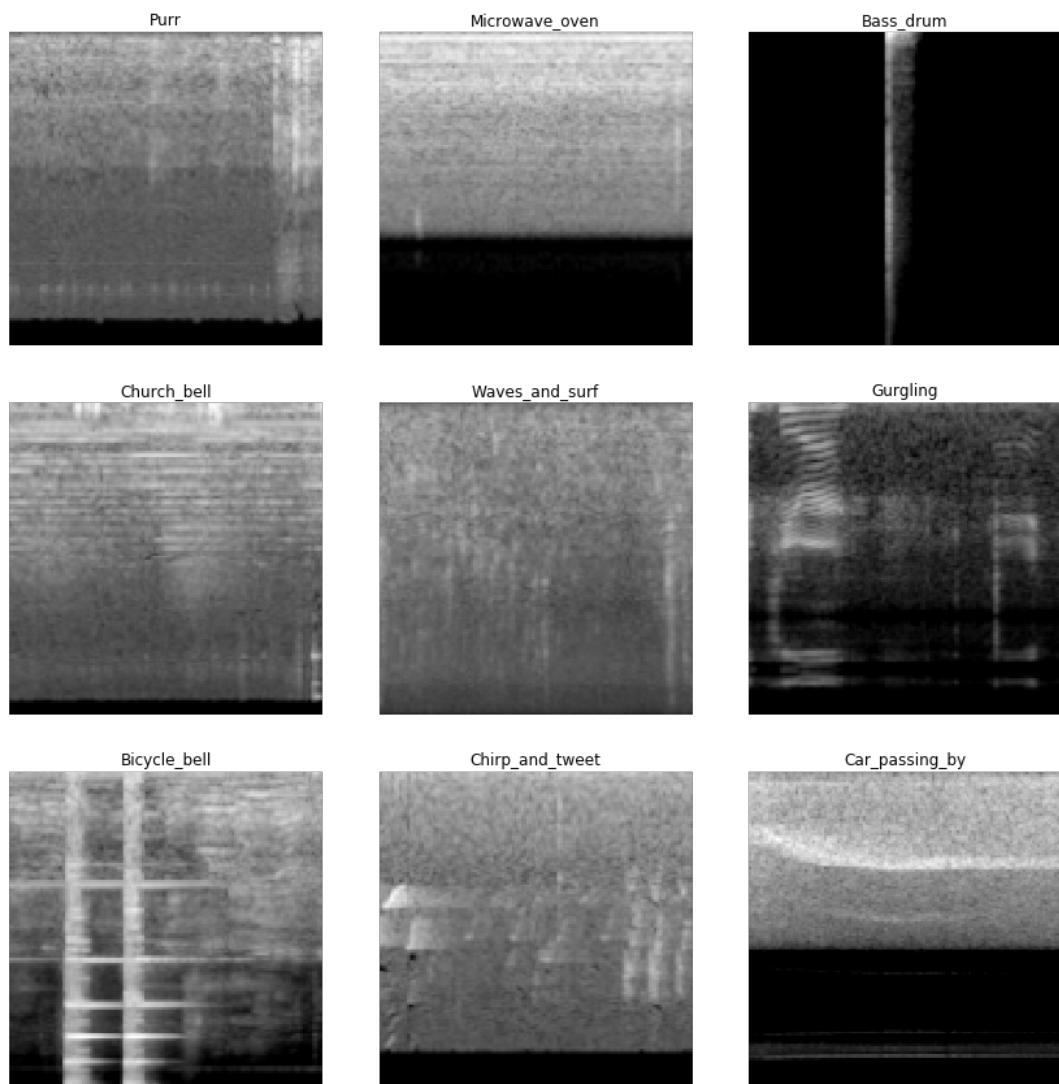


epoch	train_loss	valid_loss	lwlrap	time
0	0.045100	0.021470	0.838579	00:27
1	0.044601	0.021310	0.829119	00:26
2	0.044545	0.021357	0.840019	00:26
3	0.044782	0.022040	0.834100	00:26
4	0.044891	0.022975	0.823205	00:27
5	0.045007	0.024228	0.816506	00:27
6	0.045211	0.023376	0.818554	00:27
7	0.045713	0.026815	0.775371	00:25
8	0.046147	0.023903	0.813561	00:25
9	0.046387	0.026452	0.789482	00:25
10	0.046769	0.026806	0.774576	00:25
11	0.047871	0.033742	0.688085	00:25
12	0.048118	0.030523	0.739423	00:25
13	0.048927	0.031536	0.727823	00:26
14	0.049331	0.030449	0.750361	00:26
15	0.049573	0.029467	0.766518	00:26
16	0.050419	0.038229	0.656096	00:25
17	0.050668	0.039095	0.633960	00:25
18	0.051056	0.034598	0.705267	00:25
19	0.051257	0.041764	0.612356	00:25
20	0.051728	0.034870	0.698935	00:25
21	0.051893	0.043390	0.596447	00:25
22	0.052174	0.035359	0.689726	00:25
23	0.051927	0.041796	0.610383	00:25
24	0.052075	0.043105	0.580537	00:25
25	0.052281	0.037687	0.656785	00:25
26	0.052243	0.032515	0.725419	00:25
27	0.052179	0.036708	0.671157	00:25
28	0.051892	0.038258	0.646912	00:25
29	0.051946	0.033105	0.709620	00:25
30	0.051745	0.033110	0.715120	00:25
31	0.051482	0.035552	0.679425	00:25
32	0.051281	0.036850	0.667832	00:25
33	0.051466	0.036506	0.668783	00:25
34	0.051669	0.033576	0.707144	00:25
35	0.051080	0.032450	0.708011	00:25



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

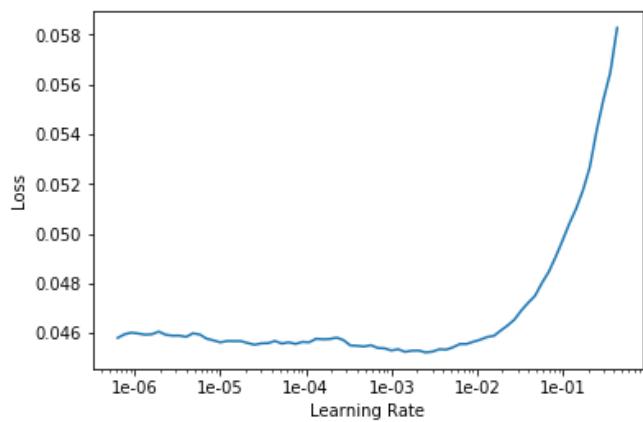
```
----- CURATED+NOISY+2 - Fold 2/10  
-----
```



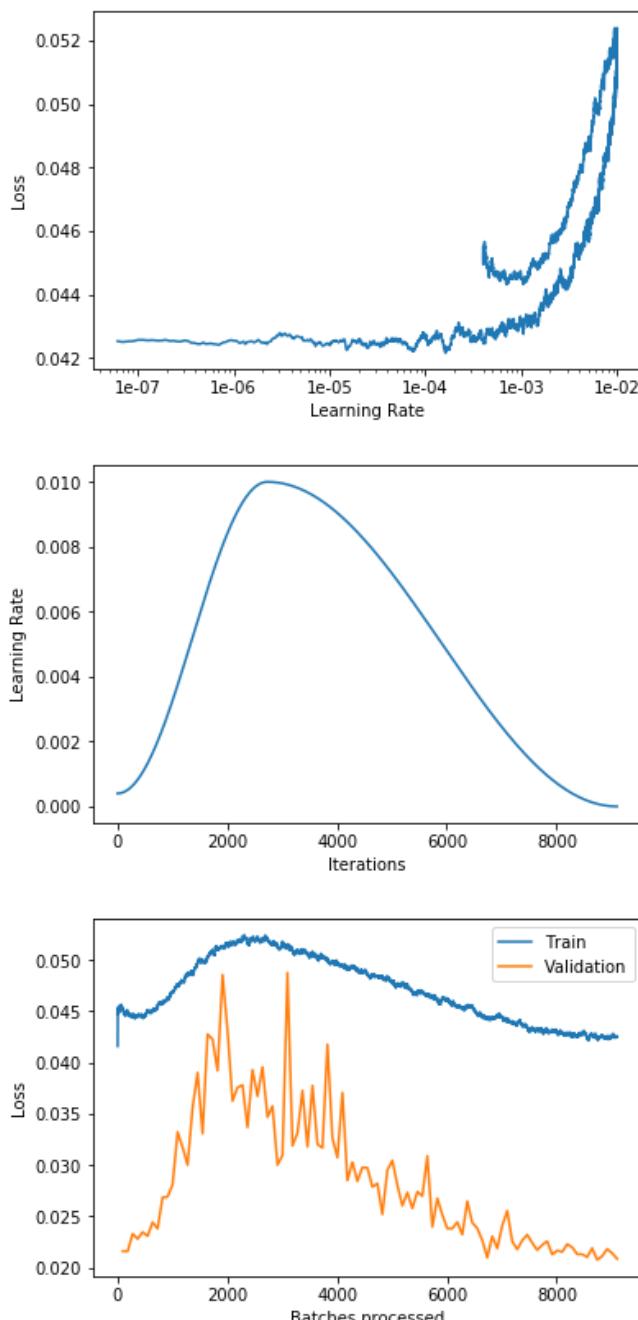
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

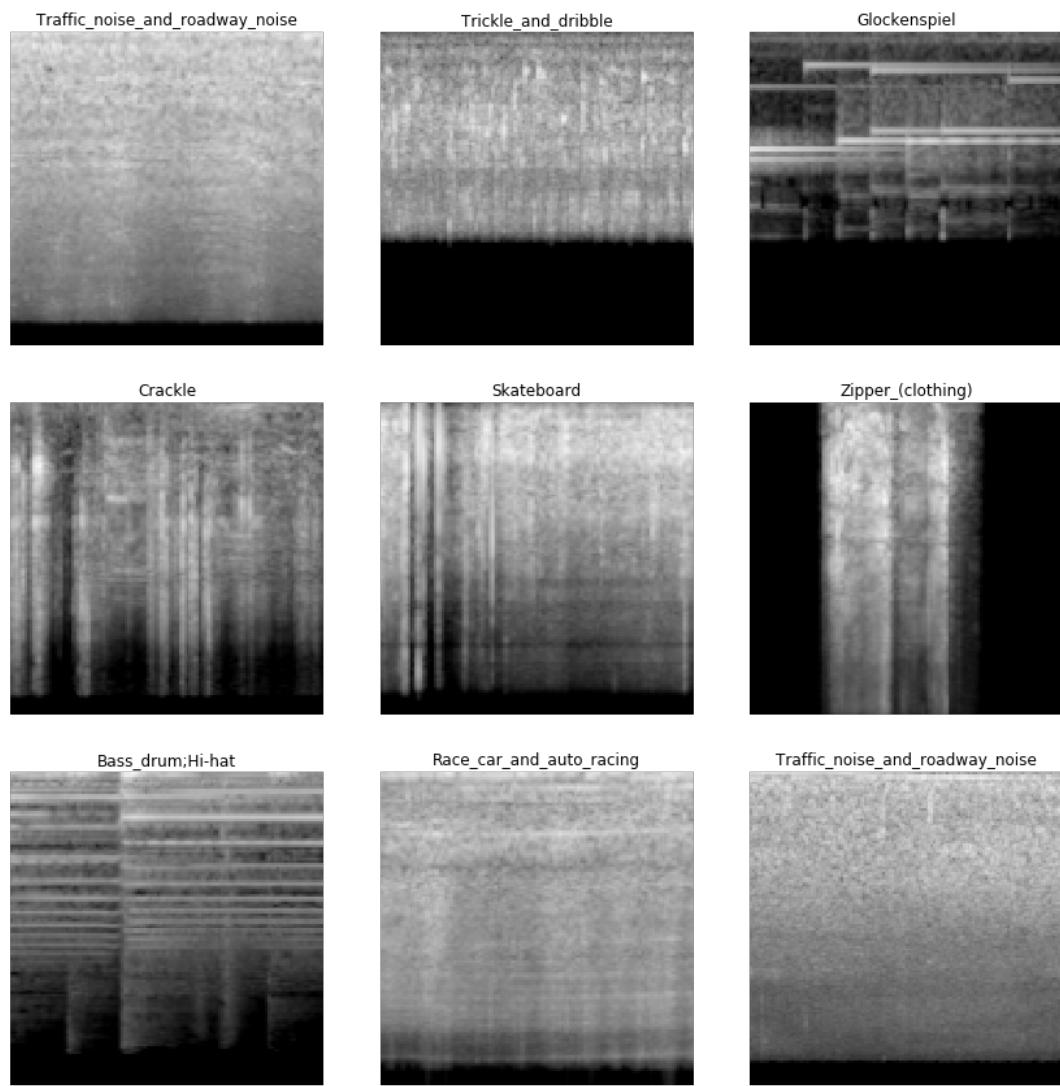


epoch	train_loss	valid_loss	lwlrap	time
0	0.045213	0.021607	0.830264	00:25
1	0.044984	0.021602	0.838019	00:26
2	0.044675	0.023301	0.815386	00:25
3	0.044534	0.022808	0.825725	00:25
4	0.044433	0.023463	0.816815	00:25
5	0.045037	0.023090	0.817090	00:25
6	0.045081	0.024431	0.807810	00:25
7	0.045585	0.023809	0.816962	00:25
8	0.045829	0.026857	0.781265	00:25
9	0.046290	0.026912	0.794355	00:25
10	0.046892	0.028103	0.766619	00:25
11	0.047487	0.033233	0.718541	00:25
12	0.047935	0.031729	0.733148	00:25
13	0.048734	0.030014	0.742035	00:25
14	0.048848	0.035629	0.688885	00:25
15	0.049707	0.039016	0.639436	00:25
16	0.050013	0.033083	0.709333	00:25
17	0.050385	0.042763	0.608866	00:25
18	0.050874	0.042274	0.603696	00:25
19	0.051096	0.039215	0.659876	00:25
20	0.051294	0.048574	0.553624	00:25
21	0.051591	0.043131	0.591334	00:25
22	0.051648	0.036259	0.661901	00:25
23	0.051914	0.037574	0.647684	00:25
24	0.051895	0.037808	0.638827	00:25
25	0.051654	0.033690	0.698149	00:25
26	0.052135	0.039299	0.646311	00:25
27	0.052142	0.036725	0.670991	00:25
28	0.052108	0.039573	0.652428	00:25
29	0.051892	0.034693	0.680182	00:25
30	0.051525	0.035757	0.687664	00:25
31	0.051347	0.030014	0.747893	00:25
32	0.051729	0.030954	0.752910	00:25
33	0.051468	0.048754	0.520705	00:25
34	0.050982	0.031895	0.731571	00:25
35	0.051000	0.033133	0.707510	00:25



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

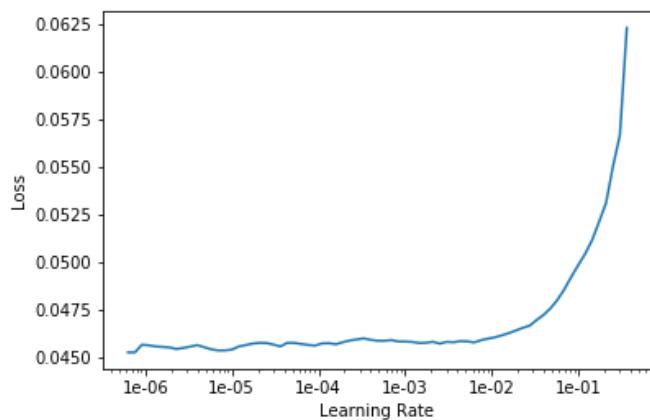
```
----- CURATED+NOISY+2 - Fold 3/10  
-----
```



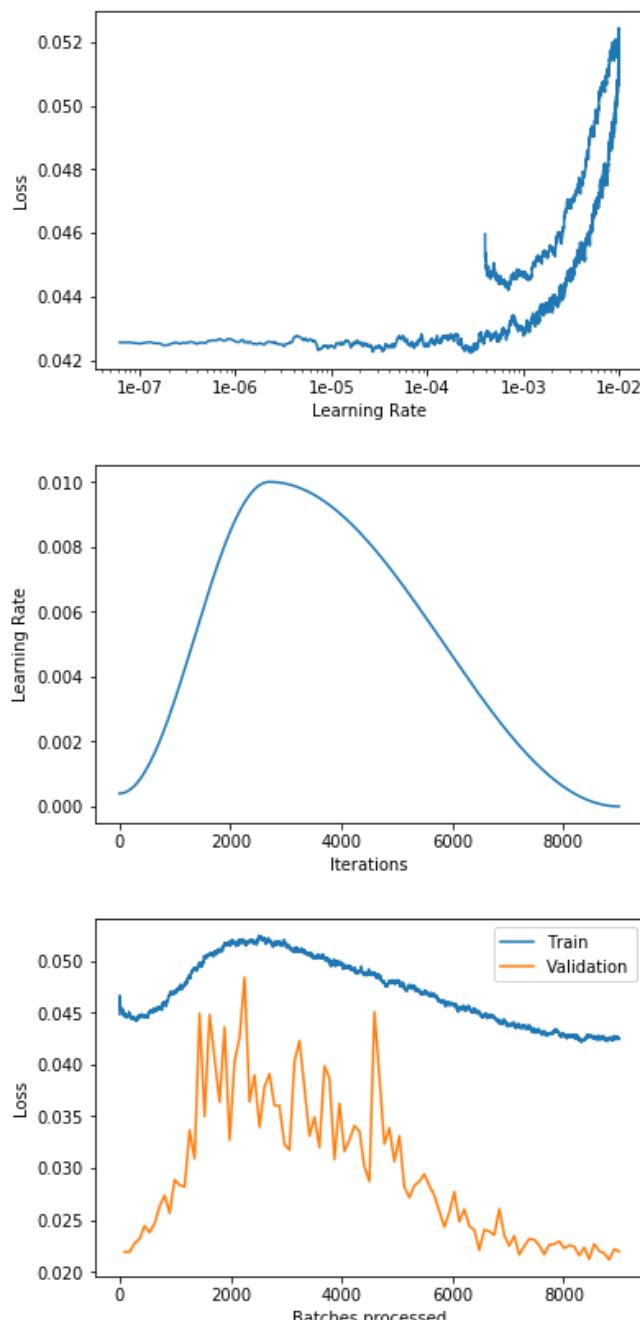
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

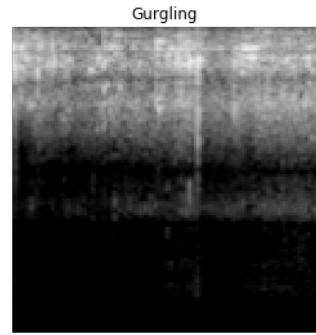
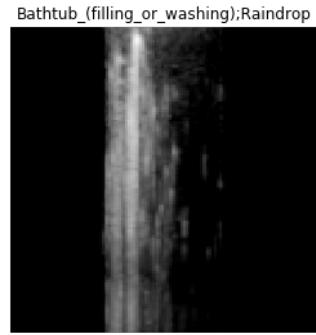
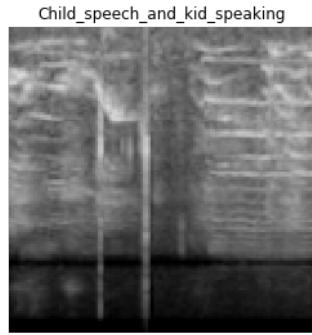
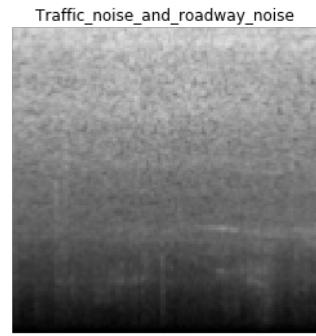
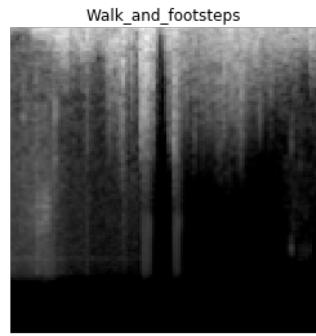
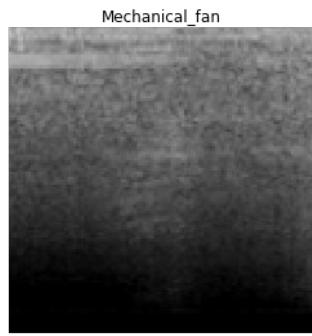
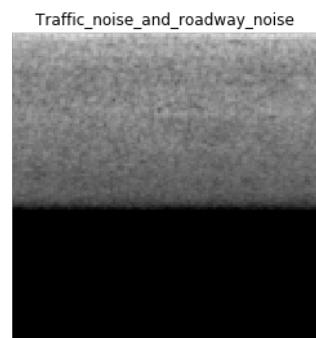
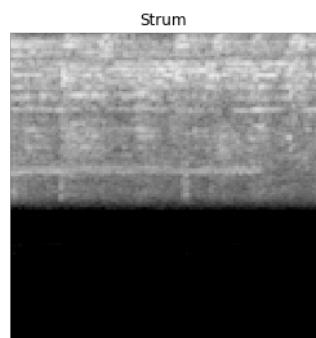
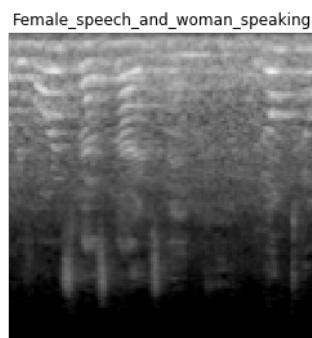


epoch	train_loss	valid_loss	lwlrap	time
0	0.044810	0.021927	0.832009	00:25
1	0.044830	0.021930	0.828242	00:25
2	0.044544	0.022750	0.813221	00:25
3	0.044608	0.023198	0.825139	00:25
4	0.044770	0.024439	0.801027	00:25
5	0.045114	0.023799	0.815405	00:25
6	0.045373	0.024665	0.806384	00:25
7	0.045485	0.026328	0.792087	00:25
8	0.045982	0.027379	0.770572	00:25
9	0.046758	0.025664	0.793724	00:25
10	0.047087	0.028880	0.766610	00:25
11	0.047336	0.028381	0.762528	00:25
12	0.047875	0.028210	0.773939	00:25
13	0.048388	0.033666	0.699171	00:25
14	0.049351	0.030934	0.732604	00:25
15	0.049844	0.044929	0.571007	00:25
16	0.050447	0.035003	0.680238	00:25
17	0.050642	0.044810	0.566439	00:25
18	0.050819	0.040639	0.620930	00:25
19	0.050990	0.036378	0.668018	00:25
20	0.051671	0.043612	0.593512	00:25
21	0.051718	0.032714	0.719869	00:25
22	0.051773	0.040286	0.611162	00:25
23	0.052024	0.042576	0.606073	00:25
24	0.051880	0.048408	0.527379	00:25
25	0.051984	0.036419	0.664247	00:25
26	0.051892	0.038982	0.643048	00:25
27	0.052291	0.033959	0.702004	00:25
28	0.052121	0.037860	0.651004	00:25
29	0.052011	0.039099	0.638051	00:25
30	0.051775	0.036053	0.679553	00:25
31	0.051814	0.036036	0.677994	00:25
32	0.051851	0.032292	0.724593	00:25
33	0.051460	0.031767	0.721287	00:25
34	0.051159	0.040309	0.619343	00:25
35	0.050958	0.042292	0.602991	00:25



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
"type " + obj.__name__ + ". It won't be checked "
```

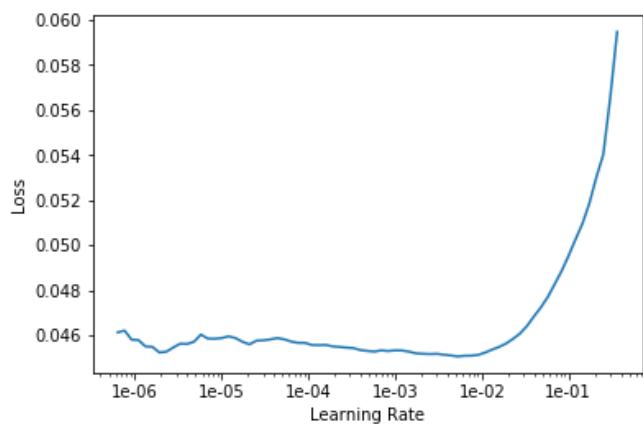
```
----- CURATED+NOISY+2 - Fold 4/10
-----
```



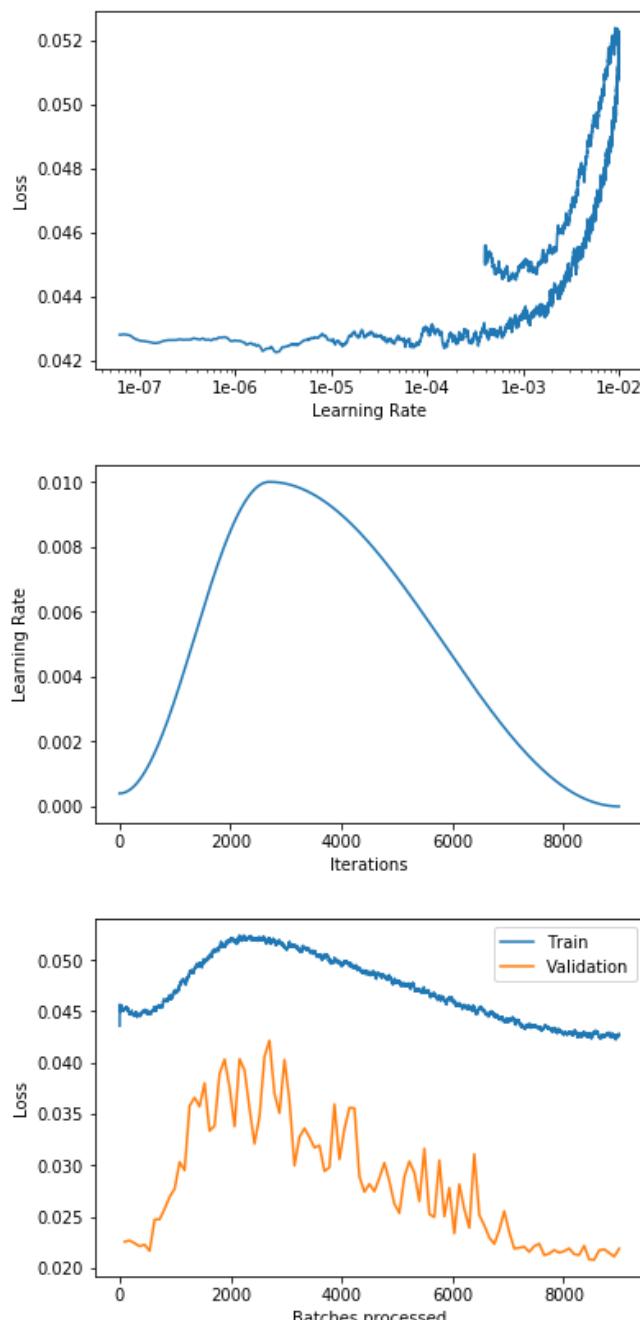
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.
```

```
"type " + obj.__name__ + ". It won't be checked "
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

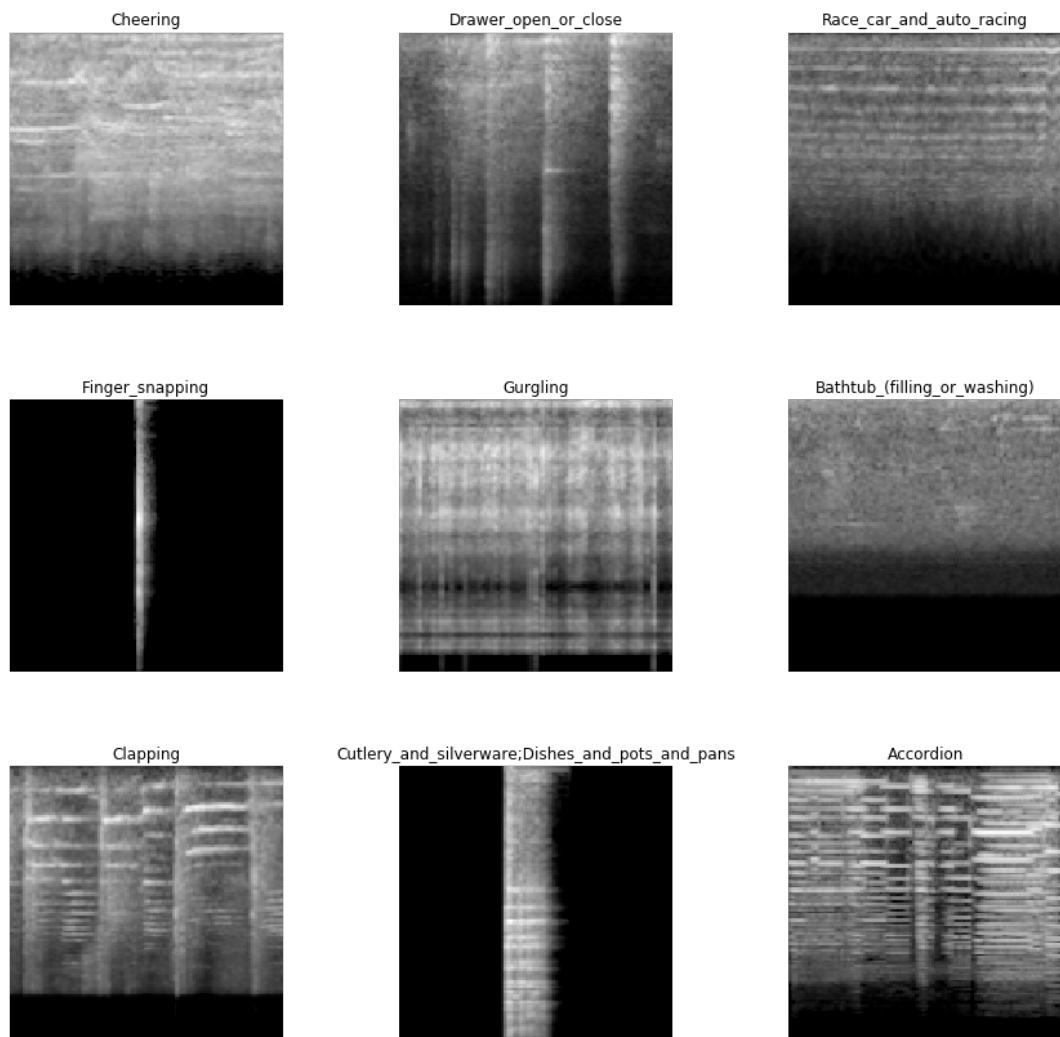


epoch	train_loss	valid_loss	lwlrap	time
0	0.045247	0.022581	0.820164	00:25
1	0.044779	0.022697	0.828605	00:25
2	0.044696	0.022430	0.825033	00:25
3	0.044629	0.022135	0.823214	00:25
4	0.045060	0.022292	0.835347	00:25
5	0.044883	0.021687	0.829611	00:25
6	0.045365	0.024743	0.801705	00:25
7	0.045463	0.024758	0.808124	00:25
8	0.046008	0.025780	0.798363	00:25
9	0.046553	0.026947	0.776240	00:25
10	0.047005	0.027721	0.765959	00:25
11	0.047495	0.030329	0.737568	00:25
12	0.047876	0.029534	0.751820	00:25
13	0.048701	0.035793	0.693793	00:25
14	0.049153	0.036610	0.662627	00:25
15	0.049554	0.035744	0.674412	00:25
16	0.049969	0.038013	0.641812	00:25
17	0.050425	0.033375	0.699730	00:25
18	0.050974	0.033858	0.695717	00:25
19	0.051299	0.038918	0.638510	00:25
20	0.051425	0.040336	0.610238	00:25
21	0.051850	0.037617	0.653423	00:25
22	0.051925	0.033792	0.705563	00:25
23	0.052329	0.040359	0.616626	00:25
24	0.052182	0.039279	0.641247	00:25
25	0.052272	0.035521	0.688328	00:25
26	0.052070	0.032118	0.709550	00:25
27	0.052190	0.034778	0.689666	00:25
28	0.052057	0.040589	0.609517	00:25
29	0.052026	0.042168	0.614866	00:25
30	0.051859	0.036996	0.656229	00:25
31	0.051864	0.035090	0.672209	00:25
32	0.051391	0.040286	0.623947	00:25
33	0.051708	0.036299	0.664889	00:25
34	0.051491	0.030002	0.743917	00:25
35	0.051076	0.032771	0.702108	00:25



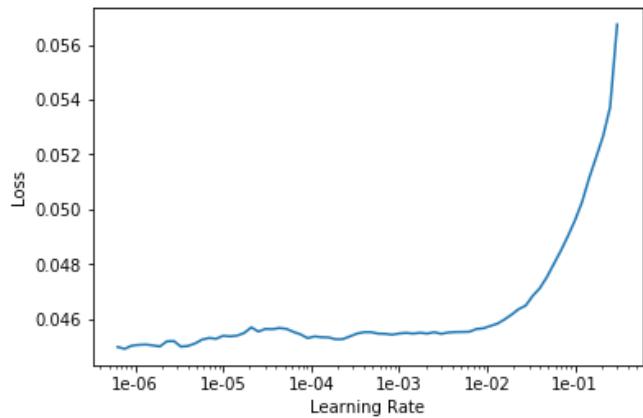
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

```
----- CURATED+NOISY+2 - Fold 5/10  
-----
```

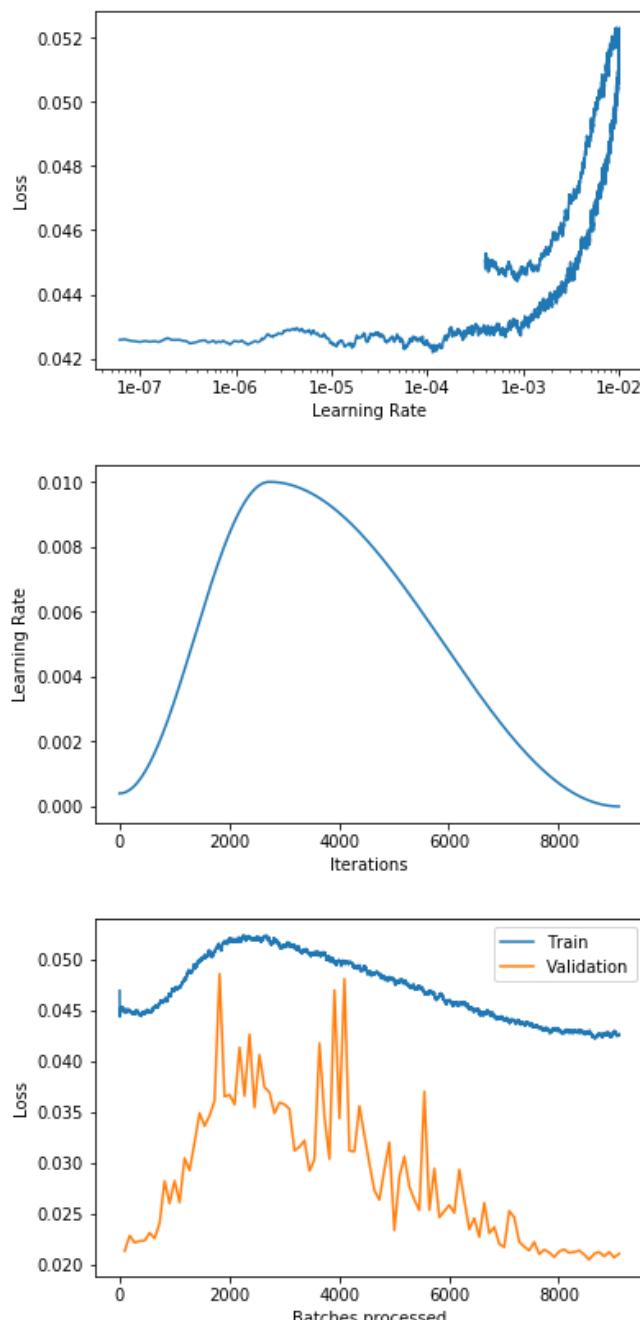


```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

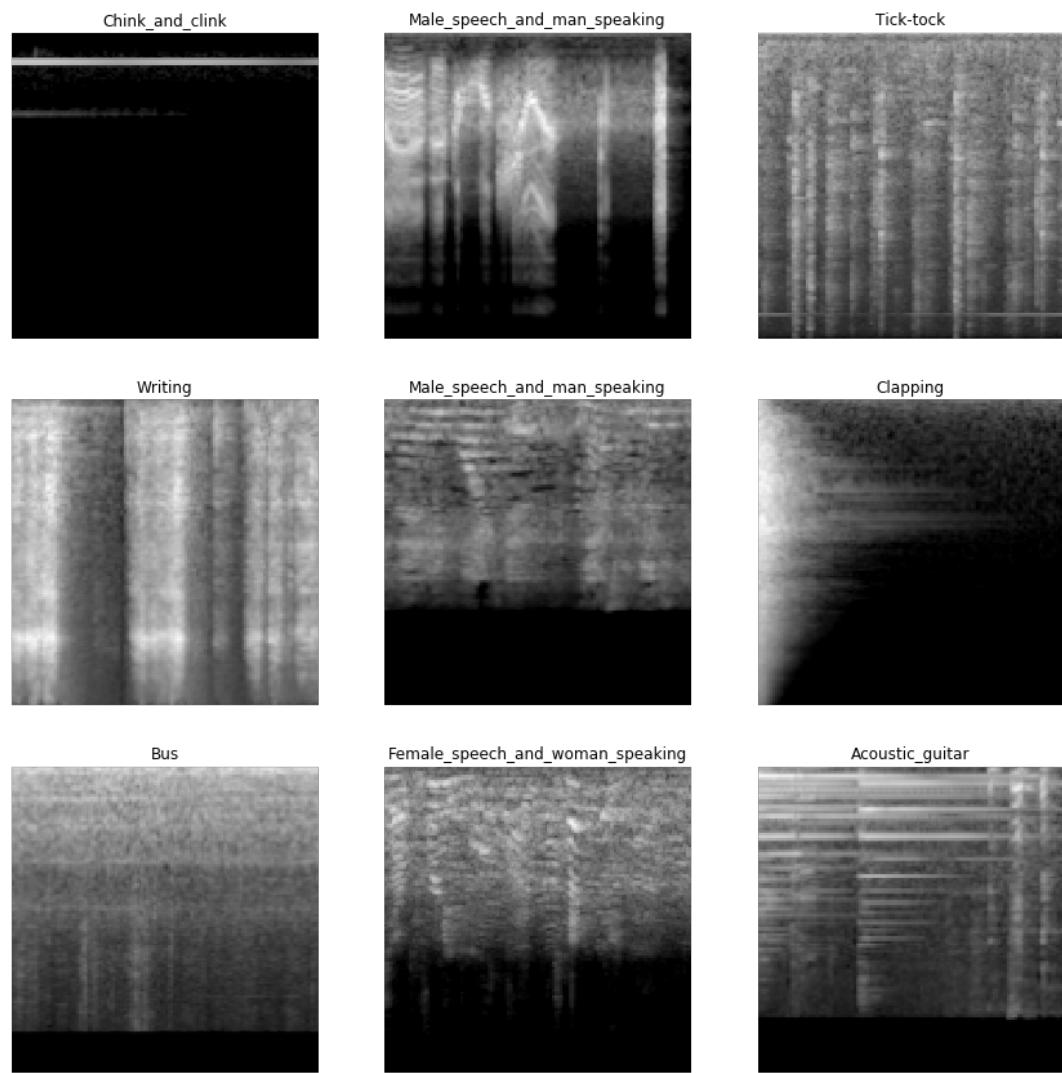


epoch	train_loss	valid_loss	lwlrap	time
0	0.044952	0.021348	0.860392	00:25
1	0.044910	0.022858	0.844565	00:25
2	0.044635	0.022177	0.844344	00:25
3	0.044656	0.022334	0.834913	00:25
4	0.044752	0.022382	0.843901	00:25
5	0.044839	0.023134	0.831639	00:25
6	0.045287	0.022595	0.845363	00:25
7	0.045670	0.024075	0.825879	00:25
8	0.046078	0.028216	0.774827	00:25
9	0.046577	0.026030	0.804676	00:25
10	0.047041	0.028240	0.778204	00:25
11	0.047243	0.026121	0.804085	00:25
12	0.048119	0.030463	0.754129	00:25
13	0.048840	0.029273	0.765705	00:25
14	0.049276	0.032002	0.723154	00:25
15	0.049980	0.034878	0.709891	00:25
16	0.050344	0.033616	0.706110	00:25
17	0.050778	0.034624	0.697845	00:25
18	0.050819	0.036094	0.697790	00:25
19	0.051372	0.048533	0.526991	00:25
20	0.051752	0.036509	0.685943	00:25
21	0.052020	0.036692	0.670765	00:25
22	0.051954	0.035716	0.692913	00:25
23	0.052000	0.041321	0.622174	00:25
24	0.052181	0.036559	0.682900	00:25
25	0.052072	0.042601	0.619305	00:25
26	0.051939	0.035454	0.694068	00:25
27	0.052004	0.040603	0.633639	00:25
28	0.052188	0.037421	0.663873	00:25
29	0.051749	0.036913	0.679679	00:25
30	0.051804	0.034860	0.706731	00:25
31	0.051359	0.035912	0.685562	00:25
32	0.051251	0.035747	0.682998	00:25
33	0.051545	0.035277	0.685752	00:25
34	0.051339	0.031203	0.753492	00:25
35	0.050956	0.031569	0.750021	00:25



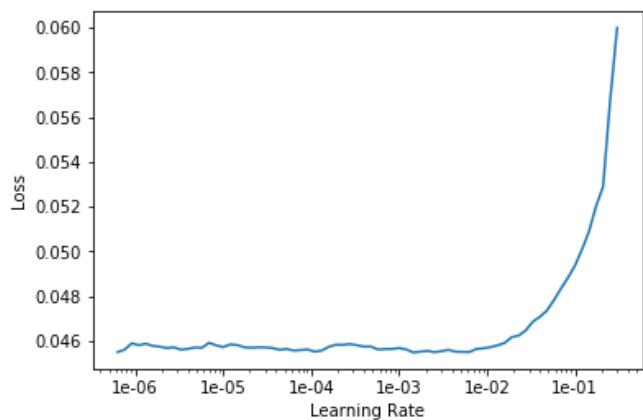
```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

```
----- CURATED+NOISY+2 - Fold 6/10  
-----
```

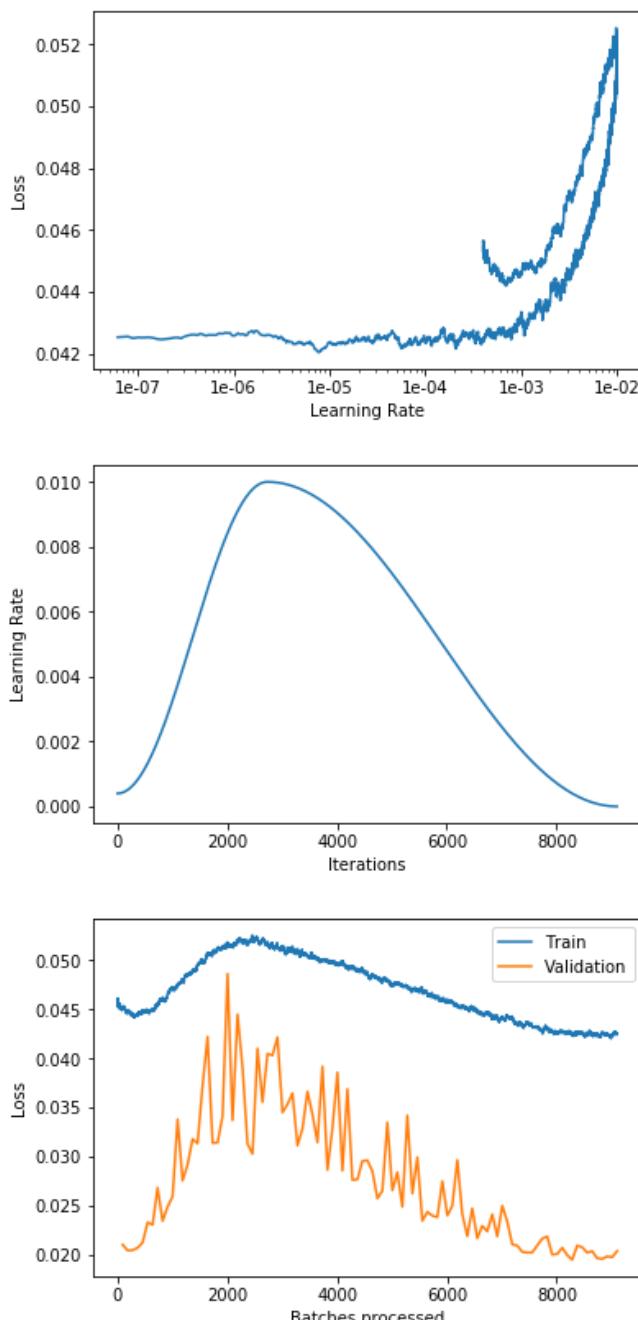


```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

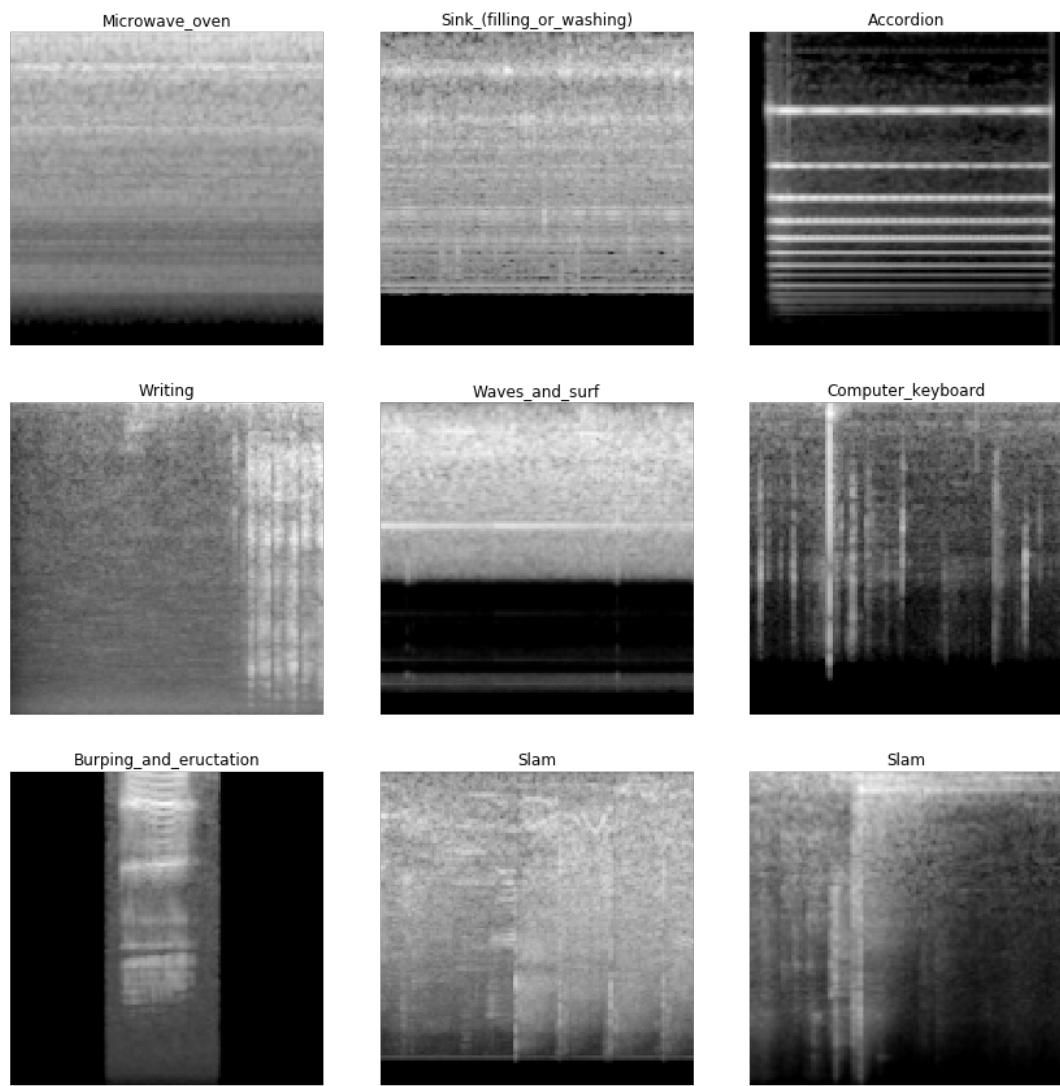


epoch	train_loss	valid_loss	lwlrap	time
0	0.045087	0.021009	0.842633	00:25
1	0.044685	0.020424	0.851296	00:25
2	0.044517	0.020458	0.854397	00:25
3	0.044481	0.020694	0.850244	00:25
4	0.044849	0.021231	0.847346	00:25
5	0.044783	0.023296	0.821920	00:25
6	0.044990	0.023048	0.813183	00:25
7	0.045531	0.026824	0.776833	00:25
8	0.046085	0.023432	0.817616	00:25
9	0.046644	0.024874	0.813148	00:25
10	0.047071	0.025933	0.791440	00:25
11	0.047532	0.033799	0.713228	00:25
12	0.047938	0.027552	0.772413	00:25
13	0.048474	0.029176	0.765144	00:25
14	0.048900	0.031796	0.734717	00:25
15	0.049529	0.031335	0.728312	00:25
16	0.050021	0.037213	0.658105	00:25
17	0.050526	0.042216	0.592901	00:25
18	0.050683	0.031384	0.737929	00:25
19	0.051231	0.031408	0.725384	00:25
20	0.051410	0.034096	0.695295	00:25
21	0.051696	0.048631	0.514281	00:25
22	0.051612	0.033693	0.704761	00:25
23	0.051851	0.044496	0.570528	00:25
24	0.051986	0.039078	0.634773	00:25
25	0.051721	0.031276	0.733442	00:25
26	0.052445	0.030281	0.737977	00:25
27	0.052198	0.041019	0.596253	00:25
28	0.052201	0.035542	0.692876	00:25
29	0.051830	0.040488	0.631482	00:25
30	0.051652	0.040312	0.630409	00:25
31	0.051512	0.042166	0.616382	00:25
32	0.051423	0.034496	0.701739	00:25
33	0.051303	0.035325	0.678638	00:25
34	0.051363	0.036449	0.675938	00:25
35	0.051165	0.031126	0.738059	00:25



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

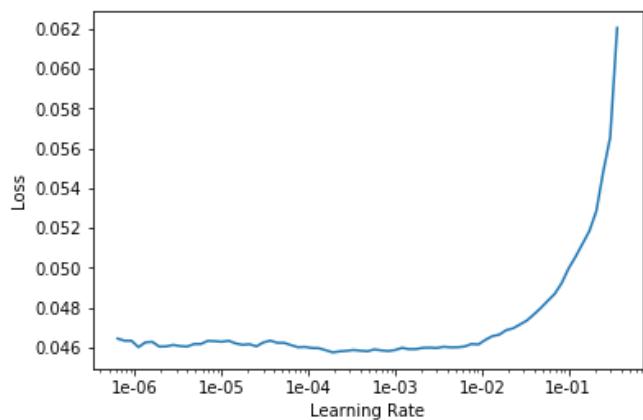
```
----- CURATED+NOISY+2 - Fold 7/10  
-----
```



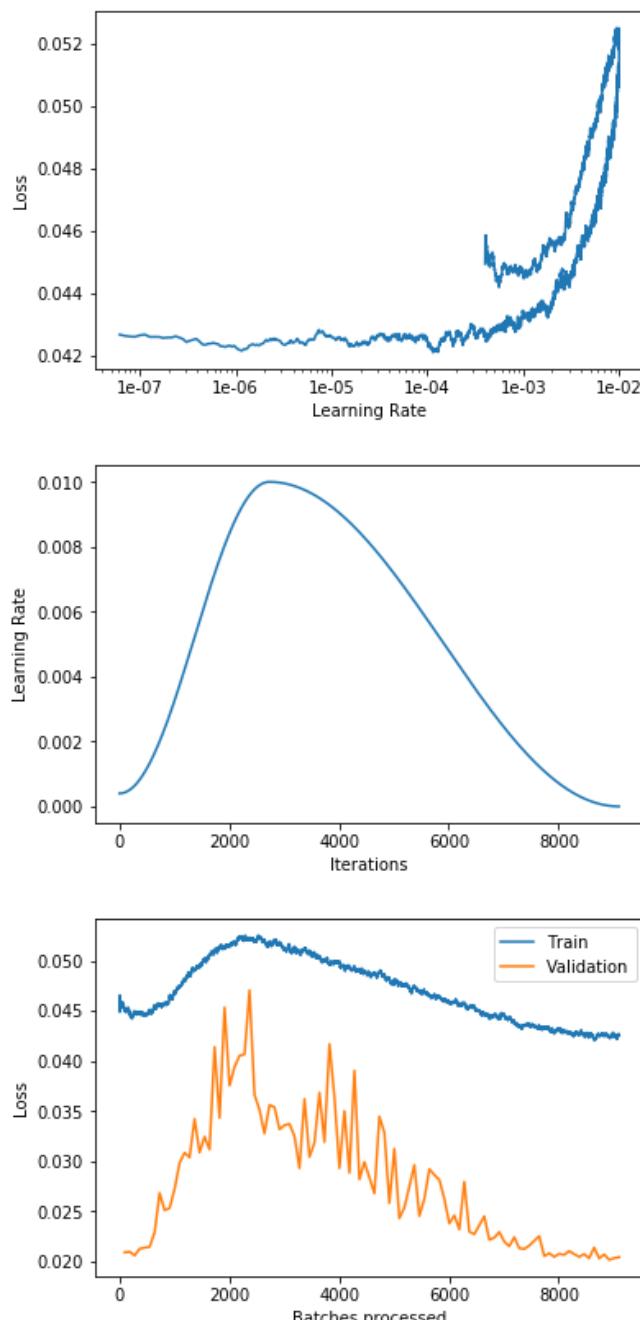
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

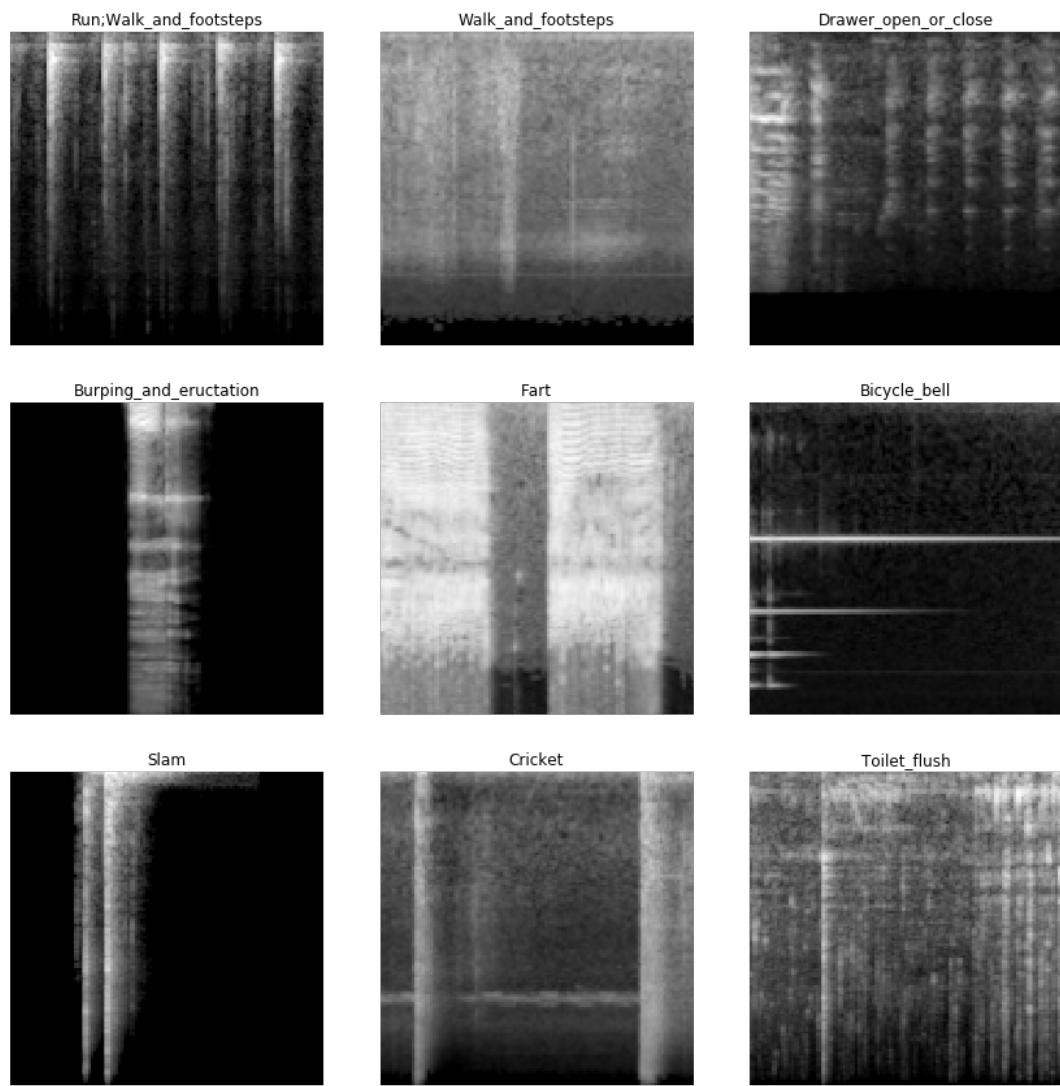


epoch	train_loss	valid_loss	lwlrap	time
0	0.045329	0.020936	0.858026	00:25
1	0.044764	0.020994	0.867423	00:25
2	0.044909	0.020625	0.855844	00:25
3	0.044785	0.021293	0.857766	00:25
4	0.044639	0.021398	0.841620	00:25
5	0.045081	0.021482	0.840105	00:25
6	0.045538	0.022920	0.827396	00:25
7	0.045632	0.026858	0.783709	00:25
8	0.045805	0.025121	0.813611	00:25
9	0.046495	0.025349	0.808950	00:25
10	0.046892	0.027252	0.787841	00:25
11	0.047432	0.029848	0.759528	00:25
12	0.048188	0.030854	0.750268	00:25
13	0.048704	0.030402	0.751913	00:25
14	0.049149	0.034221	0.712998	00:25
15	0.049597	0.030886	0.746641	00:25
16	0.050095	0.032454	0.732669	00:25
17	0.050678	0.031178	0.738871	00:25
18	0.050964	0.041400	0.640485	00:25
19	0.051364	0.034328	0.727827	00:25
20	0.051357	0.045337	0.583563	00:25
21	0.051484	0.037548	0.665594	00:25
22	0.051676	0.039403	0.649706	00:25
23	0.052237	0.040529	0.641655	00:25
24	0.052150	0.040650	0.628510	00:25
25	0.052187	0.047053	0.577420	00:25
26	0.051975	0.036627	0.679022	00:25
27	0.052338	0.035135	0.689469	00:25
28	0.052013	0.032773	0.714376	00:25
29	0.051845	0.035619	0.698354	00:25
30	0.052013	0.035391	0.693829	00:25
31	0.051823	0.033189	0.734223	00:25
32	0.051158	0.033597	0.711767	00:25
33	0.051575	0.033730	0.722130	00:25
34	0.051346	0.032498	0.742174	00:25
35	0.050892	0.029318	0.769608	00:25



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: Us
erWarning: Couldn't retrieve source code for container of type ConvBlock. It wo
n't be checked for correctness upon loading.
"type " + obj.__name__ + ". It won't be checked "
```

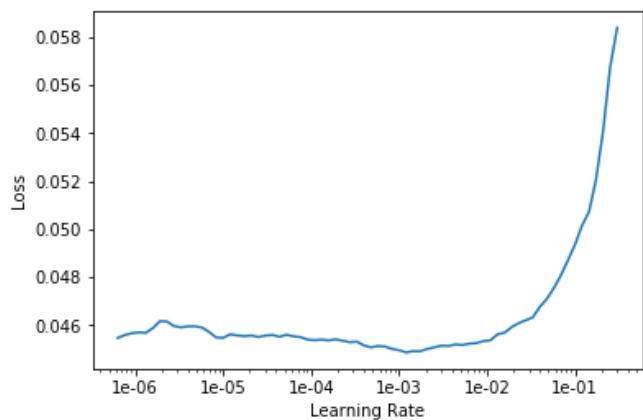
```
----- CURATED+NOISY+2 - Fold 8/10
-----
```



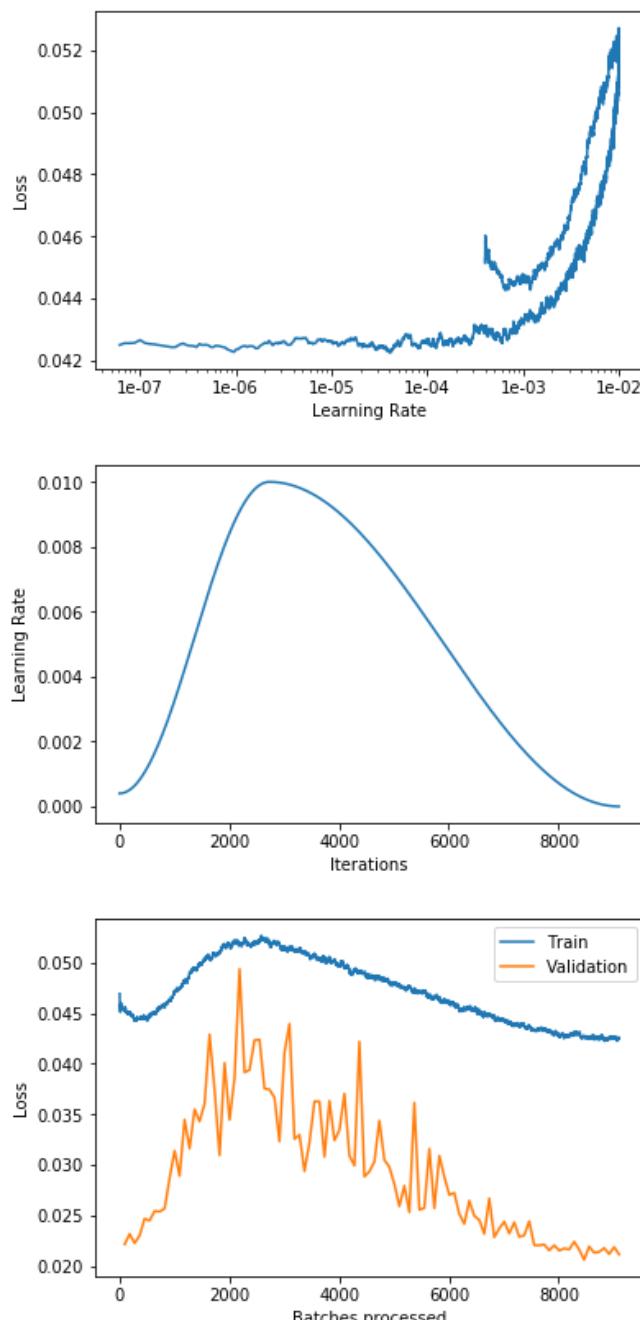
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

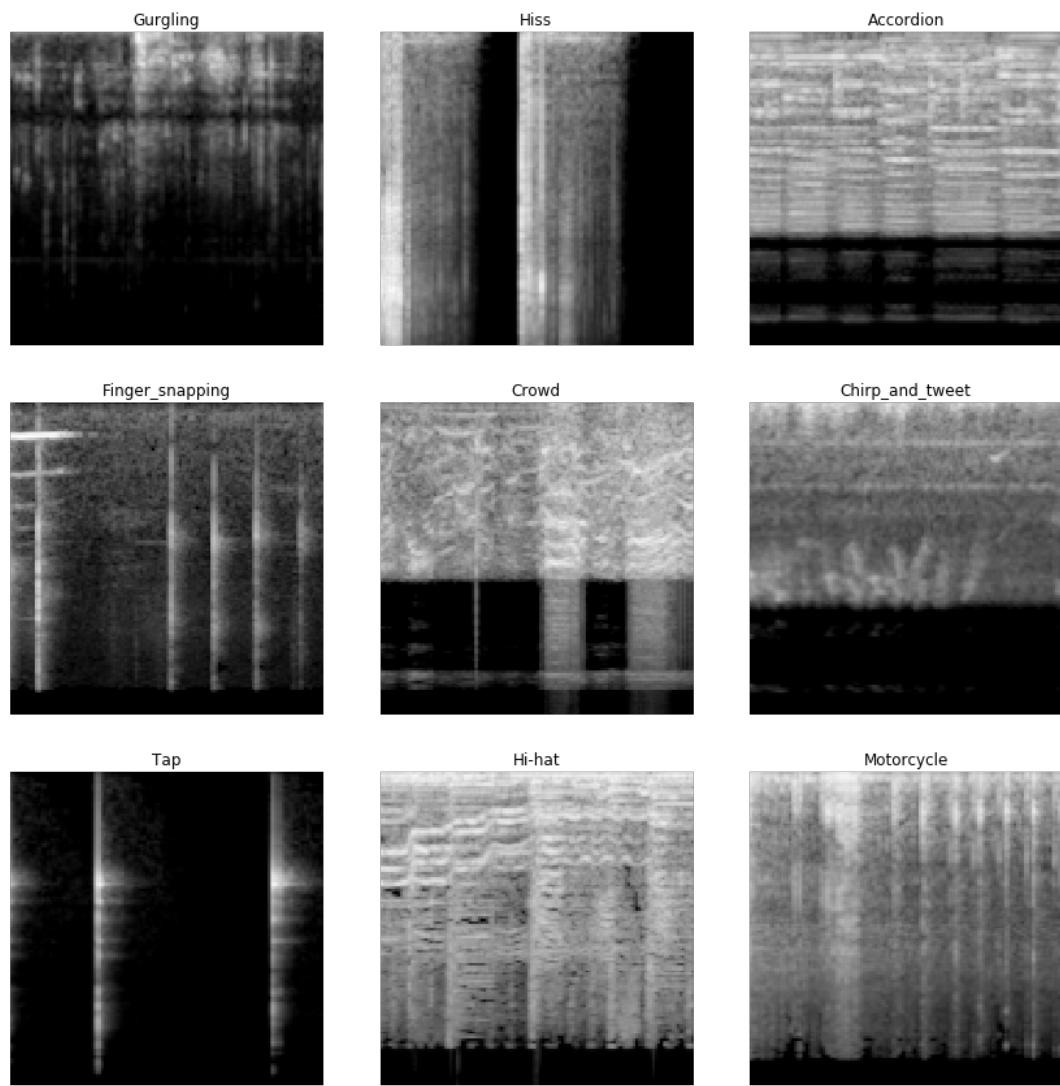


epoch	train_loss	valid_loss	lwlrap	time
0	0.045350	0.022131	0.839715	00:25
1	0.045134	0.023154	0.825232	00:25
2	0.044334	0.022236	0.840868	00:25
3	0.044522	0.023001	0.835407	00:25
4	0.044667	0.024662	0.811546	00:25
5	0.044859	0.024474	0.817020	00:25
6	0.044988	0.025426	0.804329	00:25
7	0.045664	0.025366	0.812141	00:25
8	0.045628	0.025669	0.816842	00:25
9	0.046303	0.028672	0.759739	00:25
10	0.047200	0.031401	0.747894	00:25
11	0.047404	0.028899	0.768710	00:25
12	0.048273	0.034477	0.693460	00:25
13	0.049057	0.031614	0.745836	00:25
14	0.049490	0.035496	0.704469	00:25
15	0.049902	0.034303	0.703224	00:25
16	0.050460	0.036039	0.688329	00:25
17	0.050545	0.042915	0.615571	00:25
18	0.050909	0.037484	0.665437	00:25
19	0.051170	0.030949	0.745326	00:25
20	0.051766	0.040109	0.634299	00:25
21	0.051839	0.034452	0.719640	00:25
22	0.052160	0.038578	0.672002	00:25
23	0.051953	0.049400	0.519629	00:25
24	0.052283	0.039157	0.651958	00:25
25	0.051809	0.039398	0.640746	00:25
26	0.052137	0.042347	0.626025	00:25
27	0.052327	0.042403	0.615710	00:25
28	0.052435	0.037560	0.686579	00:25
29	0.052173	0.037462	0.654493	00:25
30	0.052098	0.036663	0.672387	00:25
31	0.051908	0.032311	0.722946	00:25
32	0.051595	0.041118	0.625423	00:25
33	0.051444	0.043969	0.579514	00:25
34	0.051300	0.032568	0.712608	00:25
35	0.051300	0.032966	0.715899	00:25



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

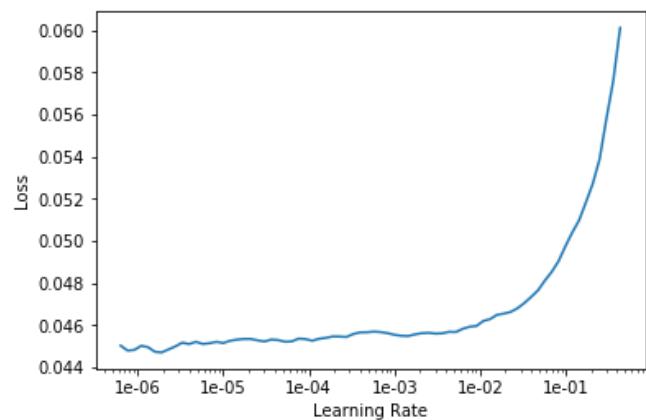
```
----- CURATED+NOISY+2 - Fold 9/10  
-----
```



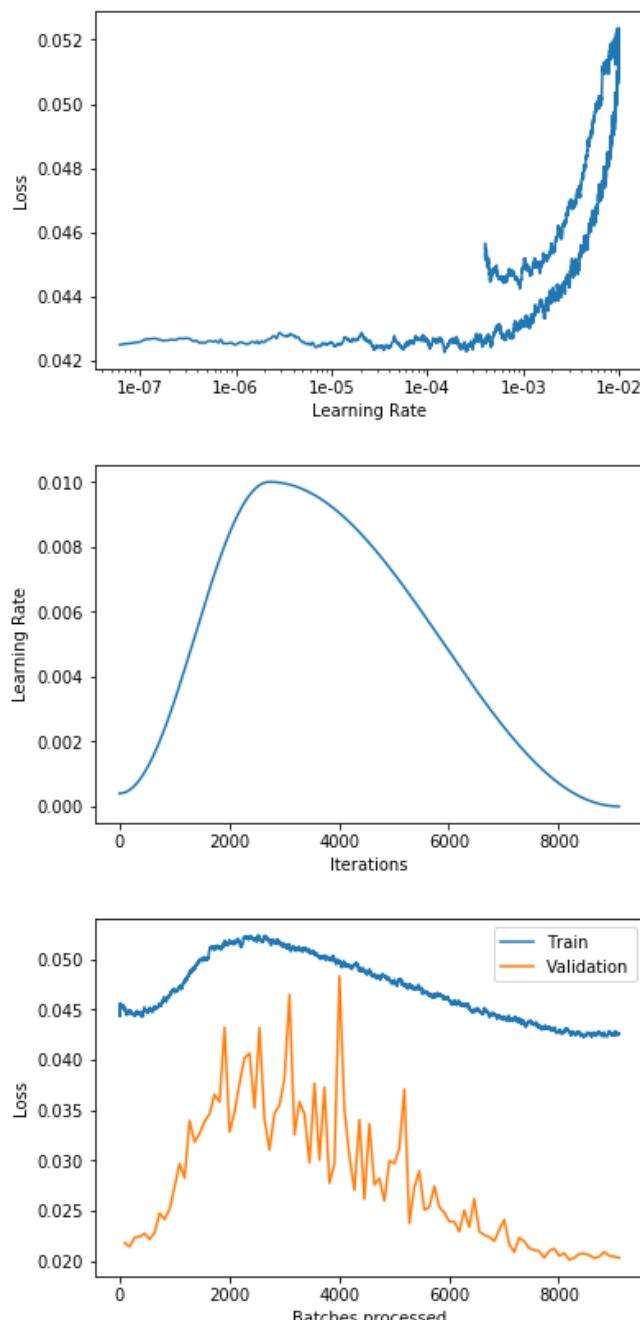
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

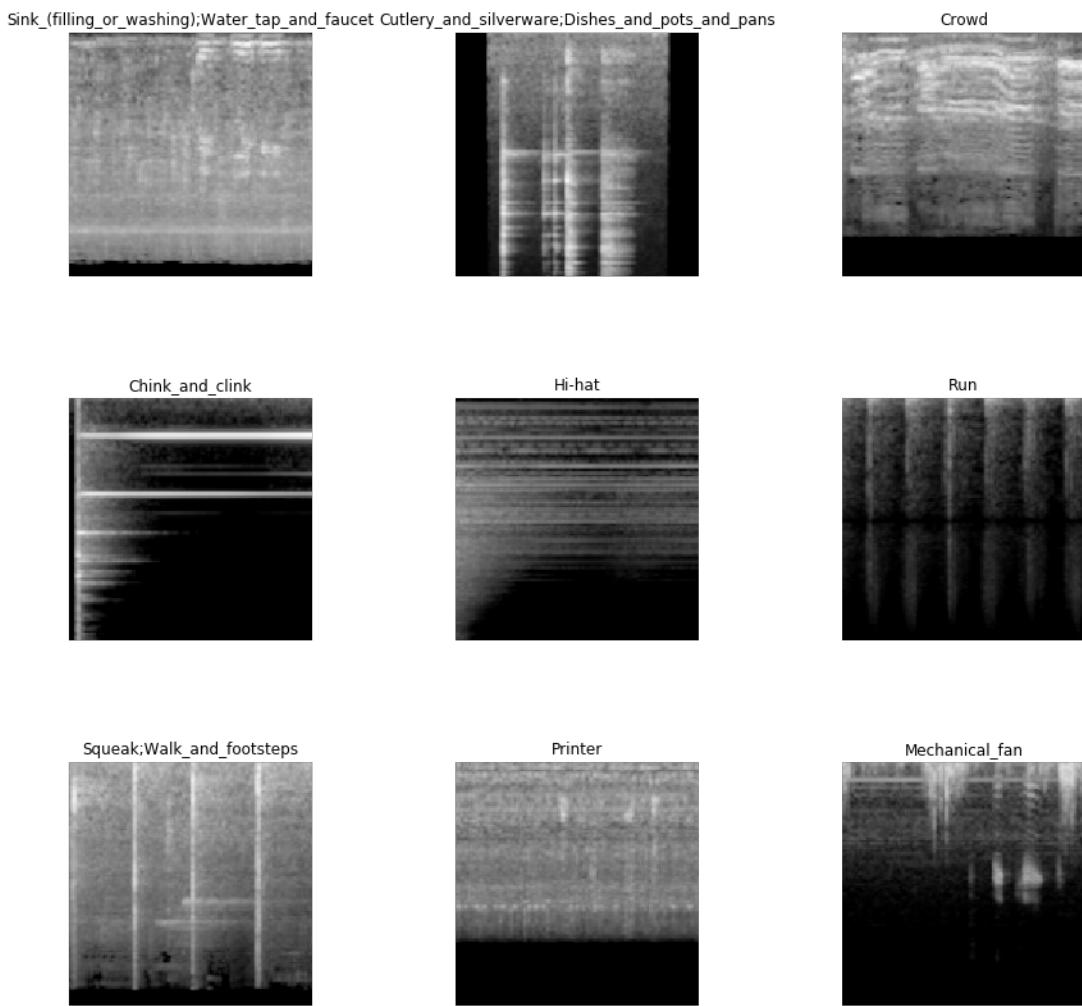


epoch	train_loss	valid_loss	lwlrap	time
0	0.044993	0.021803	0.827467	00:25
1	0.044800	0.021426	0.841152	00:25
2	0.044535	0.022355	0.836080	00:26
3	0.044658	0.022423	0.828003	00:26
4	0.045193	0.022729	0.830722	00:25
5	0.045184	0.022171	0.837914	00:25
6	0.045358	0.022818	0.832404	00:25
7	0.045625	0.024760	0.811118	00:25
8	0.046115	0.024138	0.819051	00:25
9	0.046303	0.025172	0.801480	00:25
10	0.046759	0.027398	0.787718	00:25
11	0.047279	0.029683	0.748208	00:25
12	0.047844	0.028257	0.771634	00:25
13	0.048745	0.033959	0.694849	00:25
14	0.049170	0.031846	0.728724	00:25
15	0.049845	0.032663	0.733699	00:25
16	0.050188	0.033879	0.703112	00:25
17	0.051053	0.034619	0.693361	00:25
18	0.051195	0.036559	0.664207	00:25
19	0.051275	0.035812	0.680216	00:25
20	0.051723	0.043218	0.586354	00:25
21	0.051774	0.032829	0.709797	00:25
22	0.051697	0.034804	0.707397	00:25
23	0.051774	0.037665	0.672459	00:25
24	0.052134	0.040175	0.636817	00:25
25	0.052140	0.040662	0.616127	00:25
26	0.052081	0.035235	0.680242	00:25
27	0.052108	0.043203	0.588107	00:25
28	0.052317	0.034071	0.701209	00:25
29	0.051806	0.031048	0.739443	00:25
30	0.051870	0.034690	0.694773	00:25
31	0.051634	0.035497	0.672971	00:25
32	0.051498	0.038150	0.674148	00:25
33	0.051439	0.046485	0.541313	00:25
34	0.050961	0.032579	0.723273	00:25
35	0.051147	0.035816	0.686766	00:25



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

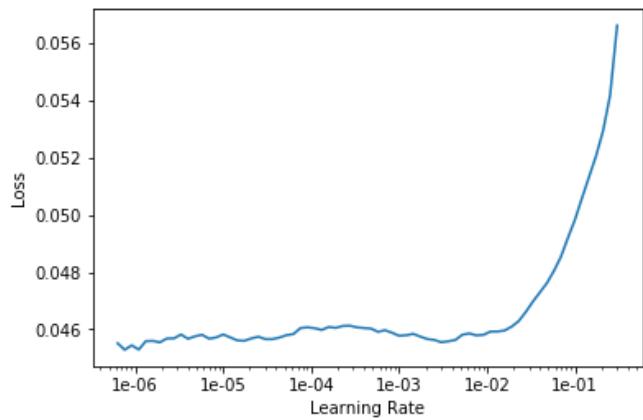
```
----- CURATED+NOISY+2 - Fold 10/10  
-----
```



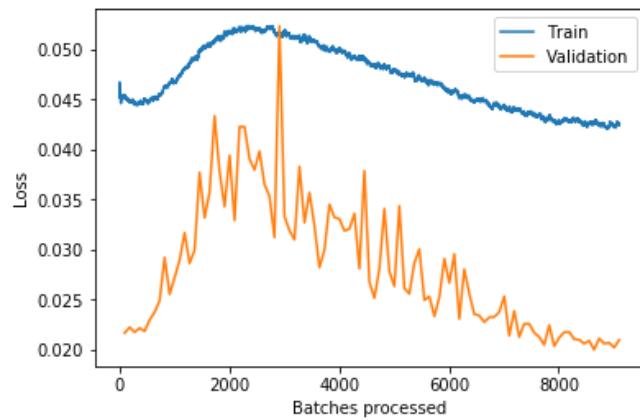
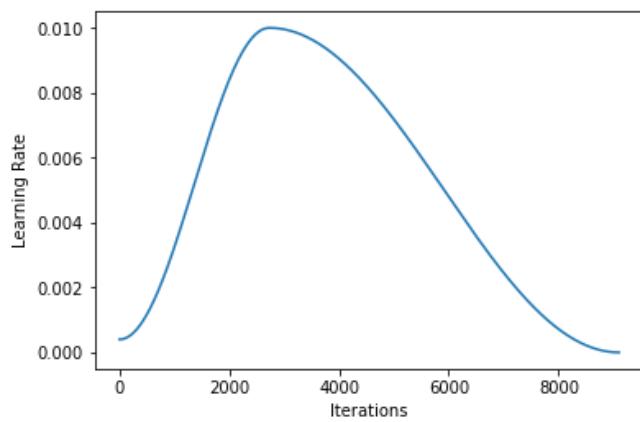
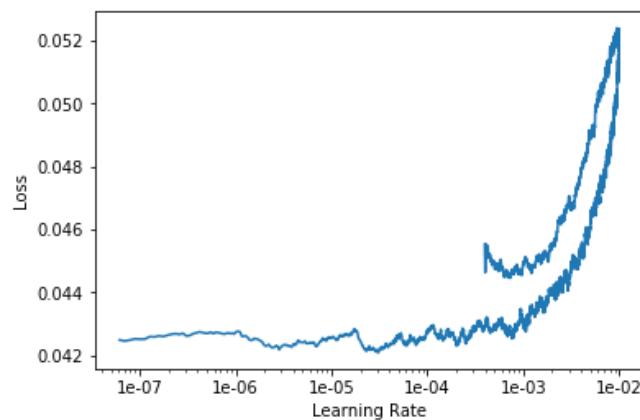
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.

"type " + obj.\_\_name\_\_ + ". It won't be checked "

LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.045427	0.021697	0.832419	00:25
1	0.044743	0.022276	0.825660	00:25
2	0.044486	0.021780	0.832564	00:25
3	0.044756	0.022190	0.826124	00:25
4	0.045080	0.021872	0.832989	00:25
5	0.044825	0.022997	0.828090	00:25
6	0.045183	0.023768	0.819289	00:25
7	0.045506	0.024924	0.795446	00:25
8	0.046322	0.029238	0.758253	00:25
9	0.046599	0.025563	0.804177	00:25
10	0.046712	0.027234	0.787818	00:25
11	0.047385	0.028976	0.760514	00:25
12	0.048051	0.031695	0.738514	00:25
13	0.048710	0.028660	0.771315	00:25
14	0.049119	0.029961	0.755926	00:25
15	0.049828	0.037721	0.657436	00:25
16	0.050278	0.033196	0.718986	00:25
17	0.050421	0.035718	0.668535	00:25
18	0.050890	0.043371	0.608266	00:25
19	0.051370	0.037841	0.654647	00:25
20	0.051718	0.034307	0.706171	00:25
21	0.051890	0.039421	0.660365	00:25
22	0.052037	0.032954	0.708540	00:25
23	0.052235	0.042304	0.589032	00:25
24	0.052082	0.042283	0.619726	00:25
25	0.052333	0.039062	0.647595	00:25
26	0.052127	0.037992	0.676852	00:25
27	0.052099	0.039849	0.633047	00:25
28	0.051820	0.036537	0.661641	00:25
29	0.052145	0.035336	0.685986	00:25
30	0.051945	0.031249	0.730280	00:25
31	0.051782	0.052341	0.495584	00:25
32	0.051721	0.033340	0.729363	00:25
33	0.051525	0.031936	0.718732	00:25
34	0.051268	0.031021	0.745373	00:25
35	0.051344	0.038310	0.663290	00:25



```
/home/eric/anaconda3/lib/python3.7/site-packages/torch/serialization.py:251: UserWarning: Couldn't retrieve source code for container of type ConvBlock. It won't be checked for correctness upon loading.  
"type " + obj.__name__ + ". It won't be checked "
```

```
In [126]: kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-11_fold-{fold}')
```

Overall lwlrap on CURATED dataset: 0.8771791215725213

	lwlrap	weight
Fill_(with_liquid)	0.604929	0.008693
Squeak	0.633349	0.013039
Walk_and_footsteps	0.694341	0.013039
Hiss	0.722582	0.013039
Mechanical_fan	0.729417	0.008519
Buzz	0.756949	0.009736
Tap	0.763608	0.013039
Chink_and_clink	0.767967	0.013039
Bus	0.768223	0.013039
Yell	0.771113	0.013039
Frying_(food)	0.778196	0.010953
Traffic_noise_and_roadway_noise	0.785805	0.013039
Cutlery_and_silverware	0.786109	0.013039
Water_tap_and_faucet	0.786708	0.013039
Sink_(filling_or_washing)	0.796633	0.013039
Trickle_and_dribble	0.797200	0.009214
Male_speech_and_manSpeaking	0.808048	0.013039
Motorcycle	0.808794	0.013039
Clapping	0.814652	0.013039
Crowd	0.824471	0.013039
Stream	0.826571	0.013039
Slam	0.828571	0.013039
Bathtub_(filling_or_washing)	0.830581	0.013039
Dishes_and_pots_and_pans	0.833289	0.013039
Run	0.834745	0.013039
Microwave_oven	0.839947	0.013039
Car_passing_by	0.842000	0.013039
Waves_and_surf	0.853302	0.013039
Gurgling	0.853802	0.013039
Sneeze	0.862469	0.010953
...	...	...
Keys_jangling	0.923778	0.013039
Race_car_and_auto_racing	0.924745	0.009736
Writing	0.927600	0.013039
Screaming	0.929111	0.013039

```
In [127]: kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-11_fold-{fold}')
```

Overall lwlrapp on NOISY dataset: 0.5759039317915355

	<b>lwlrap</b>	<b>weight</b>
Burping_and_eructation	0.090100	0.0125
Cupboard_open_or_close	0.100206	0.0125
Finger_snapping	0.109617	0.0125
Sigh	0.178116	0.0125
Gasp	0.199521	0.0125
Zipper_(clothing)	0.207674	0.0125
Fart	0.249710	0.0125
Shatter	0.258938	0.0125
Writing	0.266425	0.0125
Raindrop	0.274716	0.0125
Keys_jangling	0.290114	0.0125
Scissors	0.300699	0.0125
Sneeze	0.317570	0.0125
Buzz	0.326343	0.0125
Tap	0.335593	0.0125
Computer_keyboard	0.340296	0.0125
Slam	0.377709	0.0125
Tick-tock	0.378701	0.0125
Knock	0.410644	0.0125
Drip	0.433027	0.0125
Chink_and_clink	0.456415	0.0125
Purr	0.468610	0.0125
Bicycle_bell	0.480672	0.0125
Chewing_and_mastication	0.481202	0.0125
Bass_guitar	0.493159	0.0125
Toilet_flush	0.505264	0.0125
Fill_(with_liquid)	0.524587	0.0125
Skateboard	0.529284	0.0125
Crackle	0.530347	0.0125
Squeak	0.533910	0.0125
...	...	...
Applause	0.686390	0.0125
Screaming	0.689811	0.0125
Female_singing	0.690193	0.0125
Clapping	0.691024	0.0125

## Test prediction and submission file creation

- Switch to test data.
- Overwrite results to sample submission; simple way to prepare submission file.

```
In [40]: X_test = pickle.load(open(MELS_TEST, 'rb'))
CUR_X_FILES, CUR_X = list(test_df.fname.values), X_test
```

```
In [41]: test_df_multi = pd.DataFrame(columns=test_df.columns)

for index, row in tqdm_notebook(test_df.iterrows(), total=test_df.shape[0]):
    idx = CUR_X_FILES.index(row.fname)
    x = PIL.Image.fromarray(CUR_X[idx])
    time_dim, base_dim = x.size
    s = math.ceil((time_dim-conf.n_mels) / TTA_SHIFT) + 1

    fname = row.fname
    tmp = pd.DataFrame(columns=test_df.columns)
    for crop_x in [int(np.around((time_dim-conf.n_mels)*x/(s-1))) if s != 1 else 0 for x in range(s)]:
        row.fname = fname + '!' + str(crop_x)
        tmp = tmp.append(row)
    test_df_multi = test_df_multi.append(tmp)

test_df.shape, test_df_multi.shape
```

Out[41]: ((1120, 81), (19007, 81))

```
In [42]: test = ImageList.from_df(test_df_multi, WORK)
```

```
In [43]: for fold in range(n_splits):
    learn = load_learner(WORK, f'stage-10_fold-{fold}.pkl', test=test)
    preds, _ = learn.get_preds(ds_type=DatasetType.Test)
    preds = preds.cpu().numpy()
#    preds = pd.DataFrame(preds).rank(1) / preds.shape[1]
    if fold == 0:
        predictions = preds
    else:
        predictions += preds
    if TOY_MODE:
        break
predictions /= n_splits
```

```
In [44]: test_df_multi[learn.data.classes] = predictions
test_df_multi['fname'] = test_df_multi.fname.apply(lambda x: x.split('!')[0])
```

```
In [45]: test_df_multi.head()
```

Out[45]:

	fname	Accelerating_and_revving_and_vroom	Accordion	Acoustic_guitar	Applause
0	000ccb97.wav	0.00155336	0.000586058	0.00220364	0.00108214
1	0012633b.wav	0.216832	0.00053429	0.00112215	0.00952838
1	0012633b.wav	0.230812	0.000695401	0.00182883	0.00876809
1	0012633b.wav	0.175237	0.000861128	0.00281544	0.0088084
1	0012633b.wav	0.248624	0.00229723	0.00255869	0.00528565

5 rows × 81 columns

```
In [46]: submission = test_df_multi.infer_objects().groupby('fname').mean().reset_index()
```

```
In [47]: submission.to_csv('submission.csv', index=False)
submission.head()
```

Out[47]:

	fname	Accelerating_and_revving_and_vroom	Accordion	Acoustic_guitar	Applause	
0	000ccb97.wav	0.001553	0.000586	0.002204	0.001082	0.00
1	0012633b.wav	0.252510	0.001052	0.002137	0.006514	0.00
2	001ed5f1.wav	0.003829	0.002567	0.001460	0.006579	0.00
3	00294be0.wav	0.000610	0.000038	0.001022	0.000200	0.00
4	003fde7a.wav	0.001541	0.001391	0.001930	0.001363	0.00

5 rows × 81 columns

```
In [48]: submission.set_index('fname').idxmax(1)
```

```
Out[48]: fname
000ccb97.wav           Hi-hat
0012633b.wav          Motorcycle
001ed5f1.wav           Run
00294be0.wav           Purr
003fde7a.wav          Bicycle_bell
0040ccc9.wav          Electric_guitar
0046b732.wav           Meow
004f3bbc.wav          Bass_guitar
00526050.wav          Bass_drum
00559da4.wav          Zipper_(clothing)
00582bbe.wav           Knock
0064aedf.wav          Drawer_open_or_close
0065512b.wav          Skateboard
006a91d2.wav           Stream
006ea9ee.wav           Strum
006f9dca.wav          Sink_(filling_or_washing)
007450dc.wav           Screaming
00979c8a.wav           Purr
00992464.wav          Sink_(filling_or_washing)
00b44a8a.wav          Cutlery_and_silverware
00bfaaaaf.wav          Gasp
00cf9680.wav           Stream
00eae94d.wav          Bathtub_(filling_or_washing)
011ec5b8.wav          Drawer_open_or_close
013200dc.wav           Fart
01440c2e.wav           Harmonica
01567a3d.wav          Microwave_oven
0163d203.wav           Purr
017c24b2.wav           Meow
01a992c5.wav          Cricket
...
406306b6.wav           Gong
406a1cc1.wav           Knock
406ade5e.wav          Trickle_and_dribble
4093913f.wav          Toilet_flush
40957335.wav           Gasp
4098028d.wav          Computer_keyboard
40afd1b3.wav           Whispering
40c1ed87.wav          Burping_and_eructation
40c50046.wav          Computer_keyboard
40d0726d.wav          Car_passing_by
40dd1274.wav          Female_speech_and_womanSpeaking
40e25750.wav          Burping_and_eructation
410837eb.wav          Acoustic_guitar
410c2138.wav          Toilet_flush
411993fd.wav           Hi-hat
413b5427.wav           Harmonica
41522a65.wav           Cricket
415f4c8f.wav          Keys_jangling
41759973.wav           Stream
419271cc.wav          Bicycle_bell
4195cb1a.wav          Marimba_and_xylophone
41af688a.wav           Fart
41c60bba.wav          Microwave_oven
41cfee6d.wav           Accordion
41f16193.wav           Slam
41f1ea4a.wav          Traffic_noise_and_roadway_noise
41f86bc4.wav          Toilet_flush
4215309a.wav           Scissors
4248d196.wav           Applause
42542036.wav          Race_car_and_auto_racing
Length: 1120, dtype: object
```



## Freesound Audio Taggin 2019 - Solution Part 2 (VGG16)

This is based upon

- <https://www.kaggle.com/daisukelab/cnn-2d-basic-solution-powered-by-fast-ai> (<https://www.kaggle.com/daisukelab/cnn-2d-basic-solution-powered-by-fast-ai>)
- Based on the 2nd version. <https://www.kaggle.com/daisukelab/clf-to-multi-cnn-2d-basic-2-preprocessed-dataset> (<https://www.kaggle.com/daisukelab/clf-to-multi-cnn-2d-basic-2-preprocessed-dataset>)
- Borrowing model from <https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch> (<https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch>)
- Using preprocessed dataset. [https://www.kaggle.com/daisukelab/fat2019\\_prep\\_mels1](https://www.kaggle.com/daisukelab/fat2019_prep_mels1) ([https://www.kaggle.com/daisukelab/fat2019\\_prep\\_mels1](https://www.kaggle.com/daisukelab/fat2019_prep_mels1))

Please read README.md for more information like SpecMix...

```
In [1]: import os
from pathlib import Path
import pickle
import random
import time
import warnings
from io import StringIO
from csv import writer

import numpy as np
import pandas as pd
from scipy.stats import rankdata
from sklearn.model_selection import KFold
import librosa
import librosa.display
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
import IPython
import IPython.display
import PIL
import seaborn as sns

import torch
import torch.nn as nn
import torch.nn.functional as F

from fastai import *
from fastai.vision import *
from fastai.vision.data import *
from fastai.callbacks import *
```

```
In [2]: def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True

SEED = 2019
seed_everything(SEED)
```

```
In [3]: TOY_MODE = False
TTA_SHIFT = 32 # TTA: predict every TTA_SHIFT
bs=128
n_splits = 10
```

```
In [4]: DATA = Path('../input')
PREPROCESSED = Path('/media/eric/Data/freesound-audio-tagging-2019/preprocessed')
WORK = Path('work')
Path(WORK).mkdir(exist_ok=True, parents=True)
Path(PREPROCESSED).mkdir(exist_ok=True, parents=True)

CSV_TRN_CURATED = DATA/'train_curated.csv'
CSV_TRN_NOISY = DATA/'train_noisy.csv'
CSV_SUBMISSION = DATA/'sample_submission.csv'

MELS_TRN_CURATED = PREPROCESSED/'mels_train_curated.pkl'
MELS_TRN_NOISY = PREPROCESSED/'mels_train_noisy.pkl'
MELS_TEST = PREPROCESSED/'mels_test.pkl'

DATA_CURATED = DATA/'train_curated'
DATA_NOISY = DATA/'train_noisy'
DATA_TEST = DATA/'test'

trn_curated_df = pd.read_csv(CSV_TRN_CURATED)
trn_noisy_df = pd.read_csv(CSV_TRN_NOISY)
test_df = pd.read_csv(CSV_SUBMISSION)
```

```
In [5]: # # Drop samples with incorrect label
# # Discussion from organizers: https://www.kaggle.com/c/freesound-audio-tagging-2019/discussion/93480
# useless = ['f76181c4.wav', '77b925c2.wav', '6a1f682a.wav', 'c7db12aa.wav', '7752cc8a.wav', '1d44b0bd.wav']
# trn_curated_df = trn_curated_df[~trn_curated_df.fname.isin(useless)]
```

## Audio conversion to 2D

Almost copied from my repository: <https://github.com/daisukelab/ml-sound-classifier> (<https://github.com/daisukelab/ml-sound-classifier>)

Handle sampling rate 44.1kHz as is, no information loss.  
 Size of each file will be 128 x L, L is audio seconds x 128; [128, 256] if sound is 2 s long.  
 Convert to Mel-spectrogram, not MFCC. We are handling general sound rather than human voice. <https://en.wikipedia.org/wiki/Spectrogram>

```
In [6]: def read_audio(conf, pathname, trim_long_data):
    y, sr = librosa.load(pathname, sr=conf.sampling_rate)
    # trim silence
    if 0 < len(y): # workaround: 0 length causes error
        y, _ = librosa.effects.trim(y) # trim, top_db=default(60)
    # make it unified length to conf.samples
    if len(y) > conf.samples: # long enough
        if trim_long_data:
            y = y[0:0+conf.samples]
    else: # pad blank
        padding = conf.samples - len(y)      # add padding at both ends
        offset = padding // 2
        y = np.pad(y, (offset, conf.samples - len(y) - offset), 'constant')
    return y

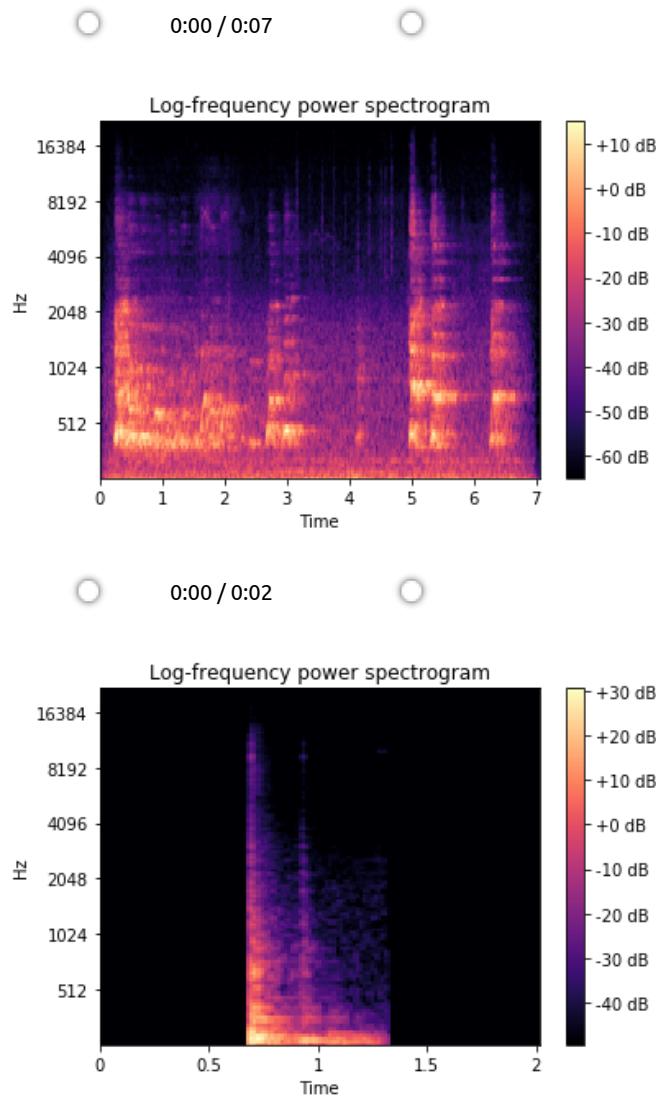
def audio_to_melspectrogram(conf, audio):
    spectrogram = librosa.feature.melspectrogram(audio,
                                                sr=conf.sampling_rate,
                                                n_mels=conf.n_mels,
                                                hop_length=conf.hop_length,
                                                n_fft=conf.n_fft,
                                                fmin=conf.fmin,
                                                fmax=conf.fmax)
    spectrogram = librosa.power_to_db(spectrogram)
    spectrogram = spectrogram.astype(np.float32)
    return spectrogram

def show_melspectrogram(conf, mels, title='Log-frequency power spectrogram'):
    librosa.display.specshow(mels, x_axis='time', y_axis='mel',
                            sr=conf.sampling_rate, hop_length=conf.hop_length,
                            fmin=conf.fmin, fmax=conf.fmax)
    plt.colorbar(format='%+2.0f dB')
    plt.title(title)
    plt.show()

def read_as_melspectrogram(conf, pathname, trim_long_data, debug_display=False):
    x = read_audio(conf, pathname, trim_long_data)
    mels = audio_to_melspectrogram(conf, x)
    if debug_display:
        IPython.display.display(IPython.display.Audio(x, rate=conf.sampling_rate))
    show_melspectrogram(conf, mels)
    return mels

class conf:
    # Preprocessing settings
    sampling_rate = 44100
    duration = 2
    hop_length = 347*duration # to make time steps 128
    fmin = 20
    fmax = sampling_rate // 2
    n_mels = 128
    n_fft = n_mels * 20
    samples = sampling_rate * duration

    # example
    x = read_as_melspectrogram(conf, DATA_CURATED/'0006ae4e.wav', trim_long_data=False,
                               debug_display=True)
    x = read_as_melspectrogram(conf, DATA_CURATED/'05e4ad19.wav', trim_long_data=False,
                               debug_display=True)
```



### Making 2D mel-spectrogram data as 2D 3ch images

So that normal CNN image classifier can handle.

```
In [7]: def mono_to_color(X, mean=None, std=None, norm_max=None, norm_min=None, eps=1e-6):
    # Stack X as [X,X,X]
    X = np.stack([X, X, X], axis=-1)

    # Standardize
    mean = mean or X.mean()
    std = std or X.std()
    Xstd = (X - mean) / (std + eps)
    _min, _max = Xstd.min(), Xstd.max()
    norm_max = norm_max or _max
    norm_min = norm_min or _min
    if (_max - _min) > eps:
        # Scale to [0, 255]
        V = Xstd
        V[V < norm_min] = norm_min
        V[V > norm_max] = norm_max
        V = 255 * (V - norm_min) / (norm_max - norm_min)
        V = V.astype(np.uint8)
    else:
        # Just zero
        V = np.zeros_like(Xstd, dtype=np.uint8)
    return V

def convert_wav_to_image(df, source, img_dest):
    print(f'Converting {source} -> {img_dest}')
    X = []
    for i, row in tqdm_notebook(df.iterrows(), total=df.shape[0]):
        x = read_as_melspectrogram(conf, source=str(row.fname), trim_long_data=False)
        x_color = mono_to_color(x)
        X.append(x_color)
    pickle.dump(X, open(img_dest, 'wb'))
    return X
```

```
In [8]: if not MELS_TRN_CURATED.is_file():
    convert_wav_to_image(trn_curated_df, source=DATA_CURATED, img_dest=MELS_TRN_CURATED)

if not MELS_TRN_NOISY.is_file():
    convert_wav_to_image(trn_noisy_df, source=DATA_NOISY, img_dest=MELS_TRN NOI SY)

if not MELS_TEST.is_file():
    convert_wav_to_image(test_df, source=DATA_TEST, img_dest=MELS_TEST)
```

## Official Metric

```
In [9]: # from official code https://colab.research.google.com/drive/1AgPdhSp7ttY1803fEOH0QKlt_3HJDLi8#scrollTo=cRCaC1b9ogU
def _one_sample_positive_class_precisions(scores, truth):
    """Calculate precisions for each true class for a single sample.

    Args:
        scores: np.array of (num_classes,) giving the individual classifier scores.
        truth: np.array of (num_classes,) bools indicating which classes are true.

    Returns:
        pos_class_indices: np.array of indices of the true classes for this sample.
        pos_class_precisions: np.array of precisions corresponding to each of those classes.
    """
    num_classes = scores.shape[0]
    pos_class_indices = np.flatnonzero(truth > 0)
    # Only calculate precisions if there are some true classes.
    if not len(pos_class_indices):
        return pos_class_indices, np.zeros(0)
    # Retrieval list of classes for this sample.
    retrieved_classes = np.argsort(scores)[::-1]
    # class_rankings[top_scoring_class_index] == 0 etc.
    class_rankings = np.zeros(num_classes, dtype=np.int)
    class_rankings[pos_class_indices] = range(num_classes)
    # Which of these is a true label?
    retrieved_class_true = np.zeros(num_classes, dtype=np.bool)
    retrieved_class_true[class_rankings[pos_class_indices]] = True
    # Num hits for every truncated retrieval list.
    retrieved_cumulative_hits = np.cumsum(retrieved_class_true)
    # Precision of retrieval list truncated at each hit, in order of pos_label
    precision_at_hits = (
        retrieved_cumulative_hits[class_rankings[pos_class_indices]] /
        (1 + class_rankings[pos_class_indices].astype(np.float)))
    return pos_class_indices, precision_at_hits

def calculate_per_class_lwlrap(truth, scores):
    """Calculate label-weighted label-ranking average precision.

    Arguments:
        truth: np.array of (num_samples, num_classes) giving boolean ground-truth
               of presence of that class in that sample.
        scores: np.array of (num_samples, num_classes) giving the classifier-uncertainty's
               real-valued score for each class for each sample.

    Returns:
        per_class_lwlrap: np.array of (num_classes,) giving the lwlrap for each class.
        weight_per_class: np.array of (num_classes,) giving the prior of each class within the truth labels. Then the overall unbalanced lwlrap is simply np.sum(per_class_lwlrap * weight_per_class)
    """
    assert truth.shape == scores.shape
    num_samples, num_classes = scores.shape
    # Space to store a distinct precision value for each class on each sample.
    # Only the classes that are true for each sample will be filled in.
    precisions_for_samples_by_classes = np.zeros((num_samples, num_classes))
```

```
In [10]: class Lwlrap(Callback):
    """ class to compute lwlrap during fastai training"""

    def on_epoch_begin(self, **kwargs):
        self.accumulator = lwlrap_accumulator()

    def on_batch_end(self, last_output, last_target, **kwargs):
        self.accumulator.accumulate_samples(last_target.cpu().numpy(), torch.sigmoid(last_output).cpu().numpy())

    def on_epoch_end(self, last_metrics, **kwargs):
        return add_metrics(last_metrics, self.accumulator.overall_lwlrap())
```

```
In [11]: def _sliding_df(df):
    output = StringIO()
    csv_writer = writer(output)
    csv_writer.writerow(df.columns)

    for index, row in tqdm_notebook(df.iterrows(), total=df.shape[0]):
        idx = CUR_X_FILES.index(row.fname)
        time_dim = CUR_X[idx].shape[1]
        s = math.ceil((time_dim-conf.n_mels) / TTA_SHIFT) + 1

        fname = row.fname
        for crop_x in [int(np.around((time_dim-conf.n_mels)*x/(s-1))) if s != 1
        else 0 for x in range(s)]:
            row.fname = fname + '!' + str(crop_x)
            csv_writer.writerow(row)

    output.seek(0)
    return pd.read_csv(output).reset_index(drop=True)

def kfold_prediction(kf, df, file_pattern):
    """ Compute prediction, thruth, index on validation set on models trained using K-Fold """
    overall_preds = None
    overall_thruth = None
    overall_index = None

    for fold, (train_index, valid_index) in enumerate(kf.split(df)):
        print(f'Fold {fold+1}/{n_splits}')

        filename = file_pattern.format(fold=fold)
        learn.load(filename)

        df_multi = _sliding_df(df.iloc[valid_index])

        src = (ImageList.from_df(df_multi, WORK)
               .split_by_idxs(df_multi.index, df_multi.index)
               .label_from_df(label_delim=','))
    )
        data = (src.transform(tfms, size=128)
                .databunch(bs=bs))
    )

    learn.data = data

    # Ensure we have enough data
    assert learn.data.classes == labels, set(labels) - set(learn.data.classes)

    preds, thruth = learn.get_preds(ds_type=DatasetType.Valid)

    fname = df_multi.fname.apply(lambda x: x.split('!')[0])

    preds = pd.DataFrame(preds.cpu().numpy())
    preds['fname'] = fname
    preds = preds.groupby('fname').mean().reset_index(drop=True)

    thruth = pd.DataFrame(thruth.cpu().numpy())
    thruth['fname'] = fname
    thruth = thruth.groupby('fname').mean().reset_index(drop=True)
```

## Borrowed model

Thanks to <https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch> (<https://www.kaggle.com/mhiro2/simple-2d-cnn-classifier-with-pytorch>), borrowing simple model here.

```
In [12]: # class ConvBlock(nn.Module):
#     def __init__(self, in_channels, out_channels):
#         super().__init__()

#         self.conv1 = nn.Sequential(
#             nn.Conv2d(in_channels, out_channels, 3, 1, 1),
#             nn.BatchNorm2d(out_channels),
#             nn.ReLU(),
#         )
#         self.conv2 = nn.Sequential(
#             nn.Conv2d(out_channels, out_channels, 3, 1, 1),
#             nn.BatchNorm2d(out_channels),
#             nn.ReLU(),
#         )

#         self._init_weights()

#     def _init_weights(self):
#         for m in self.modules():
#             if isinstance(m, nn.Conv2d):
#                 nn.init.kaiming_normal_(m.weight)
#                 if m.bias is not None:
#                     nn.init.zeros_(m.bias)
#             elif isinstance(m, nn.BatchNorm2d):
#                 nn.init.constant_(m.weight, 1)
#                 nn.init.zeros_(m.bias)

#     def forward(self, x):
#         x = self.conv1(x)
#         x = self.conv2(x)
#         x = F.avg_pool2d(x, 2)
#         return x

# class Classifier(nn.Module):
#     def __init__(self, num_classes=1000): # ===== modification to comply
#         fast.ai
#         super().__init__()

#         self.conv = nn.Sequential(
#             ConvBlock(in_channels=3, out_channels=64),
#             ConvBlock(in_channels=64, out_channels=128),
#             ConvBlock(in_channels=128, out_channels=256),
#             ConvBlock(in_channels=256, out_channels=512),
#         )
#         self.avgpool = nn.AdaptiveAvgPool2d((1, 1)) # ===== modification
#         to comply fast.ai
#         self.fc = nn.Sequential(
#             nn.Dropout(0.2),
#             nn.Linear(512, 128),
#             nn.PReLU(),
#             nn.BatchNorm1d(128),
#             nn.Dropout(0.1),
#             nn.Linear(128, num_classes),
#         )

#     def forward(self, x):
#         x = self.conv(x)
#         #x = torch.mean(x, dim=3) # ===== modification to comply fast.
#         ai
#         #x, _ = torch.max(x, dim=2) # ===== modification to comply fast.
#         ai
#         x = self.avgpool(x) # ===== modification to comply fast.
#         ai
```

## File/folder definitions

- df will handle training data.
- test\_df will handle test data.

```
In [13]: df = pd.concat([trn_curated_df, trn_noisy_df], ignore_index=True, sort=True)

X_train = pickle.load(open(MELS_TRN_CURATED, 'rb')) + pickle.load(open(MELS_TRN_NOISY, 'rb'))

CUR_X_FILES, CUR_X = list(df.fname.values), X_train

del df; gc.collect();
```

## Custom open\_image for fast.ai library to load data from memory

- Important note: Random cropping 1 sec, this is working like augmentation.

```
In [14]: # !!! use globals CUR_X_FILES, CUR_X
def open_fat2019_image(fn, convert_mode, after_open)->Image:
    # open
    fname = fn.split('/')[-1]
    if '!' in fname:
        fname, crop_x = fname.split('!')
        crop_x = int(crop_x)
    else:
        crop_x = -1
    idx = CUR_X_FILES.index(fname)
    x = PIL.Image.fromarray(CUR_X[idx])
    # crop
    time_dim, base_dim = x.size
    if crop_x == -1:
        crop_x = random.randint(0, time_dim - base_dim)
    x = x.crop([crop_x, 0, crop_x+base_dim, base_dim])
    # standardize
    return Image(pil2tensor(x, np.float32).div_(255))

vision.data.open_image = open_fat2019_image
```

```
In [15]: # Define image augmentation
tfms = get_transforms(
    do_flip=False, max_rotate=0, max_lighting=0.1, max_zoom=1.05, max_warp=0.,
)
```

## Follow multi-label classification

- Almost following fast.ai course: <https://nbviewer.jupyter.org/github/fastai/course-v3/blob/master/nbs/dl1/lesson3-planet.ipynb> (<https://nbviewer.jupyter.org/github/fastai/course-v3/blob/master/nbs/dl1/lesson3-planet.ipynb>)
- But pretrained=False

```
In [16]: # def borrowed_model(pretrained=False, **kwargs):
#     return Classifier(**kwargs)
```

```
In [17]: # Run "fit_one_cycle" and display some graphics
def learning(learn, cycle, lr):
    assert learn.data.classes == labels, set(labels) - set(learn.data.classes)

    learn.lr_find()
    learn.recorder.plot()
    plt.show()

    learn.fit_one_cycle(3 if TOY_MODE else cycle, lr)

    learn.recorder.plot()
    plt.show()

    learn.recorder.plot_lr()
    plt.show()

    learn.recorder.plot_losses()
    plt.show()
```

```
In [18]: class MyMixUpCallback(LearnerCallback):
    def __init__(self, learn:Learner):
        super().__init__(learn)

        # spec_augment hyperparameter
        self.masking_max_percentage=0.25

        # mixup hyperparameter
        self.alpha = .4

        # Inspiration from: https://arxiv.org/pdf/1904.08779.pdf
        # Do: Frequency masking, Frequency masking
        # Dont: Time warping
        # Bonus: Replace masking with a piece of another sample
    def _spec_augment(self, last_input, last_target):
        # Spec Augmentation
        shuffle = torch.randperm(last_target.size(0)).to(last_input.device)
        x1, y1 = last_input[shuffle], last_target[shuffle]

        batch_size, channels, height, width = last_input.size()
        h_percentage = np.random.uniform(low=0., high=self.masking_max_percentage, size=batch_size)
        w_percentage = np.random.uniform(low=0., high=self.masking_max_percentage, size=batch_size)
        alpha = (h_percentage + w_percentage) - (h_percentage * w_percentage)
        alpha = last_input.new(alpha)
        alpha = alpha.unsqueeze(1)

        new_input = last_input.clone()

        for i in range(batch_size):
            h_mask = int(h_percentage[i] * height)
            h = int(np.random.uniform(0.0, height - h_mask))
            new_input[i, :, h:h + h_mask, :] = x1[i, :, h:h + h_mask, :]

            w_mask = int(w_percentage[i] * width)
            w = int(np.random.uniform(0.0, width - w_mask))
            new_input[i, :, :, w:w + w_mask] = x1[i, :, :, w:w + w_mask]

        new_target = (1-alpha) * last_target + alpha*y1
        return new_input, new_target

    # Inspired from fastai implementation of https://arxiv.org/abs/1710.09412
    def _mixup(self, last_input, last_target):
        lambd = np.random.beta(self.alpha, self.alpha, last_target.size(0))
        lambd = np.concatenate([lambd[:,None], 1-lambd[:,None]], 1).max(1)
        lambd = last_input.new(lambd)
        shuffle = torch.randperm(last_target.size(0)).to(last_input.device)
        x1, y1 = last_input[shuffle], last_target[shuffle]
        new_input = (last_input * lambd.view(lambd.size(0),1,1,1)) + x1 * (1-lambd).view(lambd.size(0),1,1,1)
        if len(last_target.shape) == 2:
            lambd = lambd.unsqueeze(1).float()
        new_target = last_target.float() * lambd + y1.float() * (1-lambd)
        return new_input, new_target

    def on_batch_begin(self, last_input, last_target, train, **kwargs):
        if not train: return
        new_input, new_target = self._mixup(last_input, last_target)
        new_input, new_target = self._spec_augment(new_input, new_target)
        return {'last_input': new_input, 'last_target': new_target}
```

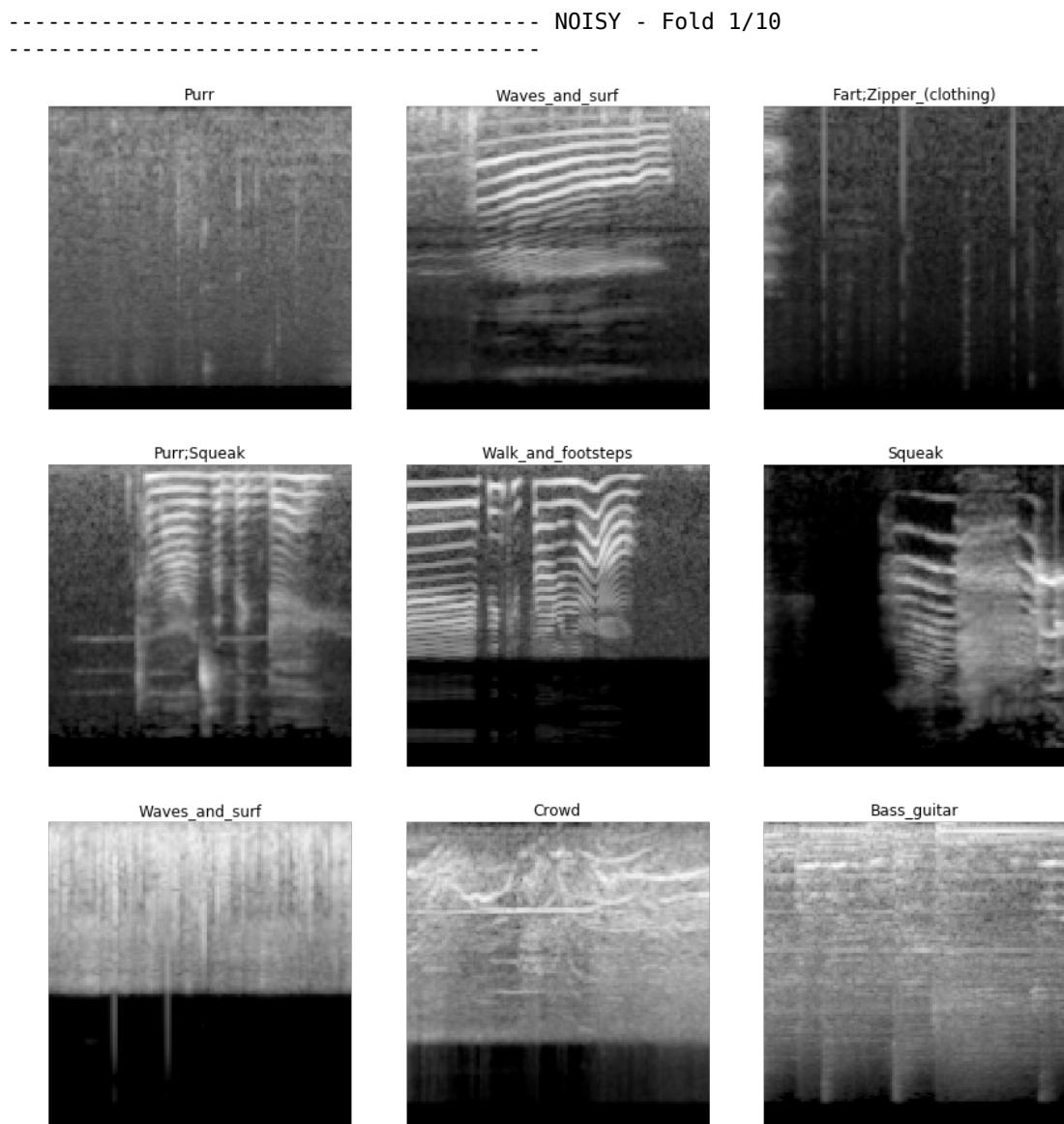
```
In [19]: labels = [
    'Accelerating_and_revving_and_vroom',
    'Accordion',
    'Acoustic_guitar',
    'Applause',
    'Bark',
    'Bass_drum',
    'Bass_guitar',
    'Bathtub_(filling_or_washing)',
    'Bicycle_bell',
    'Burping_and_eructation',
    'Bus',
    'Buzz',
    'Car_passing_by',
    'Cheering',
    'Chewing_and_mastication',
    'Child_speech_and_kidSpeaking',
    'Chink_and_clink',
    'Chirp_and_tweet',
    'Church_bell',
    'Clapping',
    'Computer_keyboard',
    'Crackle',
    'Cricket',
    'Crowd',
    'Cupboard_open_or_close',
    'Cutlery_and_silverware',
    'Dishes_and_pots_and_pans',
    'Drawer_open_or_close',
    'Drip',
    'Electric_guitar',
    'Fart',
    'Female_singing',
    'Female_speech_and_womanSpeaking',
    'Fill_(with_liquid)',
    'Finger_snapping',
    'Frying_(food)',
    'Gasp',
    'Glockenspiel',
    'Gong',
    'Gurgling',
    'Harmonica',
    'Hi-hat',
    'Hiss',
    'Keys_jangling',
    'Knock',
    'Male_singing',
    'Male_speech_and_manSpeaking',
    'Marimba_and_xylophone',
    'Mechanical_fan',
    'Meow',
    'Microwave_oven',
    'Motorcycle',
    'Printer',
    'Purr',
    'Race_car_and_auto_racing',
    'Raindrop',
    'Run',
    'Scissors',
    'Screaming',
    'Shatter',
    'Sigh',
    'Sink_(filling_or_washing)'
```

```
In [20]: kf_noisy = KFold(n_splits=n_splits, shuffle=True, random_state=67890)
```

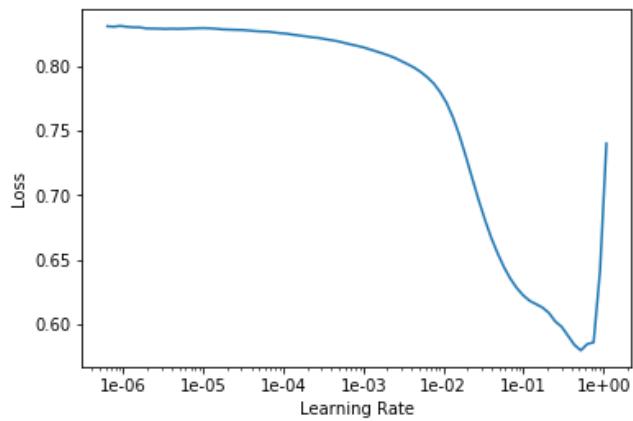
```
In [21]: kf_curated = KFold(n_splits=n_splits, shuffle=True, random_state=123)
```

## Train on noisy data

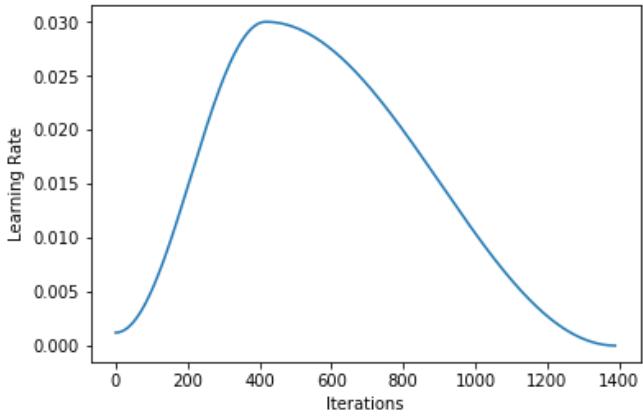
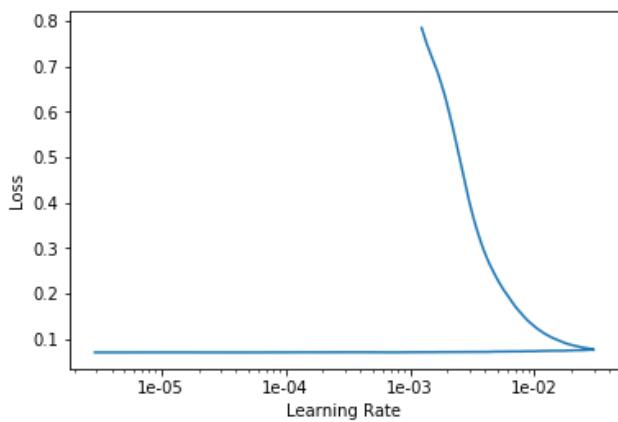
```
In [22]: for fold, (train_index, valid_index) in enumerate(kf_noisy.split(trn_noisy_d  
f)):  
    print('-' * 40, f'NOISY - Fold {fold+1}/{n_splits}', '-' * 40)  
  
    src = (ImageList.from_df(trn_noisy_df, WORK)  
           .split_by_idxs(train_index, valid_index)  
           .label_from_df(label_delim=',')  
    )  
    data = (src.transform(tfms, size=128)  
            .databunch(bs=bs)  
    )  
    data.show_batch(3)  
    plt.show()  
  
#     learn = cnn_learner(data, borrowed_model, pretrained=False, metrics=[Lwlr  
ap()])  
#         learn = cnn_learner(data, models.vgg16_bn, pretrained=False, metrics=[Lwlra  
p()])  
#         learn.callback_fns.append(MyMixUpCallback)  
#         learn.unfreeze()  
  
#     assert learn.data.classes == labels  
#     break  
learning(learn, 10, slice(1e-5, 3e-2))  
learning(learn, 100, 1e-3)  
  
learn.save(f'stage-1_fold-{fold}')  
  
if TOY_MODE:  
    break
```

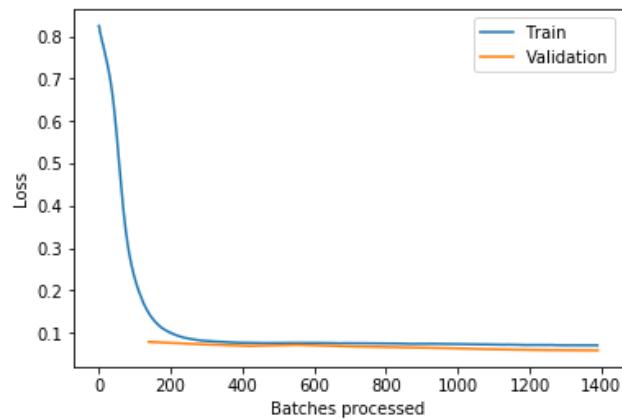


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

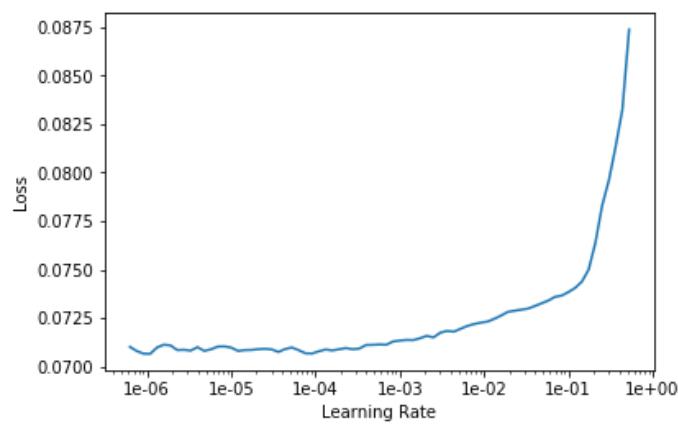


epoch	train_loss	valid_loss	lwlrap	time
0	0.148128	0.078748	0.103011	00:44
1	0.082926	0.073374	0.168196	00:43
2	0.076452	0.069277	0.224743	00:43
3	0.076551	0.071729	0.202598	00:44
4	0.075411	0.068220	0.249814	00:43
5	0.074666	0.066020	0.278212	00:44
6	0.073718	0.063887	0.324000	00:43
7	0.072488	0.061060	0.370094	00:43
8	0.071268	0.059312	0.398680	00:44
9	0.070517	0.058612	0.406515	00:43

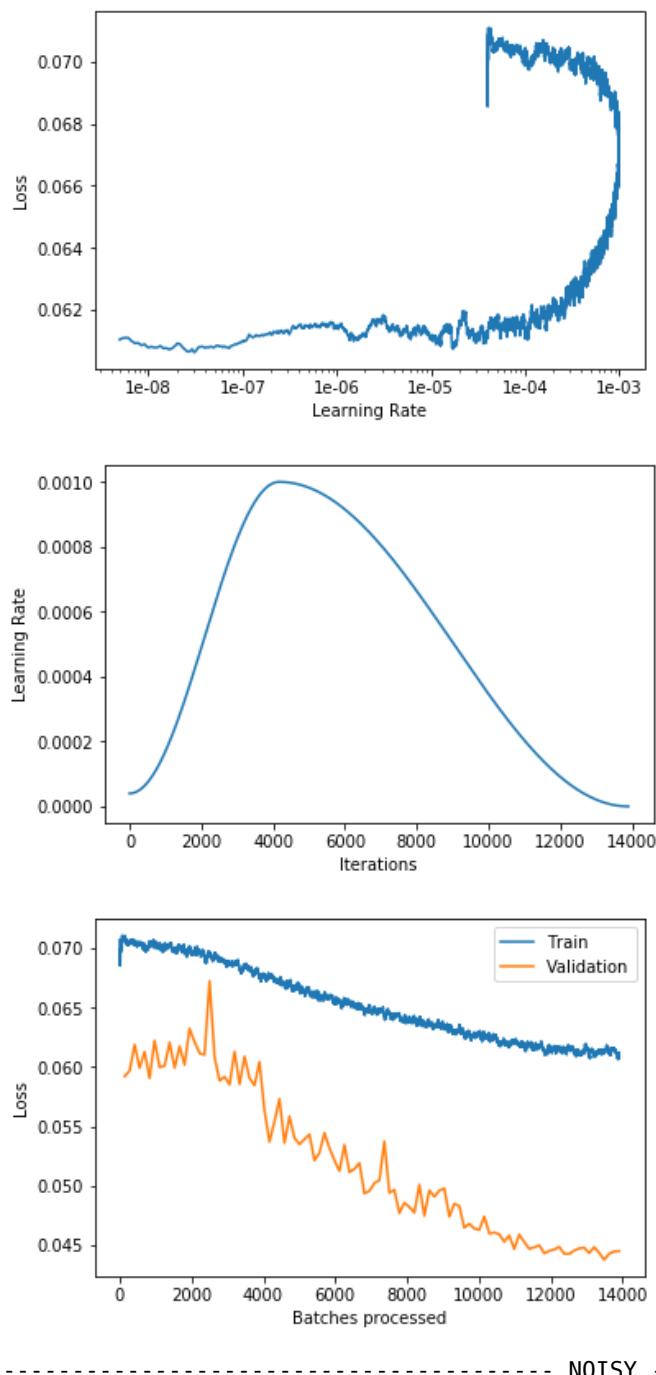




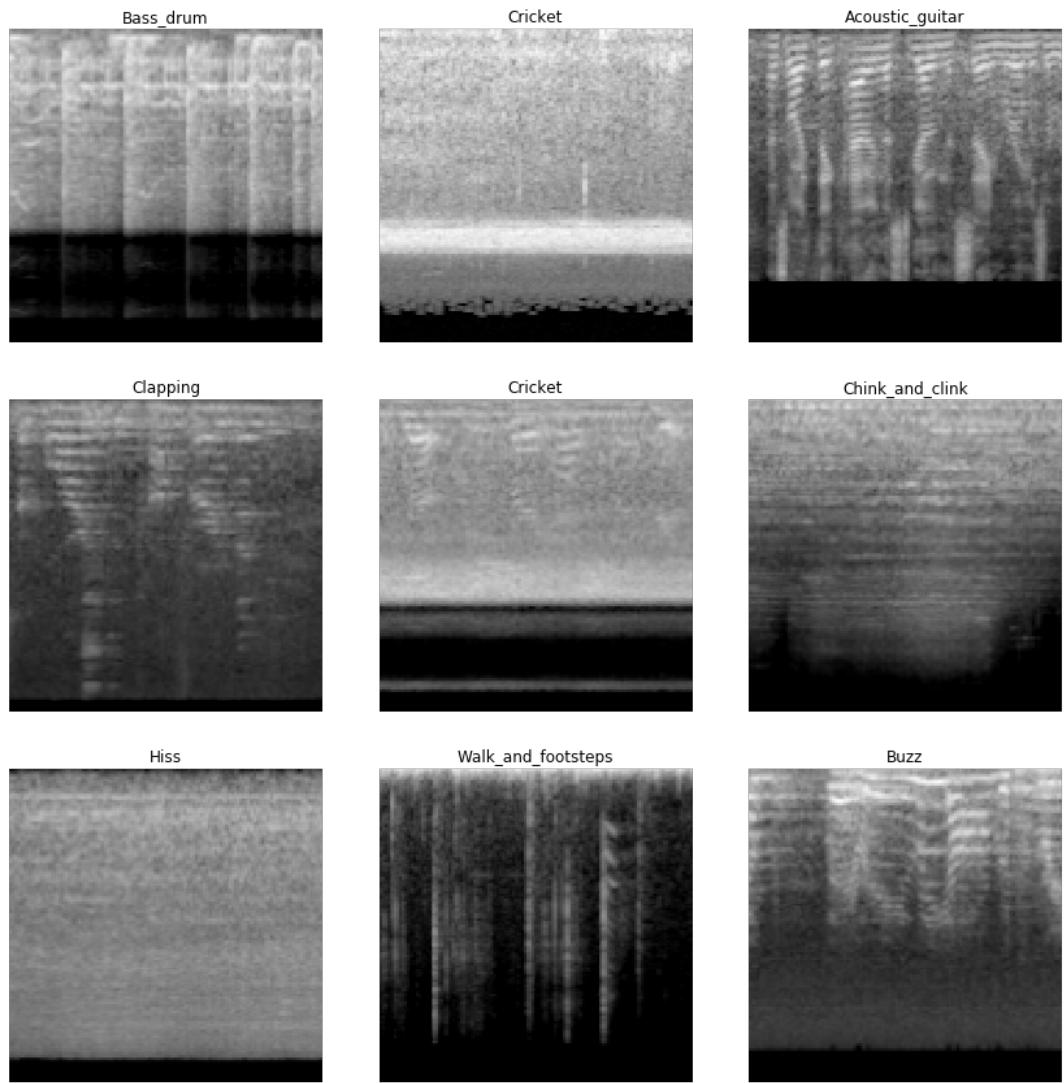
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



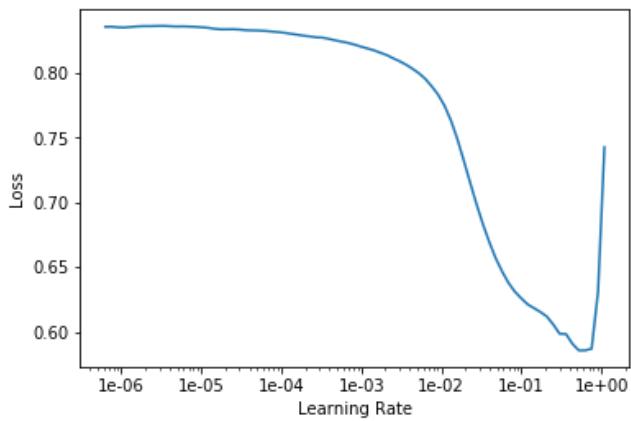
epoch	train_loss	valid_loss	lwlrap	time
0	0.071049	0.059216	0.399500	00:43
1	0.070501	0.059688	0.391669	00:44
2	0.070510	0.061886	0.351979	00:43
3	0.070389	0.059920	0.389449	00:43
4	0.070347	0.061287	0.348993	00:42
5	0.070464	0.059059	0.385445	00:42
6	0.070086	0.062235	0.338238	00:42
7	0.069978	0.059972	0.375585	00:42
8	0.070262	0.060109	0.368324	00:42
9	0.070247	0.062080	0.344630	00:42
10	0.069839	0.059923	0.374843	00:42
11	0.070184	0.061742	0.354508	00:42
12	0.069880	0.060183	0.382723	00:42
13	0.070093	0.063241	0.335163	00:42
14	0.069848	0.062105	0.336509	00:42
15	0.069692	0.061169	0.364875	00:42
16	0.069381	0.061027	0.354314	00:42
17	0.069514	0.067251	0.269632	00:42
18	0.069617	0.060870	0.362559	00:42
19	0.069161	0.058845	0.392601	00:42
20	0.068783	0.059187	0.392077	00:42
21	0.068720	0.058517	0.400924	00:42
22	0.068651	0.061301	0.357434	00:42
23	0.068200	0.058554	0.384336	00:42
24	0.068624	0.060892	0.350458	00:42
25	0.068222	0.059061	0.385690	00:42
26	0.068097	0.058458	0.412248	00:42
27	0.067663	0.060436	0.363688	00:42
28	0.067830	0.056324	0.436781	00:42
29	0.067670	0.053689	0.476624	00:42
30	0.067445	0.055432	0.445960	00:42
31	0.067164	0.057320	0.412541	00:42
32	0.067068	0.053593	0.492184	00:42
33	0.066914	0.055837	0.442829	00:42
34	0.066852	0.054031	0.475349	00:42
35	0.066151	0.053191	0.477451	00:42



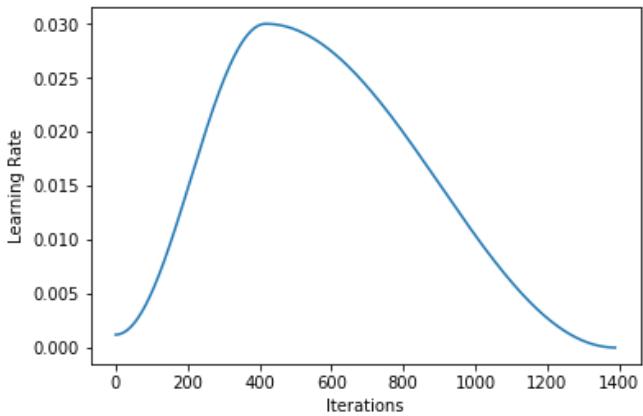
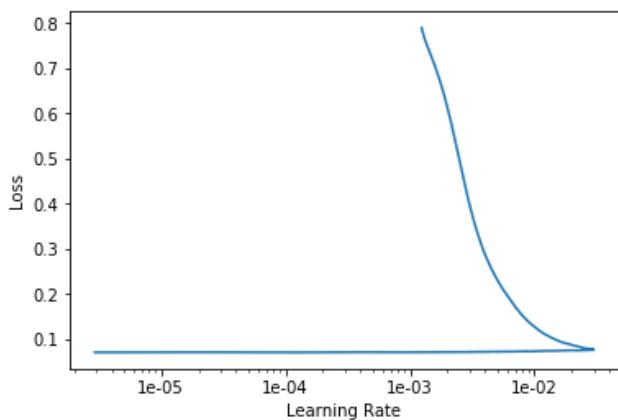
-- NOISY - Fold 2/10

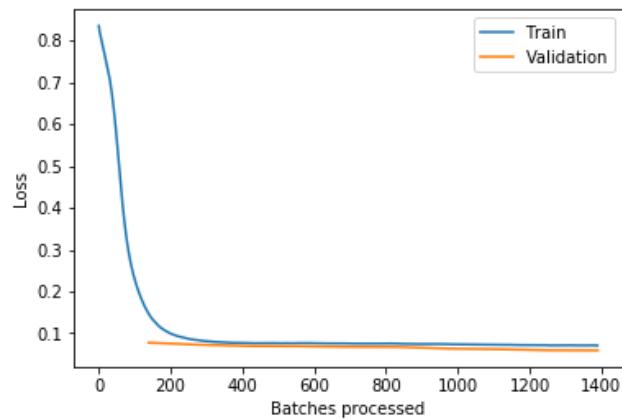


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

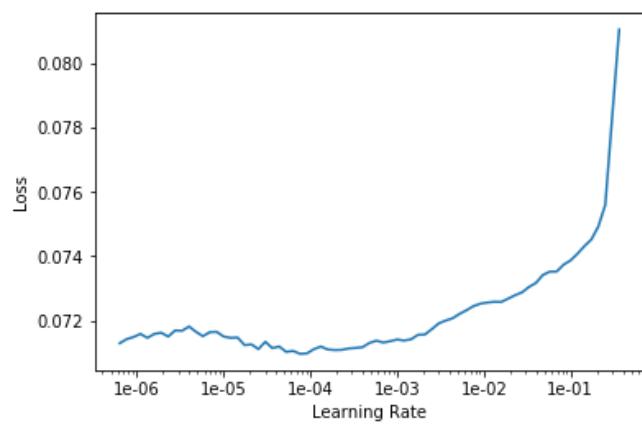


epoch	train_loss	valid_loss	lwlrap	time
0	0.147783	0.077391	0.120547	00:42
1	0.083220	0.072447	0.182533	00:42
2	0.076688	0.069553	0.223122	00:42
3	0.076203	0.068898	0.232062	00:42
4	0.075539	0.067470	0.250977	00:42
5	0.075102	0.067181	0.258895	00:42
6	0.073968	0.063321	0.331042	00:42
7	0.072506	0.061822	0.351818	00:42
8	0.071307	0.059215	0.386638	00:42
9	0.070682	0.058917	0.394058	00:42

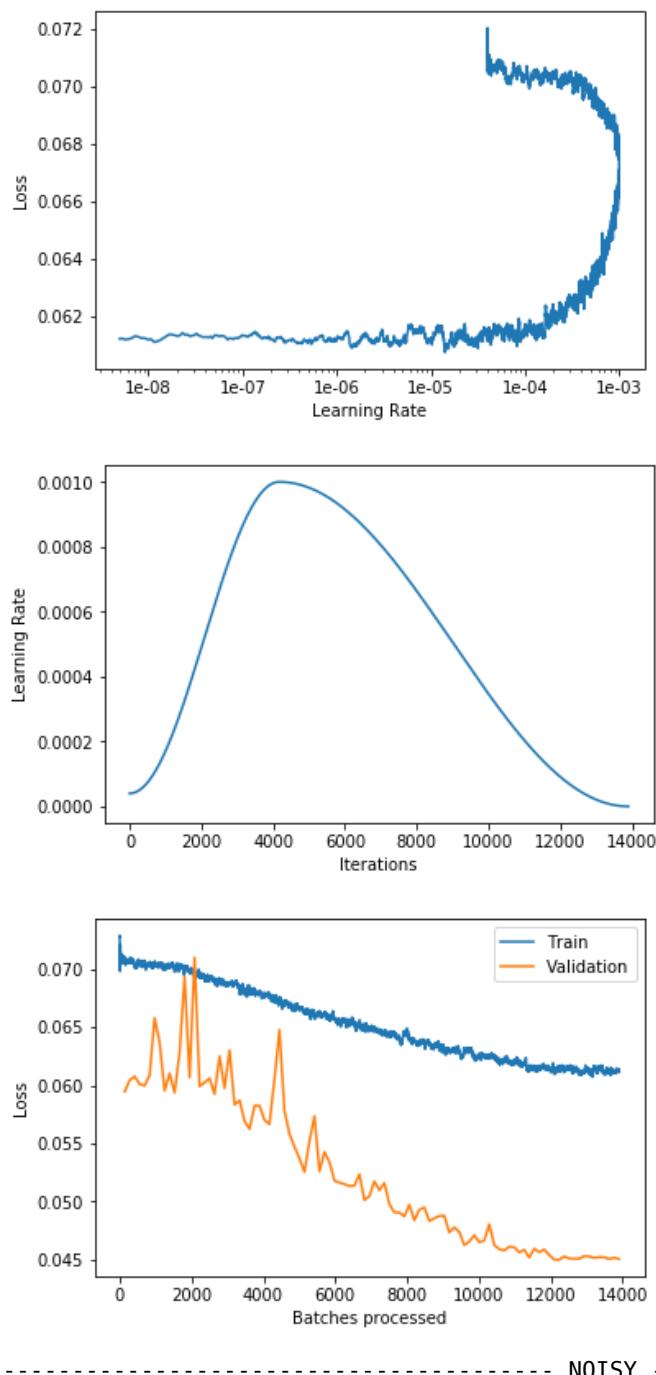


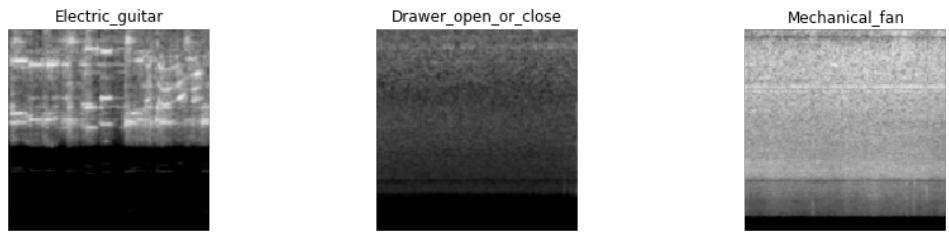


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

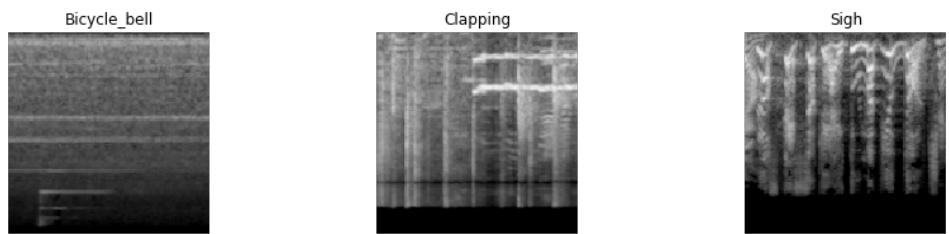
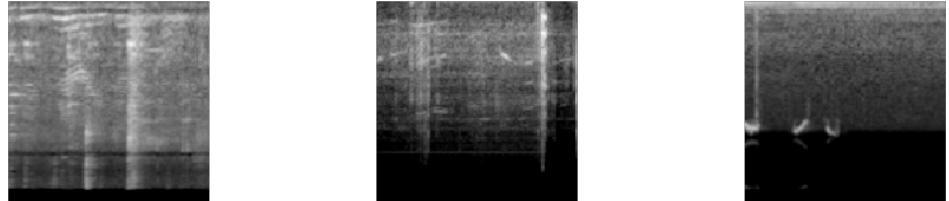


epoch	train_loss	valid_loss	lwlrap	time
0	0.070876	0.059459	0.371469	00:42
1	0.070602	0.060441	0.365991	00:42
2	0.070115	0.060780	0.367069	00:42
3	0.070367	0.060097	0.365776	00:42
4	0.070678	0.059978	0.381350	00:42
5	0.070461	0.060876	0.347206	00:42
6	0.070546	0.065797	0.274200	00:42
7	0.070414	0.063691	0.312276	00:42
8	0.070196	0.059533	0.383456	00:42
9	0.070245	0.061034	0.357311	00:42
10	0.070093	0.059373	0.381966	00:42
11	0.070189	0.062821	0.324970	00:42
12	0.069648	0.069361	0.259044	00:42
13	0.069920	0.060685	0.364735	00:42
14	0.069425	0.070974	0.219474	00:42
15	0.069493	0.059915	0.369788	00:42
16	0.069185	0.060239	0.367830	00:42
17	0.069197	0.060620	0.369582	00:42
18	0.068844	0.059248	0.388126	00:42
19	0.069054	0.062500	0.328715	00:42
20	0.068896	0.059755	0.366559	00:42
21	0.068828	0.063013	0.326688	00:42
22	0.068834	0.058337	0.396358	00:42
23	0.068140	0.058709	0.387317	00:42
24	0.068021	0.056929	0.419206	00:42
25	0.068291	0.056245	0.425128	00:42
26	0.067975	0.058244	0.400814	00:42
27	0.068005	0.058257	0.401928	00:42
28	0.067826	0.057023	0.415991	00:42
29	0.067595	0.056661	0.432297	00:42
30	0.067189	0.060541	0.370934	00:42
31	0.067059	0.064776	0.311421	00:42
32	0.067175	0.057807	0.399603	00:42
33	0.066877	0.055822	0.431481	00:42
34	0.066643	0.054713	0.448211	00:42
35	0.066416	0.053715	0.461191	00:42

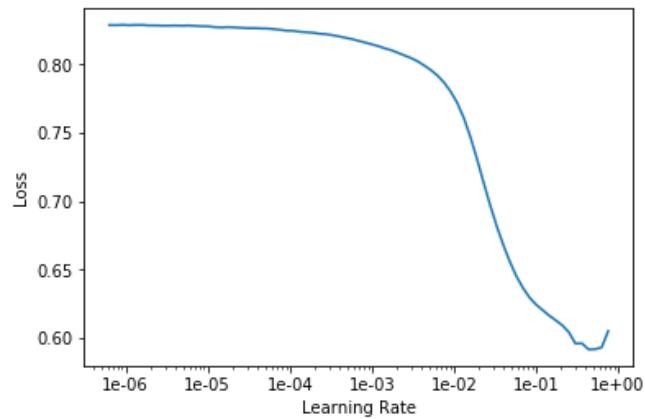




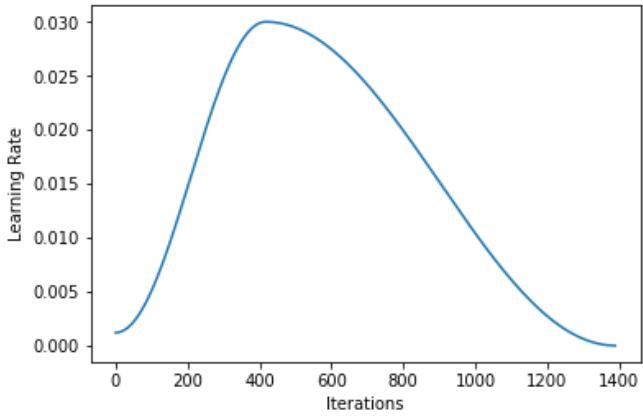
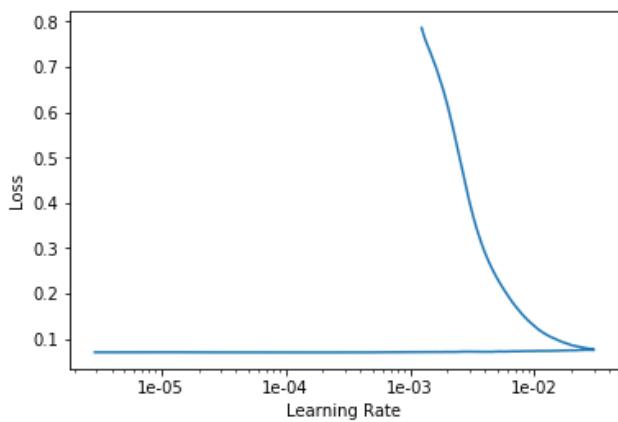
Cutlery\_and\_silverware;Dishes\_and\_pots\_and\_pans;Frying\_(food);Eating\_and\_mastication;Skateboard

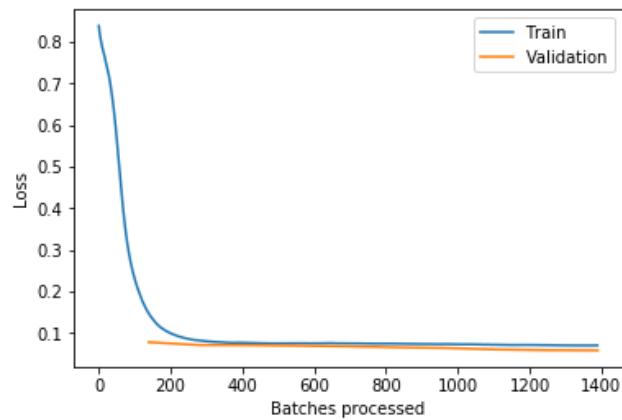


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

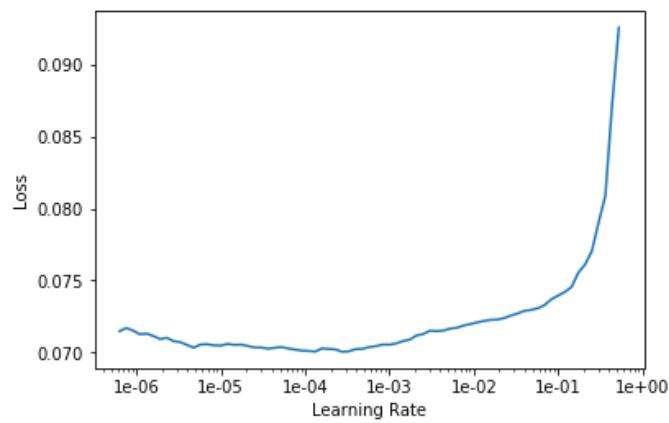


epoch	train_loss	valid_loss	lwlrap	time
0	0.149126	0.078414	0.102357	00:42
1	0.083061	0.071654	0.193986	00:42
2	0.076529	0.071440	0.199304	00:42
3	0.076248	0.070073	0.225322	00:42
4	0.075748	0.068472	0.239523	00:42
5	0.074753	0.066171	0.273265	00:42
6	0.073916	0.064352	0.306267	00:42
7	0.072247	0.060977	0.362813	00:42
8	0.071416	0.059287	0.389484	00:42
9	0.070897	0.058743	0.404796	00:42

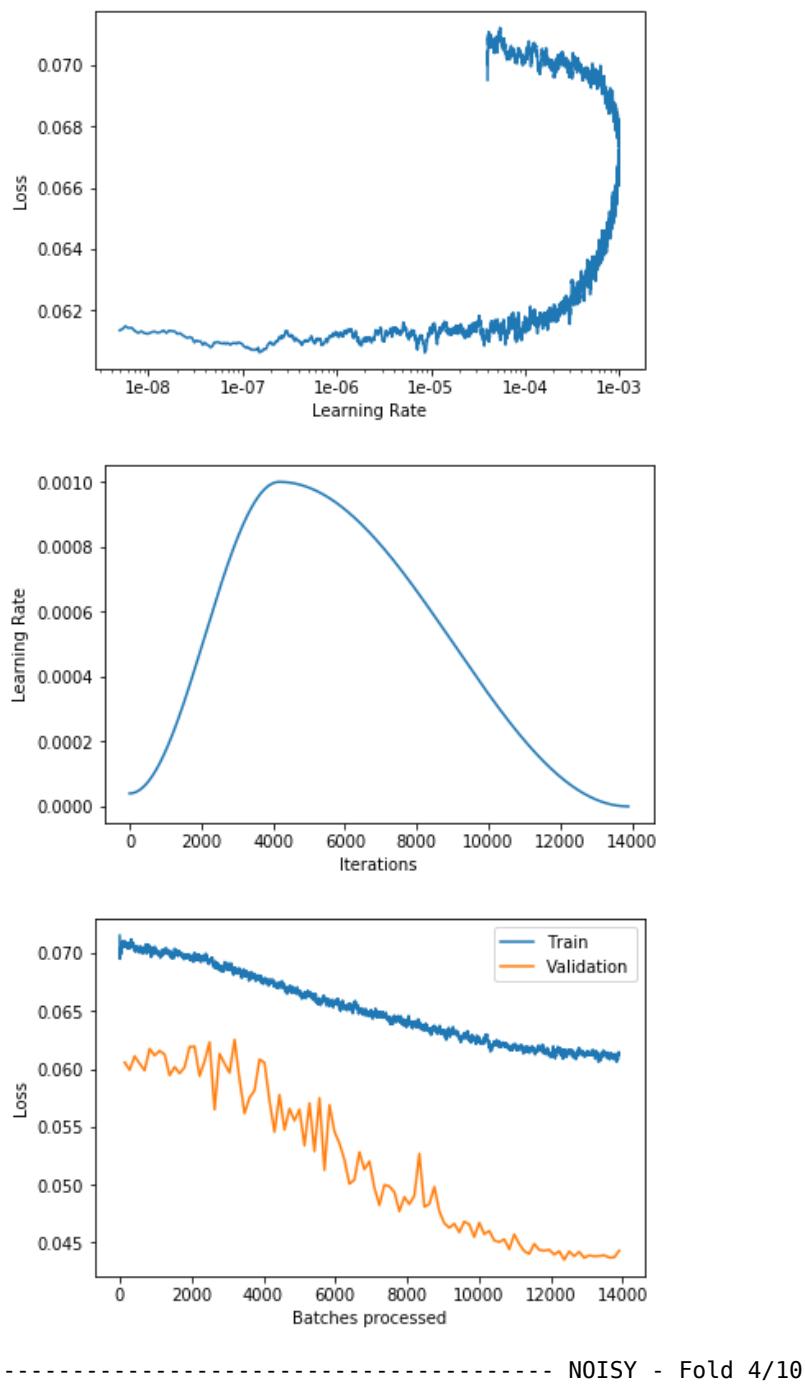


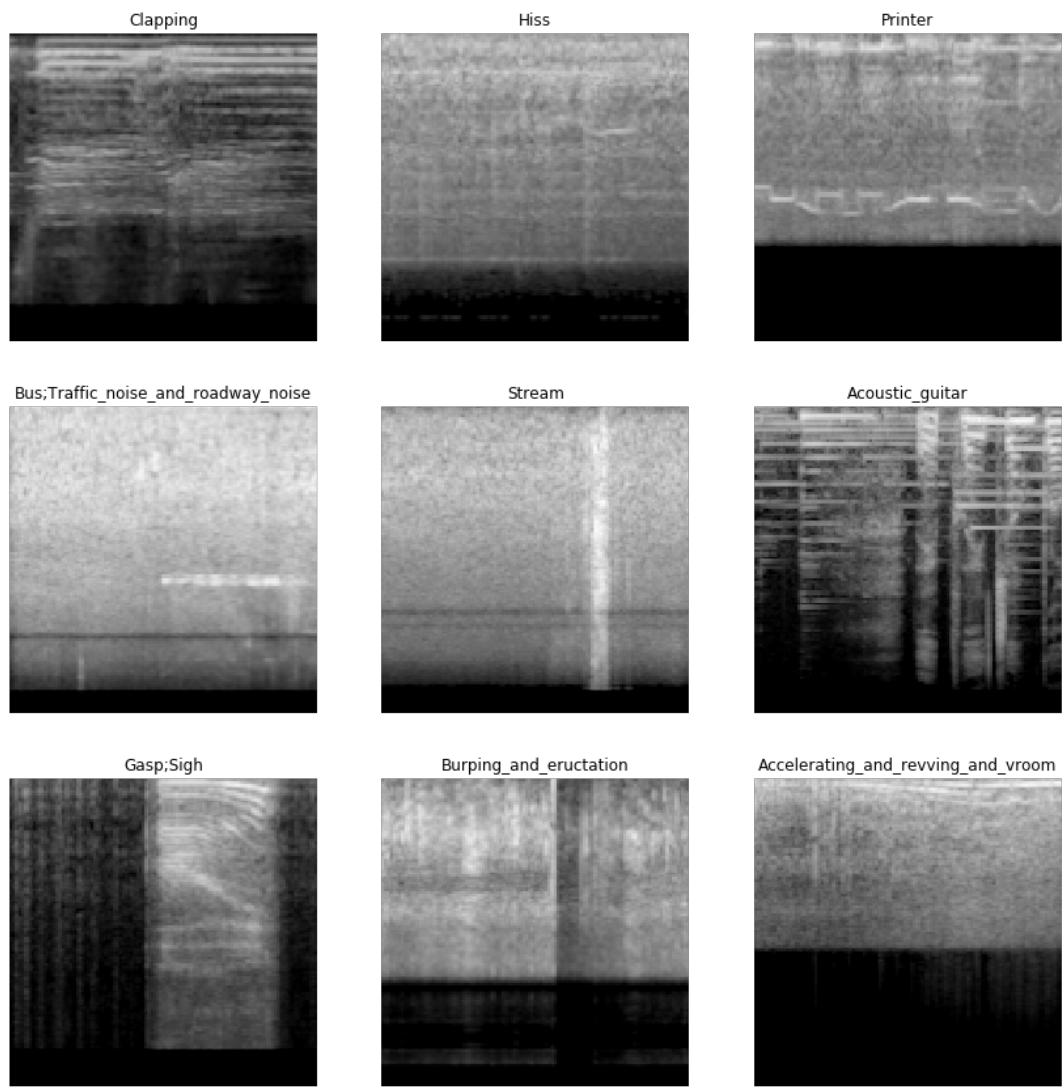


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

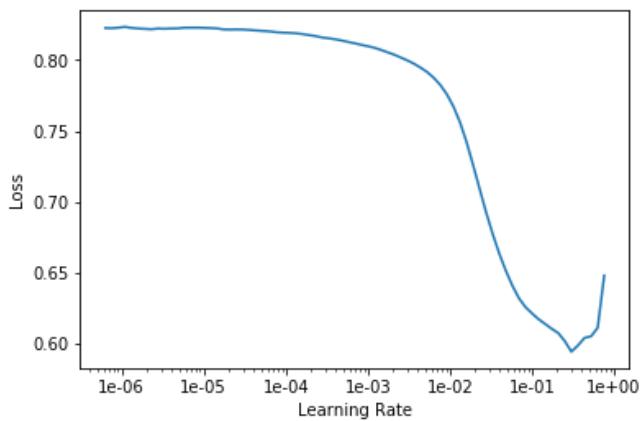


epoch	train_loss	valid_loss	lwlrap	time
0	0.070874	0.060580	0.361667	00:42
1	0.070524	0.059929	0.394165	00:42
2	0.070253	0.061115	0.357654	00:42
3	0.070255	0.060474	0.376407	00:42
4	0.070528	0.059887	0.386724	00:42
5	0.070405	0.061763	0.347908	00:42
6	0.070201	0.061181	0.368456	00:42
7	0.069995	0.061578	0.364539	00:42
8	0.070189	0.061270	0.361442	00:42
9	0.069902	0.059451	0.386670	00:42
10	0.070371	0.060197	0.388887	00:42
11	0.069870	0.059634	0.393511	00:42
12	0.069902	0.060163	0.383622	00:42
13	0.069720	0.061920	0.346482	00:42
14	0.070066	0.061958	0.352400	00:42
15	0.069678	0.059406	0.393746	00:42
16	0.069515	0.060682	0.364878	00:42
17	0.069190	0.062332	0.334932	00:43
18	0.068783	0.056500	0.432858	00:42
19	0.068893	0.061325	0.367833	00:42
20	0.069135	0.060481	0.369221	00:42
21	0.068625	0.059681	0.402135	00:42
22	0.068517	0.062546	0.343693	00:42
23	0.068519	0.059111	0.390755	00:42
24	0.068277	0.056180	0.439194	00:42
25	0.067959	0.057531	0.420470	00:42
26	0.067975	0.058154	0.416848	00:42
27	0.067863	0.060845	0.377558	00:42
28	0.067654	0.060554	0.369405	00:42
29	0.067649	0.057268	0.421311	00:42
30	0.067283	0.054569	0.454868	00:42
31	0.067428	0.057788	0.414267	00:42
32	0.067191	0.054758	0.461145	00:42
33	0.067032	0.056577	0.435080	00:42
34	0.066659	0.055551	0.441448	00:42
35	0.066188	0.056508	0.429065	00:42

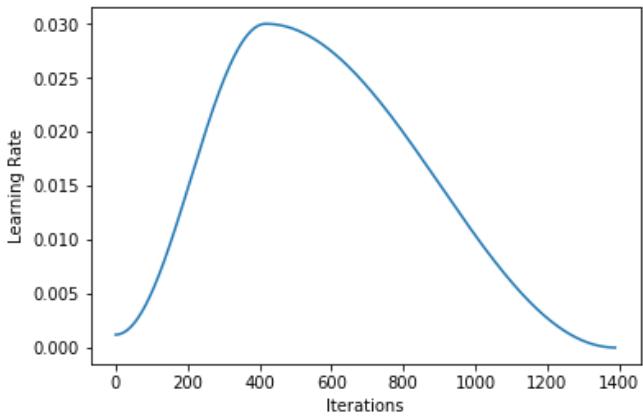
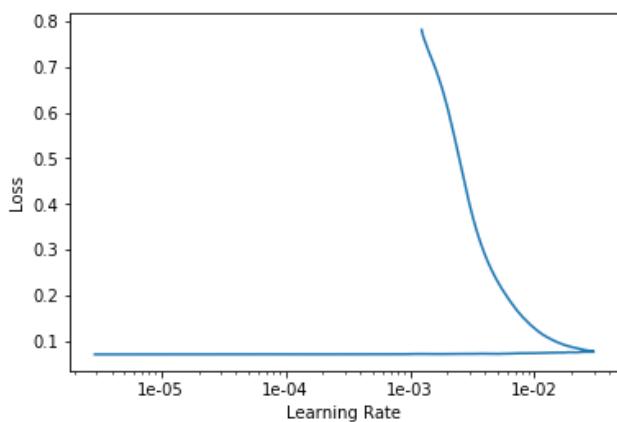


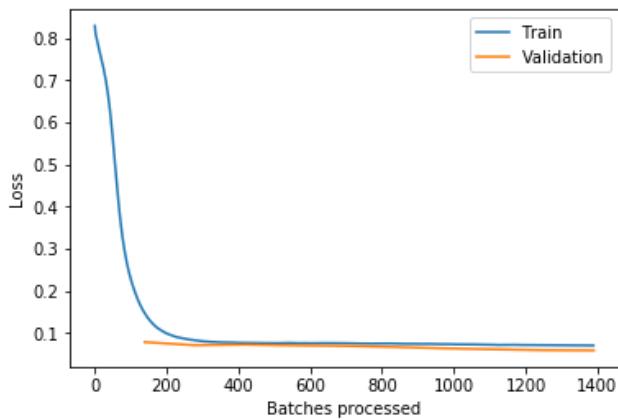


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

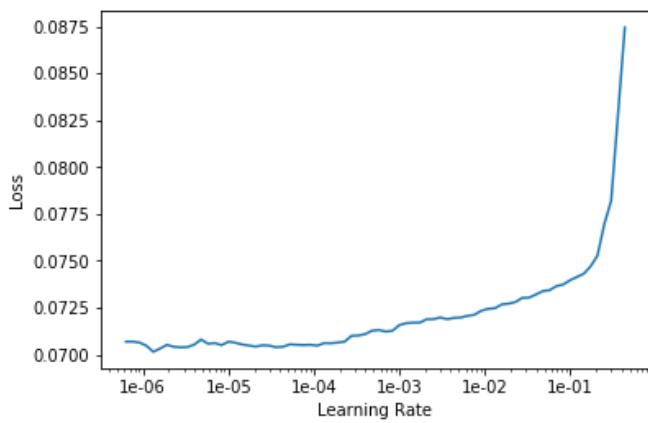


epoch	train_loss	valid_loss	lwlrap	time
0	0.148341	0.078540	0.115747	00:42
1	0.083288	0.071285	0.188557	00:43
2	0.076572	0.072606	0.183949	00:43
3	0.076205	0.070806	0.213178	00:42
4	0.076142	0.069677	0.235775	00:42
5	0.075179	0.067014	0.277744	00:42
6	0.073708	0.063847	0.310712	00:42
7	0.072211	0.061541	0.351736	00:42
8	0.071339	0.059417	0.387625	00:42
9	0.070667	0.058912	0.394203	00:42

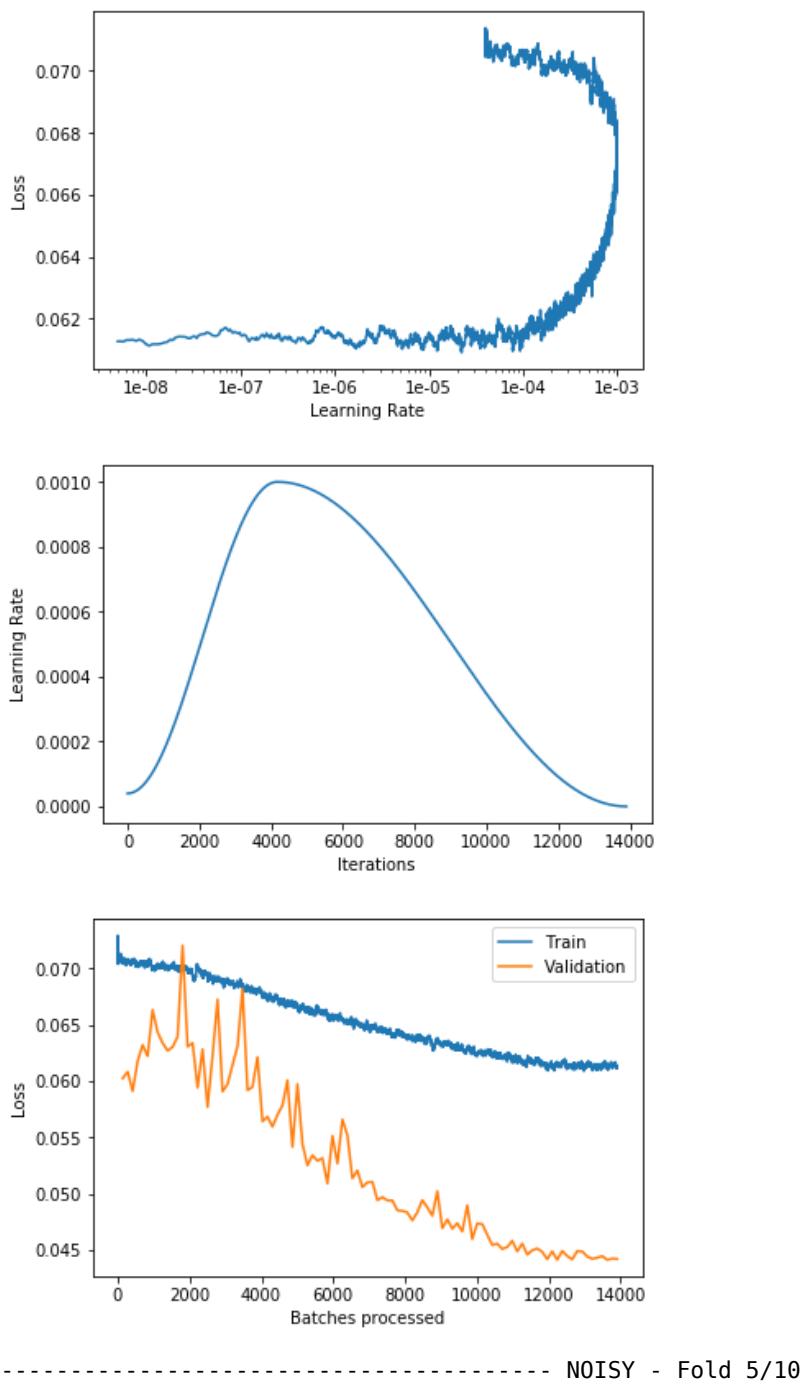


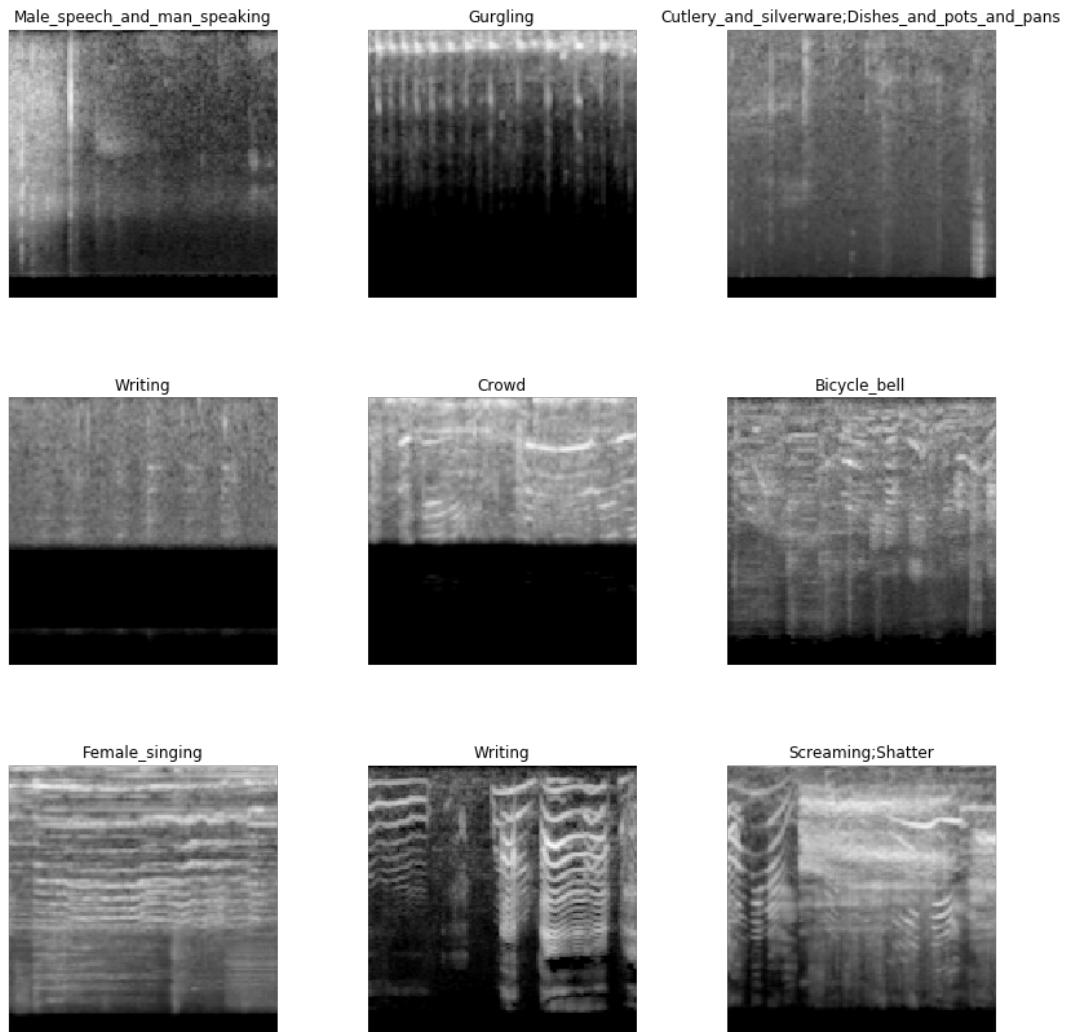


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

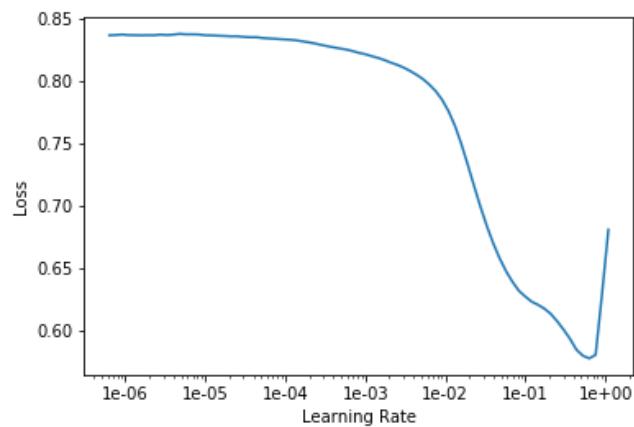


epoch	train_loss	valid_loss	lwlrap	time
0	0.070763	0.060266	0.368253	00:43
1	0.070736	0.060851	0.361829	00:43
2	0.070500	0.059133	0.392227	00:42
3	0.070280	0.061804	0.349631	00:43
4	0.070279	0.063231	0.327380	00:42
5	0.070491	0.062241	0.349068	00:43
6	0.070043	0.066343	0.279507	00:43
7	0.070478	0.064367	0.302730	00:42
8	0.070498	0.063411	0.333621	00:43
9	0.070155	0.062727	0.335953	00:43
10	0.070219	0.063048	0.338541	00:42
11	0.069936	0.063971	0.309910	00:42
12	0.070163	0.072062	0.247996	00:43
13	0.070028	0.063079	0.332955	00:43
14	0.069505	0.063402	0.331016	00:43
15	0.070145	0.059444	0.385839	00:43
16	0.069345	0.062855	0.336077	00:43
17	0.069203	0.057715	0.426729	00:43
18	0.069216	0.062261	0.347500	00:43
19	0.069245	0.067278	0.271110	00:42
20	0.069033	0.059098	0.387984	00:43
21	0.069155	0.059742	0.388437	00:42
22	0.068648	0.061458	0.362851	00:43
23	0.068759	0.063118	0.335576	00:42
24	0.068389	0.068238	0.254179	00:43
25	0.068269	0.059211	0.390115	00:43
26	0.068058	0.059510	0.396263	00:43
27	0.068131	0.062160	0.439257	00:43
28	0.067539	0.056440	0.428286	00:43
29	0.067276	0.056862	0.435529	00:45
30	0.067380	0.055974	0.438348	00:43
31	0.067241	0.057019	0.428377	00:42
32	0.066836	0.057867	0.403100	00:42
33	0.067010	0.060094	0.367525	00:42
34	0.066903	0.054180	0.461599	00:42
35	0.066808	0.050738	0.390461	00:42

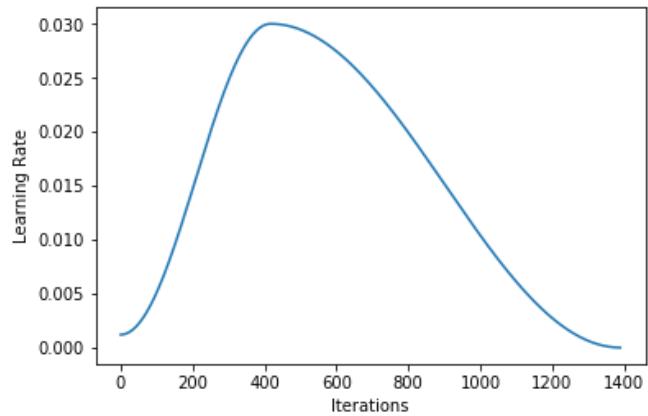
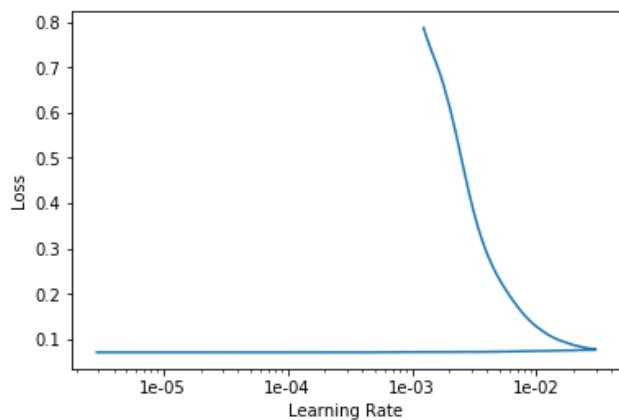


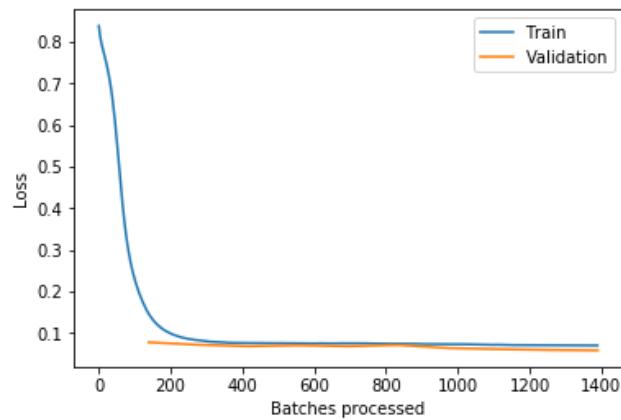


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

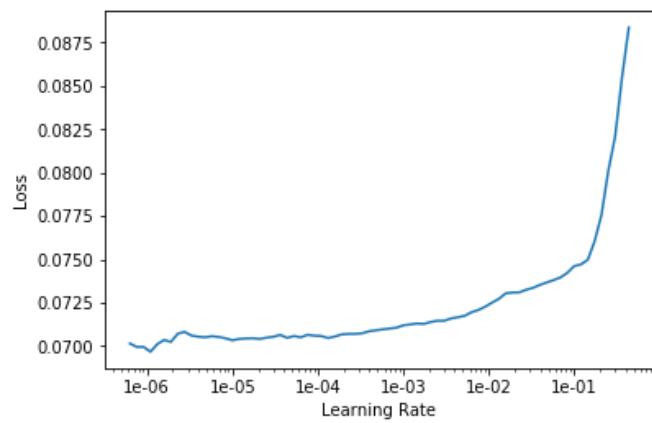


epoch	train_loss	valid_loss	lwlrap	time
0	0.148026	0.078105	0.118938	00:42
1	0.083103	0.072263	0.182812	00:42
2	0.076407	0.068909	0.242997	00:42
3	0.075931	0.070927	0.220612	00:42
4	0.075409	0.068812	0.248228	00:42
5	0.074782	0.071751	0.220190	00:42
6	0.073717	0.064361	0.325921	00:42
7	0.072332	0.061749	0.357944	00:42
8	0.071234	0.059880	0.380404	00:42
9	0.070672	0.058732	0.400363	00:42

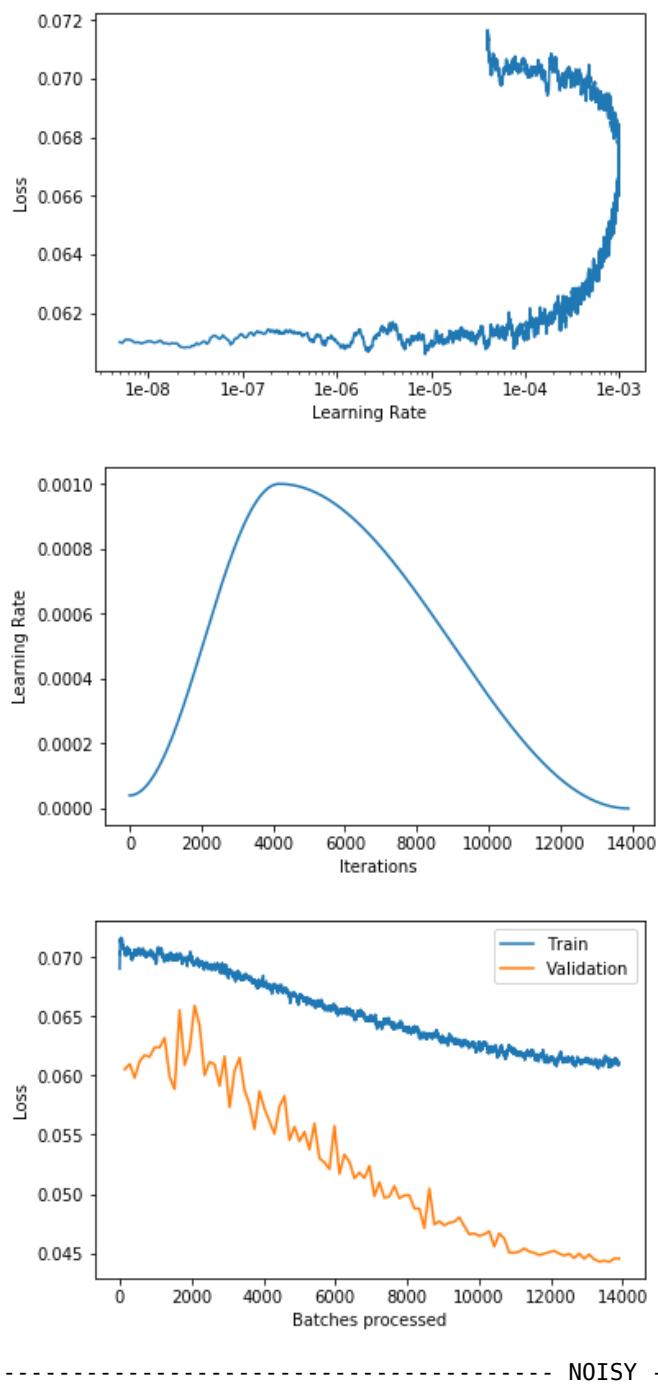




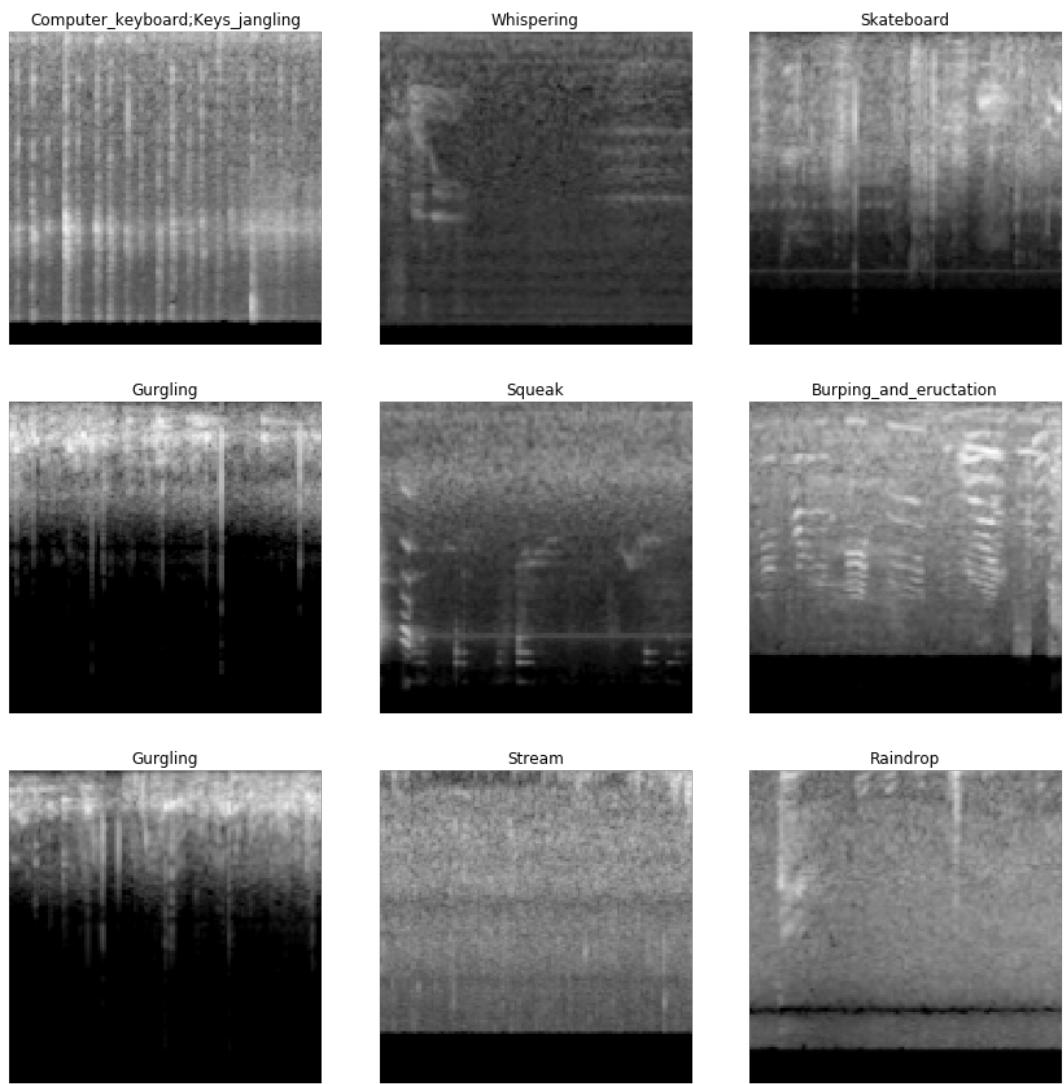
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



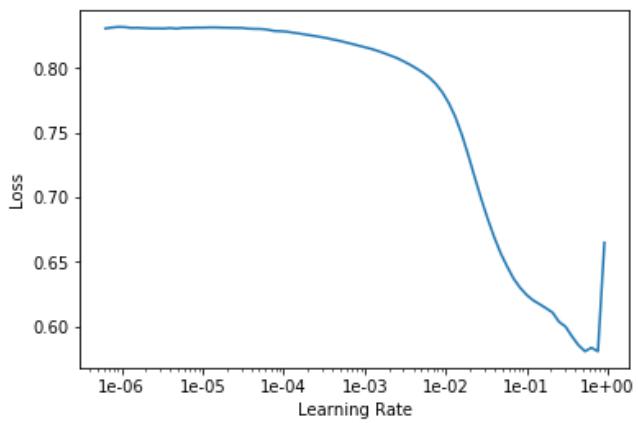
epoch	train_loss	valid_loss	lwlrap	time
0	0.070420	0.060524	0.379553	00:42
1	0.070512	0.060977	0.368481	00:42
2	0.070678	0.059810	0.385213	00:42
3	0.070385	0.061213	0.379882	00:42
4	0.070276	0.061729	0.352152	00:42
5	0.070323	0.061591	0.353113	00:42
6	0.069990	0.062389	0.350947	00:42
7	0.070376	0.062374	0.345067	00:42
8	0.069774	0.063190	0.338780	00:42
9	0.070195	0.059899	0.394256	00:42
10	0.070147	0.058863	0.391491	00:42
11	0.070006	0.065534	0.299404	00:42
12	0.069842	0.060898	0.373797	00:42
13	0.069888	0.062201	0.345276	00:42
14	0.069367	0.065911	0.300270	00:42
15	0.069349	0.064264	0.314757	00:42
16	0.069074	0.060040	0.392166	00:42
17	0.069144	0.061144	0.366851	00:42
18	0.069294	0.060961	0.362486	00:42
19	0.069134	0.059129	0.397452	00:42
20	0.069095	0.061632	0.367681	00:42
21	0.068867	0.057334	0.420280	00:42
22	0.068623	0.060385	0.376229	00:42
23	0.068392	0.061510	0.349452	00:42
24	0.068429	0.058769	0.404076	00:42
25	0.068230	0.057552	0.419447	00:42
26	0.068063	0.055478	0.455790	00:42
27	0.067823	0.058672	0.416027	00:42
28	0.067555	0.057194	0.426676	00:42
29	0.067642	0.056135	0.430970	00:42
30	0.067653	0.055075	0.461613	00:42
31	0.067564	0.057393	0.423281	00:42
32	0.067095	0.058273	0.409078	00:42
33	0.066757	0.054560	0.467973	00:42
34	0.066831	0.055671	0.458112	00:42
35	0.066793	0.054459	0.466936	00:42



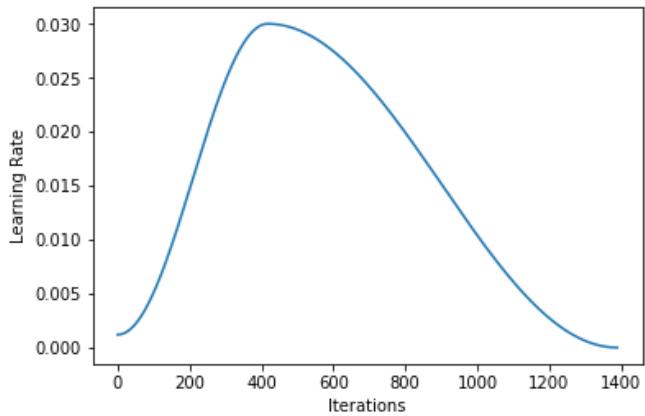
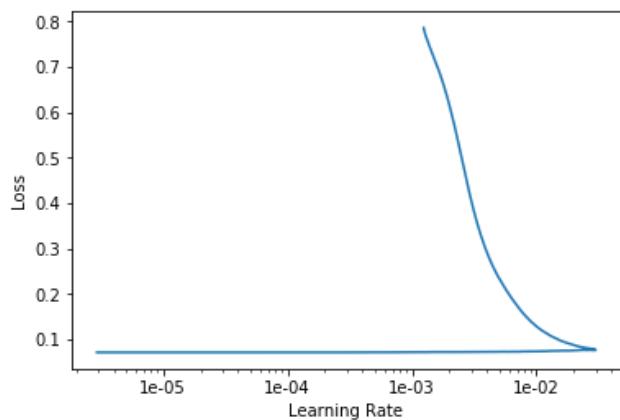
-- NOISY - Fold 6/10

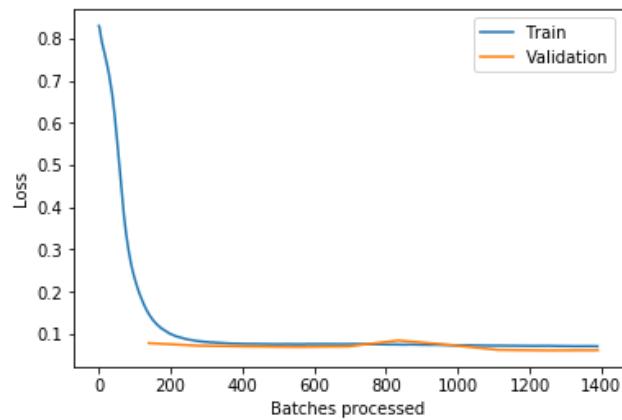


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

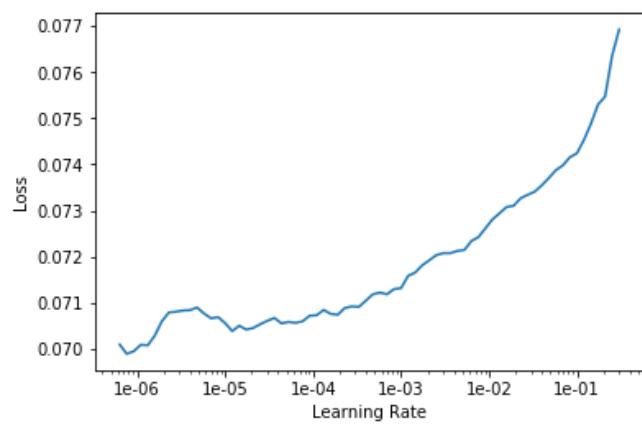


epoch	train_loss	valid_loss	lwlrap	time
0	0.149136	0.078318	0.106236	00:43
1	0.083100	0.072342	0.184487	00:42
2	0.076589	0.070576	0.226010	00:42
3	0.075842	0.069413	0.229210	00:42
4	0.076280	0.070643	0.226593	00:42
5	0.074930	0.084681	0.253418	00:42
6	0.073713	0.074577	0.297143	00:42
7	0.072741	0.062447	0.341560	00:42
8	0.071934	0.061142	0.362052	00:42
9	0.070965	0.061506	0.376394	00:42

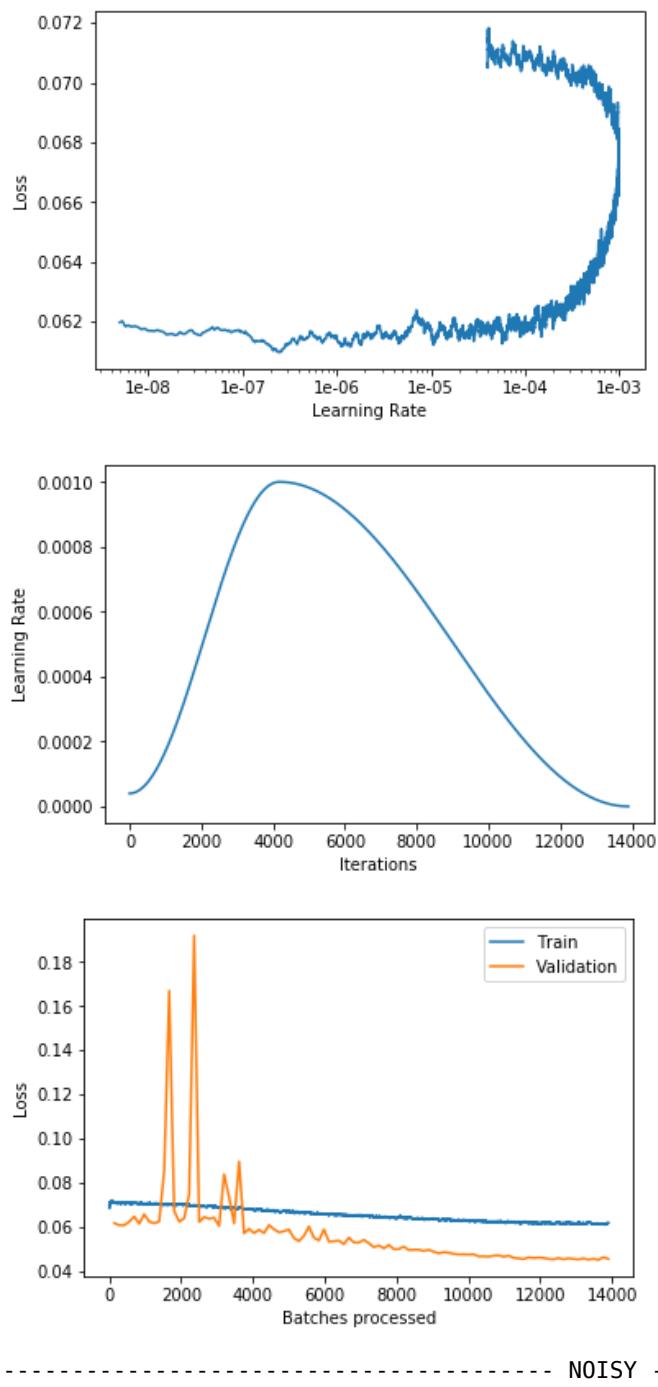




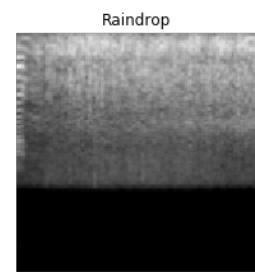
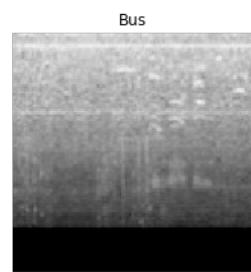
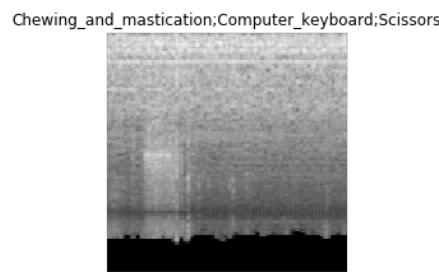
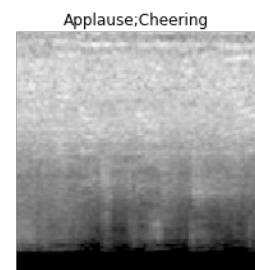
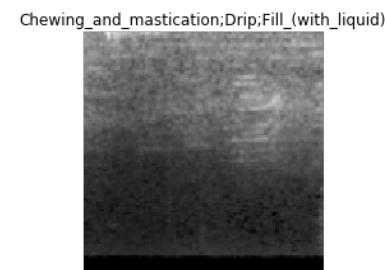
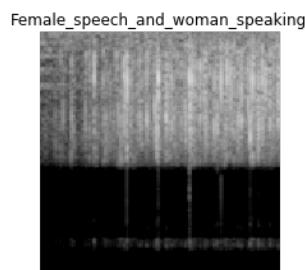
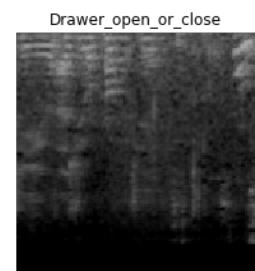
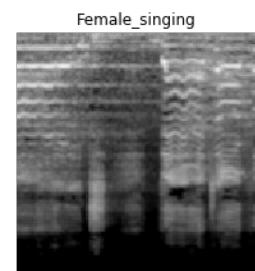
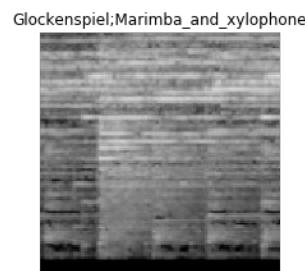
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



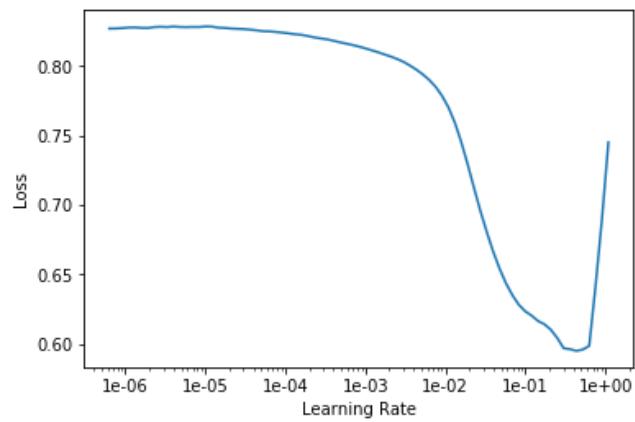
epoch	train_loss	valid_loss	lwlrap	time
0	0.071115	0.061795	0.368953	00:42
1	0.070919	0.060841	0.376133	00:42
2	0.070792	0.060823	0.368350	00:42
3	0.070763	0.062461	0.342814	00:42
4	0.070804	0.064744	0.327239	00:42
5	0.070713	0.061510	0.372073	00:42
6	0.070384	0.065794	0.279799	00:42
7	0.070661	0.062423	0.350800	00:42
8	0.070691	0.061755	0.355391	00:42
9	0.070177	0.062529	0.347566	00:42
10	0.070280	0.085550	0.326765	00:42
11	0.070394	0.166921	0.345761	00:42
12	0.070478	0.067223	0.364202	00:42
13	0.070225	0.062459	0.351680	00:42
14	0.070048	0.063988	0.324474	00:42
15	0.069912	0.074813	0.327527	00:42
16	0.069967	0.191870	0.365767	00:42
17	0.069816	0.062341	0.354223	00:42
18	0.069440	0.064535	0.311165	00:42
19	0.069571	0.063771	0.335954	00:42
20	0.069285	0.064265	0.320188	00:42
21	0.069009	0.060514	0.373329	00:42
22	0.068677	0.083848	0.381189	00:42
23	0.068737	0.073448	0.282491	00:42
24	0.068698	0.061668	0.397528	00:42
25	0.068760	0.089658	0.383596	00:42
26	0.068306	0.057144	0.431629	00:42
27	0.068221	0.059179	0.393152	00:42
28	0.067878	0.057263	0.425414	00:42
29	0.067940	0.058781	0.406713	00:42
30	0.067560	0.057265	0.427530	00:42
31	0.067623	0.060922	0.395096	00:42
32	0.067199	0.059012	0.403619	00:42
33	0.067039	0.057605	0.407634	00:42
34	0.067447	0.058136	0.409500	00:42
35	0.067080	0.058850	0.402507	00:42



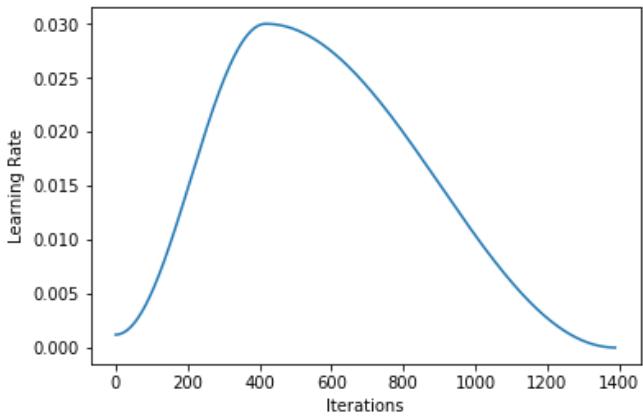
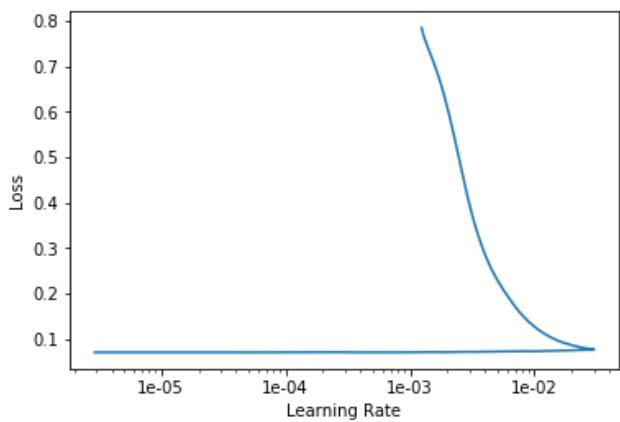
- - - NOISY - Fold 7/10

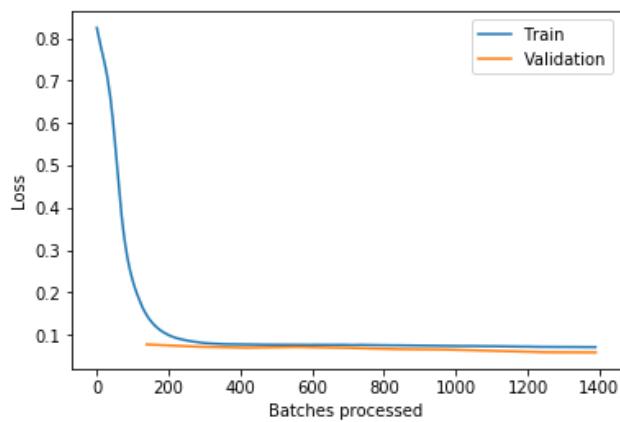


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

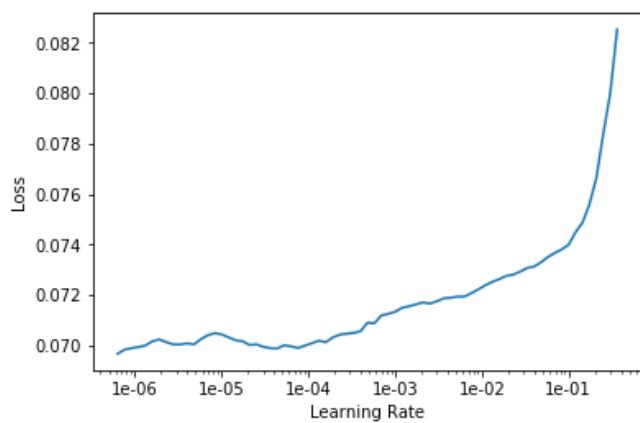


epoch	train_loss	valid_loss	lwlrap	time
0	0.147762	0.077152	0.101257	00:42
1	0.082898	0.071878	0.191941	00:42
2	0.076975	0.069592	0.209513	00:42
3	0.076313	0.071201	0.212902	00:42
4	0.076074	0.069200	0.221094	00:42
5	0.075088	0.066151	0.280039	00:42
6	0.073683	0.065071	0.279434	00:42
7	0.072745	0.061866	0.327641	00:42
8	0.071500	0.058750	0.386891	00:42
9	0.070864	0.058322	0.389676	00:42

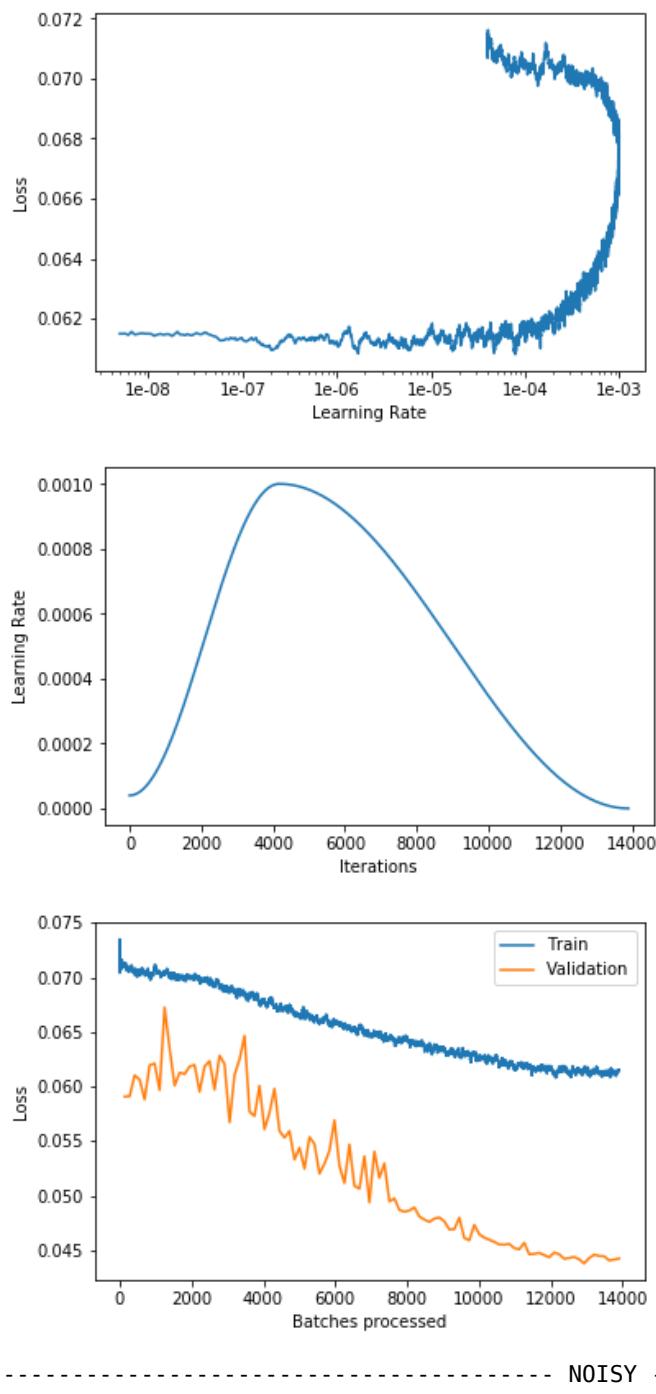




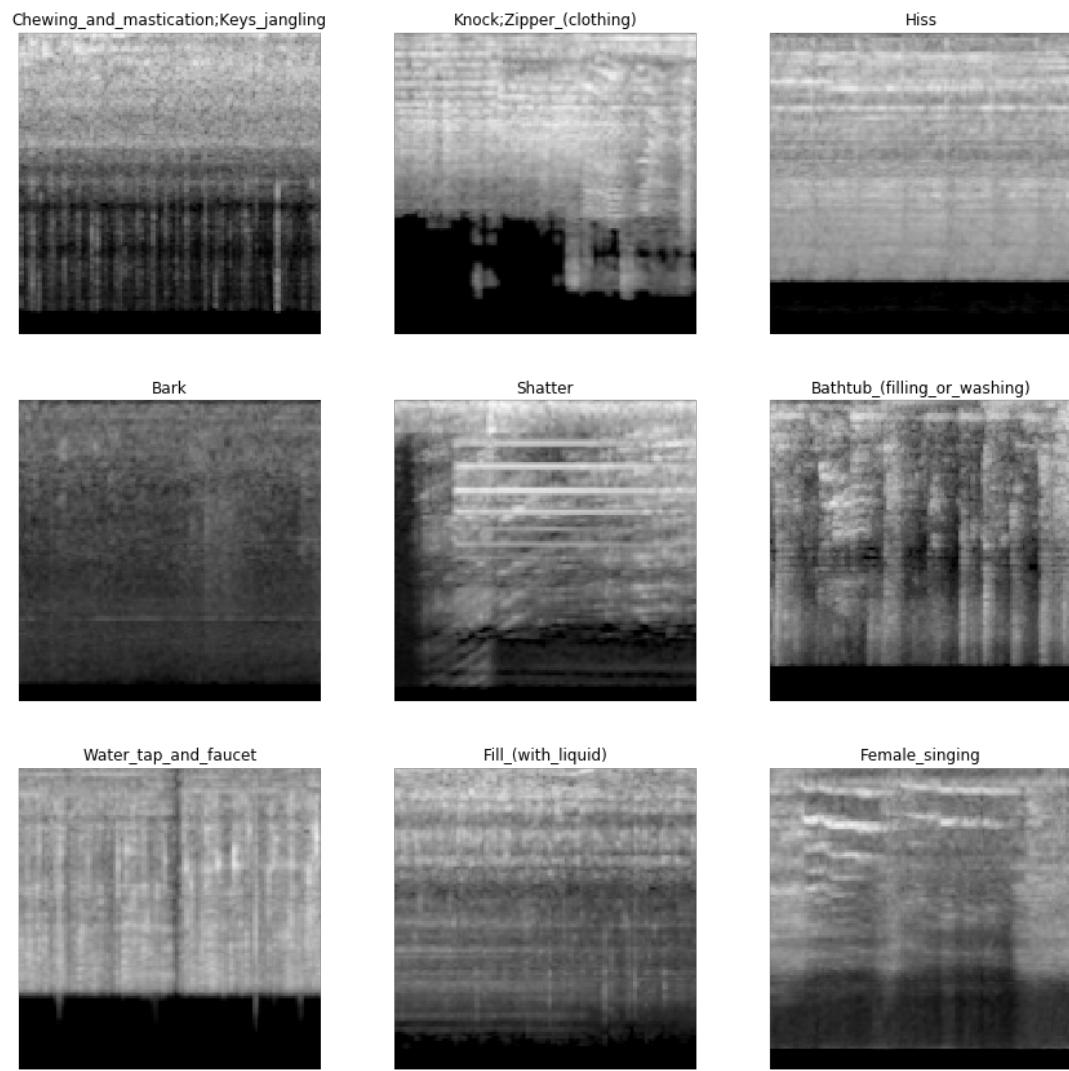
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



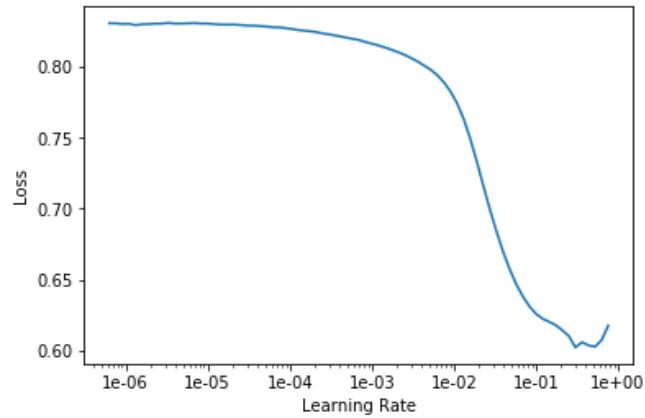
epoch	train_loss	valid_loss	lwlrap	time
0	0.071026	0.059082	0.374250	00:42
1	0.070714	0.059111	0.380408	00:42
2	0.070251	0.061041	0.355085	00:42
3	0.070405	0.060642	0.355452	00:42
4	0.070490	0.058827	0.379182	00:42
5	0.070066	0.061958	0.325832	00:42
6	0.070673	0.062121	0.322212	00:42
7	0.070365	0.059686	0.367130	00:42
8	0.070390	0.067250	0.255946	00:42
9	0.069985	0.063501	0.297125	00:42
10	0.069936	0.060101	0.364936	00:42
11	0.069952	0.061276	0.351825	00:42
12	0.069950	0.061145	0.352124	00:42
13	0.069950	0.061838	0.332860	00:42
14	0.070048	0.061984	0.326167	00:42
15	0.069702	0.059537	0.367049	00:42
16	0.069901	0.061843	0.326814	00:42
17	0.069418	0.062342	0.328735	00:42
18	0.069342	0.059716	0.376038	00:42
19	0.069117	0.062846	0.322710	00:42
20	0.068899	0.062062	0.315905	00:42
21	0.068907	0.056727	0.414699	00:42
22	0.068681	0.061081	0.349463	00:42
23	0.068800	0.062493	0.318985	00:42
24	0.068503	0.064648	0.298691	00:42
25	0.068413	0.057700	0.401722	00:42
26	0.068161	0.057305	0.401627	00:42
27	0.067701	0.060075	0.352148	00:42
28	0.067985	0.056093	0.414784	00:42
29	0.067920	0.057613	0.399660	00:42
30	0.067257	0.059818	0.360877	00:42
31	0.067246	0.055946	0.420816	00:42
32	0.067167	0.055330	0.437311	00:42
33	0.067372	0.055936	0.425673	00:42
34	0.066738	0.053333	0.463988	00:42
35	0.066391	0.054412	0.417992	00:42



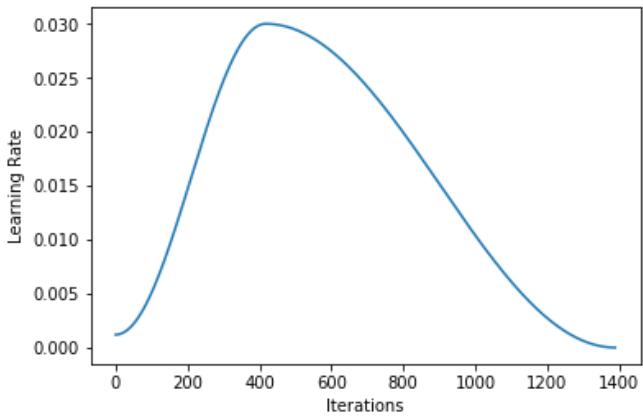
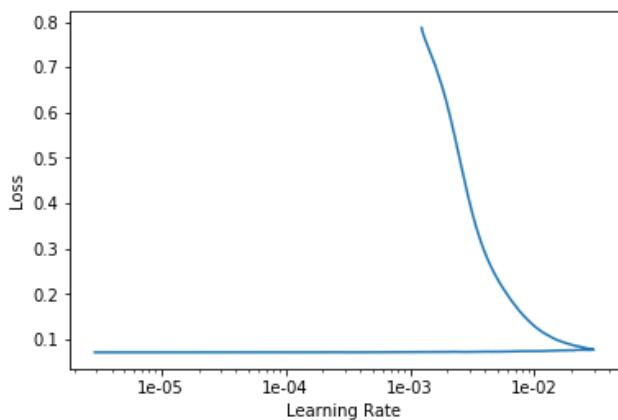
----- NOISY - Fold 8/10 -----

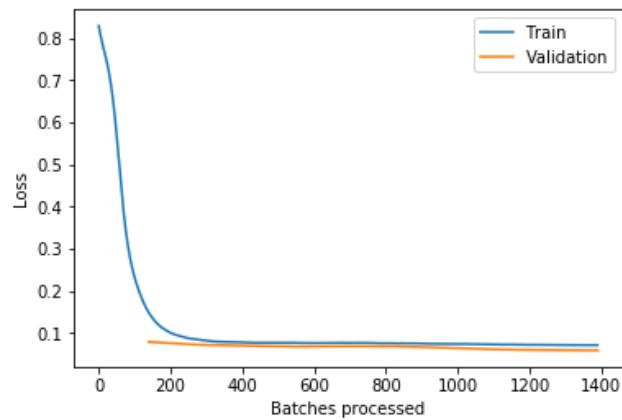


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

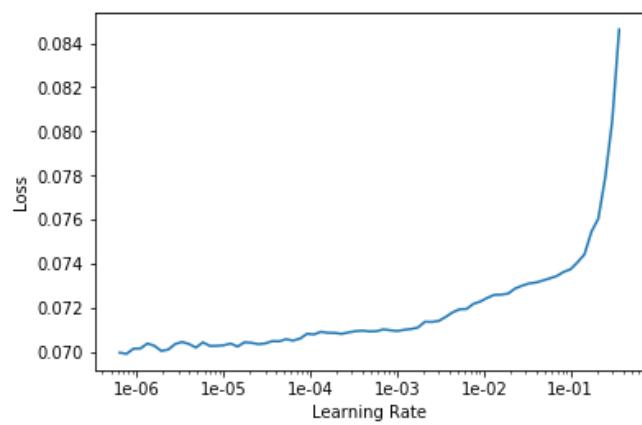


epoch	train_loss	valid_loss	lwlrap	time
0	0.149713	0.078450	0.102263	00:42
1	0.083741	0.071578	0.185226	00:42
2	0.076732	0.069044	0.238676	00:42
3	0.076140	0.067308	0.247796	00:42
4	0.076039	0.068050	0.246344	00:42
5	0.074907	0.067717	0.255963	00:42
6	0.073642	0.064282	0.294044	00:42
7	0.072327	0.060536	0.350004	00:42
8	0.071365	0.058925	0.385252	00:42
9	0.070759	0.058237	0.390941	00:42

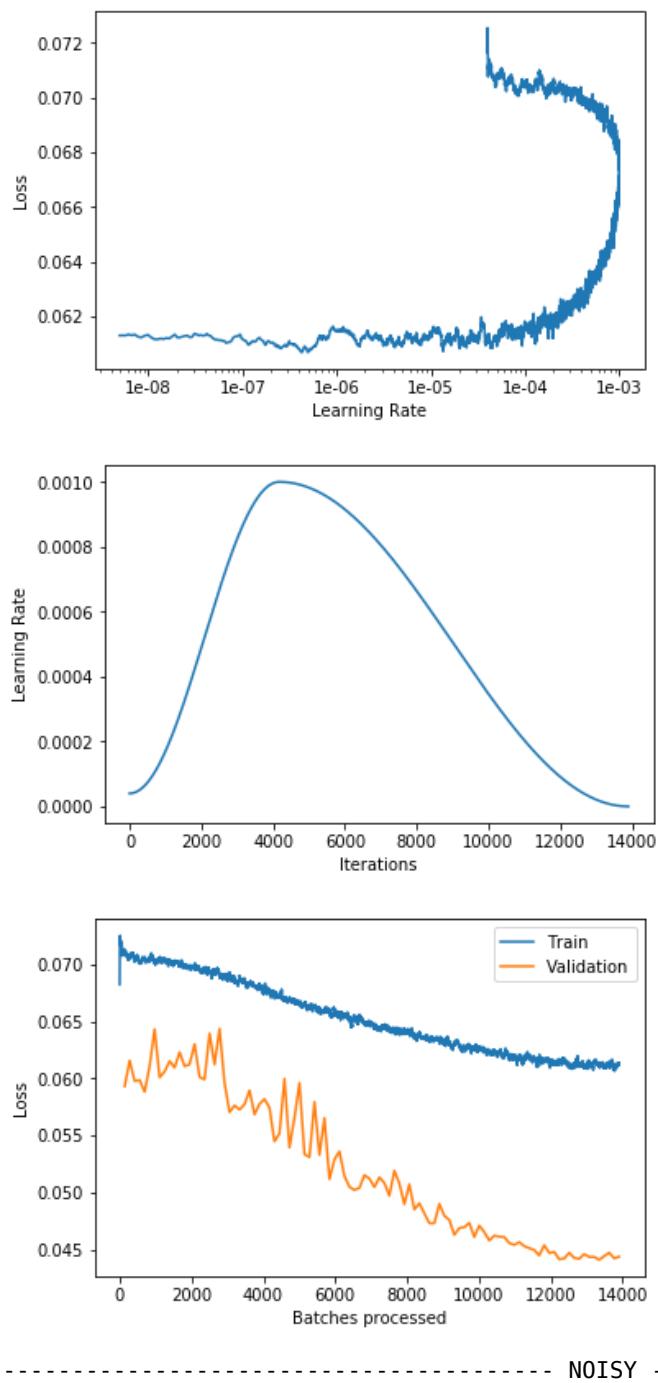




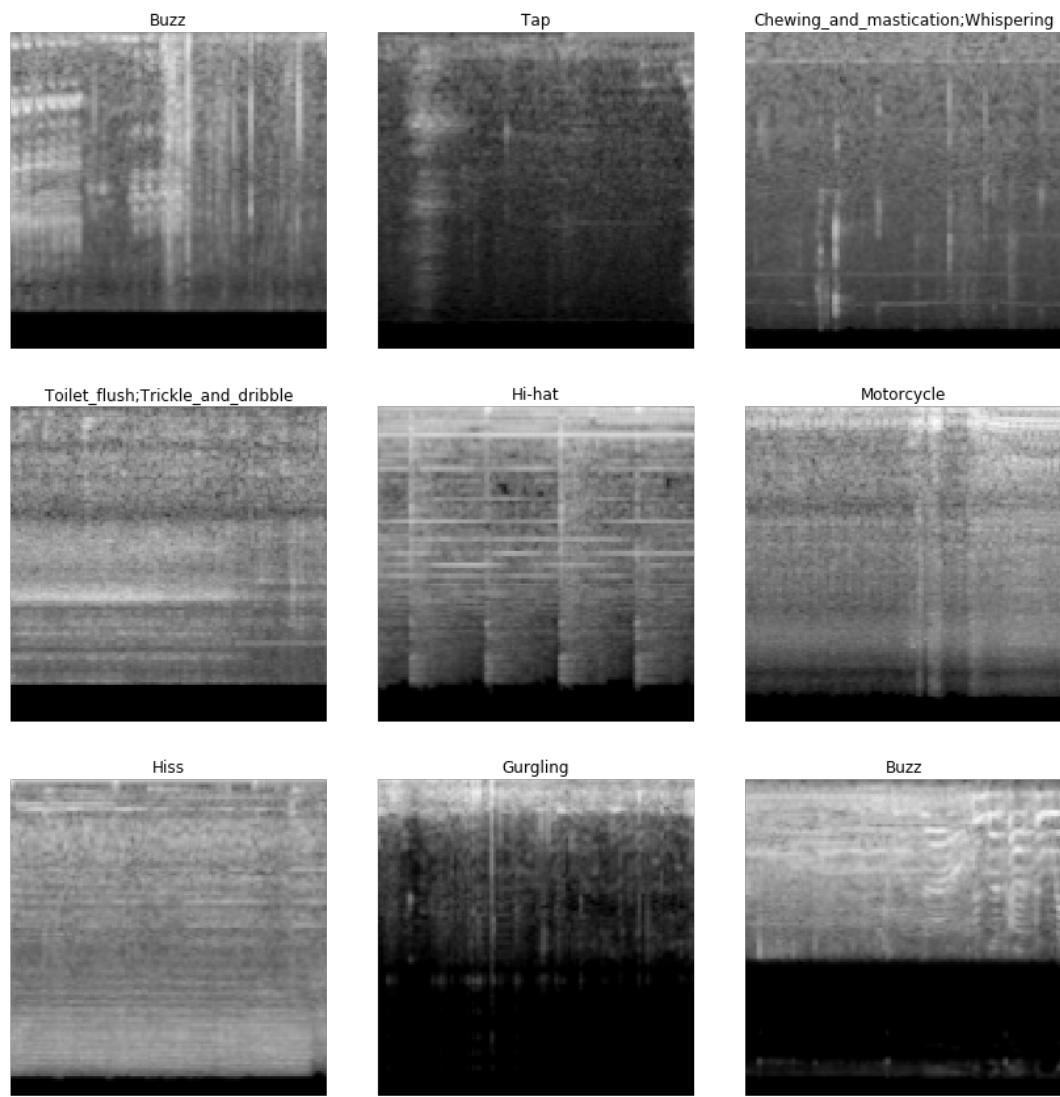
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



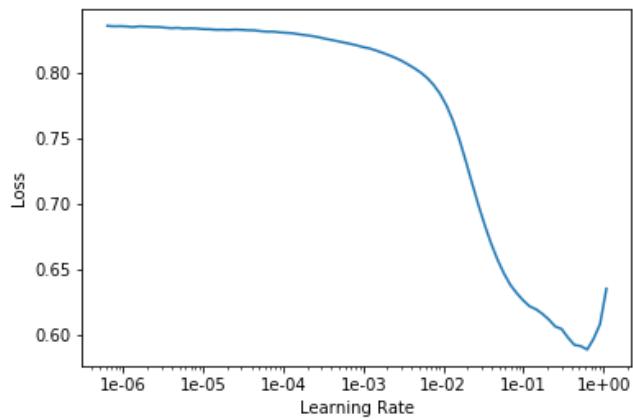
epoch	train_loss	valid_loss	lwlrap	time
0	0.070909	0.059308	0.388595	00:42
1	0.070795	0.061570	0.333819	00:42
2	0.070345	0.059779	0.372170	00:42
3	0.070270	0.059864	0.374899	00:42
4	0.070556	0.058832	0.386087	00:42
5	0.070545	0.061082	0.342538	00:42
6	0.070283	0.064326	0.292842	00:42
7	0.070411	0.060093	0.368689	00:42
8	0.070398	0.060603	0.348672	00:42
9	0.070262	0.061529	0.350124	00:42
10	0.070242	0.060957	0.353821	00:42
11	0.069946	0.062302	0.336794	00:42
12	0.069748	0.061079	0.354938	00:42
13	0.069768	0.061234	0.344943	00:42
14	0.069729	0.063013	0.319081	00:42
15	0.069471	0.060131	0.368607	00:42
16	0.069775	0.059914	0.366845	00:42
17	0.069441	0.063955	0.308244	00:42
18	0.069222	0.061225	0.346392	00:42
19	0.069029	0.064373	0.297282	00:42
20	0.068825	0.059607	0.370338	00:42
21	0.069148	0.057045	0.407325	00:42
22	0.068731	0.057628	0.420129	00:42
23	0.068944	0.057273	0.408286	00:42
24	0.068673	0.057740	0.399998	00:42
25	0.068263	0.058958	0.382988	00:42
26	0.068157	0.056853	0.411023	00:42
27	0.068315	0.057765	0.402575	00:42
28	0.068045	0.058205	0.399820	00:42
29	0.067565	0.057428	0.411207	00:42
30	0.067285	0.054476	0.464856	00:42
31	0.067190	0.055183	0.443667	00:42
32	0.066909	0.059970	0.369955	00:42
33	0.066996	0.053945	0.462611	00:42
34	0.066716	0.056564	0.414579	00:42
35	0.066676	0.059607	0.363791	00:42



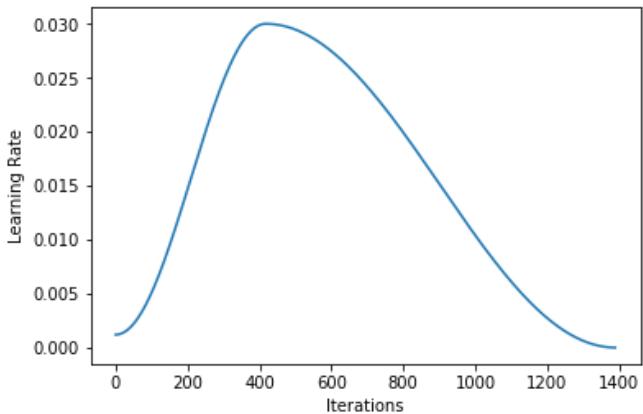
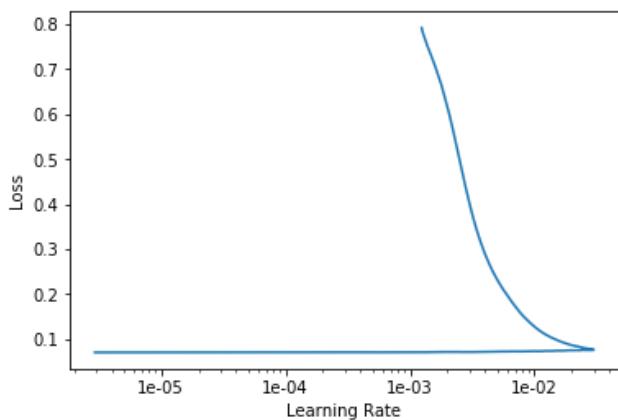
-- NOISY - Fold 9/10

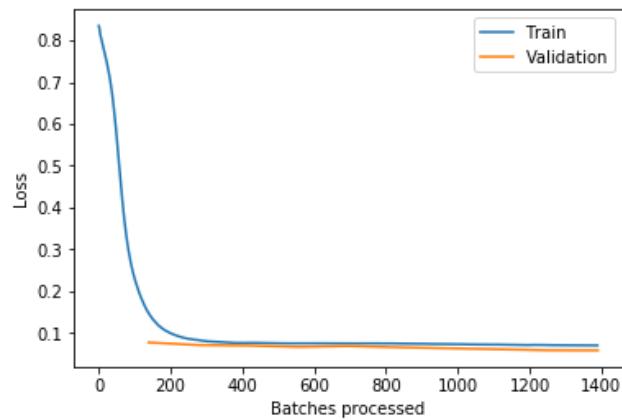


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

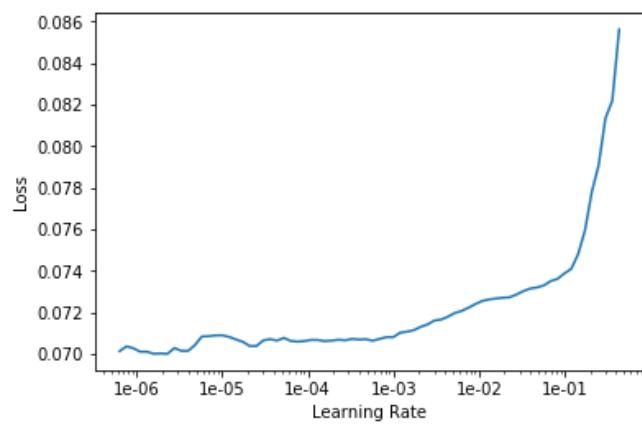


epoch	train_loss	valid_loss	lwlrap	time
0	0.148915	0.077666	0.115871	00:42
1	0.083465	0.071556	0.200527	00:42
2	0.076954	0.070341	0.219116	00:42
3	0.075958	0.067246	0.253718	00:42
4	0.075942	0.069302	0.223203	00:42
5	0.075046	0.066221	0.283408	00:42
6	0.073442	0.063858	0.307669	00:42
7	0.072704	0.061612	0.353923	00:42
8	0.071422	0.059084	0.395182	00:42
9	0.070750	0.058777	0.400403	00:42

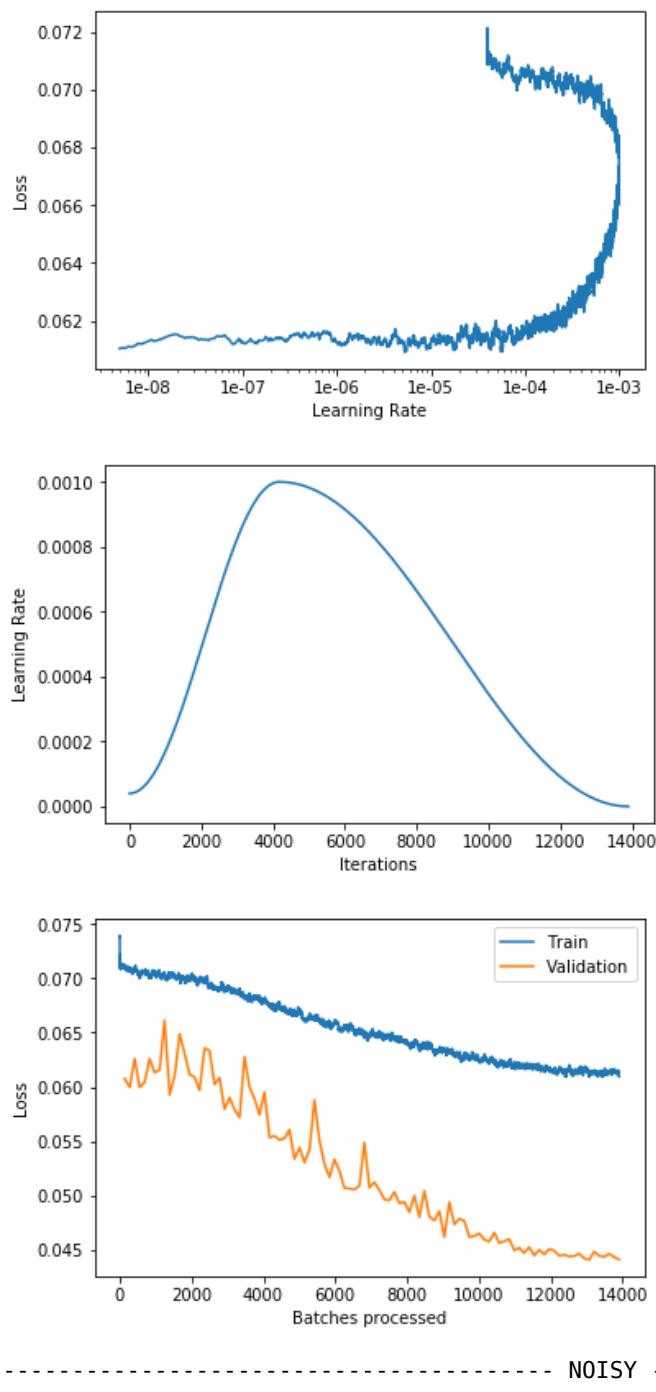


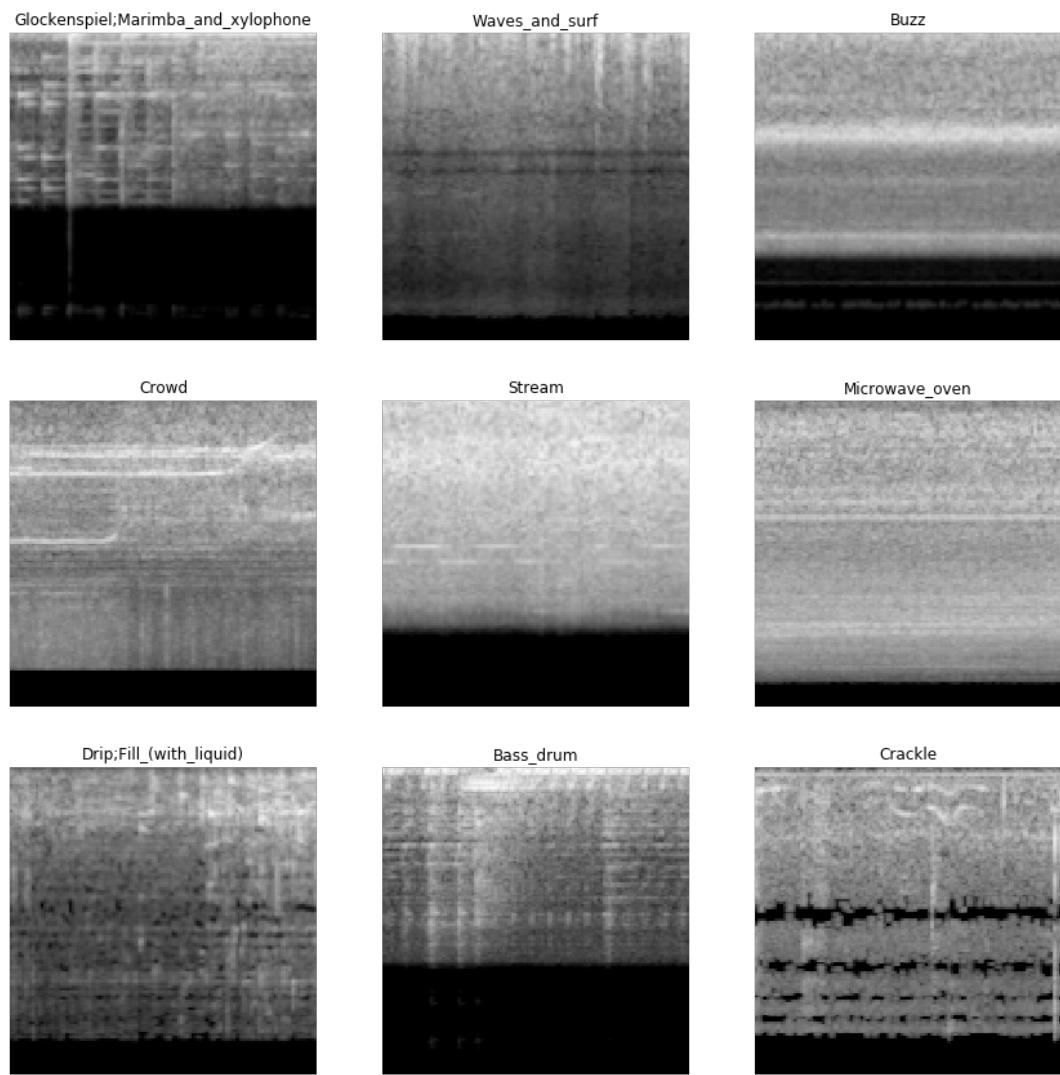


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

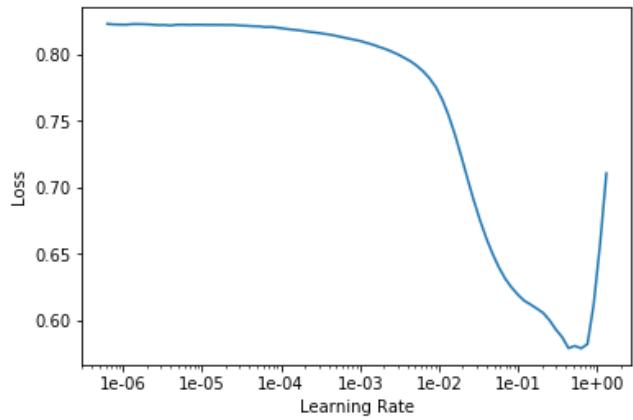


epoch	train_loss	valid_loss	lwlrap	time
0	0.071101	0.060758	0.360969	00:42
1	0.070670	0.059966	0.374037	00:42
2	0.070713	0.062598	0.325224	00:42
3	0.070364	0.059961	0.379399	00:42
4	0.070549	0.060420	0.369451	00:42
5	0.070231	0.062609	0.327592	00:42
6	0.070454	0.061328	0.345611	00:42
7	0.070455	0.061563	0.351197	00:42
8	0.070188	0.066132	0.269604	00:42
9	0.070280	0.059276	0.393061	00:42
10	0.070299	0.060994	0.375347	00:42
11	0.070174	0.064884	0.296622	00:42
12	0.069771	0.063084	0.327939	00:42
13	0.069935	0.061151	0.353773	00:42
14	0.069979	0.060888	0.351677	00:42
15	0.070058	0.059696	0.373946	00:42
16	0.069544	0.063557	0.313917	00:42
17	0.069194	0.063342	0.311165	00:42
18	0.068932	0.060233	0.365421	00:42
19	0.069009	0.060847	0.369127	00:42
20	0.068912	0.057954	0.408382	00:42
21	0.069128	0.059025	0.387031	00:42
22	0.068826	0.057903	0.397717	00:42
23	0.068951	0.057188	0.412754	00:42
24	0.068319	0.062777	0.328161	00:42
25	0.068598	0.059998	0.378338	00:42
26	0.068341	0.058911	0.383179	00:42
27	0.067924	0.057447	0.403279	00:42
28	0.068086	0.059517	0.378186	00:45
29	0.067640	0.055338	0.443709	00:44
30	0.067509	0.055501	0.447332	00:42
31	0.067593	0.055113	0.441346	00:42
32	0.067312	0.055279	0.456346	00:42
33	0.067075	0.056063	0.434585	00:42
34	0.066943	0.053375	0.476685	00:42
35	0.066783	0.054416	0.470001	00:42

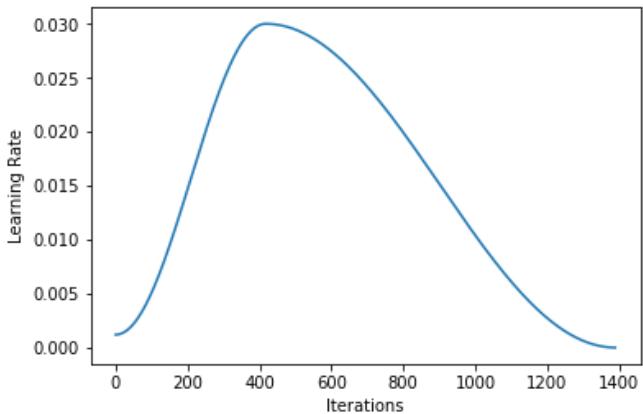
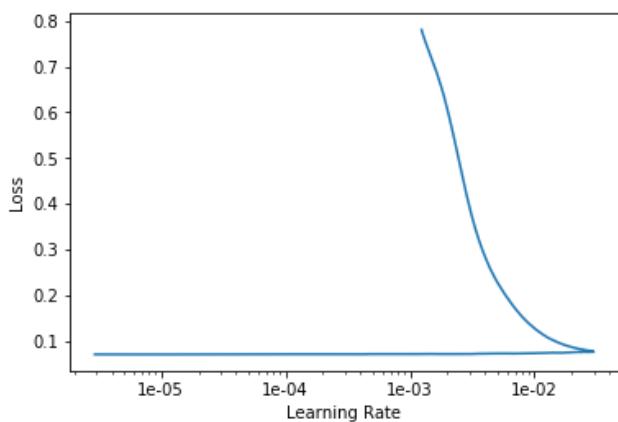


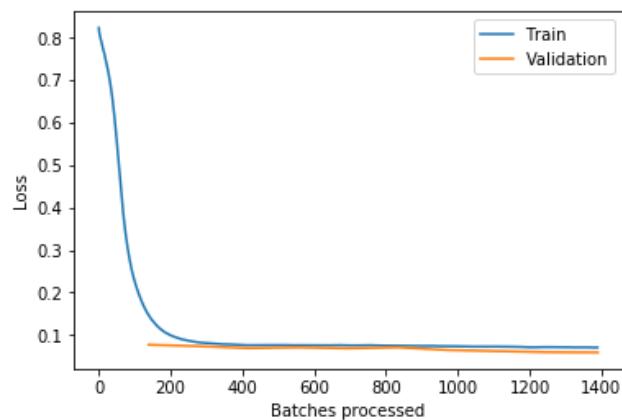


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

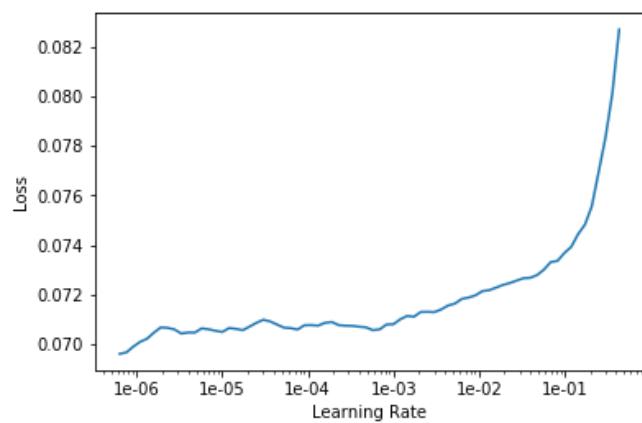


epoch	train_loss	valid_loss	lwlrap	time
0	0.147818	0.077280	0.107933	00:42
1	0.082930	0.073854	0.157279	00:42
2	0.076382	0.069312	0.231389	00:42
3	0.076128	0.070990	0.198264	00:42
4	0.075669	0.068838	0.249929	00:42
5	0.074982	0.071090	0.223742	00:42
6	0.073817	0.064750	0.290848	00:42
7	0.072931	0.062116	0.333173	00:42
8	0.071850	0.059757	0.378994	00:42
9	0.070840	0.059230	0.391235	00:42

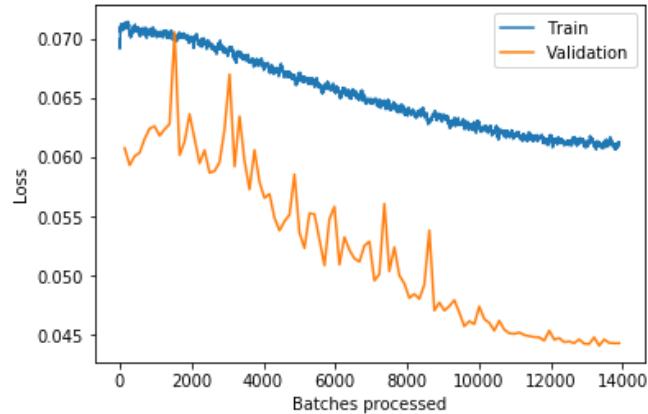
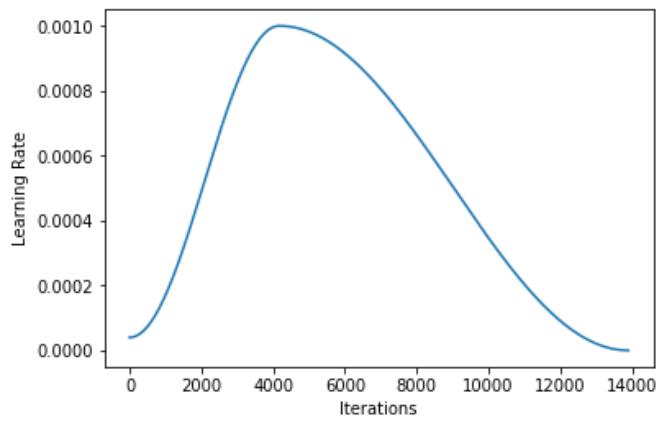
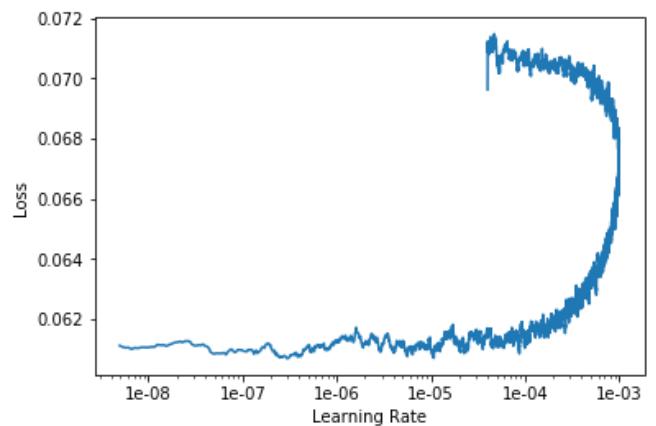




LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.071177	0.060797	0.351584	00:42
1	0.070534	0.059349	0.385828	00:42
2	0.070957	0.060103	0.365446	00:42
3	0.070504	0.060401	0.371242	00:42
4	0.070566	0.061566	0.341732	00:42
5	0.070594	0.062452	0.332109	00:42
6	0.070449	0.062635	0.324150	00:42
7	0.070506	0.061831	0.341816	00:42
8	0.070649	0.062379	0.331261	00:42
9	0.070136	0.062838	0.327107	00:42
10	0.070201	0.070567	0.325133	00:42
11	0.070428	0.060202	0.362748	00:42
12	0.070223	0.061346	0.338680	00:42
13	0.070004	0.063650	0.315103	00:42
14	0.069779	0.061574	0.344201	00:42
15	0.069537	0.059506	0.374753	00:43
16	0.069677	0.060620	0.344731	00:42
17	0.069631	0.058708	0.390709	00:42
18	0.069164	0.058855	0.392949	00:42
19	0.069444	0.059598	0.379888	00:42
20	0.069042	0.062378	0.332469	00:42
21	0.069150	0.067016	0.267168	00:42
22	0.068745	0.059231	0.385477	00:42
23	0.068533	0.063477	0.306335	00:42
24	0.068256	0.059707	0.383529	00:42
25	0.068103	0.057311	0.417874	00:42
26	0.068082	0.060626	0.352705	00:42
27	0.068162	0.057921	0.400101	00:42
28	0.067617	0.056575	0.430392	00:42
29	0.067230	0.056934	0.420485	00:42
30	0.067412	0.054903	0.449051	00:42
31	0.067351	0.053835	0.466298	00:42
32	0.066947	0.054619	0.451194	00:42
33	0.067117	0.055152	0.438795	00:42
34	0.066797	0.058569	0.407601	00:42
35	0.067032	0.053653	0.460450	00:42



```
In [23]: learn.model
```

```
Out[23]: Sequential(  
    (0): Sequential(  
        (0): Sequential(  
            (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (2): ReLU(inplace)  
            (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (5): ReLU(inplace)  
            (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
            (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (9): ReLU(inplace)  
            (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (12): ReLU(inplace)  
            (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
            (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (16): ReLU(inplace)  
            (17): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (19): ReLU(inplace)  
            (20): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (22): ReLU(inplace)  
            (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
            (24): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (25): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (26): ReLU(inplace)  
            (27): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (28): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (29): ReLU(inplace)  
            (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (32): ReLU(inplace)  
            (33): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
            (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (36): ReLU(inplace)  
            (37): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (38): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (39): ReLU(inplace)  
            (40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
            (41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
            (42): ReLU(inplace)
```

```
In [24]: kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-1_fold-{fold}')
```

Overall lwlrapi on CURATED dataset: 0.41793918072817265

	lwlrap	weight
Raindrop	0.034137	0.013039
Tap	0.037902	0.013039
Yell	0.045295	0.013039
Female_speech_and_woman_speaking	0.049855	0.013039
Cupboard_open_or_close	0.093520	0.013039
Male_speech_and_man_speaking	0.099777	0.013039
Run	0.106769	0.013039
Whispering	0.121909	0.013039
Bass_drum	0.131903	0.013039
Fill_(with_liquid)	0.170228	0.008693
Strum	0.177163	0.013039
Slam	0.191150	0.013039
Walk_and_footsteps	0.209183	0.013039
Gurgling	0.210482	0.013039
Burping_and_eructation	0.233793	0.013039
Clapping	0.238994	0.013039
Trickle_and_dribble	0.245641	0.009214
Buzz	0.249595	0.009736
Mechanical_fan	0.254770	0.008519
Finger_snapping	0.254996	0.013039
Drip	0.257066	0.013039
Fart	0.266189	0.013039
Waves_and_surf	0.266513	0.013039
Electric_guitar	0.268338	0.013039
Shatter	0.272929	0.013039
Writing	0.274073	0.013039
Meow	0.274090	0.013039
Male_singing	0.283799	0.013039
Child_speech_and_kid_speaking	0.290150	0.013039
Drawer_open_or_close	0.294430	0.013039
...	...	...
Skateboard	0.495465	0.013039
Car_passing_by	0.498440	0.013039
Toilet_flush	0.502495	0.013039
Hi-hat	0.513052	0.013039

```
In [25]: kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-1_fold-{fold}')
```

Overall lwlrapp on NOISY dataset: 0.6596067715667252

	<b>lwlrap</b>	<b>weight</b>
<b>Zipper_(clothing)</b>	0.253919	0.0125
<b>Burping_and_eructation</b>	0.267249	0.0125
<b>Fart</b>	0.303172	0.0125
<b>Writing</b>	0.361099	0.0125
<b>Buzz</b>	0.396116	0.0125
<b>Gasp</b>	0.398673	0.0125
<b>Sneeze</b>	0.408500	0.0125
<b>Whispering</b>	0.447978	0.0125
<b>Computer_keyboard</b>	0.453490	0.0125
<b>Finger_snapping</b>	0.455088	0.0125
<b>Tick-tock</b>	0.473409	0.0125
<b>Scissors</b>	0.479686	0.0125
<b>Sigh</b>	0.495320	0.0125
<b>Shatter</b>	0.497038	0.0125
<b>Cupboard_open_or_close</b>	0.497147	0.0125
<b>Toilet_flush</b>	0.506049	0.0125
<b>Purr</b>	0.538479	0.0125
<b>Knock</b>	0.539005	0.0125
<b>Strum</b>	0.549273	0.0125
<b>Meow</b>	0.556268	0.0125
<b>Male_singing</b>	0.559957	0.0125
<b>Bicycle_bell</b>	0.565894	0.0125
<b>Chewing_and_mastication</b>	0.573042	0.0125
<b>Chink_and_clink</b>	0.575655	0.0125
<b>Squeak</b>	0.594120	0.0125
<b>Skateboard</b>	0.597092	0.0125
<b>Electric_guitar</b>	0.617756	0.0125
<b>Bass_guitar</b>	0.643524	0.0125
<b>Screaming</b>	0.644957	0.0125
<b>Keys_jangling</b>	0.647853	0.0125
...	...	...
<b>Sink_(filling_or_washing)</b>	0.733934	0.0125
<b>Bus</b>	0.734063	0.0125
<b>Bass_drum</b>	0.734765	0.0125
<b>Motorcycle</b>	0.740051	0.0125

**Train on curated data**

```
In [26]: for fold, (train_index, valid_index) in enumerate(kf_curated.split(trn_curated_df)):
    print('-' * 40, f'CURATED - Fold {fold+1}/{n_splits}', '-' * 40)

    learn.load(f'stage-1_fold-{fold}')

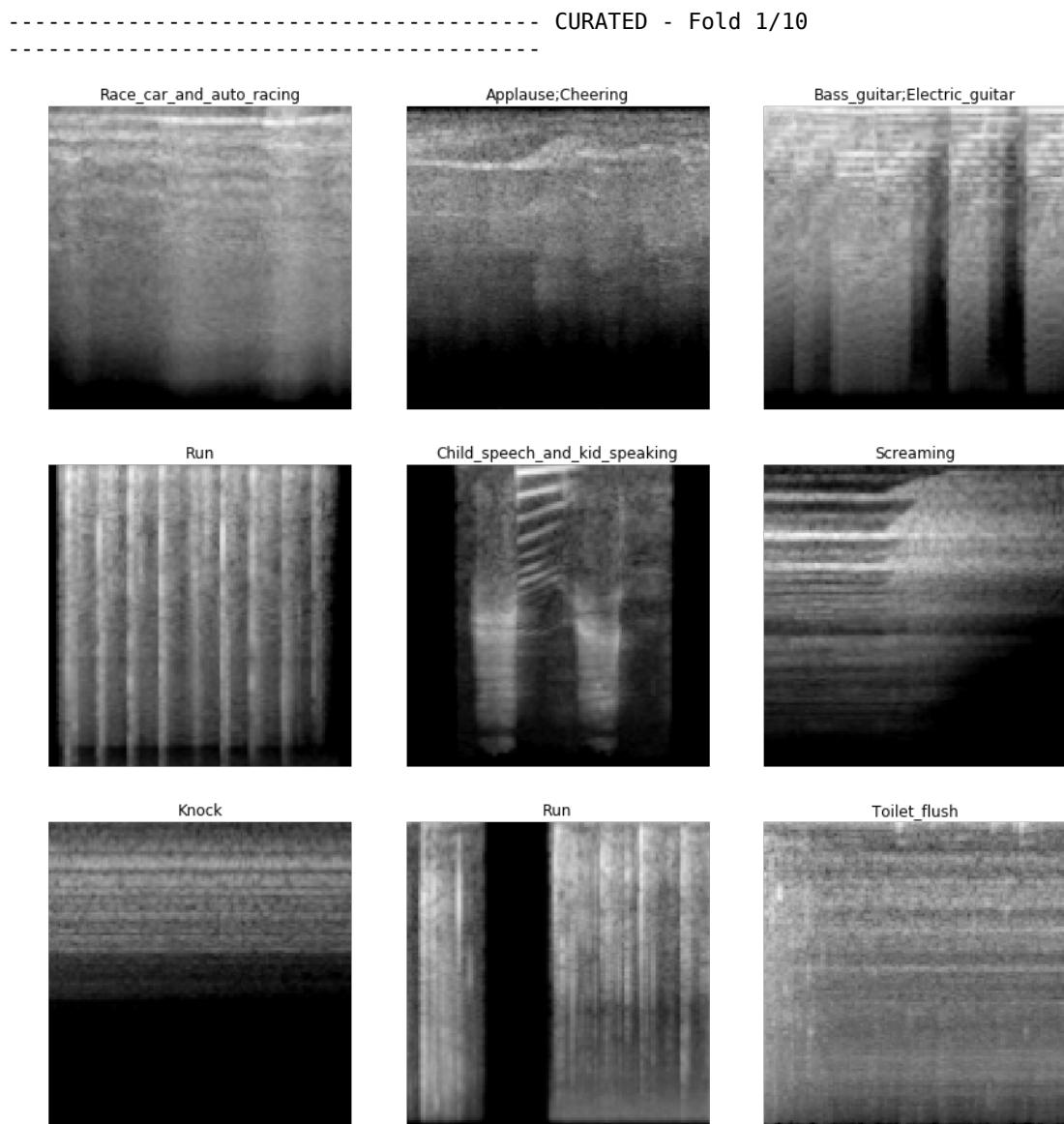
    src = (ImageList.from_df(trn_curated_df, WORK)
           .split_by_idxs(train_index, valid_index)
           .label_from_df(label_delim=','))
    )
    data = (src.transform(tfms, size=128)
            .databunch(bs=bs))
    )
    data.show_batch(3)
    plt.show()

    learn.data = data

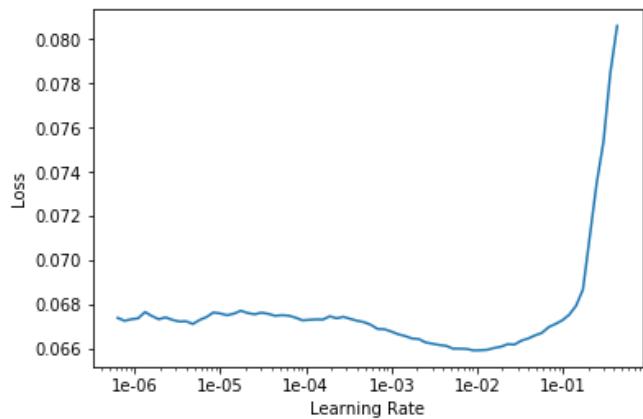
    learning(learn, 100, 3e-3 / 5)

    learn.save(f'stage-2_fold-{fold}')
    learn.export(f'stage-2_fold-{fold}.pkl')

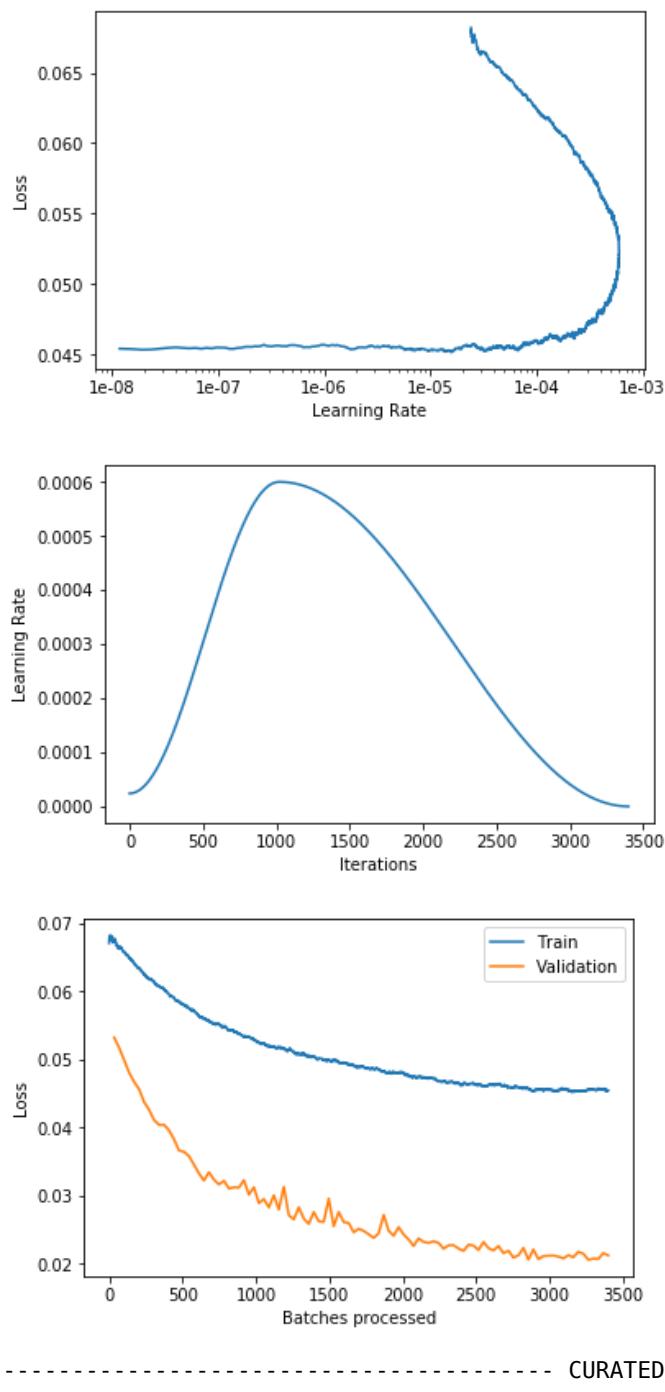
if TOY_MODE:
    break
```



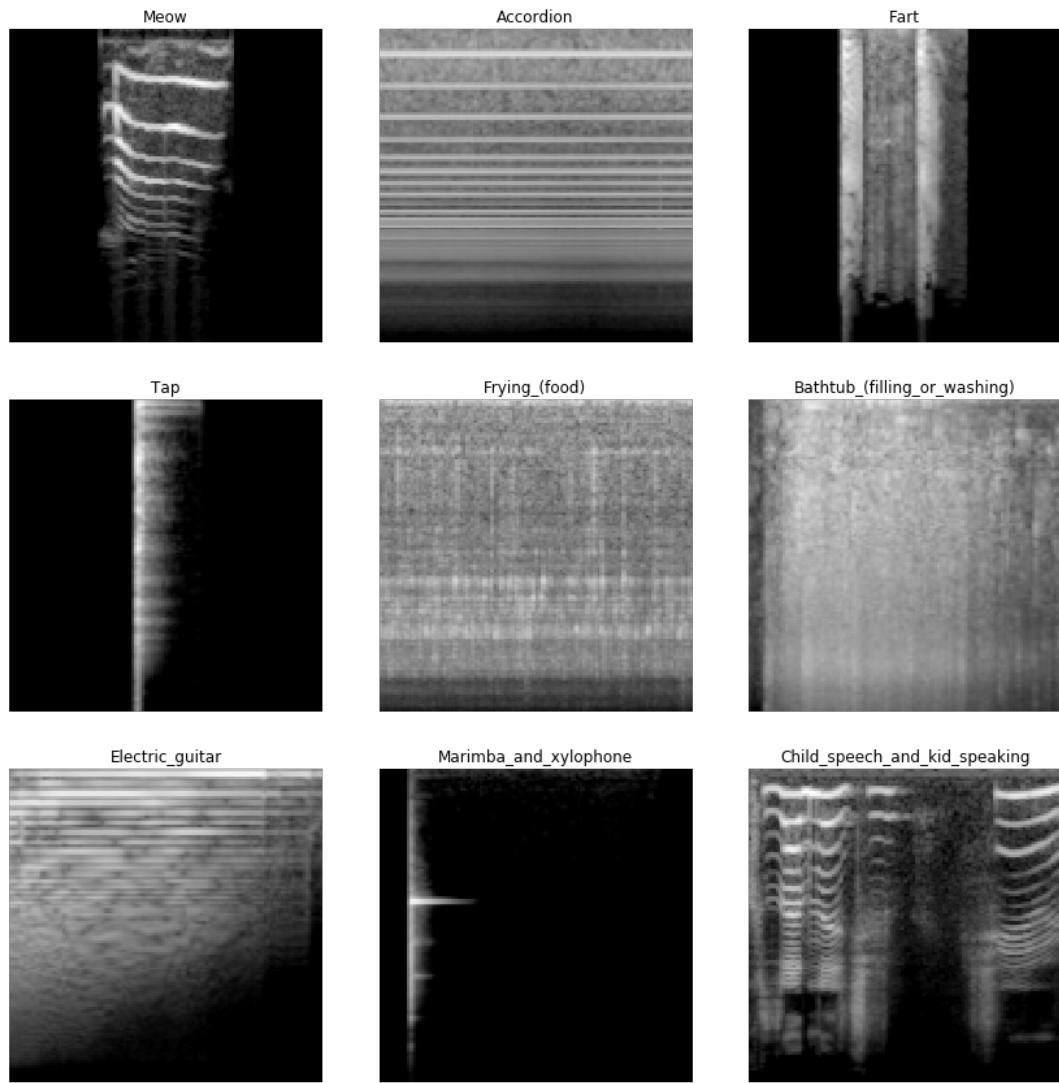
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



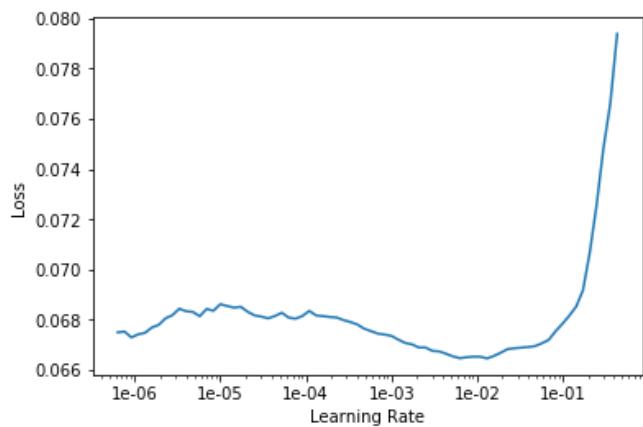
epoch	train_loss	valid_loss	lwlrap	time
0	0.067560	0.053204	0.447290	00:14
1	0.066541	0.051695	0.460949	00:13
2	0.065804	0.049896	0.493618	00:12
3	0.064905	0.048046	0.513841	00:12
4	0.064113	0.046691	0.536240	00:12
5	0.063435	0.045611	0.553793	00:12
6	0.062606	0.043731	0.579125	00:12
7	0.061863	0.042592	0.599247	00:12
8	0.061307	0.041082	0.630123	00:12
9	0.060750	0.040351	0.628844	00:12
10	0.060261	0.040375	0.629698	00:12
11	0.059540	0.039563	0.633443	00:12
12	0.059121	0.038240	0.663157	00:13
13	0.058437	0.036584	0.688425	00:12
14	0.058024	0.036418	0.690308	00:12
15	0.057481	0.035771	0.700057	00:12
16	0.057090	0.034504	0.706951	00:12
17	0.056417	0.033244	0.723141	00:13
18	0.056048	0.032185	0.739622	00:12
19	0.055651	0.033408	0.718466	00:12
20	0.055319	0.032320	0.741972	00:12
21	0.055152	0.031608	0.760927	00:12
22	0.054840	0.032149	0.743341	00:12
23	0.054285	0.031003	0.744096	00:12
24	0.054142	0.031204	0.748405	00:12
25	0.053762	0.031141	0.733361	00:12
26	0.053323	0.032232	0.734128	00:12
27	0.053299	0.030118	0.749438	00:12
28	0.052959	0.031198	0.752609	00:12
29	0.052545	0.028865	0.768935	00:12
30	0.052348	0.029471	0.770934	00:12
31	0.052025	0.028228	0.771342	00:12
32	0.051837	0.030021	0.755170	00:12
33	0.051721	0.027924	0.784299	00:12
34	0.051542	0.031270	0.739997	00:12
35	0.051482	0.027100	0.701879	00:12



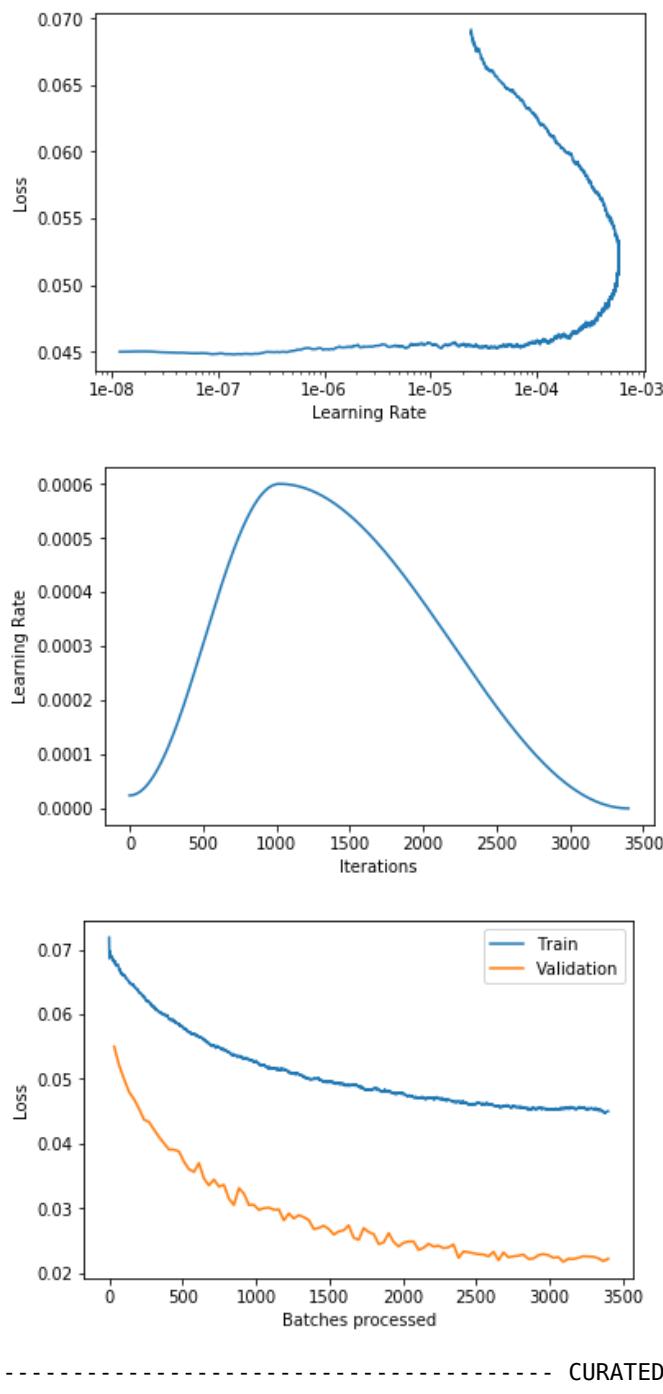
----- CURATED - Fold 2/10 -----

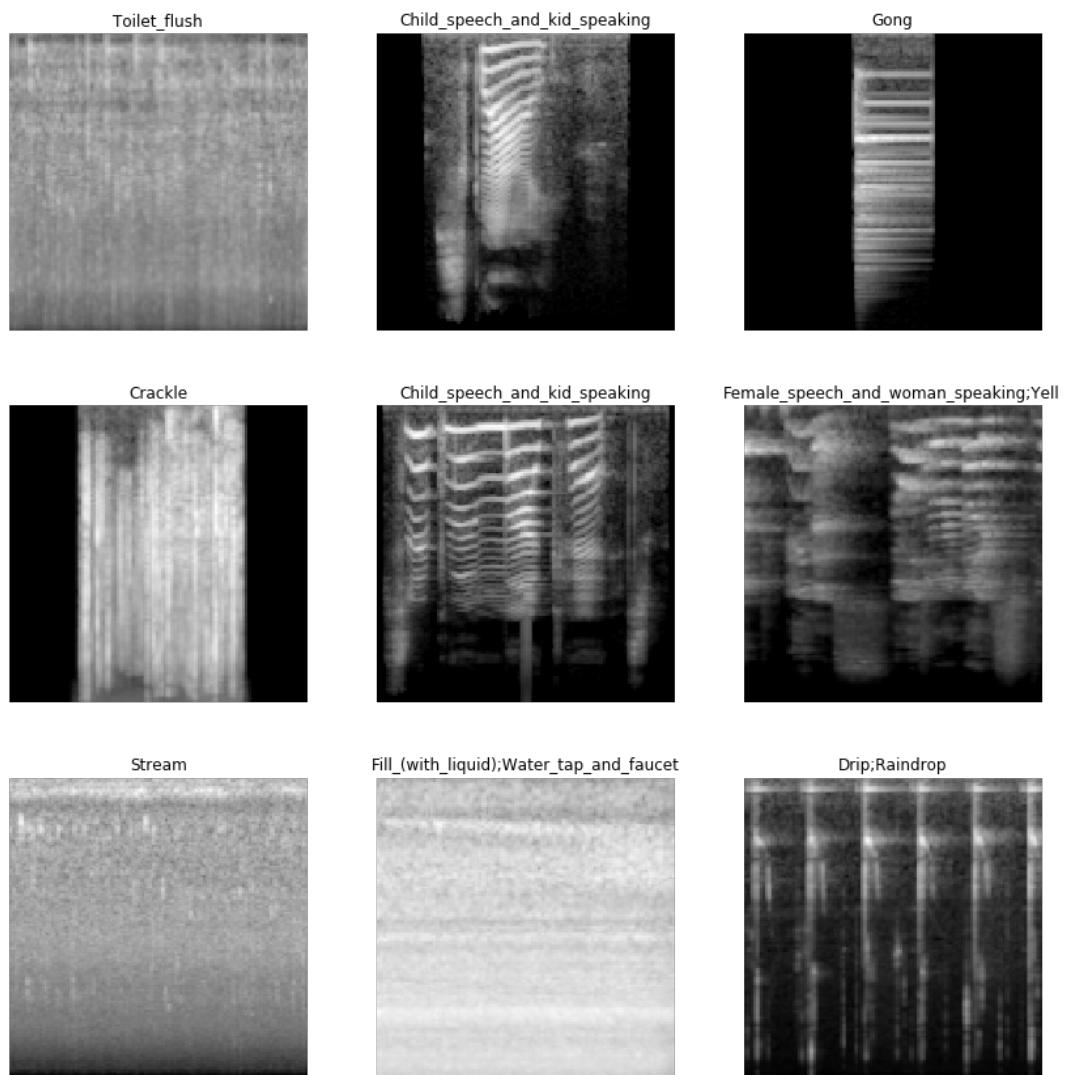


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

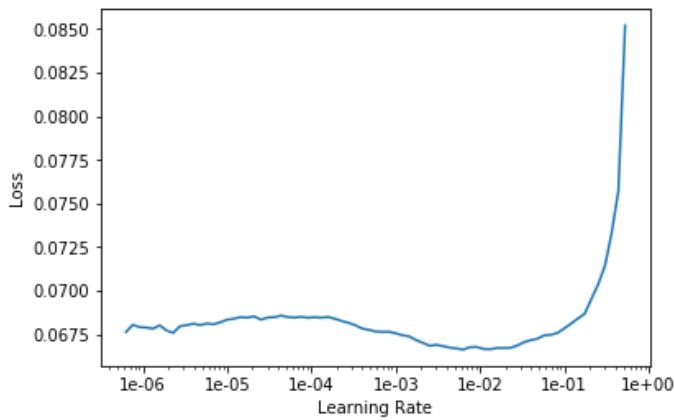


epoch	train_loss	valid_loss	lwlrap	time
0	0.068257	0.055024	0.427220	00:12
1	0.067036	0.052123	0.461244	00:12
2	0.065987	0.049973	0.493262	00:12
3	0.065098	0.047989	0.523737	00:12
4	0.064399	0.046871	0.543114	00:12
5	0.063415	0.045396	0.576932	00:13
6	0.062692	0.043728	0.596825	00:13
7	0.061947	0.043355	0.592185	00:12
8	0.061102	0.042175	0.628479	00:12
9	0.060399	0.041039	0.640604	00:12
10	0.059925	0.040064	0.648386	00:12
11	0.059325	0.039098	0.679134	00:12
12	0.058943	0.039090	0.674609	00:12
13	0.058553	0.038803	0.662971	00:12
14	0.057854	0.037206	0.682272	00:13
15	0.057246	0.036057	0.686394	00:13
16	0.056978	0.035647	0.692508	00:12
17	0.056459	0.037020	0.681542	00:12
18	0.056087	0.034624	0.718838	00:12
19	0.055475	0.033576	0.728959	00:12
20	0.055192	0.034474	0.726303	00:12
21	0.054885	0.033385	0.735400	00:12
22	0.054499	0.033661	0.720843	00:12
23	0.054158	0.031508	0.754357	00:12
24	0.053819	0.030532	0.775088	00:12
25	0.053536	0.033144	0.743126	00:12
26	0.053265	0.032310	0.726064	00:12
27	0.053067	0.030547	0.754036	00:12
28	0.052647	0.030570	0.764539	00:12
29	0.052368	0.029779	0.784526	00:12
30	0.052217	0.030042	0.771057	00:12
31	0.051777	0.030133	0.756232	00:12
32	0.051654	0.029811	0.768412	00:12
33	0.051577	0.029852	0.766178	00:12
34	0.051440	0.028160	0.778739	00:12
35	0.051046	0.029235	0.767095	00:12

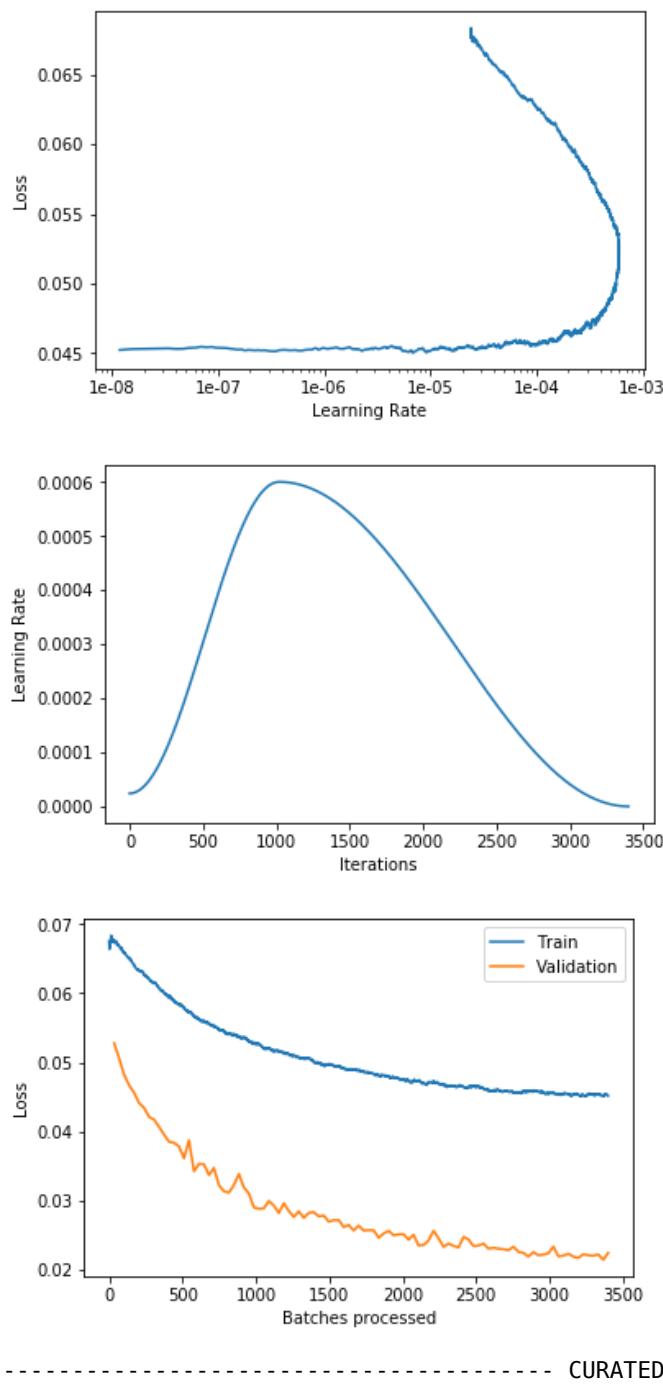




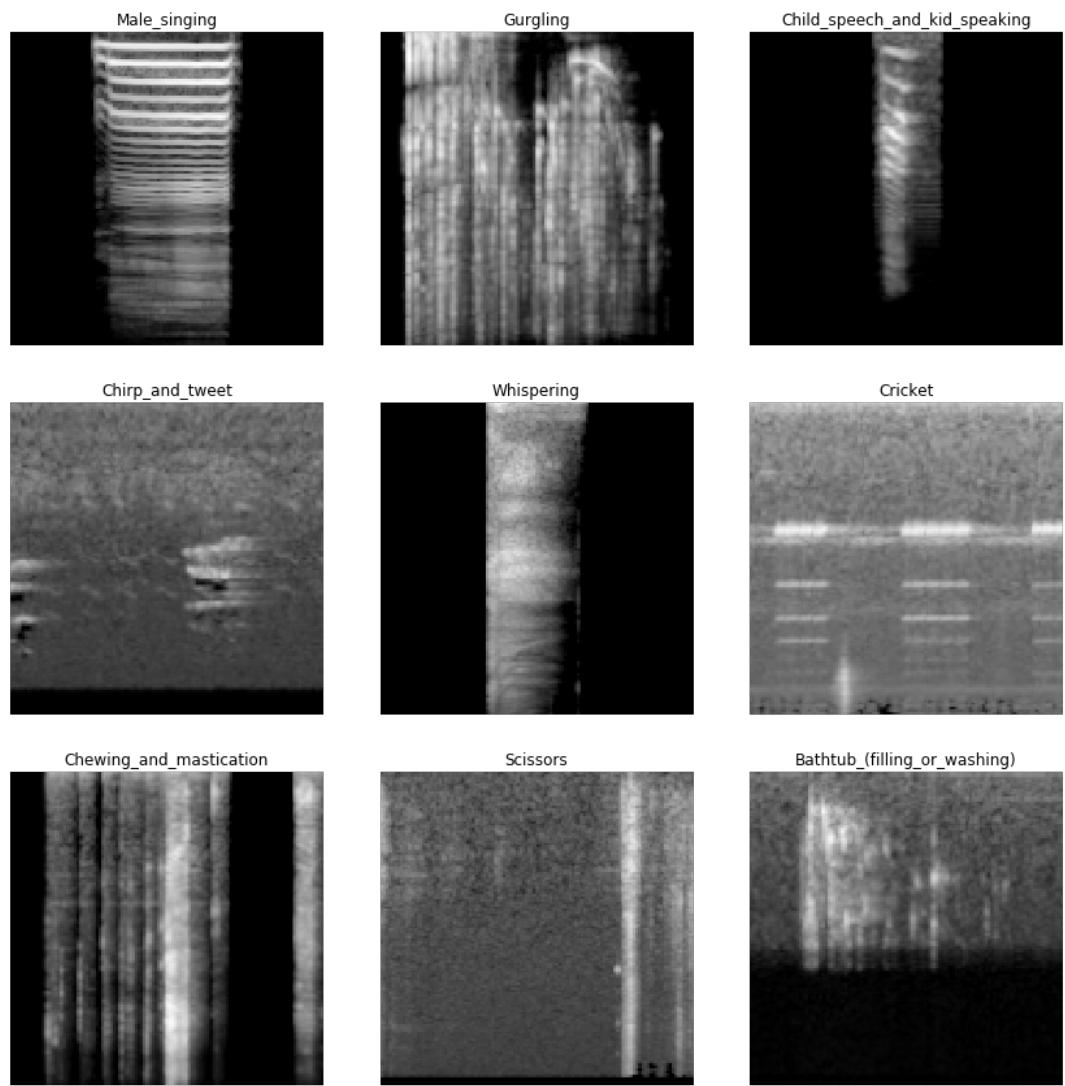
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



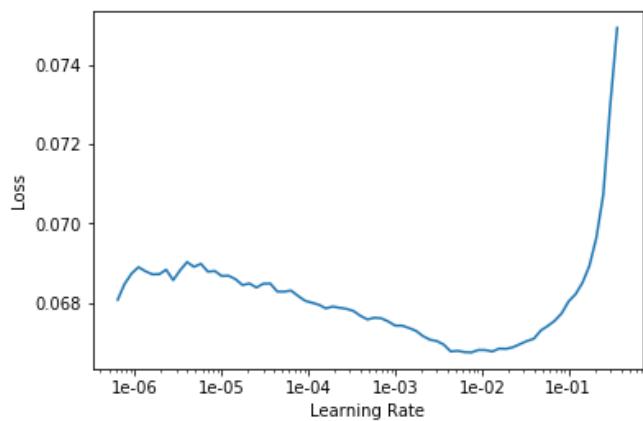
epoch	train_loss	valid_loss	lwlrap	time
0	0.067696	0.052757	0.446849	00:12
1	0.066911	0.050572	0.469442	00:12
2	0.065835	0.048194	0.509667	00:12
3	0.065186	0.046691	0.529259	00:12
4	0.064128	0.045609	0.555236	00:12
5	0.063347	0.044103	0.564535	00:12
6	0.062794	0.043421	0.570729	00:12
7	0.062088	0.042098	0.607889	00:12
8	0.061501	0.041699	0.603424	00:13
9	0.060792	0.040660	0.621594	00:12
10	0.060118	0.039547	0.653398	00:12
11	0.059471	0.038490	0.651767	00:12
12	0.058980	0.038359	0.673830	00:12
13	0.058400	0.037825	0.669275	00:12
14	0.057877	0.036103	0.688396	00:12
15	0.057393	0.038725	0.648441	00:12
16	0.056686	0.034237	0.695949	00:12
17	0.056284	0.035296	0.697400	00:12
18	0.055968	0.035255	0.699751	00:12
19	0.055633	0.033683	0.716455	00:12
20	0.055176	0.034727	0.704816	00:12
21	0.054737	0.032226	0.732953	00:12
22	0.054528	0.031330	0.739811	00:12
23	0.054124	0.031140	0.742684	00:12
24	0.054054	0.032254	0.724375	00:12
25	0.053770	0.033834	0.719849	00:12
26	0.053548	0.031889	0.731751	00:12
27	0.053040	0.031003	0.740491	00:12
28	0.052777	0.029003	0.772224	00:12
29	0.052672	0.028803	0.774658	00:13
30	0.051991	0.028872	0.757229	00:13
31	0.052068	0.029940	0.758473	00:13
32	0.051716	0.029270	0.742415	00:13
33	0.051520	0.028189	0.774384	00:13
34	0.051407	0.029603	0.758669	00:13
35	0.051207	0.028195	0.770373	00:13



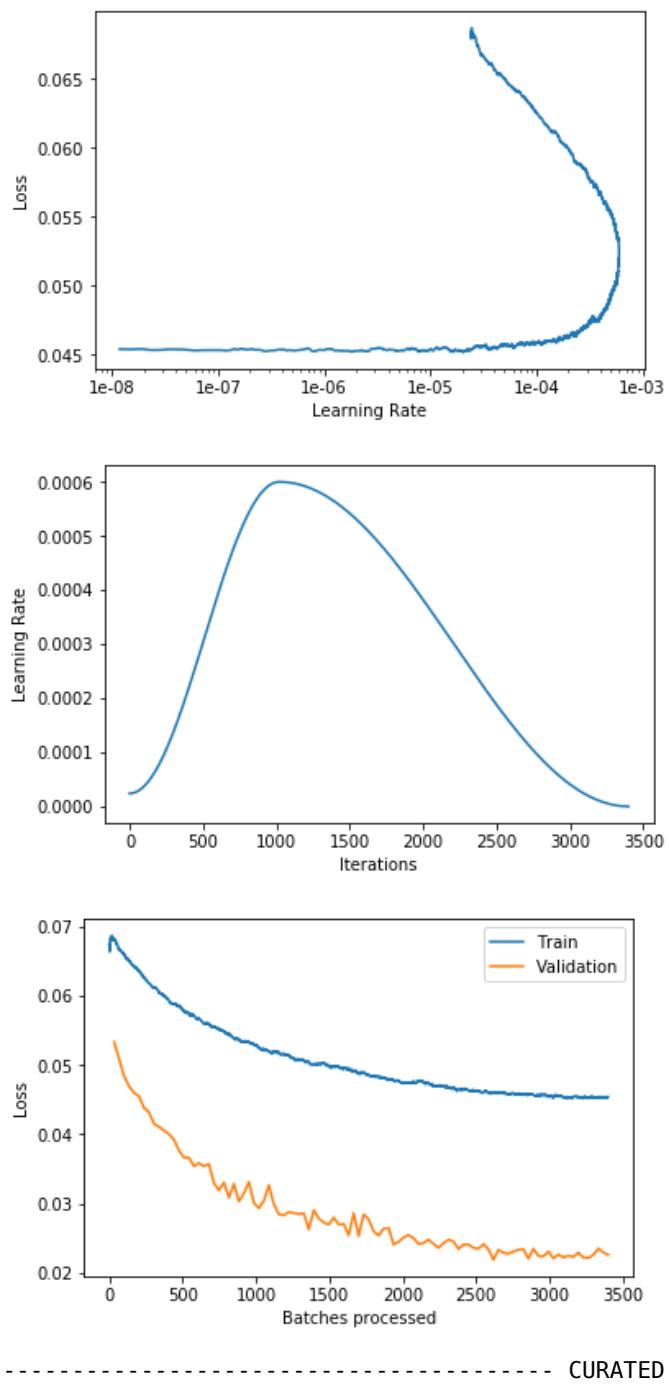
----- CURATED - Fold 4/10 -----



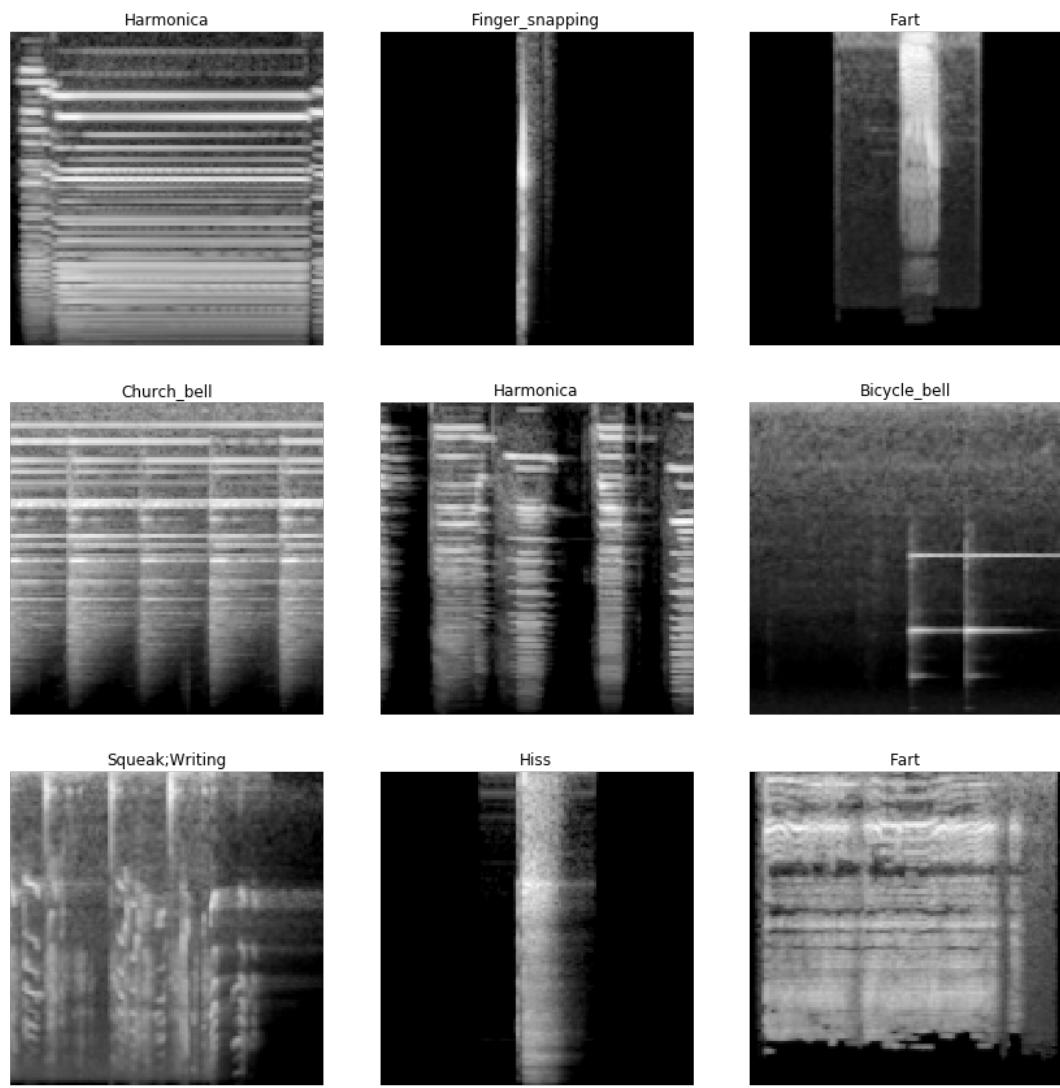
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



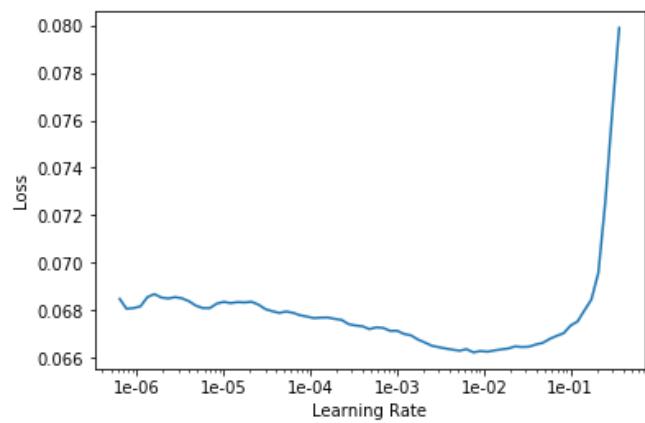
epoch	train_loss	valid_loss	lwlrap	time
0	0.068007	0.053342	0.451241	00:12
1	0.066780	0.050868	0.493089	00:12
2	0.066012	0.048397	0.516816	00:12
3	0.065103	0.046911	0.536954	00:12
4	0.064287	0.046005	0.556202	00:12
5	0.063576	0.045448	0.557447	00:12
6	0.062644	0.043829	0.591507	00:12
7	0.061780	0.043165	0.601312	00:12
8	0.061209	0.041513	0.623603	00:12
9	0.060520	0.041064	0.615113	00:12
10	0.060028	0.040531	0.627271	00:13
11	0.059302	0.040015	0.630428	00:12
12	0.058790	0.039182	0.652548	00:13
13	0.058483	0.037641	0.682813	00:13
14	0.057796	0.036627	0.684035	00:12
15	0.057507	0.036608	0.680749	00:12
16	0.057116	0.035394	0.701415	00:12
17	0.056786	0.035830	0.690755	00:12
18	0.055929	0.035383	0.698576	00:13
19	0.055919	0.035698	0.689572	00:12
20	0.055541	0.032959	0.742613	00:12
21	0.055207	0.031916	0.750800	00:12
22	0.054886	0.033034	0.732516	00:12
23	0.054455	0.030864	0.747602	00:12
24	0.053858	0.032849	0.717339	00:12
25	0.053706	0.030307	0.768675	00:12
26	0.053433	0.031477	0.745374	00:12
27	0.053325	0.033117	0.725582	00:12
28	0.052979	0.030083	0.759108	00:12
29	0.052502	0.029314	0.765045	00:12
30	0.052322	0.030393	0.755825	00:12
31	0.052052	0.032660	0.711829	00:13
32	0.051871	0.029930	0.768033	00:13
33	0.051801	0.028435	0.773453	00:12
34	0.051647	0.028300	0.775788	00:13
35	0.051301	0.028762	0.761317	00:13



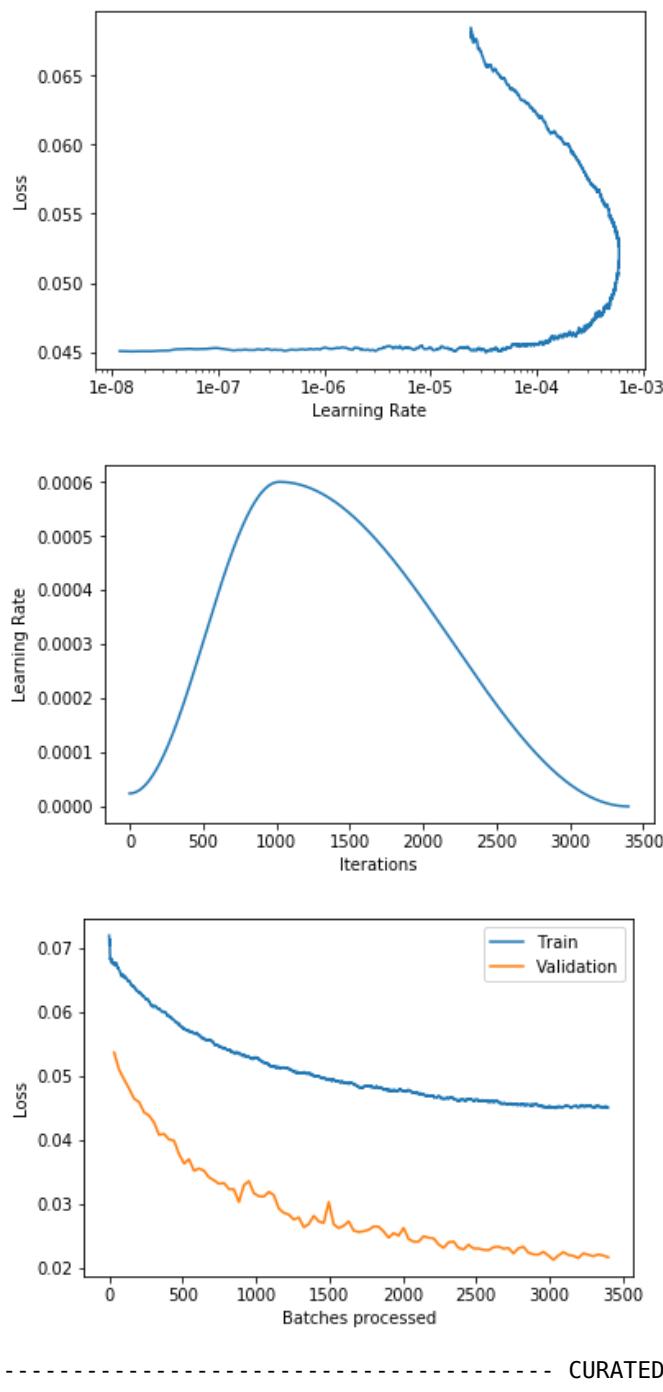
----- CURATED - Fold 5/10 -----

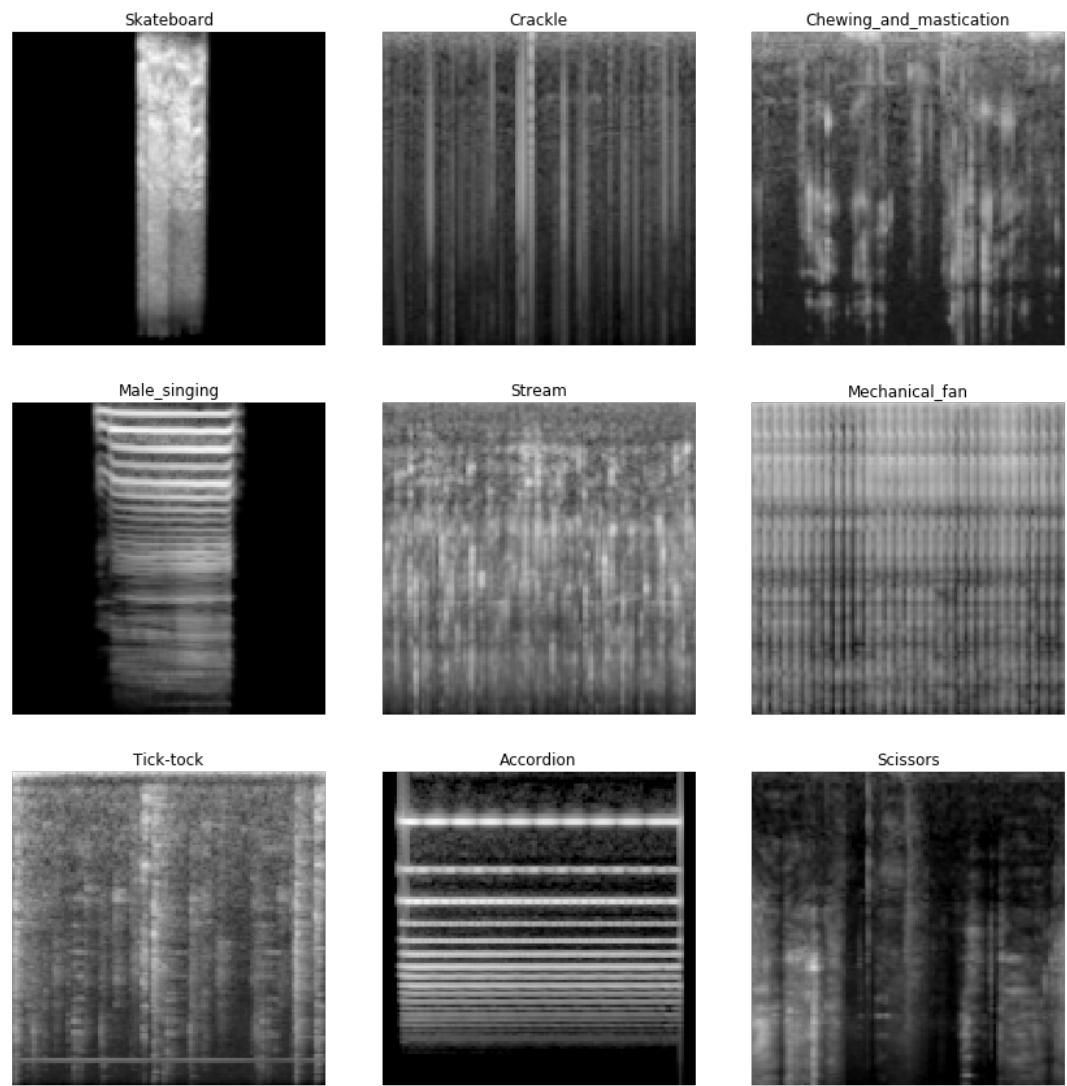


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

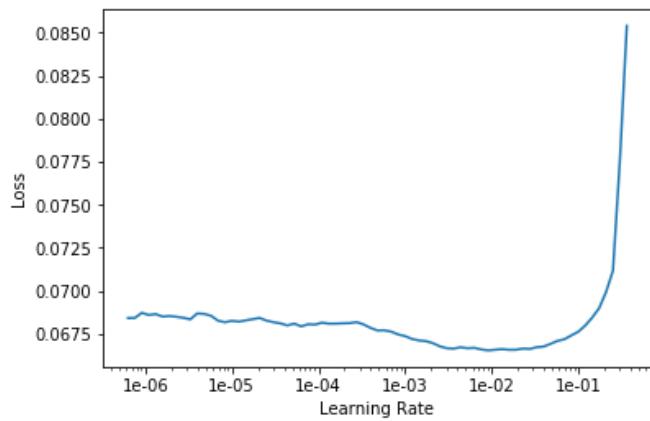


epoch	train_loss	valid_loss	lwlrap	time
0	0.067379	0.053684	0.463618	00:12
1	0.066623	0.050964	0.497225	00:12
2	0.065550	0.049476	0.520282	00:12
3	0.064839	0.048047	0.542487	00:12
4	0.063869	0.046466	0.564797	00:12
5	0.063132	0.045956	0.567790	00:12
6	0.062393	0.044345	0.593893	00:12
7	0.061641	0.043834	0.594500	00:12
8	0.060902	0.042761	0.613727	00:12
9	0.060463	0.040828	0.644771	00:12
10	0.059986	0.040988	0.646575	00:12
11	0.059344	0.040116	0.654888	00:12
12	0.058747	0.039927	0.654299	00:12
13	0.058073	0.037834	0.687511	00:12
14	0.057479	0.036355	0.707246	00:12
15	0.057002	0.037025	0.683751	00:12
16	0.056634	0.035233	0.735909	00:12
17	0.056288	0.035586	0.712633	00:12
18	0.055816	0.035234	0.712691	00:12
19	0.055682	0.034205	0.717561	00:12
20	0.055070	0.033740	0.728541	00:12
21	0.054675	0.033193	0.739078	00:12
22	0.054434	0.033317	0.737014	00:12
23	0.054123	0.032378	0.756964	00:12
24	0.053697	0.032318	0.749551	00:12
25	0.053565	0.030346	0.776979	00:12
26	0.053191	0.032984	0.736294	00:12
27	0.053020	0.033605	0.725037	00:12
28	0.052883	0.031711	0.761934	00:12
29	0.052639	0.031254	0.757741	00:12
30	0.052171	0.031194	0.737503	00:12
31	0.051829	0.031941	0.737323	00:12
32	0.051553	0.031417	0.744646	00:12
33	0.051381	0.029321	0.769588	00:12
34	0.051345	0.028661	0.790009	00:12
35	0.051147	0.028395	0.782875	00:12

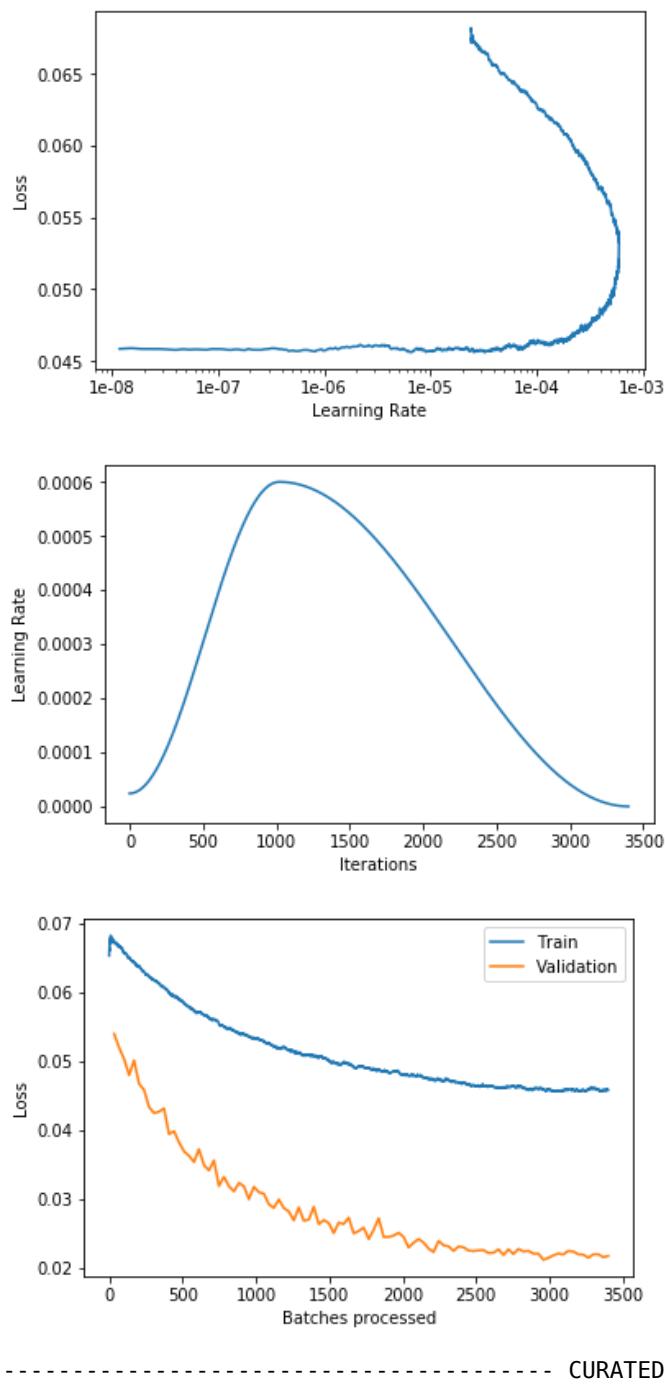




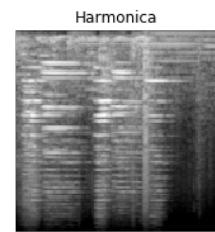
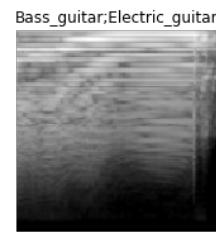
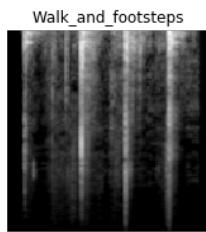
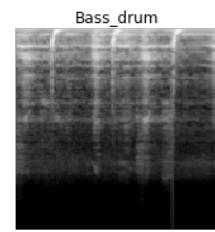
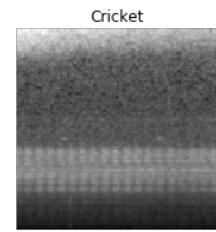
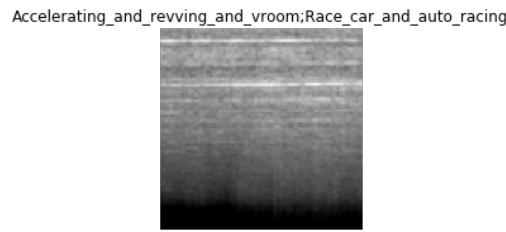
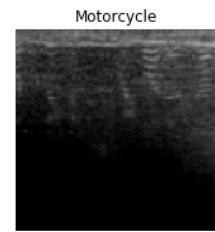
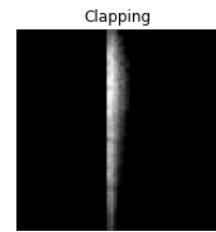
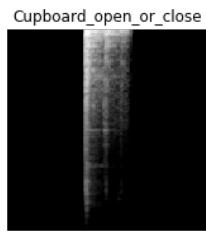
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



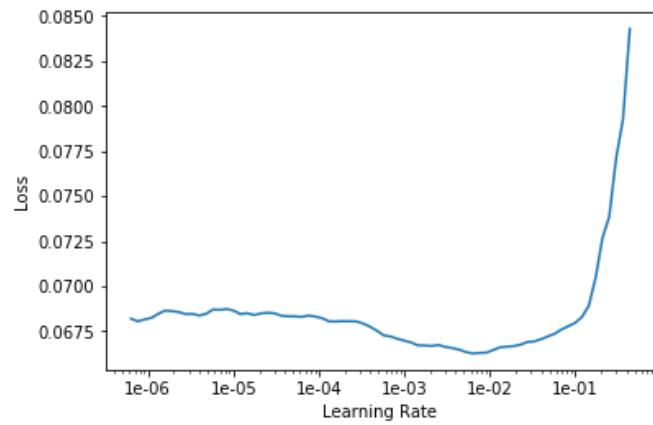
epoch	train_loss	valid_loss	lwlrap	time
0	0.067301	0.053973	0.431562	00:12
1	0.066652	0.051956	0.477210	00:12
2	0.065712	0.050327	0.502150	00:12
3	0.065050	0.047987	0.519688	00:12
4	0.064416	0.050128	0.550971	00:12
5	0.063688	0.046738	0.554768	00:12
6	0.063034	0.045867	0.585704	00:12
7	0.062357	0.043363	0.596514	00:12
8	0.061794	0.042484	0.616526	00:12
9	0.061348	0.042684	0.635755	00:12
10	0.060761	0.043176	0.629712	00:12
11	0.059993	0.039443	0.664958	00:12
12	0.059491	0.039865	0.664758	00:12
13	0.058978	0.038251	0.668487	00:12
14	0.058407	0.036957	0.697624	00:12
15	0.057878	0.036315	0.697003	00:12
16	0.057397	0.035400	0.721230	00:12
17	0.057037	0.037268	0.672375	00:12
18	0.056658	0.034972	0.720866	00:12
19	0.056325	0.034179	0.727566	00:12
20	0.056038	0.035639	0.705547	00:12
21	0.055424	0.031929	0.750967	00:12
22	0.055041	0.033267	0.735299	00:12
23	0.054786	0.031911	0.749071	00:12
24	0.054480	0.031143	0.749986	00:12
25	0.054017	0.032374	0.746096	00:12
26	0.053925	0.031888	0.746166	00:12
27	0.053608	0.030020	0.754654	00:12
28	0.053385	0.031811	0.742988	00:12
29	0.053292	0.031024	0.744825	00:12
30	0.052932	0.030826	0.758339	00:12
31	0.052615	0.029279	0.768482	00:12
32	0.052230	0.028764	0.789749	00:12
33	0.051929	0.029991	0.763823	00:12
34	0.051987	0.028749	0.780719	00:12
35	0.051615	0.028231	0.775719	00:12



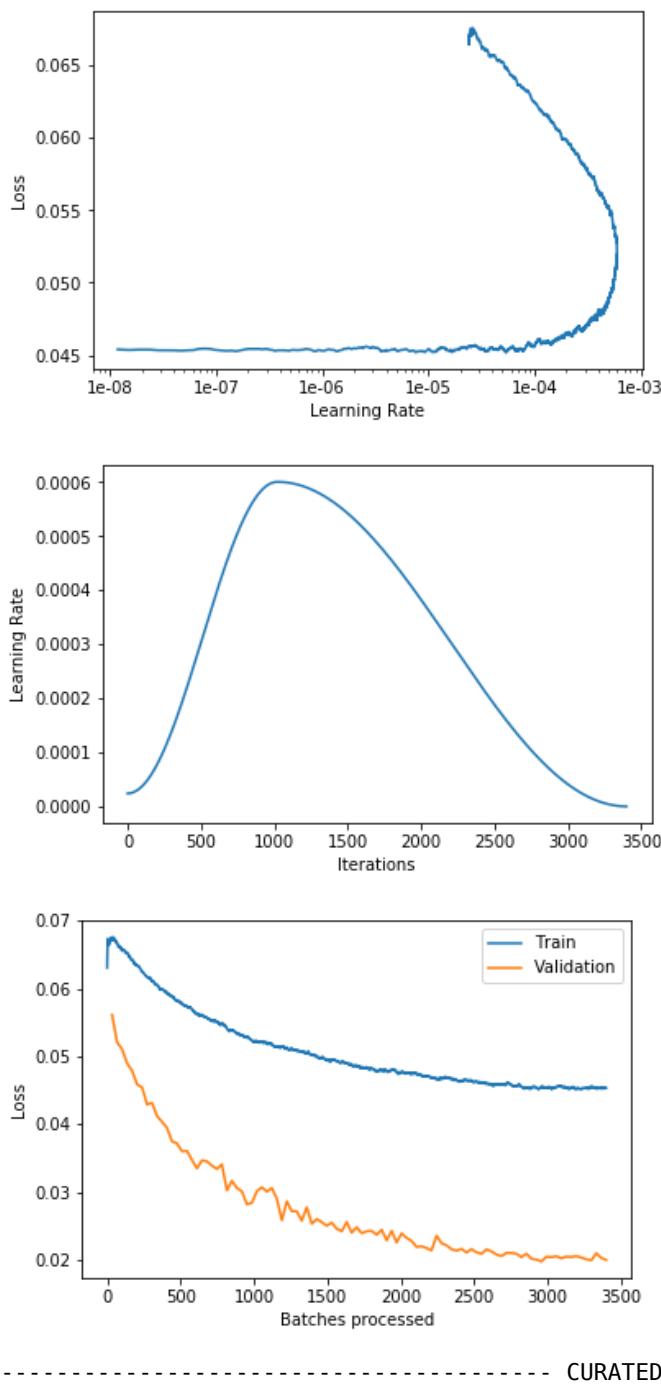
----- CURATED - Fold 7/10 -----



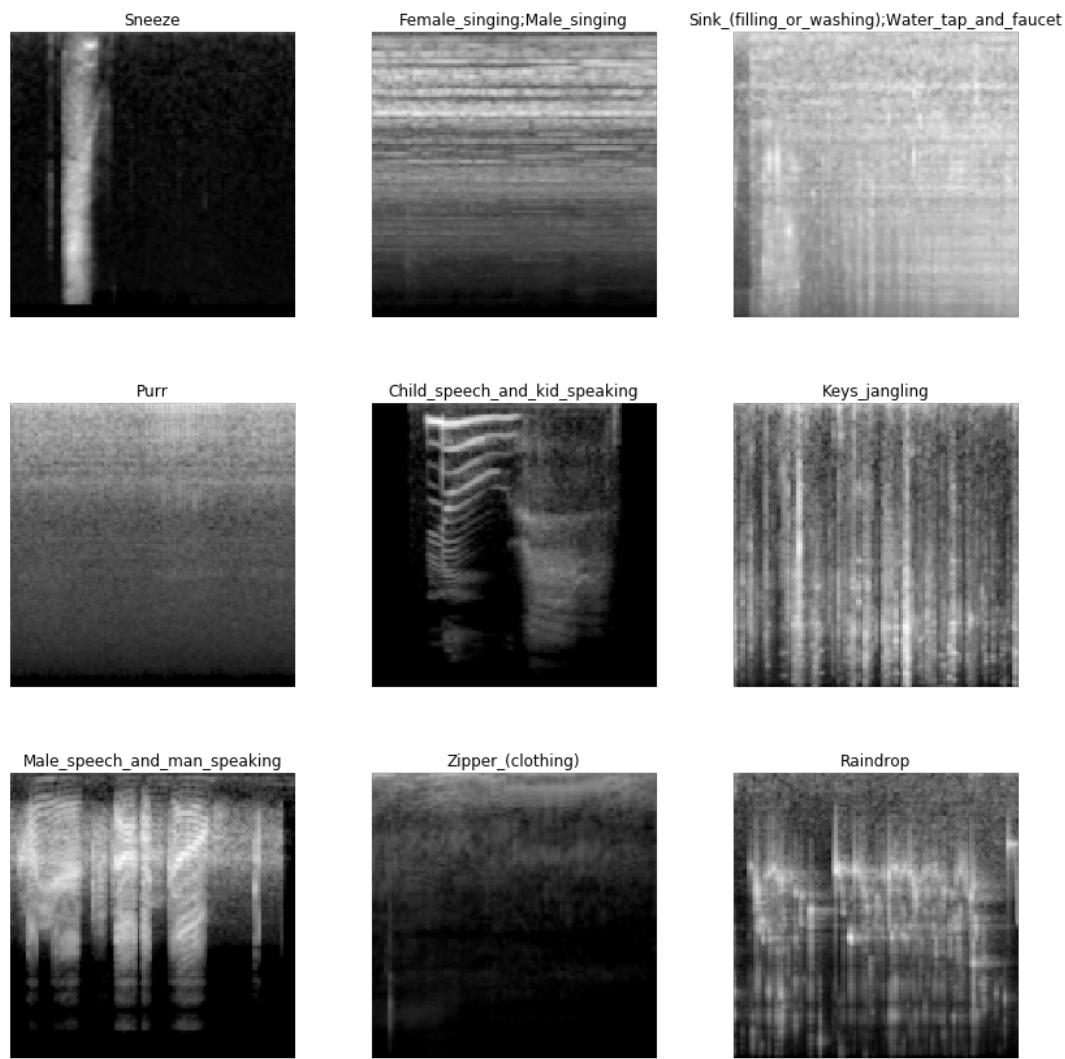
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



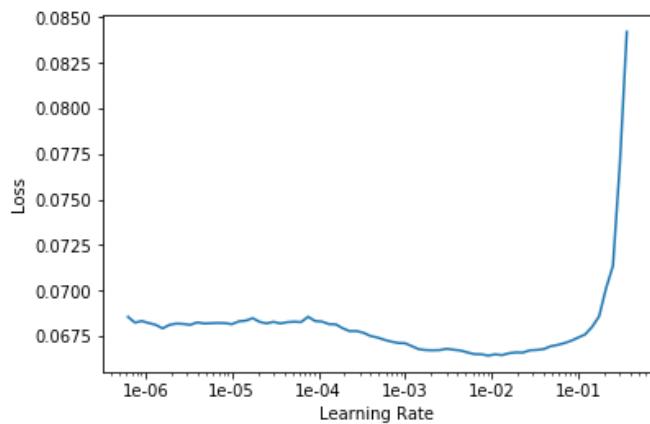
epoch	train_loss	valid_loss	lwlrap	time
0	0.067420	0.056131	0.427939	00:12
1	0.066629	0.052082	0.485151	00:12
2	0.065674	0.051075	0.502828	00:12
3	0.065159	0.048985	0.531556	00:12
4	0.064214	0.047911	0.537168	00:12
5	0.063415	0.045887	0.577013	00:12
6	0.062499	0.045468	0.579926	00:12
7	0.061755	0.042935	0.628369	00:12
8	0.061159	0.043181	0.622760	00:12
9	0.060524	0.041244	0.647976	00:12
10	0.059840	0.040370	0.666970	00:12
11	0.059314	0.039517	0.665481	00:12
12	0.058891	0.037499	0.705835	00:12
13	0.058281	0.037224	0.702397	00:12
14	0.057885	0.036054	0.721385	00:12
15	0.057426	0.036109	0.720035	00:12
16	0.056986	0.034750	0.727095	00:12
17	0.056333	0.033539	0.741492	00:12
18	0.056105	0.034720	0.722532	00:12
19	0.055721	0.034578	0.726123	00:12
20	0.055289	0.033941	0.730104	00:12
21	0.055137	0.033441	0.727710	00:12
22	0.054967	0.034144	0.734094	00:12
23	0.054181	0.030305	0.766947	00:12
24	0.053865	0.031703	0.736491	00:12
25	0.053456	0.030650	0.759275	00:12
26	0.053149	0.030156	0.763001	00:12
27	0.052831	0.028253	0.781854	00:12
28	0.052420	0.028460	0.776841	00:12
29	0.052201	0.030193	0.763407	00:12
30	0.052182	0.030756	0.740170	00:12
31	0.051991	0.030119	0.756477	00:12
32	0.051884	0.030648	0.754996	00:12
33	0.051497	0.029099	0.774056	00:12
34	0.051489	0.025913	0.798468	00:12
35	0.051327	0.028686	0.782019	00:12



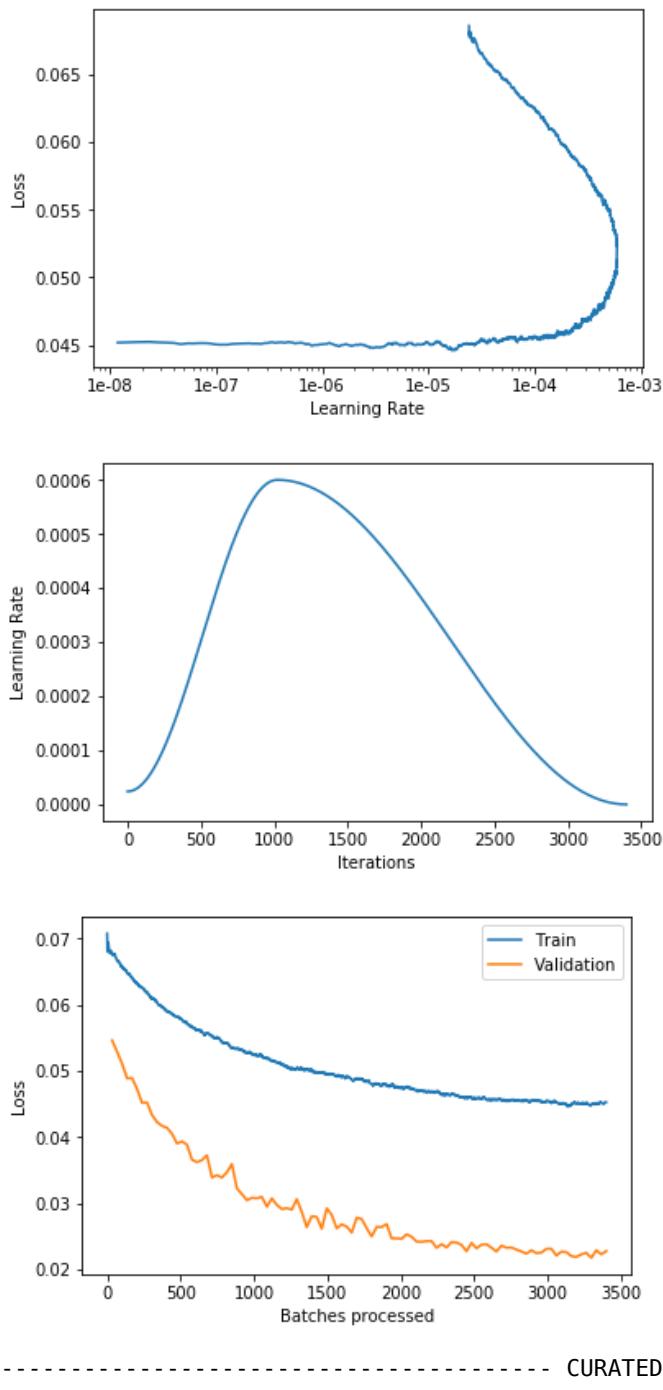
----- CURATED - Fold 8/10 -----

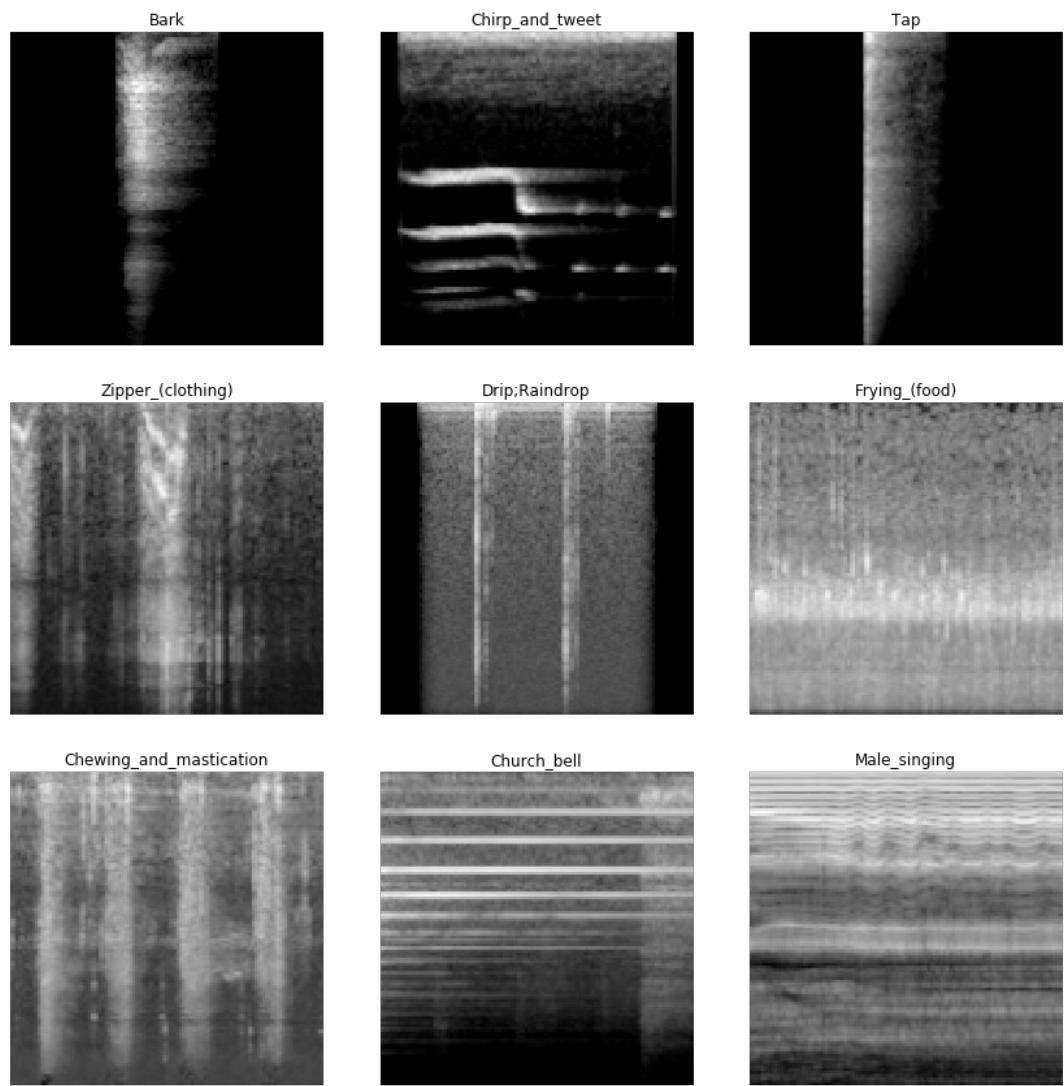


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

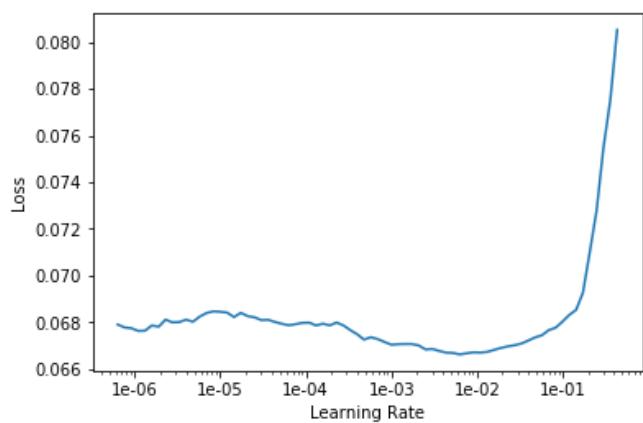


epoch	train_loss	valid_loss	lwlrap	time
0	0.067760	0.054598	0.425824	00:12
1	0.066760	0.052859	0.456503	00:12
2	0.065708	0.051067	0.489030	00:12
3	0.064870	0.048847	0.506841	00:12
4	0.064063	0.048893	0.513656	00:12
5	0.063266	0.047227	0.535575	00:12
6	0.062612	0.045146	0.574690	00:12
7	0.061758	0.045190	0.571779	00:12
8	0.060996	0.043316	0.603782	00:12
9	0.060376	0.042226	0.614534	00:12
10	0.059666	0.041665	0.626010	00:12
11	0.058975	0.041376	0.625897	00:12
12	0.058642	0.040473	0.633825	00:12
13	0.058223	0.039006	0.661106	00:12
14	0.057681	0.039320	0.664807	00:12
15	0.057268	0.038821	0.662222	00:12
16	0.056587	0.036559	0.697805	00:12
17	0.056225	0.036177	0.693916	00:12
18	0.055903	0.036501	0.688402	00:12
19	0.055616	0.037223	0.688462	00:12
20	0.055234	0.033855	0.728290	00:12
21	0.055031	0.034231	0.716283	00:12
22	0.054366	0.033852	0.732127	00:12
23	0.054104	0.034672	0.713442	00:12
24	0.053532	0.035899	0.699940	00:12
25	0.053426	0.032226	0.731442	00:12
26	0.053135	0.031359	0.757330	00:12
27	0.052806	0.030430	0.755052	00:12
28	0.052626	0.030800	0.755109	00:12
29	0.052356	0.030724	0.752321	00:12
30	0.052289	0.030932	0.754108	00:12
31	0.051798	0.029444	0.778844	00:12
32	0.051565	0.030719	0.764912	00:12
33	0.051246	0.029720	0.759493	00:12
34	0.050922	0.029105	0.767005	00:12
35	0.050601	0.029245	0.773270	00:12

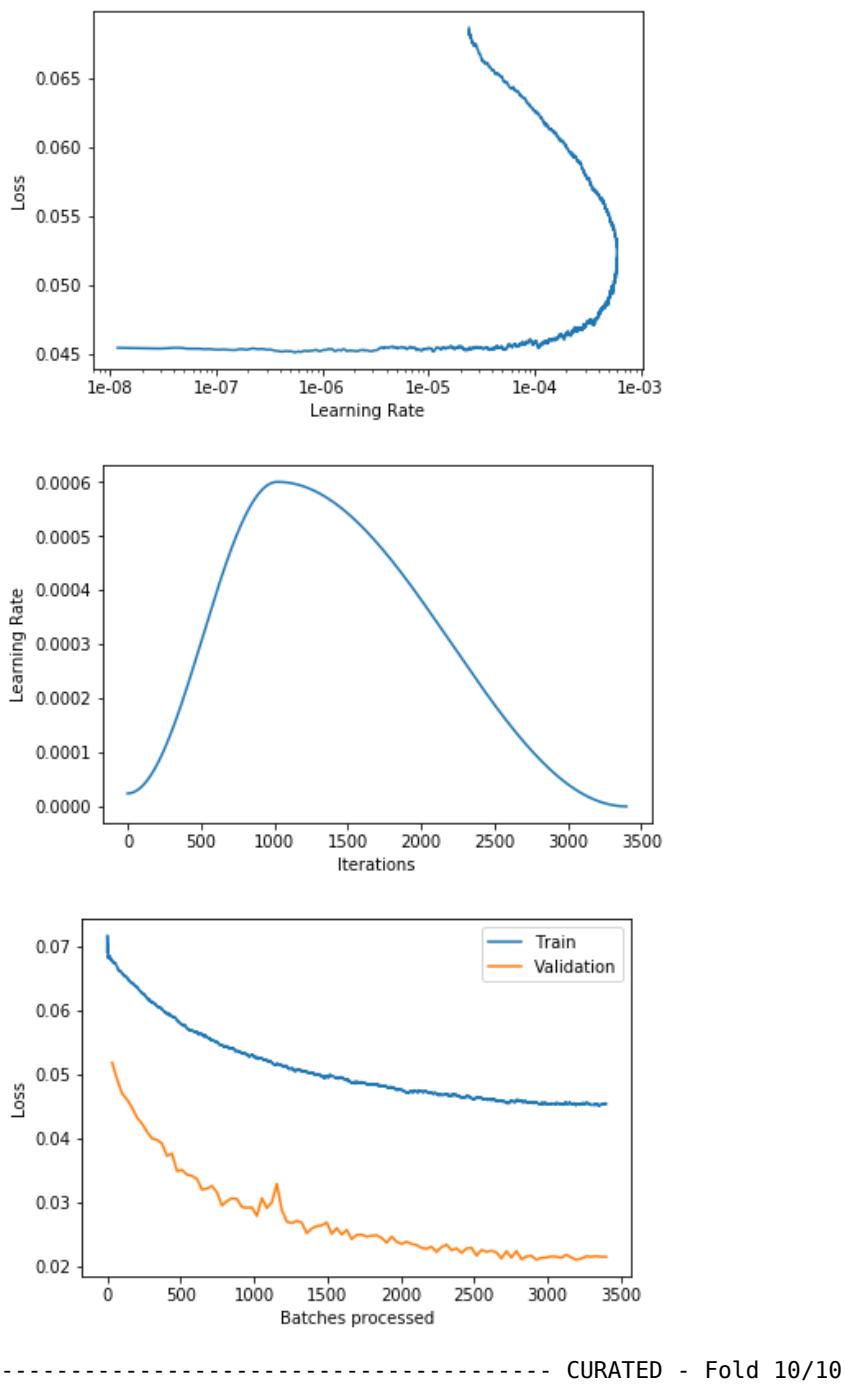


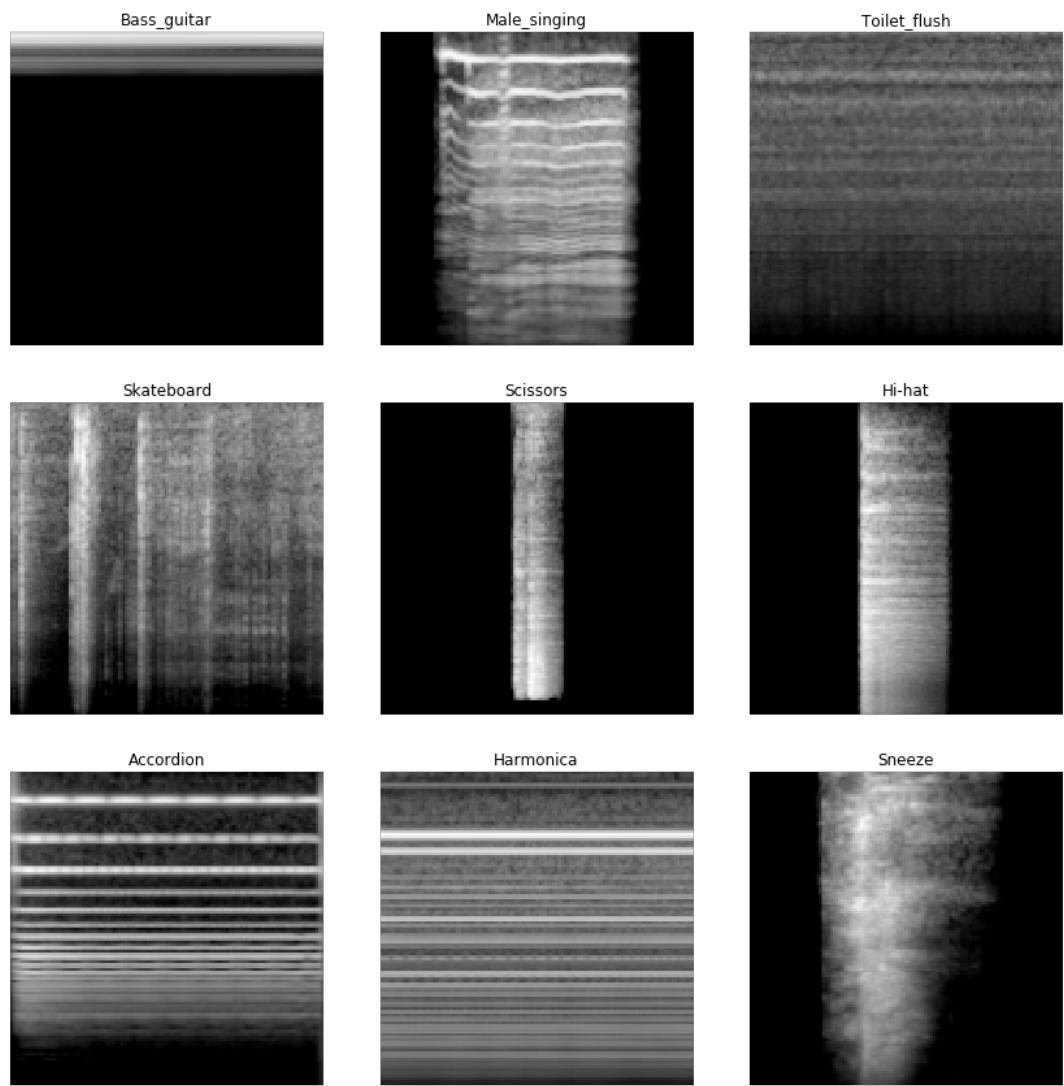


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

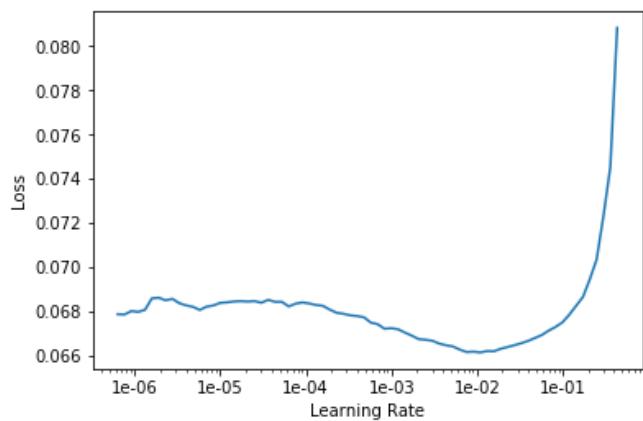


epoch	train_loss	valid_loss	lwlrap	time
0	0.067714	0.051807	0.472770	00:12
1	0.066761	0.049183	0.499797	00:12
2	0.065831	0.046970	0.533592	00:12
3	0.065099	0.046018	0.543729	00:12
4	0.064245	0.044742	0.562696	00:12
5	0.063584	0.043204	0.593512	00:12
6	0.062717	0.042229	0.600038	00:12
7	0.061972	0.040986	0.637766	00:12
8	0.061290	0.039939	0.653938	00:12
9	0.060705	0.039719	0.649138	00:12
10	0.060202	0.039138	0.660192	00:12
11	0.059488	0.037203	0.687110	00:12
12	0.059045	0.037581	0.696899	00:12
13	0.058398	0.034823	0.721076	00:12
14	0.057717	0.035021	0.726608	00:12
15	0.057176	0.034279	0.726217	00:12
16	0.056862	0.034071	0.741885	00:12
17	0.056540	0.033617	0.742465	00:12
18	0.056231	0.031934	0.756782	00:12
19	0.055817	0.032114	0.739935	00:12
20	0.055321	0.032515	0.738827	00:12
21	0.055033	0.031529	0.747198	00:12
22	0.054575	0.029437	0.765728	00:12
23	0.054311	0.030134	0.757509	00:12
24	0.054045	0.030579	0.772555	00:12
25	0.053729	0.030488	0.756076	00:12
26	0.053323	0.029225	0.766316	00:12
27	0.053136	0.029099	0.771153	00:12
28	0.052946	0.029167	0.771900	00:12
29	0.052576	0.027834	0.794479	00:12
30	0.052433	0.030603	0.748599	00:12
31	0.052230	0.029062	0.771585	00:12
32	0.051878	0.029953	0.760249	00:12
33	0.051634	0.032843	0.717843	00:12
34	0.051366	0.028753	0.771557	00:12
35	0.051201	0.026939	0.780267	00:12

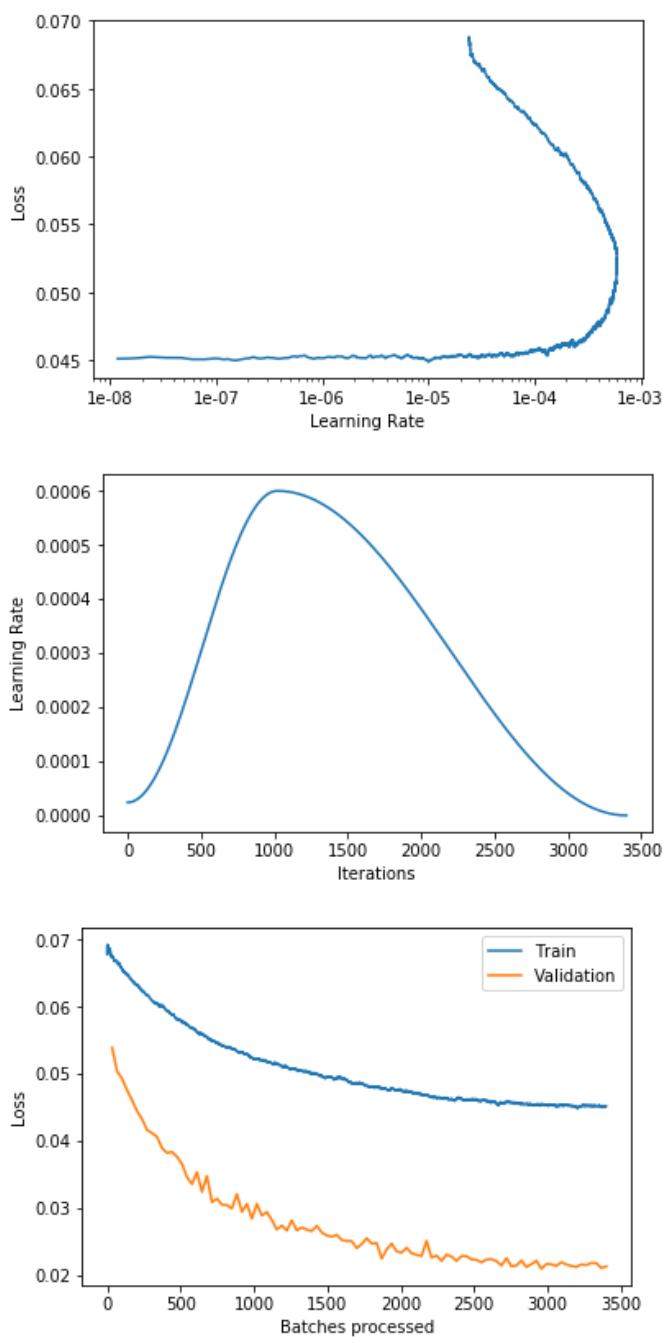




LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.067515	0.053916	0.446297	00:12
1	0.066780	0.050378	0.486182	00:12
2	0.065609	0.049290	0.518452	00:12
3	0.064861	0.047565	0.535258	00:12
4	0.063998	0.046075	0.564703	00:12
5	0.063380	0.044433	0.587063	00:12
6	0.062411	0.043256	0.594765	00:12
7	0.061697	0.041633	0.629536	00:12
8	0.060963	0.041168	0.638949	00:12
9	0.060602	0.040609	0.646001	00:12
10	0.060162	0.038942	0.657528	00:12
11	0.059458	0.038195	0.669332	00:12
12	0.058800	0.038350	0.672052	00:12
13	0.058237	0.037652	0.676540	00:12
14	0.057745	0.036562	0.689464	00:12
15	0.057315	0.034590	0.722940	00:12
16	0.056869	0.033567	0.736698	00:12
17	0.056337	0.035343	0.700830	00:12
18	0.055935	0.032392	0.735711	00:12
19	0.055474	0.034776	0.712869	00:12
20	0.055040	0.030850	0.767846	00:12
21	0.054555	0.031393	0.748229	00:12
22	0.054268	0.030484	0.756549	00:12
23	0.053992	0.030426	0.749600	00:12
24	0.053801	0.029904	0.767092	00:12
25	0.053446	0.032057	0.738000	00:12
26	0.053312	0.029434	0.774700	00:12
27	0.052967	0.030598	0.762068	00:12
28	0.052537	0.028422	0.788072	00:12
29	0.052209	0.030606	0.770626	00:12
30	0.052063	0.028853	0.779306	00:12
31	0.051874	0.029393	0.778538	00:12
32	0.051561	0.028197	0.782966	00:12
33	0.051416	0.026772	0.811054	00:12
34	0.051169	0.027368	0.789426	00:12
35	0.051108	0.026617	0.786683	00:12



```
In [27]: kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-2_fold-{fold}')
```

Overall lwlrap on CURATED dataset: 0.8754269930630387

	lwlrap	weight
Fill_(with_liquid)	0.590342	0.008693
Squeak	0.637303	0.013039
Walk_and_footsteps	0.699404	0.013039
Tap	0.704021	0.013039
Hiss	0.708685	0.013039
Mechanical_fan	0.735838	0.008519
Chink_and_clink	0.765375	0.013039
Bus	0.768763	0.013039
Trickle_and_dribble	0.771013	0.009214
Cutlery_and_silverware	0.781103	0.013039
Buzz	0.783759	0.009736
Traffic_noise_and_roadway_noise	0.791878	0.013039
Accelerating_and_revving_and_vroom	0.794537	0.013039
Microwave_oven	0.794794	0.013039
Water_tap_and_faucet	0.805175	0.013039
Sink_(filling_or_washing)	0.806955	0.013039
Motorcycle	0.815111	0.013039
Male_speech_and_manSpeaking	0.821809	0.013039
Scissors	0.823143	0.013039
Dishes_and_pots_and_pans	0.826769	0.013039
Yell	0.827063	0.013039
Clapping	0.830222	0.013039
Cupboard_open_or_close	0.843187	0.013039
Gurgling	0.845174	0.013039
Drip	0.846728	0.013039
Gasp	0.848495	0.008345
Car_passing_by	0.851683	0.013039
Slam	0.852635	0.013039
Bathtub_(filling_or_washing)	0.854349	0.013039
Chirp_and_tweet	0.856576	0.013039
...	...	...
Female_speech_and_womanSpeaking	0.912511	0.013039
Drawer_open_or_close	0.919778	0.013039
Bicycle_bell	0.921473	0.011648
Crackle	0.922804	0.013039

```
In [28]: kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-2_fold-{fold}')
```

Overall lwlrapp on NOISY dataset: 0.37736108866147494

	<b>lwlrap</b>	<b>weight</b>
Cupboard_open_or_close	0.057677	0.0125
Raindrop	0.062900	0.0125
Tap	0.064542	0.0125
Finger_snapping	0.072714	0.0125
Sigh	0.091557	0.0125
Burping_and_eructation	0.093851	0.0125
Keys_jangling	0.099547	0.0125
Shatter	0.101200	0.0125
Gasp	0.113620	0.0125
Drip	0.117029	0.0125
Knock	0.129898	0.0125
Fart	0.132301	0.0125
Slam	0.142294	0.0125
Strum	0.173150	0.0125
Bicycle_bell	0.173218	0.0125
Trickle_and_dribble	0.175289	0.0125
Glockenspiel	0.182888	0.0125
Chink_and_clink	0.206992	0.0125
Computer_keyboard	0.207558	0.0125
Writing	0.209184	0.0125
Dishes_and_pots_and_pans	0.209288	0.0125
Run	0.210554	0.0125
Scissors	0.211626	0.0125
Buzz	0.212720	0.0125
Cutlery_and_silverware	0.225744	0.0125
Zipper_(clothing)	0.233319	0.0125
Whispering	0.233408	0.0125
Fill_(with_liquid)	0.239296	0.0125
Gong	0.240580	0.0125
Yell	0.241539	0.0125
...	...	...
Screaming	0.464641	0.0125
Accelerating_and_revving_and_vroom	0.476415	0.0125
Electric_guitar	0.494699	0.0125
Frying_(food)	0.500268	0.0125

## Look for useful noisy samples

```
In [29]: def lwlrap_per_element(fname):
    overall_preds, overall_thruth, overall_index = _kfold_prediction(kf_noisy,
trn_noisy_df, fname)

    m = np.zeros_like(overall_preds)
    for i, (p, t) in enumerate(zip(overall_preds, overall_thruth)):
        idx, val = _one_sample_positive_class_precisions(p, t)
        m[i, idx] = val
    m = pd.DataFrame(m, columns=learn.data.classes, index=overall_index)

    m['fname'] = trn_noisy_df.fname

    for fold, (train_index, valid_index) in enumerate(kf_noisy.split(trn_noisy_
df)):
        m.at[valid_index, 'fold'] = fold

    return m
```

```
In [30]: m1 = lwlrap_per_element('stage-1_fold-{fold}')
```

```
In [31]: m2 = lwlrap_per_element('stage-2_fold-{fold}')
```

```
In [32]: m = m1.copy()
m[learn.data.classes] = np.sqrt(m1[learn.data.classes] * m2[learn.data.classe
s])
```

```
In [33]: del m1, m2
gc.collect();
```

```
In [34]: # Select samples from noisy dataset that look promising:  
# - suppose correct label from organiser  
# - high lwlrap on stage 1 (model with noisy elements only)  
# - high lwlrap on stage 2 (model warmed up with noisy elements and then fine tuned with curated dataset)  
  
lwlrap_threshold = .5  
count_limit = 5 # per fold and per class  
  
ok_noisy_items = []  
  
for i, x in enumerate(learn.data.classes):  
    print('\n', '*' * 20, x, '*' * 20)  
    print('Found', m[(m[x] >= lwlrap_threshold)].shape[0], 'promising element(s)')  
    for fold in range(n_splits):  
        selected = m[(m[x] >= lwlrap_threshold) & (m.fold == fold)].sort_values(x, ascending=False)[:count_limit].index  
        ok_noisy_items.extend(selected)  
        print('fold ', fold, ':', m[(m[x] >= lwlrap_threshold) & (m.fold == fold)].shape[0], m.iloc[selected].fname.tolist())  
  
print('\nFinal selection includes', len(ok_noisy_items), '"noisy" items')
```

```
***** Accelerating_and_revving_and_vroom *****
Found 157 promising element(s)
fold 0 : 16 ['a4435a9d.wav', 'bddcf02f.wav', 'd58e1919.wav', 'ae1249ca.wav', 'f242a86a.wav']
fold 1 : 14 ['f682f18c.wav', 'c092764a.wav', '3c2f3122.wav', '87fac7cf.wav', 'bc3892c1.wav']
fold 2 : 9 ['ffd1c0e2.wav', '8119602a.wav', '40a928b7.wav', '7160d0e1.wav', 'e445db5e.wav']
fold 3 : 18 ['a0da98c4.wav', '33ab75d8.wav', 'b84be9e3.wav', 'fae6e24e.wav', '9e874ee3.wav']
fold 4 : 19 ['f12f24da.wav', '4ef17f00.wav', 'a2ae680a.wav', 'e79da000.wav', '26bdbf2b.wav']
fold 5 : 17 ['ab1c9211.wav', 'ab5cca21.wav', '73bfc028.wav', '413f17c4.wav', 'e4298185.wav']
fold 6 : 16 ['0af6c5e5.wav', '32f39d14.wav', 'e5165478.wav', '1924aa99.wav', 'bc69bdfe.wav']
fold 7 : 19 ['aa2c9029.wav', 'cc41013a.wav', 'b8b1ee59.wav', '49034bb5.wav', '88c633c1.wav']
fold 8 : 12 ['77b14052.wav', '232899b2.wav', '2872af5d.wav', '6f27c5ef.wav', 'f13ae3d3.wav']
fold 9 : 17 ['a6631a37.wav', 'f683db6f.wav', '361ca0f8.wav', '683b072f.wav', '636164f0.wav']

***** Accordion *****
Found 218 promising element(s)
fold 0 : 22 ['2d876fd7.wav', '212a39e6.wav', 'e247cbd5.wav', '9ebd56b4.wav', 'a11f262f.wav']
fold 1 : 20 ['a274cb3a.wav', '54a843c4.wav', 'd14c6424.wav', '26ada8a4.wav', '20930ddf.wav']
fold 2 : 26 ['ec1de3c0.wav', '21e03976.wav', 'c0aafe6b.wav', 'bf146d39.wav', 'a480c194.wav']
fold 3 : 21 ['3605b40b.wav', 'b414e5a3.wav', 'd375b221.wav', '65de088e.wav', '0ff87b12.wav']
fold 4 : 25 ['0838b0b3.wav', 'b6acd265.wav', '32273484.wav', '56c3ebbe.wav', 'e68ce1f3.wav']
fold 5 : 20 ['07c14838.wav', 'b2de494b.wav', 'dbaaef3b.wav', 'a63fb07e.wav', '36d966be.wav']
fold 6 : 24 ['593c85d7.wav', 'd00893e2.wav', 'aea8a786.wav', '0c7bf9a1.wav', 'e2e8e544.wav']
fold 7 : 17 ['b863cb3f.wav', '6a9830e1.wav', '34364fff.wav', 'b019a385.wav', 'a75a71e3.wav']
fold 8 : 28 ['35d6ea77.wav', '6102fb7a.wav', 'd92c1a99.wav', '9c4199ce.wav', '29dc9519.wav']
fold 9 : 15 ['6e5ea69a.wav', 'ac9a944b.wav', '6c2622ca.wav', 'c554e5d0.wav', 'bc420613.wav']

***** Acoustic_guitar *****
Found 232 promising element(s)
fold 0 : 29 ['60d25862.wav', 'bc450168.wav', '971a8507.wav', '771f6fa2.wav', '3e5bca4c.wav']
fold 1 : 23 ['a9b71766.wav', '5367af5c.wav', 'c2dfb020.wav', 'fabab3e5.wav', '4af48f78.wav']
fold 2 : 21 ['1beab2e8.wav', 'd699e544.wav', '839829ea.wav', '024cefba.wav', '65f28dc5.wav']
fold 3 : 31 ['0a883fd0.wav', 'e6d98d8f.wav', '77149899.wav', '6cf1e57d.wav', 'aab1db91.wav']
fold 4 : 25 ['6408beed.wav', '9b8fecd3.wav', '6eb972c0.wav', '400d1f0f.wav', '2f2af21e.wav']
fold 5 : 18 ['9a5927fc.wav', '19ff3d96.wav', 'c552125e.wav', 'f3801075.wav', '558b2017.wav']
fold 6 : 23 ['f17c5ffe.wav', '6ce7d591.wav', 'a7c50a1b.wav', 'be2d26da.wav', 'e62a6277.wav']
fold 7 : 20 ['6ab54a9d.wav', 'abb1656a.wav', '7adf07e2.wav', 'df790c7d.wav', '
```

## Train on Selected Noisy + Curated

```
In [35]: for fold, ((train_index1, valid_index1), (train_index2, valid_index2)) in enumerate(zip(kf_curated.split(trn_curated_df), kf_noisy.split(trn_noisy_df))):
    print('-' * 40, f'CURATED+NOISY - Fold {fold+1}/{n_splits}', '-' * 40)

    train_index2 = list(set(train_index2).intersection(set(ok_noisy_items)))

    mix_df = pd.concat([
        trn_curated_df.iloc[train_index1],
        trn_noisy_df.iloc[train_index2],
        trn_curated_df.iloc[valid_index1],
#        trn_noisy_df.iloc[valid_index2], # compute lwlrap only on curated
    ], ignore_index=True)
    train_index = mix_df[:len(train_index1)+len(train_index2)].index
    valid_index = mix_df[len(train_index1)+len(train_index2):].index

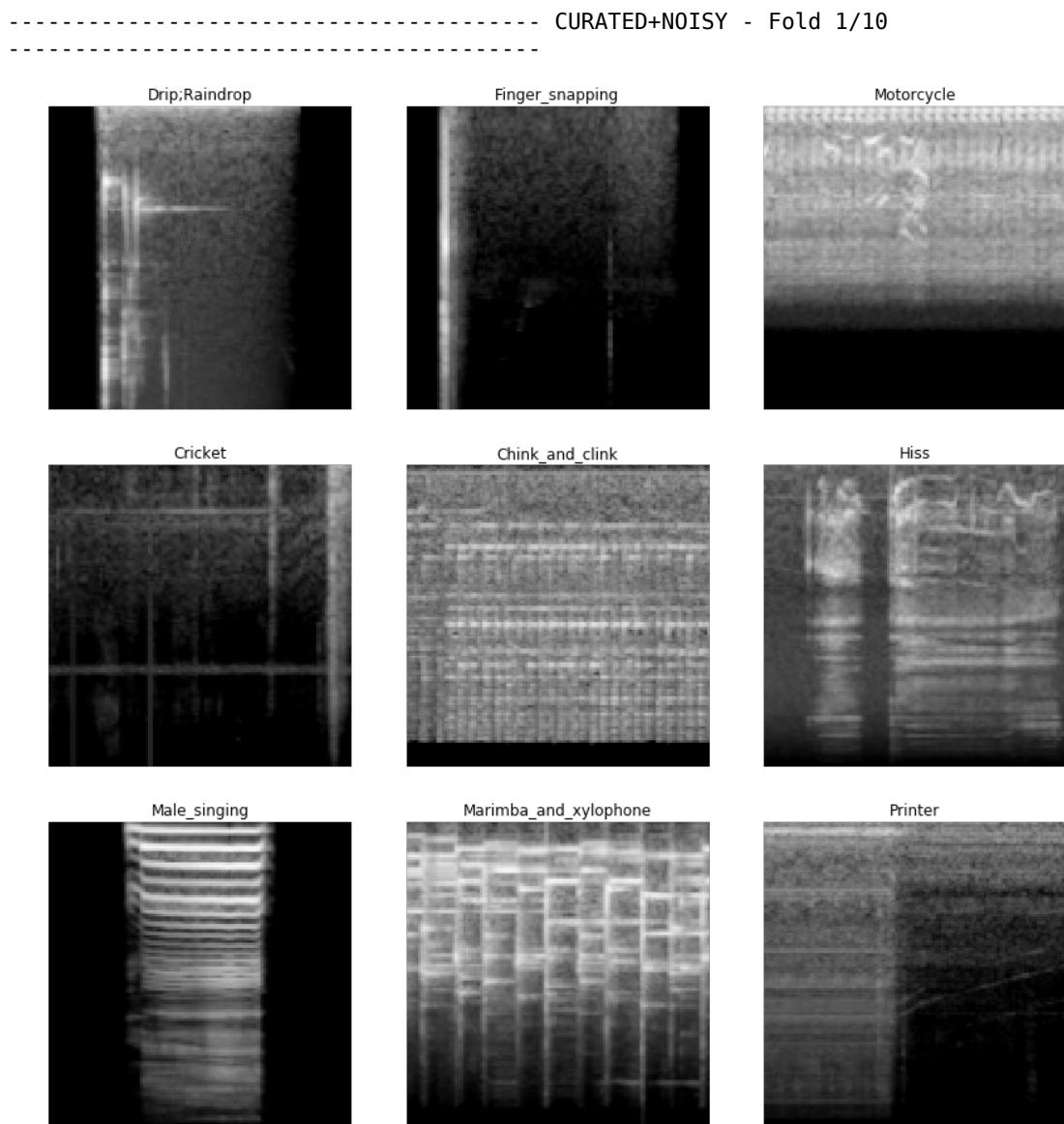
    src = (ImageList.from_df(mix_df, WORK)
           .split_by_idxs(train_index, valid_index)
           .label_from_df(label_delim=','))
    data = (src.transform(tfms, size=128)
            .databunch(bs=bs))
    data.show_batch(3)
    plt.show()

    learn.load(f'stage-2_fold-{fold}')
    learn.data = data

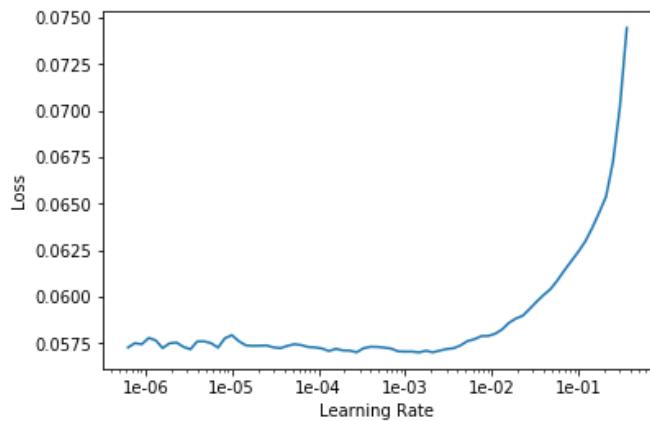
#    learning(learn, 100, 1e-2)
    learning(learn, 300, 2e-3)

    learn.save(f'stage-10_fold-{fold}')
    learn.export(f'stage-10_fold-{fold}.pkl')

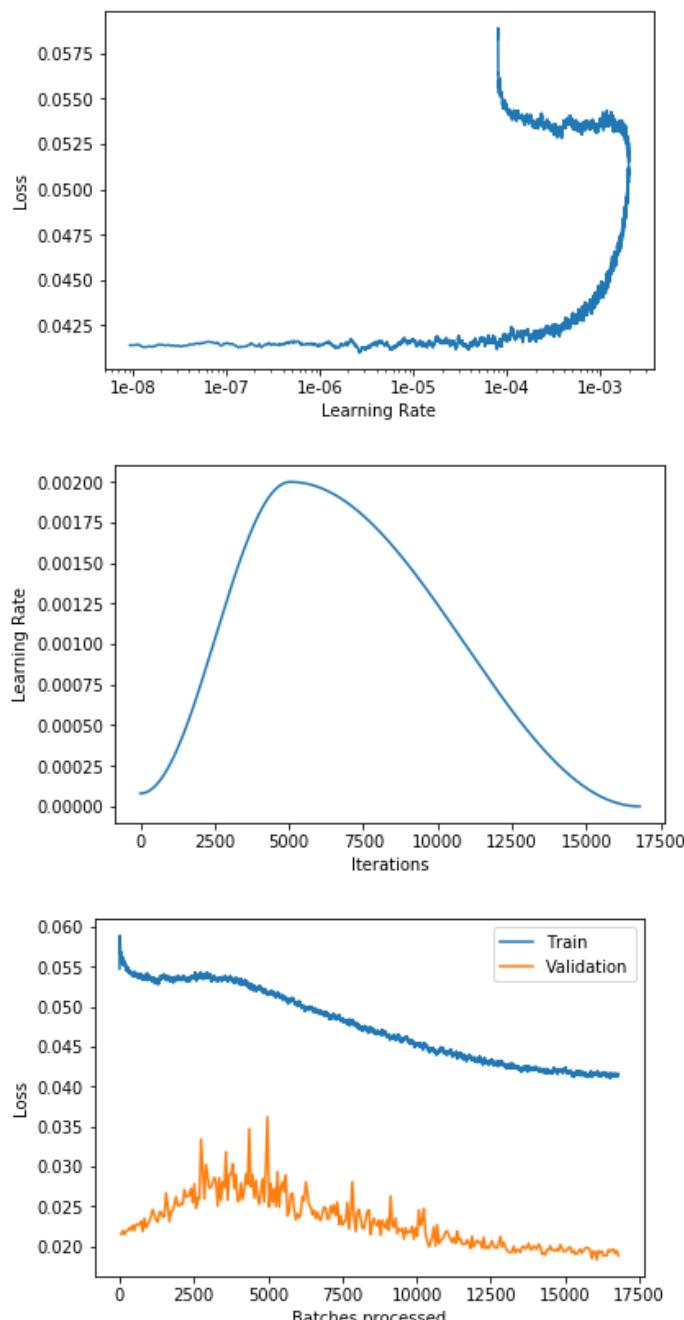
    if TOY_MODE:
        break
```



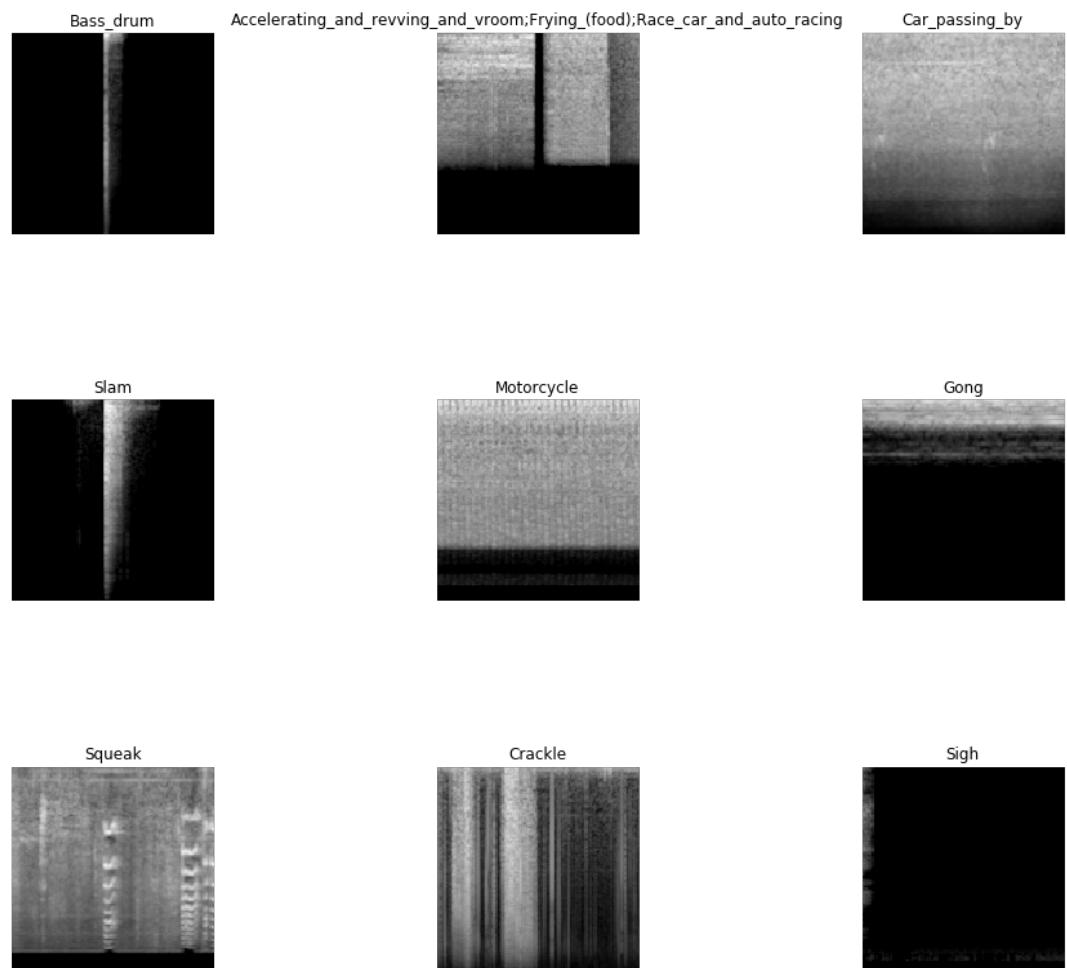
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



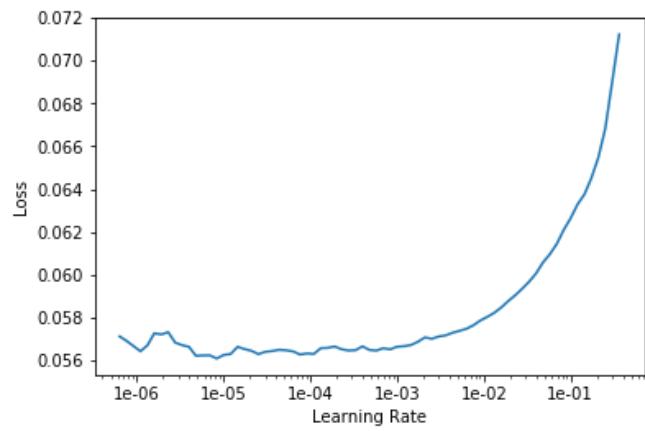
epoch	train_loss	valid_loss	lwlrap	time
0	0.055936	0.021486	0.842826	00:18
1	0.055619	0.021942	0.833198	00:18
2	0.055211	0.021491	0.849169	00:18
3	0.054815	0.021900	0.836108	00:18
4	0.054688	0.021939	0.838386	00:18
5	0.054321	0.022315	0.831652	00:18
6	0.054169	0.022045	0.840828	00:18
7	0.054192	0.022625	0.831054	00:18
8	0.054168	0.021970	0.846166	00:18
9	0.053895	0.022710	0.829208	00:18
10	0.053987	0.022762	0.826620	00:18
11	0.054084	0.022662	0.833203	00:18
12	0.054031	0.023112	0.826553	00:18
13	0.053556	0.021842	0.843506	00:18
14	0.053697	0.023479	0.820604	00:18
15	0.053840	0.022334	0.826226	00:18
16	0.053406	0.022760	0.830151	00:18
17	0.053711	0.023128	0.821373	00:18
18	0.053541	0.023860	0.815841	00:18
19	0.053428	0.024583	0.816040	00:18
20	0.053461	0.023867	0.811233	00:18
21	0.053277	0.023255	0.825196	00:18
22	0.053287	0.023016	0.825171	00:18
23	0.053566	0.024268	0.811865	00:18
24	0.053490	0.024252	0.800785	00:18
25	0.053882	0.023649	0.829865	00:18
26	0.053850	0.023744	0.831209	00:18
27	0.053721	0.026656	0.789778	00:18
28	0.053735	0.025087	0.799787	00:18
29	0.053489	0.023093	0.826898	00:18
30	0.053655	0.024286	0.816171	00:18
31	0.053587	0.024436	0.816510	00:18
32	0.053620	0.025222	0.806639	00:18
33	0.053585	0.026185	0.789195	00:18
34	0.053637	0.025152	0.807837	00:18
35	0.053673	0.025031	0.801315	00:18



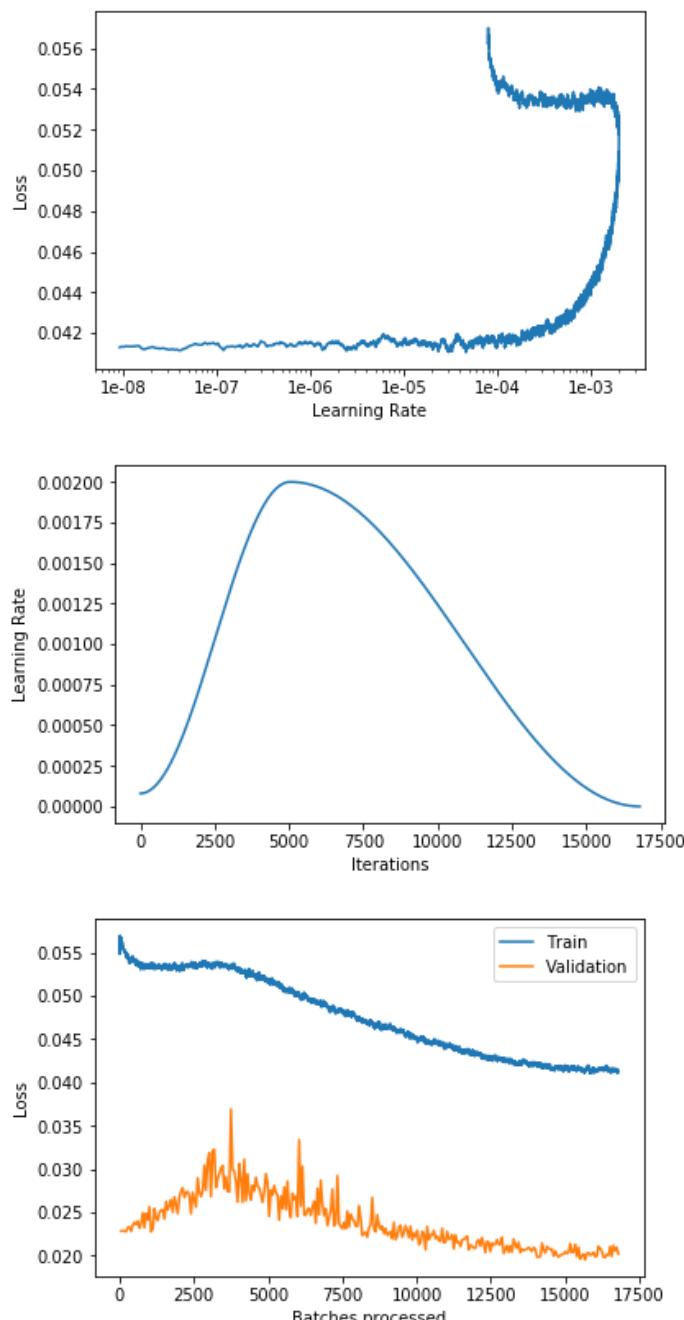
----- CURATED+NOISY - Fold 2/10 -----



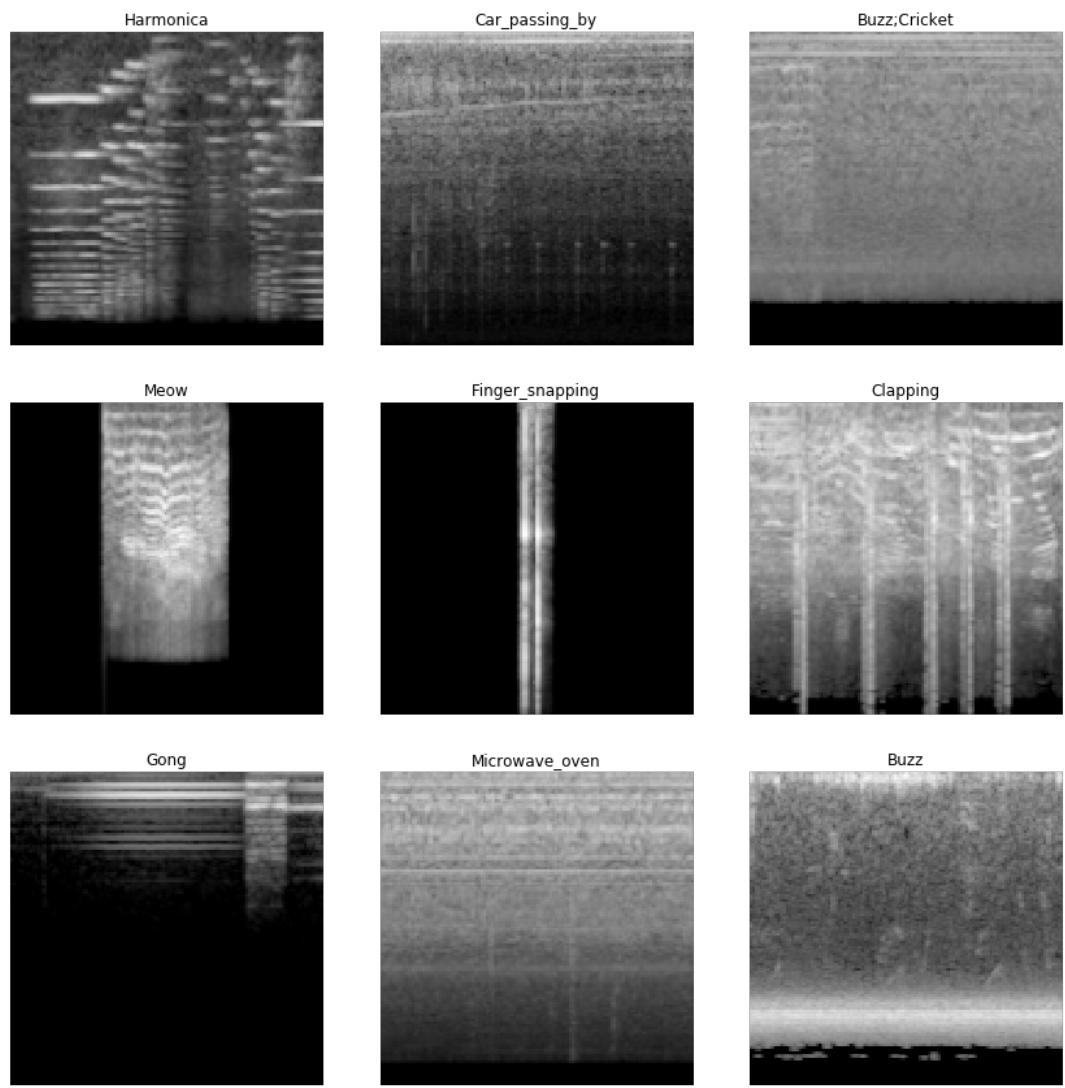
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



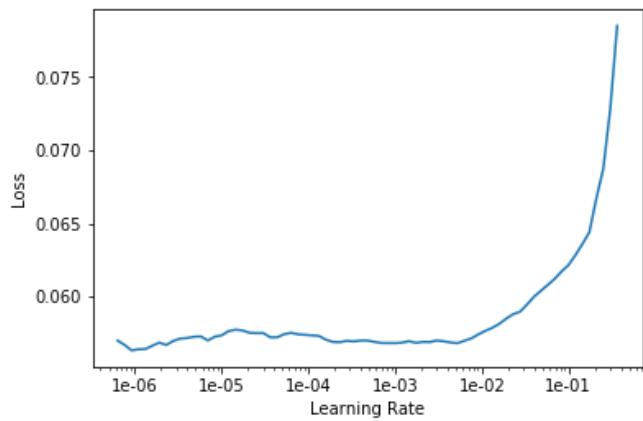
epoch	train_loss	valid_loss	lwlrap	time
0	0.056675	0.022811	0.821578	00:18
1	0.055601	0.022851	0.828076	00:18
2	0.055161	0.022786	0.833821	00:18
3	0.054793	0.022760	0.840814	00:18
4	0.054513	0.023243	0.828458	00:18
5	0.054357	0.023276	0.826105	00:18
6	0.054176	0.022794	0.837813	00:18
7	0.054097	0.023511	0.840875	00:18
8	0.054003	0.023821	0.820588	00:18
9	0.053884	0.023558	0.836191	00:18
10	0.053588	0.023969	0.827862	00:18
11	0.053690	0.023139	0.839618	00:18
12	0.053489	0.023719	0.828642	00:18
13	0.053256	0.024912	0.813356	00:18
14	0.053346	0.023202	0.834724	00:18
15	0.053634	0.024857	0.811826	00:18
16	0.053360	0.024105	0.818461	00:18
17	0.053441	0.025618	0.802739	00:18
18	0.053498	0.022734	0.845655	00:18
19	0.053307	0.023060	0.837956	00:18
20	0.053654	0.025407	0.807976	00:18
21	0.053536	0.024355	0.821352	00:18
22	0.053314	0.024032	0.821691	00:18
23	0.053320	0.024931	0.815678	00:18
24	0.053299	0.025243	0.809006	00:18
25	0.053415	0.025269	0.797242	00:18
26	0.053631	0.024213	0.810697	00:18
27	0.053417	0.026231	0.790713	00:18
28	0.053395	0.025254	0.815050	00:18
29	0.053688	0.025796	0.801579	00:18
30	0.053380	0.026426	0.801116	00:18
31	0.053296	0.024987	0.801733	00:18
32	0.053612	0.026753	0.790721	00:19
33	0.053130	0.025746	0.796553	00:18
34	0.053336	0.024837	0.818145	00:18
35	0.053110	0.026731	0.798118	00:18



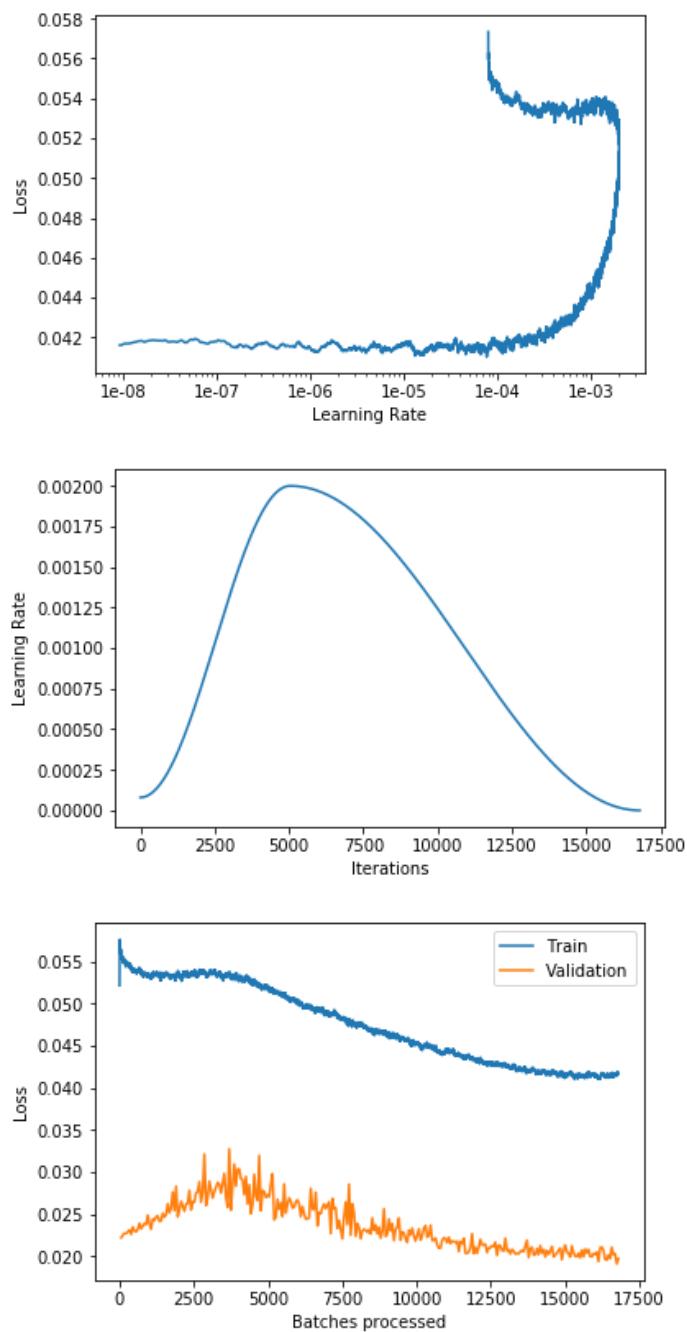
----- CURATED+NOISY - Fold 3/10 -----



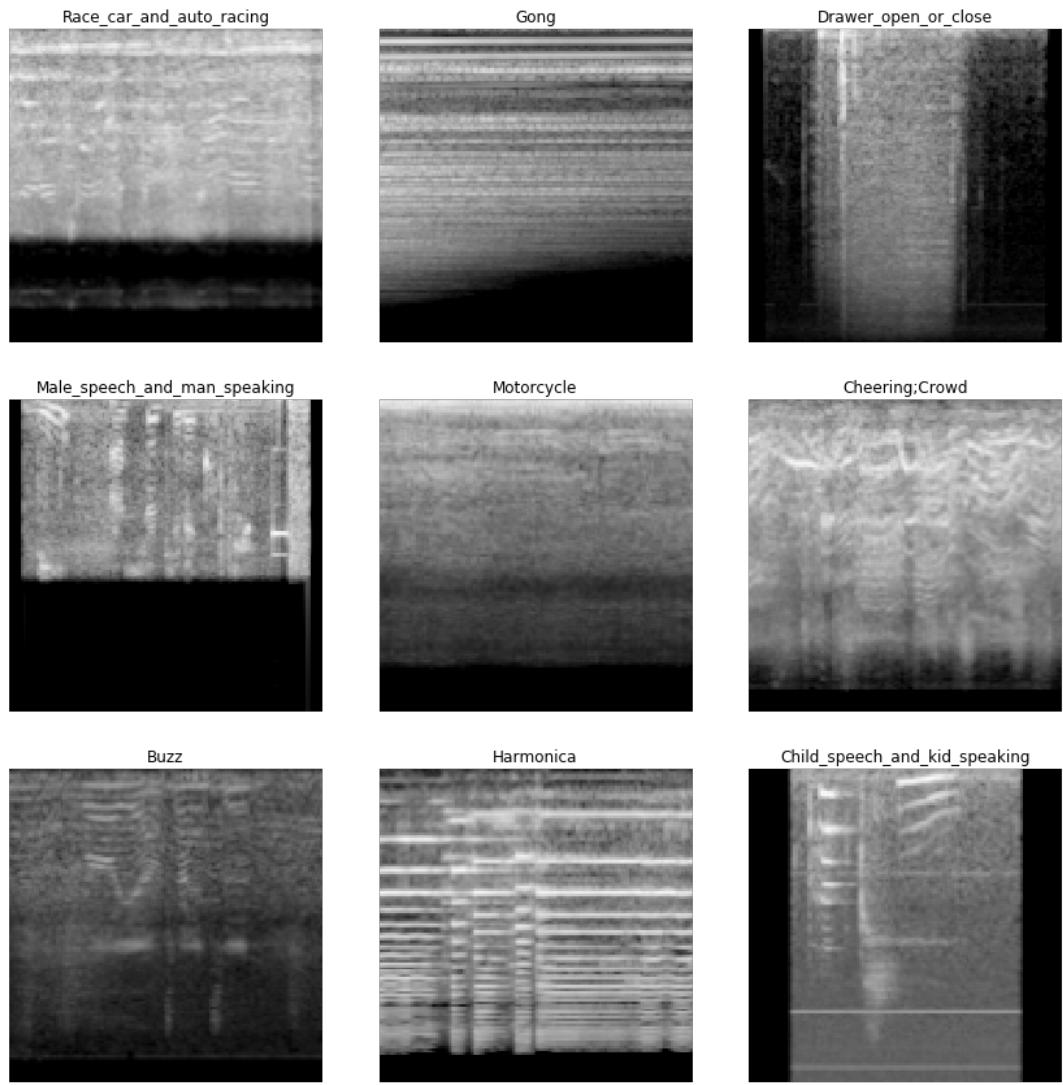
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



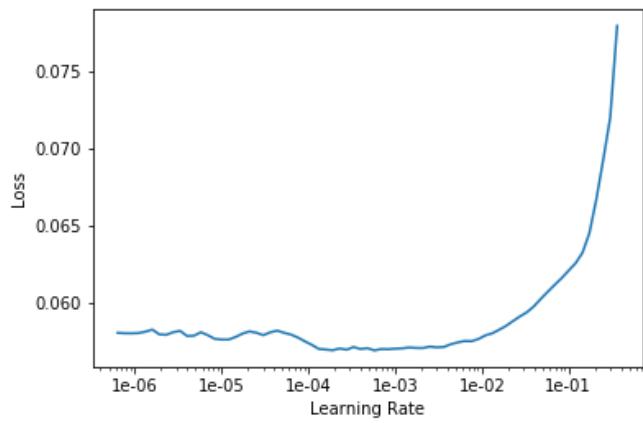
epoch	train_loss	valid_loss	lwlrap	time
0	0.055870	0.022223	0.818474	00:18
1	0.055106	0.022528	0.822440	00:18
2	0.055218	0.022734	0.822572	00:18
3	0.055080	0.022723	0.829360	00:18
4	0.054653	0.022913	0.820428	00:18
5	0.054581	0.023141	0.817792	00:18
6	0.054294	0.022620	0.821838	00:18
7	0.053966	0.023486	0.821051	00:18
8	0.054090	0.023009	0.819484	00:18
9	0.053878	0.022923	0.824686	00:18
10	0.053868	0.023222	0.823690	00:18
11	0.054033	0.023932	0.823667	00:18
12	0.053703	0.023269	0.820914	00:18
13	0.053741	0.023585	0.818008	00:18
14	0.053663	0.023905	0.817423	00:18
15	0.053397	0.023780	0.829058	00:18
16	0.053320	0.024886	0.810749	00:18
17	0.053340	0.023851	0.818982	00:18
18	0.053137	0.023229	0.827378	00:18
19	0.053495	0.024446	0.803577	00:18
20	0.053344	0.024251	0.810601	00:18
21	0.053516	0.024704	0.815529	00:18
22	0.053521	0.024789	0.812039	00:18
23	0.053515	0.024161	0.813290	00:18
24	0.053481	0.024217	0.801610	00:18
25	0.053356	0.025116	0.805252	00:18
26	0.053487	0.024672	0.805380	00:18
27	0.053792	0.024973	0.802179	00:18
28	0.053554	0.026248	0.791841	00:18
29	0.053243	0.025174	0.799299	00:18
30	0.053131	0.025737	0.794324	00:18
31	0.053241	0.027630	0.769516	00:18
32	0.053225	0.025466	0.804916	00:18
33	0.053498	0.028313	0.759945	00:18
34	0.053673	0.024713	0.812853	00:18
35	0.053310	0.025515	0.803852	00:18



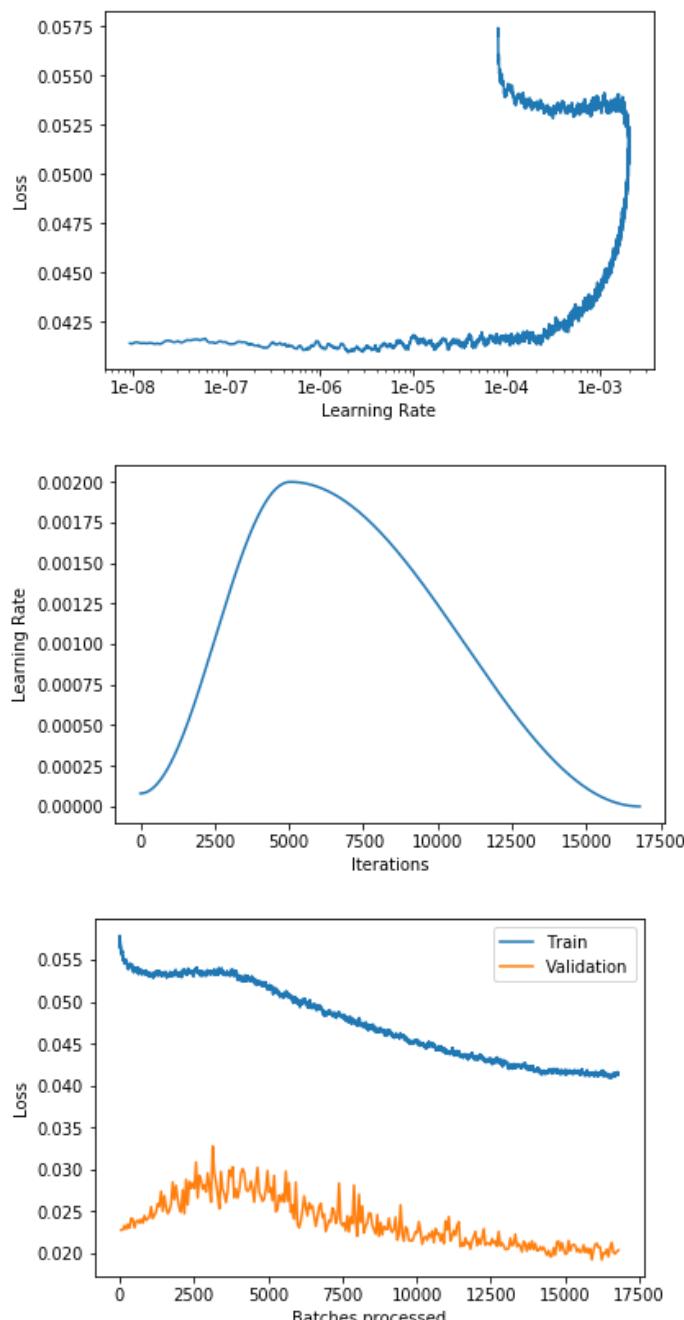
----- CURATED+NOISY - Fold 4/10 -----



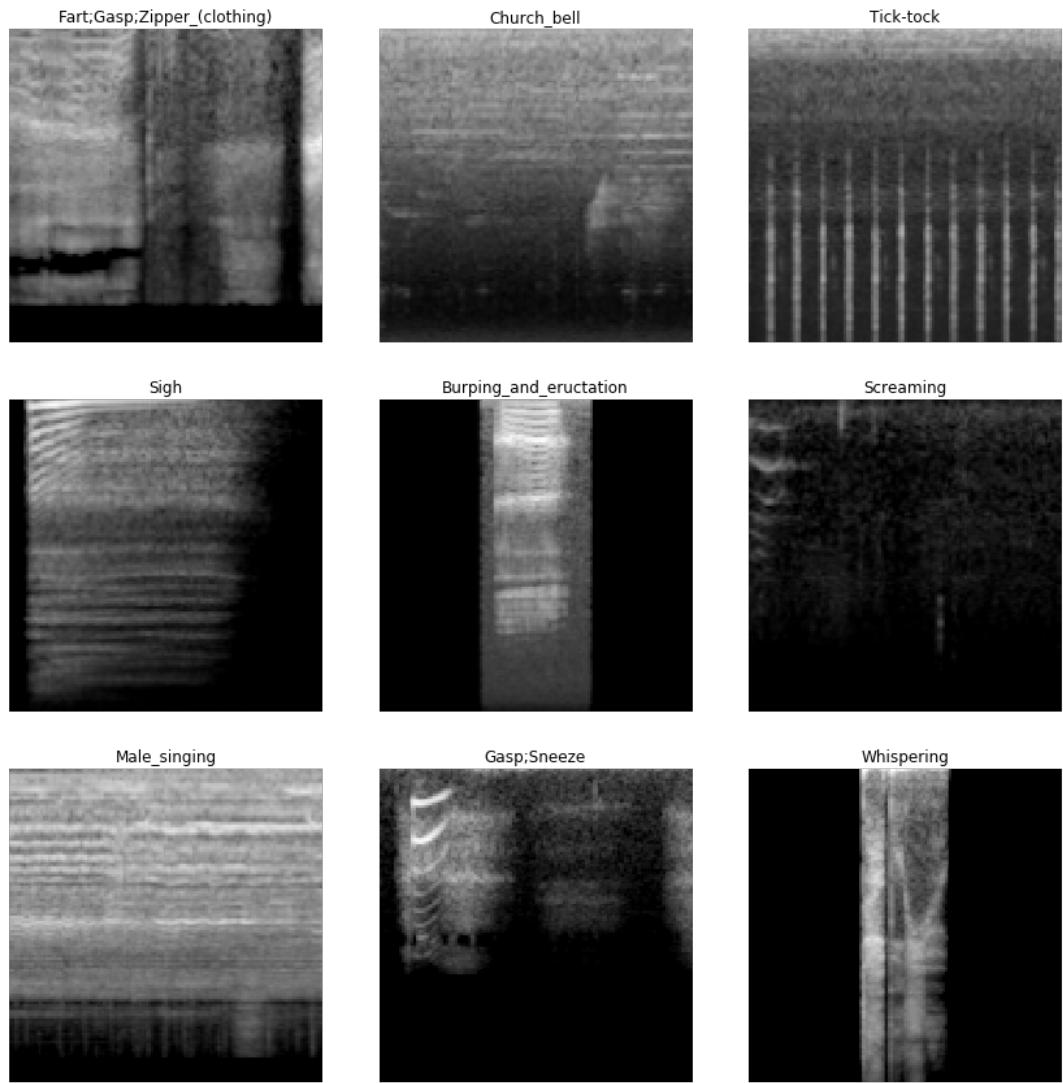
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



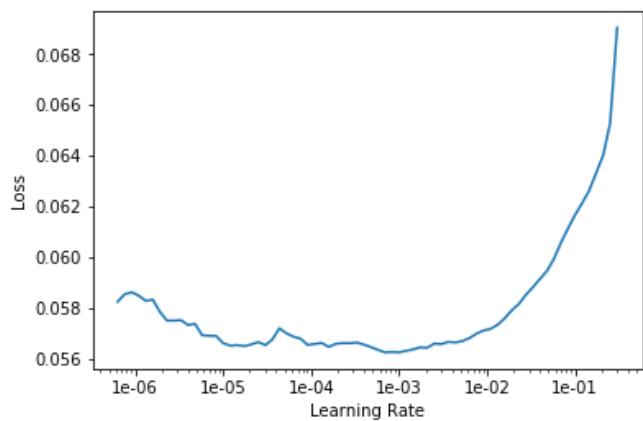
epoch	train_loss	valid_loss	lwlrap	time
0	0.055895	0.022783	0.812890	00:18
1	0.055318	0.022807	0.827679	00:18
2	0.054754	0.023301	0.815320	00:18
3	0.054565	0.022896	0.833821	00:18
4	0.054060	0.023467	0.818548	00:18
5	0.054244	0.023036	0.826318	00:18
6	0.054352	0.024181	0.818976	00:18
7	0.053974	0.024058	0.812351	00:18
8	0.053839	0.023059	0.826322	00:18
9	0.053823	0.023775	0.818993	00:18
10	0.053936	0.023932	0.821480	00:18
11	0.053597	0.023610	0.823153	00:18
12	0.053737	0.024070	0.813016	00:18
13	0.053419	0.023617	0.811571	00:18
14	0.053532	0.024496	0.818285	00:18
15	0.053624	0.023940	0.818402	00:18
16	0.053447	0.024133	0.812069	00:18
17	0.053241	0.024364	0.818426	00:18
18	0.053078	0.025549	0.811353	00:18
19	0.053118	0.024152	0.818083	00:18
20	0.053267	0.024564	0.805648	00:19
21	0.053529	0.024912	0.807139	00:19
22	0.053267	0.025913	0.798972	00:18
23	0.053610	0.025624	0.809571	00:18
24	0.053285	0.027388	0.784792	00:19
25	0.053443	0.025032	0.795076	00:19
26	0.053237	0.026845	0.780312	00:19
27	0.053106	0.025471	0.806080	00:19
28	0.053162	0.024739	0.805136	00:18
29	0.053305	0.025107	0.804630	00:20
30	0.053165	0.025877	0.804735	00:19
31	0.053438	0.028492	0.776555	00:19
32	0.053269	0.026432	0.788552	00:19
33	0.053415	0.028057	0.769911	00:19
34	0.053320	0.026464	0.790622	00:18
35	0.053173	0.025397	0.810353	00:18



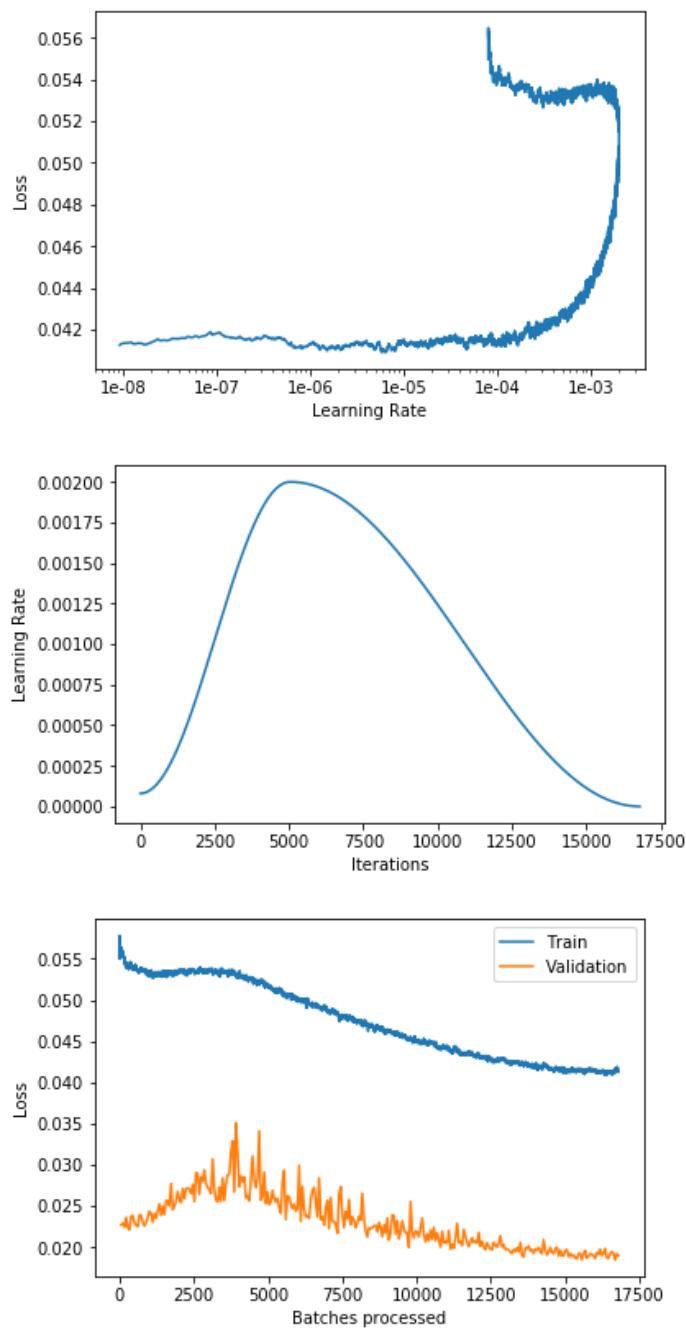
----- CURATED+NOISY - Fold 5/10 -----



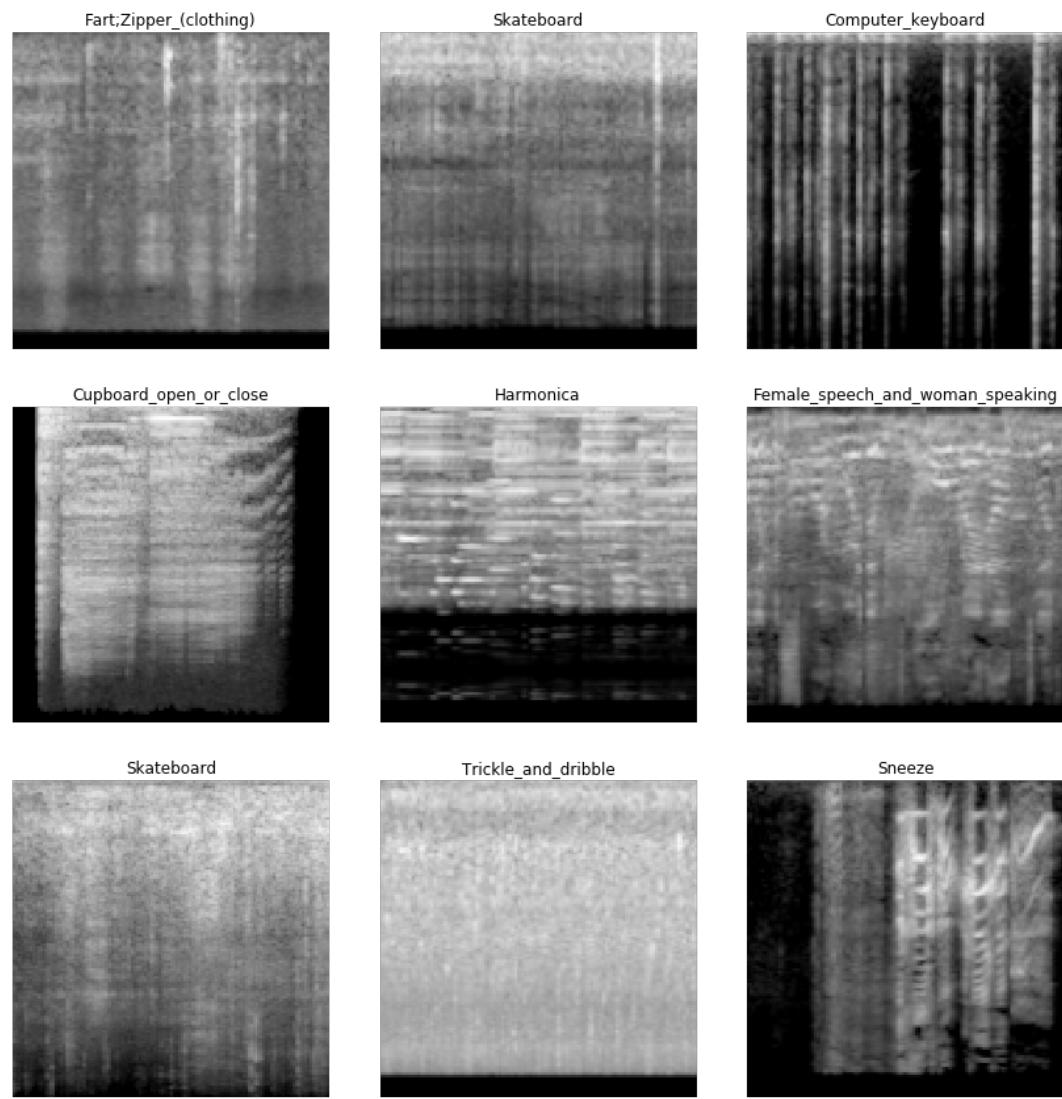
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



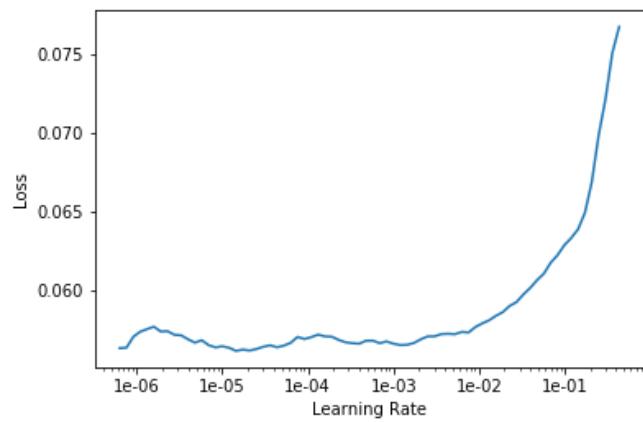
epoch	train_loss	valid_loss	lwlrap	time
0	0.056210	0.022737	0.837922	00:18
1	0.055442	0.022906	0.836853	00:18
2	0.054829	0.022448	0.840432	00:18
3	0.054241	0.023622	0.829569	00:18
4	0.054181	0.022393	0.848995	00:18
5	0.054292	0.022135	0.856921	00:18
6	0.054159	0.023784	0.829728	00:18
7	0.053879	0.023823	0.827585	00:18
8	0.054048	0.022927	0.842064	00:18
9	0.053660	0.022679	0.842641	00:18
10	0.053576	0.022525	0.850303	00:18
11	0.053644	0.023933	0.815128	00:18
12	0.053755	0.023860	0.841308	00:18
13	0.053656	0.023027	0.844859	00:18
14	0.053675	0.022594	0.848413	00:18
15	0.053394	0.022955	0.835513	00:18
16	0.053072	0.023660	0.824235	00:18
17	0.053174	0.024348	0.826747	00:18
18	0.053232	0.024091	0.833577	00:18
19	0.052856	0.023046	0.844694	00:18
20	0.053225	0.024076	0.835458	00:18
21	0.052983	0.023535	0.838160	00:18
22	0.053268	0.024459	0.823872	00:18
23	0.053329	0.025245	0.827821	00:18
24	0.053276	0.023880	0.833576	00:18
25	0.053380	0.024976	0.816164	00:18
26	0.053117	0.023978	0.841376	00:18
27	0.052883	0.025658	0.811138	00:18
28	0.053524	0.025631	0.813646	00:18
29	0.053468	0.025025	0.819915	00:18
30	0.053442	0.027714	0.788740	00:18
31	0.053267	0.024566	0.825400	00:18
32	0.053539	0.024955	0.821579	00:18
33	0.053335	0.025723	0.826764	00:18
34	0.053315	0.026758	0.804920	00:18
35	0.053538	0.025818	0.809859	00:18



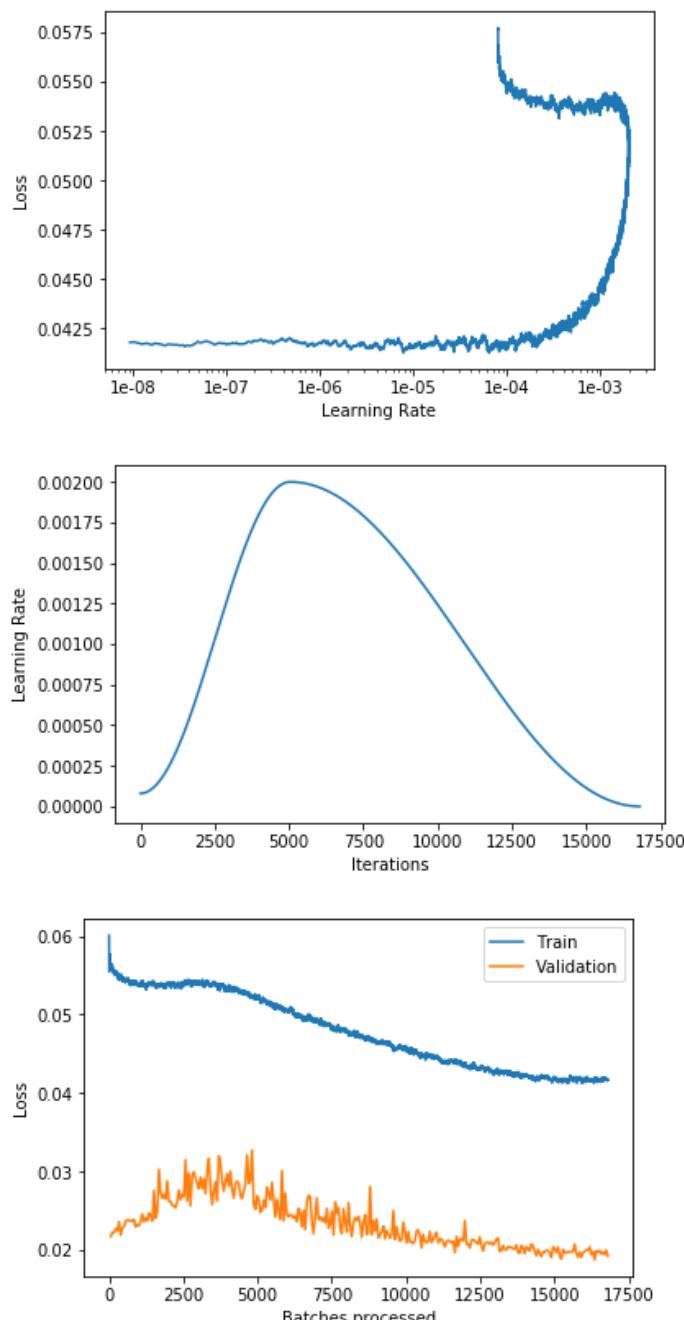
----- CURATED+NOISY - Fold 6/10 -----



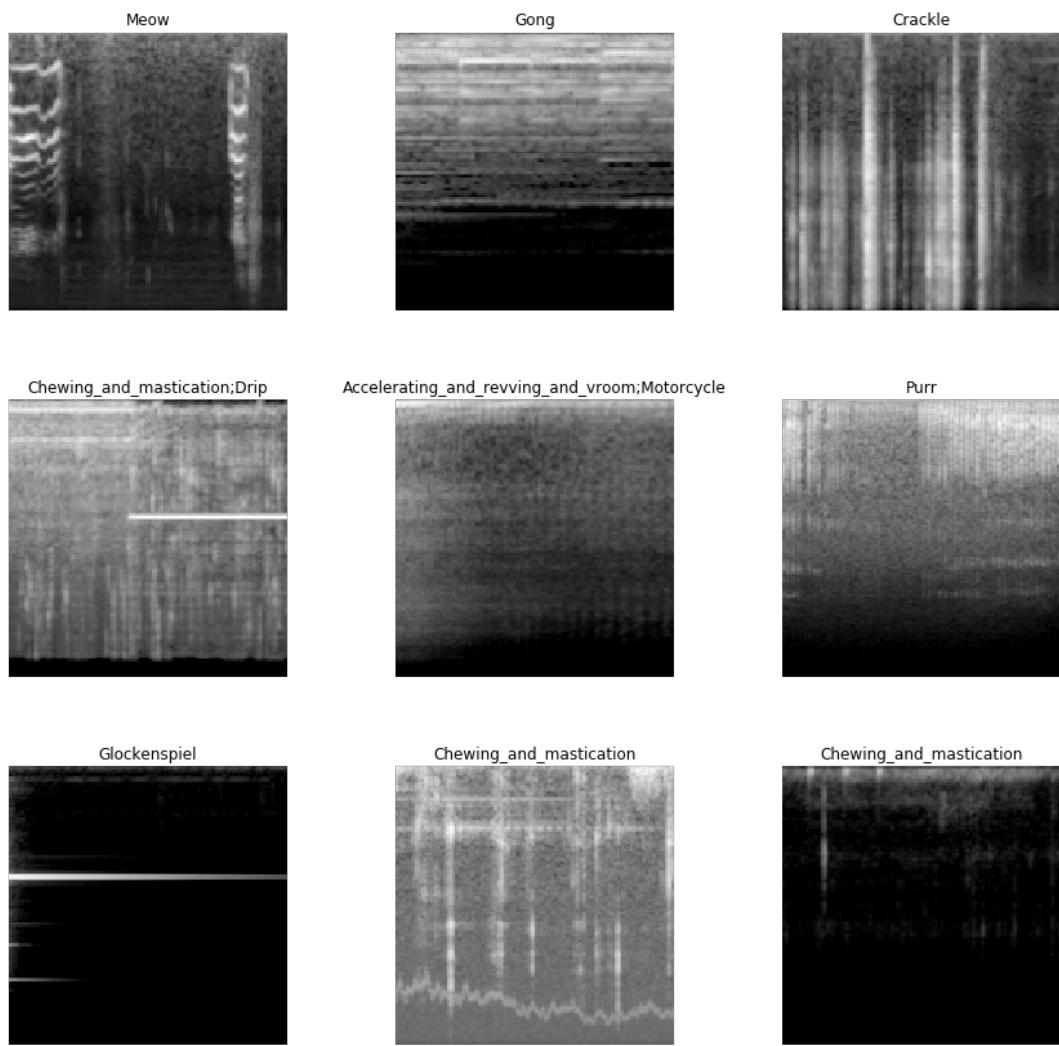
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



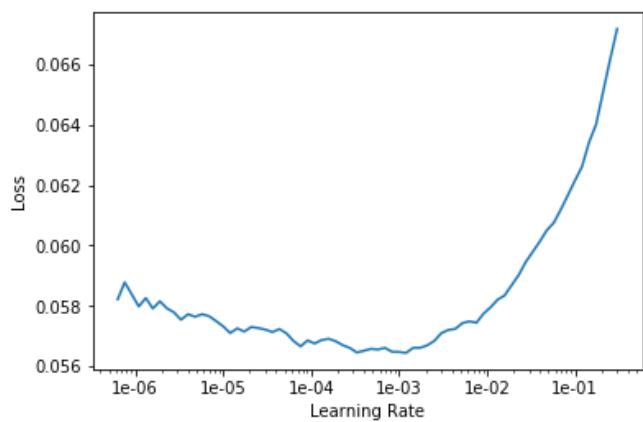
epoch	train_loss	valid_loss	lwlrap	time
0	0.056386	0.021757	0.835802	00:18
1	0.055731	0.022283	0.830422	00:18
2	0.055402	0.022243	0.828274	00:18
3	0.055344	0.022744	0.823428	00:18
4	0.055154	0.022484	0.832291	00:18
5	0.054708	0.023635	0.823219	00:18
6	0.054820	0.021958	0.840708	00:18
7	0.054438	0.022829	0.832367	00:18
8	0.054663	0.023001	0.831237	00:18
9	0.054391	0.023699	0.821501	00:18
10	0.054433	0.023873	0.821184	00:18
11	0.054207	0.023771	0.819972	00:18
12	0.053984	0.023687	0.815431	00:18
13	0.054196	0.023820	0.822336	00:18
14	0.054109	0.023923	0.819806	00:18
15	0.054142	0.022968	0.830109	00:18
16	0.053926	0.023264	0.823483	00:18
17	0.053862	0.023144	0.821807	00:18
18	0.053970	0.023644	0.823852	00:18
19	0.054118	0.024604	0.817386	00:18
20	0.053713	0.023656	0.824739	00:18
21	0.053537	0.024121	0.813272	00:18
22	0.053945	0.023760	0.822508	00:18
23	0.053873	0.024711	0.807807	00:18
24	0.053938	0.024438	0.819685	00:18
25	0.053870	0.023738	0.822435	00:18
26	0.053883	0.027604	0.781709	00:18
27	0.053598	0.024112	0.825158	00:18
28	0.053640	0.024835	0.816462	00:18
29	0.053694	0.030269	0.746298	00:18
30	0.053788	0.027342	0.771036	00:18
31	0.053726	0.026650	0.791091	00:18
32	0.053556	0.027126	0.795507	00:18
33	0.053821	0.026536	0.788505	00:18
34	0.053916	0.028801	0.763086	00:18
35	0.054028	0.026213	0.803111	00:18



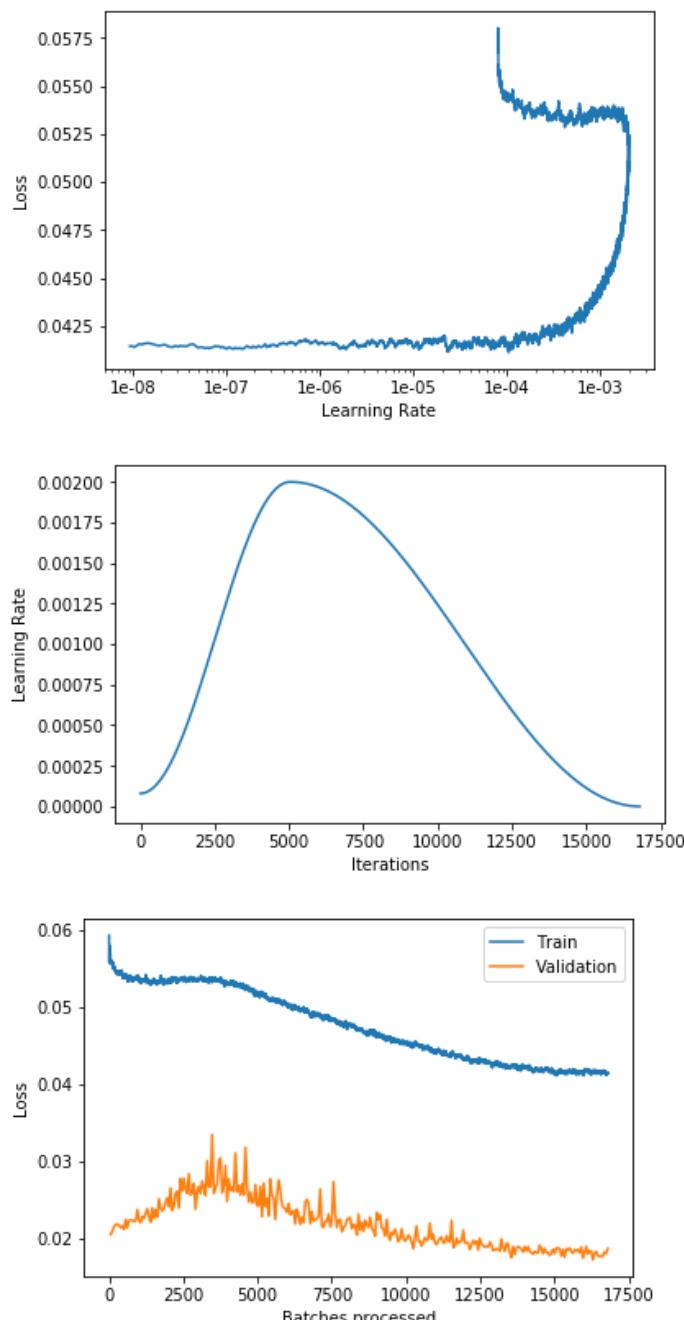
----- CURATED+NOISY - Fold 7/10 -----



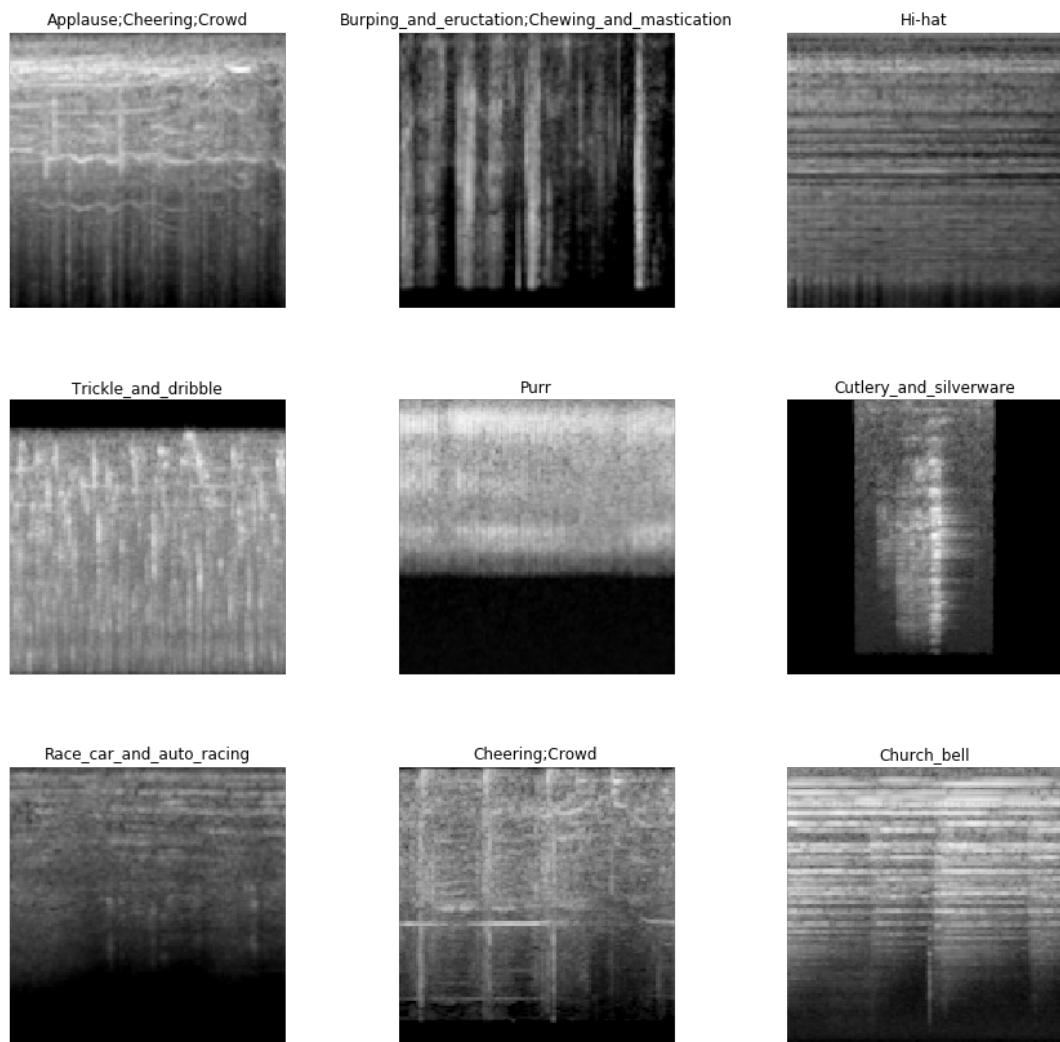
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



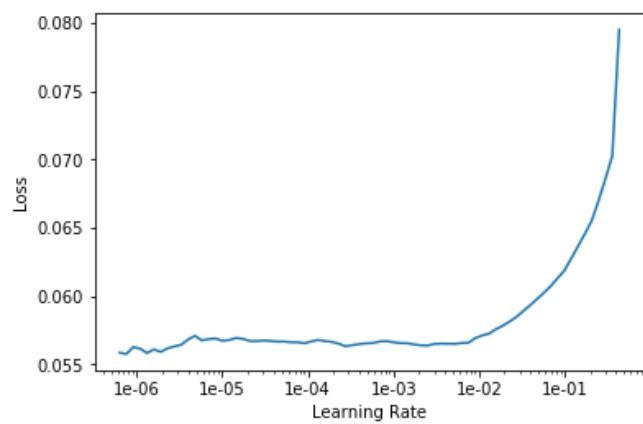
epoch	train_loss	valid_loss	lwlrap	time
0	0.055995	0.020570	0.855194	00:18
1	0.055574	0.021007	0.862009	00:18
2	0.055002	0.021568	0.850429	00:18
3	0.054638	0.021804	0.849458	00:18
4	0.054689	0.021882	0.849943	00:18
5	0.054611	0.021717	0.849684	00:18
6	0.054281	0.021455	0.849257	00:18
7	0.054109	0.021670	0.850144	00:18
8	0.054108	0.021222	0.861421	00:18
9	0.053932	0.022482	0.838830	00:18
10	0.053671	0.021430	0.861915	00:18
11	0.053863	0.022351	0.838967	00:18
12	0.053717	0.022450	0.845485	00:18
13	0.053560	0.022392	0.850328	00:18
14	0.053843	0.022240	0.844736	00:18
15	0.053678	0.022328	0.843882	00:18
16	0.053840	0.023752	0.836396	00:18
17	0.053690	0.022384	0.844192	00:18
18	0.053569	0.022059	0.842647	00:18
19	0.053321	0.022841	0.841278	00:18
20	0.053463	0.022645	0.846706	00:18
21	0.053659	0.023359	0.836397	00:18
22	0.053316	0.023858	0.837036	00:18
23	0.053368	0.022833	0.844492	00:18
24	0.053134	0.022143	0.849492	00:18
25	0.053303	0.023994	0.823914	00:18
26	0.053474	0.022613	0.847705	00:18
27	0.053189	0.023734	0.826731	00:18
28	0.053232	0.022808	0.841412	00:18
29	0.053441	0.025141	0.817336	00:18
30	0.053846	0.023924	0.818718	00:18
31	0.053346	0.023591	0.833653	00:18
32	0.053018	0.023905	0.834889	00:18
33	0.053280	0.024644	0.825039	00:18
34	0.053218	0.025284	0.829069	00:18
35	0.053106	0.024115	0.838441	00:18



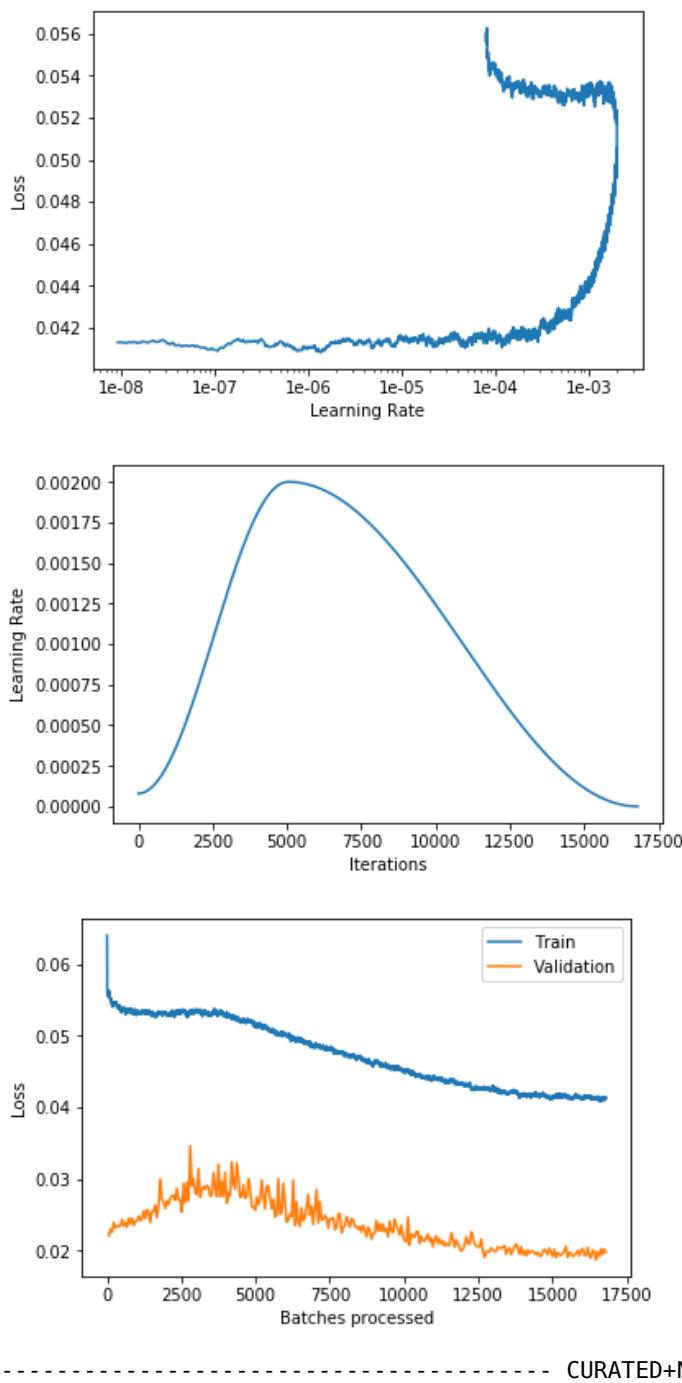
----- CURATED+NOISY - Fold 8/10 -----



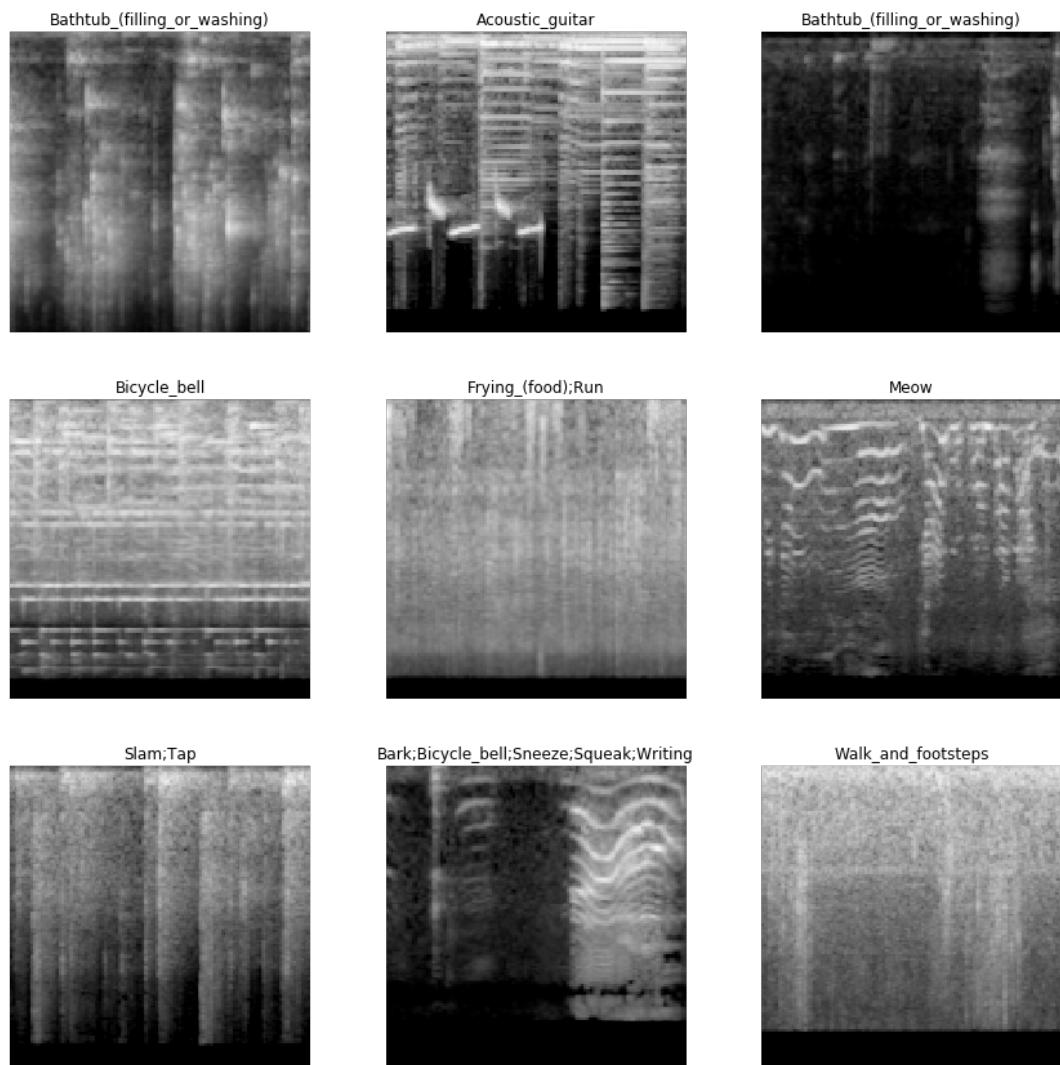
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



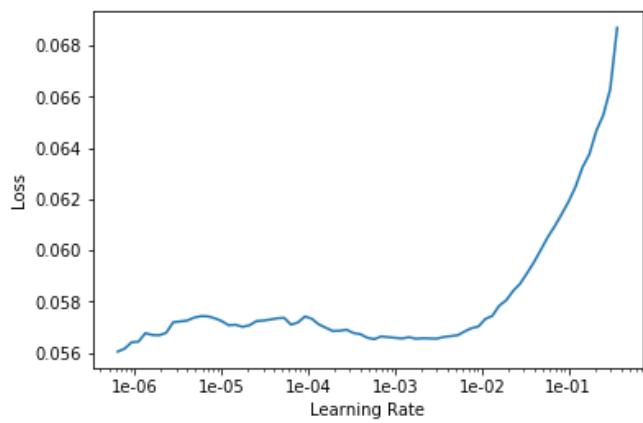
epoch	train_loss	valid_loss	lwlrap	time
0	0.055949	0.022097	0.837921	00:18
1	0.055093	0.022951	0.831325	00:18
2	0.054435	0.022570	0.843118	00:18
3	0.054514	0.023912	0.818493	00:18
4	0.054421	0.023202	0.828404	00:18
5	0.054150	0.023331	0.831797	00:18
6	0.054101	0.023485	0.830500	00:18
7	0.053798	0.023437	0.836027	00:18
8	0.053553	0.024333	0.820789	00:18
9	0.053703	0.023447	0.827507	00:18
10	0.053196	0.023909	0.827289	00:18
11	0.053458	0.023292	0.839203	00:19
12	0.053531	0.024529	0.818920	00:18
13	0.053399	0.023738	0.825449	00:18
14	0.053239	0.024530	0.819471	00:18
15	0.053421	0.023539	0.836022	00:19
16	0.053348	0.024126	0.829396	00:20
17	0.053290	0.024154	0.828254	00:19
18	0.053298	0.024828	0.830914	00:18
19	0.053199	0.024180	0.825127	00:18
20	0.053399	0.024520	0.823050	00:18
21	0.053325	0.025543	0.806852	00:18
22	0.053335	0.024480	0.825805	00:18
23	0.053063	0.025690	0.808189	00:19
24	0.053297	0.025371	0.819757	00:18
25	0.053096	0.023900	0.833476	00:18
26	0.053007	0.025506	0.809195	00:18
27	0.052916	0.025475	0.816588	00:19
28	0.053097	0.026352	0.809922	00:19
29	0.053065	0.024624	0.818962	00:19
30	0.053166	0.026278	0.801745	00:18
31	0.053241	0.029991	0.756724	00:18
32	0.053178	0.027124	0.795808	00:19
33	0.053240	0.026691	0.809075	00:20
34	0.053106	0.025902	0.815944	00:20
35	0.052971	0.026918	0.798125	00:20



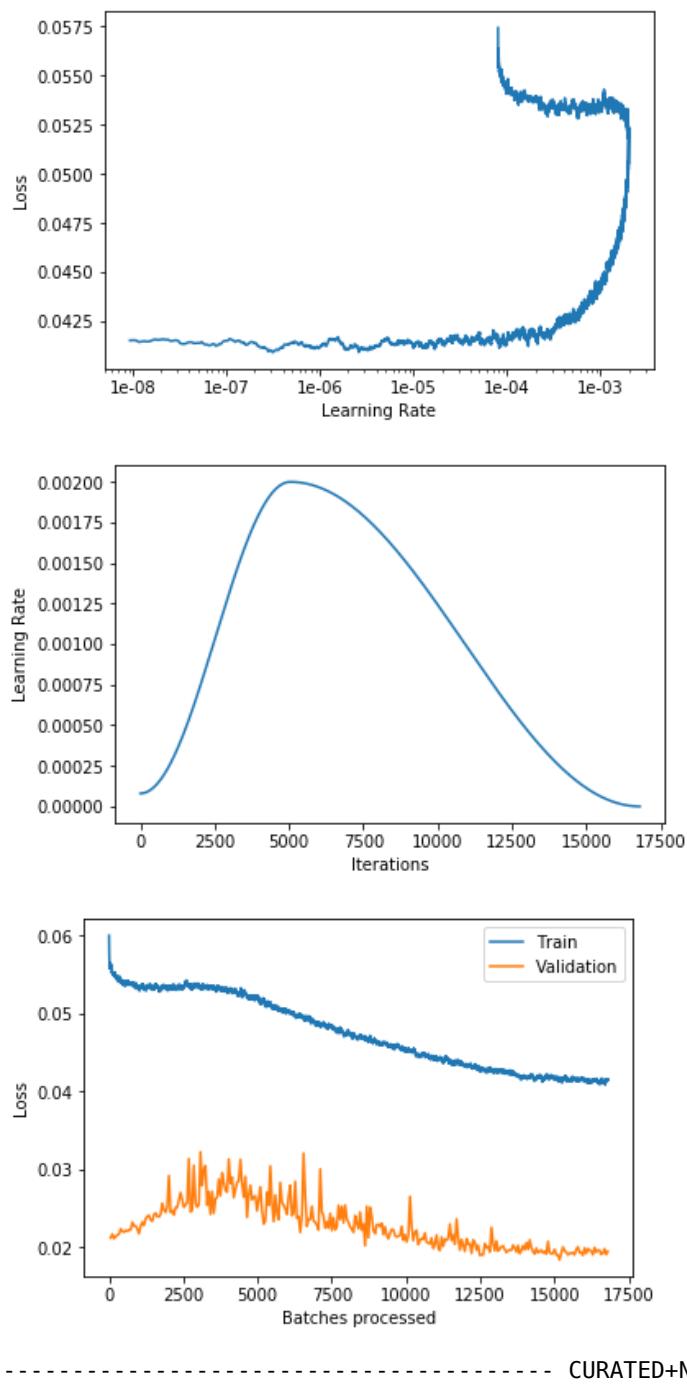
----- CURATED+NOISY - Fold 9/10 -----



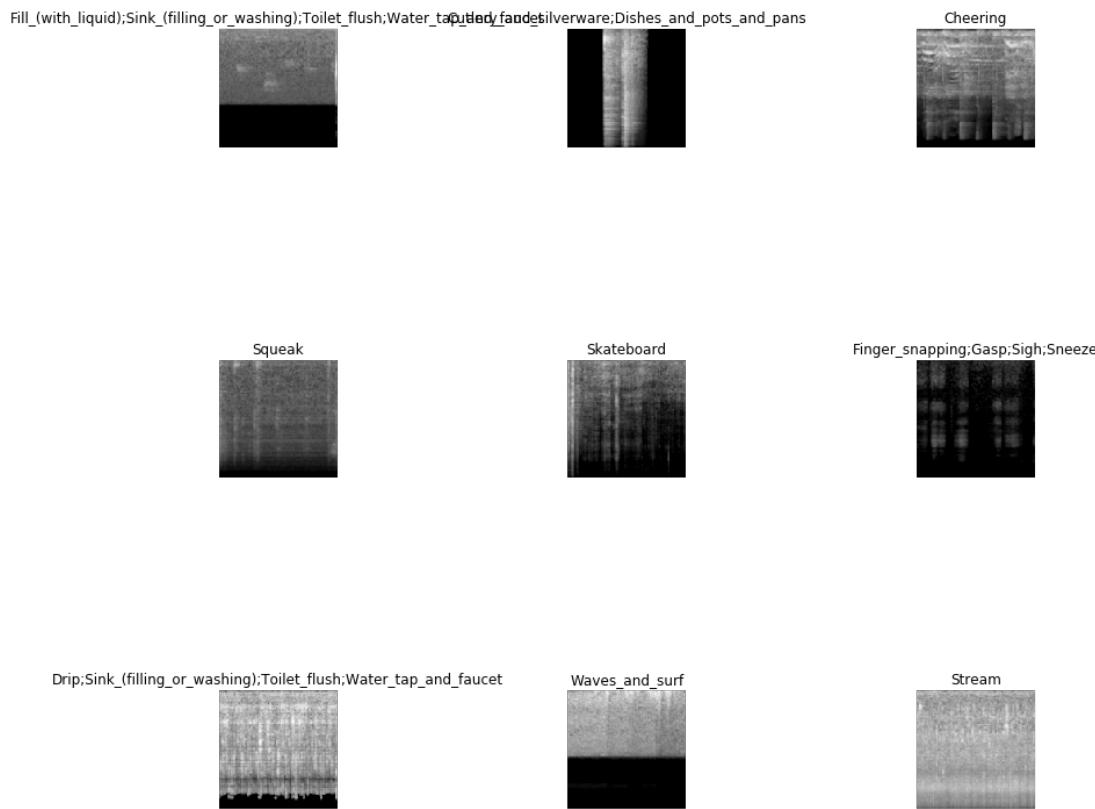
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



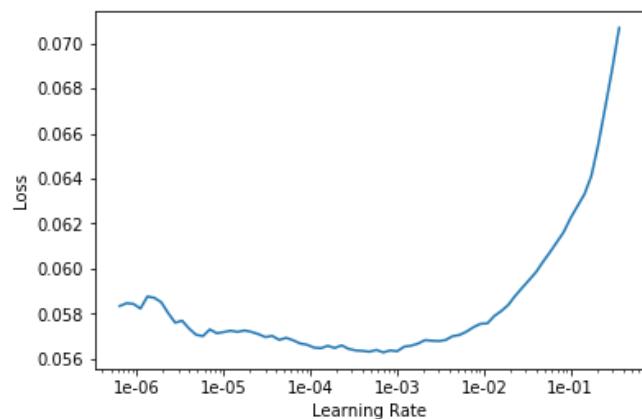
epoch	train_loss	valid_loss	lwlrap	time
0	0.056028	0.021243	0.846011	00:18
1	0.055253	0.021727	0.842090	00:18
2	0.055243	0.021143	0.851824	00:18
3	0.054712	0.021438	0.841214	00:18
4	0.054633	0.021602	0.850444	00:18
5	0.054618	0.021735	0.840150	00:19
6	0.054283	0.022394	0.835329	00:18
7	0.054082	0.022125	0.841425	00:18
8	0.053881	0.021972	0.848040	00:18
9	0.054147	0.022193	0.844640	00:18
10	0.053923	0.022099	0.848030	00:18
11	0.053918	0.022278	0.838072	00:18
12	0.053817	0.022276	0.842307	00:19
13	0.053896	0.023251	0.828931	00:19
14	0.053773	0.022773	0.839418	00:20
15	0.053700	0.022781	0.835512	00:21
16	0.053456	0.022405	0.843159	00:20
17	0.053182	0.021801	0.849949	00:21
18	0.053465	0.022755	0.834880	00:20
19	0.053525	0.023256	0.838330	00:20
20	0.053424	0.022850	0.835823	00:20
21	0.053435	0.023695	0.836860	00:20
22	0.053527	0.024002	0.824058	00:20
23	0.053426	0.024129	0.817829	00:20
24	0.053447	0.023559	0.826354	00:20
25	0.053309	0.023399	0.835721	00:20
26	0.053425	0.024219	0.826341	00:20
27	0.053516	0.024436	0.822819	00:20
28	0.053519	0.023996	0.832837	00:20
29	0.053117	0.023951	0.828439	00:20
30	0.053331	0.024227	0.831234	00:20
31	0.053551	0.025681	0.821834	00:20
32	0.053485	0.024850	0.813692	00:20
33	0.053436	0.024523	0.826374	00:20
34	0.053256	0.025234	0.818903	00:20
35	0.053285	0.029209	0.750866	00:20



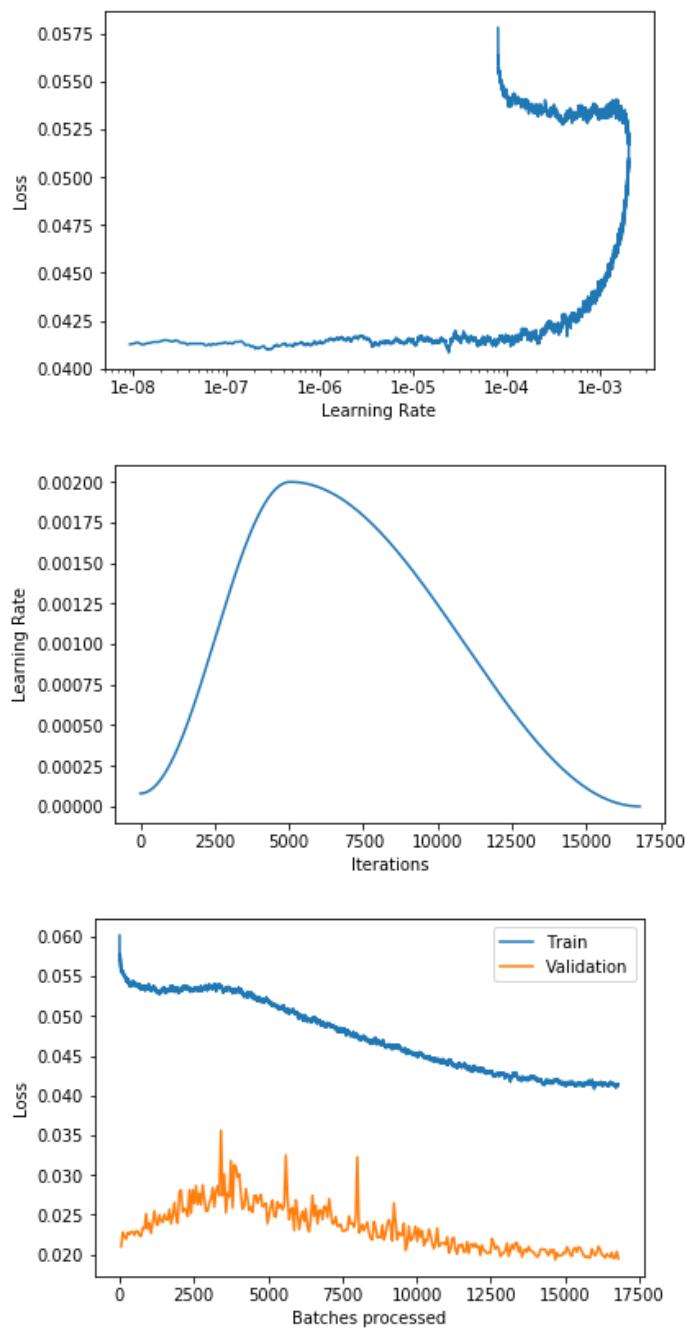
----- CURATED+NOISY - Fold 10/10 -----



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.056011	0.021008	0.838507	00:18
1	0.055592	0.022793	0.819009	00:19
2	0.055018	0.022287	0.829713	00:18
3	0.054742	0.021948	0.840029	00:18
4	0.054316	0.022781	0.824015	00:18
5	0.054165	0.022527	0.834342	00:19
6	0.053936	0.022869	0.829888	00:19
7	0.054166	0.022847	0.831993	00:18
8	0.053971	0.022437	0.836979	00:19
9	0.054052	0.022957	0.836431	00:18
10	0.053779	0.022901	0.833013	00:18
11	0.053924	0.022407	0.843543	00:18
12	0.053692	0.022310	0.837015	00:19
13	0.053610	0.023364	0.822462	00:19
14	0.053576	0.023274	0.834462	00:19
15	0.053744	0.025114	0.803574	00:18
16	0.053618	0.023228	0.828334	00:18
17	0.053503	0.023948	0.814733	00:18
18	0.053426	0.022678	0.829235	00:18
19	0.053561	0.024137	0.820738	00:18
20	0.053452	0.025486	0.809924	00:18
21	0.053069	0.024030	0.818401	00:18
22	0.053155	0.023686	0.829950	00:18
23	0.052972	0.024984	0.806474	00:18
24	0.053234	0.024230	0.818625	00:18
25	0.053201	0.024050	0.825784	00:19
26	0.053395	0.025061	0.801171	00:19
27	0.053159	0.024412	0.823876	00:19
28	0.053270	0.024387	0.813583	00:19
29	0.053228	0.025700	0.792096	00:19
30	0.053754	0.023870	0.828663	00:18
31	0.053630	0.024894	0.805898	00:19
32	0.053342	0.026519	0.805560	00:20
33	0.053411	0.024726	0.820268	00:18
34	0.053076	0.023936	0.823604	00:18
35	0.053301	0.027852	0.779209	00:18



```
In [36]: kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-10_fold-{fold}')
```

Overall lwlrapi on CURATED dataset: 0.8867165030236389

	lwlrap	weight
Squeak	0.647021	0.013039
Walk_and_footsteps	0.682881	0.013039
Mechanical_fan	0.706404	0.008519
Fill_(with_liquid)	0.718342	0.008693
Hiss	0.750569	0.013039
Tap	0.762854	0.013039
Chink_and_clink	0.771752	0.013039
Cutlery_and_silverware	0.772668	0.013039
Sink_(filling_or_washing)	0.778612	0.013039
Traffic_noise_and_roadway_noise	0.780981	0.013039
Yell	0.783729	0.013039
Bus	0.801715	0.013039
Male_speech_and_manSpeaking	0.803987	0.013039
Buzz	0.812107	0.009736
Dishes_and_pots_and_pans	0.812634	0.013039
Microwave_oven	0.822744	0.013039
Stream	0.825904	0.013039
Water_tap_and_faucet	0.826924	0.013039
Clapping	0.832853	0.013039
Accelerating_and_revving_and_vroom	0.834608	0.013039
Frying_(food)	0.844347	0.010953
Slam	0.851981	0.013039
Bathtub_(filling_or_washing)	0.852930	0.013039
Motorcycle	0.854571	0.013039
Trickle_and_dribble	0.861317	0.009214
Waves_and_surf	0.871239	0.013039
Knock	0.872580	0.013039
Chirp_and_tweet	0.873667	0.013039
Run	0.876085	0.013039
Gurgling	0.880096	0.013039
...	...	...
Chewing_and_mastication	0.915778	0.013039
Cricket	0.919175	0.013039
Screaming	0.925524	0.013039
Bass_drum	0.929037	0.013039

```
In [37]: kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-10_fold-{fold}')
```

Overall lwlrapp on NOISY dataset: 0.5651347027383816

	<b>lwlrap</b>	<b>weight</b>
Burping_and_eructation	0.127234	0.0125
Raindrop	0.185532	0.0125
Finger_snapping	0.191969	0.0125
Tap	0.194861	0.0125
Keys_jangling	0.211629	0.0125
Cupboard_open_or_close	0.226710	0.0125
Sigh	0.241519	0.0125
Shatter	0.279704	0.0125
Zipper_(clothing)	0.285012	0.0125
Writing	0.288094	0.0125
Buzz	0.300866	0.0125
Fart	0.304988	0.0125
Gasp	0.342969	0.0125
Scissors	0.362456	0.0125
Knock	0.391017	0.0125
Computer_keyboard	0.392559	0.0125
Tick-tock	0.418538	0.0125
Squeak	0.449235	0.0125
Chink_and_clink	0.466047	0.0125
Bicycle_bell	0.484772	0.0125
Strum	0.486592	0.0125
Sneeze	0.491287	0.0125
Purr	0.493699	0.0125
Crackle	0.497481	0.0125
Slam	0.500715	0.0125
Chewing_and_mastication	0.505453	0.0125
Skateboard	0.511400	0.0125
Whispering	0.518535	0.0125
Drip	0.525664	0.0125
Microwave_oven	0.535433	0.0125
...	...	...
Water_tap_and_faucet	0.654592	0.0125
Marimba_and_xylophone	0.654900	0.0125
Electric_guitar	0.657811	0.0125
Screaming	0.658451	0.0125

```
In [38]: def lwlrp_per_sample(fname, kf, df):
    overall_preds, overall_thruth, overall_index = _kfold_prediction(kf, df, fname)

    m = pd.DataFrame(
        [_one_sample_positive_class_precisions(p, t)[1].mean() for i, (p, t) in
        enumerate(zip(overall_preds, overall_thruth))],
        columns=['lwlrp'],
        index=overall_index
    )

    m[['fname', 'labels']] = df[['fname', 'labels']]

    return m.sort_values('lwlrp', ascending=False)
```

```
In [39]: m4 = lwlrp_per_sample('stage-10_fold-{fold}', kf_noisy, trn_noisy_df)
```

In [40]: m4

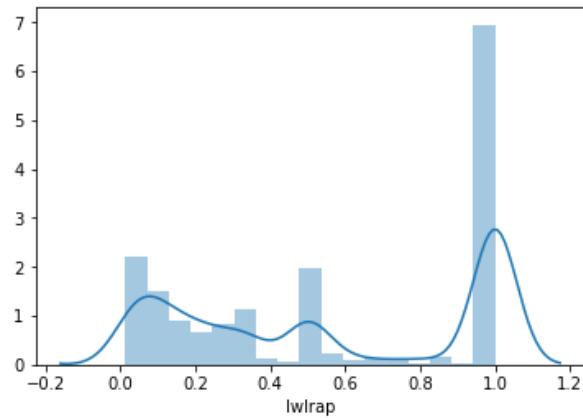
Out[40]:

	lwlrap	fname	labels
<b>16</b>	1.000000	002d93e0.wav	Crowd
<b>5757</b>	1.000000	4a0a6161.wav	Hi-hat
<b>4976</b>	1.000000	400377ba.wav	Accordion
<b>4994</b>	1.000000	404452c1.wav	Bass_drum,Hi-hat,Bass_guitar
<b>5062</b>	1.000000	412a484d.wav	Gong
<b>5151</b>	1.000000	425a058d.wav	Yell
<b>5186</b>	1.000000	42b835ed.wav	Female_singing
<b>5191</b>	1.000000	42c7ec6b.wav	Zipper_(clothing)
<b>5245</b>	1.000000	4379e27a.wav	Walk_and_footsteps
<b>5249</b>	1.000000	43855905.wav	Motorcycle
<b>5335</b>	1.000000	44976cd0.wav	Electric_guitar
<b>5436</b>	1.000000	460b81ad.wav	Strum,Meow,Electric_guitar
<b>5457</b>	1.000000	4632bfb5.wav	Bus
<b>5458</b>	1.000000	46363cf3.wav	Mechanical_fan
<b>5471</b>	1.000000	466e8ae2.wav	Gurgling
<b>5520</b>	1.000000	4723080c.wav	Female_singing
<b>5547</b>	1.000000	47829a76.wav	Gong
<b>5550</b>	1.000000	47951e20.wav	Bass_drum
<b>5553</b>	1.000000	47974aea.wav	Applause
<b>5608</b>	1.000000	483b2c92.wav	Bass_drum,Hi-hat
<b>5638</b>	1.000000	48914120.wav	Yell
<b>4974</b>	1.000000	3ffb4126.wav	Cricket
<b>4971</b>	1.000000	3fe99bcd.wav	Fill_(with_liquid),Drip
<b>4902</b>	1.000000	3ef5a6c6.wav	Tap
<b>4592</b>	1.000000	3ac7b6f9.wav	Accordion
<b>4391</b>	1.000000	38184ccf.wav	Drawer_open_or_close
<b>4426</b>	1.000000	38878b3a.wav	Waves_and_surf
<b>4495</b>	1.000000	39819c62.wav	Yell,Hi-hat
<b>4499</b>	1.000000	398bb2cd.wav	Chirp_and_tweet
<b>4518</b>	1.000000	39bdef2e.wav	Yell
...	...	...	...
<b>10833</b>	0.013158	8bb9b5b0.wav	Sigh
<b>505</b>	0.013158	066f2ac5.wav	Sigh
<b>802</b>	0.013158	0a84be10.wav	Cupboard_open_or_close
<b>18927</b>	0.013158	f46c9de4.wav	Hi-hat

```
In [41]: (m4.lwlrp < .5).sum(), (m4.lwlrp >= .5).sum(), (m4.lwlrp == 1).sum()
```

```
Out[41]: (8583, 11232, 7991)
```

```
In [42]: sns.distplot(m4.lwlrp);
```



```
In [43]: ok_noisy_items2 = m4[m4.lwlrp == 1].index  
len(ok_noisy_items2)
```

```
Out[43]: 7991
```

```
In [44]: for fold, ((train_index1, valid_index1), (train_index2, valid_index2)) in enumerate(zip(kf_curated.split(trn_curated_df), kf_noisy.split(trn_noisy_df))):
    print('-' * 40, f'CURATED+NOISY+2 - Fold {fold+1}/{n_splits}', '-' * 40)

    train_index2 = list(set(train_index2).intersection(set(ok_noisy_items2)))

    mix_df = pd.concat([
        trn_curated_df.iloc[train_index1],
        trn_noisy_df.iloc[train_index2],
        trn_curated_df.iloc[valid_index1],
#        trn_noisy_df.iloc[valid_index2], # compute lwlrap only on curated
    ], ignore_index=True)
    train_index = mix_df[:len(train_index1)+len(train_index2)].index
    valid_index = mix_df[len(train_index1)+len(train_index2):].index

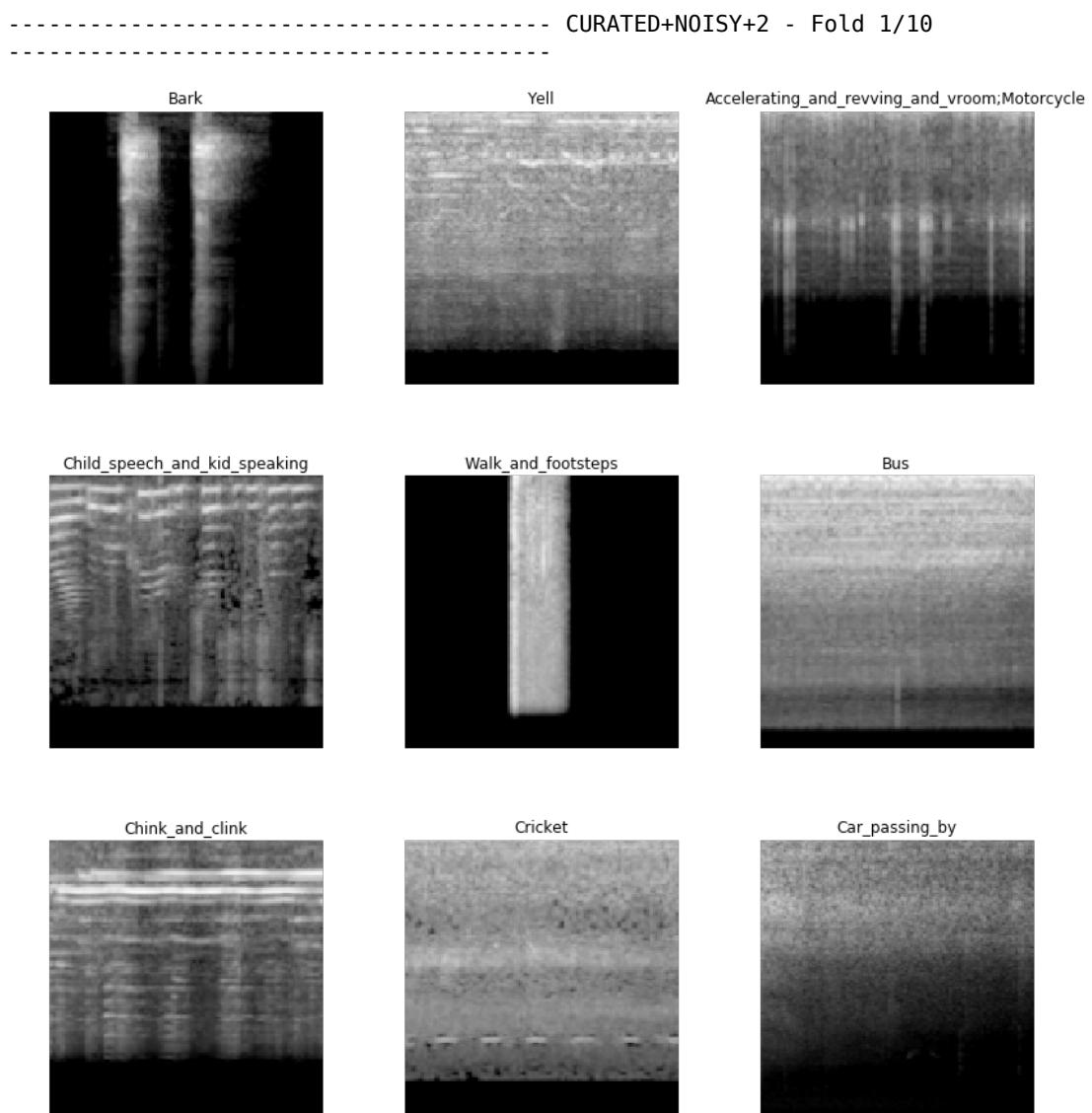
    src = (ImageList.from_df(mix_df, WORK)
           .split_by_idxs(train_index, valid_index)
           .label_from_df(label_delim=','))
    data = (src.transform(tfms, size=128)
            .databunch(bs=bs))
    data.show_batch(3)
    plt.show()

    learn.load(f'stage-10_fold-{fold}')
    learn.data = data

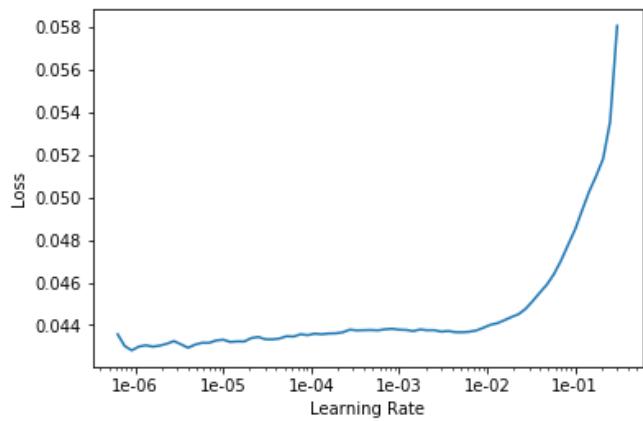
    learning(learn, 100, 1e-2)

    learn.save(f'stage-11_fold-{fold}')
    learn.export(f'stage-11_fold-{fold}.pkl')

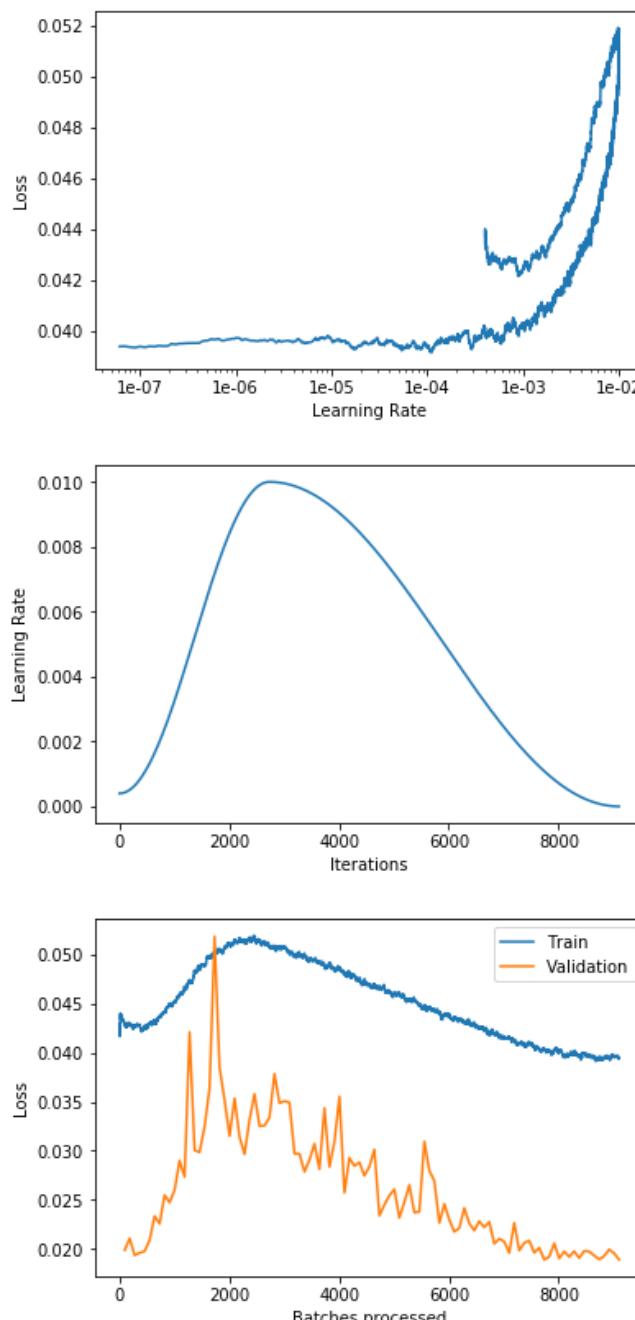
    if TOY_MODE:
        break
```



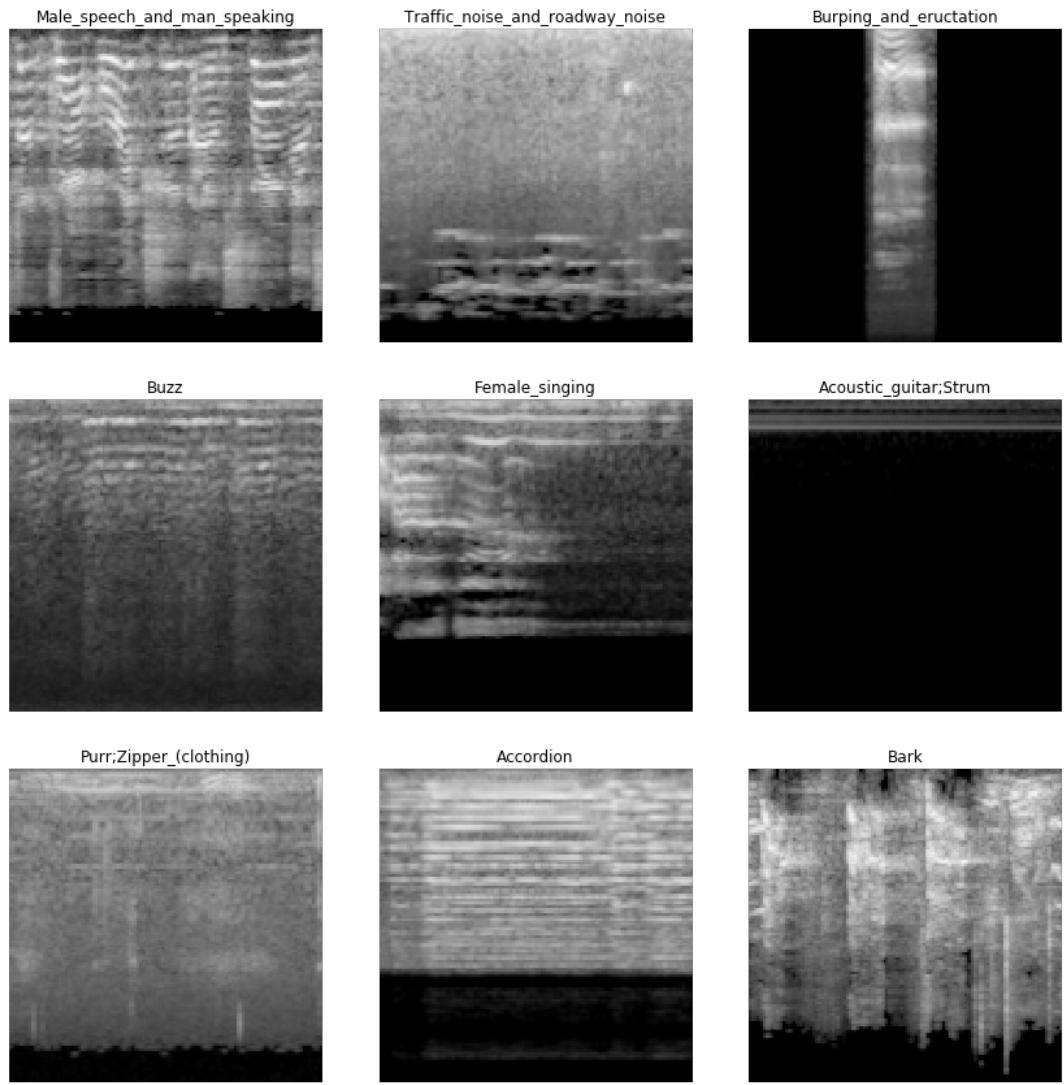
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



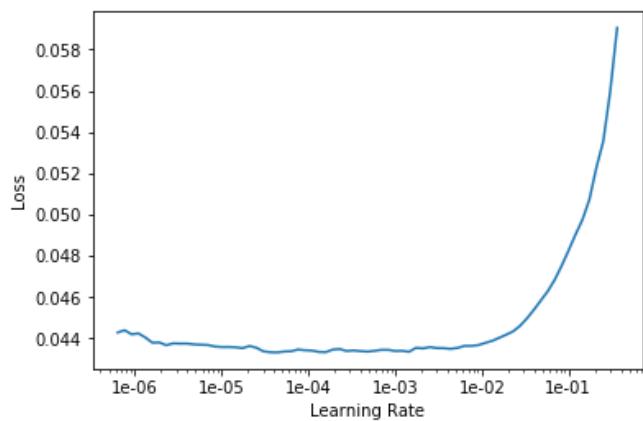
epoch	train_loss	valid_loss	lwlrap	time
0	0.042935	0.019889	0.846959	00:28
1	0.042857	0.021078	0.828129	00:28
2	0.042752	0.019367	0.841821	00:28
3	0.042803	0.019608	0.836665	00:28
4	0.042730	0.019780	0.842020	00:28
5	0.042950	0.020898	0.843485	00:28
6	0.043252	0.023353	0.808791	00:28
7	0.043668	0.022577	0.822999	00:28
8	0.044181	0.025492	0.791100	00:28
9	0.044854	0.024727	0.799859	00:28
10	0.045228	0.026019	0.778055	00:28
11	0.045954	0.029009	0.752605	00:28
12	0.046692	0.027342	0.770408	00:28
13	0.047312	0.042088	0.611700	00:28
14	0.048305	0.030053	0.737137	00:28
15	0.048603	0.029862	0.742315	00:28
16	0.048949	0.032444	0.713332	00:28
17	0.049826	0.036347	0.660451	00:28
18	0.050064	0.051834	0.470751	00:28
19	0.050558	0.038473	0.637620	00:28
20	0.050725	0.035193	0.686550	00:28
21	0.051099	0.031504	0.725979	00:28
22	0.051512	0.035345	0.686117	00:28
23	0.051593	0.031409	0.730024	00:28
24	0.051509	0.029652	0.747456	00:28
25	0.051667	0.033173	0.723042	00:28
26	0.051821	0.035785	0.686121	00:28
27	0.051404	0.032527	0.706676	00:28
28	0.051154	0.032563	0.715993	00:28
29	0.050780	0.033394	0.716970	00:28
30	0.050866	0.037839	0.649347	00:28
31	0.050718	0.034858	0.682100	00:28
32	0.050705	0.035061	0.689763	00:28
33	0.050215	0.034909	0.685357	00:28
34	0.049902	0.029711	0.745053	00:28
35	0.050060	0.029692	0.750209	00:28



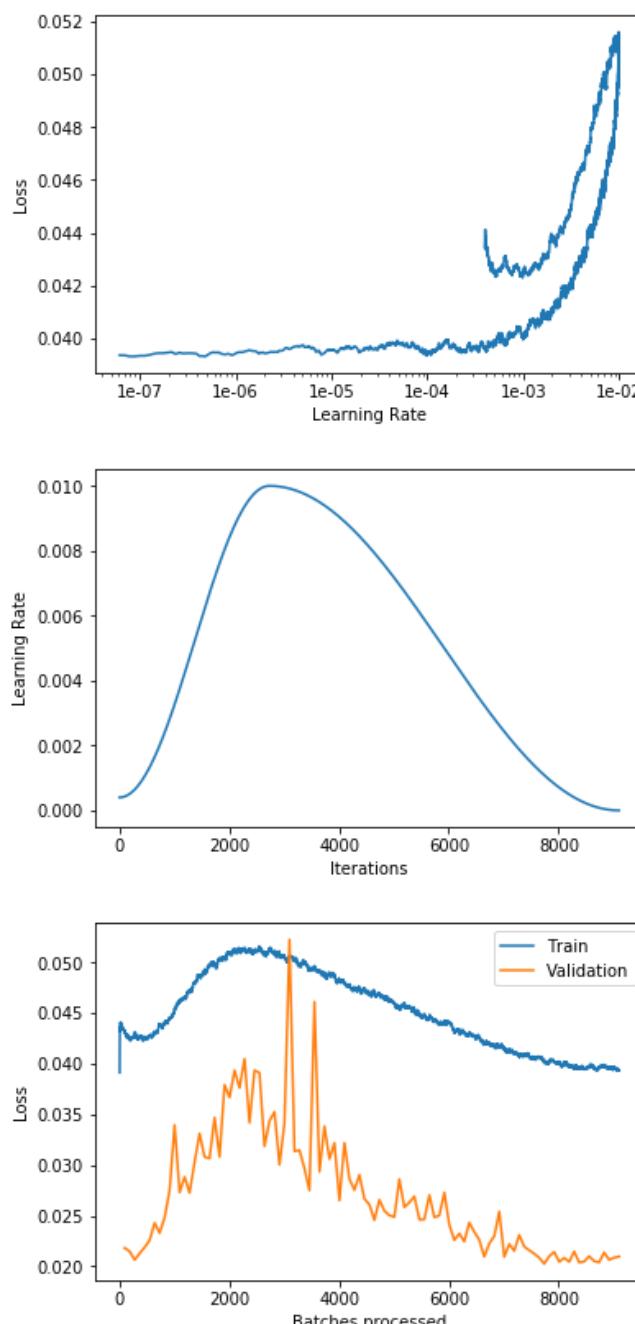
----- CURATED+NOISY+2 - Fold 2/10 -----

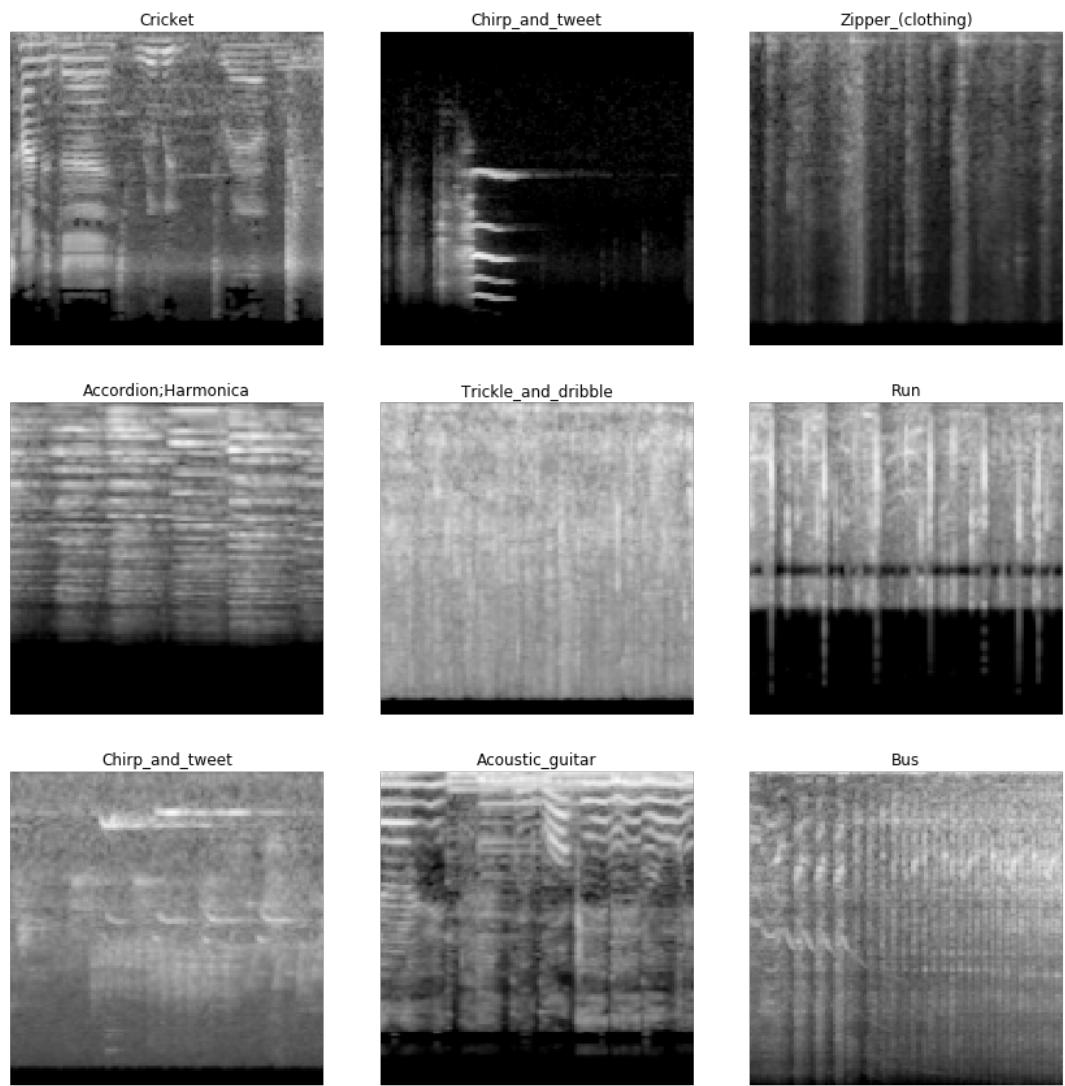


LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.

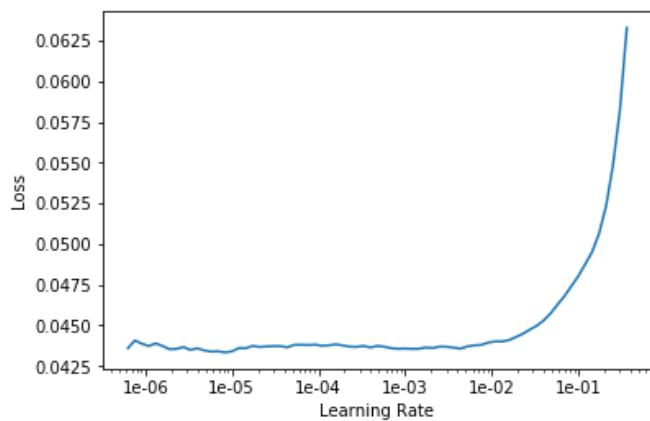


epoch	train_loss	valid_loss	lwlrap	time
0	0.043296	0.021824	0.822916	00:28
1	0.042450	0.021464	0.839787	00:28
2	0.043019	0.020633	0.839777	00:28
3	0.042656	0.021287	0.834935	00:28
4	0.042643	0.021873	0.836098	00:28
5	0.042795	0.022571	0.830352	00:28
6	0.043154	0.024288	0.807497	00:28
7	0.043886	0.023286	0.825902	00:28
8	0.044071	0.024812	0.799834	00:28
9	0.044575	0.027522	0.778919	00:28
10	0.045462	0.033959	0.688418	00:28
11	0.046235	0.027306	0.770432	00:28
12	0.046607	0.028829	0.770090	00:28
13	0.047271	0.027261	0.778266	00:28
14	0.048003	0.030274	0.744391	00:28
15	0.048629	0.033115	0.721460	00:28
16	0.049331	0.030802	0.749267	00:28
17	0.049841	0.030649	0.733798	00:28
18	0.049895	0.034691	0.694255	00:28
19	0.050413	0.030812	0.729265	00:28
20	0.050477	0.037920	0.655957	00:28
21	0.050953	0.036688	0.666488	00:28
22	0.051314	0.039353	0.626077	00:28
23	0.051411	0.037614	0.665263	00:28
24	0.051470	0.040493	0.619530	00:28
25	0.051279	0.034172	0.701660	00:28
26	0.051152	0.039364	0.648060	00:28
27	0.051535	0.039105	0.647597	00:28
28	0.050967	0.031857	0.719783	00:28
29	0.051231	0.034349	0.697021	00:28
30	0.050850	0.035259	0.688844	00:28
31	0.051026	0.030023	0.746444	00:28
32	0.050549	0.034153	0.699369	00:28
33	0.050628	0.052275	0.509485	00:28
34	0.050372	0.031370	0.742805	00:28
35	0.049701	0.031461	0.742890	00:28

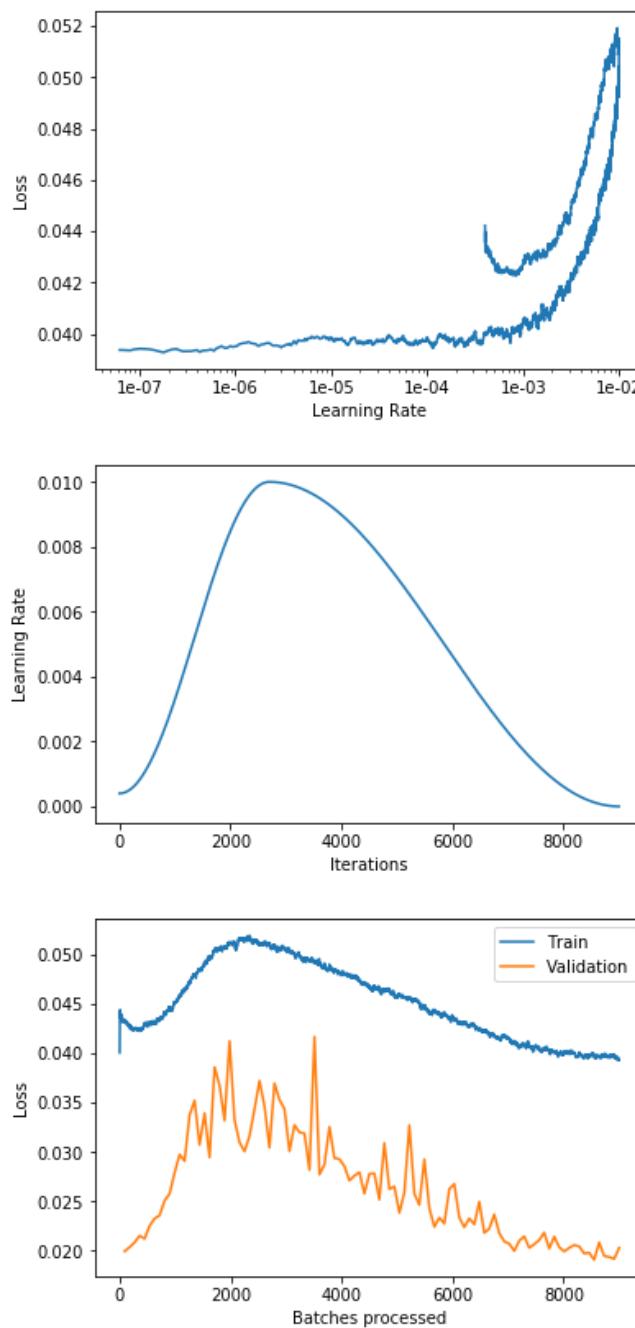




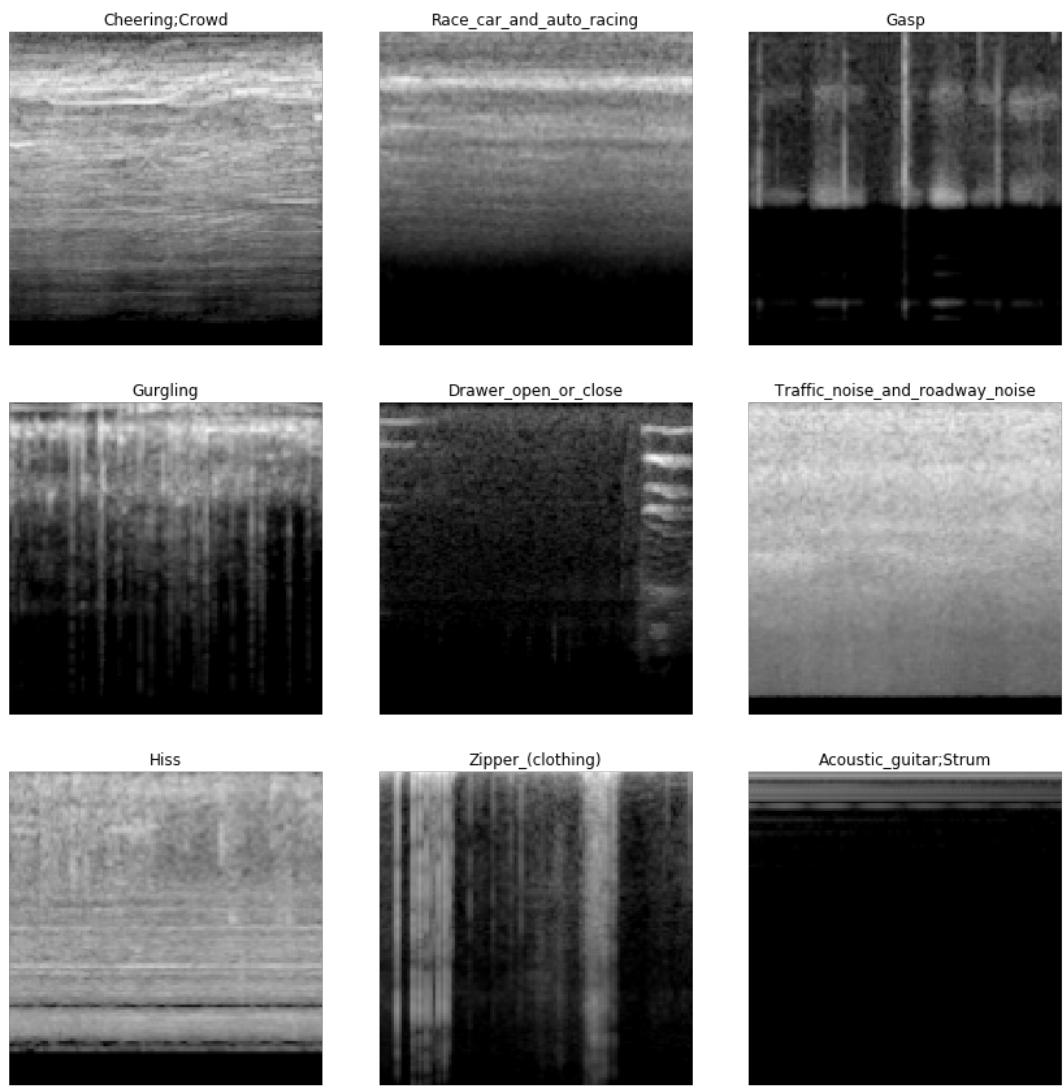
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



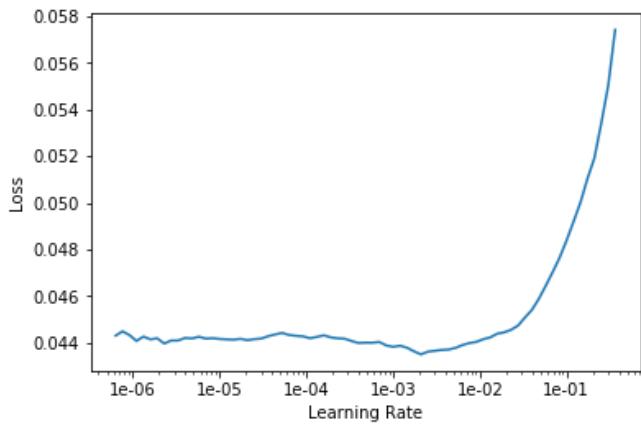
epoch	train_loss	valid_loss	lwlrap	time
0	0.043160	0.019954	0.840080	00:28
1	0.042673	0.020375	0.837184	00:28
2	0.042497	0.020862	0.841071	00:28
3	0.042451	0.021532	0.836928	00:28
4	0.042928	0.021225	0.835139	00:28
5	0.043060	0.022534	0.815963	00:28
6	0.043297	0.023265	0.802702	00:28
7	0.043472	0.023596	0.804842	00:28
8	0.044149	0.025085	0.795881	00:28
9	0.044705	0.025789	0.785652	00:28
10	0.045195	0.027920	0.757481	00:28
11	0.045846	0.029744	0.752419	00:28
12	0.046402	0.029097	0.762213	00:28
13	0.047195	0.033736	0.703361	00:28
14	0.048029	0.035216	0.672461	00:28
15	0.048453	0.030729	0.730923	00:28
16	0.049280	0.033936	0.690798	00:28
17	0.049553	0.029468	0.725918	00:28
18	0.050363	0.038603	0.625294	00:28
19	0.050560	0.036748	0.651779	00:28
20	0.050691	0.033185	0.696664	00:28
21	0.051247	0.041244	0.605095	00:28
22	0.051274	0.033251	0.693949	00:28
23	0.051533	0.030975	0.723279	00:28
24	0.051622	0.030066	0.747032	00:28
25	0.051888	0.031635	0.711386	00:28
26	0.051515	0.034351	0.697674	00:28
27	0.051073	0.037220	0.650649	00:28
28	0.051278	0.034656	0.693732	00:28
29	0.051141	0.030451	0.744778	00:28
30	0.050955	0.036959	0.656071	00:28
31	0.050495	0.035292	0.674145	00:28
32	0.050730	0.034365	0.689639	00:28
33	0.050431	0.030085	0.737648	00:28
34	0.050199	0.032725	0.702202	00:28
35	0.049892	0.032006	0.727806	00:28



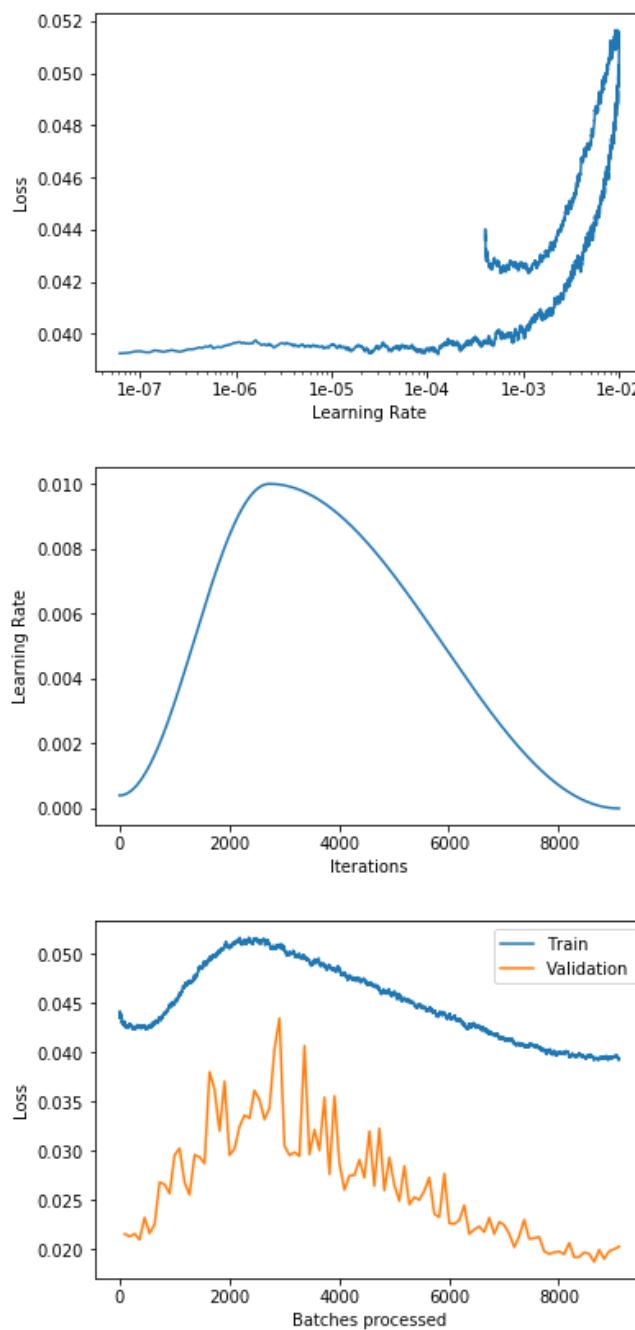
----- CURATED+NOISY+2 - Fold 4/10 -----



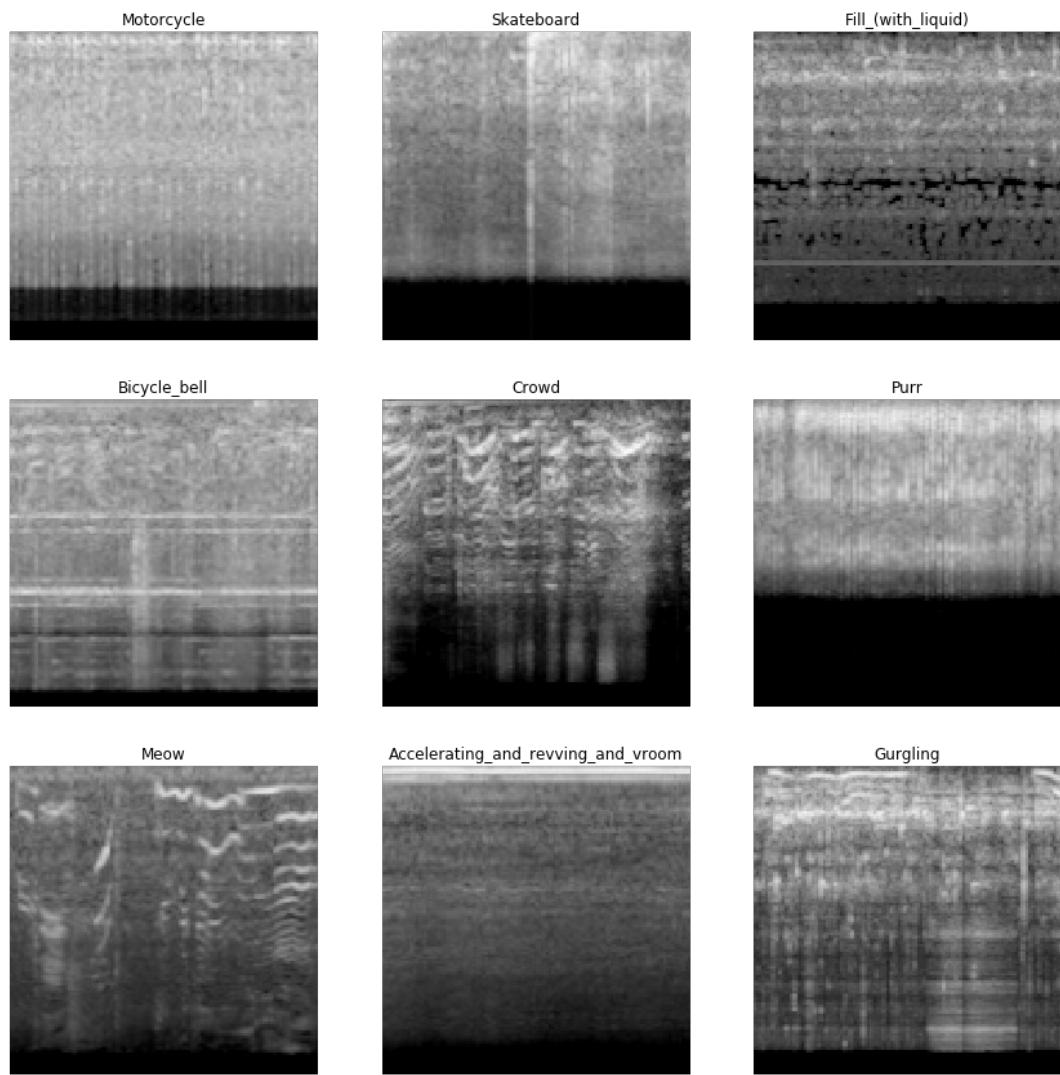
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



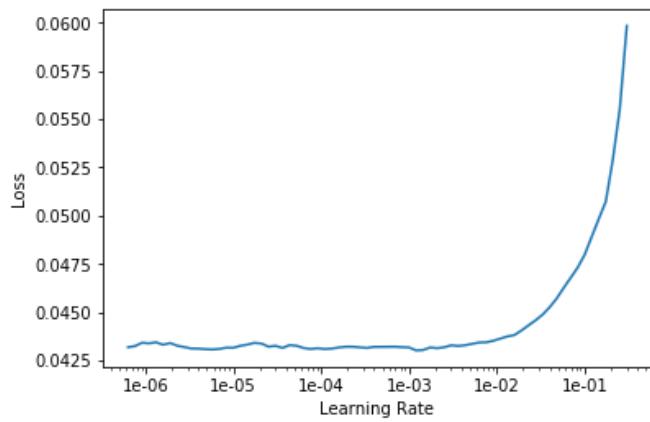
epoch	train_loss	valid_loss	lwlrap	time
0	0.042998	0.021572	0.831654	00:28
1	0.042905	0.021325	0.833628	00:28
2	0.042562	0.021585	0.828675	00:28
3	0.042623	0.021001	0.828636	00:28
4	0.042564	0.023242	0.819349	00:28
5	0.042739	0.021647	0.825810	00:28
6	0.042965	0.022543	0.824630	00:28
7	0.043444	0.026828	0.776562	00:28
8	0.044022	0.026609	0.777323	00:28
9	0.044872	0.025690	0.795780	00:28
10	0.045185	0.029578	0.742443	00:28
11	0.046019	0.030272	0.737693	00:28
12	0.046664	0.026808	0.787402	00:28
13	0.047112	0.025562	0.797055	00:28
14	0.047879	0.029596	0.752926	00:28
15	0.048527	0.029393	0.748331	00:28
16	0.049276	0.028728	0.756359	00:28
17	0.049692	0.038018	0.666330	00:28
18	0.049913	0.036256	0.671981	00:28
19	0.050185	0.032037	0.723593	00:28
20	0.050874	0.037074	0.649930	00:28
21	0.050683	0.029593	0.742233	00:28
22	0.051169	0.030219	0.738446	00:28
23	0.051570	0.032492	0.716751	00:28
24	0.051426	0.033621	0.692563	00:28
25	0.051564	0.033320	0.702181	00:28
26	0.051559	0.036134	0.676195	00:28
27	0.051233	0.035218	0.694225	00:28
28	0.051291	0.033203	0.707247	00:28
29	0.051049	0.034354	0.693870	00:28
30	0.050579	0.040322	0.628030	00:28
31	0.050719	0.043480	0.584244	00:28
32	0.050395	0.030539	0.744510	00:28
33	0.050203	0.029572	0.744629	00:28
34	0.050140	0.029861	0.741206	00:28
35	0.049958	0.029189	0.747461	00:28



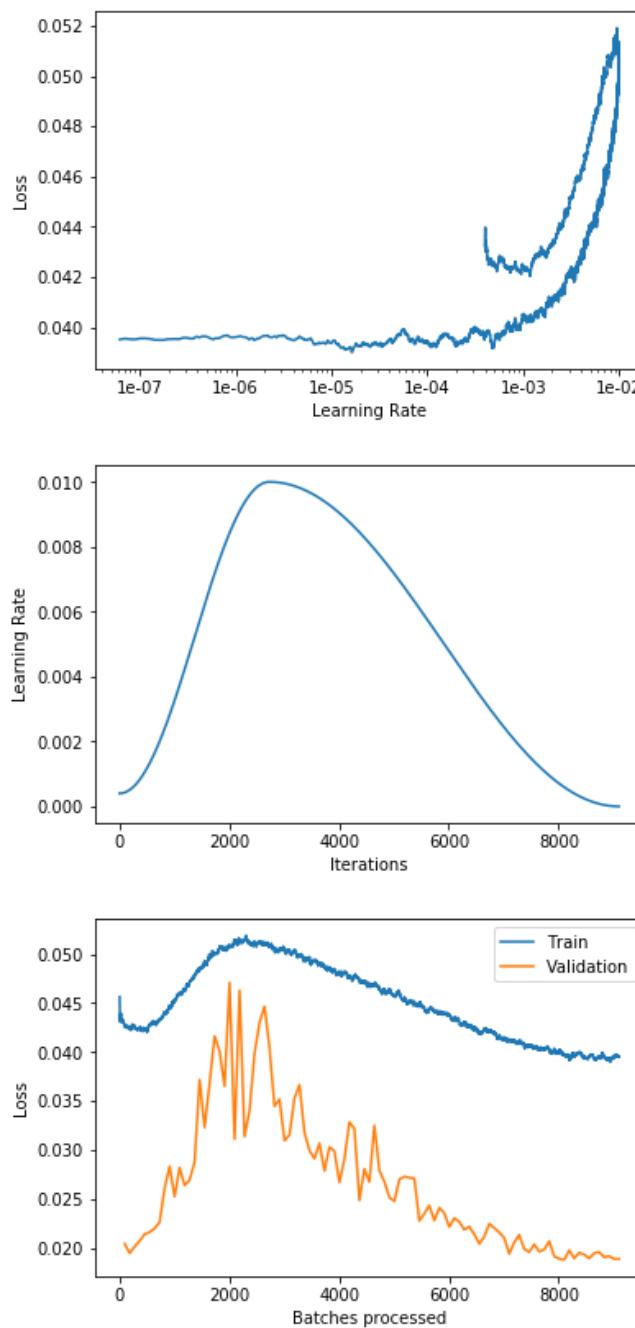
----- CURATED+NOISY+2 - Fold 5/10 -----



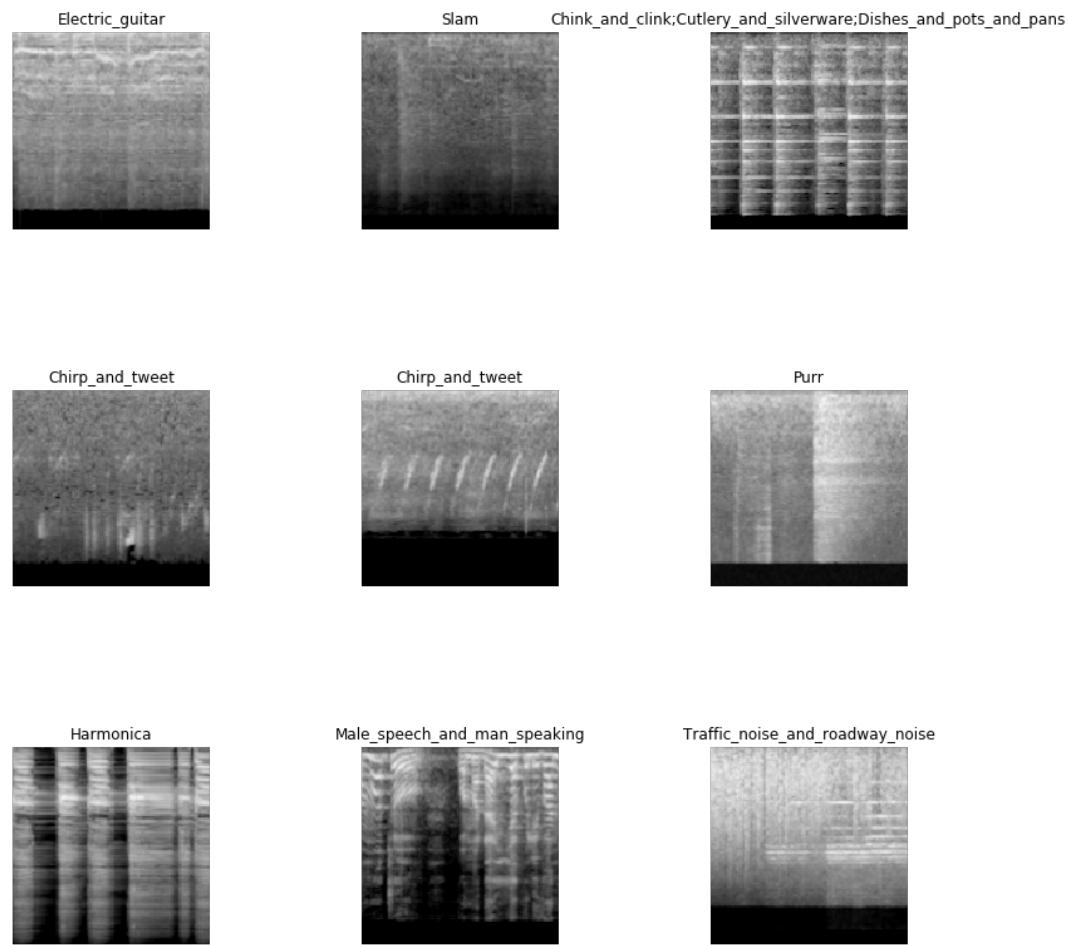
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



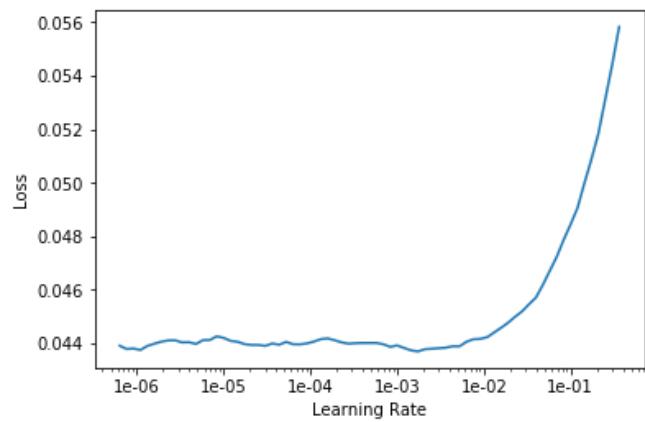
epoch	train_loss	valid_loss	lwlrap	time
0	0.042755	0.020401	0.856749	00:28
1	0.042490	0.019440	0.872099	00:28
2	0.042581	0.020097	0.849413	00:28
3	0.042420	0.020698	0.846904	00:28
4	0.042249	0.021378	0.847453	00:28
5	0.042707	0.021587	0.837580	00:28
6	0.043049	0.021965	0.833528	00:28
7	0.043333	0.022606	0.836849	00:28
8	0.044101	0.026104	0.792257	00:28
9	0.044672	0.028307	0.770110	00:28
10	0.045141	0.025230	0.817658	00:28
11	0.046022	0.028179	0.772314	00:28
12	0.046405	0.026379	0.785810	00:28
13	0.047236	0.026908	0.798164	00:28
14	0.047929	0.028760	0.771129	00:28
15	0.048625	0.037169	0.674308	00:28
16	0.049216	0.032296	0.739238	00:28
17	0.049866	0.036754	0.686192	00:28
18	0.050119	0.041625	0.638820	00:28
19	0.050638	0.040104	0.624641	00:28
20	0.050564	0.036499	0.678493	00:28
21	0.050870	0.047092	0.571763	00:28
22	0.051312	0.031116	0.740837	00:28
23	0.051520	0.046297	0.581967	00:28
24	0.051551	0.031355	0.738225	00:28
25	0.051211	0.034009	0.732576	00:28
26	0.051250	0.039839	0.642238	00:28
27	0.051182	0.043009	0.610634	00:28
28	0.051247	0.044655	0.575671	00:28
29	0.050728	0.040461	0.631406	00:28
30	0.050551	0.034455	0.697195	00:28
31	0.050561	0.035182	0.686486	00:28
32	0.050562	0.030940	0.744431	00:28
33	0.050319	0.031536	0.728306	00:28
34	0.050432	0.035303	0.687804	00:28
35	0.0519903	0.036615	0.668991	00:28



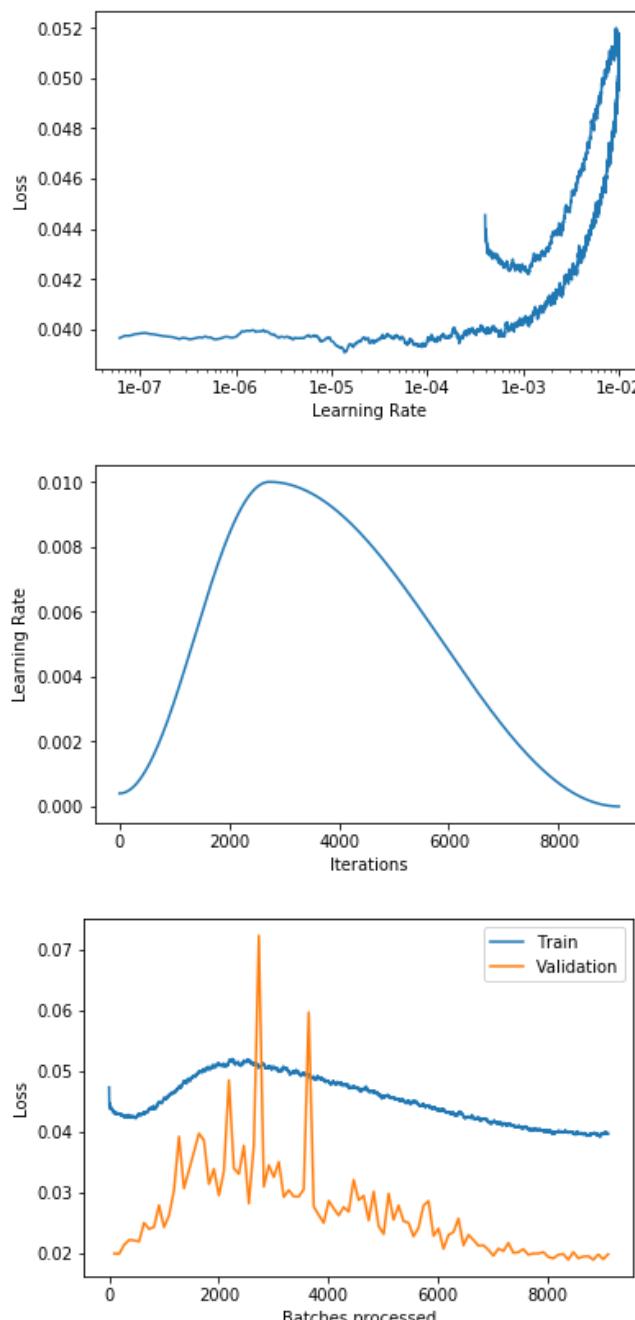
----- CURATED+NOISY+2 - Fold 6/10 -----



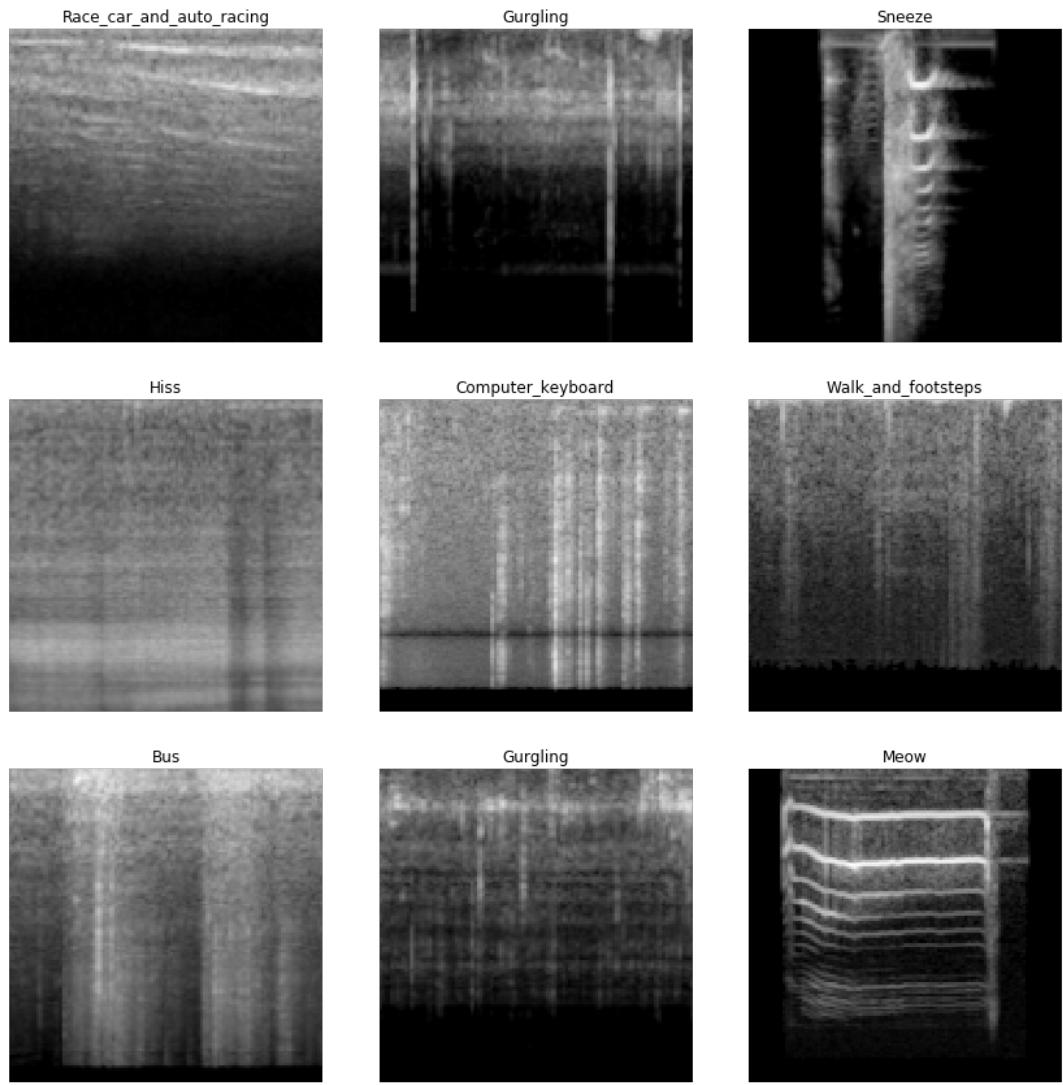
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



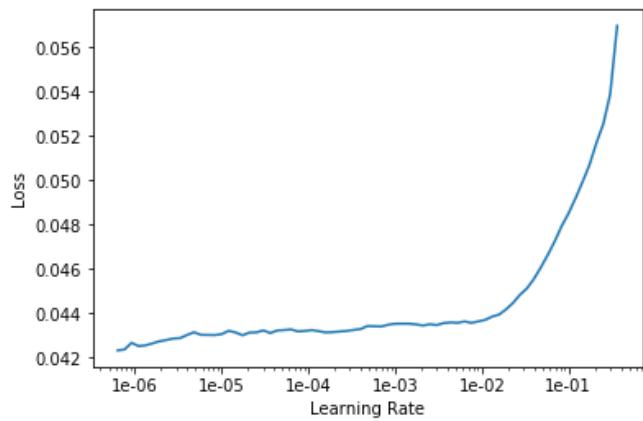
epoch	train_loss	valid_loss	lwlrap	time
0	0.043028	0.019850	0.850044	00:28
1	0.042861	0.019806	0.850240	00:28
2	0.042489	0.021277	0.842763	00:28
3	0.042480	0.022089	0.826565	00:28
4	0.042458	0.022023	0.831203	00:28
5	0.042898	0.021832	0.835009	00:28
6	0.043036	0.024928	0.796979	00:28
7	0.043696	0.023909	0.795459	00:28
8	0.044139	0.024237	0.808137	00:28
9	0.044947	0.027838	0.754402	00:28
10	0.045468	0.024207	0.805959	00:28
11	0.045979	0.026116	0.786727	00:28
12	0.046512	0.030306	0.739800	00:28
13	0.047400	0.039172	0.638611	00:28
14	0.048336	0.030631	0.742774	00:28
15	0.048776	0.033597	0.705634	00:28
16	0.049198	0.036637	0.669147	00:28
17	0.049903	0.039687	0.624980	00:28
18	0.049997	0.038575	0.641260	00:28
19	0.050565	0.031313	0.731823	00:28
20	0.050853	0.033847	0.713971	00:28
21	0.051245	0.029441	0.739342	00:28
22	0.051075	0.033442	0.717038	00:28
23	0.051720	0.048486	0.527490	00:28
24	0.051741	0.033912	0.692849	00:28
25	0.051314	0.032992	0.689345	00:28
26	0.051779	0.037679	0.657583	00:28
27	0.051761	0.028137	0.763434	00:28
28	0.051276	0.037759	0.723554	00:28
29	0.050913	0.072384	0.624538	00:28
30	0.051160	0.030885	0.734116	00:28
31	0.050818	0.034452	0.681393	00:28
32	0.050684	0.032486	0.713749	00:28
33	0.050013	0.034995	0.676827	00:28
34	0.050656	0.029194	0.754747	00:28
35	0.049988	0.030333	0.742161	00:28



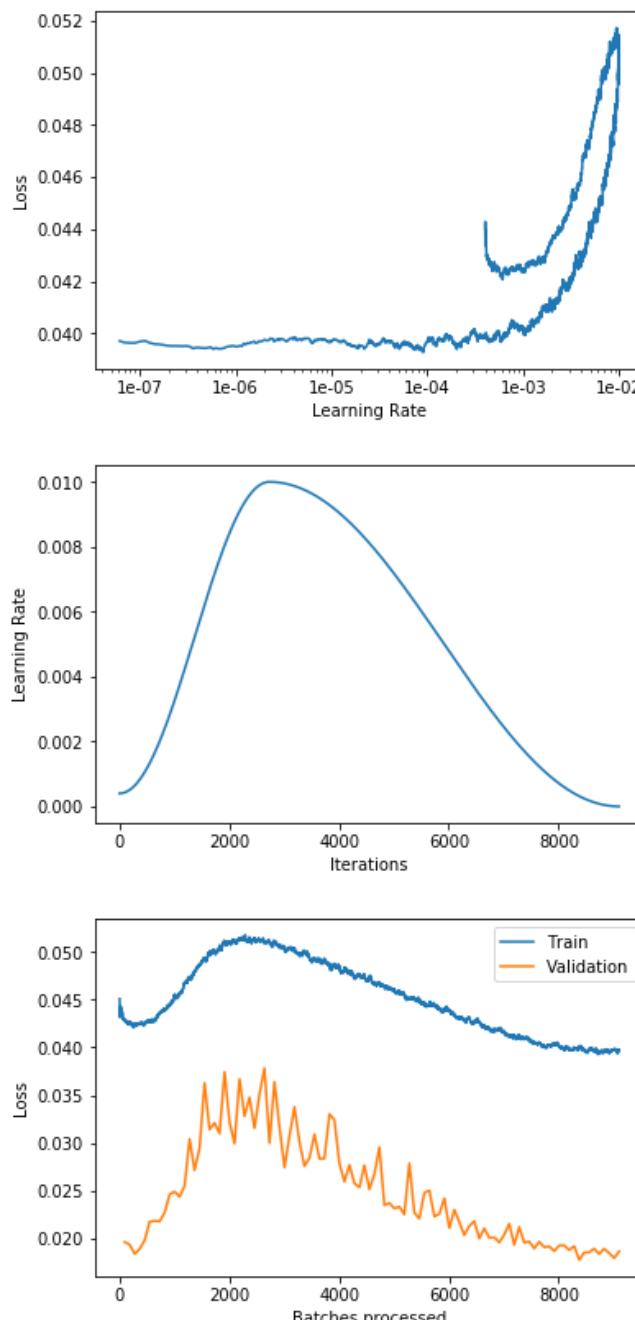
----- CURATED+NOISY+2 - Fold 7/10 -----



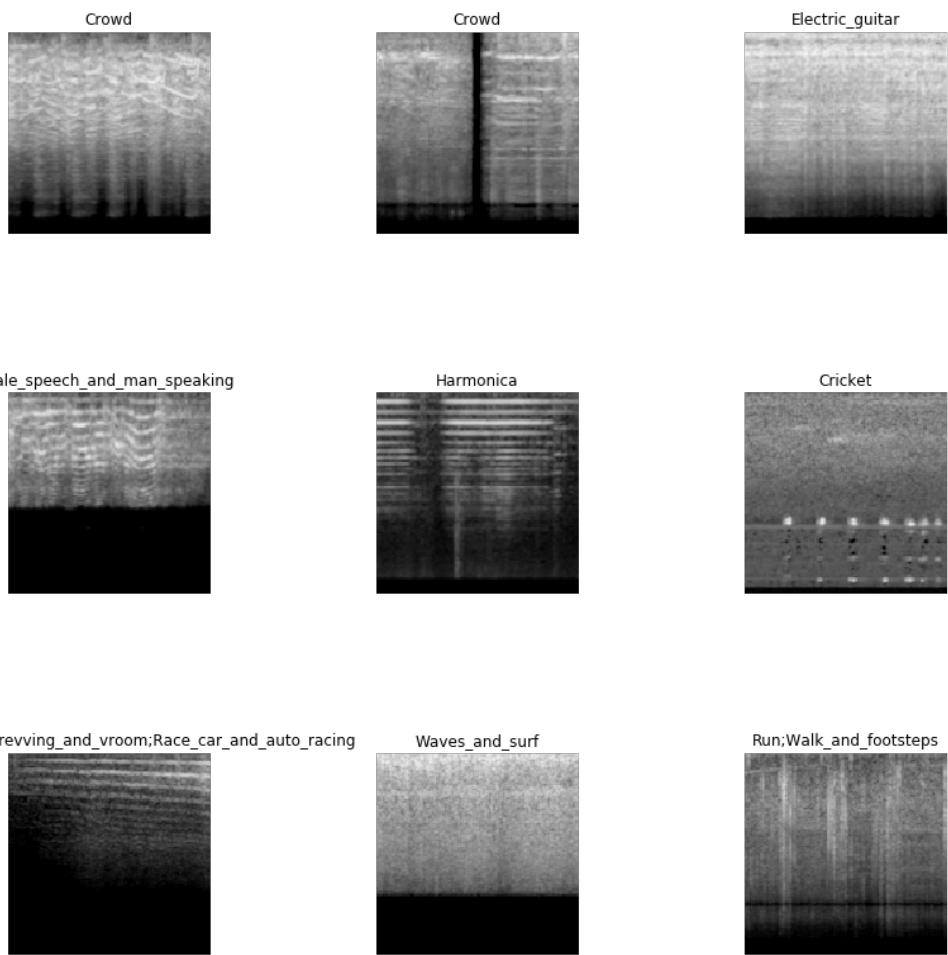
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



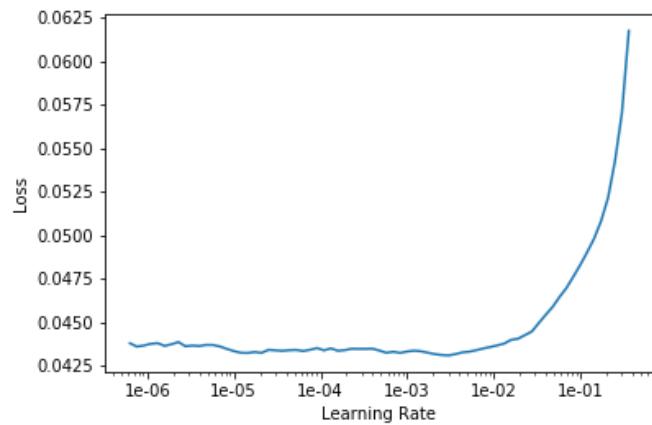
epoch	train_loss	valid_loss	lwlrap	time
0	0.042921	0.019631	0.863024	00:29
1	0.042653	0.019355	0.862083	00:29
2	0.042490	0.018391	0.865054	00:31
3	0.042402	0.018913	0.874159	00:29
4	0.042402	0.019790	0.863554	00:29
5	0.042664	0.021761	0.848285	00:29
6	0.042913	0.021835	0.834962	00:29
7	0.043549	0.021797	0.827482	00:29
8	0.044065	0.022691	0.831820	00:29
9	0.044520	0.024622	0.809264	00:29
10	0.045497	0.024911	0.809548	00:30
11	0.045624	0.024365	0.806591	00:28
12	0.046798	0.025523	0.805459	00:29
13	0.047346	0.030408	0.732632	00:30
14	0.048016	0.027162	0.793170	00:28
15	0.048732	0.029468	0.759425	00:28
16	0.049269	0.036288	0.681682	00:32
17	0.049921	0.031414	0.737845	00:30
18	0.050200	0.032103	0.726692	00:32
19	0.050372	0.030982	0.758204	00:31
20	0.051028	0.037438	0.655673	00:32
21	0.051074	0.032183	0.726456	00:31
22	0.051197	0.029944	0.764746	00:32
23	0.051315	0.036694	0.686842	00:31
24	0.051652	0.032808	0.739383	00:31
25	0.051508	0.034771	0.702387	00:31
26	0.051463	0.031549	0.745194	00:29
27	0.051215	0.035116	0.698419	00:31
28	0.051045	0.037825	0.669791	00:29
29	0.051000	0.029985	0.751436	00:28
30	0.051050	0.036408	0.697054	00:28
31	0.050694	0.031670	0.739929	00:28
32	0.050322	0.027440	0.780451	00:28
33	0.050367	0.030731	0.745478	00:29
34	0.050309	0.033766	0.706087	00:29
35	0.049710	0.030100	0.767771	00:28



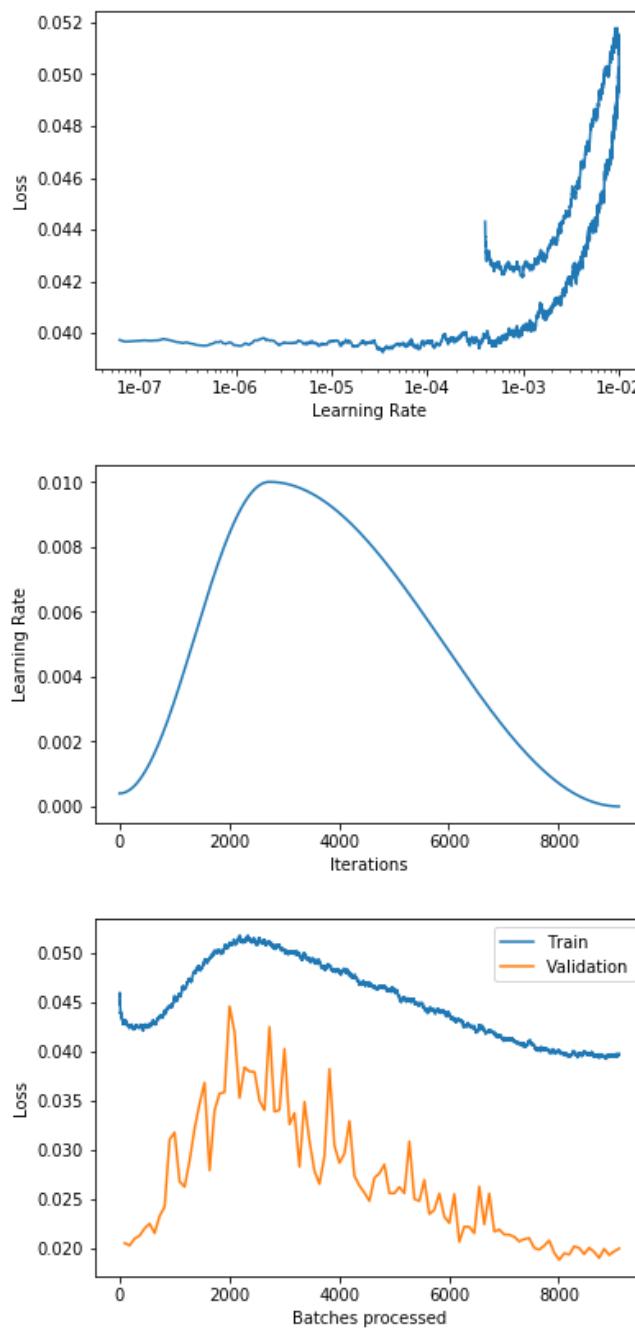
----- CURATED+NOISY+2 - Fold 8/10 -----



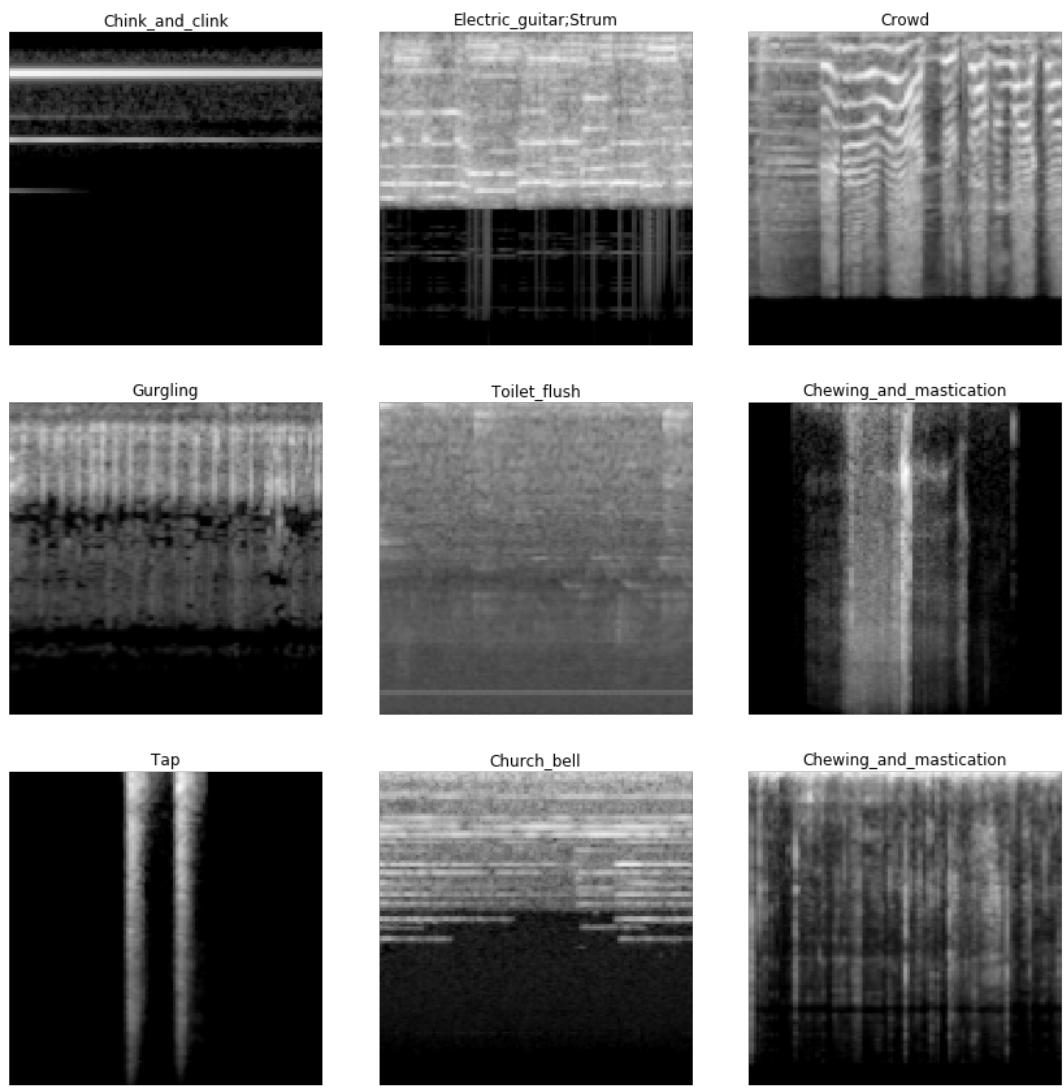
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



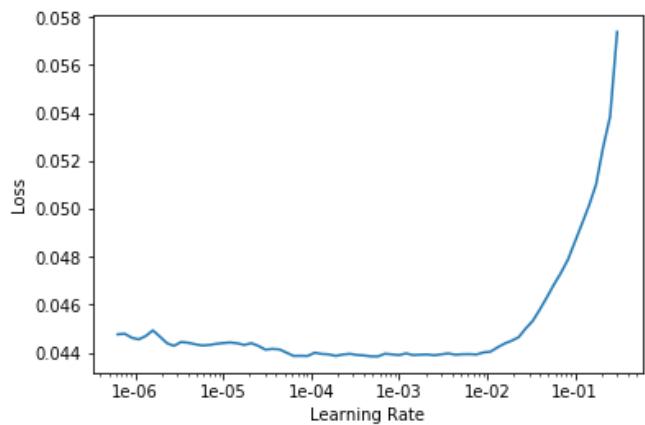
epoch	train_loss	valid_loss	lwlrap	time
0	0.043032	0.020551	0.860047	00:28
1	0.042571	0.020318	0.859572	00:28
2	0.042329	0.021011	0.843818	00:28
3	0.042456	0.021308	0.848391	00:28
4	0.042553	0.022048	0.837199	00:29
5	0.042793	0.022529	0.836762	00:28
6	0.043046	0.021558	0.853350	00:28
7	0.043575	0.023314	0.813656	00:28
8	0.044139	0.024228	0.827587	00:28
9	0.044696	0.031070	0.734769	00:28
10	0.045666	0.031786	0.727425	00:29
11	0.045737	0.026804	0.785588	00:28
12	0.046485	0.026262	0.794783	00:28
13	0.047416	0.028909	0.767331	00:28
14	0.048124	0.032166	0.727191	00:28
15	0.048620	0.034700	0.699334	00:28
16	0.049210	0.036844	0.692666	00:28
17	0.049473	0.027958	0.779530	00:29
18	0.050200	0.034029	0.701559	00:30
19	0.050257	0.035748	0.683458	00:29
20	0.050732	0.035869	0.687694	00:29
21	0.051096	0.044592	0.572166	00:29
22	0.051233	0.042013	0.600057	00:29
23	0.051695	0.035300	0.689584	00:28
24	0.051372	0.038408	0.663034	00:29
25	0.051376	0.037985	0.664494	00:28
26	0.051382	0.037942	0.659446	00:29
27	0.051457	0.035019	0.682533	00:30
28	0.051314	0.034046	0.725597	00:30
29	0.051156	0.042502	0.592032	00:31
30	0.050786	0.033899	0.704723	00:31
31	0.050862	0.034071	0.717043	00:32
32	0.050359	0.040283	0.622464	00:31
33	0.050137	0.032617	0.723067	00:32
34	0.049994	0.033754	0.690640	00:32
35	0.049956	0.028319	0.767169	00:32



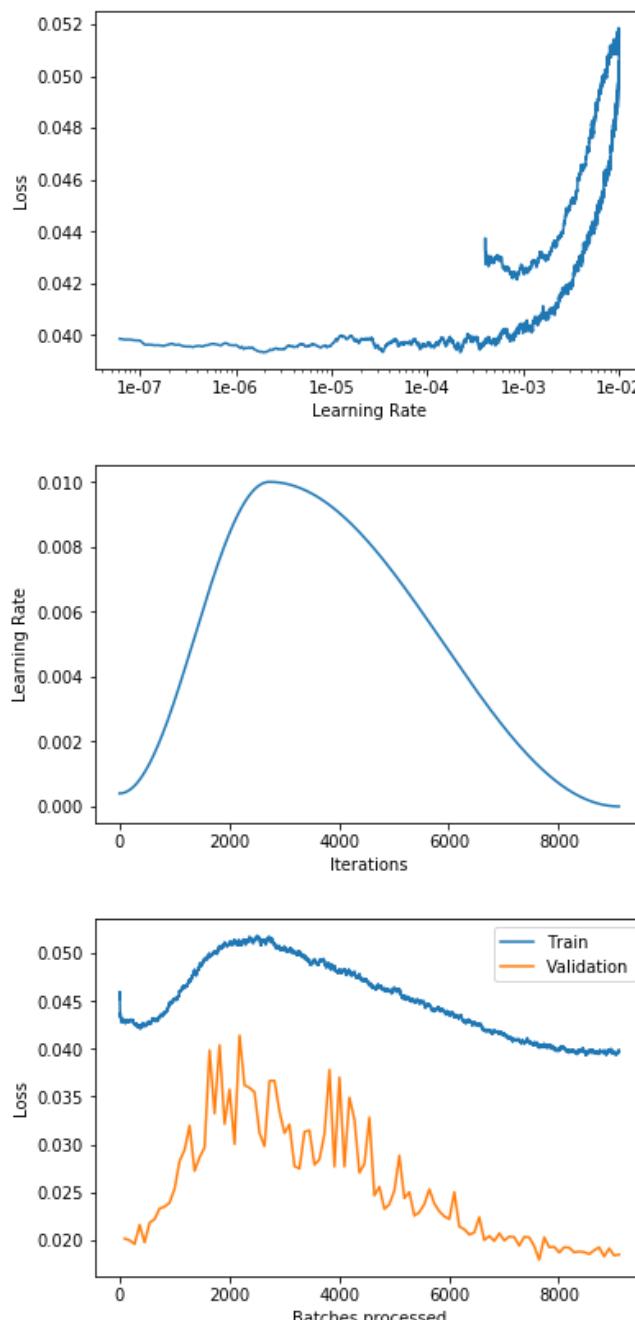
----- CURATED+NOISY+2 - Fold 9/10 -----



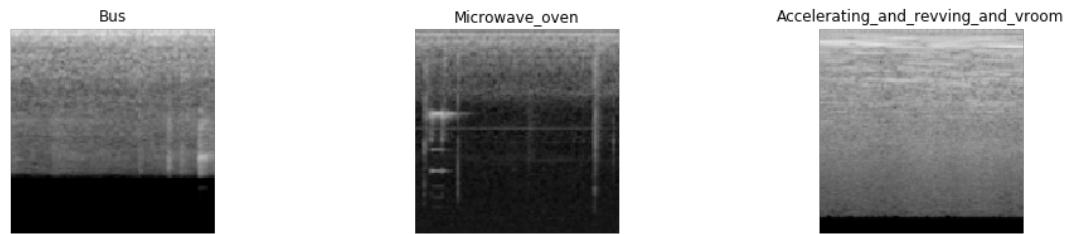
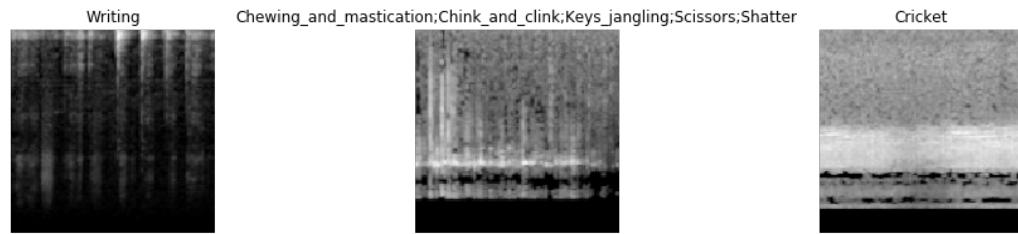
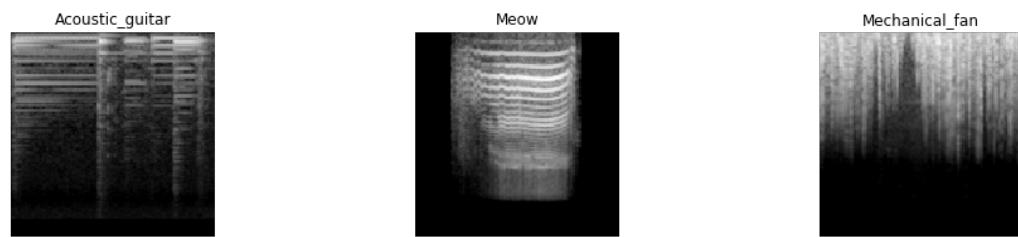
LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



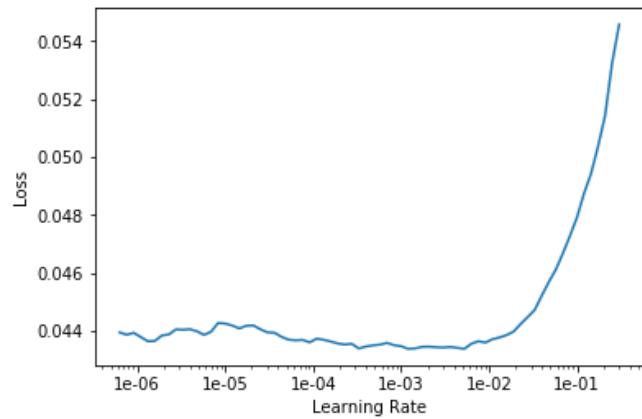
epoch	train_loss	valid_loss	lwlrap	time
0	0.043009	0.020133	0.854789	00:31
1	0.042918	0.019938	0.858473	00:31
2	0.042595	0.019562	0.855153	00:31
3	0.042383	0.021561	0.846836	00:31
4	0.042573	0.019727	0.858404	00:31
5	0.042811	0.021788	0.836047	00:32
6	0.043108	0.022167	0.827666	00:31
7	0.043314	0.023283	0.823623	00:31
8	0.043769	0.023468	0.821826	00:31
9	0.044811	0.023898	0.803154	00:31
10	0.045204	0.025353	0.795972	00:31
11	0.046000	0.028246	0.773966	00:31
12	0.046519	0.029427	0.754282	00:31
13	0.047220	0.031952	0.722419	00:31
14	0.048270	0.027219	0.779917	00:32
15	0.048968	0.028625	0.749861	00:31
16	0.049477	0.029667	0.752703	00:31
17	0.049952	0.039850	0.652449	00:31
18	0.050146	0.033221	0.696079	00:32
19	0.050553	0.040384	0.635132	00:31
20	0.051110	0.032124	0.723711	00:31
21	0.051142	0.035754	0.680192	00:31
22	0.051395	0.030025	0.765822	00:31
23	0.051046	0.041405	0.605861	00:31
24	0.051401	0.036188	0.685595	00:32
25	0.051665	0.035924	0.680799	00:31
26	0.051499	0.035460	0.697676	00:31
27	0.051504	0.031189	0.731984	00:31
28	0.051428	0.029791	0.748140	00:31
29	0.051511	0.036639	0.679768	00:31
30	0.051018	0.036667	0.670481	00:31
31	0.050638	0.033451	0.710256	00:31
32	0.050583	0.031167	0.736935	00:31
33	0.050413	0.032077	0.722083	00:31
34	0.050012	0.027706	0.773631	00:31
35	0.050203	0.027452	0.776128	00:31



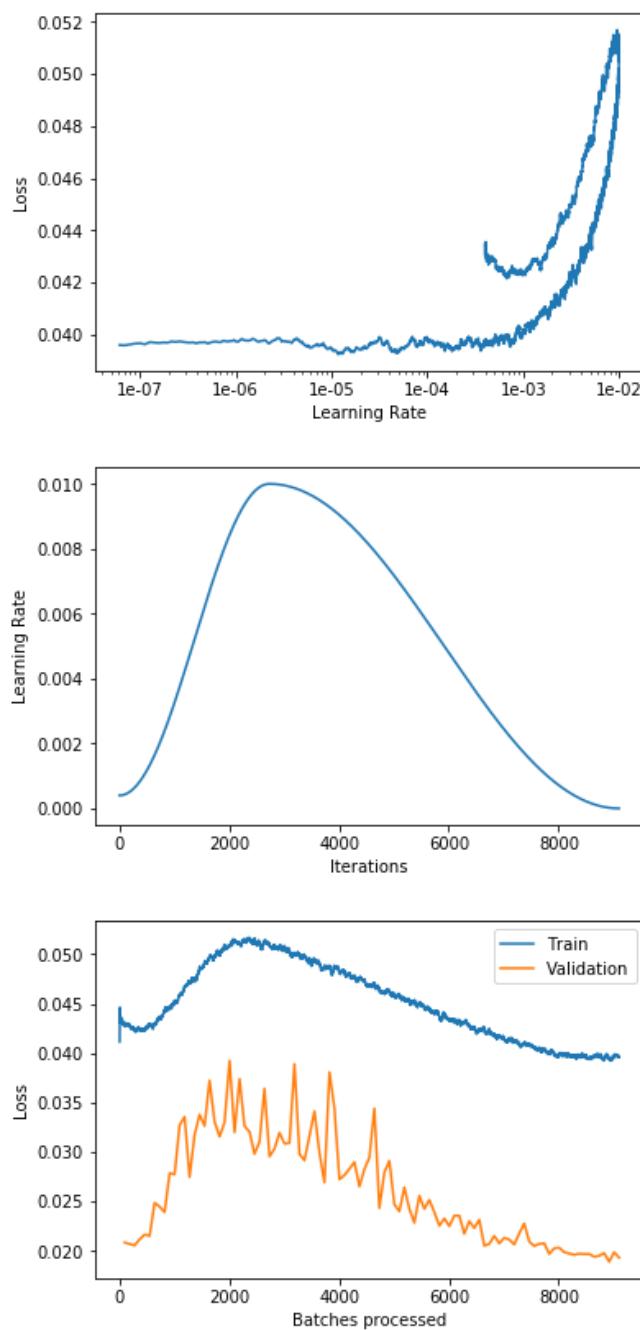
----- CURATED+NOISY+2 - Fold 10/10 -----



LR Finder is complete, type {learner\_name}.recorder.plot() to see the graph.



epoch	train_loss	valid_loss	lwlrap	time
0	0.042963	0.020839	0.835666	00:31
1	0.042799	0.020678	0.844583	00:31
2	0.042367	0.020539	0.839177	00:31
3	0.042443	0.021153	0.839919	00:31
4	0.042304	0.021635	0.830909	00:31
5	0.042901	0.021505	0.829510	00:31
6	0.042960	0.024847	0.796706	00:31
7	0.043593	0.024467	0.798468	00:31
8	0.044300	0.023894	0.812451	00:32
9	0.044604	0.027841	0.764079	00:31
10	0.045320	0.027720	0.776904	00:32
11	0.045754	0.032707	0.712146	00:31
12	0.046775	0.033547	0.705088	00:31
13	0.047280	0.027468	0.766426	00:31
14	0.047606	0.031714	0.724045	00:31
15	0.048695	0.033780	0.703577	00:31
16	0.049162	0.032628	0.725587	00:31
17	0.049639	0.037227	0.665825	00:31
18	0.049908	0.033074	0.723016	00:32
19	0.050062	0.031544	0.738019	00:31
20	0.050932	0.032985	0.721121	00:31
21	0.050912	0.039241	0.638179	00:31
22	0.051304	0.031982	0.715467	00:31
23	0.051112	0.037389	0.667927	00:31
24	0.051547	0.032603	0.714999	00:31
25	0.051620	0.032031	0.723631	00:31
26	0.051445	0.029798	0.739893	00:31
27	0.051122	0.031013	0.727657	00:31
28	0.051293	0.036411	0.665097	00:31
29	0.051019	0.029529	0.750985	00:31
30	0.050947	0.030309	0.747078	00:31
31	0.050878	0.031950	0.727903	00:32
32	0.050752	0.030838	0.742977	00:31
33	0.050296	0.030898	0.730990	00:31
34	0.050100	0.038889	0.641486	00:31
35	0.050031	0.029776	0.750872	00:31



```
In [45]: kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-11_fold-{fold}')
```

Overall lwlrap on CURATED dataset: 0.8846200848932246

	lwlrap	weight
Fill_(with_liquid)	0.658388	0.008693
Squeak	0.679500	0.013039
Mechanical_fan	0.687030	0.008519
Walk_and_footsteps	0.720895	0.013039
Hiss	0.755910	0.013039
Tap	0.770000	0.013039
Male_speech_and_manSpeaking	0.771817	0.013039
Chink_and_clink	0.775800	0.013039
Cutlery_and_silverware	0.777220	0.013039
Buzz	0.778739	0.009736
Traffic_noise_and_roadway_noise	0.793735	0.013039
Yell	0.798571	0.013039
Stream	0.798816	0.013039
Bus	0.803657	0.013039
Water_tap_and_faucet	0.807834	0.013039
Bathtub_(filling_or_washing)	0.819831	0.013039
Dishes_and_pots_and_pans	0.820404	0.013039
Sink_(filling_or_washing)	0.821253	0.013039
Motorcycle	0.825778	0.013039
Frying_(food)	0.833333	0.010953
Slam	0.840221	0.013039
Trickle_and_dribble	0.842163	0.009214
Accelerating_and_revving_and_vroom	0.846482	0.013039
Clapping	0.849807	0.013039
Run	0.849889	0.013039
Crowd	0.854015	0.013039
Scissors	0.869365	0.013039
Drip	0.870667	0.013039
Sneeze	0.870807	0.010953
Waves_and_surf	0.871017	0.013039
...	...	...
Raindrop	0.922032	0.013039
Gasp	0.924319	0.008345
Child_speech_and_kidSpeaking	0.924421	0.013039
Screaming	0.928148	0.013039

```
In [46]: kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-11_fold-{fold}')
```

Overall lwlrapp on NOISY dataset: 0.5823532924290014

	<b>lwlrap</b>	<b>weight</b>
Burping_and_eructation	0.075615	0.0125
Raindrop	0.125448	0.0125
Finger_snapping	0.146021	0.0125
Cupboard_open_or_close	0.198613	0.0125
Sigh	0.201656	0.0125
Zipper_(clothing)	0.217091	0.0125
Keys_jangling	0.221410	0.0125
Writing	0.239949	0.0125
Gasp	0.241696	0.0125
Buzz	0.258949	0.0125
Tap	0.287101	0.0125
Scissors	0.296071	0.0125
Fart	0.304466	0.0125
Computer_keyboard	0.353252	0.0125
Shatter	0.356479	0.0125
Tick-tock	0.405760	0.0125
Strum	0.406395	0.0125
Knock	0.413641	0.0125
Chewing_and_mastication	0.453551	0.0125
Sneeze	0.468797	0.0125
Squeak	0.482955	0.0125
Bicycle_bell	0.499331	0.0125
Chink_and_clink	0.504634	0.0125
Drip	0.508558	0.0125
Skateboard	0.515089	0.0125
Toilet_flush	0.515176	0.0125
Fill_(with_liquid)	0.516785	0.0125
Slam	0.526764	0.0125
Microwave_oven	0.535447	0.0125
Purr	0.537506	0.0125
...	...	...
Printer	0.685713	0.0125
Screaming	0.688348	0.0125
Marimba_and_xylophone	0.695730	0.0125
Clapping	0.705704	0.0125

```
In [47]: # m5 = lwlrap_per_sample('stage-11_fold-{fold}', kf_noisy, trn_noisy_df)

In [48]: # (m5.lwlrap < .5).sum(), (m5.lwlrap >= .5).sum(), (m5.lwlrap == 1).sum()

In [49]: # ok_noisy_items3 = m5[m5.lwlrap == 1].index
# len(ok_noisy_items3)

In [50]: # for fold, ((train_index1, valid_index1), (train_index2, valid_index2)) in enumerate(zip(kf_curated.split(trn_curated_df), kf_noisy.split(trn_noisy_df))):
#     print('-' * 40, f'CURATED+NOISY+3 - Fold {fold+1}/{n_splits}', '-' * 40)
#
#     train_index2 = list(set(train_index2).intersection(set(ok_noisy_items3)))
#
#     mix_df = pd.concat([
#         trn_curated_df.iloc[train_index1],
#         trn_noisy_df.iloc[train_index2],
#         trn_curated_df.iloc[valid_index1],
#         # #         trn_noisy_df.iloc[valid_index2], # compute lwlrap only on curated
#         # ignore_index=True)
#         # train_index = mix_df[:len(train_index1)+len(train_index2)].index
#         # valid_index = mix_df[len(train_index1)+len(train_index2):].index
#
#         # src = (ImageList.from_df(mix_df, WORK)
#         #         .split_by_idxs(train_index, valid_index)
#         #         .label_from_df(label_delim=','))
#     ])
#     data = (src.transform(tfms, size=128)
#             .databunch(bs=bs))
#     )
#     data.show_batch(3)
#     plt.show()
#
#     learn.load(f'stage-11_fold-{fold}')
#     learn.data = data
#
#     learning(learn, 100, 1e-2)
#
#     learn.save(f'stage-12_fold-{fold}')
#     learn.export(f'stage-12_fold-{fold}.pkl')
#
#     if TOY_MODE:
#         break

In [52]: # kfold_lwlrap('CURATED', kf_curated, trn_curated_df, 'stage-12_fold-{fold}')

In [ ]: # kfold_lwlrap('NOISY', kf_noisy, trn_noisy_df, 'stage-12_fold-{fold}')
```

## Test prediction and submission file creation

- Switch to test data.
- Overwrite results to sample submission; simple way to prepare submission file.

```
In [ ]: X_test = pickle.load(open(MELS_TEST, 'rb'))
CUR_X_FILES, CUR_X = list(test_df.fname.values), X_test
```

```
In [ ]: output = StringIO()
csv_writer = writer(output)
csv_writer.writerow(test_df.columns)

for _, row in tqdm_notebook(test_df.iterrows(), total=test_df.shape[0]):
    idx = CUR_X_FILES.index(row.fname)
    time_dim = CUR_X[idx].shape[1]
    s = math.ceil((time_dim-conf.n_mels) / TTA_SHIFT) + 1

    fname = row.fname
    for crop_x in [int(np.around((time_dim-conf.n_mels)*x/(s-1))) if s != 1 else 0 for x in range(s)]:
        row.fname = fname + '!' + str(crop_x)
        csv_writer.writerow(row)

output.seek(0)
test_df_multi = pd.read_csv(output)
```

```
In [ ]: test = ImageList.from_df(test_df_multi, WORK)
```

```
In [ ]: for fold in range(n_splits):
    learn = load_learner(WORK, f'stage-12_fold-{fold}.pkl', test=test)
    preds, _ = learn.get_preds(ds_type=DatasetType.Test)
    preds = preds.cpu().numpy()
#    preds = pd.DataFrame(preds).rank(1) / preds.shape[1]
    if fold == 0:
        predictions = preds
    else:
        predictions += preds
    if TOY_MODE:
        break
predictions /= n_splits
```

```
In [ ]: test_df_multi[learn.data.classes] = predictions
test_df_multi['fname'] = test_df_multi.fname.apply(lambda x: x.split('!')[0])
```

```
In [ ]: test_df_multi.head()
```

```
In [ ]: submission = test_df_multi.infer_objects().groupby('fname').mean().reset_index()
```

```
In [ ]: submission.to_csv('submission.csv', index=False)
submission.head()
```

```
In [ ]: submission.set_index('fname').idxmax(1)
```

```
In [1]: import os
from pathlib import Path
import pickle
import random
import time
from io import StringIO
from csv import writer
import gc

import numpy as np
import pandas as pd
import librosa
import librosa.display
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
import IPython
import IPython.display
# import PIL

import torch
import torch.nn as nn
import torch.nn.functional as F

from fastai import *
from fastai.vision import *
from fastai.vision.data import *
```

```
In [2]: # start_time = time.time()
```

```
In [3]: models_list = (
    (Path('../input/fat2019ssl4multistage/work/work'), 'stage-2_fold-{fold}.pkl'),
    (Path('../input/fat2019ssl4multistage/work/work'), 'stage-10_fold-{fold}.pkl'),
    (Path('../input/fat2019ssl4multistage/work/work'), 'stage-11_fold-{fold}.pkl'),
    (Path('../input/fat2019ssl8vgg16full/work/work'), 'stage-2_fold-{fold}.pkl'),
    (Path('../input/fat2019ssl8vgg16full/work/work'), 'stage-10_fold-{fold}.pkl'),
    (Path('../input/fat2019ssl8vgg16full/work/work'), 'stage-11_fold-{fold}.pkl')
)
```

```
In [4]: TTA_SHIFT = 48 # TTA: predict every TTA_SHIFT
n_splits = 10
DATA = Path('../input/freesound-audio-tagging-2019')
DATA_TEST = DATA/'test'
CSV_SUBMISSION = DATA/'sample_submission.csv'
test_df = pd.read_csv(CSV_SUBMISSION)
```

```
In [5]: # # visually check everything is present in datasets
# for work, name in models_list:
#     print('*'*10, work, name, '*'*10)
#     !ls {work}
```

```
In [6]: def read_audio(conf, pathname, trim_long_data):
    y, sr = librosa.load(pathname, sr=conf.sampling_rate)
    # trim silence
    if 0 < len(y): # workaround: 0 length causes error
        y, _ = librosa.effects.trim(y) # trim, top_db=default(60)
    # make it unified length to conf.samples
    if len(y) > conf.samples: # long enough
        if trim_long_data:
            y = y[0:0+conf.samples]
    else: # pad blank
        padding = conf.samples - len(y)      # add padding at both ends
        offset = padding // 2
        y = np.pad(y, (offset, conf.samples - len(y) - offset), 'constant')
    return y

def audio_to_melspectrogram(conf, audio):
    spectrogram = librosa.feature.melspectrogram(audio,
                                                sr=conf.sampling_rate,
                                                n_mels=conf.n_mels,
                                                hop_length=conf.hop_length,
                                                n_fft=conf.n_fft,
                                                fmin=conf.fmin,
                                                fmax=conf.fmax)
    spectrogram = librosa.power_to_db(spectrogram)
    spectrogram = spectrogram.astype(np.float32)
    return spectrogram

def show_melspectrogram(conf, mels, title='Log-frequency power spectrogram'):
    librosa.display.specshow(mels, x_axis='time', y_axis='mel',
                            sr=conf.sampling_rate, hop_length=conf.hop_length,
                            fmin=conf.fmin, fmax=conf.fmax)
    plt.colorbar(format='%+2.0f dB')
    plt.title(title)
    plt.show()

def read_as_melspectrogram(conf, pathname, trim_long_data, debug_display=False):
    x = read_audio(conf, pathname, trim_long_data)
    mels = audio_to_melspectrogram(conf, x)
    if debug_display:
        IPython.display.display(IPython.display.Audio(x, rate=conf.sampling_rate))
    show_melspectrogram(conf, mels)
    return mels

class conf:
    # Preprocessing settings
    sampling_rate = 44100
    duration = 2
    hop_length = 347*duration # to make time steps 128
    fmin = 20
    fmax = sampling_rate // 2
    n_mels = 128
    n_fft = n_mels * 20
    samples = sampling_rate * duration

# example
# x = read_as_melspectrogram(conf, DATA_CURATED/'0006ae4e.wav', trim_long_data=False, debug_display=True)
```

```
In [7]: def mono_to_color(X, mean=None, std=None, norm_max=None, norm_min=None, eps=1e-6):
    # Stack X as [X,X,X]
    X = np.stack([X, X, X], axis=-1)

    # Standardize
    mean = mean or X.mean()
    std = std or X.std()
    Xstd = (X - mean) / (std + eps)
    _min, _max = Xstd.min(), Xstd.max()
    norm_max = norm_max or _max
    norm_min = norm_min or _min
    if (_max - _min) > eps:
        # Scale to [0, 255]
        V = Xstd
        V[V < norm_min] = norm_min
        V[V > norm_max] = norm_max
        V = 255 * (V - norm_min) / (norm_max - norm_min)
        V = V.astype(np.uint8)
    else:
        # Just zero
        V = np.zeros_like(Xstd, dtype=np.uint8)
    return V

def convert_wav_to_image(df, source, img_dest):
    print(f'Converting {source} -> {img_dest}')
    X = []
    for i, row in tqdm_notebook(df.iterrows(), total=df.shape[0]):
        x = read_as_melspectrogram(conf, source=str(row.fname), trim_long_data=False)
        x_color = mono_to_color(x)
        X.append(x_color)
    #     pickle.dump(X, open(img_dest, 'wb'))
    return X
```

```
In [8]: X_test = convert_wav_to_image(test_df, source=DATA_TEST, img_dest=None)
```

```
Converting ../input/freesound-audio-tagging-2019/test -> None
```

```
In [9]: class MyMixUpCallback(LearnerCallback):
    def __init__(self, learn:Learner):
        super().__init__(learn)
        self.num_mask=2
        self.masking_max_percentage=0.25

    def on_batch_begin(self, last_input, last_target, train, **kwargs):
        if not train: return

        shuffle = torch.randperm(last_target.size(0)).to(last_input.device)
        x1, y1 = last_input[shuffle], last_target[shuffle]

        batch_size, channels, height, width = last_input.size()
        h_percentage = np.random.uniform(low=0., high=self.masking_max_percentage, size=batch_size)
        w_percentage = np.random.uniform(low=0., high=self.masking_max_percentage, size=batch_size)
        #     alpha = self.num_mask * (h_percentage + w_percentage) - (self.num_mask*self.num_mask) * ((h_percentage * w_percentage))
        alpha = (h_percentage + w_percentage) - (h_percentage * w_percentage)
        alpha = last_input.new(alpha)
        alpha = alpha.unsqueeze(1)

        new_input = last_input.clone()

        for i in range(batch_size):
            h_mask = int(h_percentage[i] * height)
            h = int(np.random.uniform(0.0, height - h_mask))
            new_input[i, :, h:h + h_mask, :] = x1[i, :, h:h + h_mask, :]

            w_mask = int(w_percentage[i] * width)
            w = int(np.random.uniform(0.0, width - w_mask))
            new_input[i, :, :, w:w + w_mask] = x1[i, :, :, w:w + w_mask]

        #     new_target = torch.max(last_target, y1)
        new_target = (1-alpha) * last_target + alpha*y1
        return {'last_input': new_input, 'last_target': new_target}
```

```
In [10]: # from official code https://colab.research.google.com/drive/1AgPdhSp7ttY1803fEOH0QKlt_3HJDLi8#scrollTo=cRCaC1b9ogU
def _one_sample_positive_class_precisions(scores, truth):
    """Calculate precisions for each true class for a single sample.

    Args:
        scores: np.array of (num_classes,) giving the individual classifier scores.
        truth: np.array of (num_classes,) bools indicating which classes are true.

    Returns:
        pos_class_indices: np.array of indices of the true classes for this sample.
        pos_class_precisions: np.array of precisions corresponding to each of those classes.
    """
    num_classes = scores.shape[0]
    pos_class_indices = np.flatnonzero(truth > 0)
    # Only calculate precisions if there are some true classes.
    if not len(pos_class_indices):
        return pos_class_indices, np.zeros(0)
    # Retrieval list of classes for this sample.
    retrieved_classes = np.argsort(scores)[::-1]
    # class_rankings[top_scoring_class_index] == 0 etc.
    class_rankings = np.zeros(num_classes, dtype=np.int)
    class_rankings[pos_class_indices] = range(num_classes)
    # Which of these is a true label?
    retrieved_class_true = np.zeros(num_classes, dtype=np.bool)
    retrieved_class_true[class_rankings[pos_class_indices]] = True
    # Num hits for every truncated retrieval list.
    retrieved_cumulative_hits = np.cumsum(retrieved_class_true)
    # Precision of retrieval list truncated at each hit, in order of pos_label
    precision_at_hits = (
        retrieved_cumulative_hits[class_rankings[pos_class_indices]] /
        (1 + class_rankings[pos_class_indices].astype(np.float)))
    return pos_class_indices, precision_at_hits

def calculate_per_class_lwlrap(truth, scores):
    """Calculate label-weighted label-ranking average precision.

    Arguments:
        truth: np.array of (num_samples, num_classes) giving boolean ground-truth
               of presence of that class in that sample.
        scores: np.array of (num_samples, num_classes) giving the classifier-uncertainty's
               real-valued score for each class for each sample.

    Returns:
        per_class_lwlrap: np.array of (num_classes,) giving the lwlrap for each class.
        weight_per_class: np.array of (num_classes,) giving the prior of each class within the truth labels. Then the overall unbalanced lwlrap is simply np.sum(per_class_lwlrap * weight_per_class)
    """
    assert truth.shape == scores.shape
    num_samples, num_classes = scores.shape
    # Space to store a distinct precision value for each class on each sample.
    # Only the classes that are true for each sample will be filled in.
    precisions_for_samples_by_classes = np.zeros((num_samples, num_classes))
```

```
In [11]: class Lwlrap(Callback):

    def on_epoch_begin(self, **kwargs):
        self.accumulator = lwlrap_accumulator()

    def on_batch_end(self, last_output, last_target, **kwargs):
        self.accumulator.accumulate_samples(last_target.cpu().numpy(), torch.sigmoid(last_output).cpu().numpy())

    def on_epoch_end(self, last_metrics, **kwargs):
        return add_metrics(last_metrics, self.accumulator.overall_lwlrap())
```

```
In [12]: class ConvBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()

        self.conv1 = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, 3, 1, 1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(out_channels, out_channels, 3, 1, 1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )

        self._init_weights()

    def _init_weights(self):
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.kaiming_normal_(m.weight)
                if m.bias is not None:
                    nn.init.zeros_(m.bias)
            elif isinstance(m, nn.BatchNorm2d):
                nn.init.constant_(m.weight, 1)
                nn.init.zeros_(m.bias)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = F.avg_pool2d(x, 2)
        return x

class Classifier(nn.Module):
    def __init__(self, num_classes=1000): # <===== modificaition to comply fast.ai
        super().__init__()

        self.conv = nn.Sequential(
            ConvBlock(in_channels=3, out_channels=64),
            ConvBlock(in_channels=64, out_channels=128),
            ConvBlock(in_channels=128, out_channels=256),
            ConvBlock(in_channels=256, out_channels=512),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1)) # <===== modificaition to comply fast.ai
        self.fc = nn.Sequential(
            nn.Dropout(0.2),
            nn.Linear(512, 128),
            nn.PReLU(),
            nn.BatchNorm1d(128),
            nn.Dropout(0.1),
            nn.Linear(128, num_classes),
        )

    def forward(self, x):
        x = self.conv(x)
        #x = torch.mean(x, dim=3) # <===== modificaition to comply fast.ai
        #x, _ = torch.max(x, dim=2) # <===== modificaition to comply fast.ai
        x = self.avgpool(x) # <===== modificaition to comply fast.ai
        x = self.fc(x)
        return x
```

```
In [13]: # !!! use globals CUR_X_FILES, CUR_X
def open_fat2019_image(fn, convert_mode, after_open)->Image:
    # open
    fname = fn.split('/')[-1]
    if '!' in fname:
        fname, crop_x = fname.split('!')
        crop_x = int(crop_x)
    else:
        crop_x = -1
    idx = CUR_X_FILES.index(fname)
    x = CUR_X[idx]
    # crop
    base_dim, time_dim, _ = x.shape
    if crop_x == -1:
        crop_x = random.randint(0, time_dim - base_dim)
    x = x[0:base_dim, crop_x:crop_x+base_dim, :]
    x = np.transpose(x, (1, 0, 2))
    x = np.transpose(x, (2, 1, 0))
    # standardize
    return Image(torch.from_numpy(x.astype(np.float32, copy=False)).div_(255))

vision.data.open_image = open_fat2019_image
```

```
In [14]: CUR_X_FILES, CUR_X = list(test_df.fname.values), X_test
```

```
In [15]: output = StringIO()
csv_writer = writer(output)
csv_writer.writerow(test_df.columns)

for _, row in tqdm_notebook(test_df.iterrows(), total=test_df.shape[0]):
    idx = CUR_X_FILES.index(row.fname)
    time_dim = CUR_X[idx].shape[1]
    s = math.ceil((time_dim-conf.n_mels) / TTA_SHIFT) + 1

    fname = row.fname
    for crop_x in [int(np.around((time_dim-conf.n_mels)*x/(s-1))) if s != 1 else 0 for x in range(s)]:
        row.fname = fname + '!' + str(crop_x)
        csv_writer.writerow(row)

output.seek(0)
test_df_multi = pd.read_csv(output)

del row, test_df, output, csv_writer; gc.collect();
```

```
In [16]: test = ImageList.from_df(test_df_multi, models_list[0][0])

for model_nb, (work, name) in enumerate(models_list):
    for fold in range(n_splits):
        learn = load_learner(work, name.format(fold=fold), test=test)
        preds, _ = learn.get_preds(ds_type=DatasetType.Test)
        preds = preds.cpu().numpy()
        if (fold == 0) and (model_nb == 0):
            predictions = preds
        else:
            predictions += preds

predictions /= (n_splits * len(models_list))
```

54.59% [113/207 00:09<00:07]

```
In [17]: test_df_multi[learn.data.classes] = predictions
test_df_multi['fname'] = test_df_multi.fname.apply(lambda x: x.split('!')[0])
```

```
In [18]: submission = test_df_multi.infer_objects().groupby('fname').mean().reset_index()
```

```
In [19]: submission.to_csv('submission.csv', index=False)
submission.head()
```

Out[19]:

	fname	Accelerating_and_revving_and_vroom	Accordion	Acoustic_guitar	Applause	
0	000ccb97.wav	0.001941	0.000921	0.002814	0.001162	0.00
1	0012633b.wav	0.262763	0.001583	0.001652	0.003722	0.00
2	001ed5f1.wav	0.002876	0.001479	0.001354	0.005268	0.00
3	00294be0.wav	0.000540	0.000288	0.001012	0.000787	0.00
4	003fde7a.wav	0.001529	0.001032	0.002001	0.000763	0.00

```
In [20]: submission.set_index('fname').idxmax(1)
```

```
Out[20]: fname
000ccb97.wav          Bass_drum
0012633b.wav          Motorcycle
001ed5f1.wav          Run
00294be0.wav          Purr
003fde7a.wav          Bicycle_bell
0040ccc9.wav          Electric_guitar
0046b732.wav          Meow
004f3bbc.wav          Bass_guitar
00526050.wav          Bass_drum
00559da4.wav          Zipper_(clothing)
00582bbe.wav          Knock
0064aedf.wav          Drawer_open_or_close
0065512b.wav          Slam
006a91d2.wav          Stream
006ea9ee.wav          Strum
006f9dca.wav          Water_tap_and_faucet
007450dc.wav          Screaming
00979c8a.wav          Purr
00992464.wav          Fill_(with_liquid)
00b44a8a.wav          Cutlery_and_silverware
00bfaaaaf.wav         Gasp
00cf9680.wav          Stream
00eae94d.wav          Bathtub_(filling_or_washing)
011ec5b8.wav          Drawer_open_or_close
013200dc.wav          Fart
01440c2e.wav          Harmonica
01567a3d.wav          Microwave_oven
0163d203.wav          Purr
017c24b2.wav          Meow
01a992c5.wav          Cricket
...
406306b6.wav          Strum
406a1cc1.wav          Knock
406ade5e.wav          Trickle_and_dribble
4093913f.wav          Toilet_flush
40957335.wav          Gasp
4098028d.wav          Computer_keyboard
40afd1b3.wav          Whispering
40c1ed87.wav          Keys_jangling
40c50046.wav          Computer_keyboard
40d0726d.wav          Car_passing_by
40dd1274.wav          Female_speech_and_womanSpeaking
40e25750.wav          Burping_and_eructation
410837eb.wav          Acoustic_guitar
410c2138.wav          Toilet_flush
411993fd.wav          Hi-hat
413b5427.wav          Harmonica
41522a65.wav          Chirp_and_tweet
415f4c8f.wav          Cutlery_and_silverware
41759973.wav          Stream
419271cc.wav          Bicycle_bell
4195cb1a.wav          Marimba_and_xylophone
41af688a.wav          Fart
41c60bba.wav          Microwave_oven
41cfee6d.wav          Accordion
41f16193.wav          Slam
41f1ea4a.wav          Traffic_noise_and_roadway_noise
41f86bc4.wav          Toilet_flush
4215309a.wav          Scissors
4248d196.wav          Applause
42542036.wav          Race_car_and_auto_racing
Length: 1120, dtype: object
```

```
In [21]: # print('Done in', time.time() - start_time, 'seconds')
```