# Parallelizing NN for sound classification

HPPL Project

Svetlana Pavlova

# Motivation

**Neural Networks (NNs)** have proven very effective in image classification tasks, which gave rise to the design of various architectures.

NNs achieve state of the art results on image classification tasks and offer a variety of ready to use pre trained backbones.

Instead of directly using the sound file as an amplitude vs time signal it is wished to convert the audio signal into an image - spectrogram.

# Data overview - ESC-50 dataset

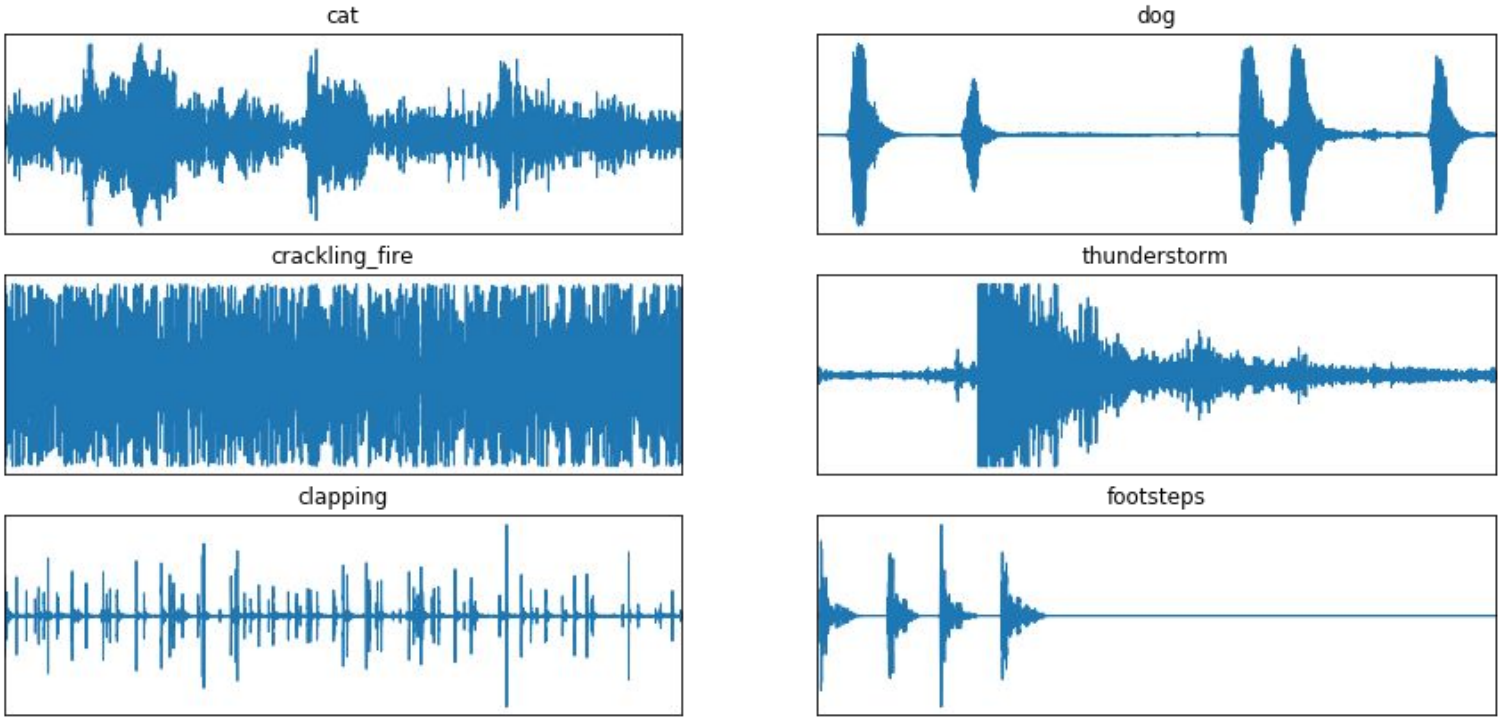
| Animals | Natural soundscapes & water sounds | Human, non-speech sounds | Interior/domestic sounds | Exterior/urban noises |
|---|---|---|---|---|
| Dog | Rain | Crying baby | Door knock | Helicopter |
| Rooster | Sea waves | Sneezing | Mouse click | Chainsaw |
| Pig | Crackling fire | Clapping | Keyboard typing | Siren |
| Cow | Crickets | Breathing | Door, wood creaks | Car horn |
| Frog | Chirping birds | Coughing | Can opening | Engine |
| Cat | Water drops | Footsteps | Washing machine | Train |
| Hen | Wind | Laughing | Vacuum cleaner | Church bells |
| Insects (flying) | Pouring water | Brushing teeth | Clock alarm | Airplane |
| Sheep | Toilet flush | Snoring | Clock tick | Fireworks |
| Crow | Thunderstorm | Drinking, sipping | Glass breaking | Hand saw |

2000 environmental audio recordings

5-second-long recordings

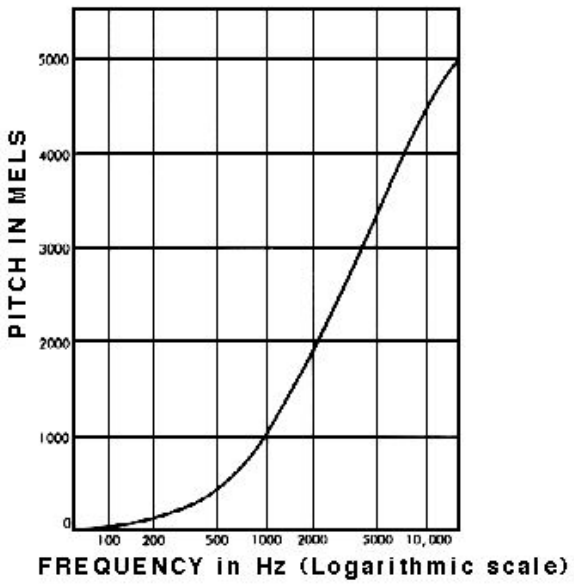
50 semantical classes (with 40 examples per class)

# Data overview – waveplots



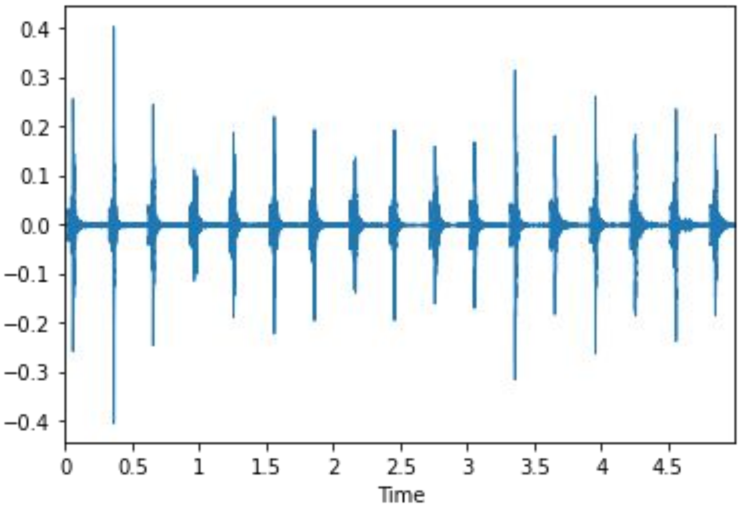
# Mel scale

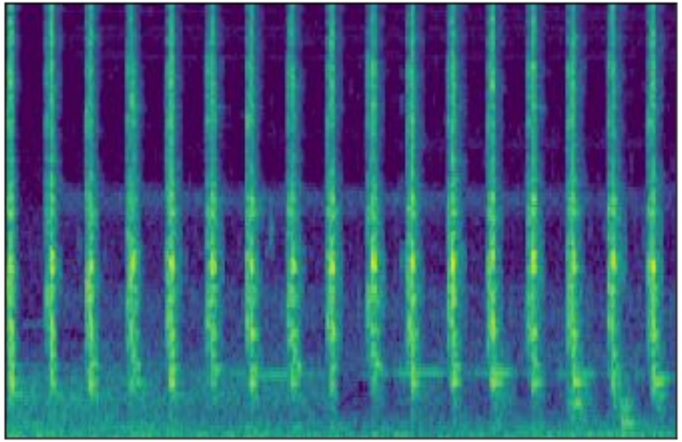
Humans do not perceive frequencies on a linear scale. We are better at detecting differences in lower frequencies than higher frequencies.
For example, we can easily tell the difference between 500 and 1000 Hz, but we will hardly be able to tell a difference between 10,000 and 10,500 Hz, even though the distance between the two pairs are the same.
In 1937, Stevens, Volkmann, and Newmann proposed a unit of pitch such that equal distances in pitch sounded equally distant to the listener. This is called the **mel scale**.



# From wav to melspectrograms

Using librosa library to convert wav to melspectrogram and then convert melspectrogram to image
(mapped the y-axis (frequency) onto the mel scale to form the melspectrogram)



clock tick

# From wav to melspectrograms - useful functions

```
def get_melspectrogram_db(file_path, sr=None, n_fft=2048, hop_length=512, n_mels=128, fmin=20, fmax=8300, top_db=80):
  wav,sr = librosa.load(file_path,sr=sr)
  if wav.shape[0]<5*sr:
    wav=np.pad(wav,int(np.ceil((5*sr-wav.shape[0])/2)),mode='reflect')
  else:
    wav=wav[:5*sr]
  spec=librosa.feature.melspectrogram(wav, sr=sr, n_fft=n_fft,
            hop_length=hop_length,n_mels=n_mels,fmin=fmin,fmax=fmax)
  spec_db=librosa.power_to_db(spec,top_db=top_db)
  return spec_db
```

# From wav to melspectrograms – useful functions

```
def spec_to_image(spec, eps=1e-6):
  mean = spec.mean()
  std = spec.std()
  spec_norm = (spec - mean) / (std + eps)
  spec_min, spec_max = spec_norm.min(), spec_norm.max()
  spec_scaled = 255 * (spec_norm - spec_min) / (spec_max - spec_min)
  spec_scaled = spec_scaled.astype(np.uint8)
  return spec_scaled
```

# Next steps for NN building

1. **Loading data in Pytorch** - build dataloaders to preprocess and load data
2. **Building Model** - use custom model
3. **Training** - CrossEntropyLoss and Adam optimizer are used. The model is trained for 20 epochs with stable learning rate

# Train a neural network on a GPU

**CUDA** (Compute Unified Device Architecture) is a parallel computing platform developed by NVIDIA for general computing on graphics processing units (GPUs). With CUDA, developers can accelerate computing applications by leveraging the power of GPUs.

Let's first define our GPU as the first visible cuda device.

```
if torch.cuda.is_available():
  device=torch.device('cuda')
else:
  device=torch.device('cpu')
```

# Train a neural network on a GPU

Send the network to the GPU:

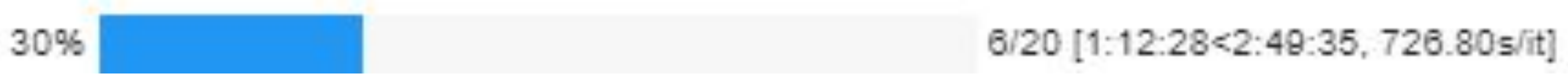
```
model = ESC50Model(input_shape=(1,128,431), batch_size=16, num_cats=50).to(device)
```

Send data at each step to the GPU:

```
x = x.to(device, dtype=torch.float32)
y = y.to(device, dtype=torch.long)
```

# Runtime comparison

CPU:




With the use of GPU the network training lasted about 12 minutes. The same training on a regular processor will last for 3 hours. The difference is significant, this is because our network is big. When using large arrays for training, the difference between the speed of a GPU and a traditional processor increases.

GPU:

# Final model results

```
Valid-Accuracy : 0.545
```

```
dog.wav                    [    <=>              ]   3.61M  4.89MB/s    in 0.7s

2021-12-20 03:48:42 (4.89 MB/s) - ‘dog.wav’ saved [3786304]

dog
0:00 / 0:21
```

```
cat.wav                    [   <=>               ] 873.05K  1.92MB/s    in 0.4s

2021-12-20 03:33:42 (1.92 MB/s) - ‘cat.wav’ saved [894000]

laughing
0:00 / 0:05
```

# Project results

- Learned how to use NN’s
- Learned about sound processing
- Learned about image classification with NN’s
- Learned how to use GPU to speed up model training

# Improvements

- Writing own parallelization scripts instead of using Pytorch distributed package
- Think about better NN’s model architecture