Research Paper

Team Members:
Parker Bray
Kai Sandstrom
Robert Kellems
Cole Metzger

Project Title:
Sound Recognition with Neural Networks

Abstract Research Question and Main Findings:

Our main research question was if we were able to design a neural network that was able to discriminate between different sounds. The goal of this project was to explore the problem space of environmental sound recognition/discrimination using neural networks by examining both our own work and existing implementations. By creating our own neural networks and comparing/contrasting them in both design and performance to other examples in the literature, we can provide some insight into how such networks should and shouldn't be designed. We chose to do our data on both binary and 10-way classification. For the binary classification we have discovered that we were able to use our neural network to detect such sounds that we provided and were able to distinctly pick out what sound was what with greater than random accuracy. Our neural network was able to distinguish the different sounds we gave it. For binary classification we discovered that our accuracy was around 90% on average, but this accuracy is somewhat misleading, since there is a high false positive rate compared to the true positive rate. For our first implementation of 10-way classification we also discovered that the accuracy was around 46% on average. The training accuracy still remained very high, but when it came to the testing data it dropped significantly. This could also be from overfitting our dataset. For our second implementation we achieved an accuracy of around 50%, a slight improvement.

Related Works:

Many related works that we found used other classification methods such as random forest and support vector machines, but while doing some more research we discovered others that use neural networks to detect sound using the UrbanSound8k data (1st link below). The main difference between our two networks is that they are using a novel deep convolutional neural network (CNN) while we are using a more traditional feedforward neural network. Their input layer consisted of eight convolutional layers and two fully connected layers. They watched patterns across frequencies and time using 3x1 and 1x5 convolutional filters. This gave them an achieved performance of 83.7% on UrbanSound8K.

Another AI research paper from November of 2016 (2nd link below) also built a convolutional neural network. Their input layer consisted of time frequency patches that were

obtained from mel spectrograms which they ran through 3 convolution layers, 2 pooling operations, and 2 dense layers. The results for their CNN gave a mean accuracy of 73% for a first trial. Their network achieved a mean accuracy of 79% on a second run when they augmented their dataset using different sound deformations such as time stretching, dynamic range compression, and background noise mixing.

- https://arxiv.org/pdf/1808.08405.pdf. (1)
- https://arxiv.org/pdf/1608.04363.pdf (2)

Implemented Techniques:

The dataset we have been using to both train and test our network is the UrbanSound8K dataset (https://urbansounddataset.weebly.com/urbansound8k.html), which contains over 8000 short clips of urban environmental sounds split into 10 classes. In order to decrease the amount of time necessary to train and test the network, we first took the metadata.csv file included with the UrbanSound8K dataset (containing information such as filename, folder, duration etc.) and appended a column containing array representations of the mel spectrogram (extracted using the Librosa library) for each sound in the dataset for quick access. In essence, we are sacrificing space (the CSV files are quite large) for time. Since the duration of our sound clips vary, we also had to standardize our data; this was done by first transposing each array and then using NumPy's mean function, which would create an array containing 128 features. Since we are going through each sound in our dataset, this is an O(n) operation, but repetitive processing can be avoided entirely by creating a new CSV containing these standardized spectrograms, which we have done.

Accordingly, the input layers for both the binary and 10-way classification networks have a size of 128, and use the ReLU activation function. From there, both networks contain a single hidden layer of size 256 (this size was mostly chosen arbitrarily based on examples we had seen) which uses the ReLU activation function. For our binary classification network, the output layer is of size 2 and uses the softmax activation function in order to determine whether the given sound is more likely of the class currently being looked for or not; the 10-way network's output layer is of size 10 and uses the softmax activation function to determine which of the 10 classes the given sound most likely belongs to. Also for the 10-way classification, we created an improved version that combined 10 instances of our binary network design, each one attempting to identify one of our 10 sound classes. We expand further on how these all performed below. To determine the loss values, the binary network uses binary cross-entropy and the 10-way network uses categorical cross-entropy. In accordance with the validation method strongly suggested by UrbanSound8K, we used 10-fold cross validation to achieve our results. This means that for each of the 10 folders in our dataset, we created a network, used the other 9 folders as training data and the remaining folder to test; we then calculated the average testing accuracy of each of these networks to get our total accuracy. Each network would go through 30 epochs in order to train.

Given that our networks were all basic feedforward networks containing a single hidden layer, it is clear that they are somewhat limited in their simplicity. We were perhaps also limited by the number of features taken from each sound clip, since we only took a representation of the mel spectrogram while other features exist which we could have included as well. We could have instead implemented a more complex convolutional neural network to analyse our spectral data, which is an approach many use in this problem space. As far as applicability is concerned, neural networks such as the ones we have created could be used to assist autonomous machinery in better perceiving/navigating its environment and to automate video content moderation based on audio, to give some examples.

How our solution models human thought processes:

We used standard feedforward neural networks to classify sounds. Neural networks by their nature operate in a manner somewhat similar to the human brain and its connection of biological neurons. The brain's neurons pass electrical signals to other neurons to form a large network of neurons, similarly to how the neurons in an artificial neural network pass numerical values to other neurons. However, this is not to say that our model closely resembles or is intended to resemble human/animal hearing processes at all, but that its basic structure is loosely based on real neurological structures.

Empirical Analysis of Algorithms:

As stated earlier, our binary classification networks were able to classify sounds in the testing set with an accuracy of about 90%. The level of accuracy would vary depending on the sound class that was currently being looked for, as shown in the output below:

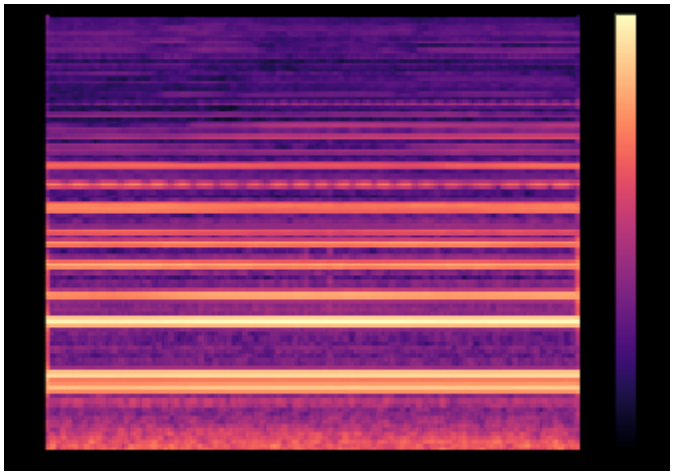
```
For each class: average acc, total tp, total tn, total fp, total fn:
air_conditioner:  0.8822843825962503 243 7469 263 757
car_horn:         0.9732768223187321 275 8220 83 154
children_playing: 0.8754904813640589 412 7234 498 588
dog_bark:         0.9055448762823632 585 7324 408 415
drilling:         0.8691407132213597 418 7172 560 582
engine_idling:    0.8953395033104903 347 7472 260 653
gun_shot:         0.9657493393426868 228 8208 150 146
jackhammer:       0.8925648629284577 309 7485 247 691
siren:            0.9324613168697752 546 7594 209 383
street_music:     0.8865544815877838 508 7230 502 492

Global average accuracy: 0.9078406779821957
Global total confusion value counts (tp, tn, fp, fn):
3871 75408 3180 4861
```

Given the average accuracy value of around 90%, the issue was raised that a binary classifier network that always returns "False" and never classifies inputs as the chosen sound class would also have an accuracy of around 90%, as only 10% of the sounds in the dataset are of each class. To test the issue raised by this criticism, we changed our binary classifier to return

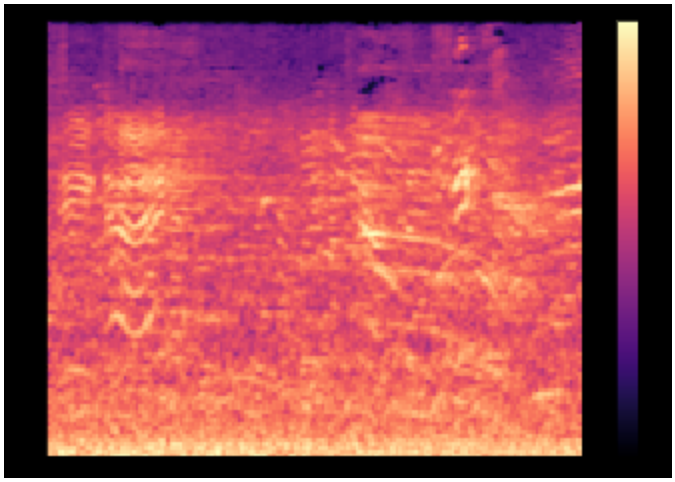
confusion matrix values for each run. The values for number of true positives, number of true negatives, number of false positives, and number of false negatives were used to calculate the accuracy for each fold used as testing data and for each sound. The totals are reported in the screenshot above. In this example run, dividing the number of true positives by the sum of true positives and false negatives reveals that, given that a sound corresponds to the class that a given binary classifier is trained to recognize, the sound is correctly classified as this sound approximately 44% of the time. While this value is much lower than the global average accuracy of around 90%, it is a huge improvement over random chance. When a sound does not belong to the class that the binary classifier is trained to identify, the network correctly identifies it approximately 96% of the time, also an improvement over random chance. In this example run, the network returned “False” approximately 92% of the time, which is only slightly greater than the expected distribution of 90%. Note that as can be found from the data in the screenshot above, the confusion matrix values vary between the different sound classes.

The differing levels of accuracy between the different sound classes make sense given the kinds of sounds being classified; accuracy for car horns is very high since car horns are generally very loud, easily recognisable sounds with clear harmonic content, whereas accuracy for sounds such as children playing and air conditioners is lower since they tend to be noisy and indistinct. Taking a look at their spectrograms can further elucidate this point.

Example of car horn from dataset:



Example of children playing from dataset:



The car horn contains very clear bands of harmonic information, while the sample of children playing is largely noisy, with some small recognizable curves in the spectrogram seemingly indicating the presence of a voice. Although we expected that the network would perform well with sounds such as the former, we were surprised to see that it performed with such a high level of accuracy with sounds such as the latter, especially considering that multiple classes contained such noisy data. This may indicate that for binary sound classification problems, relatively simple neural network designs such as ours (with some tweaks) may suffice.

For our original 10-way classification network, we received the following results:

```
Average accuracy: 0.4679172277450562
Average loss:     7.529865217208862 5
```

Clearly, this performance is much lower than that of the binary classification network, which is to be expected; the task has been made much more difficult by allowing for the choice between all 10 sound classes instead of between simply 1 sound class and the rest. However, it's worth pointing out that ~47% is still well above random (10%), so this network is not entirely without merit; with some further tweaking and/or introducing some ideas mentioned earlier, performance could certainly be improved. One can note the extremely large loss value; since performance on test data isn't great for this network, this may be partially blamed on overfitting. An earlier version of our binary classifier also reported loss values, which were abnormally high yet much lower than this value. This functionality was lost in the transition to confusion matrix values, but the code responsible for producing this output still exists as comments in the code.

In an attempt to improve the accuracy of our ten-way classifier, we also tried a different approach that uses ten binary classifiers, one for each sound class, instead of one ten-way classifier. To accomplish this, we saved the prediction percentages from each run of the binary classifier. After testing is complete, a loop iterates through the list of prediction values and determines which sound class's binary classifier gave the sample the highest probability of positive classification. This is taken to be the classifiers' prediction. Here are the results of the improved ten-way classifier using binary classifiers:

```
Accuracy for each sound class (accuracy, total correct, total incorrect):
air_conditioner:  0.319 319 681
car_horn:         0.668997668997669 287 142
children_playing: 0.566 566 434
dog_bark:         0.659 659 341
drilling:         0.446 446 554
engine_idling:    0.403 403 597
gun_shot:         0.6149732620320856 230 144
jackhammer:       0.308 308 692
siren:            0.6361679224973089 591 338
street_music:     0.579 579 421

Total   correct: 4388
Total incorrect: 4344
Accuracy: 0.5025194686211636
```

~50% is an improvement over ~47%, but only by a few points. This function also gives a breakdown of accuracies and total counts by sound class. We attempted to introduce this functionality to the original ten-way classifier but abandoned this idea due to time constraints, as it would have required a significant re-write of the function.

Outside Sources and Explanation for outside code:

We have utilized both of these articles in figuring out some of the details in utilizing this specific dataset as well as TensorFlow/Keras as a whole:
https://towardsdatascience.com/urban-sound-classification-part-1-99137c6335f9
https://towardsdatascience.com/urban-sound-classification-using-neural-networks-9b6fcd8a9150
We have also repeatedly referenced the Librosa documentation in order to extract data from the sounds in our dataset:
https://librosa.org/doc/latest/tutorial.html#overview

Expand and Improve:

In further developing this project, we will also likely continue to experiment with different ways of building and training feedforward neural networks, including different hidden layer sizes, number of hidden layers, batch sizes, and number of epochs in training. We may also engage in more secondary research to find more networks to compare to our own, both in approach and performance. Another thing to note is that this implementation of the improved ten-way classifier only uses the ten binary classifiers; there are no new layers and there is no new network. A potential way to further improve the ten-way classifier could be to use the array of classification probabilities that the binary classifiers return as inputs to an additional neural network, the idea being that the binary classifiers may mistake sounds for other sounds in a predictable way that the network can learn. We considered attempting to implement this approach, but were limited by time constraints. In this project, we have learned about many of the challenges of creating a functional neural network, particularly in regards to organizing and

utilizing large datasets. We have also learned that judging the success of one's machine learning implementation purely on accuracy can be misleading, and that other factors may be at play.