Urban Sound Classification

Sam Richmond

The problem

- Urban sound can be defined as noise produced by humans living in any city
  - Spans massive area
  - Constantly increasing
- Machines that can intelligently interpret urban data is getting more important
  - A cornerstone of this is detecting threats from raw unstructured urban sound

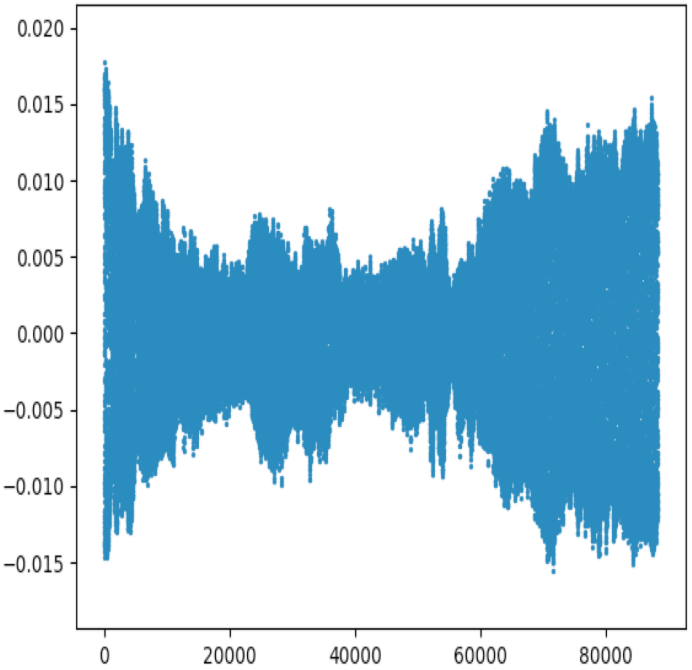
Solution

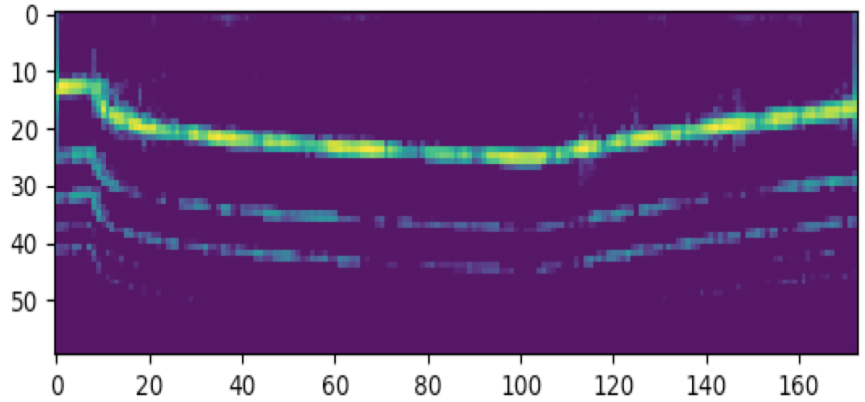
- UrbanSound8K dataset
  - https://urbansounddataset.weebly.com/urbansound8k.html
  - Contains ~8700 labeled ~4 second .wav sounds of 10 classes:
    - 0 = air_conditioner, 1 = car_horn, 2 = children_playing, 3 = dog_bark, 4 = drilling, 5 = engine_idling, 6 = gun_shot, 7 = jackhammer, 8 = siren, 9 = street_music
- Train neural net
  - RNNs and CNNs both commonly used
  - I chose to use a CNN to classify mel-spectrograms of the sounds (authors of above dataset used this approach with 0.79 total accuracy)

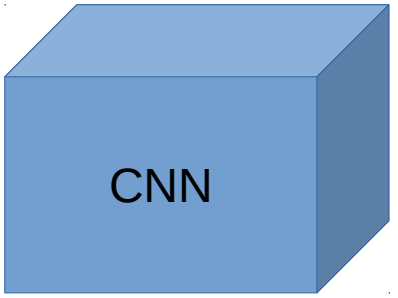
Method

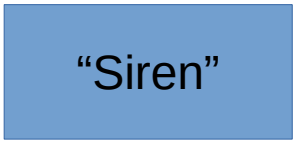
- Parsed through each .wav file in UrbanSound8K
- Used Librosa to create a mel-scaled spectrogram for each sound, cropped result by time to get a 128x128 resultant array (some sounds were less than 4 seconds)
- Saved feature array of (N, 128, 128, 1) and labels (N,) array to .npy files
- Loaded numpy files, shuffled and split into train/test,trained conv net model in Keras with:
  - Adam optimizer, learning rate = 0.0001
  - Batch size = 32, epochs = 30
  - Loss function = sparse categorical crossentropy

Method cont.


0.020
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
0
20000
40000
60000
80000


0
10
20
30
40
50
0
20
40
60
80
100
120
140
160


CNN

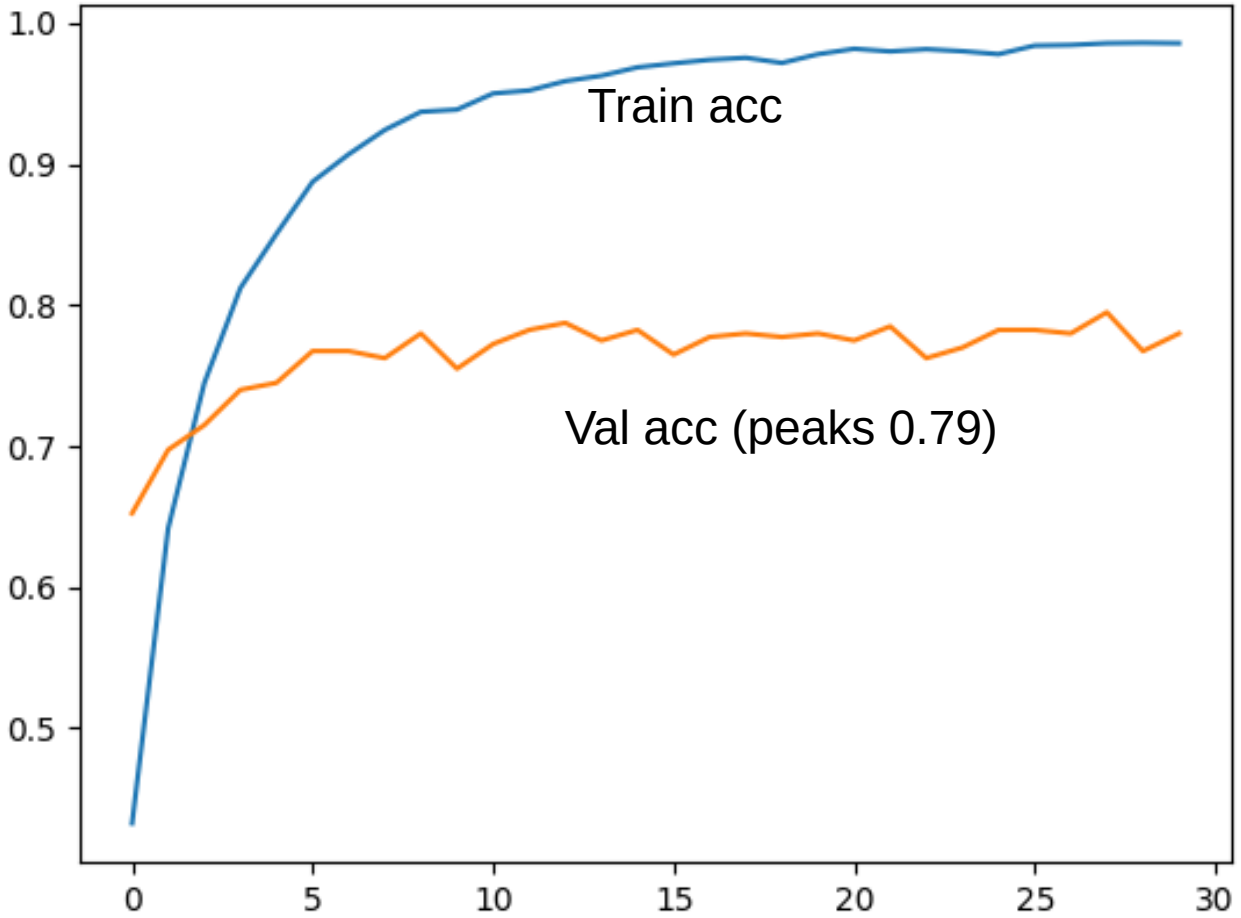

“Siren”

Method continued

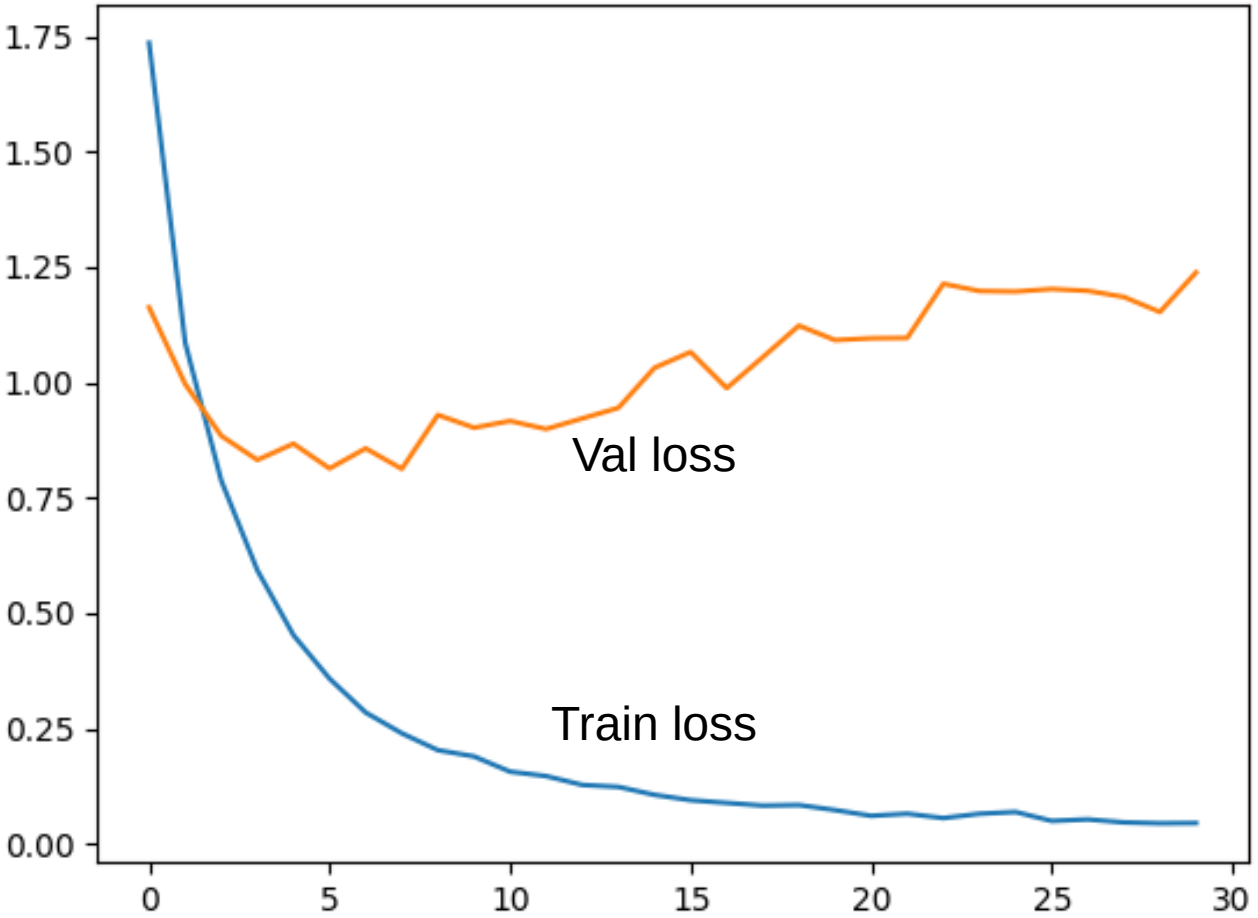
- Architecture:

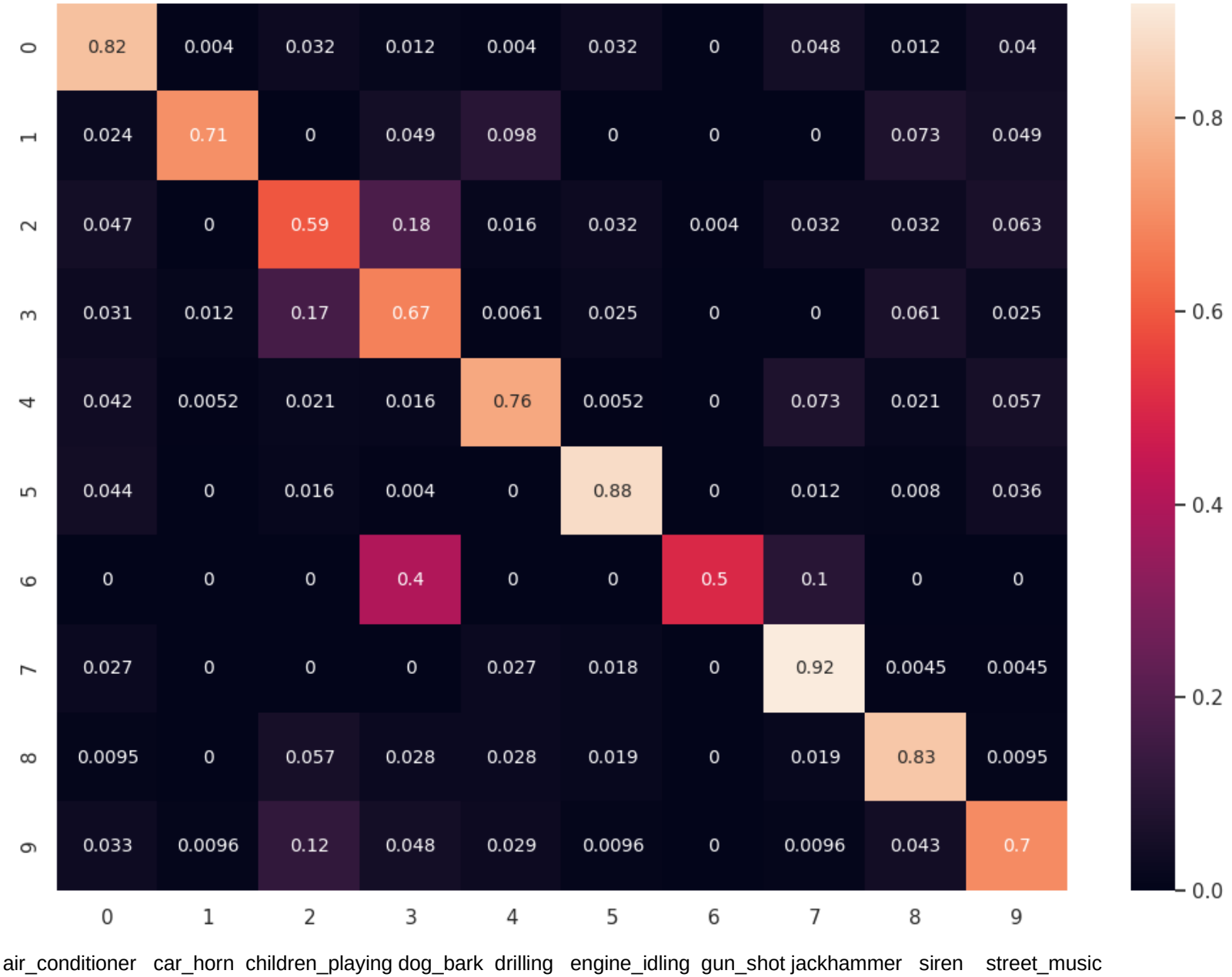
```
input_shape = (128, 128, 1)

model = Sequential()
model.add(Conv2D(64, kernel_size=(8, 8), strides=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

Results



Results


1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
25
30
Val loss
Train loss


0 0.82 0.004 0.032 0.012 0.004 0.032 0 0.048 0.012 0.04
1 0.024 0.71 0 0.049 0.098 0 0 0 0.073 0.049
2 0.047 0 0.59 0.18 0.016 0.032 0.004 0.032 0.032 0.063
3 0.031 0.012 0.17 0.67 0.0061 0.025 0 0 0.061 0.025
4 0.042 0.0052 0.021 0.016 0.76 0.0052 0 0.073 0.021 0.057
5 0.044 0 0.016 0.004 0 0.88 0 0.012 0.008 0.036
6 0 0 0 0.4 0 0 0.5 0.1 0 0
7 0.027 0 0 0 0.027 0.018 0 0.92 0.0045 0.0045
8 0.0095 0 0.057 0.028 0.028 0.019 0 0.019 0.83 0.0095
9 0.033 0.0096 0.12 0.048 0.029 0.0096 0 0.0096 0.043 0.7
0 1 2 3 4 5 6 7 8 9
air_conditioner car_horn children_playing dog_bark drilling engine_idling gun_shot jackhammer siren street_music
0.8
0.6
0.4
0.2
0.0

Conclusion

- Got total accuracy comparable to approaches from J. Salamon and J. P. Bello using CNNs
- I got lower accuracy for certain classes (ex. gunshots) due to my method of cropping the sound eliminating soundclips that were too short (mostly gunshots)
  - Future approaches would most likely take shorter cropped spectrograms from more sounds, or find a way to have variable sizes in the set