Deep learning for Classifying Audio of Babies crying

sisil mehta
Feb 25, 2019 · 3 min read

(Reach out for collaborating on projects)

In this post I'll talk about using deep learning to help classify audio into categories. As an example I'll be trying the task of classifying sounds of a baby crying. Not only was this a fun exercise in using CNNs for audio classification, it could also be of practical use in building out a monitor to inform parents that their baby is crying.

Building a dataset

First step to building a classifier was getting a useable dataset. I couldn't find a readily available dataset to build a model. So I decided to build one from data I collected from the internet. I used the Donate-A-Cry corpus for positive samples of sounds of babies crying. The dataset has about ~1000 sound clips of 7 secs in length. The corpus didn't have negative samples, so I used the Environmental-Sound-Classification(ESC50) dataset which has about ~2000 samples of 5 secs length. The ECS50 dataset also contains sounds of babies crying, so be sure to remove those clips from the negative samples. Divide the dataset into test and validation (90:10) into their respective folders.

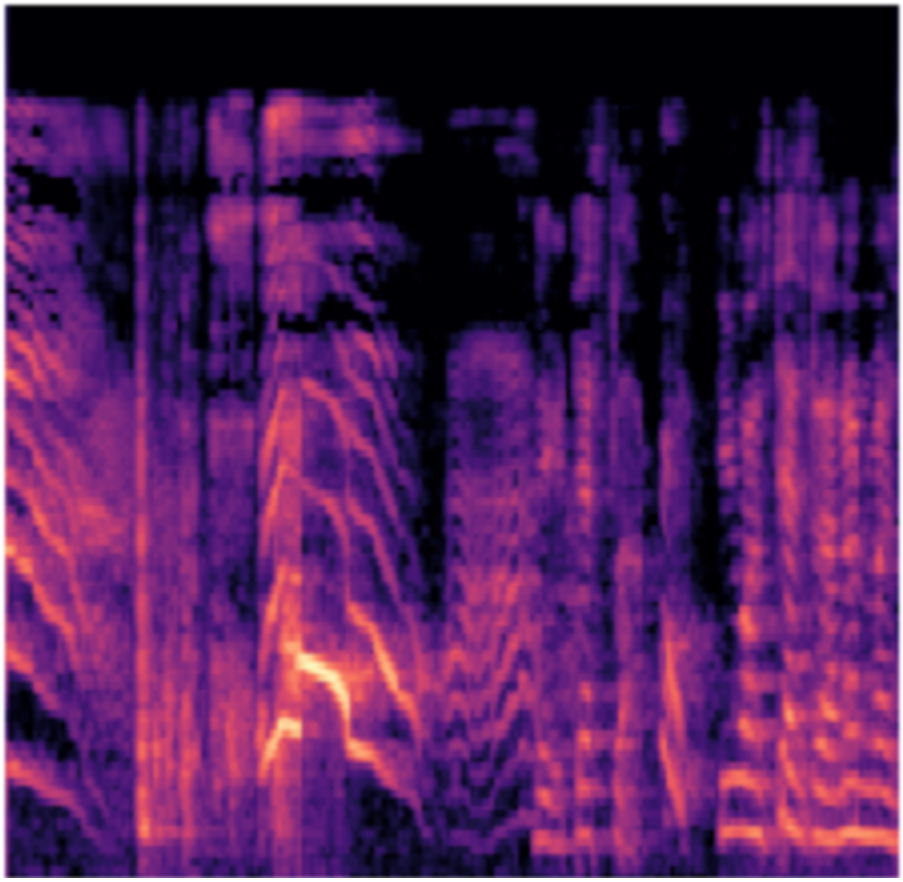
Building a model

The audio clips have a sample rate of 16000 Hz and a duration of about ~7 secs. This means there are about 16000*7 numbers per second representing the audio data. We take a fast fourier transform (FFT) of a 2048 sample window, slide it by 512 samples and repeat the process of the 7 sec clip. The resulting representation can be shown as a 2D image and is called a Short-Time Fourier Transform (STFT). Since humans perceive sound on a logarithmic scale, we'll convert the STFT to the mel scale. The librosa library lets us load an audio file and convert it to a melspectrogram

```
samples, sample_rate = librosa.load(fname)

fig = plt.figure(figsize=[4,4])
ax = fig.add_subplot(111)
ax.axes.get_xaxis().set_visible(False)
ax.axes.get_yaxis().set_visible(False)
ax.set_frame_on(False)
S = librosa.feature.melspectrogram(y=samples, sr=sample_rate)
librosa.display.specshow(librosa.power_to_db(S, ref=np.max))
```

audio-spectrogram hosted with ❤ by **GitHub** **view raw**

The melspectrogram of a baby crying looks like the image below



I'm using the FastAI v1 library to train the neural network. In order to build the spectrograms of the audio samples needed for training the model, we'll be using the fantastic audio loader module for fastai v1 built by Jason Hartquist.

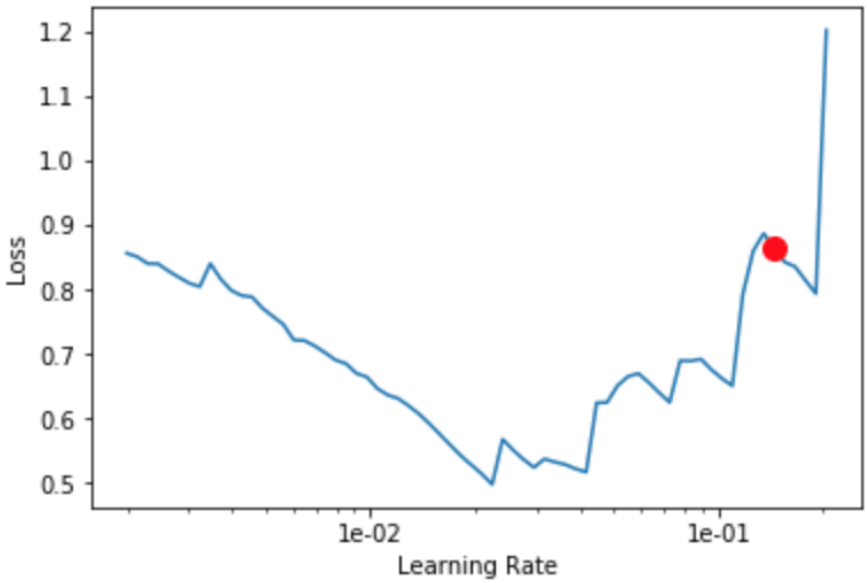
```
n_fft = 2048 # output of fft will have shape [1024 x n_frames]
n_hop = 512  # 50% overlap between frames
n_mels = 128 # compress 2048 dimensions to 128 via mel frequency scale
sample_rate = 16000
```

```
                                        n_mels=n_mels, sample_rate=sample_rate)

batch_size = 64
data = (AudioItemList.from_folder(CRYING_PATH)
        .split_by_folder()
        .label_from_folder()
        .databunch(bs=batch_size, tfms=tfms, equal_lengths=False))

learn = create_cnn(data, models.resnet34, metrics=accuracy)

learn.lr_find(start_lr=0.001, end_lr=1)
learn.recorder.plot()
```

audio-model hosted with ❤ by **GitHub** **view raw**

Fastai's cyclical learning rate finder runs the model against a small batch of training samples to find a good learning rate.

```
LR Finder is complete, type {learner_name}.recorder.plot() to see t
he graph.
Min numerical gradient: 1.45E-01
```



As the learning rate increases to 10e-2, you can see the model loss decrease. However, for higher learning rates, the loss begins to increase. Hence we pick 10e-2 as the learning rate for training the model.

After training the model over a few epochs, we see an accuracy of 95% over the validation set

```
learn.unfreeze()
```

Total time: 00:36

epoch	train_loss	valid_loss	accuracy
1	0.537459	0.416604	0.872000
2	0.438086	0.255379	0.902000

```
lr = 0.001
lrs = np.array([lr/9, lr/3, lr])

learn.fit(epochs=2, lr=lrs)
```

Total time: 00:48

epoch	train_loss	valid_loss	accuracy
1	0.156069	0.514582	0.874000
2	0.136377	0.128923	0.950000

Predicting over realtime audio samples

Now that we have a really good model, in order to use it in a real application, we need to be able to run predictions over an audio stream in real time.

We use the pyaudio library to read audio samples from the device microphone and then convert the audio data into numpy arrays and feed it to the model.

The above code reads a 7 sec audio clip from the microphone and loads that into memory. It converts it to a numpy array and runs the model on them to get a prediction. This simple piece of code is now ready to be deployed to a service or an embedded device and be used in real applications !

Machine Learning Deep Learning Audio Classification Classification Neural Networks